WhaTap

Kubernetes Monitoring

release date. 2026. 09. 11.

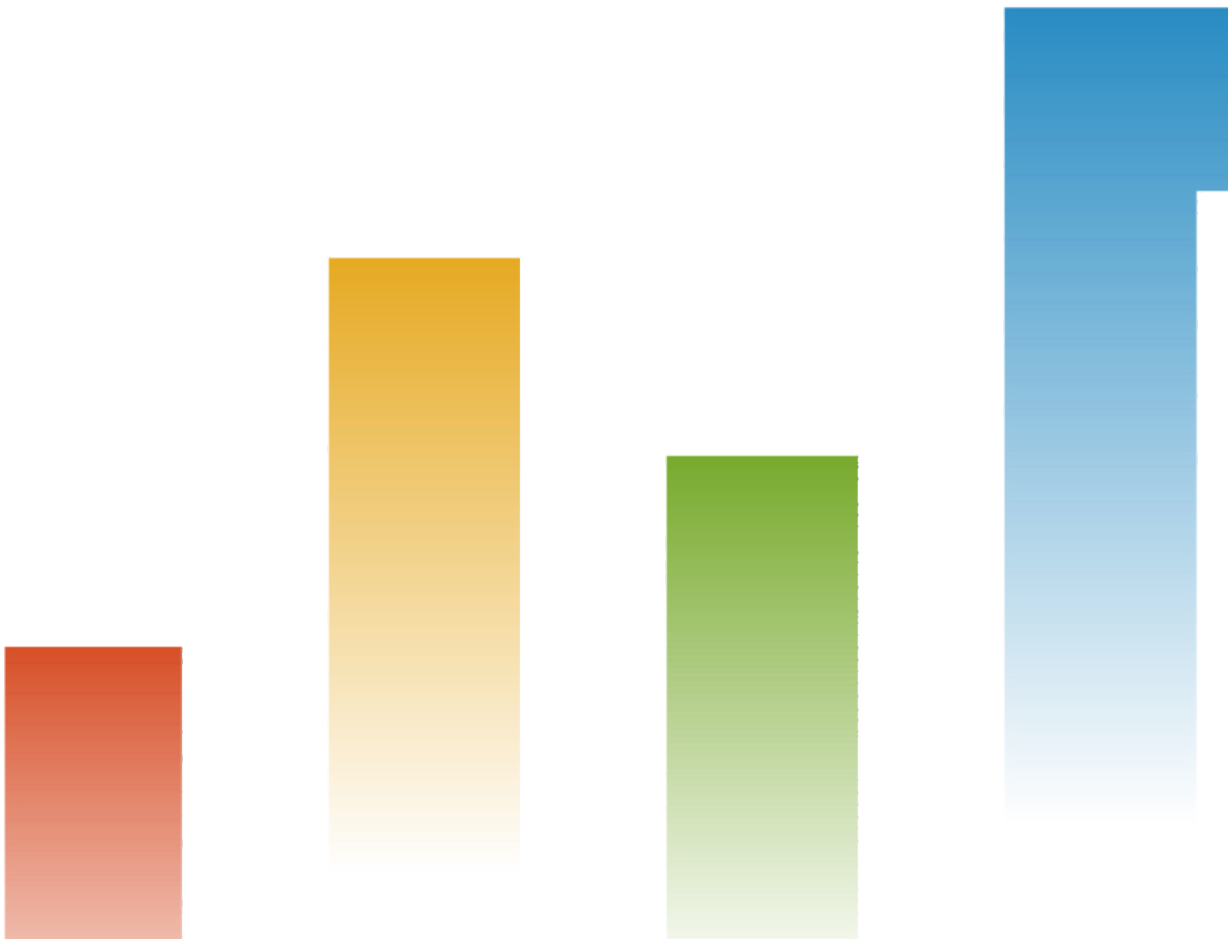


Table of Contents

- Kubernetesモニタリング 7
  - はじめに 10
  - 動作環境 13
  - インストール 19
    - WhaTap Operator 23
      - Whatap Operator インストール 26
      - APM自動インストール構成 39
      - オープンエージェント構成 50
      - WhaTap CR 設定オプション 74
    - マスターおよびノードエージェントのインストール 82
    - 管理 > アプリケーションエージェントのインストール 99
      - Docker Javaインストール 102
      - Docker Node.jsインストール 113
      - Docker Pythonインストール 117
      - Docker PHPインストール 123
      - Docker Goインストール 127
    - インストール問題の解決 138
  - 設定 144
  - 管理
    - ネームスペース管理 159
    - 連携プロジェクト管理 163
    - Kubernetesエージェント 166
    - Kubernetesエージェントリスト 168
    - エージェントのアップデート 172

- エージェント削除 173

◦ ダッシュボード

- コンテナマップ 176
  - コンテナビュー、Podビュー 190
  - 対象グループ化 194
  - コンテナマップの指標と状態 201
  - 外部プロジェクトの連携と設定 205
  - コンテナマップ詳細分析 213
    - Target Information 217
    - Trace 221
    - Metrics 224
    - Log 229
    - Event 234
    - Call Information 241
- ノードマップ 247
- 性能サマリー 262
- Flexボード 265
  - Flexボードの使用方法 268
  - ウィジェットの管理 279

◦ OpenMetrics

- テンプレート一覧 284
- OpenMetricsエクスプローラー 287

◦ クラスター

- クラスターサマリー 290
- ネームスペース状況 296

◦ ノード

- ノードリスト 298
- ノードタイムライン 310
- Node詳細情報 319
- ノードディスク一覧 321

◦ GPU 325

- GPUダッシュボード 330
- GPUトレンド 335
- GPUサービスグルーピング 337

◦ コンテナ

- コンテナリスト 343
- コンテナタイムライン 348
- コンテナボリューム 352
- コンテナイメージ 355
- Out Of Memory Killedコンテナ 357

◦ ワークロード

- Podリスト 360
- Pod タイムライン 366
- Pending Podの状況 371
- コンテナアプリケーション一覧 375
- Deploymentリスト 381

◦ サービスネットワーキング

- Service リスト 385
- Ingress一覧 390

◦ ストレージ

- PV/PVC リスト 394

◦ コントロールプレーン

- kube-apiserver ダッシュボード 402
- kube-apiserverメトリクス照会 411
- etcd ダッシュボード 420
- kube-scheduler ダッシュボード 427

◦ 分析

- Object Manifest情報 435
- クバネティスイベント 442
- Pod起動分析 446
- Pod起動分析 450
- アプリケーション ヒットマップ 455
- マルチサービスヒットマップ 457
- MSA分析 459
- アプリケーションスタック 467

◦ メトリクス 468

- Kubernetesメトリクス 471
- GPUメトリクス 522
- メトリクスチャート 525
- メトリクスチャート 548
- メトリクス異常検出 551
- メトリクス照会 554

◦ ログ 556

◦ レポート 561

◦ 統合レポート 564

◦ アプリケーション

- アプリケーションサービスダッシュボード 573
- ヒットマップ - トレース分析 596

- 日付別のアプリケーション状況 622
- トランザクション検索 624
- インスタンス性能モニタリング
  - Java 0
  - Node.js 0
  - Python 0
  - PHP 0
  - Go 0

◦ アラート通知 636

- イベント設定 0
  - メトリクス 643
  - 複合メトリクス 664
  - ヒットマップパターン 671
  - 異常値 676
  - リアルタイムログ 680
  - 複合ログ 692
  - トランザクション 706
- イベント設定 717
- 通知設定 721
  - イベント通知フォーマット 743
- イベント履歴 751
  - リアルタイムアラート 759
  - システムイベント 762

◦ 実験室 764

Kubernetesモニタリング

WhaTap Kubernetesモニタリングは、コンテナ、Pod、ノード、クラスター全体からコンテナ内で実行されるアプリケーションまで、1つの画面で統合的に分析できるモニタリングソリューションです。マルチクラウドとハイブリッドクラウド環境の両方をサポートし、インフラ担当者とアプリケーション担当者が同じデータを見ながら協業できます。

Kubernetesモニタリングの構造

Kubernetesの最小デプロイ単位はコンテナではなくPodであり、1つのPodには1つ以上のコンテナが含まれます。したがって、Kubernetesモニタリングはコンテナ視点とPod視点の両方が必要です。WhaTapは、コンテナとPodのメトリクスを別々のカテゴリで収集し、**コンテナマップ**でコンテナビューとPodビューを同時に提供します。

カテゴリは、ユーザープロジェクトから収集したメトリクスを区別する名前で、各カテゴリは異なるTagとFieldで構成されます。

主な特長

- **統合モニタリング** — マルチクラウドとハイブリッドクラウド環境のKubernetesシステムを1つの画面でモニタリングします。
- **MSA呼び出し関係の分析** — Pod単位の分散環境でトランザクションのAPI呼び出し関係を追跡します。
- **Root Cause Analysis** — Host、コンテナ、アプリケーションを同時にモニタリングし、障害の原因を迅速に分析できます。
- **コンテナ内アプリケーションの分析** — Java、Node.js、Pythonアプリケーションのヒットマップ、トレース、マルチトランザクション追跡をサポートします。
- **コンテナマップによる統合可視化** — Kubernetesリソース、メトリクス、イベント、ログを1つの画面で提供します。インフラとアプリケーション担当者が同じビューを見ながら協業できます。
- **リソース使用量の推移分析** — コンテナとNodeの使用量推移ダッシュボードを提供します。WhaTapプロジェクトに属するコンテナ/Nodeの全体一覧から、個別対象の詳細まで確認できます。
- **メタ情報の即時参照** — マスターノードに接続しなくても、コンテナマップでメタ情報を直接確認できます。別途のマスターメタダッシュボードで全構成要素のメタ情報を一目で確認できます。
- **ログ分析への集中支援** — コンテナのログファイルと標準出力をリアルタイムで保存し、分類とインデックス作成を

自動化します。コンテナに直接接続しなくても、意味のあるログに集中できます。

次のチュートリアル動画でWhaTap Kubernetesモニタリングを体験してみてください。

モニタリング対象

Kubernetesモニタリングの対象は、Kubernetesクラスターとその上で実行されるアプリケーションです。

- **Kubernetesクラスター**
  - コンテナのCPU、Memory、Disk(Volume)、Network
  - Namespace、Deployment、ReplicaSet、Podなどのクラスター構成要素
  - Node(ホスト)のCPU、Memory、Disk、Network
- **コンテナ内アプリケーション** — Java、Node.js、Python、PHP、Go

次に説明するモニタリング手順は、クラスターモニタリングとアプリケーションモニタリングに分けて説明します。

モニタリング手順

クラスターモニタリング手順

1. WhaTapモニタリングサービスで**クラスタープロジェクト**を作成してください。最初に作成したKubernetesプロジェクトが自動的にクラスタープロジェクトになります。詳細については、はじめにを参照してください。
2. Kubernetes環境にマスターエージェント(`whatap-master-agent`)とノードエージェント(`whatap-node-agent`)をインストールしてください。両エージェントはユーザーのクラスター内でPod形式で実行されます。詳細については、エージェントのインストールを参照してください。
3. (選択) クラスター内にネームスペースが多い場合、ネームスペース単位でプロジェクトを分離できます。詳細については、ネームスペースプロジェクトの作成を参照してください。

> Kubernetes Namespaceは、単一クラスター内のリソースグループを分離するためのメカニズムです。クラスターのNamespace一覧は `kubectl get namespace` コマンドで確認できます。

アプリケーションモニタリング手順

コンテナ内で実行されるアプリケーションをモニタリングするには、アプリケーションエージェントを追加でインストールします。対応言語はJava、Node.js、Python、PHP、Goです。詳細については、アプリケーションエージェントのインストールを参照してください。

はじめに

WhaTapモニタリングは次の手順で開始します。

ステップ1. 会員登録
ステップ2. 製品選択
ステップ3. プロジェクト作成
ステップ4. アクセスキー確認

その後、サポート環境を確認してからエージェントのインストール手順に進みます。

会員登録

WhaTapモニタリングサービスを利用するには、まず会員登録が必要です。 会員登録に関する詳細は、アカウント管理 > 会員登録ドキュメントを参照してください。

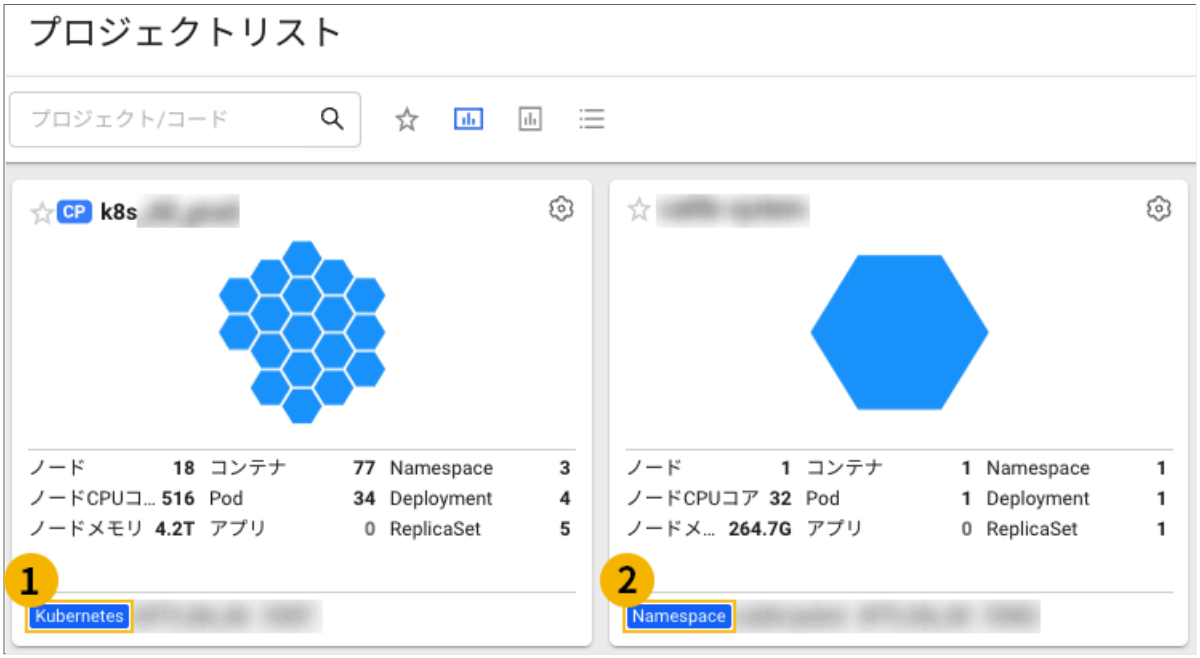
製品選択

1. WhaTapモニタリングサービスにログインします。
2. **プロジェクト全体** > ＋ **Project**ボタンをクリックします。
3. **商品選択**画面でモニタリングするGo製品を選択してから作成ボタンをクリックしてください。

プロジェクト作成

Kubernetesプロジェクト構造の理解

Kubernetesモニタリングを開始するには、まず**クラスタープロジェクト**を作成する必要があります。クラスター内にネームスペースが多く、コンテナ/Pod数が多い場合は、**ネームスペースプロジェクト**を追加で作成し、モニタリング範囲を分割できます。



1 クラスタープロジェクト

- Kubernetesクラスター全体のリソースをモニタリングするプロジェクトです。
- マスターエージェントとノードエージェントは**クラスタープロジェクト**にインストールします。
- プロジェクト一覧では、**クラスタープロジェクト**は名前の横に `CP` タグ、下部に `Kubernetes` タグが表示されます。
- 詳細な作成方法については、はじめにドキュメントを参照してください。

2 ネームスペースプロジェクト

- Kubernetesクラスターのネームスペース単位で、リソースとアプリケーションコンテナをモニタリングするプロジェクトです。
- プロジェクト一覧では、**ネームスペースプロジェクト**は下部に `Namespace` タグが表示されます。
- 詳細な管理方法については、ネームスペースプロジェクトの管理ドキュメントを参照してください。

Kubernetesプロジェクトの作成

モニタリングする製品を選択したら、エージェントをインストールする前にまずプロジェクトを作成します。

1. **プロジェクト作成**画面でプロジェクトを作成するには、次の項目を入力または選択してください。
   - **プロジェクト名**(必須): プロジェクト名を入力します。
   - **データサーバーリージョン**(必須): データサーバーが位置するリージョン(地域)を選択します。リージョンはクラウドサービスを提供するデータセンターのグループです。選択したリージョンにユーザーのデータが保存されます。
   - **プロジェクトタイムゾーン**(必須): アラートおよびレポート生成時の基準となる時間を設定します。
   - **通知言語の設定**(必須): プロジェクトで発生する警告通知メッセージの言語を設定します(韓国語、英語対応)。
   - **プロジェクトグループ**: 複数のプロジェクトをグループにまとめて管理できます。所属するグループがあれば選択してください。
   - **プロジェクト説明**: プロジェクトに関する追加説明を入力します。
2. すべての設定を完了したら**プロジェクトを作成する**ボタンをクリックしてください。

> ⓘ 組織を選択した状態でプロジェクトを追加する場合、**組織下位のグループ**を必須で設定する必要があります。 グループに関する詳細は、次のドキュメントを参照してください。

アクセスキー確認

アクセスキーは、WhaTapサービスを有効化するための固有IDです。

インストールガイドセクションで**プロジェクトアクセスキーの発行**ボタンを選択してください。**アクセスキー**を自動的に発行した後、次のステップに進みます。

> ⓘ **アクセスキー**は他のプロジェクトに重複使用できず、**ネームスペースプロジェクト**にも別途**アクセスキー**が必要です。

> ⓘ プロジェクトを作成すると、自動的に**エージェントインストール**ページに移動します。**エージェントインストール**ページに移動しない場合は、画面左側のメニューから**管理** > **エージェントインストール**を選択してください。

動作環境

Kubernetesモニタリングを開始する前に、次の対応事項を確認してください。WhaTap Kubernetesは、標準のKubernetes 1.16以降のバージョンおよび対応する環境にインストールできます。

対応プラットフォーム

プラットフォーム	サポート
Amazon EKS (Elastic Kubernetes Service)	✅
AKS (Azure Kubernetes Service)	✅
GKE (Google Kubernetes Engine)	✅
OKE (Oracle Container Engine for Kubernetes)	✅
Naver Cloud Kubernetes Service	✅
NHN Kubernetes Service (NKS)	✅
Red Hat OpenShift (3.x / 4.x)	✅
Cocktail	✅
Accordion	✅
PaaS-TA	✅
Kakao i Cloud	✅
Rancher	✅

ⓘ WhaTap は、CNCF（Cloud Native Computing Foundation）の認証を受けた標準 Kubernetes（Standard

> Kubernetes）環境であれば、どこにでもインストールできます。

NVIDIA GPU

WhaTap Kubernetesノードエージェントは、NVIDIAのデータセンターGPUマネージャー（DCGM）Exporterを通じてGPUメトリクスを収集します。収集構成にはサイドカー（Sidecar）パターンを使用します。

- **サイドカーパターン**

  DCGM Exporterは、メインアプリケーションコンテナと同じPod内で実行される補助コンテナとして構成されます。このサイドカーパターンにより、DCGM ExporterがGPUの状態情報を効率的に収集できます。

- **DCGM Exporter**

  `dcgm-exporter` コンテナは、DCGMを通じてGPUの状態と性能関連メトリクスを収集します。

- **メトリクス収集と転送**

  `whatap-node-agent` コンテナは、 `dcgm-exporter` のHTTPエンドポイントを通じてGPUメトリクスを要求し収集します。

  > `dcgm-exporter` のHTTPエンドポイントは通常ポート9400を使用します。

- **MIG環境のサポート**

  WhaTap Kubernetesは、NVIDIA GPUクラスター環境で**MIG（Multi-Instance GPU）**機能をサポートし、**物理GPU単位だけでなくMIGインスタンスまで細分化されたモニタリング**を提供します。

  MIGモードを使用すると、1つの物理GPUを複数の論理インスタンスに分割してワークロードを分離し、リソースを割り当てることができます。WhaTapはこのような環境でも正確なメトリクス収集と可視化をサポートします。

  > GPUモニタリングを利用するには、この機能を有効にする必要があります。 詳細については、GPUモニタリングのインストールを参照してください。

収集されるGPUメトリクスの一覧は、GPUメトリクスドキュメントを参照してください。

対応ブラウザ

WhaTapモニタリングサービスは、ウェブブラウザとモバイルアプリで利用できます。

Webブラウザ	推奨	対応バージョン
Google Chrome	O	111以降
Mozilla FireFox	X	最新バージョン
Edge	X	最新バージョン
Safari	X	最新バージョン

サポート案内

- ブラウザの互換性および性能の観点から、Chrome の最新バージョンの使用を推奨します。
- ユーザーインターフェース（User Interface, UI）は HTML5 標準技術で実装されているため、Internet Explorer はサポートしていません。

制約事項

WhaTap の Web インターフェースはモバイルブラウザをサポートしていません。モバイル端末から WhaTap にアクセスするには、Android または iOS 用の WhaTap アプリをインストールしてください。WhaTap モバイルアプリはモバイル環境に最適化されています。WhaTap モバイルアプリの詳細については、こちらのドキュメント を参照してください。

サービス提供形態

WhaTapモニタリングは、SaaSまたはオンプレミス（インストール型）方式でご利用いただけます。どちらを選択しても、同じエージェントで同じモニタリング機能をご利用いただけます。

Deployment	Support	Description
SaaS	✅	WhaTapが運用する収集サーバーに接続。リージョン別アドレスはファイアウォール項目を参照
オンプレミス（インストール型）	✅	収集サーバーを自社で構築して運用（閉域網環境に対応）。導入範囲は営業担当者と協議

> ⓘ インターネットに接続しない閉域網環境でも、オンプレミスで構築できます。

提供形態による機能の違い

どの形態を選択しても、モニタリング機能は同じです。モバイルアプリのみが例外で、service.whatap.io SaaSでのみご利用いただけます。

> ⓘ オンプレミス環境で一部の機能をご利用いただくには、別途契約が必要です。

ファイアウォール

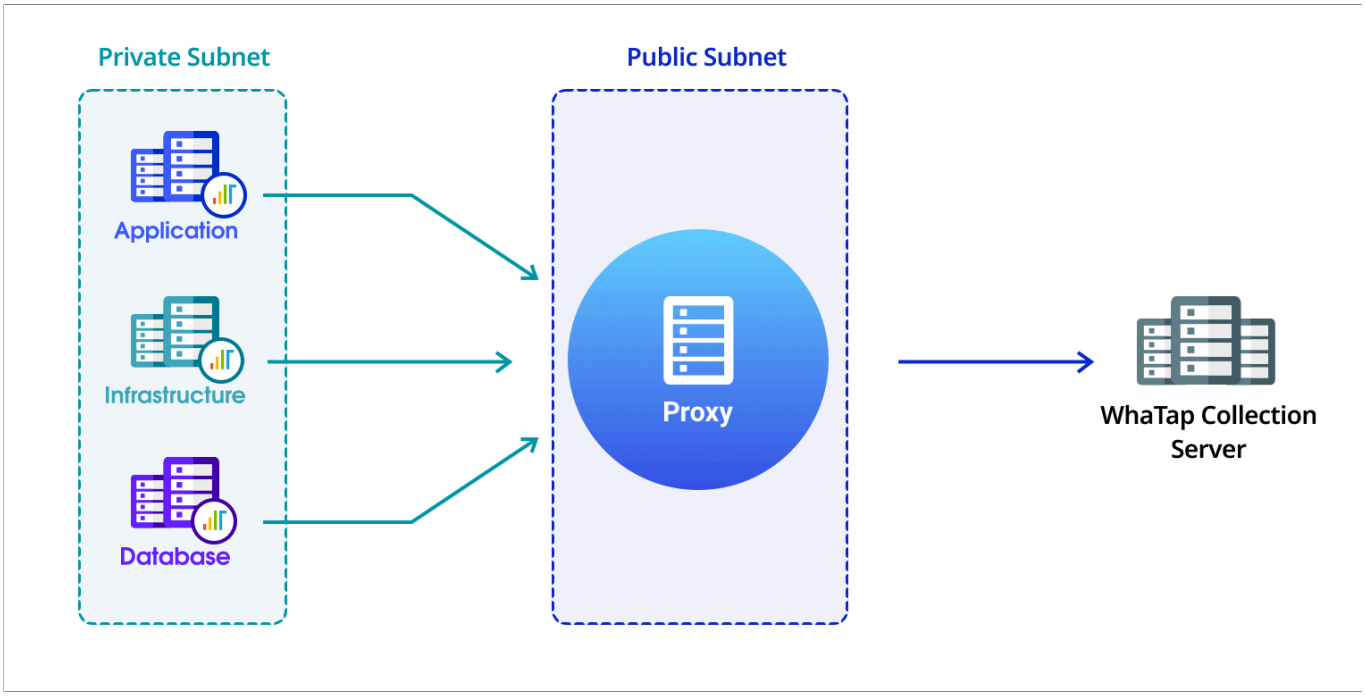
WhaTap エージェントは、収集サーバーの **TCP 6600** ポートにアクセスできる必要があります。監視対象に最も近い収集サーバーのアドレスを許可してください。

出発地：WhaTapエージェント

クラウド	目的地	目的地IP	ポート番号
AWS	WhaTapソウル収集サーバー	13.124.11.223 / 13.209.172.35	TCP 6600
	WhaTap東京収集サーバー	52.68.36.166 / 52.193.60.176	TCP 6600
	WhaTapシンガポール収集サーバー	18.138.0.93 / 18.139.67.236	TCP 6600
	WhaTapジャカルタ収集サーバー	108.136.91.69 / 108.137.158.44	TCP 6600

クラウド	目的地	目的地IP	ポート番号
	WhaTapタイ収集サーバー	43.209.109.219 / 43.209.48.86	TCP 6600
	WhaTapムンバイ収集サーバー	13.127.125.69 / 13.235.15.118	TCP 6600
	WhaTapカリフォルニア収集サーバー	52.8.223.130 / 52.8.239.99	TCP 6600
	WhaTapバージニア収集サーバー	107.23.220.101 / 54.236.221.105	TCP 6600
	WhaTapフランクフルト収集サーバー	3.125.142.162 / 3.127.76.140	TCP 6600
Azure	WhaTapソウル収集サーバー	52.231.66.38 / 20.194.5.115	TCP 6600
	WhaTap東京収集サーバー	52.246.169.54 / 20.210.27.232	TCP 6600
Kakao	WhaTapソウル収集サーバー	61.109.237.237 / 61.109.238.166	TCP 6600

エージェントから収集サーバーに直接接続できない場合は、提供するProxyモジュールを使用して経由します。



モバイルアプリ

WhaTap モバイルアプリは、Android および iOS 環境をサポートしています。リンクへ移動するか、QR コードをスキャンして WhaTap アプリをインストールできます。モバイルアプリの詳しい説明は、次の文書を参照してください。

iOS	Android
iOS 12バージョン以降	Android 5.0バージョン以降

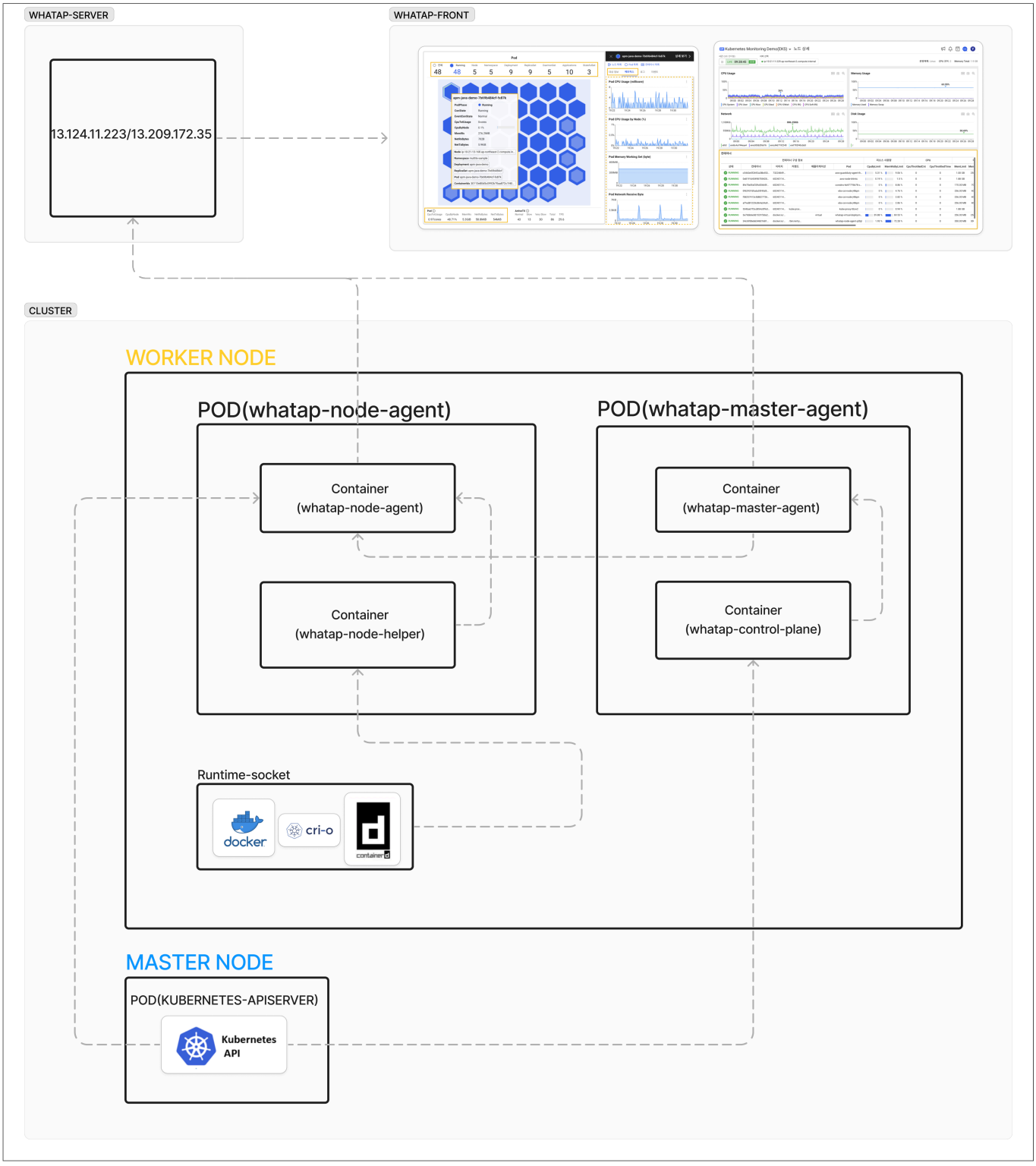
エージェントのインストール

このページでは、WhaTap Kubernetesモニタリングエージェントの構造と2種類のインストール方式を紹介します。事前準備とアーキテクチャを確認したうえで、運用環境に合うインストール方式を選択してください。

> エージェントをインストールする前に、はじめにドキュメントで次の手順を完了してください。
>
> - WhaTapへの会員登録
> - **クラスタープロジェクト**の作成
> - アクセスキーの発行

Kubernetesエージェントのアーキテクチャ

WhaTap Kubernetesエージェントは、マスターエージェントのPodとノードエージェントのPodで構成されます。


WHATAP-SERVER
13.124.11.223/13.209.172.35
WHATAP-FRONT
CLUSTER
WORKER NODE
POD(whatap-node-agent)
Container
(whatap-node-agent)
Container
(whatap-node-helper)
POD(whatap-master-agent)
Container
(whatap-master-agent)
Container
(whatap-control-plane)
Runtime-socket
docker
cri-o
MASTER NODE
POD(KUBERNETES-APISERVER)
Kubernetes
API

ノードエージェントPod

`whatap-node-agent` Podは、DaemonSet形式でデプロイされ、ノードごとに1つのPodが実行されます。Pod内には次の2つのコンテナがあります。

- **whatap-node-agent**

  kube-apiserverと通信し、ノード上のワークロードに対するメトリクスを収集します。`kube_pod`、`container`、`kube_pod_volume`などのメトリクスに影響します。

- **whatap-node-helper**

  Runtime socketとルートボリュームをマウントし、ノードとコンテナのリソース使用量を収集します。

マスターエージェントPod

`whatap-master-agent` Podは、Deployment形式でデプロイされ、クラスターごとに1つのPodが実行されます。Podには次のコンテナが含まれます。

- **whatap-master-agent** (必須)

  クラスター全体のメトリクスを収集します。ネームスペースプロジェクト情報のように、すべてのノードエージェントが共有すべきデータを各ノードエージェントに伝達・同期する役割も担います。

- **whatap-control-plane** (オプション)

  Kubernetesコントロールプレーンノードのメトリクスを収集します。現在はapiserverの状態とリクエスト件数のメトリクスを収集します。コントロールプレーンモニタリングを使用するには、追加機能の適用が必要です。詳細については、コントロールプレーンダッシュボードドキュメントを参照してください。

> ⓘ 今後、複数のマスターエージェントPodが同時に動作できるよう、多重化機能を提供する予定です。

インストール方式の選択

WhaTap Kubernetesモニタリングは、次の2種類の方式でエージェントをインストールできます。運用環境と管理方式に合った方法を選択してください。

WhaTap Operatorでインストール

WhaTap Operatorは、単一のカスタムリソース(CR)でマスター・ノードエージェント、アプリケーションの自動計測、OpenAgentを統合管理するKubernetesオペレーターです。CRの状態を継続的に監視し、変更内容を自動で調整します。

次のページでインストールと構成方法を確認してください。

- WhaTap Operatorのインストール — Helmチャートでオペレーターをデプロイし、`WhatapAgent` CRを作成します。
- アプリケーションエージェントの自動注入 — CR設定だけでコンテナ内のAPMエージェントを自動注入します。
- Open AgentによるOpenMetrics収集 — 外部エクスポーターのOpenMetricsデータを収集します。
- WhatapAgent CR構成オプション — CRで使用できるすべての設定オプションをまとめています。

マスター・ノードエージェントを直接インストール

コンソールの**インストール案内**ページに従って、YAMLまたはHelmチャートでマスター・ノードエージェントを直接デプロイする方式です。コンテナランタイム、Kubernetesバージョン、GKE/OpenShiftなどの環境別オプションを段階的に選択できます。

- マスター・ノードエージェントのインストール

アプリケーションエージェントの追加インストール

上記いずれの方式を選択しても、コンテナ内で実行されるJava/Node.js/Python/PHP/Goアプリケーションをモニタリングするには、アプリケーションエージェントを追加でインストールする必要があります。

- アプリケーションエージェントのインストール

インストール動画を見る

Whatap Operator

WhaTap Operatorは、Kubernetes環境でWhaTapエージェントをデプロイおよび構成できるオープンソースのKubernetesオペレーターです。

オペレーターを活用することで、単一のカスタムリソース（CRD）を通じて、K8sおよびGPU（MIG）モニタリングから自動計測、OpenMetrics収集に至るまで、すべての機能を統合的に管理できます。 このプロセスにおいて、オペレーターはデプロイされるリソースの整合性を自動的に検証し、複雑な構成で発生する可能性のあるエラーを最小限に抑え、安定的な運用をサポートします。

主な特徴

- **カスタムリソースを通じたデプロイ**: WhaTapエージェントおよび関連コンポーネントをKubernetesカスタムリソースを用いてデプロイおよび管理します。
- **簡素化されたデプロイ構成**: WhaTapエージェントのバージョン、リソース要求／制限、モニタリング対象の登録などの必須要素を、ネイティブKubernetesリソースで簡単に設定できます。
- **自動APM計測**: Kubernetes標準のラベルセレクターを利用し、特定のPodにAPMエージェントを自動的にインジェクトします。
- **Open Agentを活用したOpenMetrics収集**: Open Agentをインストールすることで、OpenMetrics（OpenMetrics）データを収集・活用できます。
- **統合モニタリング管理**: 単一のCRを通じてアプリケーション性能モニタリング（APM）とKubernetesインフラモニタリングを同時に構成・管理し、運用効率を高めます。

**WhaTap Operatorの目的 **

WhaTap Operatorは、Kubernetes環境におけるモニタリング構成の複雑さを軽減し、管理を簡素化することに重点を置いています。

- **エージェントのインストールおよび構成の負担軽減**: WhaTapモニタリングエージェントのインストールと管理の手間を大幅に削減します。
- **Kubernetesネイティブリソースによる自動化**: KubernetesのCRDを活用し、WhaTapモニタリング対象に対する設定を自動的に管理します。

```
apiVersion: monitoring.whatap.com/v2alpha1
kind: WhatapAgent
metadata:
  name: whatap
spec:
  features:
    apm:
      instrumentation:
        targets:
          - name: "java-app"
            enabled: true
            language: "java"
            whatapApmVersions:
              java: "2.2.58"
            # セレクタ構成...
```

- **構成の抽象化とバリデーション**: 複雑なモニタリング構成を単純化し、標準的なKubernetesセレクター（matchLabels、matchExpressions）をサポートします。 また、構成の有効性を検証することでエラーを最小限に抑え、安定した運用を支援します。

なぜHelmチャートや手動のDaemonSetインストールではなく、WhaTap Operatorを使用するのでしょうか？

HelmチャートやDaemonSetを使用してWhaTapエージェントをインストールすることも可能ですが、WhaTap Operatorには以下のような利点があります。

1. 自動状態調整

HelmチャートやDaemonSetは状態変更時に手動での介入が必要ですが、WhaTap OperatorはKubernetesの**調整ループ（reconciliation loop）**に組み込まれており、CRの状態を継続的に監視し、自動的に調整します。 例えば、CRによって作成されたリソースが誤って削除されたり変更された場合でも、オペレーターがそれを検知し、CRに定義された状態に自動で復元します。

2. 構成エラーの最小化

手動インストール方式ではエラーが発生する可能性が高くなります。 WhaTap Operatorはエージェント構成に対する厳密なバリデーションを実施することで、こうしたエラーを最小限に抑えます。

3. 統合管理

K8sクラスタ内のWhaTapモニタリングコンポーネントの構成を1つのファイルで統合管理でき、運用効率が向上します。

4. Kubernetes標準の遵守

オペレーターはKubernetes APIのファーストクラスリソースとして扱われ、Kubernetes標準のラベルセレクターを完全にサポートします。 これにより、Kubernetesの慣れ親しんだパターンを用いて柔軟にモニタリング対象を選択できます。

Whatap Operator インストール

このガイドでは、Kubernetes 環境に WhaTap Operator をインストールし、`WhatapAgent` CR を設定して WhaTap モニタリングを開始する方法について説明します。

インストール前の確認事項

- Kubernetes クラスター（v1.19 以上）
- Helm 3.2 以上
- WhaTap アカウントおよびライセンスキー

> - WhaTapエージェントは**whatap-monitoring**ネームスペースにインストールされます。エージェントがインストールされるネームスペースは変更できません。
> - 収集サーバーへデータを送信するため、収集サーバーのアドレスとポートが開いている必要があります（TCP Outbound）。アドレスとポートはWhaTapモニタリングサービスの**管理 > エージェントインストール**画面で確認できます。

インストール方法

次の手順に従って WhaTap Operator をインストールし、`WhatapAgent` CR を作成します。

> WhaTapモニタリングサービスの**管理 > エージェントインストール > Kubernetes**画面でインストール方式に**Operator**を選択すると、以下の手順の実行コマンドと `WhatapAgent` CRファイル（whatap_operator.yaml）が自動生成されます。画面でイメージバージョン、コンテナランタイム、環境（GKE、OpenShift）、追加機能（GPUモニタリング、APM自動インストール、OpenMetrics収集）を選択すると、その設定が反映されたファイルをダウンロードできます。このドキュメントは、ファイルを自分で作成する場合やダウンロードしたファイルを修正する場合に参照してください。

1. **WhaTap Operator をインストールします。**

- 初めてエージェントをインストールするユーザーは、以下のコマンドを実行してください。

```
helm repo add whatap https://whatap.github.io/helm/
helm repo update
```

```
kubectl create ns whatap-monitoring
export WHATAP_HOST=<収集サーバーのIP>
export WHATAP_LICENSE=<WhaTapライセンス>
export WHATAP_PORT=<WhaTapポート>
kubectl create secret generic whatap-credentials --namespace whatap-monitoring --from-literal
WHATAP_LICENSE=$WHATAP_LICENSE --from-literal WHATAP_HOST=$WHATAP_HOST --from-literal
WHATAP_PORT=$WHATAP_PORT
helm install whatap-operator whatap/whatap-operator --namespace whatap-monitoring
```

高度な設定

イメージレジストリとプルシークレット

プライベートレジストリを使用する場合、またはイメージタグや pullPolicy を調整する必要がある場合は、values.yaml で次の値を設定します。

```
image:
repository: public.ecr.aws/whatap/whatap-operator
tag: latest
pullPolicy: Always

# プライベートレジストリを使用する場合の設定
imagePullSecret:
name: <your-registry-secret-name>
```

スケジューリング/優先度オプション

オペレーターPodのスケジューリング制約と優先度を制御できます。値を指定した場合にのみテンプレートへレンダリングされ、既存ユーザーとの互換性が保たれます。

```
priorityClassName: ""   # 例: system-cluster-critical
```

```
nodeName: ""        # 特定ノードへ固定スケジューリングが必要な場合
nodeSelector: {}    # 例: { kubernetes.io/os: linux }
affinity: {}        # 標準の K8s Affinity スキーマ
tolerations: []     # 標準の K8s Toleration リスト
```

- 例

```
priorityClassName: "high-priority"
nodeSelector:
 nodepool: system

affinity:
 nodeAffinity:
   requiredDuringSchedulingIgnoredDuringExecution:
   nodeSelectorTerms:
    - matchExpressions:
      - key: kubernetes.io/os
        operator: In
        values: ["linux"]

tolerations:
 - key: "node-role.kubernetes.io/control-plane"
   effect: NoSchedule
```

デフォルトの values.yaml ではなくカスタム設定で使用する場合は、次のコマンドでオペレーターをインストールします。

- すでにネームスペースがある場合

```
helm install whatap-operator whatap/whatap-operator -n whatap-monitoring -f
custom-values.yaml
```

- ネームスペースがない場合は自動作成してインストール

```
helm upgrade --install whatap-operator whatap/whatap-operator -n whatap-monitoring
```

```
--create-namespace -f custom-values.yaml
```

- 既存のWhaTap Kubernetesエージェントのユーザーは、次のコマンドを実行してエージェントをインストールします。

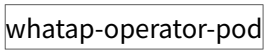
```
kubectl delete ns whatap-monitoring
kubectl delete clusterrole whatap
kubectl delete clusterrolebinding whatap
kubectl create ns whatap-monitoring
kubectl create secret generic whatap-credentials --namespace whatap-monitoring --from-literal
WHATAP_LICENSE=$WHATAP_LICENSE --from-literal WHATAP_HOST=$WHATAP_HOST --from-literal
WHATAP_PORT=$WHATAP_PORT
helm install whatap-operator whatap/whatap-operator --namespace whatap-monitoring
```

既存の yaml ファイルまたは helm 方式で WhaTap Kubernetes エージェントをインストールした場合、クリーンインストールが必要です。 既存の Kubernetes エージェントを削除した後、Operator の使用を推奨します。

- Operator は Deployment としてデプロイされます。 以下のコマンドで Operator が正常にデプロイされたかを確認します。

```
kubectl get pods -n whatap-monitoring | grep -i operator
```

- コマンドの実行結果として、 `whatap-operator` Pod が **Running** 状態である必要があります。



2. **WhatapAgent CR を作成します。**

- インストール画面でダウンロードしたwhatap_operator.yamlファイルを適用して `WhatapAgent` カスタムリソースを作成します。CRファイルを自分で作成した場合は、そのファイル名を使用します。

```
kubectl apply -f whatap_operator.yaml
```

- `WhatapAgent` CR は、WhaTap エージェントのデプロイと構成を定義します。 K8s エージェント、APM の自動インストール、Open Agent を使用した OpenMetrics の収集を設定できます。

最小構成

基本的な Kubernetes モニタリングのために、Whatap マスターエージェントとノードエージェントを有効にする最小構成です。

whatap_operator.yaml

```yaml
apiVersion: monitoring.whatap.com/v2alpha1
kind: WhatapAgent
metadata:
  name: whatap
  namespace: whatap-monitoring
spec:
  features:
    k8sAgent:
      masterAgent:
        enabled: true
      nodeAgent:
        enabled: true
        runtime: containerd                          # ノードのコンテナランタイム
        runtimeSocketPath: /var/run/containerd/containerd.sock  # コンテナランタイムのソケットパス
      gpuMonitoring:
        enabled: false
```

`runtime` と `runtimeSocketPath` は、ノードのコンテナランタイムに合わせて設定します。インストール画面でコンテナランタイムを選択すると、以下の値が自動的に反映されます。

Container runtime	`runtime`	`runtimeSocketPath`
containerd（既定値）	`containerd`	/var/run/containerd/containerd.sock
Docker Engine	`docker`	/var/run/docker.sock

Container runtime	runtime	runtimeSocketPath
CRI-O	crio	/var/run/crio/crio.sock

GKE環境では、 k8sAgent 配下に次の設定を追加します。

```
k8sAgent:
  gke:
    enabled: true
```

最小構成でインストールすると、以下のように whatap-master-agent , whatap-node-agent が追加でインストールされます。

whatap-master-agent, whatap-node-agent

> ⓘ オペレーターで WhaTap エージェントをインストールする場合、CR 設定（リソース制約、tolerations、imagePullSecret）の詳細は WhaTap CR 設定オプション を参照してください。

基本構成

```
apiVersion: monitoring.whatap.com/v2alpha1
kind: WhatapAgent
metadata:
  name: whatap
spec:
  features:
    ### APM 自動インストールを使用する場合はコメントを解除 - アプリケーション Pod に APM エージェントを自動で注入し、アプリケーションのパフォーマンスをモニタリングします。
    # apm:
    #   instrumentation:
    #     targets:
    #       - name: hello-world
    #         enabled: true
    #         language: "java"        # 対応言語:: java, python, nodejs
    #         whatapApmVersions:
    #           java: "2.2.58"        # (任意) 特定のAPMエージェントバージョンに固定する場合に指定します
```

```
  #       namespaceSelector:
  #         matchNames:
  #           - default          # モニタリング対象アプリケーションのあるネームスペース
  #       podSelector:
  #         matchLabels:
  #           app: "hello-world"  # モニタリング対象アプリケーション Pod のラベル
  #       config:
  #         mode: default        # APM エージェントモード設定

  ### K8s モニタリングを使用する場合はコメントを解除 - Kubernetes クラスター、ノード、コンテナのモニタリングを有効にします。
  # k8sAgent:
  #   masterAgent:
  #     enabled: true            # マスターエージェントを有効化（クラスター単位のメトリクス収集）
  #   nodeAgent:
  #     enabled: true            # ノードエージェントを有効化（ノードおよびコンテナ単位のメトリクス収集）
  #     runtime: containerd        # ノードのコンテナランタイム（containerd, docker, crio）
  #     runtimeSocketPath: /var/run/containerd/containerd.sock  # コンテナランタイムのソケットパス
  ### GPU モニタリング使用時はコメントを解除 - NVIDIA GPU メトリクス収集を有効にします。
  #   gpuMonitoring:
  #     enabled: true            # GPU モニタリング有効化（NVIDIA DCGM-EXPORTER が whatap-node-agent サイドカーとしてインストールされます）

  ### OpenMetrics（Prometheus 形式のメトリクス収集）使用時はコメントを解除
  # openAgent:
  #   enabled: true            # OpenAgent 有効化
  #   targets:
  #     - targetName: kube-apiserver
  #       type: ServiceMonitor     # 対象タイプ: ServiceMonitor, PodMonitor, StaticEndpoints
  #       namespaceSelector:
  #         matchNames:
  #           - "default"        # メトリクスを収集するネームスペース
  #       selector:
  #         matchLabels:
  #           component: apiserver # メトリクスを収集するサービス/Pod のラベル
  #           provider: kubernetes
  #       endpoints:
  #         - port: "https"      # メトリクスエンドポイントのポート
  #           path: "/metrics"     # メトリクスパス
  #           interval: "30s"     # スクレイピング間隔
  #           scheme: "https"     # HTTP スキーム（http または https）
```

```
#       tlsConfig:
#         insecureSkipVerify: true  # TLS 証明書検証をスキップ
  #     metricRelabelConfigs:
  #       - source_labels: ["__name__"]
  #         regex: "apiserver_request_total" # 収集対象のメトリクスフィルタ
  #         action: "keep"              # 一致するメトリクスのみ保持
```

> ⓘ Operatorインストール方式は、Istioサイドカー注入の設定をサポートしません。Istio環境の設定が必要な場合は、HelmまたはYAMLインストール方式を使用してください。kube-apiserverなどのコントロールプレーンのメトリクスは、OpenAgent構成で収集します。詳細はOpenAgent構成ドキュメントを参照してください。

様々な構成例

WhaTapオペレーターは、対応するKubernetesエージェントの他に、APM自動インストール、カスタムメトリックエージェントのインストールをサポートしています。 以下の構成例を参考にインストールできます。

GPU モニタリング

Kubernetesモニタリングエージェントをインストールせず、WhaTapノードエージェントPodにDCGM-EXPORTERコンテナをサイドカーとしてデプロイします。 ユーザーは特別な構成を行わずに、以下の YAML を適用するだけで、収集されるメトリクスは **GPU ダッシュボード** で確認できます。

> ⓘ ノードエージェントに内蔵されたサイドカーではなく、直接 DCGM-EXPORTER を構成した場合は、OpenAgent（オープンエージェント）構成 ドキュメントを参照して dcgm OpenMetricsを収集することができます。

GPUモニタリング自動インストール

GPUモニタリング自動インストール

```
apiVersion: monitoring.whatap.com/v2alpha1
```

```
kind: WhatapAgent
metadata:
 name: whatap
spec:
 features:
  openAgent:
   enabled: true  # Open Agent 有効化
  k8sAgent:
   masterAgent:
    enabled: true
   nodeAgent:
    enabled: true
   gpuMonitoring:
    enabled: true  # GPUモニタリングを有効にします
```

参考,自動インストール時、Open Agentは自動的にDCGM Exporterを検出し、スクレイピングを行います。

GPUノードのトレランス設定

GPUノードにテイントがある場合は、トレランスを追加する必要があります。

GPUノードToleration設定

```
apiVersion: monitoring.whatap.com/v2alpha1
kind: WhatapAgent
metadata:
 name: whatap
spec:
 features:
  k8sAgent:
   nodeAgent:
    enabled: true
    tolerations:
     - key: "nvidia.com/gpu"
       operator: "Exists"
       effect: "NoSchedule"
     - key: "gpu"
       operator: "Exists"
       effect: "NoSchedule"
   gpuMonitoring:
```

```
    enabled: true
```

APM自動インストール

Kubernetesモニタリングエージェントを有効にせず、Kubernetesクラスター内に存在するAPMを自動インストールします。

> ⓘ 詳細はAPM自動インストール構成のドキュメントを参照してください。

APM自動インストール

```
apiVersion: monitoring.whatap.com/v2alpha1
kind: WhatapAgent
metadata:
 name: whatap
spec:
 features:
  apm:
   instrumentation:
    targets:
     - name: hello-world
       enabled: true
       language: "java"
       whatapApmVersions:
        java: "2.2.58"
       namespaceSelector:
        matchNames:
         - default
       podSelector:
        matchLabels:
         app: "hello-world"
       config:
        mode: default
```

OpenAgent

KubernetesモニタリングやAPM計測を有効にせず、PrometheusスタイルのメトリクスをOpenAgent

コンポーネントのみを構成します。

> ⓘ 詳細はオープンエージェント構成のドキュメントを参照してください。

```
apiVersion: monitoring.whatap.com/v2alpha1
kind: WhatapAgent
metadata:
 name: whatap
spec:
 features:
  openAgent:
    enabled: true
    targets:
     - targetName: kube-apiserver
       type: ServiceMonitor
       namespaceSelector:
        matchNames:
         - "default"
       selector:
        matchLabels:
         component: apiserver
         provider: kubernetes
       endpoints:
       - port: "https"
         path: "/metrics"
         interval: "30s"
         scheme: "https"
         tlsConfig:
          insecureSkipVerify: true
         metricRelabelConfigs:
          - source_labels: ["__name__"]
            regex: "apiserver_request_total"
            action: "keep"
```

Kubernetesモニタリングと APM計測を併用

```
apiVersion: monitoring.whatap.com/v2alpha1
```

```yaml
kind: WhatapAgent
metadata:
 name: whatap
spec:
 features:
  apm:
   instrumentation:
    targets:
     - name: hello-world
       enabled: true
       language: "java"
       whatapApmVersions:
        java: "2.2.58"
       namespaceSelector:
        matchNames:
         - default
       podSelector:
        matchLabels:
         app: "hello-world"
       config:
        mode: default
  k8sAgent:
   masterAgent:
    enabled: true
   nodeAgent:
    enabled: true
  openAgent:
    enabled: true
    targets:
     - targetName: kube-apiserver
       type: ServiceMonitor
       namespaceSelector:
        matchNames:
         - "default"
       selector:
        matchLabels:
         component: apiserver
         provider: kubernetes
       endpoints:
       - port: "https"
         path: "/metrics"
```

```
    interval: "30s"
    scheme: "https"
    tlsConfig:
      insecureSkipVerify: true
    metricRelabelConfigs:
    - source_labels: ["__name__"]
      regex: "apiserver_request_total"
      action: "keep"
```

APM自動インストール構成

WhatapOperator をインストールした後、ユーザーは `WhatapAgent` CR（カスタムリソース）を定義することで、各 Kubernetes ポッドやネームスペースに APM を自動注入できます。

インストール前の要件

- Kubernetes 環境
- Helm 3.2 以上
- Whatap ライセンスキー（必須）
- Whatap収集サーバー情報（IPアドレスとポート）
- whatap-operator(オペレーターインストール ガイド)

CRの基本構成

> ⓘ このドキュメントのCR例は、whatap-operator Helmチャート **1.9.9** を基準に検証しています。チャートのバージョンによってCR構造が変わる場合があるため、インストールしたバージョンが異なる場合は `kubectl explain whatapagent.spec` で確認してください。

以下はCRを構成した基本サンプルです。 追加の設定が必要な場合 例. APM の設定方法を参照してください。

```
apiVersion: monitoring.whatap.com/v2alpha1
kind: WhatapAgent
metadata:
  name: default                    # リソース名
  namespace: whatap-monitoring          # インストール対象のネームスペース
spec:
  features:
    apm:
      instrumentation:
        targets:
```

```
    - name: "java-sample"          # APM 対象名（識別用）
      enabled: "true"              # 自動注入の有効化
      language: "java"             # 言語：java、python、nodejs など
      whatapApmVersions:
        java: "2.2.58"             # 該当言語の APM エージェントバージョン
      namespaceSelector:
        matchNames:
          - default                # APM 注入対象のネームスペース
      podSelector:
        matchLabels:
          app: hello-world         # APM 注入対象の Pod ラベル
      config:
        mode: default              # default または custom。custom は configMapRef が必要
      envs:
        WHATAP_JAVA_AGENT_PATH: /whatap-agent/whatap.agent.java.jar
        OKIND: hello
```

例題. APM設定方法

例題１. 特定のネームスペースのPodにAPMを自動注入

- `backend` ネームスペースに存在し、 `app=shop` ラベルを持つPodにJava APMを自動的に注入します。
- WhaTap APMは現在、 `latest` Javaバージョンのみをサポートしています。 (他の言語は今後サポート予定)
- 設定情報はConfigMapRefで渡します。 (configMap はアプリケーションと同じネームスペースに存在する必要があります。)

```
apiVersion: monitoring.whatap.com/v2alpha1
kind: WhatapAgent
metadata:
  name: whatap
  namespace: whatap-monitoring
spec:
  features:
    apm:
      instrumentation:
        targets:
          - name: shop-api
```

```
    enabled: "true"
    language: "java"
    whatapApmVersions:
     java: "latest"
    namespaceSelector:
     matchNames:
      - backend
    podSelector:
     matchLabels:
      app: shop
    config:
     mode: custom
     configMapRef:
      name: shop-api-config
```

shop-api-config.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
 name: shop-api-config
 namespace: backend
data:
 whatap.conf: |
  mtrace_enabled=true
  mtrace_basetime=0
  trace_normalize_enabled=true
```

例題２. 異なるConfigMapを使用して異なるワークロードをモニタリング

`system` ネームスペースの `app=router` Podと、 `backend` ネームスペースの `app=shop` Podに、それぞれ異なるConfigMapRef設定（ `request-router-config` 、 `shop-api-config` ）を注入します。

```
apiVersion: monitoring.whatap.com/v2alpha1
kind: WhatapAgent
metadata:
 name: whatap
 namespace: whatap-monitoring
spec:
```

```
features:
  apm:
    instrumentation:
      targets:
        - name: request-router
          enabled: "true"
          language: "java"
          whatapApmVersions:
            java: "latest"
          namespaceSelector:
            matchNames:
              - system
          podSelector:
            matchLabels:
              app: router
          config:
            mode: custom
            configMapRef:
              name: request-router-config
        - name: shop-api
          enabled: "true"
          language: "java"
          whatapApmVersions:
            java: "latest"
          namespaceSelector:
            matchNames:
              - backend
          podSelector:
            matchLabels:
              app: shop
          config:
            mode: custom
            configMapRef:
              name: shop-api-config
```

shop-api-config.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
```

```
  name: shop-api-config
  namespace: backend
data:
  whatap.conf: |
    mtrace_enabled=true
    mtrace_basetime=0
    trace_normalize_enabled=true
```

request-router-config.yaml

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: request-router-config
  namespace: backend
data:
  whatap.conf: |
    mtrace_enabled=false
    mtrace_basetime=100
    trace_normalize_enabled=true
```

フィールド

```
apiVersion: monitoring.whatap.com/v2alpha1
kind: WhatapAgent
metadata:
  name: whatap
spec:
  features:
    apm:
      instrumentation:
        targets:
          - name: <name>
            enabled: <true | false>
            language: <java | python | nodejs>
            whatapApmVersions:
              <language> : <version>
            namespaceSelector:
              matchNames:
```

```
    - <namespace>
  podSelector:
    matchLabels:
      <key>: <values>
  config:
    mode: default
```

フィールド名	説明
`name`	ターゲット名, モニタリングには影響せず、識別用です。
`enabled`	`"true"` の場合、該当ターゲットにAPMエージェントが自動注入されます。
`language`	ターゲットアプリケーションの言語。 `java`、`python`、`nodejs` など
`whatapApmVersions`	言語別のWhaTap APMエージェントのバージョンを指定
`namespaceSelector`	APM自動注入対象のネームスペース一覧
`podSelector`	APM自動注入対象のPodをラベルベースで選択
`config.mode`	`"default"`(既定値)または `"custom"` です。`"custom"` を指定する場合は `config.configMapRef.name` が必須です。

セレクタ（Selector）

WhatapAgentはKubernetes標準のラベルセレクタをサポートします。

namespaceSelector

フィールド名	説明
`matchNames`	対象のネームスペース名のリスト
`matchLabels`	キーと値のペアでネームスペースのラベルを選択（AND条件）

フィールド名	説明
matchExpressions	高度なラベル選択式（AND条件）

podSelector

フィールド名	説明
matchLabels	キーと値のペアでPodのラベルを選択（AND条件）
matchExpressions	高度なラベル選択式（AND条件）

matchExpressions演算子

オペレーター	説明
In	ラベルの値が指定された値のリストに含まれている必要があります。
NotIn	ラベルの値が指定された値のリストに含まれていない必要があります。
Exists	指定されたキーのラベルが存在している必要があります（values フィールドは空）
DoesNotExist	指定されたキーのラベルが存在していない必要があります（values フィールドは空）

K8sAgent 設定

フィールド名	説明
agentImageVersion	エージェントイメージのバージョン（デフォルト: "latest"）
masterAgent.enabled	マスターエージェントの有効化可否
masterAgent.resources	マスターエージェントのリソース要求および制限の設定
masterAgent.envs	マスターエージェントに追加する環境変数のリスト

フィールド名	説明
nodeAgent.enabled	ノードエージェントの有効化可否
nodeAgent.resources	ノードエージェントのリソース要求および制限の設定
nodeAgent.envs	ノードエージェントに追加する環境変数のリスト
gpuMonitoring.enabled	GPUモニタリングの有効化可否
apiserverMonitoring.enabled	API Serverモニタリングの有効化可否
etcdMonitoring.enabled	etcdモニタリングの有効化可否
schedulerMonitoring.enabled	schedulerモニタリングの有効化可否

```
# 複数条件を組み合わせたセレクターの例
namespaceSelector:
 matchNames:
  - production
  - staging
 matchLabels:
  environment: production
 matchExpressions:
  - {key: tier, operator: In, values: [frontend, backend]}
  - {key: restricted, operator: DoesNotExist}

podSelector:
 matchLabels:
  app: web-service
  version: v2
 matchExpressions:
  - {key: tier, operator: In, values: [backend, backend2, backend3]}
  - {key: environment, operator: NotIn, values: [dev, test]}
  - {key: ready, operator: Exists}
```

- namespaceSelector の詳細説明

  ネームスペースセレクターは、次の3つの方法で対象ネームスペースをフィルタリングします。

1. **matchNames**: ネームスペース名を直接指定して選択
    1. `production` または `staging` という名前のネームスペースのみを選択します。
    2. 名前ベースのフィルタリングは最も基本的な方法です。
2. **matchLabels**: キーと値のペアでラベル一致を確認（AND条件）
    1. `environment=production` ラベルを持つネームスペースのみを選択します。
    2. すべてのキーと値のペアが一致する必要があります（複数のラベルがある場合はすべてを満たす必要があります）。
3. **matchExpressions**: 複雑な条件を指定する高度な表現式（AND条件）
    1. `tier` ラベルが `frontend` または `backend` の値を持つ必要があります（ `In` 演算子）
    2. `restricted` ラベルが存在してはいけません（ `DoesNotExist` 演算子）
    3. すべての表現式が真である必要があります（AND条件）

- podSelector の詳細説明

  Podセレクターは、次の2つの方法で対象Podをフィルタリングします。

  1. **matchLabels**: キーと値のペアでラベル一致を確認（AND条件）
      1. `app=web-service` および `version=v2` のラベルをすべて持つPodのみを選択します。
      2. 指定されたすべてのラベルが正確に一致する必要があります。
  2. **matchExpressions**: 複雑な条件を指定する高度な表現式（AND条件）
      1. `tier` ラベルが `backend` 、 `backend2` 、 `backend3` のいずれかである必要があります（ `In` 演算子）
      2. `environment` ラベルが `dev` または `test` であってはならない（ `NotIn` 演算子）
      3. `ready` ラベルが存在している必要がある（ `Exists` 演算子）
      4. すべての表現式が真である必要があります（AND条件）

テスト

APM 自動インジェクションのテストが目的の場合は、以下の yaml を使用してインストールします。

```yaml title='test-values.yaml'
### テスト目的のサンプル Java アプリケーション
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-world
  labels:
    app: hello-world
spec:
  replicas: 1
  selector:
    matchLabels:
      app: hello-world
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
      - name: hello-world
        image: whatap/base_java:0.0.13  # またはユーザーアプリケーションイメージ
        ports:
        - containerPort: 8080
---

apiVersion: monitoring.whatap.com/v2alpha1
kind: WhatapAgent
metadata:
  name: default                         # リソース名
  namespace: whatap-monitoring          # インストール対象のネームスペース
spec:
  features:
    apm:
      instrumentation:
        targets:
          - name: "java-sample"           # APM 対象名（識別用）
            enabled: "true"               # 自動インジェクション有効化
            language: "java"              # 言語：java, python, nodejs など
            whatapApmVersions:
              java: "latest"              # 対応言語の APM エージェントバージョン
            namespaceSelector:
```

```
      matchNames:
        - default              # APM インジェクション対象のネームスペース
      podSelector:
        matchLabels:
          app: hello-world     # APM インジェクション対象の Pod ラベル
      config:
        mode: default          # default または custom。custom は configMapRef が必要
```

オープンエージェント構成

OpenAgentは、PrometheusエンドポイントからメトリクスをスクレイピングしてWhaTapサーバーに送信する役割を担います。 PrometheusエンドポイントからメトリクスをWhaTapサーバーに送信するGoベースのエージェントです。

設定

スタンドアロン YAML でインストールする場合

Open Agent はデフォルトで whatap-monitoring ネームスペースに作成されます。必要に応じて調整できます。

```yaml
# Whatap Open Agent 単体マニフェスト
# ネームスペースは必要に応じて調整してください。例では whatap-monitoring を使用しています。
---
#apiVersion: v1
#kind: Namespace
#metadata:
#  name: whatap-monitoring
---
# Whatap 接続情報（ライセンス／ホスト／ポート）
# すでに同じ名前の Secret を作成済みの場合、この Secret ブロックを削除し、
# 下の Deployment の secretKeyRef.name を既存の Secret 名に合わせてください。
apiVersion: v1
kind: Secret
metadata:
  name: whatap-open-credentials
  namespace: whatap-monitoring
type: Opaque
stringData:
  WHATAP_LICENSE: "<YOUR_LICENSE>"
  WHATAP_HOST: "<WHATAP_HOST>"
  WHATAP_PORT: "6600"

---
```

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: whatap-open-agent-sa
  namespace: whatap-monitoring

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: whatap-open-agent-role
rules:
  - apiGroups: ["*"]
    resources: ["pods", "services", "endpoints", "endpointslices", "namespaces", "secrets"]
    verbs: ["get", "list", "watch"]
  - nonResourceURLs: ["/metrics"]
    verbs: ["*"]
  - apiGroups: [""]
    resources: ["pods/exec"]
    verbs: ["create"]
  - apiGroups: [""]
    resources: ["configmaps"]
    verbs: ["*"]
  - nonResourceURLs: ["/metrics"]
    verbs: ["*"]

---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: whatap-open-agent-role-binding
subjects:
  - kind: ServiceAccount
    name: whatap-open-agent-sa
    namespace: whatap-monitoring
roleRef:
  kind: ClusterRole
  name: whatap-open-agent-role
  apiGroup: rbac.authorization.k8s.io
```

```
---
apiVersion: v1
kind: ConfigMap
metadata:
 name: whatap-open-agent-config
 namespace: whatap-monitoring
data:
 scrape_config.yaml: |
  features:
   openAgent:
    enabled: true
    globalInterval: "60s"
    globalPath: "/metrics"
    targets:
     - targetName: kube-apiserver
       type: ServiceMonitor
       namespaceSelector:
        matchNames:
         - "default" # API サーバーのサービスが存在するネームスペースに変更してください
       selector:
        matchLabels:
         component: apiserver
         provider: kubernetes
       endpoints:
        - port: "https"
          path: "/metrics"
          interval: "30s"
          scheme: "https"
          timeout: "30s"
          tlsConfig:
           insecureSkipVerify: true
          metricRelabelConfigs:
           - source_labels: [__name__]
             regex: "apiserver_request_total"
             action: keep
           - source_labels: [verb]
             target_label: http_verb
             replacement: "${1}"
```

```
            action: replace
          - target_label: metric_src
            replacement: "whatap-open-agent"
            action: replace

---
apiVersion: apps/v1
kind: Deployment
metadata:
 name: whatap-open-agent
 namespace: whatap-monitoring
 labels:
  name: whatap-open-agent
spec:
 replicas: 1
 selector:
  matchLabels:
   name: whatap-open-agent
 template:
  metadata:
   labels:
    name: whatap-open-agent
  spec:
   serviceAccountName: whatap-open-agent-sa
   volumes:
    - name: config-volume
      configMap:
       name: whatap-open-agent-config
    - name: logs-volume
      emptyDir: {}
   containers:
    - name: whatap-open-agent
      image: public.ecr.aws/whatap/open_agent:latest
      imagePullPolicy: Always
      resources:
       requests:
        memory: "300Mi"
        cpu: "100m"
       limits:
```

```
    memory: "1000Mi"
    cpu: "1000m"
  env:
   - name: WHATAP_LICENSE
     valueFrom:
      secretKeyRef:
       name: whatap-open-credentials
       key: WHATAP_LICENSE
   - name: WHATAP_HOST
     valueFrom:
      secretKeyRef:
       name: whatap-open-credentials
       key: WHATAP_HOST
   - name: WHATAP_PORT
     valueFrom:
      secretKeyRef:
       name: whatap-open-credentials
       key: WHATAP_PORT
   - name: debug
     value: "false"
  volumeMounts:
   - name: config-volume
     mountPath: /app/scrape_config.yaml
     subPath: scrape_config.yaml
   - name: logs-volume
     mountPath: /app/logs
  # 必要に応じてコンテナ／ノードのトレランスや affinity などを追加してください。
```

Helm でのインストール

1. 次のコマンドを実行して、WhaTap Helm リポジトリを追加してください。

```
helm repo add whatap https://whatap.github.io/helm/
helm repo update
```

2. 初めてエージェントをインストールする場合は、次のコマンドを実行してエージェントアプリケーションをインストールしてください。values.yaml は以下の例を参考にしてください。

- 初回インストール 既存ユーザー

```
# 初めてエージェントをインストールする場合は、次のコマンドを実行してエージェントアプリケーションをインストールしてください。
kubectl create ns whatap-monitoring
helm install whatap-kube-agent --namespace whatap-monitoring whatap/kube -f values.yaml
```

```
#既存の WhaTap Kubernetes エージェント利用者は、次のコマンドを実行してエージェントをインストールしてください。
#既存の YAML ファイル方式で WhaTap Kubernetes エージェントをインストールした場合、Clean Install が必要です。 既存の Kubernetes エージェントを削除した後、Helm チャートを使用してインストールしてください。.
kubectl delete ns whatap-monitoring
kubectl delete clusterrole whatap
kubectl delete clusterrolebinding whatap
kubectl create ns whatap-monitoring
helm install whatap-kube-agent --namespace whatap-monitoring whatap/kube -f values.yaml
```

values.yaml

```yaml
whatap:
  license: #ライセンスキーを入力
  host: #収集サーバーの IP を入力
  port: #収集サーバーのポートを入力
  createSecret: true #default : true, if credentials already exist, set false
  existingSecretName: "whatap-credentials" #default : whatap-credentials
collect_custom_resources: true
containerRuntime: "containerd"
deployment:
  containers:
    controlPlaneHelper:
      enabled: false
      image: "whatap/kube_mon:1.8.8"
      name: "whatap-control-plane-helper"
      port: 9496
      resources:
        limits:
          cpu: "200m"
```

```
        memory: "350Mi"
      requests:
       cpu: "100m"
       memory: "300Mi"
   masterAgent:
    image: "whatap/kube_mon:1.8.8"
    name: "whatap-master-agent"
    port: 6600
    resources:
     limits:
      cpu: "200m"
      memory: "350Mi"
     requests:
      cpu: "100m"
      memory: "300Mi"
  label: "whatap-master-agent"
  name: "whatap-master-agent"
  replicas: 1

daemonSet:
 containers:
  nodeAgent:
   image: "whatap/kube_mon:1.8.8"
   limits:
    cpu: "200m"
    memory: "350Mi"
   name: "whatap-node-agent"
   requests:
    cpu: "100m"
    memory: "300Mi"
  nodeHelper:
   image: "whatap/kube_mon:1.8.8"
   limits:
    cpu: "200m"
    memory: "350Mi"
   name: "whatap-node-helper"
   requests:
    cpu: "100m"
    memory: "100Mi"
```

```
 initContainers:
  nodeDebugger:
   enabled: false
   image: "whatap/kube_mon:1.8.8"
   name: "whatap-node-debug"
 label: "whatap-node-agent"
 name: "whatap-node-agent"

clusterrole:
 extraResources:
 - "*"

deploymentOpen:
 enabled: true
 containers:
  openAgent:
   image: "public.ecr.aws/whatap/open_agent:latest"
 config:
  features:
   openAgent:
    enabled: true
    #globalInterval: "60s"
    #globalPath: "/metrics"
    targets:
     - targetName: "kube-apiserver"
       type: "ServiceMonitor"
       enabled: true
       namespaceSelector:
        matchNames:
         - "default"  # クラスターのデフォルトラベルに合わせて調整してください
        matchLabels:
         component: "apiserver"
         provider: "kubernetes"
       endpoints:
        - port: "https"
          path: "/metrics"
          interval: "30s"
          scheme: "https"
          timeout: "30s"
```

```
        tlsConfig:
          insecureSkipVerify: true  # 自己署名証明書の場合の便宜用です。セキュリティが必要な場合は false にし、CA を指定してください
        metricRelabelConfigs:
         - source_labels: [__name__]
           regex: "apiserver_request_total"
           action: keep
         - source_labels: [verb]
           target_label: http_verb
           replacement: "${1}"
           action: replace
         - target_label: metric_src
           replacement: "whatap-open-agent"
           action: replace
```

WhatapOperatorをインストールした後、ユーザーは `WhatapAgent` CR（カスタムリソース）を通じてOpenMetricsの対象を設定します。

1. **PodMonitor**: Podラベルセレクターを利用した動的ディスカバリ（Prometheus OperatorのPodMonitorと類似）
2. **ServiceMonitor**: Serviceラベルセレクターを利用した動的ディスカバリ（Prometheus OperatorのServiceMonitorと類似）
3. **StaticEndpoints**: 固定のIPアドレスとポートを直接入力（Prometheusのstatic_configsと類似）

```
features:
 openAgent:
  enabled: true
  targets:
   # 1. PodMonitor: Podラベルセレクターを利用した動的ディスカバリー
   - targetName: my-app-pod-metrics
     type: PodMonitor
     namespaceSelector:
       matchNames:
        - "production"
     selector:
       matchLabels:
         app: my-app
```

```
  endpoints:
  - port: "web-metrics"  # Pod Specに定義されたPort名または実際のPort番号
    path: "/metrics"    # デフォルトは /metrics、必要に応じて再定義
    interval: "15s"     # デフォルトはグローバル設定、必要に応じて再定義
    scheme: "http"
    timeout: "10s"
    metricRelabelConfigs:  # スクレイピング後のメトリクス再ラベリング設定
      - source_labels: [__name__]
        regex: "http_requests_total"
        action: keep
      - source_labels: [method]
        target_label: http_method
        replacement: "${1}"
        action: replace

  # 2. ServiceMonitor: Serviceラベルセレクターを利用した動的ディスカバリー
  - targetName: my-service-metrics
    type: ServiceMonitor
    namespaceSelector:
      matchNames:
        - "default"
    selector:
      matchLabels:
        service: my-backend-service
    endpoints:
    - port: "http-metrics"  # Service Specに定義されたPort名または実際のTarget Port番号
      path: "/actuator/prometheus"
      interval: "30s"
      metricRelabelConfigs:
        - source_labels: [__name__]
          regex: "jvm_.*"
          action: keep
        - source_labels: [area]
          target_label: memory_area
          replacement: "${1}"
          action: replace

  # 3. StaticEndpoints: 固定されたIPアドレスとポートを直接入力
  - targetName: my-external-db-metrics
    type: StaticEndpoints
    # enabled: true  # ターゲットの有効化有無（既定値: true、省略可能）
```

```
  endpoints:
   - address: "192.168.1.100:9100"
     scheme: "http"
     path: "/metrics"
     interval: "60s"
     metricRelabelConfigs:
      - source_labels: [__name__]
        regex: "node_(cpu|memory).*"
        action: keep
      - source_labels: [instance]
        target_label: server
        replacement: "${1}"
        action: replace
```

PodMetricsおよびServiceMetricsの設定要素

- **targetName**: ターゲットの名前（ログおよび識別用）
- **type**: ターゲットの種類（"PodMetrics" または "ServiceMetrics"）
- **namespaceSelector**: スクレイピング対象のネームスペース
  - `matchNames`: 名前でネームスペースを選択
  - `matchLabels`: ラベルでネームスペースを選択
  - `matchExpressions`: 式でネームスペースを選択
- **selector**: スクレイピング対象のPodまたはServiceの選択
  - `matchLabels`: ラベルでPodまたはServiceを選択
  - `matchExpressions`: 式でPodまたはServiceを選択
- **relabelConfigs**: ターゲット検出（Discovery）段階で、メタデータ（メタラベルなど）を基にラベルを追加または変更します。
  （Prometheus の `relabel_configs` と同様）
- **endpoints**: スクレイピング対象のエンドポイント定義
  - `port`: スクレイピングするポート名または番号
  - `path`: メトリクスパス（デフォルトは globalPath、必要に応じて上書き可能）
  - `interval`: スクレイピング間隔（デフォルトは globalInterval、必要に応じて上書き可能）

- `scheme` : スクレイピングプロトコル（http または https、デフォルトは http）
- `timeout` : スクレイピングのタイムアウト
- `honorLabels` : 対象から提供されるラベルを優先するかどうか
- `metricRelabelConfigs` : スクレイピング後のメトリクス再ラベリング設定（Prometheus の metric_relabel_configs に類似）

StaticEndpoints 設定要素

- **targetName**: ターゲットの名前（ログおよび識別用）
- **type**: ターゲットのタイプ（"StaticEndpoints"）
- **endpoints**: スクレイピング対象のエンドポイント定義
  - `address` : スクレイピング対象のアドレス（IP:PORT または HOSTNAME:PORT）
  - `path` : メトリクスのパス（デフォルト: /metrics）
  - `scheme` : スクレイピングプロトコル（http または https、デフォルトは http）
  - `interval` : スクレイピング間隔（デフォルト: 60s）
  - `tlsConfig` : TLS設定
  - `metricRelabelConfigs` : スクレイピング後のメトリック再ラベリング設定

addNodeLabel 機能

スクレイピング対象がノードにスケジューリングされている場合、 `addNodeLabel` オプションを使用して、スケジューリングされたノード名をメトリクスのラベルとして追加できます。

- **設定位置**：ターゲットレベルまたはエンドポイントレベルで設定可能
- **既定値**: `false`
- **動作**： `true` に設定すると、すべてのメトリクスに `node` ラベルが追加され、オブジェクトが実行されているノード名が値として付加されます。

使用例

```
# ターゲットレベルで addNodeLabel を設定
- targetName: node-exporter
```

```
  type: PodMonitor
  namespaceSelector:
    matchNames:
      - "monitoring"
  selector:
    matchLabels:
      app: node-exporter
  endpoints:
    - port: "metrics"
      path: "/metrics"
      interval: "30s"
      addNodeLabel: true  # このエンドポイントにノードラベルを追加
      metricRelabelConfigs:
        # ノードラベルを使用したリラベリングの例
        - source_labels: [node]
          target_label: kubernetes_node
          action: replace

# エンドポイントレベルで addNodeLabel を設定
- targetName: dcgm-exporter
  type: PodMonitor
  namespaceSelector:
    matchNames:
      - "kube-system"
  selector:
    matchLabels:
      app: dcgm-exporter
  endpoints:
  - port: "metrics"
    path: "/metrics"
    interval: "30s"
    addNodeLabel: true  # エンドポイントレベルで addNodeLabel を設定
            metricRelabelConfigs:
      - source_labels: [node]
        target_label: gpu_node
        action: replace
```

Basic Auth 設定

OpenAgent は、認証が必要なエンドポイントをスクレイピングするために HTTP Basic Authentication をサポートし

ています。 ユーザー名（username）とパスワード（password）は、Kubernetes Secret を使用して安全に管理されます。

- 設定場所: endpoints 配下の basicAuth 項目で設定します。
- **username**: ユーザー名を格納した Secret 情報
  - `name` : Secret の名前
  - `key` : Secret 内のキー
  - `namespace` : Secret が配置されているネームスペース（省略した場合、エージェントと同じネームスペースが使用されます。）
- **password**: パスワードを格納した Secret 情報
  - `name` : Secret の名前
  - `key` : Secret 内のキー
  - `namespace` : Secret が配置されているネームスペース

TLS 設定

OpenAgent は、HTTPS エンドポイントに接続する際に TLS（Transport Layer Security）をサポートします。以下は TLS に関する設定オプションです。

HTTP vs HTTPS の決定方法

OpenAgent は以下のルールに従って、HTTP または HTTPS プロトコルを使用するかを決定します。

- **PodMonitor および ServiceMonitor ターゲット**
  - ポート名が "https" の場合は、デフォルトで HTTPS が使用されます。
  - それ以外の場合は、デフォルトで HTTP が使用されます。
- **StaticEndpoints ターゲット**
  - TLS 設定が存在する場合は、デフォルトで HTTPS が使用されます。
  - それ以外の場合は、デフォルトで HTTP が使用されます。
- **すべてのターゲットタイプ**

- エンドポイントまたはターゲットに `scheme` 設定が明示的にある場合、その設定が既定値を上書きします。

TLS 設定オプション

TLS の設定は `tlsConfig` セクションで構成できます。

```
endpoints:
 - port: "https"
   path: "/metrics"
   scheme: "https"  # HTTPS を明示的に使用
   tlsConfig:
    insecureSkipVerify: true  # 証明書の検証をスキップ
```

insecureSkipVerify

insecureSkipVerify` オプションは、サーバー証明書の有効性検証をスキップする設定です。このオプションは以下のような場合に有用です。

- 自己署名証明書を使用するサーバーに接続する場合
- 開発またはテスト環境で証明書の検証が不要な場合
- 内部ネットワーク内で信頼できるサーバーに接続する場合

> **注意**
>
> 本番環境ではセキュリティ上の理由から、`insecureSkipVerify: false` を使用することを推奨します。自己署名証明書を使用する場合は、その証明書を信頼できる認証局（CA）として追加する方が、より安全な方法です。

設定例

ServiceMonitorでのTLS設定例

```
- targetName: kube-apiserver
  type: ServiceMonitor
```

```
namespaceSelector:
  matchNames:
    - "default"
selector:
  matchLabels:
    component: apiserver
    provider: kubernetes
endpoints:
  - port: "https"  # ポート名が "https" のため、デフォルトで HTTPS を使用
    path: "/metrics"
    interval: "30s"
    scheme: "https"  # 明示的に HTTPS を指定（任意）
    tlsConfig:
      insecureSkipVerify: true  # 証明書の検証をスキップ
    metricRelabelConfigs:
      - source_labels: [__name__]
        regex: "apiserver_request_total"
        action: keep
      - source_labels: [verb]
        target_label: http_verb
        replacement: "${1}"
        action: replace
```

ターゲット再ラベリング設定

`relabelConfigs` は、スクレイピングを実行する前に、
ターゲット（Target）自体のメタデータを基にフィルタリングやラベル操作を行う機能です。
（Prometheus の `relabel_configs` と同様です）

1. **ターゲットフィルタリング（Target Filtering）**
   - 特定のメタデータ条件を満たさないターゲットを収集対象から除外します。
   - 主な Action: `keep` , `drop`
2. **メタラベルの昇格（Meta Label Promotion）**
   - `__meta_` で始まるメタデータラベルは、スクレイピング後に削除されます。
   - 永続的なラベルとして保存するには、この段階で `replace` action を使用して新しい名前に変更する必要があります。

Meta Labels 一覧

以下は、OpenAgent が提供する主要なメタラベルの一覧です。

Common

- `__meta_kubernetes_namespace` : オブジェクトのネームスペース
- `__address__` : ターゲットアドレス (IP:Port)

PodMonitor

- `__meta_kubernetes_pod_name` : Pod名
- `__meta_kubernetes_pod_ip` : Pod IP
- `__meta_kubernetes_pod_ready` : Podの準備状態 (`true` / `false`)
- `__meta_kubernetes_pod_phase` : Podの状態 (`Running` , `Pending` など)
- `__meta_kubernetes_pod_node_name` : Podが実行されているノード名
- `__meta_kubernetes_pod_label_<labelname>` : Podのラベル
- `__meta_kubernetes_pod_annotation_<annotationname>` : Podのアノテーション

ServiceMonitor

- `__meta_kubernetes_service_name` : Service名
- `__meta_kubernetes_service_cluster_ip` : Cluster IP
- `__meta_kubernetes_service_label_<labelname>` : Serviceのラベル

```
targets:
  - targetName: my-pod-monitor
    type: PodMonitor
    selector:
      matchLabels:
        app: my-app
    # ターゲット検出段階のラベリング
    relabelConfigs:
      # 1. Pod 名を pod_name ラベルとして保存（メトリクスに pod_name ラベルが付与されます）
```

```
- source_labels: ["__meta_kubernetes_pod_name"]
  target_label: "pod_name"
  action: "replace"

# 2. 特定のラベル（scrape_enabled=true）が設定されていない Pod は収集対象から除外
- source_labels: ["__meta_kubernetes_pod_label_scrape_enabled"]
  regex: "true"
  action: "keep"
```

メトリック再ラベリング設定

OpenAgent は、Prometheus の `metric_relabel_configs` に類似したメトリック再ラベリング機能をサポートしています。この機能を使用すると、スクレイピング後のメトリクスをフィルタリングしたり、ラベルを変更したりできます。

再ラベリング設定項目

- **source_labels**: ソースラベルのリスト（配列）
- **separator**: ソースラベルの値を連結する際に使用する区切り文字（デフォルト：`;`）
- **target_label**: 対象ラベル（結果を保存するラベル）
- **regex**: ソースラベルの値に適用する正規表現
- **replacement**：置換値（正規表現キャプチャグループの参照が可能、例：`${1}`）
- **action**：実行するアクション（keep、drop、replace）

対応しているアクション（action）

- **keep**：正規表現と一致するメトリクスのみを保持
- **drop**：正規表現と一致するメトリクスを削除
- **replace**：対象ラベルの値を置換値に変更

特殊ラベル

- **name**：メトリクス名を表す特殊ラベル

例題

例題１. 特定のメトリクスのみを保持

この設定は `http_requests_total` メトリクスのみを保持し、それ以外をすべて削除します。

```
metricRelabelConfigs:
 - source_labels: [__name__]
   regex: "http_requests_total"
   action: keep
```

- **動作例**

  スクレイプされたメトリクスが以下のようなものだと仮定します。

  ```
  http_requests_total{method="GET", status="200"} 100
  http_errors_total{method="GET", status="500"} 5
  node_cpu_seconds_total{cpu="0", mode="idle"} 1000
  ```

  上記の `metricRelabelConfigs` を適用すると、`http_requests_total` メトリクスのみが保持され、その他のメトリクスはすべて削除されます。

  ```
  http_requests_total{method="GET", status="200"} 100
  ```

例題２. 正規表現を使用したメトリクスのフィルタリング

この設定は `node_cpu` または `node_memory` で始まるメトリクスのみを保持します。

```
metricRelabelConfigs:
 - source_labels: [__name__]
   regex: "node_(cpu|memory).*"
   action: keep
```

- **動作例**

  スクレイプされたメトリクスが以下のようなものだと仮定します。

```
node_cpu_seconds_total{cpu="0", mode="idle"} 1000
node_memory_MemTotal_bytes{} 16777216
node_disk_io_time_seconds_total{device="sda"} 100
http_requests_total{method="GET", status="200"} 100
```

上記の `metricRelabelConfigs` を適用すると、`node_cpu` や `node_memory` で始まるメトリクスのみが保持され、それ以外のメトリクスはすべて削除されます。正規表現を使うことで、複数のメトリクスパターンを一度にフィルタリングすることができます。

```
node_cpu_seconds_total{cpu="0", mode="idle"} 1000
node_memory_MemTotal_bytes{} 16777216
```

例題３. ラベル名の変更

この設定は `method` ラベルの値を `http_method` ラベルにコピーします。

```
metricRelabelConfigs:
 - source_labels: [method]
   target_label: http_method
   replacement: "${1}"
   action: replace
```

- **動作例**

  スクレイプされたメトリクスが以下のようなものだと仮定します。

  ```
  http_requests_total{method="GET", path="/api", status="200"} 100
  http_requests_total{method="POST", path="/api/users", status="201"} 50
  ```

  上記の metricRelabelConfigs を適用すると、各メトリクスに `method` ラベルの値をコピーした `http_method` ラベルが追加されます。元のラベルは維持され、新しいラベルが追加されます。`${1}` はソースラベルの値を参照します。

  ```
  http_requests_total{method="GET", path="/api", status="200", http_method="GET"} 100
  http_requests_total{method="POST", path="/api/users", status="201", http_method="POST"} 50
  ```

例題 4. 複数のソースラベルの組み合わせ

```
metricRelabelConfigs:
 - source_labels: [__name__, status]
   regex: "http_requests_total;(200|500)"
   action: keep
```

この設定では、`http_requests_total` メトリクスのうち、`status` ラベルが `200` または `500` のメトリクスのみを保持します。

- **動作例**

  スクレイプされたメトリクスが以下のようなものだと仮定します。

  ```
  http_requests_total{method="GET", path="/api", status="200"} 100
  http_requests_total{method="POST", path="/api/users", status="201"} 50
  http_requests_total{method="GET", path="/api/error", status="500"} 10
  http_requests_total{method="GET", path="/api/error", status="404"} 5
  ```

  上記の metricRelabelConfigs を適用すると、`http_requests_total` メトリクスのうち、`status` ラベルが `200` または `500` のメトリクスのみが保持されます。複数のソースラベルを組み合わせる場合、デフォルトでは `;` 区切りで連結され、`separator` フィールドで変更可能です。

  ```
  http_requests_total{method="GET", path="/api", status="200"} 100
  http_requests_total{method="GET", path="/api/error", status="500"} 10
  ```

例題 5. 静的ラベルの追加

```
metricRelabelConfigs:
 - target_label: metric_src
   replacement: "whatap-open-agent"
   action: replace
```

この設定では、すべてのメトリクスに `metric_src="whatap-open-agent"` ラベルを追加します。ソースラベルを指定しない場合、`replacement` の値がそのままラベルの値として使用されます。この方法を利用すると、すべてのメトリクスに環境名、リージョン、アプリケーション名などの静的ラベルを追加できます。

- **動作例**

  元々スクレイピングされたメトリクスが以下のようなものであると仮定します：

  ```
  http_requests_total{method="GET", path="/api", status="200"} 100
  node_cpu_seconds_total{cpu="0", mode="idle"} 1000
  ```

  上記の `metricRelabelConfigs` を適用すると、すべてのメトリクスに `metric_src="whatap-open-agent"` ラベルが追加されます。この方法は、メトリクスの出所を示したり、環境（例：production、staging）、リージョン（例：us-east、eu-west）、またはアプリケーション名などを示すのに有用です。

  ```
  http_requests_total{method="GET", path="/api", status="200", metric_src="whatap-open-agent"} 100
  node_cpu_seconds_total{cpu="0", mode="idle", metric_src="whatap-open-agent"} 1000
  ```

例題. 総合的な動作

スクレイプされたメトリクスが以下のようなものだと仮定します。

```
apiserver_request_total{code="200", resource="pods", verb="GET"} 100
some_other_metric{label="value"} 50
```

上記の `metricRelabelConfigs` を適用すると

1. 1つ目のルール（keep apiserver_request_total）の適用：
   - apiserver_request_total メトリクスは保持されます。
   - some_other_metricメトリクスは削除されます。
2. 2つ目のルール（replace verb → http_verb）の適用：
   - 保持された `apiserver_request_total` メトリクスに `verb` ラベルが存在するため、このラベルの値（例： `GET` ）が `http_verb` という新しいラベルにコピーされます。

したがって、最終的に収集されるメトリクスは次のようになります。

```
apiserver_request_total{code="200", resource="pods", verb="GET", http_verb="GET"} 100
```

Kubernetes メトリクス収集の例

以下は、Kubernetes API サーバーからメトリクスを収集する例です。

- この設定では、kube-system ネームスペース内で component=apiserver および provider=kubernetes ラベルを持つサービスを対象に、該当サービスのエンドポイントからメトリクスを収集します。
- metricRelabelConfigs を使用して apiserver_request_total メトリクスのみを収集し、verb ラベルを http_verb ラベルに変換し、すべてのメトリクスに metric_src="whatap-open-agent" という静的ラベルを追加するよう指定できます。

Kubernetes メトリクス収集例

```yaml
# scrape_config.yaml
features:
 openAgent:
  enabled: true
  targets:
   - targetName: kube-apiserver
     type: ServiceMonitor
     namespaceSelector:
      matchNames:
       - "kube-system"
     selector:
      matchLabels:
       component: apiserver
       provider: kubernetes
     endpoints:
      - port: "https"
        path: "/metrics"
        interval: "30s"
        metricRelabelConfigs:
         - source_labels: [__name__]
           regex: "apiserver_request_total"
           action: keep
         - source_labels: [verb]
           target_label: http_verb
           replacement: "${1}"
           action: replace
```

```
    # 静的ラベルの追加
    - target_label: metric_src
      replacement: "whatap-open-agent"
      action: replace
```

WhaTap CR 設定オプション

WhaTap CR の設定オプションは、イメージ、環境変数、リソース、スケジューリング、認証、優先度などを細かく指定してエージェント動作を制御する方法を提供します。

イメージ構成

```
apiVersion: monitoring.whatap.com/v2alpha1
kind: WhatapAgent
metadata:
 name: whatap
spec:
 features:
  k8sAgent:
   # フルイメージ名
   customImageFullName: public.ecr.aws/whatap/kube_agent:1.8.8
   masterAgent:
    enabled: true
    masterAgentContainer:
     # コンテナ単位で上書き可能
     customImageFullName: public.ecr.aws/whatap/kube_agent:1.8.8
   nodeAgent:
    enabled: true
    nodeAgentContainer:
     # コンテナ単位で上書き可能
     customImageFullName: public.ecr.aws/whatap/kube_agent:1.8.8
    nodeHelperContainer:
     # コンテナ単位で上書き可能
     customImageFullName: public.ecr.aws/whatap/kube_agent:1.8.8
  openAgent:
   enabled: true
   # コンテナ単位で上書き可能
   customImageFullName: public.ecr.aws/whatap/open_agent:latest
```

1. コンテナ env でオプションを設定

Kubernetes エージェントはコンテナの環境変数でオプションを設定できます。

- オペレーターは既定の環境変数(WHATAP_LICENSE/HOST/PORT、WHATAP_MEM_LIMIT など)を自動で追加します。 `nodeAgent.nodeAgentContainer.envs` に指定した値は既定値の後にマージされます。

```
# WhaTap ノードエージェント DaemonSet の whatap-node-agent コンテナオプションを設定する場合
apiVersion: monitoring.whatap.com/v2alpha1
kind: WhatapAgent
metadata:
 name: whatap
spec:
 features:
  k8sAgent:
   masterAgent:
    enabled: true
   nodeAgent:
    enabled: true
    nodeAgentContainer:
     envs:
      - name: debug
       value: "true"
   gpuMonitoring:
    enabled: false
```

2. resources の設定方法

オペレーターには k8s エージェント導入の既定値が含まれています。

```
# WhaTap ノードエージェント DaemonSet の whatap-node-agent コンテナオプションを設定する場合
apiVersion: monitoring.whatap.com/v2alpha1
kind: WhatapAgent
metadata:
 name: whatap
spec:
 features:
  k8sAgent:
   masterAgent:
    enabled: true
    # マスターエージェントのコンテナリソース(コンテナ単位の上書き)
    masterAgentContainer:
     resources:
```

```
      requests:
       cpu: "150m"
       memory: "256Mi"
      limits:
       cpu: "300m"
       memory: "512Mi"
  nodeAgent:
   enabled: true
   # ノードエージェントのコンポーネントレベル既定リソース
   resources:
    requests:
     cpu: "100m"
     memory: "300Mi"
    limits:
     cpu: "200m"
     memory: "350Mi"
   # ノードエージェントのコンテナ上書き
   nodeAgentContainer:
    resources:
     requests:
      cpu: "150m"
      memory: "350Mi"
     limits:
      cpu: "300m"
      memory: "512Mi"
   # ヘルパーコンテナ上書き
   nodeHelperContainer:
    resources:
     requests:
      cpu: "100m"
      memory: "100Mi"
     limits:
      cpu: "200m"
      memory: "350Mi"
  gpuMonitoring:
   enabled: false
```

- コンテナごとに `resources` を定義した場合、その値が当該コンテナのコンポーネント既定値を上書きします。

コンテナ	requests	limits
whatap-master-agent	cpu: 100m memory: 300Mi	cpu: 200m memory: 350Mi
whatap-node-agent	request: 100m limit: 300Mi	request: 200Mi limit: 350Mi
whatap-node-helper	request: 100m limit: 100m	request:200Mi limit: 350Mi

3. tolerations、affinity、nodeSelector

tolerations

```yaml
spec:
 features:
  k8sAgent:
   masterAgent:
    enabled: true
    tolerations:
     - key: "node.kubernetes.io/not-ready"
       operator: "Exists"
       effect: "NoExecute"
       tolerationSeconds: 300
   nodeAgent:
    enabled: true
    tolerations:
     - key: "dedicated"
       operator: "Equal"
       value: "observability"
       effect: "NoSchedule"
```

affinity

マスターおよびノードエージェントには次の既定の tolerations が内蔵されています。CR で設定した値はこれらの既定値の後にマージされます。

- `node-role.kubernetes.io/master:NoSchedule`
- `node-role.kubernetes.io/control-plane:NoSchedule`

```
spec:
 features:
  k8sAgent:
   masterAgent:
    enabled: true
    affinity:
     nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
       nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/os
            operator: In
            values: ["linux"]
   nodeAgent:
    enabled: true
    affinity:
     podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
       - weight: 100
         podAffinityTerm:
          topologyKey: kubernetes.io/hostname
          labelSelector:
           matchLabels:
            name: whatap-node-agent
```

nodeSelector

- マスターエージェントに nodeSelector を適用

```
spec:
 features:
  k8sAgent:
   masterAgent:
    enabled: true
    nodeSelector:
     nodepool: system        # 例) system ノードプールへ固定
```

```
    kubernetes.io/os: linux    # Linux ノードのみ
```

- ノードエージェント(DaemonSet) に `nodeSelector` を適用

```
spec:
 features:
  k8sAgent:
   nodeAgent:
    enabled: true
    nodeSelector:
     nodepool: worker         # 例) ワーカーノードのみ
     kubernetes.io/arch: amd64  # AMD64 アーキテクチャノードのみ
```

> ⓘ ノードエージェントは全ノードに配備される DaemonSet です。`nodeSelector` を設定すると、条件を満たすノードにのみ Pod が作成されます。

  - GPU クラスタで GPU ノードのみに配備したい場合は次のように設定します。

```
spec:
 features:
  k8sAgent:
   nodeAgent:
    enabled: true
    nodeSelector:
     nvidia.com/gpu.present: "true"
   gpuMonitoring:
    enabled: true
```

- OpenAgent に `nodeSelector` を適用

```
spec:
 features:
  openAgent:
   enabled: true
   nodeSelector:
    nodepool: system          # OpenAgent を system ノードプールへ固定
    kubernetes.io/os: linux
```

```
    # 任意: より細かなスケジューリングには affinity と併用
    affinity:
     nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
       nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/hostname
            operator: In
            values: ["ip-10-0-0-12"]
```

4. imagePullSecret

```
spec:
 features:
  k8sAgent:
   # グローバル pull secret(マスター/ノード共通の既定)
   imagePullSecrets:
    - name: global-regcred
   masterAgent:
    enabled: true
    # マスター専用の上書き
    imagePullSecrets:
     - name: master-regcred
   nodeAgent:
    enabled: true
    # ノード専用の上書き
    imagePullSecrets:
     - name: node-regcred
  openAgent:
   enabled: true
   imagePullSecrets:
    - name: openagent-regcred
```

5. PriorityClassName

```
spec:
 features:
```

```
k8sAgent:
  masterAgent:
    enabled: true
    priorityClassName: system-cluster-critical
  nodeAgent:
    enabled: true
    priorityClassName: system-node-critical
openAgent:
  enabled: true
  priorityClassName: system-cluster-critical
```

マスターおよびノードエージェントのインストール

次のように、マスター及びノードエージェントのインストール手順を案内します。 **インストール案内**画面と同じ手順です。

> EKS Fargateは、対応予定です。

インストール方法の選択

アクセスキー確認後、**インストールの案内**画面で**インストール方法の選択**タブを選択します。 WhaTap Kubernetesエージェントがサポートする**Yaml**ファイルと**Helm**チャートのインストール方法の中から希望するインストール方法を選択し、ユーザーのKubernetes環境に合わせて適切な項目を選択します。



1. WhaTap Kubernetesエージェントのイメージバージョンを推奨する**Stable**バージョンと**Preview**バージョンの中から選択します。
2. **yaml**ファイルと**helm**チャートのインストール方法の中から希望する**インストール方式**を選択します。
3. **1.16以降**バージョンまたは**1.16未満**バージョンの中で該当する**Kubernetesバージョン**を選択します。
4. **containerd**, **Docker Engine**, **CRI-O**の中、**コンテナランタイム**を選択します。
5. **ユーザーKubernetes環境**(GKE, OpenShift, Istio, GPU Cluster) の中、該当する項目がある場合は選択します。
6. インストール方法に関する項目を選択した後、画面右下の**次**ボタンをクリックするか、左の**追加機能の適用**タブを選択して次のステップへ移動します。

WhaTap Kubernetesエージェントイメージリポジトリの追加対応

WhaTap Kubernetesエージェントは、従来のDockerHubの他にPublic ECRに追加対応します。 使用環境に応じて、2つのイメージリポジトリのいずれかを選択してインストールできます。

Public ECR

- リポジトリ: `public.ecr.aws/whatap/kube_mon` 推奨

ECRは、pull制限ポリシーの影響を受けないため、頻繁なアップデートや大規模な配布環境で安定的に使用できるのでお勧めします。

DockerHub

- リポジトリ: `whatap/kube_mon`

DockerHubは、認証されていないクライアントに対してpull制限(policy)が適用されることがあります。 ノード数が多かったり、同時にpullリクエストが発生した場合、rate limitに引っかかってインストールが失敗する可能性が高いです。 詳しくはDockerHubポリシー案内の公式文書を参考にしてください。

> pull制限が発生した場合は、次の文書を参考にしてください。

追加機能の適用

WhaTap Kubernetesモニタリングは、様々な追加機能を提供します。



追加適用された機能が多いほど、収集されるデータの量とエージェントが使用するリソースの量が増加する可能性があります。 インストール後、エージェント設定により、その機能を活性化または非活性化できます。

追加機能を適用するかどうかを選択した後、画面右下の**次へ**ボタンをクリックするか、左の**インストールファイルのダウンロード**タブを選択して次のステップへ移動します。

コントロールプレーンモニタリング活性化

コントロールプレーンモニタリングアクティベーションと各機能の詳細については、次の文書を参照してください。

- **kube-apiserver ダッシュボード** ショートカット
- **kube-apiserver メトリクス照会** ショートカット
- **etcd ダッシュボード** ショートカット
- **kube-scheduler ダッシュボード** ショートカット

Custom Resourceモニタリング活性化

- Yaml Helm Chart

WhaTap Kubernetesエージェント設置yamlファイルで `ClusterRole` リソースの `rules` 下位の `resources` 項目に使用しようとする `customResource` を追加します。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  creationTimestamp: "2025-01-15T01:48:26Z"
  name: whatap
  resourceVersion: "125200773"
  uid: 54b0bcdb-1f12-4779-b0cb-e8de2f6be64a
rules:
- apiGroups:
  - '*'
  resources:
  - namespaces
  - pods
  - pods/log
  - configmaps
  - services
  - endpoints
  - daemonsets
  - ingresses
  - deployments
  - nodes
  - persistentvolumes
  - persistentvolumeclaims
  - events
  - replicasets
  - roles
  - rolebindings
  - clusterroles
  - clusterrolebindings
  - jobs
  - cronjobs
  - statefulsets
  - serviceaccounts
```

```
  - configmaps
  - storageclasses
  - horizontalpodautoscalers
  - replicationcontrollers
  - customresourcedefinitions # CRD収集権限を追加
  - samplecrds # モニタリングするCustom Resourceの名前を追加
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - ""
  resources:
  - pods/exec
  verbs:
  - create
- apiGroups:
  - ""
  resources:
  - configmaps
  verbs:
  - '*'
- nonResourceURLs:
  - /metrics
  verbs:
  - '*'
```

WhaTap Kubernetesエージェントを Helmチャート方式で設置する場合、基本values.yamlファイルにモニタリングするCustom Resource名前を追加します。 次のように `resources` 項目に該当Custom Resource名前を追加します。

```
collect_custom_resource: true
clusterrole:
  extraResources:
    - "customresourcedefinitions" ##crd収集権限を追加
    - "samplecrds" ##収集しようとするcustom resourceを追加
```

> ⓘ **Object Manifest**メニューで**Custom Resourceモニタリング**ができます。

新規収集対象のClusterRole権限

次の収集機能は、`ClusterRole`リソースの`rules`配下の`resources`項目に照会権限を追加した場合にのみ動作します。

Resource	Feature	Description
`networkpolicies`	NetworkPolicyイベント収集	`kube_network_policy`カテゴリーの収集に必要
`resourcequotas`	ネームスペースのResourceQuota収集	`kube_stat`カテゴリーの`quota_*`フィールドの収集に必要

- 権限がない場合、その収集のみが省略され、残りの収集は継続されます。

インストール ファイル ダウンロード

インストールファイルをダウンロードする前に、ユーザーKubernetes環境に適した項目が選択されていることを確認してください。 選択項目を確認した後、**インストールファイルのダウンロード**ボタンをクリックしてインストールに必要なデフォルト設定ファイル(values.yaml)をダウンロードします。



ダウンロード後、画面右下の**次へ**ボタンをクリックするか、**実行コマンドの入力**タブを選択して次のステップへ移動します。

実行コマンドの入力

インストール方法の選択タブで選択した環境とインストール方法の案内に沿って、実行コマンドを入力し、インストールします。

インストール前の注意事項

Kubernetesエージェントは**whatap-monitoringネームスペース**にインストールされます。
エージェントがインストールされるネームスペースは変更できません。
WhaTap Server 127.0.0.1へデータを転送するには、 **6606番のポート**を開く必要があります。(TCP Outbound)

インストールガイド

チャットでのお問い合わせ

1. アクセスキーの確認
2. インストール方式の選択
3. 追加機能の適用
4. インストールファイルのダウンロード
5. 実行コマンドの入力

追加インストール

1. Application Monitoringの追加インストール

実行コマンドの入力

次のコマンドを実行して、WhaTap Helmリポジトリを追加してください。

```
helm repo add whatap https://whatap.github.io/helm/
helm repo update
```
コピー

エージェントを初めてインストールするユーザーは、次のコマンドを実行してエージェントプログラムをインストールします。

```
helm install whatap-kube-agent whatap/kube -f values.yaml
```
コピー

既存のWhaTap Kubernetesエージェントのユーザーは、次のコマンドを実行してエージェントをインストールします。

```
kubectl delete ns whatap-monitoring
kubectl delete clusterrole whatap
kubectl delete clusterrolebinding whatap
helm install whatap-kube-agent whatap/kube -f values.yaml
```
コピー

*既存のyamlファイルでWhaTap Kubernetesエージェントをインストールした場合は、Clean Installが必要です。
既存のKubernetesエージェントを削除した後、Helmチャートを通じてインストールできます。

Kubernetesエージェントのインストールが正常に完了すると、
次のコマンドを実行してマスターエージェントとノードエージェントの状態が確認できます。

```
kubectl get pod -n whatap-monitoring
```
コピー

次のように、**STATUS**エントリが **Running** の場合、インストールは完了です。

```
[root@kube-master ~]# kubectl get pod -n whatap-monitoring
NAME                                READY   STATUS    RESTARTS   AGE
whatap-master-agent-b4d7cbcc8-wblzt   1/1     Running   0          5m5s
whatap-node-agent-qw9dl             2/2     Running   0          5m5s
whatap-node-agent-rgjk8             2/2     Running   0          5m5s
whatap-node-agent-xvwt2             2/2     Running   0          5m5s
[root@kube-master ~]#
```

前へ　　追加モニタリングのインストール　　モニタリング開始ガイド

Kubernetes環境

インストール方法の選択タブで**GKE**または**OpenShift**環境を選択した場合は、次のガイドを参照してください。 該当しない場合は、次のKubernetesエージェント作成 ステップに進みます。

- **GKE環境** **OpenShift環境**

インストール方法の選択タブで**Google Kubernetes Engine**(**GKE**) 環境を選択した場合は、ユーザークラスターを作成した後、権限設定のために次のコマンドを実行します。

```
ACCOUNT=$(gcloud info --format='value(config.account)')
kubectl create clusterrolebinding owner-cluster-admin-binding \
  --clusterrole cluster-admin \
  --user $ACCOUNT
```

> ⓘ gcloudコマンドのインストールの詳細については、グーグルクラウド文書を参照してください。

インストール方法の選択タブで**OpenShift**環境を選択した場合は、権限設定のために次のコマンドを実行します。

```
oc create -f whatap_openshift_scc.yaml
```

Kubernetesエージェント作成

WhaTap Kubernetesエージェントのインストール時**Yaml**ファイル基準のインストール方法の案内と**Helm**チャート基準のインストール方法の案内を次のように提供します。

- **Yaml** **Helm Chart**

Yamlファイル基準インストール

インストール方法の選択タブで**yaml**ファイルのインストール方法を選択した場合は、エージェントを生成するために次のコマンドを実行します。

```
kubectl apply -f whatap_kube_agent.yaml
```

> ⓘ **マスターエージェントが使用するリソースの量**は、**ユーザー環境のノード数**にある程度**比例**します。 つまり、クラスタ内にノード数が多い場合、既定値に設定されたメモリLimit(350MB)以上のリソースを使ってOOM(Out Of Memory)が発生する可能性があります。 マスターエージェントに割り当てられたメモリLimitは、エージェントのインストールyamlファイルから変更できます。

Helmチャート基準のインストール

> ⓘ WhaTap Kubernetesエージェントは基本的に `whatap-monitoring` ネームスペースを使用します。 該当ネームスペースが使用できないか、ユーザーが特定のネームスペースを使用する必要がある場合は、次の文書を参照してください。

インストール方法の選択タブで**helm**チャートのインストール方法を選択した場合は、次のガイドを参照してください。

1. 次のコマンドを通じてhelmバージョンを確認します。 3.xバージョン以降をサポートします。

```
helm Version

# 出力例
# version.BuildInfo{Version:"v3.x", GitCommit:"xxx", GitTreeState:"clean", GoVersion:"go1.20.8"}
```

2. WhaTap Helmリポジトリに対するネットワーク接続状態を確認してください。

```
curl https://whatap.github.io/helm/ping
# 出力例
# pong
```

3. 次のコマンドを実行して、WhaTap Helmリポジトリを追加します。

```
helm repo add whatap https://whatap.github.io/helm/
helm repo update
```

4. 次のようにエージェントアプリケーションをインストールします。
    - エージェントを**初めて**インストールするユーザーは、次のコマンドを実行してエージェントアプリケー

ションをインストールします。

```
helm install whatap-kube-agent whatap/kube -f values.yaml
```

- **既存**WhaTap Kubernetesエージェントのユーザーは、次のコマンドを実行してエージェントアプリケーションをインストールします。

```
kubectl delete ns whatap-monitoring
kubectl delete clusterrole whatap
kubectl delete clusterrolebinding whatap
helm install whatap-kube-agent whatap/kube -f values.yaml
```

> ⓘ 既存 yaml ファイル方式でWhaTap Kubernetesエージェントをインストールした場合、Clean Installが必要です。 既存のKubernetesエージェントを削除した後、helm チャートによるインストールが可能です。

values.yamlの主な設定項目

WhaTap Kubernetesエージェントの設定を修正するには、デフォルト設定ファイル(values.yaml)を修正する必要があります。 values.yaml ファイル内の主な設定項目を次のように案内します。

キー	タイプ	既定値	説明
whatap.license	String	# <license-key>	Whatapライセンスキー
whatap.host	String	# <whatap-server-host>	Whatapサーバーのホストアドレス
whatap.port	Int	# <whatap-server-port>	Whatapポート番号
containerRuntime	String	"docker"	使用中のコンテナランタイム。 "docke

キー	タイプ	既定値	説明
			r"、"containerd"、"crio"のいずれかを選択
daemonSet.name	String	whatap-node-agent	DaemonSetの名前
daemonSet.label	String	whatap-node-agent	DaemonSetに指定するラベル
daemonSet.initContainers.nodeDebugger.enabled	Bool	true	Whatapノードデバッガコンテナの活性化有無
daemonSet.containers.nodeHelper.image	String	whatap/kube_mon	nodeHelperコンテナのイメージ
daemonSet.containers.nodeHelper.requests.memory	String	100Mi	nodeHelper MEMORY request
daemonSet.containers.nodeHelper.requests.cpu	String	100m	nodeHelper CPU request
daemonSet.containers.nodeHelper.limits.memory	String	350Mi	nodeHelper MEMORY limit
daemonSet.containers.nodeHelper.limits.cpu	String	200m	nodeHelper CPU limit
daemonSet.containers.nodeAgent.image	String	whatap/kube_mon	nodeAgentコンテナのイメージ
daemonSet.containers.nodeAgent.requests.memory	String	300Mi	nodeAgent MEMORY request
daemonSet.containers.nodeAgent.requests.cpu	String	100m	nodeAgent CPU request

キー	タイプ	既定値	説明
daemonSet.containers.nodeAgent.limits.memory	String	350Mi	nodeAgent MEMORY limit
daemonSet.containers.nodeAgent.limits.cpu	String	200m	nodeAgent CPU limit
deployment.name	String	whatap-master-agent	Deploymentの名前
deployment.label	String	whatap-master-agent	Deploymentに指定するラベル
deployment.replicas	Int	1	Deploymentのreplica数
deployment.containers.controlPlaneHelper.enabled	Boolean	true	whatap-control-plane-helperコンテナの活性化有無
deployment.containers.controlPlaneHelper.image	String	"whatap/kube_mon"	controlPlaneHelperコンテナのイメージ
deployment.containers.controlPlaneHelper.port	String	9496	controlPlaneHelperコンテナのポート番号
deployment.containers.controlPlaneHelper.resources.requests.memory	String	"300Mi"	controlPlaneHelperコンテナMEMORY request
deployment.containers.controlPlaneHelper.resources.requests.cpu	String	"100m"	controlPlaneHelperコンテナCPU request
deployment.containers.controlPlaneHelper.resources.limits.memory	String	"350Mi"	controlPlaneHelperコンテナMEMORY limit

キー	タイプ	既定値	説明
deployment.containers.controlPlaneHelper.resources.limits.cpu	String	"200m"	controlPlaneHelperコンテナCPU limit
deployment.containers.masterAgent.name	String	whatap-master-agent	masterAgentコンテナ名
deployment.containers.masterAgent.image	String	"whatap/kube_mon"	masterAgentコンテナイメージ
deployment.containers.masterAgent.port	Int	6600	masterAgentコンテナポート番号
deployment.containers.masterAgent.resources.requests.memory	String	"300Mi"	masterAgentコンテナMEMORY request
deployment.containers.masterAgent.resources.requests.cpu	String	"100m"	masterAgentコンテナCPU request
deployment.containers.masterAgent.resources.limits.memory	String	"350Mi"	masterAgentコンテナMEMORY limit
deployment.containers.masterAgent.resources.limits.cpu	String	"200m"	masterAgentコンテナCPU request

ⓘ Helmチャート基準での**インストール問題の解決**についての案内は、次の文書を参照してください。

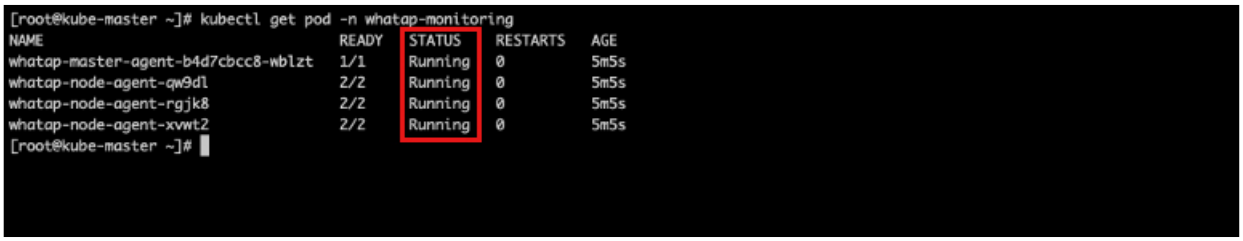
エージェントのインストール確認

Kubernetesエージェントのインストールが正常に完了した場合、次のコマンドを実行してマスターエージェントとノードエージェントの状態を確認できます。

```
kubectl get pod -n whatap-monitoring
```

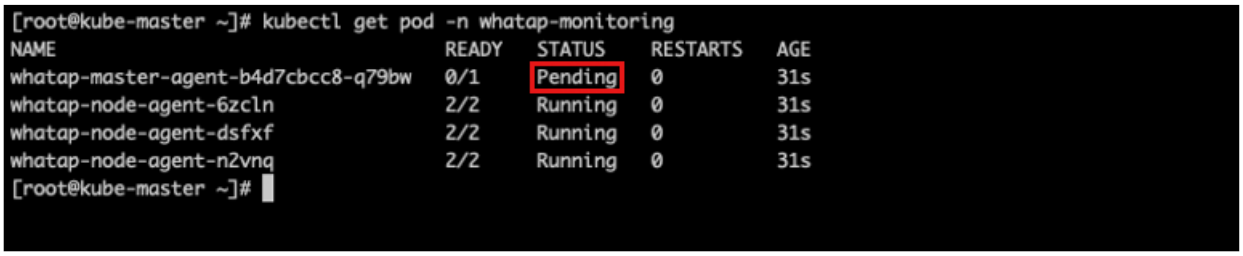
Running状態の場合

次のように**STATUS**項目が**Running**である場合は、**インストール済み**です。 マスター及びノードエージェントのインストールを完了すると、Kubernetes環境のリソースモニタリングを開始します。 WhaTapモニタリングサービスに移動し、**プロジェクトリスト**と**ダッシュボード** >**コンテナマップ**メニューから性能情報を収集することを確認します。



Running状態でない場合

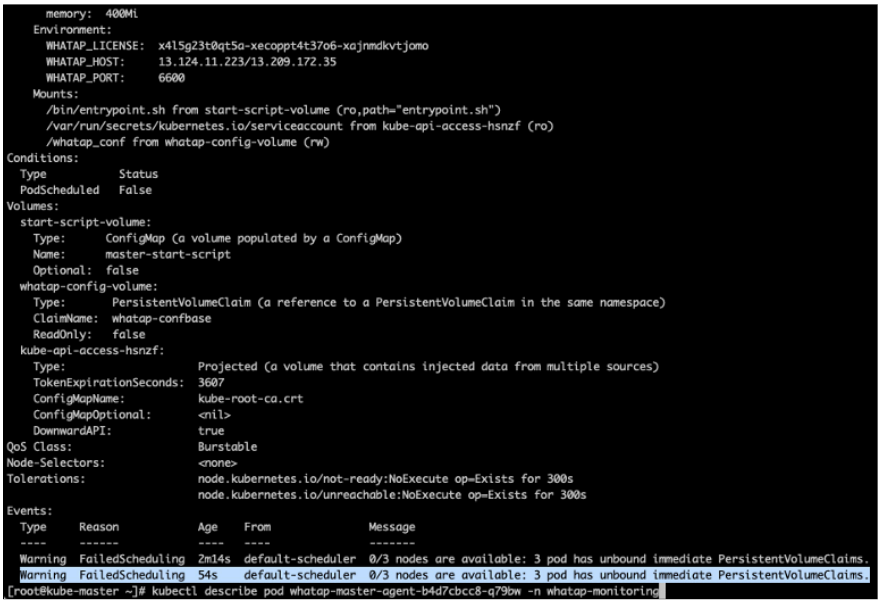
次のように**STATUS**項目が**Running**でない場合、インストールが正しく行われていない状態です。



1. マスターノードで次のコマンドを実行し、Podの生成に失敗した原因を特定します。

   ```
   kubectl describe pod {POD_NAME} -n whatap-monitoring
   ```

2. 実行結果の下部の **Events:** 項目を確認してください。



kubectl コマンドを使用する場合、 -n {ネームスペース} オプションを使用すると、特定のネームスペースを対象にタスクを実行します。 WhaTapエージェントは、基本的に whatap-monitoring というネームスペースにインストールされるため、 -n whatap-monitoring というオプションを追加します。

- 画面右下に**モニタリング開始ガイド**ボタンを選択すると、WhaTap Kubernetesモニタリングの主要メニューに関する案内を確認できます。 ガイド画面の下側にある各メニューのボタンをクリックすると、そのメニューへ移動します。
- アプリケーションエージェントのインストールを進めるには、メニューの画面右下の**追加モニタリングのインストール**ボタンを選択します。 詳細については、次の文書を参照してください。
- エージェントを最新バージョンにアップデートするには次の文書を参照してください。

管理 > アプリケーションエージェントのインストール

コンテナ内のJava、Python、Node.jsなどのアプリケーションをモニタリングするためのエージェントをインストールする過程です。

マイクロアプリケーションエージェントのインストールは、WhaTapのアプリケーションモニタリングエージェントのインストールの手順と同じです。ただし、コンテナ化(Containerization) 作業の追加が異なります。

アプリケーションエージェントのインストールを進めるには、管理 > エージェントのインストールメニューの画面左の一覧追加インストール下にアプリケーションモニタリングの追加インストールタブを選択後、下段のアプリケーションモニタリングの追加インストールボタンをクリックします。



アクセスキーの確認



インストール手順ページのアクセスキーの確認セクションで、プロジェクトアクセスキーを発行するボタンをクリックします。発行済みの場合、アクセスキーが表示されます。

> **アクセスキー**は、他のプロジェクトと国合わせて使用することはできません。**クラスタープロジェクト**認証キーと異なります。

アプリケーションエージェントのコンテナ化

次の文書を参照して、アプリケーションエージェントのコンテナ化を進めてください。

Docker Javaインストール

コンテナ内のJavaアプリケーションをモニタリングするためにエージェントをインストール段階です。

Docker Node.jsインストール

コンテナ内のNode.jsアプリケーションをモニタリングするためのエージェントのインストール手順です。

Docker Pythonインストール

コンテナ内のPythonアプリケーションをモニタリングするためのエージェントのインストール手順です。

Docker PHPインストール

コンテナ内のPHPアプリケーションをモニタリングするためのエージェントのインストール手順です。

Docker Goインストール

コンテナ内のGoアプリケーションをモニタリングするためのエージェントをインストールする過程です。

Docker Javaインストール

管理 > エージェントインストール > 下段のアプリケーションインストール クリック > 設置案内の中で Docker Java タブを選択

Dockerコンテナベースで動作するJavaアプリケーションのJVM Optionにエージェント適用の設定を追加し、コンテナイメージをパッケージングする過程を次のように案内します。

> ⓘ
> - EKS Fargateは、対応予定です。
> - Javaアプリケーションとの連携手順を説明するためGitのサンプルコードを提供します。次の文書を参考にしてください。

エージェントダウンロード

- **Docker Install 直接設置**

WhaTapアプリケーションエージェントのインストールのために whatap.conf ファイルを作成します。
`{YOUR_PROJECT_ROOT}` は、エージェントをインストールするプロジェクトの最上位パスに変更します。

```
cat >{YOUR_PROJECT_ROOT}/whatap.conf <<EOL
whatap.server.host={proxyServer}
EOL
```

Kubernetes内のアプリケーションエージェントのオプション設定

Kubernetes内のアプリケーションエージェントに必要な設定を、whatap.confファイルに作成するか、エージェント配布時のコンテナ `env` フィールドを通して設定できます。

- whatap.confファイル内のオプションがコンテナ `env` フィールドに設定されたオプションよりも高い優先順位を持ちます。
- whatap.confまたはコンテナ `env` に設定された値がない場合は、既定値が使用されます。
- `license` 及び `whatap.server.host` オプションの場合**コンテナ環境変数**設定を活用してください。詳しくは次の文書を確認してください。

設定ファイルの内容が変更されたり、重要な設定項目が消失した場合、正常に動作しない可能性があるため、Kubernetes環境内の**コンテナ環境変数**設定をお勧めします。その他のエージェント機能の制御オプションの詳細は、次の文書を参照してください。

> **Kubernetes環境での使用不可オプション**
>
> - `whatap.name`：収集サーバーがエージェントを識別する固有の名前です。この名前は、エージェントが実行されるオブジェクト情報に基づいて生成されます。ユーザーの任意指定の際、エージェントの識別に問題が発生する可能性があります。
> - `whatap.onode`：デフォルトとして、Kubernetesのクラスターノード名が設定されます。エージェントが属するノード情報を受け取るために使用されます。ユーザーが任意に指定すると、正確なノードの識別ができなくなる可能性があります。

> **アプリケーションの `whatap.server.host` 設定確認**
>
> 一部コンテナ画像では `whatap.server.host` のように `.` が含まれる環境変数を直接設定することはできません。これにより、アプリケーションエージェントが正常に起動しない場合、whatap.confファイルを直接修正し `whatap.server.host` 値を設定する必要があります。アプリケーションの `whatap.server.host` 設定の確認方法は次のとおりです。
>
> 1. WhaTapアプリケーションコンテナに進入します。
> 2. 次のコマンドを実行して `whatap.server.host` 値が含まれていることを確認します。
>
> ```bash
> $ env | grep whatap.server.host
> ```
>
> **値が出力されない場合**当該コンテナの画像が `.` が含まれる環境変数の設定に対応していないことです。この場合、環境変数ではなく設定ファイルを活用する必要があります。whatap.confファイルを開いて次のように `whatap.server.host` 値を手動で設定します。
>
> ```bash
> whatap.server.host={WhaTap収集サーバーIP}
> ```

- 最新バージョンのエージェントをアプリケーションイメージに含
- 指定したバージョンのエージェントをアプリケーションイメージに含む

最新バージョンのエージェントをアプリケーションイメージに含めるように案内します。

1. アプリケーション画像にワタブエージェントを含めるには、まずDockrfileの最終イメージのビルド段階で次の内容を追加します。 {YOUR_PROJECT_ROOT} をエージェントをインストールしようとするプロジェクトの最上位パスに変更します。

```
# WhaTapエージェント用のディレクトリをユーザーコンテナに作成
RUN mkdir -p /whatap

# WhaTap Javaエージェントをユーザーコンテナにコピー
COPY --from=whatap/kube_mon /data/agent/micro/whatap.agent.kube.jar /whatap

# 作成したwhatap.conf設定ファイルをユーザーコンテナにコピー
COPY {YOUR_PROJECT_ROOT}/whatap.conf /whatap/
```

2. Dockerfileの開始コマンドに次の内容を追加します。

```
-javaagent:/whatap/whatap.agent.kube.jar -Dwhatap.micro.enabled=true
```

次のDockerfileで定義の例を参考にしてください。

Dockerfile

```
1 FROM openjdk:8-jdk-slim
2 RUN mkdir -p /app && mkdir /whatap
3 WORKDIR /app
4 COPY --from=whatap/kube_mon /data/agent/micro/whatap.agent.kube.jar /whatap
5 COPY ./whatap.conf /whatap
6 COPY ./target/myApp.jar /app/
7 CMD ["java","-javaagent:/whatap/whatap.agent.kube.jar","-Dwhatap.micro.enabled=true","-jar","/app/
  myApp.jar"]
8 EXPOSE 8080
```

3. Dockerをビルドします。 {YOUR*Dockerfile*DIR} をDockerfileが位置したパスに変更します。

```
cd {YOUR*Dockerfile*DIR}
```

```
docker build -t {IMAGE_NAME} .
```

指定したバージョンのエージェントをアプリケーションイメージに含めるように案内します。

1. アプリケーション画像にワタブエージェントを含めるには、まずDockrfileの最終イメージのビルド段階で次の内容を追加します。`{YOUR_PROJECT_ROOT}`をエージェントをインストールしようとするプロジェクトの最上位パスに変更します。

```
# WhaTapエージェント用のディレクトリをユーザーコンテナに作成
RUN mkdir -p /whatap

# WhaTap Javaエージェントをユーザーコンテナにコピー
COPY --from=whatap/kube_mon /data/agent/micro/whatap.agent-*.jar /whatap

# 作成したwhatap.conf設定ファイルをユーザーコンテナにコピー
COPY {YOUR_PROJECT_ROOT}/whatap.conf /whatap/
```

2. 現在のWhaTapエージェントのバージョンを確認してください。

```
docker run whatap/kube_mon ls /data/agent/micro | grep -E 'whatap\.agent-(.*?\.)(.*?\.)(.*?\.)jar' | sort | tail -1

# 出力例
## whatap.agent-X.Y.Z.jar
```

3. Dockerfileの開始コマンドに次の内容を追加します。**X.Y.Z**に、前もって確認したエージェントバージョンを入力します。

```
-javaagent:/whatap/whatap.agent-X.Y.Z.jar -Dwhatap.micro.enabled=true
```

次のDockerfileの定義例を参照してください。

Dockerfile

```
1 FROM openjdk:8-jdk-slim
2 RUN mkdir -p /app && mkdir /whatap
3 WORKDIR /app
4 COPY --from=whatap/kube_mon /data/agent/micro/whatap.agent-*.jar /whatap
```

```
COPY ./whatap.conf /whatap
COPY ./target/myApp.jar /app/
CMD ["java","-javaagent:/whatap/whatap.agent-X.Y.Z.jar","-Dwhatap.micro.enabled=true","-jar","/app/myApp.jar"]
EXPOSE 8080
```

4. Dockerをビルドします。 {YOUR*Dockerfile*DIR} をDockerfileが位置したパスに変更します。

```
cd {YOUR*Dockerfile*DIR}
docker build -t {IMAGE_NAME} .
```

1. インストールファイル(whatap.agent.java.tar.gz)をダウンロードするには、ダウンロードボタンを選択します。
2. Dockerfileを作成できるサーバーにアップロードし、圧縮を解凍してください。

> インストールするサーバーから直接ダウンロードするには、次のコマンドを実行してください。
>
> ```
> wget https://api.whatap.io/agent/whatap.agent.java.tar.gz
> ```

3. 圧縮を解除する際に、whatap ディレクトリが作成されます。whatap ディレクトリの whatap.conf ファイルで設定内容を確認し、次のように修正します。

```
license={licenseKey}
whatap.server.host={proxyServer}
```

> **Kubernetes内のアプリケーションエージェントのオプション設定**
>
> Kubernetes内のアプリケーションエージェントに必要な設定を、whatap.confファイルに作成するか、エージェント配布時のコンテナ env フィールドを通して設定できます。
>
> - whatap.confファイル内のオプションがコンテナ env フィールドに設定されたオプションよりも高い優先順位を持ちます。

- whatap.confまたはコンテナ `env` に設定された値がない場合は、既定値が使用されます。
- `license` 及び `whatap.server.host` オプションの場合**コンテナ環境変数**設定を活用してください。詳しくは次の文書を確認してください。

設定ファイルの内容が変更されたり、重要な設定項目が消失した場合、正常に動作しない可能性があるため、Kubernetes環境内の**コンテナ環境変数**設定をお勧めします。その他のエージェント機能の制御オプションの詳細は、次の文書を参照してください。

Kubernetes環境での使用不可オプション

- `whatap.name` ：収集サーバーがエージェントを識別する固有の名前です。この名前は、エージェントが実行されるオブジェクト情報に基づいて生成されます。ユーザーの任意指定の際、エージェントの識別に問題が発生する可能性があります。
- `whatap.onode` ：デフォルトとして、Kubernetesのクラスターノード名が設定されます。エージェントが属するノード情報を受け取るために使用されます。ユーザーが任意に指定すると、正確なノードの識別ができなくなる可能性があります。

4. Dockerfileの開始コマンドに次の内容を追加します。_whatap_ディレクトリ内でエージェントファイルを確認した後、**X.Y.Z**に該当バージョンを入力します。

```
-javaagent:/whatap/whatap.agent-X.Y.Z.jar -Dwhatap.micro.enabled=true
```

次のDockerfileの定義例を参照してください。

Dockerfile

```
FROM openjdk:8-jdk-alpine
RUN mkdir -p /app && mkdir /whatap
WORKDIR /app
COPY ./whatap.agent-X.Y.Z.jar /whatap/
COPY ./whatap.conf /whatap/
COPY ./target/myApp.jar myApp.jar
COPY ./paramkey.txt /whatap/
CMD ["java","-javaagent:/whatap/whatap.agent-X.Y.Z.jar","-Dwhatap.micro.enabled=true","-jar","/app/
myApp.jar"]
```

```
EXPOSE 8080
```

- whatap-virtual-X.Y.Z.jar：モニタリング対象(サンプルアプリケーション)
- whatap.agent-X.Y.Z.jar：WhaTapエージェント
- エージェントのファイル名がwhatap.agent-1.2.3.jarの場合、**X.Y.Z**の部分を**1.2.3**に変更してください。
- JVMオプション追加の詳細については、次の文書を参照してください。

アプリケーションの `whatap.server.host` 設定確認

一部コンテナ画像では `whatap.server.host` のように `.` が含まれる環境変数を直接設定することはできません。これにより、アプリケーションエージェントが正常に起動しない場合、whatap.confファイルを直接修正し `whatap.server.host` 値を設定する必要があります。アプリケーションの `whatap.server.host` 設定の確認方法は次のとおりです。

1. WhaTapアプリケーションコンテナに進入します。
2. 次のコマンドを実行して `whatap.server.host` 値が含まれていることを確認します。

```bash
$ env | grep whatap.server.host
```

値が出力されない場合当該コンテナの画像が `.` が含まれる環境変数の設定に対応していないことです。この場合、環境変数ではなく設定ファイルを活用する必要があります。whatap.confファイルを開いて次のように `whatap.server.host` 値を手動で設定します。

```bash
whatap.server.host={WhaTap収集サーバーIP}
```

Java 17バージョン以降の場合、reflectionに関する `--add-opens=java.base/java.lang=ALL-UNNAMED` オプションを追加します。

セキュリティキーの設定

SQL変数、HTTPクエリを検索、Thread停止機能を使用する場合は、セキュリティキーを設定してください。

- Javaエージェント2.2.2バージョン以降

  `$WHATAP_HOME` パスに security.conf ファイルを作成した後、次のように 6桁の英数字のパスワードを入力します。

  security.conf

  ```
  paramkey=ABCDEF # SQL変数およびHTTPクエリの検索
  threadkill=ABCDEF # Thread停止機能
  ```

- Javaエージェント2.2.2バージョン未満

  `$WHATAP_HOME` パスに paramkey.txt ファイルを作成した後、次のように 6桁の英数字のパスワードを入力します。

  paramkey.txt

  ```
  ABCDEF # SQL変数およびHTTPクエリの検索、スレッド停止機能
  ```

Javaエージェント 2.2.2 のバージョン未満から2.2.2バージョンにアップデートする場合、既存の paramkey.txt ファイルのキー値が seuciry.conf ファイルのキー値に自動的に適用されます。例えば、paramkey.txt に `FEDCBA` を使用して、バージョン2.2.2にアップデートした場合、security.conf ファイルには次のように適用されます。

```
paramkey=FEDCBA
threadkill=FEDCBA
```

paramkey.txt ファイルが存在しない場合、security.conf ファイルのキー値は、**WHATAP** というキー値で自動的に生成されます。

- SQLパラメータ情報の記録に関するエージェント設定は、次の文書を参照してください。
- HTTPパラメータ情報の記録に関するエージェント設定は、次の文書を参照してください。

コンテナ環境変数の設定

Dockerビルド後、Kubernetes環境内の**コンテナ環境変数**を設定してください。アプリケーション配布の yaml ファイルに次の内容を追加します。

```
env:
- name: NODE_IP
  valueFrom: {fieldRef: {fieldPath: status.hostIP}}
- name: NODE_NAME
  valueFrom: {fieldRef: {fieldPath: spec.nodeName}}
- name: POD_NAME
  valueFrom: {fieldRef: {fieldPath: metadata.name}}
- name: OKIND
  value: {YOUR_OKIND_NAME}
- name: license
  value: {licenseKey}
- name: whatap.server.host
  value: {proxyServer}
- name: whatap.micro.enabled
  value: "true"
```

次の例を参照してください。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: #DeploymentName
  labels:
    app: #AppLabel
spec:
  replicas: 3
  selector:
    matchLabels:
      app: #AppLabel
  template:
    metadata:
      labels:
        app: #AppLabel
```

```
spec:
  containers:
  - name: #ContainerName
    image: nginx
    ports:
    - containerPort: 80
    env:
    - name: NODE_IP
      valueFrom: {fieldRef: {fieldPath: "status.hostIP"}}
    - name: NODE_NAME
      valueFrom: {fieldRef: {fieldPath: "spec.nodeName"}}
    - name: POD_NAME
      valueFrom: {fieldRef: {fieldPath: "metadata.name "}}
    - name: OKIND
      value: #DeploymentName
    - name: license
      value: #licenseKey
    - name: whatap.server.host
      value: #proxyServer
    - name: whatap.micro.enabled
      value: "true"
```

環境変数の役割

- `NODE_IP` : 現在のPodがホストされているノード(Node)のIPアドレスを収集します。
- `NODE_NAME` : 現在Podが実行中のノードの名前を収集します。
- `POD_NAME` : 現在のPodの名前を収集します。
- `OKIND` (選択事項)：Podに該当するアプリケーションをグループ化します。デプロイメント(Deployment) 名に設定すると、該当するPodを一つにグループ化します。
- `license` ：エージェント認証を確認するためのキーです。
- `whatap.server.host` ：WhaTap収集サーバーのホストIPです。
- `whatap.micro.enabled` ：コンテナとの連動を活性化します。

エージェントのインストール確認

- エージェントが正常にインストールされたことを確認するには、ダッシュボード > アプリケーションサービスダッシュボードメニューに移動してください。
- ダウンロードしたファイルをインストールした後にダッシュボードメニューからエージェントが表示されない場合は、次の事項を確認してください。
  - コンテナで `ps -ef | grep whatap` コマンドを実行し、エージェントオプションを適用されていることを確認してください。
  - コンテナの /whatap/logs パスの内容を確認してください。エージェントログは logs/`{whatapの設定ファイル名}`-yyyymmdd.log 形式のファイル名で出力されます。

Docker Node.jsインストール

管理 > **エージェントインストール** > 下段の**アプリケーションエージェントのインストール** クリック > インストール方法から**Docker Node.js**タブを選択

Dockerコンテナベースで実行するNode.jsアプリケーションに、WhaTapモニタリングエージェントを適用し、コンテナイメージをパッケージングする手順を次のように案内します。

- EKS Fargateは、対応予定です。
- Node.jsアプリケーションの統合プロセスを理解しやすくするために、Git サンプルコードを提供します。次の文書を参照してください。

エージェントダウンロード

次のコマンドを使用して、WhaTapをインストールします。

```
npm install --save whatap
```

アップデートを継続するには、次のコマンドを実行します。

```
npm update whatap
```

エージェント設定

アプリケーションのメインモジュールの最初の行に、次のコードを追加します。

Example

```
const WhatapAgent = require('whatap').NodeAgent;
WhatapAgent.NodeAgent;
```

ECMAScript(ES) を活用する時、次のコードを追加します。

```
import WhatapAgent from 'whatap';
WhatapAgent.NodeAgent;
```

コンテナ環境変数の設定

Dockerビルド後、Kubernetes環境内の**コンテナ環境変数**を設定してください。アプリケーション配布のyamlファイルに次の内容を追加します。

```
env:
- name: NODE_IP
  valueFrom: {fieldRef: {fieldPath: status.hostIP}}
- name: NODE_NAME
  valueFrom: {fieldRef: {fieldPath: spec.nodeName}}
- name: POD_NAME
  valueFrom: {fieldRef: {fieldPath: metadata.name}}
- name: WHATAP_OKIND
  value: {YOUR_OKIND_NAME}
- name: WHATAP_LICENSE
  value: {licenseKey}
- name: WHATAP_SERVER_HOST
  value: {proxyServer}
- name: WHATAP_MICRO_ENABLED
  value: "true"
```

次の例を参照してください。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: #DeploymentName
  labels:
    app: #AppLabel
spec:
  selector:
    matchLabels:
      app: #AppLabel
  replicas: 3
  template:
```

```
metadata:
  labels:
    app: #AppLabel
spec:
  containers:
  - name: #ContainerName
    image: nginx
    ports:
    - containerPort: 80
    env:
    - name: NODE_IP
      valueFrom: {fieldRef: {fieldPath: status.hostIP}}
    - name: NODE_NAME
      valueFrom: {fieldRef: {fieldPath: spec.nodeName}}
    - name: POD_NAME
      valueFrom: {fieldRef: {fieldPath: metadata.name}}
    - name: WHATAP_OKIND
      value: #DeploymentName
    - name: WHATAP_LICENSE
      value: {licenseKey}
    - name: WHATAP_SERVER_HOST
      value: {proxyServer}
    - name: WHATAP_MICRO_ENABLED
      value: "true"
```

環境変数の役割

- `NODE_IP` : 現在のPodがホストされているノード(Node)のIPアドレスを収集します。
- `NODE_NAME` : 現在Podが実行中のノードの名前を収集します。
- `POD_NAME` : 現在のPodの名前を収集します。
- `WHATAP_OKIND` (選択事項): Podに該当するアプリケーションをグループ化します。 デプロイメント(Deployment) 名に設定すると、該当するPodを一つにグループ化します。
- `license` ：エージェント認証を確認するためのキーです。
- `whatap_server_host` ：WhaTap収集サーバーのホストIPです。
- `whatap_micro_enabled` ：コンテナとの連動を活性化します。

エージェントのインストール確認

- エージェントが正常にインストールされたことを確認するには、**ダッシュボード** > **アプリケーションサービスダッシュボード**メニューに移動してください。
- ダウンロードしたファイルをインストールした後に**ダッシュボード**メニューからエージェントが表示されない場合は、次の事項を確認してください。
  - コンテナで `ps -ef | grep whatap` コマンドを実行し、エージェントオプションを適用されていることを確認してください。
  - コンテナの /whatap/logs パスの内容を確認してください。

Docker Pythonインストール

管理 > エージェントインストール > 下段のアプリケーションインストール クリック > 設置案内の中で Docker Python タブを選択

Dockerコンテナベースで実行するPythonアプリケーションに、WhaTapモニタリングエージェントを適用し、コンテナイメージをパッケージングする手順を次のように案内します。Kubernetesアプリケーションをリリースするために、Dockerイメージが必要です。次の手順でwhatap-pythonパッケージがインストールされたDockerイメージを作成します。

- EKS Fargateは、対応予定です。
- Pythonアプリケーションの統合プロセスを理解しやすくするために、Git のサンプルコードを提供します。次の文書を参照してください。

エージェントダウンロード

PythonアプリケーションのDockerイメージをビルドする場合、whatap-pythonパッケージをインストールします。

```
RUN pip3 install --upgrade whatap-python
```

次のDockerfileの定義例を参照してください。

python ver 3.10

```
# python3.10をドッカー環境にインストールします。
FROM python:3.10

# 作業ディレクトリを/appに設定します。
WORKDIR /app

# 現在のディレクトリ内のすべてのファイルとフォルダーをコンテナ内の/appディレクトリにコピーします。
ADD . /app/

# Pythonでは、pipを利用して外部ライブラリがインストールできます。
```

```
# WhaTap Pythonエージェントをドッカーイメージのビルド時にインストールします。
RUN pip3 install --upgrade whatap-python
```

エージェントの設定及び実行

アプリケーションを実行する前に、Pythonエージェントの作業ディレクトリを設定し、デフォルトの設定ファイルを作成します。設定ファイル(entrypoint.sh)を通じてエージェントに認証情報やログの追跡有無などを設定します。

設定ファイル(entrypoint.sh)の例

entrypoint.sh

```
#!/bin/bash

# コンテナの作業ディレクトリをWhaTapパスに設定します。そのパスにエージェントログと設定ファイルを作成します。
export WHATAP_HOME=${PWD}

# 権限エラーが発生した場合は、次のコメントを削除してから進みます。
#chmod -R 777 $WHATAP_HOME

# 以下は、エージェントの構成に必須な設定値です。アプリケーションデプロイのyamlファイルを通じて設定します。
whatap-setting-config \
--host $whatap_server_host \
--license $license \
--app_name $app_name \
--app_process_name $app_process_name

# 次のコメントは、エージェントのグループ化やログ収集の活性化などの追加設定です。必要な場合に使用してください。

# エージェントのグループ化
#echo "whatap.okind=$whatap.okind" >> whatap.conf

# ログ収集の活性化
#echo "logsink_enabled=true" >> whatap.conf
#echo "logsink_trace_enabled=true" >> whatap.conf
```

```
#echo "trace_logging_enabled=true" >> whatap.conf


# 次のようにwhatap-start-agentをアプリケーション開始コマンドの前に追加して、エージェントを実行します。
whatap-start-agent uvicorn server:app --host 0.0.0.0 --port 8000
```

1. コンテナの作業ディレクトリを環境変数 `WHATAP_HOME` に設定してください。そのパスにエージェントログと設定ファイルを作成します。

   ```
   export WHATAP_HOME=${PWD}
   ```

2. `WHATAP_HOME` に設定したパスにwhatap.confファイルを作成するため、次のコマンドを実行してください。 `$` で表記された変数は、エージェント構成に必要な設定値です。アプリケーション配布のyamlファイルを通じて設定します。

   ```
   whatap-setting-config \
   --host $whatap_server_host \
   --license $license \
   --app_name $app_name \
   --app_process_name $app_process_name
   ```

3. アプリケーション開始コマンドの前に、次のように `whatap-start-agent` を追加してエージェントを実行してください。

```
whatap-start-agent uvicorn server:app --host 0.0.0.0 --port 8000
```

アクセス許可の問題が発生する場合、次のように `$WHATAP_HOME` にアクセス許可を付与します。

```
echo `sudo chmod -R 777 $WHATAP_HOME`
```

- yamlの作成例は、次のコンテナ環境変数の設定を参照してください。
- ライセンス(`license`)は、WhaTapのユーザー認証情報です。**外部に公開してはいけません。**

スクリプト実行の例

次のDockerfile定義の例を参照してください。entrypoint.shスクリプトを実行する例です。

```
FROM python:3.10
WORKDIR /app
ADD . /app/
RUN pip3 install --upgrade whatap-python

# entrypoint.shスクリプトをコンテナで実行する権限を付与します。
RUN chmod +x ./entrypoint.sh

# コンテナ作成時にentrypoint.sh スクリプトを実行します。
CMD ["./entrypoint.sh"]
```

追加設定

次の設定は、オプションとして必要な場合のみ使用してください。設定ファイル(entrypoint.sh)に次の追加設定の他にも、ログおよびトランザクション関連の設定が可能です。その他の追加設定に関しては、次の文書を参照してください。

- **エージェントグループ化**

```
echo "whatap.okind=$whatap.okind" >> whatap.conf
```

- **ログ収集の活性化**

```
echo "logsink_enabled=true" >> whatap.conf
echo "logsink_trace_enabled=true" >> whatap.conf
echo "trace_logging_enabled=true" >> whatap.conf
```

コンテナ環境変数の設定

Dockerビルド後、Kubernetes環境内の**コンテナ環境変数**を設定してください。アプリケーション配布のyamlファイルに次の内容を追加します。

```
  env:
  - name: license
    value: XXXXXXXXXXXXX-XXXXXXXXXXXXX-XXXXXXXXXXXXX
  - name: whatap_server_host
    value: XXX.XXX.XXX.XXX
  - name: app_name
    value: {YOUR_APP_NAME}
  - name: app_process_name
    value: {YOUR_PROCESS_NAME}
  - name: whatap.okind
    value: {YOUR_GROUP_NAME}
  - name: NODE_IP
    valueFrom: {fieldRef: {fieldPath: status.hostIP}}
  - name: NODE_NAME
    valueFrom: {fieldRef: {fieldPath: spec.nodeName}}
  - name: POD_NAME
    valueFrom: {fieldRef: {fieldPath: metadata.name}}
```

次の例を参照してください。

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: python-fastapi-deployment
spec:
 replicas: 3
 selector:
   matchLabels:
     app: python-fastapi-pod
 template:
   metadata:
     labels:
       app: python-fastapi-pod
 containers:
 - name: agent-python-fastapi
   image: whatap/agent-python-fastapi
   env:
   - name: license
     value: XXXXXXXXXXXXX-XXXXXXXXXXXXX-XXXXXXXXXXXXX
   - name: whatap_server_host
```

```
  value: XXX.XXX.XXX.XXX
- name: app_name
  value: "myapp-python-fastapi"
- name: app_process_name
  value: "uvicorn"
- name: NODE_IP
  valueFrom: {fieldRef: {fieldPath: status.hostIP}}
- name: NODE_NAME
  valueFrom: {fieldRef: {fieldPath: spec.nodeName}}
- name: POD_NAME
  valueFrom: {fieldRef: {fieldPath: metadata.name}}
```

環境変数の役割

- `license`：エージェント認証を確認するためのキーです。
- `whatap_server_host`：WhaTap収集サーバーのホストIPです。
- `app_name`：アプリケーションのエージェントを特定するための名前です。エージェント特定に対する詳細は、[次の文書]](https://docs.whatap.io/python/agent-name)を参考にしてください。
- `app_process_name`：CPU、Heap Memoryなど収集する対象プロセスを設定します。例としてuwsgi、gunicornなどがあります。
- `whatap.okind` (オプション)：Podに該当するアプリケーションをグループ化します。デプロイメント(Deployment)名で設定すると、該当するPodを一つにグループ化します。
- `NODE_IP`：現在のPodがホストされているノード(Node)のIPアドレスを収集します。
- `NODE_NAME`：現在Podが実行中のノードの名前を収集します。
- `POD_NAME`：現在のPodの名前を収集します。

エージェントのインストール確認

- エージェントが正常にインストールされたことを確認するには、ダッシュボード > アプリケーションサービスダッシュボードメニューに移動してください。
- コンテナで `ps -ef | grep whatap_python` コマンドを実行し、WhaTap Pythonサービスが正常に実行されているかを確認してください。

Docker PHPインストール

管理 > エージェントインストール > 下段のアプリケーションインストール クリック > 設置案内の中で Docker PHP タブを選択

Dockerコンテナベースで実行するPHPアプリケーションに、WhaTapモニタリングエージェントを適用し、コンテナイメージをパッケージングする手順を次のように案内します。

> ⓘ EKS Fargateは、対応予定です。

エージェントダウンロード

PHPアプリケーションのDockerイメージビルド時、whatap-phpパッケージをインストールしてください。

- x64 AArch64

x64

```
RUN wget https://s3.ap-northeast-2.amazonaws.com/repo.whatap.io/alpine/x86_64/whatap-php.tar.gz
RUN tar -xvzf whatap-php.tar.gz -C /
```

AArch64

```
RUN wget https://s3.ap-northeast-2.amazonaws.com/repo.whatap.io/alpine/aarch64/whatap-php.tar.gz
RUN tar -xvzf whatap-php.tar.gz -C /
```

エージェント設定

- 次のコマンドを使用してインストールスクリプトを設定してください。 WhaTapアクセスキー、サーバーIP情報を入力すると、PHP環境情報を自動的に収集します。

```
RUN (echo "[アクセスキー]"; echo "[サーバーIP]")|/usr/whatap/php/install.sh
```

- 次のような方法で、PHP環境情報を直接入力することもできます。

```
# RUN /usr/whatap/php/install.sh -l [アクセスキー] -s [サーバーIP] -v [PHPバージョンx.x] -i [php.iniのフルパス] -p [プロセス名] -z [thread safety]
RUN /usr/whatap/php/install.sh -l xxxxx -s 1.1.1.1/2.2.2.2 -v 7.0 -i /etc/php.ini -p httpd [-z]
```

> ⓘ **アクセスキー**がDocker Publicのイメージにオープンされないように注意してください。

コンテナ環境変数の設定

Dockerビルド後、Kubernetes環境内の**コンテナ環境変数**を設定します。 アプリケーションのリリースyamlファイルに次の内容を追加します。

```
env:
- name: NODE_IP
  valueFrom: {fieldRef: {fieldPath: status.hostIP}}
- name: NODE_NAME
  valueFrom: {fieldRef: {fieldPath: spec.nodeName}}
- name: POD_NAME
  valueFrom: {fieldRef: {fieldPath: metadata.name}}
- name: WHATAP_PHP_AGENT_PATH
  value: /usr/whatap/php/whatap_php_static
- name: WHATAP_CONTAINER_CONF_PATH
  value: /usr/whatap/php
```

次の例を参照してください。

```
apiVersion: v1
kind: Pod
metadata:
  name: # Pod Name
  labels: # Pod Labels
spec:
```

```
containers:
  env:
  - name: NODE_IP
    valueFrom: {fieldRef: {fieldPath: status.hostIP}}
  - name: NODE_NAME
    valueFrom: {fieldRef: {fieldPath: spec.nodeName}}
  - name: POD_NAME
    valueFrom: {fieldRef: {fieldPath: metadata.name}}
  - name: WHATAP_PHP_AGENT_PATH
    value: /usr/whatap/php/whatap_php_static
  - name: WHATAP_CONTAINER_CONF_PATH
    value: /usr/whatap/php
```

環境変数の役割

- `NODE_IP` : 現在のPodがホストされているノード(Node)のIPアドレスを収集します。
- `NODE_NAME` : 現在Podが実行中のノードの名前を収集します。
- `POD_NAME` : 現在のPodの名前を収集します。
- `WHATAP_PHP_AGENT_PATH` : PHPエージェントの実行ファイルパスを指定します。
- `WHATAP_CONTAINER_CONF_PATH` : コンテナIDが保存されるパスを指定します。

エージェント実行

- **Command** **Supervisor**

アプリケーション開始コマンドの前にwhatap-php開始コマンドを追加します。

```
sh -c "/usr/whatap/php/whatap-php start && [アプリケーション開始コマンド]"
```

Supervisorでアプリケーションを起動する場合は、次のコードを参考してwhatap-phpサービスを追加します。

```
RUN echo "[program:whatap-php]" >> /etc/supervisor/conf.d/whatap.conf
RUN echo "command = sh -c '/usr/whatap/php/whatap-php start'" >> /etc/supervisor/conf.d/whatap.conf
RUN echo "user = root" >> /etc/supervisor/conf.d/whatap.conf
```

```
RUN echo "startsecs = 0" >> /etc/supervisor/conf.d/whatap.conf
RUN echo "autostart = true" >> /etc/supervisor/conf.d/whatap.conf
RUN echo "autorestart = false" >> /etc/supervisor/conf.d/whatap.conf
RUN echo "stdout_logfile = /dev/stdout" >> /etc/supervisor/conf.d/whatap.conf
RUN echo "stdout_logfile_maxbytes=0" >> /etc/supervisor/conf.d/whatap.conf
RUN echo "stderr_logfile = /dev/stderr" >> /etc/supervisor/conf.d/whatap.conf
RUN echo "stderr_logfile_maxbytes=0" >> /etc/supervisor/conf.d/whatap.conf
```

次のコマンドを実行して、WhaTapサービスが正常に実行されているかを確認してください。 アプリケーションサーバーが実行されると、エージェントがモニタリング情報を収集します。

```
ps -ef | grep whatap_php
```

エージェントのインストール確認

エージェントが正常にインストールされたかを確認するには、**ダッシュボード** > **ダッシュボード**メニューへ移動してください。

ダウンロードしたファイルをインストールした後に**ダッシュボード**メニューからエージェントが表示されない場合は、次の事項を確認してください。

- コンテナで `ps -ef | grep whatap` コマンドを実行し、エージェントオプションを適用されていることを確認してください。
- 該当コンテナ内の_/usr/whatap/php/logs_経路の内容を確認してください。 エージェントログは、logs/whatap-`{boot、またはinstall}`-yyyymmdd.log 形式のファイル名として出力されます。

Docker Goインストール

管理 > エージェントインストール > 下段のアプリケーションインストール クリック > 設置案内の中で Docker Go タブを選択

DockerコンテナベースのGoアプリケーションにWhaTapモニタリングエージェントをインストールする方法を案内します。次の2つの方法から選択してください。

- **クイックインストール方法**: Dockerfileの作成からコンテナの実行、モニタリングデータの確認までの全インストールプロセス
- **デプロイ環境別設定**: アーキテクチャ(x64/ARM64)、Kubernetes、Docker Compose環境別の追加設定方法

> EKS Fargateはサポート予定です。

> インストールを開始する前に、次の情報を準備してください。
>
> - WhaTapプロジェクト**アクセスキー**([プロジェクトメニュー] > [管理] > [エージェントインストール]で確認)
> - WhaTap**収集サーバーIPアドレス**

クイックインストール方法

3つのファイルだけでWhaTapモニタリングが適用されたDockerイメージをビルドできます。

ステップ1. 必須ファイルの準備

プロジェクトルートに次の3つのファイルを作成してください。

- **1. Dockerfile** **2. entrypoint.sh** **3. main.go (例)**

```
# ==========================================
# Build stage - whatap-go-instビルド
# ==========================================
```

```
FROM golang:1.21-alpine AS builder

WORKDIR /app

# whatap-go-instインストール
#RUN go install github.com/whatap/go-api-inst/cmd/whatap-go-inst@latest
RUN wget -qO- https://github.com/whatap/go-api-inst/releases/latest/download/whatap-go-inst_linux_amd64.tar.gz | tar xz
-C /usr/local/bin/

# ソースコピー
COPY go.mod go.sum ./
RUN go mod download
COPY . .

# whatap-go-instビルド
RUN whatap-go-inst go build -o /app/main .

# ==========================================
# Runtime stage
# ==========================================
FROM alpine:latest

WORKDIR /app

# WhaTapデータ中継エージェントインストール
RUN apk add --no-cache wget
RUN wget https://s3.ap-northeast-2.amazonaws.com/repo.whatap.io/alpine/x86_64/whatap-agent.tar.gz
RUN tar -xvzf whatap-agent.tar.gz -C /
RUN rm whatap-agent.tar.gz

# アプリケーションコピー
COPY --from=builder /app/main .

# 起動スクリプト
COPY entrypoint.sh /app/
RUN chmod +x /app/entrypoint.sh

ENTRYPOINT ["/app/entrypoint.sh"]
```

```
#!/bin/sh
```

```
# whatap.conf生成 (環境変数から)
mkdir -p /app/whatap_home
cat > /app/whatap_home/whatap.conf << EOF
license=${WHATAP_LICENSE}
whatap.server.host=${WHATAP_SERVER_HOST}
app_name=${WHATAP_APP_NAME:-myapp}
EOF

export WHATAP_HOME=/app/whatap_home

# データ中継エージェント起動後、アプリケーション実行
/usr/whatap/agent/whatap-agent start
exec ./main
```

```
package main

import "github.com/gin-gonic/gin"

func main() {
   r := gin.Default()
   r.GET("/", func(c *gin.Context) {
      c.JSON(200, gin.H{"message": "Hello WhaTap"})
   })
   r.Run(":8080")
}
```

ステップ2. イメージビルド

```
docker build -t my-go-app .
```

ステップ3. コンテナ実行

環境変数でWhaTap設定を渡します。

```
docker run -d \
  -e WHATAP_LICENSE={アクセスキー} \
```

```
-e WHATAP_SERVER_HOST={収集サーバーIP} \
-e WHATAP_APP_NAME=myapp \
-p 8080:8080 \
my-go-app
```

ステップ4. モニタリング確認

1. アプリケーションにトラフィック発生: `curl http://localhost:8080`
2. WhaTapモニタリングサービスでデータ確認

> トラブルシューティングセクションを参照してください。

デプロイ環境別設定

アーキテクチャ別Dockerfile設定

前述のクイックスタート例は**x64 (AMD64)**を基準としています。ARM64環境ではエージェントダウンロードURLのみ変更してください。

WhaTapエージェントのアーキテクチャ別URL

アーキテクチャ	エージェントダウンロードURL
x64 (AMD64)	`https://s3.ap-northeast-2.amazonaws.com/repo.whatap.io/alpine/x86_64/whatap-agent.tar.gz`
ARM64 (AArch64)	`https://s3.ap-northeast-2.amazonaws.com/repo.whatap.io/alpine/aarch64/whatap-agent.tar.gz`

ARM64 Dockerfile修正例

Dockerfileのエージェントインストール部分のみ変更:

```
# Runtime stageで変更
RUN apk add --no-cache wget && \
  wget https://s3.ap-northeast-2.amazonaws.com/repo.whatap.io/alpine/aarch64/whatap-agent.tar.gz && \
  tar -xvzf whatap-agent.tar.gz -C / && rm whatap-agent.tar.gz
```

ARM64完全なDockerfileを表示

```
FROM golang:1.21-alpine AS builder

WORKDIR /app

# whatap-go-instインストール
#RUN go install github.com/whatap/go-api-inst/cmd/whatap-go-inst@latest
RUN wget -qO- https://github.com/whatap/go-api-inst/releases/latest/download/whatap-go-
inst_linux_amd64.tar.gz | tar xz -C /usr/local/bin/

COPY go.mod go.sum ./
RUN go mod download
COPY . .

RUN whatap-go-inst go build -o /app/main .

FROM alpine:latest

WORKDIR /app

RUN apk add --no-cache wget && \
  wget https://s3.ap-northeast-2.amazonaws.com/repo.whatap.io/alpine/aarch64/whatap-agent.tar.gz && \
  tar -xvzf whatap-agent.tar.gz -C / && rm whatap-agent.tar.gz

COPY --from=builder /app/main .
COPY entrypoint.sh /app/
RUN chmod +x /app/entrypoint.sh

ENTRYPOINT ["/app/entrypoint.sh"]
```

Kubernetesデプロイ

Dockerイメージをビルドした後、Kubernetesクラスターにデプロイできます。

ステップ1: Dockerイメージをレジストリにプッシュ

```
# イメージタグ指定
docker tag my-go-app:latest your-registry/my-go-app:v1.0

# レジストリプッシュ
docker push your-registry/my-go-app:v1.0
```

ステップ2: WhaTap設定Secret作成

```
kubectl create secret generic whatap-secret \
 --from-literal=license={アクセスキー}
```

ステップ3: Deploymentデプロイ

- deployment.yaml service.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: my-go-app
spec:
 replicas: 3
 selector:
  matchLabels:
   app: my-go-app
 template:
  metadata:
   labels:
    app: my-go-app
  spec:
   containers:
   - name: my-go-app
```

```
    image: your-registry/my-go-app:v1.0
    ports:
    - containerPort: 8080
    env:
    - name: WHATAP_LICENSE
      valueFrom:
        secretKeyRef:
          name: whatap-secret
          key: license
    - name: WHATAP_SERVER_HOST
      value: "{収集サーバーIP}"
    - name: WHATAP_APP_NAME
      value: "my-go-app"
    # Kubernetesメタデータ (オプション)
    - name: NODE_IP
      valueFrom:
        fieldRef:
          fieldPath: status.hostIP
    - name: NODE_NAME
      valueFrom:
        fieldRef:
          fieldPath: spec.nodeName
    - name: POD_NAME
      valueFrom:
        fieldRef:
          fieldPath: metadata.name
```

```
apiVersion: v1
kind: Service
metadata:
  name: my-go-app
spec:
  selector:
    app: my-go-app
  ports:
  - port: 80
    targetPort: 8080
  type: LoadBalancer
```

```
kubectl apply -f deployment.yaml
```

```
kubectl apply -f service.yaml
```

環境変数の説明

必須環境変数

環境変数	説明	例
WHATAP_LICENSE	WhaTapプロジェクトアクセスキー	x4t2r...
WHATAP_SERVER_HOST	WhaTap収集サーバーIP	13.124.11.223
WHATAP_APP_NAME	アプリケーション名 (オプション)	my-go-app

Kubernetesメタデータ (オプション)

ポッドの実行位置情報をWhaTapに渡すには追加してください:

環境変数	説明
NODE_IP	ポッドが実行されているノードのIP
NODE_NAME	ポッドが実行されているノードの名前
POD_NAME	ポッドの名前

Docker Compose

docker-compose.yml

```
version: '3.8'

services:
  my-go-app:
    build: .
    ports:
```

```
    - "8080:8080"
  environment:
    - WHATAP_LICENSE={アクセスキー}
    - WHATAP_SERVER_HOST={収集サーバーIP}
    - WHATAP_APP_NAME=my-go-app
  restart: unless-stopped
```

```
docker-compose up -d
```

トラブルシューティング

モニタリングデータが表示されない場合

ステップ1: 環境変数の確認

コンテナ内部で環境変数が正しく設定されているか確認:

```
docker exec -it <container_id> sh
echo $WHATAP_LICENSE
echo $WHATAP_SERVER_HOST
cat /app/whatap_home/whatap.conf
```

ステップ2: エージェント実行の確認

```
# エージェントプロセス確認
docker exec -it <container_id> ps -ef | grep whatap

# エージェントログ確認
docker exec -it <container_id> cat /app/whatap_home/logs/whatap-*.log
```

正常出力例:

```
root  12  1  0 Jan21 ?  00:00:03 /usr/whatap/agent/whatap-agent
```

ステップ3: ネットワークの確認

WhaTap収集サーバー(6600ポート)へのアウトバウンド接続が可能か確認:

```
docker exec -it <container_id> nc -zv {収集サーバーIP} 6600
```

計測が適用されない場合

計測詳細ログの確認

```
# ビルド時のデバッグ出力
GO_API_AST_DEBUG=1 whatap-go-inst go build ./...
```

一般的な問題と解決方法

症状	原因	解決方法
データ未受信	環境変数未設定	`WHATAP_LICENSE`、`WHATAP_SERVER_HOST`確認
エージェント起動失敗	ネットワーク遮断	6600ポートアウトバウンド許可
計測未適用	初期化漏れ	Dockerfileに`whatap-go-inst init`追加確認
ARM64エラー	誤ったバイナリ	アーキテクチャ別エージェントURL確認

追加サポート

上記の方法で解決しない場合:

1. WhaTapサポートセンターにお問い合わせ
2. 次の情報を提供:
    - Dockerfile全体の内容

- エージェントログ (`/app/whatap_home/logs/`)
- `docker logs <container_id>` の出力

インストール問題の解決

Helmチャート基準のインストール問題の解決

名前が `whatap` である他のHelmリポジトリがある場合

"whatap" already exists with the same configuration, skipping

すでに同じ名前の他の `whatap` リポジトリが存在する場合は、次のコマンドを使用して既存の `whatap` リポジトリを削除してからインストールを進めてください。

```
helm repo remove whatap
```

名前が `whatap-monitoring` である他のネームスペースがクラスターにある場合

Error: INSTALLATION FAILED: Unable to continue with install: Namespace "whatap-monitoring" in namespace "" exists and cannot be imported into the current release

クラスターに `whatap-monitoring` ネームスペースが存在する場合、ユーザーがyamlファイルベースですでにWhaTap Kubernetesエージェントをインストールしているためです。 この場合、次のコマンドにより、既存の `whatap-monitoring` ネームスペースと関連リソースを削除します。

```
kubectl delete namespace whatap-monitoring
kubectl delete clusterrolebinding whatap
kubectl delete clusterrole whatap
```

`clusterrole` 、 `clusterrolebinding` リソースがすでにクラスターにある場合

Error: INSTALLATION FAILED: Unable to continue with install: ClusterRole "whatap" in namespace "" exists and cannot be imported into the current release

クラスターへ `clusterrole` 、 `clusterrolebinding` リソースがすでに存在する場合は、次のコマンドでそのリソースを削除します。

```
kubectl delete clusterrolebinding whatap
kubectl delete clusterrole whatap
```

`whatap-monitoring` ネームスペースが使用できない場合

基本的にWhaTap Kubernetesエージェントは `whatap-monitoring` ネームスペースにインストールされます。 ただし、該当ネームスペースが使用できない場合、またはユーザーが特定のネームスペースを指定する必要がある場合は、次の手順を参考にして、必要なネームスペースにエージェントをインストールします。

事前準備

- **チャートバージョン**: whatap/kube 1.7.11以降

  次のコマンドでチャートのバージョンを確認します。

  ```
  helm search repo whatap/kube --versions
  ```

  1.7.11バージョン未満の場合は、次のコマンドでチャートのバージョンをアップデートします。

  ```
  helm repo update
  ```

- **Helmバージョン**: Helm 3以降

インストール手順

使用するネームスペース名を `<namespace>` に入力し、次のコマンドを実行してエージェントをインストールします。

```
helm install whatap-kube-agent whatap/kube --create-namespace --namespace <namespace> --set
namespaceOverrideEnabled=true --wait -f values.yaml
```

アップデートと削除

- **アップデート**

次のコマンドでエージェントをアップデートします。

```
helm update whatap-kube-agent whatap/kube --namespace <namespace> --wait
```

- **削除**

  次のコマンドを使用してエージェントを削除します。

  ```
  helm uninstall whatap-kube-agent whatap/kube --namespace <namespace>
  ```

エージェント起動時に発生する問題の解決

旧型YAMLファイルを通じてWhaTap Kubernetesモニタリングをインストールした場合

> **Unrecognized VM option 'UseCGroupMemoryLimitForHeap'**
> **Error: Could not create the Java Virtual Machine.**
> **Error: A fatal exception has occurred. Program will exit.**

2022年8月24日以前にダウンロードしたインストールyamlを通じてWhaTapのKubernetesモニタリングをインストールした場合、ノードエージェント(whatap-node-agent) 起動時にエラーが発生し、起動に失敗する場合があります。

原因は以下のとおりです。 2022年8月24日以前にダウンロードしたインストールyamlファイル内 `UseCGroupMemoryLimitForHeap` JVMオプションを使用したノードエージェントの実行スクリプトが含まれています。

WhaTapノードエージェント**1.2.0以降**(2022年11月21日リリース) バージョンからエージェント内部で使用するJDKがアップグレードされおり、アップグレードされたJDKバージョンは、これ以上 `UseCGroupMemoryLimitForHeap` オプションを提供していません。 **2022年8月24日以前**にダウンロードしたyamlファイルにWhaTap Kubernetesモニタリングをインストールした状態でノードエージェント1.2.0以降のバージョンを起動する場合、 `Unrecognized VM option 'UseCGroupMemoryLimitForHeap'` メッセージが発生し、起動に失敗します。

解決方法

この問題を解決するために、次のようにKubernetesモニタリングを再インストールします。

1. **管理** > **エージェントインストール**画面で最新のYAMLファイルをダウンロードします。
2. ユーザーKubernetes環境でダウンロードしたyamlファイルを使用してエージェントをインストールしてください。
3. ユーザーKubernetes環境でインストール中にエラーの出力有無を確認してください。
4. WhaTapモニタリングエージェントPodがRunning状態であるかを確認してください。

   > Kubernetes環境及び**ワークロード** > **Podリスト**メニューで確認します。

5. **管理** > **Kubernetesエージェントリスト**メニューで、各モニタリングエージェントで発生するログを確認します。

DockerHubイメージpull limitでエージェントが正常にインストールされない場合、

複数のノードで同時に画像をpullする場合、以下のようなエラーが発生し、インストールに失敗することがあります。
問題が発生した場合、以下のコマンドでyamlにイメージを変更して再インストールする必要があります。

エラー照会

```
kubectl describe pod/<WhaTap k8s エージェントPod名> -n whatap-monitoring
```

照会結果

```
> > Failed to pull image "whatap/kube_mon:latest": rpc error: code = Unknown desc = toomanyrequests: You have reached your pull rate limit. Please try again later or log in to increase your rate limit.
```

この場合、`yaml`ファイル内のDockerHubイメージをpublic.ecr.awsレポに変更した後、再インストールが必要です。

- エージェントインストール画面でデフォルトインストール用のyamlをダウンロードした場合、以下のコマンドを使用して画像を変更できます。 コマンド実行時、yamlにある既存のdocker Hubイメージがpublic.ecr.awsリポジトリ

イメージに置き換えられます。

```
sed -i 's|image: "whatap/kube_mon|image: "public.ecr.aws/whatap/kube_mon|g' kube_agent.yaml
```

アプリケーション連動問題の解決

コンテナマップにJavaモニタリングデータが表示されない場合

Kubernetes環境でCgroupsの設定や管理方法によっては、特定の状況でコンテナIDを確保できない問題が発生する可能性があります。 WhaTapは、このような状況に対応するために、Kubernetesモニタリングのコンテナとアプリケーション間のマッピングメカニズムを変更しました。 当該変更事項の場合は、WhaTap**Kubernetesエージェント1.8.8**バージョンと**Javaエージェント2.2.42**バージョンから適用され、Cgroupsの影響でコンテナIDが漏れる問題を解決します。

解決方法

エージェントのアップデート

Kubernetes環境で発生するコンテナIDの識別問題を解決するためには、WhaTapのKubernetesエージェントとAPMが互いにやり取りするデータを正確に識別できるよう、サポートする最新バージョンを使用する必要があります。 **クバネティス·エージェント**は**1.8.8バージョン以降**をを設置し、**Javaエージェント**は**2.2.42バージョン以降**をインストールします。

> ⓘ 現在はJavaエージェントのみサポートしており、他の言語のAPMは今後サポートする予定です。

追加環境変数の設定

エージェントバージョンのアップデート以降も、**コンテナマップ**にJavaアプリケーション情報が表示されない場合、ユーザー環境変数を追加して問題を解決できます。 この設定は、Java APMがインストールされたコンテナマニフェスト(yaml)の環境変数に、当該コンテナ内のJavaエージェントの絶対パスを表す `WHATAP_JAVA_AGENT_PATH` オプションを追加します。

次はJava APMがインストールされたコンテナ環境変数に当該オプションを追加した例です。

```
env:
- name: NODE_IP
  valueFrom: {fieldRef: {fieldPath: status.hostIP}}
- name: NODE_NAME
  valueFrom: {fieldRef: {fieldPath: spec.nodeName}}
- name: POD_NAME
  valueFrom: {fieldRef: {fieldPath: metadata.name}}
- name: OKIND
  value: {YOUR_OKIND_NAME}
- name: license
  value: <Project Access Key>
- name: whatap.server.host
  value: <収集サーバーのIPアドレス>
- name: whatap.micro.enabled
  value: "true"
- name: WHATAP_JAVA_AGENT_PATH
  value: "/whatap/whatap.agent-2.2.35.jar"
```

コンテナとAPM間の正常マッピングの確認方法

コンテナとアプリケーションが正常にマップされていることを確認する方法は、次の通りです。

- **ダッシュボード** > **コンテナマップ**メニューで**コンテナマップ**ブロック内に白いマイクロコンテナが存在する場合、正常にマッピングされた状態です。
- **分析** > **メトリクス照会**メニューで `app_counter` カテゴリの `container` フィールドにアプリケーションが実行中のコンテナIDが記録されている場合は、正常にマッピングされた状態です。

Kubernetesエージェントオプション設定

- WhaTap Kubernetesエージェントの配布時、コンテナの `env` フィールドを通して設定できます。
- コンテナ内のwhatap.confファイルを修正して設定できます。
- whatap.confファイル内のオプションがコンテナ `env` フィールドに設定されたオプションよりも高い優先順位を持ちます。
- whatap.confまたはコンテナ `env` に設定された値がない場合は、既定値が使用されます。

> **コンテナ内のアプリケーションエージェントのオプション設定**
>
> コンテナ内のJava、Python、Node.jsなどのアプリケーションをモニタリングするために必要な設定をwhatap.confファイルに作成します。 whatap.confファイルを使用して設定できるオプションの詳細については、次のリンクを参照してください。
>
> - Java - whatap.confの設定
> - Node.js - whatap.confの設定
> - Python - whatap.confの設定

コンテナ別のオプション設定

whatap-node-helper

- **collect_volume_detail_enabled** bool

  既定値 `true`

  ノードエージェントですべてのコンテナに対する `exec df` コマンドを使用して、コンテナのボリューム情報の照会を活性化します。 ノードに実行中のすべてのコンテナのストレージ使用量およびボリューム情報を収集する機能を活性化します。

- **use_cached_mount_info_enabled** bool

  デフォルト値 `false`

  マウント情報(mountinfo)キャッシュの使用有無を設定します。 `true` に設定すると、ノードヘルパーが最初のディ

スクスキャン時点のmountinfoをキャッシュして再利用します。この場合、エージェント起動後に新しくマウントされたPod PVCボリュームが永続的に欠落する可能性があるため、デフォルト値は `false` です。mountinfoの再読み込みコストは収集周期あたり軽微です。

- **inject_container_id_to_apm_agent_enabled** bool

  既定値 `true`

  WhaTap APMコンテナに `exec` を利用したcontainer IDの注入を活性化します。 WhaTap APMで識別されるコンテナのIDを自動的に抽出します。 これをAPMツールに注入し、性能モニタリングデータとコンテナを接続できます。

- **debug** bool

  既定値 `false`

  デバッグモードの活性化有無を設定します。 デバッグモードの活性化時のログレベルは `DEBUG` に、そうでない場合は `INFO` に設定します。

- **kube_config_path** string

  既定値 `~/.kube/config`

  Kubernetes設定ファイルのパスを指定します。 未設定時は、_~/.kube/config_を使います。

- **log_sys_out** bool

  既定値 `true`

  ログシステムの出力を設定します。 オプション値を `true` に設定する時、標準出力(stdOut)とファイルに同時にログを残します。 変更時、ロギングシステムにすぐに反映されます。

- **collect_kube_node_process_metric_enabled** bool

  既定値 `true`

  ノードでKubernetes関連プロセスのモニタリングを有効または無効にするフラグです。 `true` に設定すると、モニタリングが有効になり、 `false` に設定すると、モニタリングが無効になります。

- **collect_kube_node_process_metric_target_list** list

  既定値

  ```
  "kubelet,containerd-shim,containerd,docker,dockerd,crio,metrics-server, coredns,aws-k8s-agent,kube-proxy"
  ```

  基本的にKubernetes関連プロセスをモニタリングします。 ユーザーが特定のプロセス一覧を修正し、任意にモニタリングするプロセスを設定できます。

WhaTapでデフォルト設定されたモニタリング対象プロセスのリスト

- **kubelet**

  各ノードで実行されるコントローラで、コンテナの状態を継続的にモニタリングし、Kubernetesマスターと通信してワークロードが正しくスケジューリングされ、実行されるようにします。 コンテナのステータスチェックとノードリソースの管理を担当します。

- **containerd**

  コンテナ実行のための高性能コンテナランタイムです。 イメージ管理、コンテナの作成と実行、ネットワーキングなどを処理します。

- **dockerd**

  Dockerコンテナの生成を管理する主要なプロセスです。 このプロセスには、コンテナ画像のビルド、展開、実行、ネットワーク、およびデータ管理が含まれます。

- **crio**

  Kubernetesために設計された軽量化されたコンテナランタイムで、CRI(Container Runtime Interface)を実装して、コンテナワークロードを効率的に管理します。

- **coredns**

  Kubernetesクラスター内部でDNSサービスを提供するプロセスで、クラスター内サービスのネームリゾリューションを処理します。 また、サービスディスカバリーおよびロードバランシング機能を実行し、クラスター内のサービス間の通信を円滑にサポートします。

- **aws-k8s-agent**

  AWS環境でKubernetesネットワークインターフェース(VPC CNI)を管理するエージェントです。 このプロセスは、Pod間通信のためにAWSネットワークリソース(ENI)を割り当てて解除する役割をします。 問題が発生すると、ネットワーク接続障害やリソース割り当てエラーが発生する可能性があります。

- **kube-proxy**

  Kubernetesクラスター内でネットワークトラフィックをルーティングするプロセスです。 このプロセスは各ノードで実行され、サービスとPod間の通信を可能にします。 また、iptablesあるいはIPVSなどのLinuxネットワーキング技術を使用し、クラスター内の要求を正しいPodに伝えます。 これを通じ

て、Kubernetesクラスター内のサービス ディスカバリーとロードバランシングをサポートします。

- **kube-apiserver**

  Kubernetes APIサーバーで、クラスター内のすべてのリクエストを処理し、リソースの状態を管理する中央コンポーネントです。 認証、権限付与、アドミッションコントロールなど、クラスターの全般的な要求処理を担当します。

- **etcd**

  Kubernetesクラスター状態情報を保存する分散キー-値データベースです。 クラスターの状態保存と復旧に重要な役割を果たします。

- **kube-controller**

  複数のコントローラーを管理および実行するプロセスで、クラスターリソースの状態を維持し、自動化作業を実行します。 ノードとPodの状態管理、レプリケーションの維持など、様々な役割を果たします。

- **kube-scheduler**

  クラスターでは、Podを適切なノードに配置する役割を果たします。 Podの要件と各ノードの状態を考慮し、最適なノードを選択します。

whatap-node-agent

- **shutdown** bool

  既定値 `false`

  収集サーバーとの接続を強制的に停止します。 通信の停止を意味します。

- **license** string

  プロジェクトに発行されたライセンスを使用してエージェントを識別します。

- **whatap.server.host** string

  既定値 `13.124.11.223` あるいは `13.209.172.35`

  収集サーバーのIPを設定します。 `delimiter` の場合、 `/:` を使います。

- **whatap.server.port** number

  既定値 `6600`

  収集サーバーのデータ受信ポートを設定します。

- **ONODE_NAME** string

  既定値 `NODE_NAME` or `NODE_IP` 環境変数値

  エージェントノード識別のための名前を設定します。

- **tcp_connection_timeout** number

  既定値 `3000`

  whatap-master-agentと収集サーバーのソケット接続時のタイムアウト時間を設定します。

- **tcp_so_timeout** number

  既定値 `60000`

  ソケット接続後のデータの送信・受信タイムアウト時間を設定します。

- **debug** bool

  既定値 `false`

  デバッグモードの活性化有無を設定します。

- **gpuEnabled** bool

  既定値 `false`

  GPUモニタリングの活性化有無を設定します。

- **count_interval** number

  既定値 `5000`

  メトリクス(`kube_pod` 、 `container`)を収集サーバーに送信するサイクルを設定します。 既定値の単位はmsです。

  > ⓘ このオプションは、画面上のデータ更新サイクルに影響を与える可能性があります。 詳細については、以下の[メトリクス別の影響を受ける画面コンポーネント]](#k8s-affected-comp-by-metrics)を参考にしてください。

- **collect_interval** number

既定値 5000

エージェントがapiserverにリクエストしてメトリクス(kube_pod 、 container)を収集するサイクルを設定します。

> このオプションは、画面上のデータ更新サイクルに影響を与える可能性があります。 詳細については、以下の[メトリクス別の影響を受ける画面コンポーネント]](#k8s-affected-comp-by-metrics)を参考にしてください。

- **collect_apiserver_metric_by_leader** bool

  既定値 false

  このオプションが活性化されている場合、ノードエージェントはメトリクスの収集をマスターエージェントに委任します。 whatap-master-agentの collect_apiserver_metric_by_leader オプションの有効化の先行が必要です。メトリクス収集の委任により、既存のwhatap-node-agentが発生させたapiserverの負荷を低くすることができます。

- **kubelet_root_dir** string

  既定値 /var/lib/kubelet

  kubeletルートディレクトリのパスを設定します。ノードエージェントはこのパスを基準にコンテナログパスのシンボリックリンクを解析します。Huawei Cloud Container Engine（CCE）のように、kubeletルートの位置がデフォルトと異なる環境で使用します。端的に言えば、標準パスに従わない環境でエージェントがログファイルを見つけられるよう案内するオプションです。

- **container_log_path** string

  既定値なし

  コンテナログディレクトリのパスを直接指定します。未設定の場合、ノードエージェントは kubelet_root_dir を基準としたデフォルトパスと /var/log/pods などの標準パスを順に探索します。CCE環境のようにログパスが非標準で自動探索が難しい場合にのみ、このオプションで明示的に指定します。

> kubelet_root_dir と container_log_path は、コンテナランタイムやクラウドプロバイダーが標準と異なるログパスを使用する場合にのみ設定します。一般的なEKS・GKE・AKS・オンプレミス環境ではデフォルト値のままで正常に動作します。

- **collect_conntrack_enabled** bool

既定値 `true`

ノードのconntrack使用状況の収集を有効にするかどうかを設定します。有効にすると、`kube_node_perf` カテゴリーに `conntrack_count` 、 `conntrack_max` 、 `conntrack_pct` フィールドを送信します。 `nf_conntrack` モジュールがロードされていないノードはフィールドを送信しません。ノードエージェント1.9.18バージョン以上でサポートします。

- **collect_pod_emptydir_enabled** bool

  既定値 `true`

  emptyDirボリューム使用量の収集を有効にするかどうかを設定します。有効にすると、ノードエージェントがkubeletのボリュームディレクトリを60秒周期で照会し、 `kube_pod_volume` カテゴリーに送信します。 `false` に設定すると、既存の `df` ベースの収集経路がemptyDirボリュームを担当します。ノードエージェント1.9.22バージョン以上でサポートします。

- **pod_emptydir_max_files** number

  既定値 `20000`

  1つのemptyDirボリュームで照会する最大ファイル数を設定します。上限に達したボリュームは `truncated` フィールドを `1` で送信し、値が実際より小さいことを示します。

- **pod_emptydir_max_volumes** number

  既定値 `300`

  1つのノードで照会する最大emptyDirボリューム数を設定します。

- **net_send_buffer_enabled** bool

  既定値 `true`

  収集サーバーとのネットワークが切断された区間で、送信キューを超過したデータの保管および再送信の可否を設定します。有効にすると、送信キューがいっぱいになった場合にデータを破棄する代わりに保管バッファへ移し、再接続後に古いデータから順に送信します。ノードエージェント1.9.17バージョン以上でサポートします。

- **net_send_buffer_minutes** number

  既定値 `15`

  保管バッファのデータ保管時間を設定します。既定値の単位は分です。保管時間を超えたデータは古いものから破棄します。

- **net_send_buffer_max_mb** number

既定値 64

保管バッファが使用する最大メモリを設定します。既定値の単位はMBです。上限を超えると古いデータから破棄し、このとき破棄されたデータのみを pack_dropped として集計します。

whatap-control-plane-helper

- **port** string

  既定値 9496

  whatap-control-plane-helperのウェブサーバーポートを設定します。

- **cycle** number

  既定値 5

  control-plane-helperが指標をスクレイピングするサイクルを設定します。 既定値の単位はsです。

- **collect_control_plane_monitoring_enabled** bool

  既定値 false

  control-plane-monitoring機能の活性化有無を設定します。

- **kube_config_path** string

  既定値 ~.kube/config

  Kubernetes設定ファイルのパスを指定します。 未設定時_~/.kube/config_を使います。

- **kube_master_url** string

  既定値 ~/.kube/config

  スクラッピングのためにリクエストするcontrol-planeのurlを設定します。 未設定時_~/.kube/config_を使います。

- **debug** bool

  既定値 false

  whatap-control-plane-helperのデバッグ(ロギング)の活性化有無を設定します。

- **kube_client_tls_verify** bool

  既定値 true

オプション値を `false` に設定すると、TLS認証を強制することなく、認証手順を省略できます。

whatap-master-agent

- **shutdown** bool

  既定値 `false`

  収集サーバーとの接続を強制的に停止します。 通信の停止を意味します。

- **license** string

  プロジェクトに発行されたライセンスを使用してエージェントを識別します。

- **whatap.server.host** string

  既定値 `13.124.11.223` あるいは `13.209.172.35`

  収集サーバーのIPを設定します。 `delimiter` の場合、 `/:` を使います。

- **whatap.server.port** number

  既定値 `6600`

  収集サーバーのデータ受信ポートを設定します。

- **tcp_connection_timeout** number

  既定値 `3000`

  whatap-master-agentと収集サーバーのソケット接続時のタイムアウト時間を設定します。

- **tcp_so_timeout** number

  既定値 `60000`

  whatap-master-agentと収集サーバーのソケット通信時のタイムアウト時間を設定します。

- **debug** bool

  既定値 `false`

  whatap-master-agentで `DEBUG` レベルのロギングの活性化有無を設定します。

- **collect_apiserver_metric_by_leader** bool

  既定値 `false`

whatap-master-agentのスクラッパーのリーダーとしての動作有無を設定します。 活性化時、whatap-master-agent単独でapiserverにメトリクスをリクエストします。

- **collect_control_plane_monitoring_enabled** bool

  既定値 `false`

  コントロールプレーンモニタリング機能の活性化有無を設定します。

- **master_to_control_plane_connection_timeout** number

  既定値 `5000`

  whatap-master-agentがwhatap-control-plane-helperとの通信時、connection timeoutが発生するまでの時間を設定します。 既定値の単位はmsです。

- **master_to_control_plane_read_timeout** number

  既定値 `20000`

  whatap-master-agentがwhatap-control-plane-helperとの通信時、read connection timeoutが発生するまでの時間を設定します。 既定値の単位はmsです。

- **control_plane_debug** bool

  既定値 `false`

  コントロールプレーンモニタリングに対するwhatap-master-agentでのすべてのデバッグログ機能の活性化有無を設定します。

- **store_k8s_object_manifest_enabled** bool

  既定値 `true`

  オブジェクトマニフェスト収集機能の活性化有無を設定します。 活性化時、1日に1回クラスターオブジェクトのマニフェスト情報を収集します。

- **store_k8s_object_manifest_timezone** string

  既定値 `UTC`

  オブジェクトマニフェスト収集のためのタイムゾーンを設定します。 別途設定がない場合、既定値のUTCを基準にして、午前0時に収集を開始します。

  例、日本時間基準で午前0時に実行したい場合 `GMT+9` あるいは `Asia/Tokyo` のようにタイムゾーン設定

- **store_k8s_object_manifest_max_fetch_limit** number

既定値 50

一度に収集するオブジェクトの照会数を設定します。 別途設定がない場合、既定値の 50 を基準にして1回のリクエストにつき50個ずつリクエストします。

例、全体のPod数が90個のクラスタで該当オプションを 30 に設定すると、Pod情報を30個ずつ3回リクエスト

- **collect_network_policy_enabled** bool

  既定値 true

  NetworkPolicyオブジェクトのイベントベースの収集を有効にするかどうかを設定します。有効にすると、NetworkPolicyが追加、変更、削除されるたびに kube_network_policy カテゴリーへスナップショットを送信します。ClusterRoleに networkpolicies の照会権限が必要です。

- **net_send_buffer_enabled** bool

  既定値 true

  収集サーバーとのネットワークが切断された区間で、送信キューを超過したデータの保管および再送信の可否を設定します。ノードエージェントとマスターエージェントに共通で適用されるオプションです。マスターエージェント1.9.17バージョン以上でサポートします。

- **net_send_buffer_minutes** number

  既定値 15

  保管バッファのデータ保管時間を設定します。既定値の単位は分です。

- **net_send_buffer_max_mb** number

  既定値 64

  保管バッファが使用する最大メモリを設定します。既定値の単位はMBです。

メトリクス別の影響を受ける画面コンポーネント

kube_pod

- ダッシュボード > コンテナマップ

  次の例の画像のPodビュー基準データは、 kube_pod メトリクスの収集サイクルと収集サーバーの転送サイクルに

影響を受けます。



- ワークロード > Podリスト

  次の例のデータは、 kube_pod メトリクスの収集サイクルと収集サーバーの転送サイクルに影響を受けます。

Pod	PodIp	Namespace	Node	ConRestartCnt	Age	CpuByLimit	CpuLimitByNode
whatap-virtual-...	2406:da12:817:106:f...	virtual-main	ip-10-21-13-168.ap-n...	0	27d 02h 54m 54s	4.69%	15.00%
whatap-virtual-...	2406:da12:817:106:f...	virtual-main	ip-10-21-13-168.ap-n...	0	7d 19h 36m 52s	0%	15.00%
whatap-virtual-...	2406:da12:817:106:f...	virtual-main	ip-10-21-13-168.ap-n...	0	22d 06h 13m 11s	3.29%	15.00%

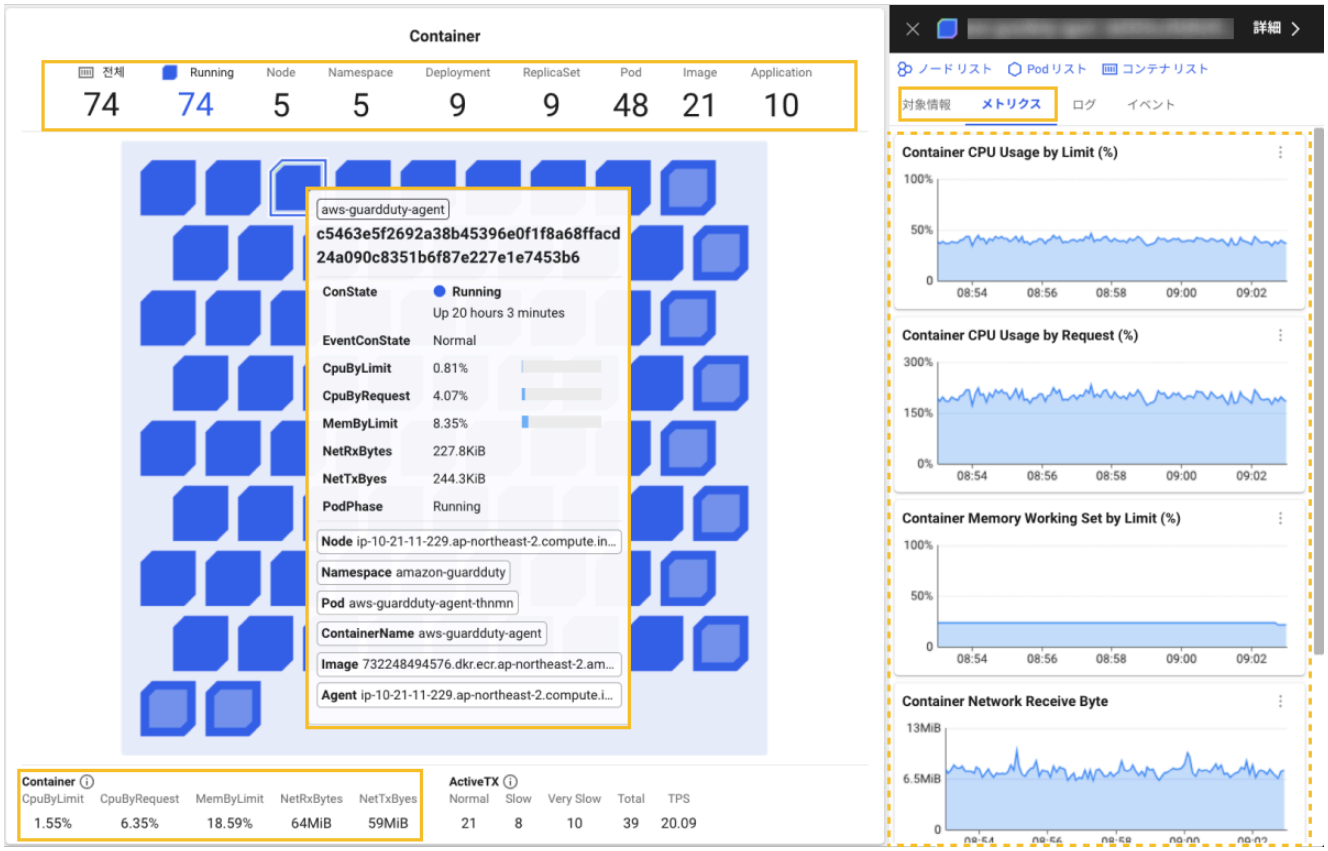
- ダッシュボード > 性能サマリー

  kube_pod メトリクスを利用する**エージェントが実行されているPodの数**チャートの更新サイクルに影響を受けます。

container

- ダッシュボード > コンテナマップ

次の例のコンテナビュー基準画面データの更新サイクルは、 container メトリクスの影響を受けます。



- コンテナ > コンテナイメージ

  次の例のデータは、 container メトリクスの影響を受けます。

Registry	Name	Tag	Repo Digest	Containers	Total Container CPU	Total Container Memory
732248494576.dkr.ecr.ap-nor...	aws-guardduty-agent	v1.5.0	sha256:0393d044ad76b911d...	5	544m	605.55MiB
602401143452.dkr.ecr.ap-nor...	eks/aws-ebs-csi-driver	v1.24.0	sha256:3dba81d0c3c8ce51a...	7	0m	81.15MiB
602401143452.dkr.ecr.ap-nor...	eks/livenessprobe	v2.10.0-eks-1-28-6	sha256:2fbd620d25bda985b...	7	4m	48.63MiB
602401143452.dkr.ecr.ap-nor...	eks/csi-node-driver-regi...	v2.9.0-eks-1-28-6	sha256:16ebeda509ca6f641...	5	0m	29.07MiB

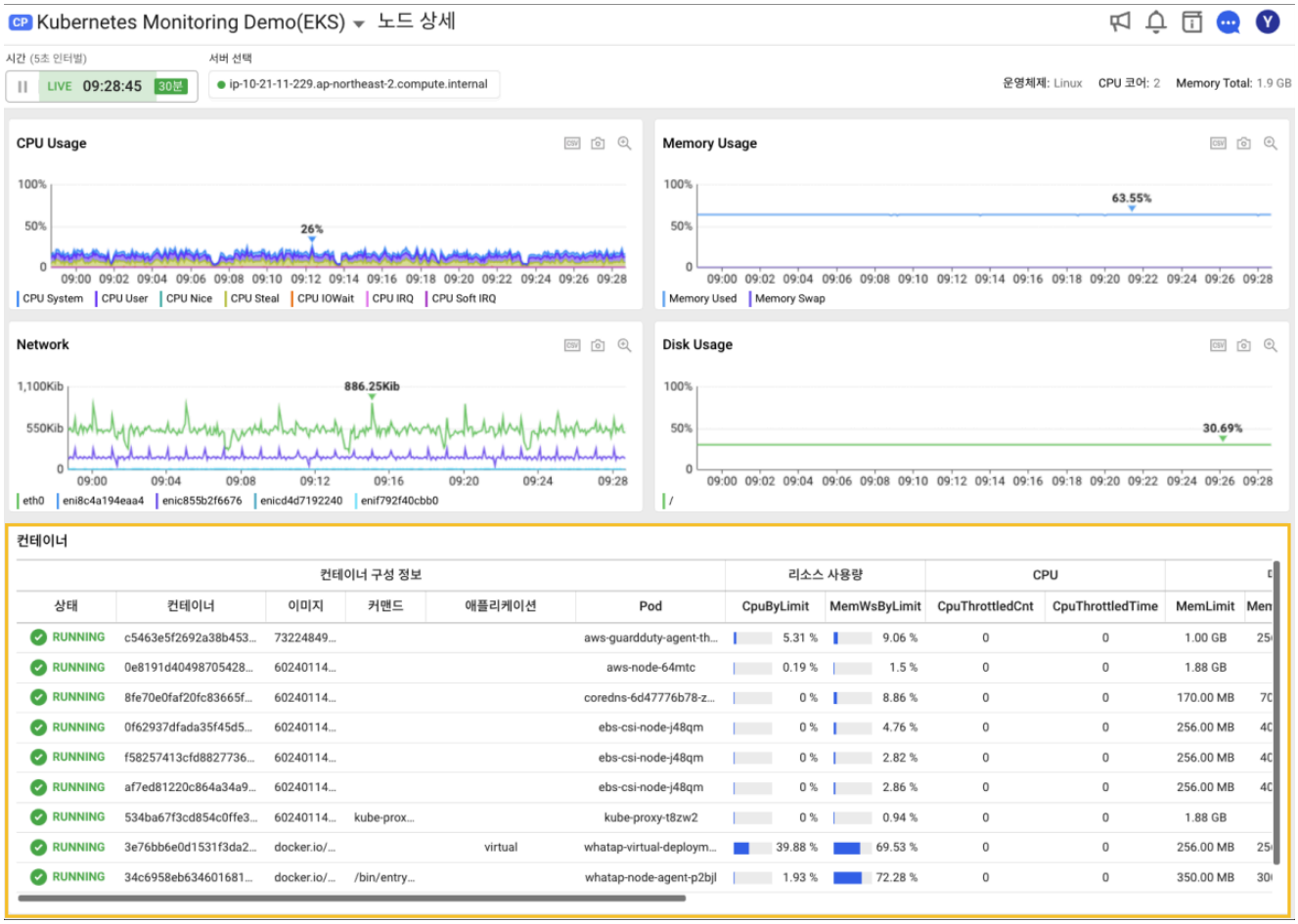
- ダッシュボード > 性能サマリー

  container メトリクスを利用する次のチャートの更新サイクルに影響を受けます。

  - エージェントが実行されているコンテナの数
  - エージェントがインストールされているノードの数
  - Total Container CPU Request

    - Total Container CPU Usage
    - Node CPU
    - Node Memory
    - Container CPU Usage by Request
    - Container CPU Usage by Node
    - Container Memory Working Set by Request
    - Container Memory Failure Count
    - Container CPU Throttling Time
    - Container Network Receive Byte
    - Container Network Transmit Byte
- クラスター > ノード詳細

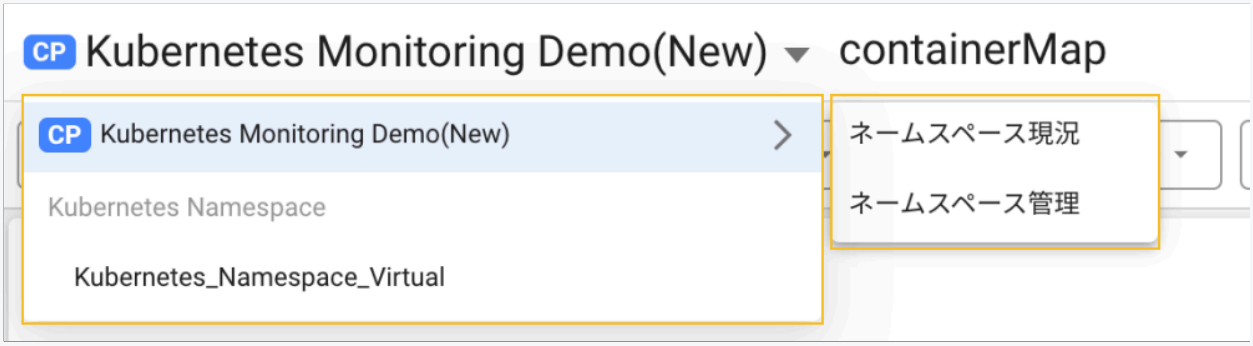
  次の例のノード詳細画面でコンテナー一覧の更新サイクルは、`container`メトリクスの影響を受けます。


Kubernetes Monitoring Demo(EKS)
노드 상세
시간 (5초 인터벌)
LIVE 09:28:45 30분
서버 선택
ip-10-21-11-229.ap-northeast-2.compute.internal
운영체제: Linux
CPU 코어: 2
Memory Total: 1.9 GB
CPU Usage
100%
50%
0
26%
09:00 09:02 09:04 09:06 09:08 09:10 09:12 09:14 09:16 09:18 09:20 09:22 09:24 09:26 09:28
CPU System
CPU User
CPU Nice
CPU Steal
CPU IOWait
CPU IRQ
CPU Soft IRQ
Memory Usage
100%
50%
0
63.55%
09:00 09:02 09:04 09:06 09:08 09:10 09:12 09:14 09:16 09:18 09:20 09:22 09:24 09:26 09:28
Memory Used
Memory Swap
Network
1,100Kib
550Kib
0
886.25Kib
09:00 09:04 09:08 09:12 09:16 09:20 09:24 09:28
eth0
eni8c4a194eaa4
enic855b2f6676
enicd4d7192240
enif792f40cbb0
Disk Usage
100%
50%
0
30.69%
09:00 09:02 09:04 09:06 09:08 09:10 09:12 09:14 09:16 09:18 09:20 09:22 09:24 09:26 09:28
/
컨테이너
컨테이너 구성 정보
리소스 사용량
CPU
상태
컨테이너
이미지
커맨드
애플리케이션
Pod
CpuByLimit
MemWsByLimit
CpuThrottledCnt
CpuThrottledTime
MemLimit
RUNNING c5463e5f2692a38b453... 73224849... aws-guardduty-agent-th... 5.31 % 9.06 % 0 0 1.00 GB
RUNNING 0e8191d40498705428... 60240114... aws-node-64mtc 0.19 % 1.5 % 0 0 1.88 GB
RUNNING 8fe70e0faf20fc83665f... 60240114... coredns-6d47776b78-z... 0 % 8.86 % 0 0 170.00 MB
RUNNING 0f62937dfada35f45d5... 60240114... ebs-csi-node-j48qm 0 % 4.76 % 0 0 256.00 MB
RUNNING f58257413cfd8827736... 60240114... ebs-csi-node-j48qm 0 % 2.82 % 0 0 256.00 MB
RUNNING af7ed81220c864a34a9... 60240114... ebs-csi-node-j48qm 0 % 2.86 % 0 0 256.00 MB
RUNNING 534ba67f3cd854c0ffe3... 60240114... kube-prox... kube-proxy-t8zw2 0 % 0.94 % 0 0 1.88 GB
RUNNING 3e76bb6e0d1531f3da2... docker.io/... virtual whatap-virtual-deploym... 39.88 % 69.53 % 0 0 256.00 MB
RUNNING 34c6958eb634601681... docker.io/... /bin/entry... whatap-node-agent-p2bjl 1.93 % 72.28 % 0 0 350.00 MB

ネームスペース管理

クラスタープロジェクト(CP)では、**ネームスペース管理**メニューからネームスペースプロジェクトの作成と管理ができます。ネームスペースプロジェクトでは、当該メニューにアクセスできません。

クラスタープロジェクトの**ダッシュボード** > **コンテナマップ**メニュー画面の上部には、下位ネームスペースまたは**ネームスペース状況**および**ネームスペース管理**メニューに移動できるリストボックスがあります。

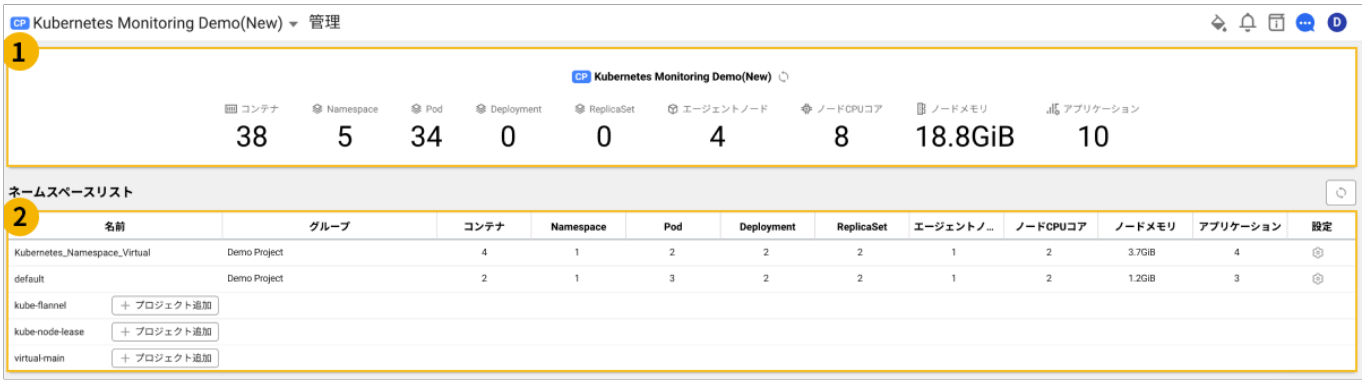


ネームスペースプロジェクトへの分離を検討するケース

次のような状況では、クラスタープロジェクトをネームスペース単位で分離することを検討してください。

- 単一のクラスタープロジェクトに存在するリソース数が多く、Kubernetesのネームスペース基準で分離できる場合
- Kubernetesのネームスペースごとに部署または担当者が異なり、特定のネームスペースのみをモニタリングする必要がある場合

ネームスペースプロジェクトの管理



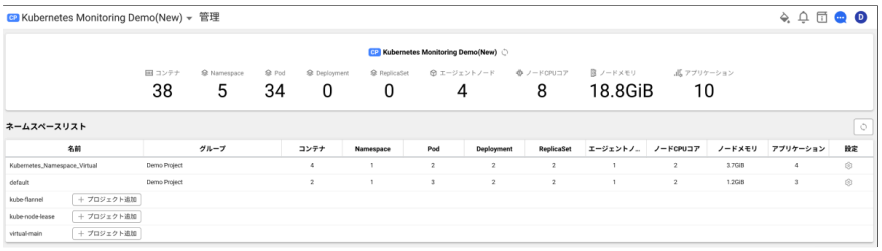
ページ上段の1領域では、全体対象数および主要なリソースの数を確認できます。2 **ネームスペース一覧**領域からネームスペースプロジェクトの検索、追加、削除ができます。

ネームスペースプロジェクトの作成

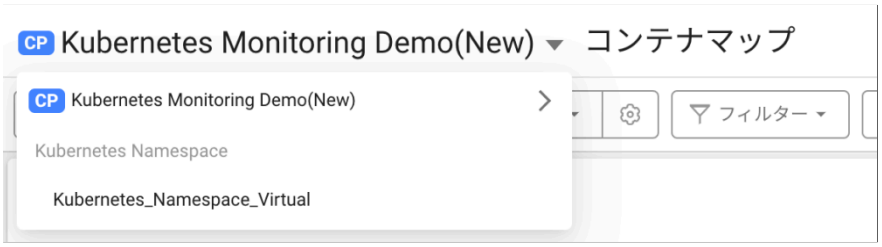
1. **ネームスペース一覧**でプロジェクトを作成する項目の**+ プロジェクト追加**ボタンをクリックしてください。
2. **プロジェクト追加**ウィンドウが表示されたら、ネームスペース名を確認して**確認**ボタンをクリックしてください。



3. 作成したネームスペースプロジェクトは別のWhaTapプロジェクトに分離され、クラスタープロジェクトの**ネームスペース一覧**には表示されなくなります。



4. 分離したネームスペースプロジェクトは、クラスタープロジェクトの**ダッシュボード** > **コンテナマップ**画面の上部にあるプロジェクト一覧ボックスから選択して移動できます。



> ネームスペースプロジェクトを追加すると、そのネームスペースはクラスタープロジェクト(CP)から別のWhaTapプロジェクトに分離されます。追加されたネームスペースプロジェクトには、新たな**プロジェクトアクセスキー**が付与されます。
>
> 分離したネームスペースにアプリケーションモニタリングがインストールされている場合、アプリケーションモニタリング設定内の**プロジェクトアクセスキー**を新たに追加されたネームスペースプロジェクトの**プロジェクトアクセスキー**に変更する必要があります。

ネームスペースプロジェクトの削除

1. **ネームスペース一覧**で削除するプロジェクト項目の右側にある⚙ **設定**アイコンをクリックしてください。
2. **プロジェクト管理**メニューに移動してください。
3. 画面右上の**プロジェクト削除**ボタンをクリックしてください。

> ネームスペースプロジェクトを削除すると、ユーザー一覧情報と収集したデータがすべて削除されます。削

除したデータは復元できないので注意してください。

プロジェクト管理の詳細については、次の文書を参照してください。

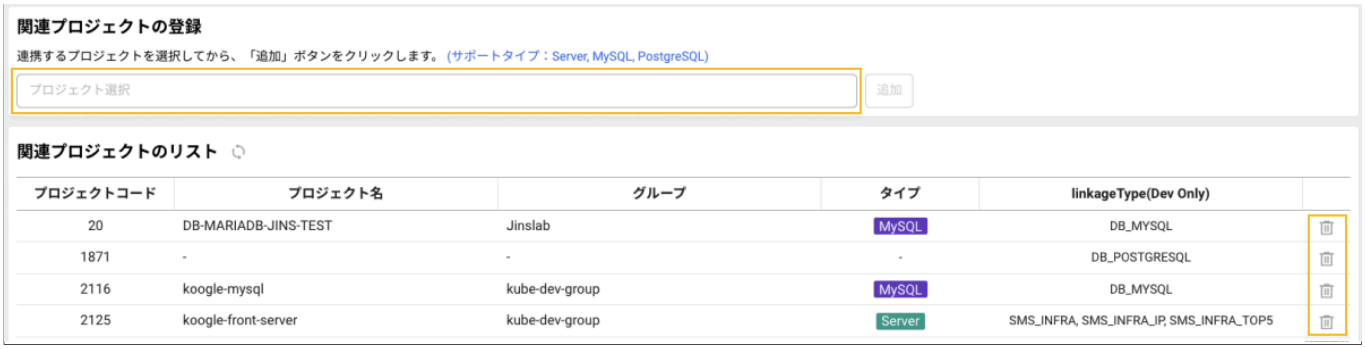
連携プロジェクト管理

本文書は、**相関プロジェクト管理**メニューを案内します。 WhaTapは、プロジェクト間のデータ連携に基づいて統合された可視性を提供します。 タグ情報を通じて連携プロジェクトを検索し、そのプロジェクトから問題の原因になりえる主な指標を素早く確認できます。

- 連携プロジェクトの照会や追加、削除などの管理機能を提供します。
- 追加したプロジェクト間の連携分析が可能です。

基本画面ガイド

ホーム画面 > プロジェクト選択 > **管理** > **相関プロジェクト管理**



次は、**相関プロジェクト管理**メニューの基本画面の構成を案内します。 上段の**連携プロジェクトの登録**と下段の**連携プロジェクトのリスト** 画面で構成されています。

連携プロジェクトの追加

連携プロジェクトを追加できます。 **プロジェクト選択**画面をクリックすると、次のように連携登録できるプロジェクトを照会できます。



- **プロジェクト選択**画面で**プロジェクトコード**及び**プロジェクト名**で検索できます。
- プロジェクト選択時、**追加**ボタンが活性化されます。 複数のプロジェクトを選択できます。
- プロジェクト選択後、**追加**ボタンをクリックすると、案内メッセージとともに下段の**連携プロジェクトのリスト**から当該プロジェクトが追加されたことを確認できます。

連携プロジェクトの削除

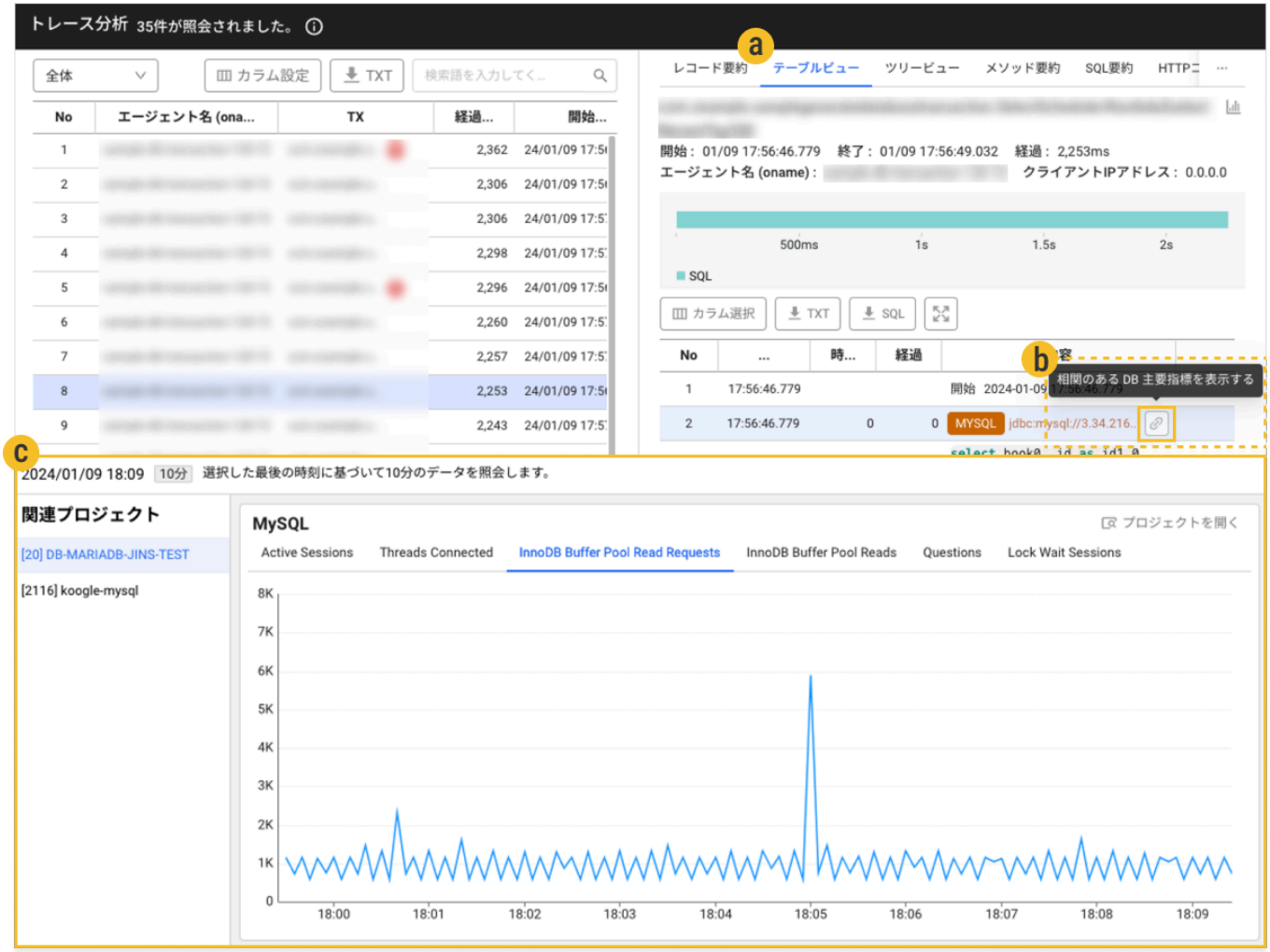
追加したプロジェクトを連携対象から除外したい場合**連携プロジェクト**一覧右側の 削除アイコンを選択して削除します。

> ⓘ
> - 現在Server、PostgreSQL、MySQLサービスに対応しています。
> - 追加した連携プロジェクトがリストに表示されない場合は、しばらく待ってから 更新アイコンを選択して更新してください。
> - **削除**の場合、**プロジェクト修正権限**が必要です。

連携分析活用の例

ヒットマップをドラッグして**トレース分析**画面に移動 > トランザクション項目の選択 > 右側の詳細分析画面から**テーブルビュー**を選択

次に、連携プロジェクトに基づく連携分析の活用例を案内します。



1. **ダッシュボード**の**ヒットマップ**をドラッグして**トレース分析**画面に移動します。
2. 照会するトランザクションを選択し、右側の詳細分析画面から a **テーブルビュー**を選択します。
3. 一覧画面から b 領域の 連携アイコンを選択すると、c 連携プロジェクトの主な指標を確認できます。 連携シナリオが複数ある場合は、ドロップダウン方式で分析シナリオの一覧を提供します。

> **照会時間**
>
> ユーザーが照会した最後の時間に基づいて、10分間のデータを表示します。
>
> 例、**ヒットマップ**で34分~36分の範囲をドラッグする場合、26分~36分間のデータを表示します。

Kubernetesエージェント

Kubernetes内で実行中のアプリケーションをモニタリングするためにインストールしたエージェント構成について説明します。

エージェント構成

WhaTap Kubernetesをインストールする場合、デフォルトでは、Kubernetes環境の<whatap-monitoring>Namespaceの下にエージェントのリソースを構成します。

マスターエージェントはDeploymentの形式であり、ノードエージェントはDaemonSet形式です。以下は、Kubernetes環境にインストールされているWhaTapエージェントを確認する方法です。

- **マスターエージェント**

```
kubectl get deployments -n whatap-monitoring
kubectl get pods -l=name=whatap-master-agent -n whatap-monitoring
```

- **ノードエージェント**

```
kubectl get daemonsets -n whatap-monitoring
kubectl get pods -l=name=whatap-node-agent -n whatap-monitoring
```

以下は、<whatap-monitoring> Namespaceで実行中のマスターエージェントであるノードエージェントが実行される例です。

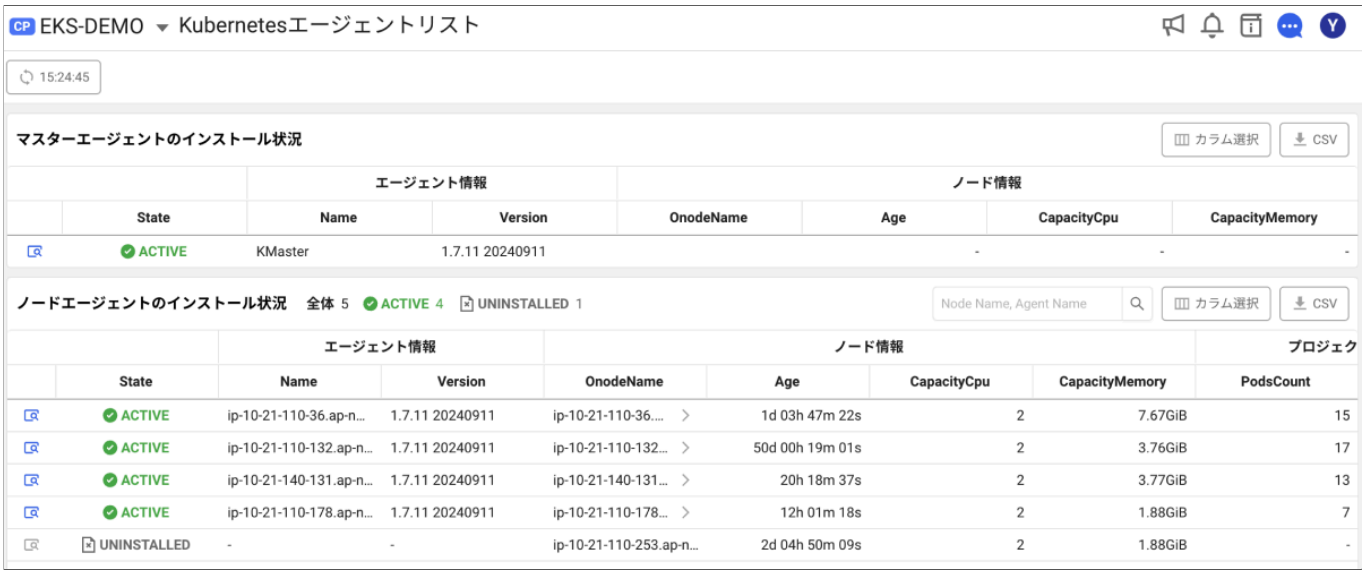
```
$ kubectl get pods -n whatap-monitoring
NAME                          READY   STATUS    RESTARTS   AGE
whatap-master-agent-57f4499567-ltbbr   1/1     Running   0          3d19h
whatap-node-agent-5q8rj               2/2     Running   0          21h
whatap-node-agent-6nb2z               2/2     Running   0          30h
whatap-node-agent-8m2vk               2/2     Running   0          21h
whatap-node-agent-9thmw               2/2     Running   0          3h31m
whatap-node-agent-kpv7d               2/2     Running   0          21h
```

次に、WhaTap Kubernetesコンテナマップに表示される<whatap-monitoring>Namespaceです。

コンテナマップ内のwhatap-monitoring Namespace

Kubernetesエージェントリスト

クラスタープロジェクト(CP)の**管理**メニューの下位から**Kubernetesエージェントリスト**メニューに移動できます。



- Kubernetesエージェントのインストール状況を確認できます。 一部のエージェントが未インストール(UNINSTALLED) もしくは非活性化(INACTIVE) 状態の場合、データ収集に歪みが生じる可能性があるため、迅速な確認が必要です。

  例、Stateカラムの UNINSTALLED、ACITVE、INACTIVE

- マスターエージェントに割り当てられたCPU Limitと設置されたノードエージェント数を確認できます。

  例、ノードエージェント数50個以上の場合、マスターエージェントCPU Limit500ミリコア以上への設定を推奨

- エージェントがインストールされているノードに割り当てられたCPUおよびメモリ情報を確認できます。

  例、ノード情報カラム下位のCapacityCpu、CapacityMemory

- エージェントがインストールされていないノードを確認できます。
- エージェントの主要情報を確認できます。
- ノードエージェントバージョンに関する案内を確認できます。

  例、ノード間バージョンが一致しない場合、上部にバージョンの不一致を案内

> **マスターエージェントリソースの割り当て**
>
> ユーザのKubernetes環境規模、つまりノード数によって、WhaTapエージェントが収集・処理するデータの量が異なるため、マスターエージェントには十分なリソース割り当てが必要です。 例えば、ノードエージェント数が多い場合、マスターエージェントCPU Limitの上方修正をお勧めします。

リフレッシュ

上段の 更新アイコンを選択すると、現在の時間に基づいてデータを更新できます。

カラム選択

グループ	カラム名	説明
エージェント情報	Name	エージェント名またはエージェント別名を設定した場合、エージェントの別名を意味します。
	Original Name	エージェント名です。 通常、ノードの名前と同じです。
	Alias	ユーザーが定義したエージェント別名を意味します。
	Initial	エージェントを指す時に使用する略語です。
	Id	一意に付与するエージェントの識別名(ID)です。
	Start Time	該当エージェントが最初に実行を開始した時刻です。
	Active Time	現在エージェントがアクティブになり、モニタリングデータの収集を開始した時刻です。
	Inactive Time	エージェントが非アクティブになり、データ収集が中断された時刻です。
	VERSION	現在のエージェントに適用されているソフトウェアバージョンです。
	IP	エージェントがインストールされているサーバーのIPアドレスを意味します。
	CPU Cores	エージェントがインストールされているサーバーのCPUコア数を意味します。

グループ	カラム名	説明
ノード情報	OnodeName	エージェントがインストールされているノード名です。
	Onode	エージェントがインストールされているノード名を変換したハッシュ値です。
	Age	ノードが作成された以降の経過時間を意味します。
	CapacityCpu	該当ノードが保有しているCPUの総容量を意味します。
	CapacityMemory	該当ノードが保有しているメモリの総容量を意味します。
プロジェクトリソース	PodsCount	ノードエージェントがインストールされているノードで実行中のPodの数を意味します。 分離されたネームスペースは除外されます。
	ContainersCount	ノードエージェントがインストールされているノードで実行中のコンテナの数を意味します。 分離されたネームスペースは除外されます。

カラムを選択オプションを使用すると、目的のエージェント情報を選択してエージェント一覧を表示することができます。

- 選択したカラム一覧は、メニューを再入力すると維持されます。
- リセットボタンを選択すると、定義済みの既定の主要指標カラムが選択されます。 この場合、既存のカラム選択は削除されます。
- カラムを選択した後、右下の適用ボタンをクリックして選択履歴を適用することができます。

ダウンロード

CSV ダウンロードアイコンの選択時範囲と表示オプションを設定し、当該データをCSVファイルにダウンロードすることができます。

詳細を確認

マスターエージェントエージェント詳細カラムのアイコンを選択すると、右側に詳細画面が表示されます。 環境変数、ヒープヒストグラム、スレッド一覧/ダンプ、エージェントログを確認できます。

ノードエージェント一覧で、照会を希望するエージェント詳細カラムのアイコンを選択すると、右側にそのノード

の詳細画面が表示されます。

> ⓘ ノードの詳細については次の文書を参照してください。

エージェントのアップデート

> - EKS Fargateは、対応予定です。
> - WhaTap Kubernetesエージェントの最新バージョンは、hub.docker.comで確認できます。

エージェントのアップデートが必要な場合は、次のコマンドを実行してください。

- **マスターエージェント**

```
kubectl set image -n whatap-monitoring deployment/whatap-master-agent whatap-master-agent=whatap/
kube_mon:{agent_version}
```

- **ノードエージェント**

```
kubectl set image -n whatap-monitoring daemonset/whatap-node-agent whatap-node-agent=whatap/
kube_mon:{agent_version}
```

- **Helmチャート基準インストール時のエージェントアプリケーションのアップデート**

```
# releaseアップグレード
helm upgrade whatap-kube-agent whatap/kube -f values.yaml
```

エージェント削除

Kubernetesエージェントの削除

エージェントのインストールでダウンロードしたKubernetesバージョンとContainer Runtimeに合う yaml ファイル名を確認し、手順に従ってKubernetesエージェントの削除を行ってください。

KubernetesエージェントのYAMLファイル名

Docker	YAMLファイル名
Kubernetesバージョン1.16以降	whatap_kube_1.16.yaml
Kubernetesバージョン1.16以降(GPUバージョン)	whatap_kube_1.16_gpu.yaml
Kubernetesバージョン1.16未満	whatap_kube.yaml
Kubernetesバージョン1.16未満(GPUバージョン)	whatap_kube_gpu.yaml

Containerd	YAMLファイル名
Kubernetesバージョン1.16以降	whatap_kube_containerd_1.16.yaml
Kubernetesバージョン1.16以降(GPUバージョン)	whatap_kube_containerd_1.16_gpu.yaml
Kubernetesバージョン1.16未満	whatap_kube_containerd.yaml
Kubernetesバージョン1.16未満(GPUバージョン)	whatap_kube_containerd_gpu.yaml

CRIO YAML	ファイル名
Kubernetesバージョン1.16以降	whatap_kube_crio_1.16.yaml
Kubernetesバージョン1.16以降(GPUバージョン)	whatap_kube_crio_1.16_gpu.yaml

CRIO YAML	ファイル名
Kubernetesバージョン1.16未満	whatap_kube_crio.yaml
Kubernetesバージョン1.16未満(GPUバージョン)	whatap_kube_crio_gpu.yaml

YAMLファイルがある場合

1. インストール段階でダウンロードしたyamlファイル名でインストールされているKubernetesエージェントを削除します。 次のコマンドの表記部分をユーザのyamlファイル名に変更して実行してください。

```
kubectl delete -f <ユーザーのYAMLファイル名>
```

2. Kubernetesエージェントの削除が正常に完了した場合、次のようなメッセージを確認できます。

```
namespace "whatap-monitoring" deleted
serviceaccount "whatap" deleted
clusterrole.rbac.authorization.k8s.io "whatap" deleted
configmap "node-start-script" deleted
clusterrolebinding.rbac.authorization.k8s.io "whatap" deleted
daemonset.apps "whatap-node-agent" deleted
configmap "master-start-script" deleted
deployment.apps "whatap-master-agent" deleted
service "whatap-master-agent" deleted
```

YAMLファイルがない場合

whatap-monitoringネームスペースに直接ユーザー定義リソースを作成した場合、そのリソースも一緒に削除されます。

1. 次のコマンドを実行して、Kubernetesエージェントを削除してください。

```
kubectl delete namespace whatap-monitoring
```

```
kubectl delete clusterrolebinding whatap
kubectl delete clusterrole whatap
```

2. Kubernetesエージェントの削除が正常に完了した場合、次のようなメッセージを確認できます。

```
namespace "whatap-monitoring" deleted
clusterrolebinding.rbac.authorization.k8s.io "whatap" deleted
clusterrole.rbac.authorization.k8s.io "whatap" deleted
```

Helmチャート基準インストール時のエージェントの削除

次のコマンドを実行して、WhaTap Kubernetesエージェントアプリケーションを削除します。

```
# release削除
helm uninstall whatap-kube-agent
```

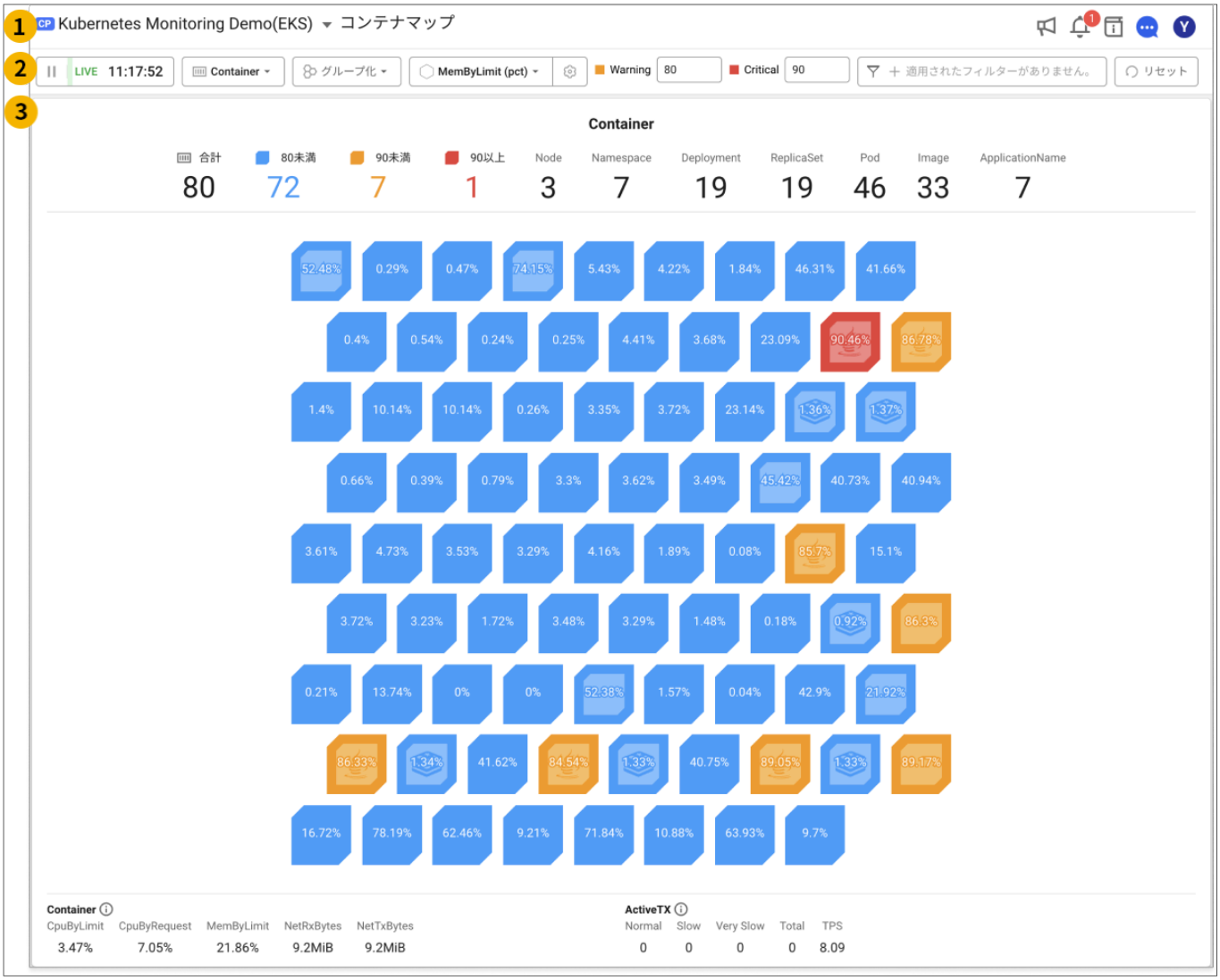
コンテナマップ

WhaTap Kubernetesのコンテナマップは、Kubernetesモニタリングの開始点です。ユーザー設定の表示によって、各オブジェクトは1つのコンテナ、または1つのPodを表示します。

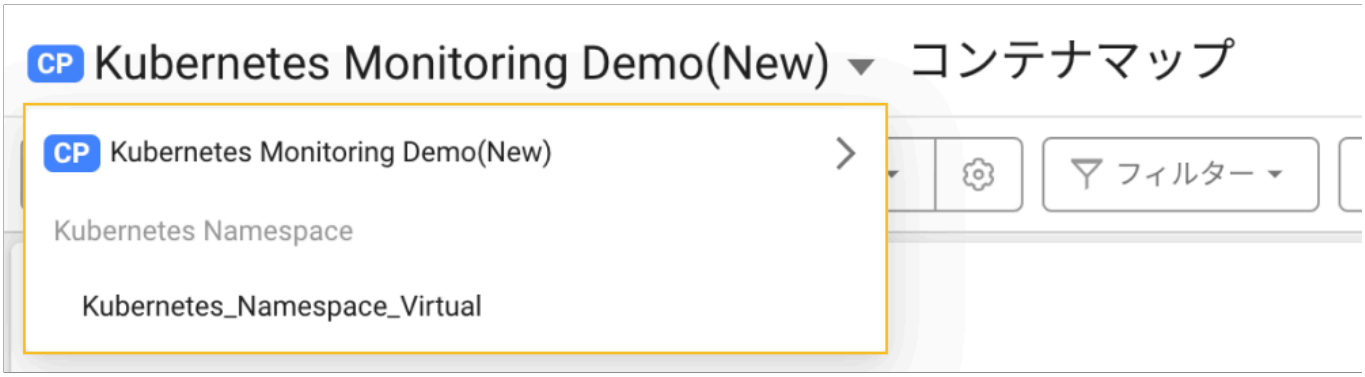
- コンテナマップは、分析対象を選択し、グループを構成できます。
- ステータス指標としきい値を設定して、個々のコンテナまたはPodの状態を一覧で見ることができます。
- 個々のコンテナまたはPodを選択して、詳細情報を確認できます。トレース、メトリクス、ログ、イベントなどの主要なモニタリングデータを提供します。

次の動画は、コンテナマップメニューに関する基本ガイドです。

基本画面ガイド


1
Kubernetes Monitoring Demo(EKS)
コンテナマップ
2
LIVE 11:17:52
Container
グループ化
MemByLimit (pct)
Warning 80
Critical 90
適用されたフィルターがありません。
リセット
3
Container
合計 80
80未満 72
90未満 7
90以上 1
Node 3
Namespace 7
Deployment 19
ReplicaSet 19
Pod 46
Image 33
ApplicationName 7
Container
CpuByLimit CpuByRequest MemByLimit NetRxBytes NetTxBytes
3.47% 7.05% 21.86% 9.2MiB 9.2MiB
ActiveTX
Normal Slow Very Slow Total TPS
0 0 0 0 8.09

1 クラスタープロジェクト - ネームスペースプロジェクトの移動



クラスタープロジェクトCPの下位にネームスペースプロジェクトを作成した場合、コンテナマップ画面の上の もっと見るアイコンを選択すると、プロジェクトリストボックスが表示されます。プロジェクトリストボックスでは、クラスタープロジェクトを含むプロジェクトリストを確認でき、プロジェクト間の移動が可能です。

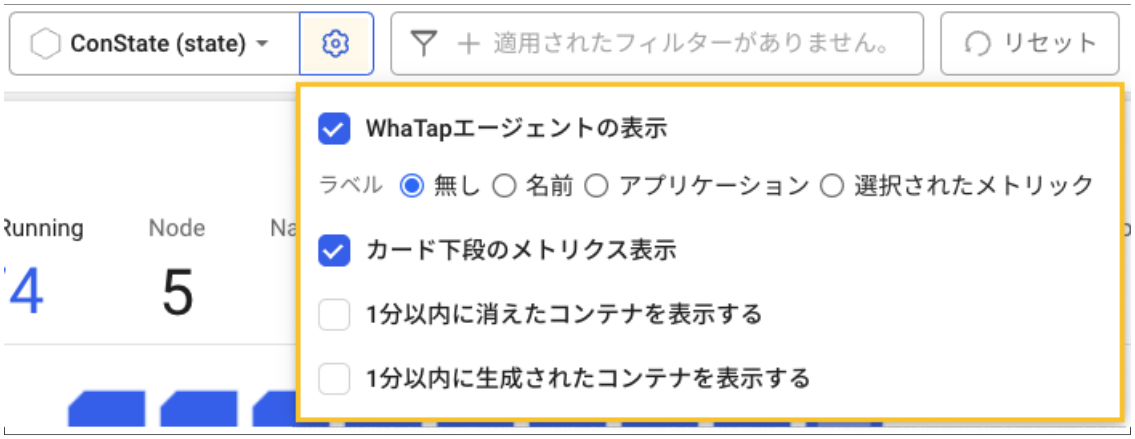
> ネームスペースプロジェクトの作成および管理に関する詳細は、次の文書を参照してください。

2 ユーザー設定



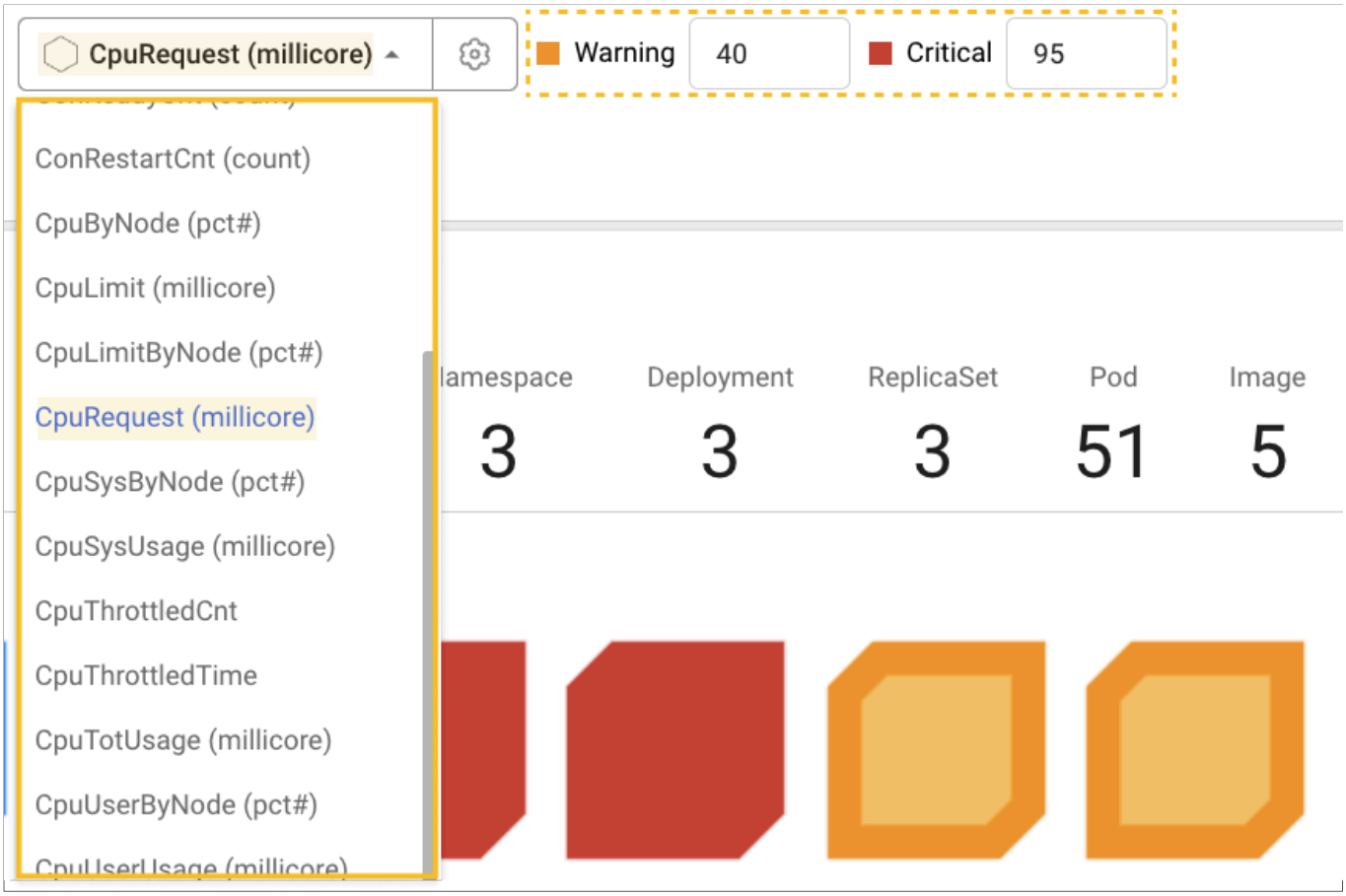
コンテナマップには、複数のユーザー設定が用意されており、目的の画面表示を作成する機能が画面順に用意されています。

- カテゴリ：分析対象を選択できます。ContainerまたはPodを指定できます。
- グループ化：グループ化(クラスタリング) オプションを使用すると、クラスターコンポーネント間の階層関係を合計2段階まで表現できます。詳しい内容は、次の文書を参照してください。
- ConState(state)：チャート指標を選択できます。
- ⚙ボタンをクリックすると、各オブジェクトにコンテナまたはPodの情報を表示できるオプションを選択できます。



  - WhaTapエージェントの表示：コンテナまたはPodの情報を表示できます。必要なラベルオプションを選択します。ラベルオプションの詳細については、次の文書を参照してください。
  - カードの下部にメトリクスを表示する：画面の下部にメトリクス情報を表示の有無を選択できます。
  - 1分以内に作成されたコンテナまたはPod表示: 1分以内に作成されたコンテナまたはPodを表示するオプションです。
  - 1分以内に削除されたコンテナまたはPod表示: 1分以内に削除されたコンテナまたはPodを表示するオプションです。
- フィルター：ユーザーが選択した項目に基づいてフィルタリングし、コンテナまたはPodの状態を確認できます。
- リセット：ユーザーが設定した画面表示を初期化します。

指標選択としきい値設定



チャート指標の選択としきい値の設定は、コンテナマップのステータス表現を決定します。選択した指標の単位が数字の場合、しきい値を設定できます。設定したしきい値は、次のカテゴリに従って色で表現されます。

- 0以上Warning以下: 青色
- Warning以上Critical以下: オレンジ色
- Critical以上: 赤色

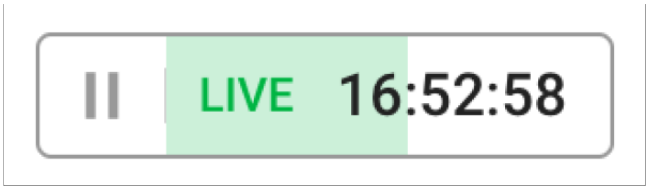
> ⓘ
> - 数値型の指標単位としては、**pct**(percentage)、**bps**、**millicore**などがあります。**pct#**は、100%以上の値が出る指標単位です。
> - チャート指標の選択としきい値設定の詳細については、次の文書を参照してください。

ラベルオプション

- なし：ブロックにラベルを表示しません。
- 名前：コンテナ/PODの名前を表示します。
- アプリケーション：コンテナ/POD内のアプリケーションを表示します。
- 選択された指標：ConState(state)項目で選択されたチャート指標を表示します。

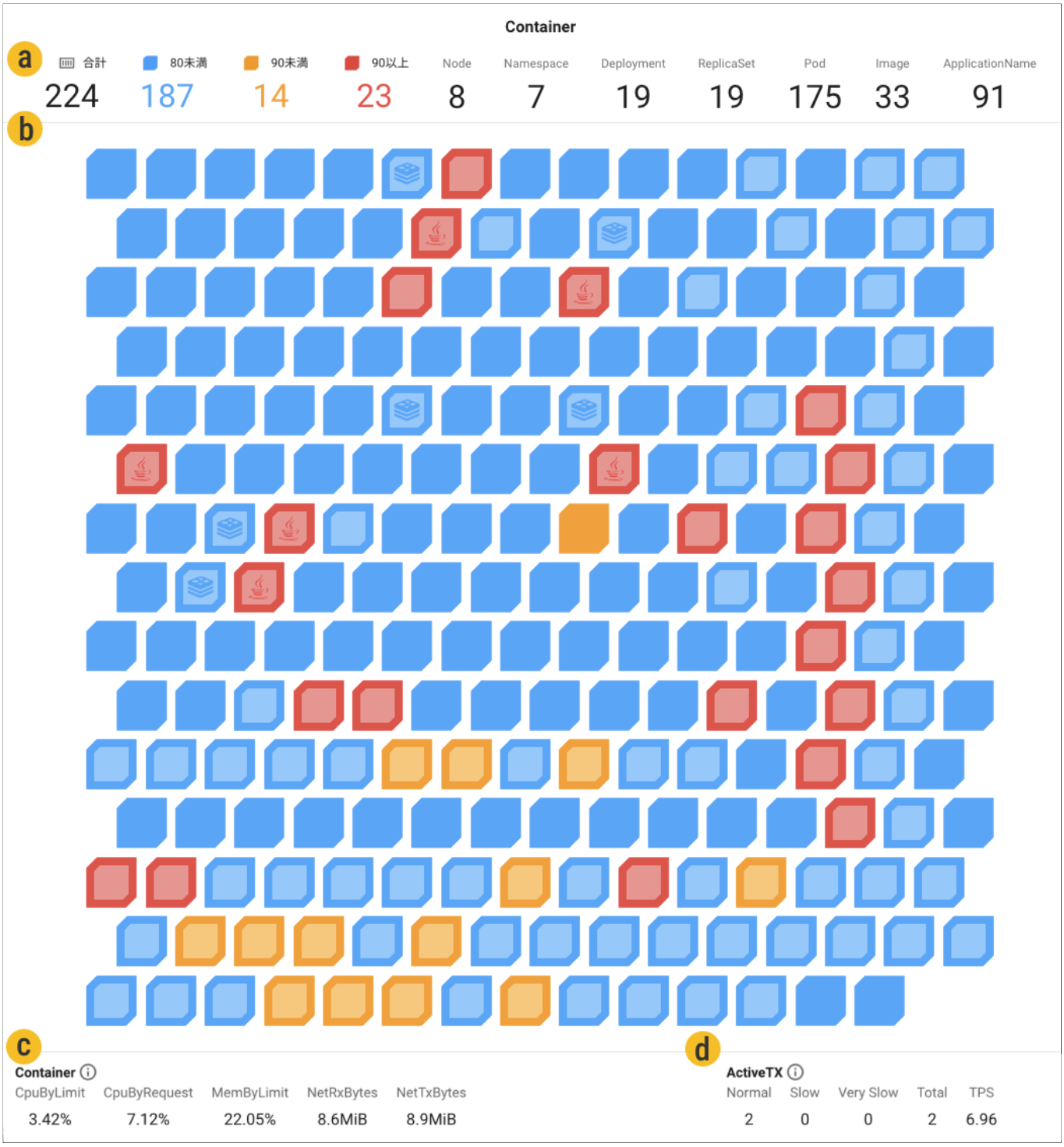
リアルタイム及び履歴参照

コンテナマップは基本的にリアルタイムの検索ベースですが、複数の視点に対応した検索機能を提供します。リアルタイムビューを変更するには、画面の左上の時間選択を使用します。



参照ステータス	説明
リアルタイム参照 (LIVE)	基本モード 10秒間隔でリアルタイムステータス変更
一時停止の参照 (PAUSE)	リアルタイムステータス変更OFF 現在のステータスを詳しく確認するために使用
過去の参照	ユーザーは、過去の特定の時点を指定し、確認 年/月/日/時/分を指定できます

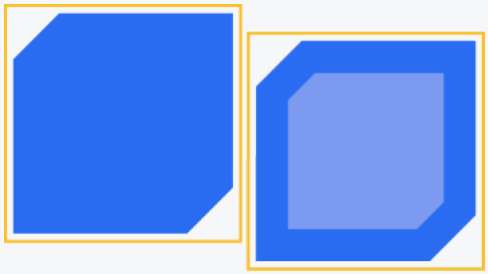
3 コンテナマップチャート



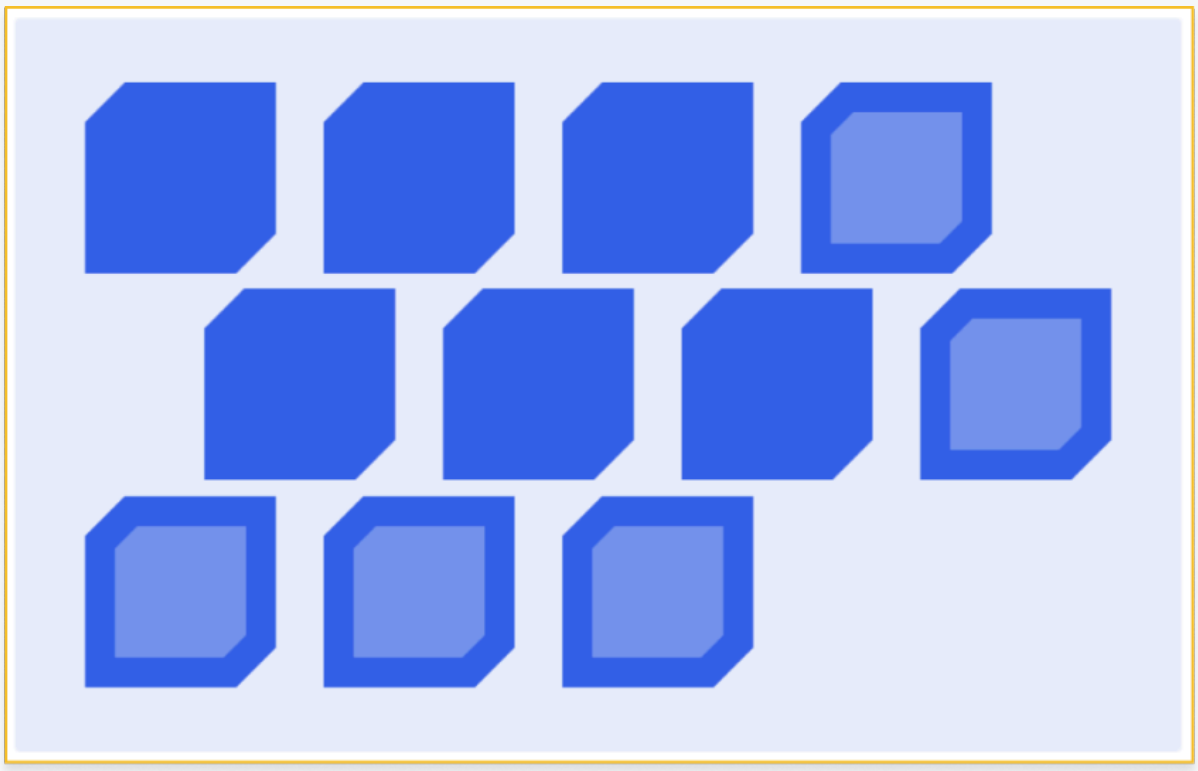
- ⓐは、グループ化された対象の総数と主要なリソース数を意味します。
- ⓑ各オブジェクトの形状と色は、コンテナまたはPodの特性と状態を意味します。
  - **外部プロジェクト連携**

    ブロック内のロゴを通じて連動した外部プロジェクトを確認できます。詳細については、次の文書を参照してください。
  - **色**

    コンテナまたはPodの状態を意味します。Normal、Warning、Critical以外の状態についての詳細は、次の文書を参照してください。
  - **境界線**
    - オブジェクトの中には境界線が表示されます。これは、コンテナに、Java、Node.js、Pythonなどのマイクロエージェントをインストールする場所です。
    - 1つのPod内に2つのコンテナがあり、2つのコンテナのうち１つだけにマイクロエージェントをインストールした場合、境界線が表示されます。つまり、Pod内のコンテナのいずれかにマイクロエージェントがインストールされている場合は、境界線が表示されます。

> ⓘ **ブロックとマイクロブロック**
>
> 
>
> コンテナマップでは、1つのポリゴンを**ブロック**(Block)と言います。このとき、エージェントを含むブロック(ブロック内のブロック)を**マイクロブロック**(Micro Block)と呼びます。

> ⓘ **グループ**



コンテナマップ画面ですべてのブロックを選択した場合、その複数のブロックをグループと呼びます。

- cは、グループ化された対象のCPUおよびメモリ関連の指標を意味します。
- dは、グループ化された対象の内部アプリケーションのトランザクション情報を意味します。

コンテナマップの対象情報

ユーザーは、関心のあるコンテナまたはPodを選択して、概要と詳細情報を確認できます。この情報には、次の主要なモニタリングデータが含まれます。

- **対象情報**

  ブロッククリックで、対象特定時に選択した対象に関連するリソース情報と、そのメニューリンクの移動を提供します。

- **リソース関係**

選択した対象に関連するリソース情報を視覚的に表現した関係チャートで提供します。

- **トレース**

  コンテナ内にWhaTapアプリケーションエージェントがインストールされている場合、ダッシュボード上でリアルタイムで確認できるよう、そのアプリケーションのトランザクショントレース情報を提供します。

- **メトリクス**

  選択したコンテナまたはPodのCPU使用量、メモリ使用量、ネットワークトラフィックなどのメトリクス情報をダッシュボードで提供します。

- **ログ**

  各コンテナの標準出力ログ(Standard Output) およびコンテナとアプリケーションで生成され、保存されたログ情報を提供します。

- **イベント**

  Kubernetesで発生したイベントとWhaTapイベント情報を提供します。

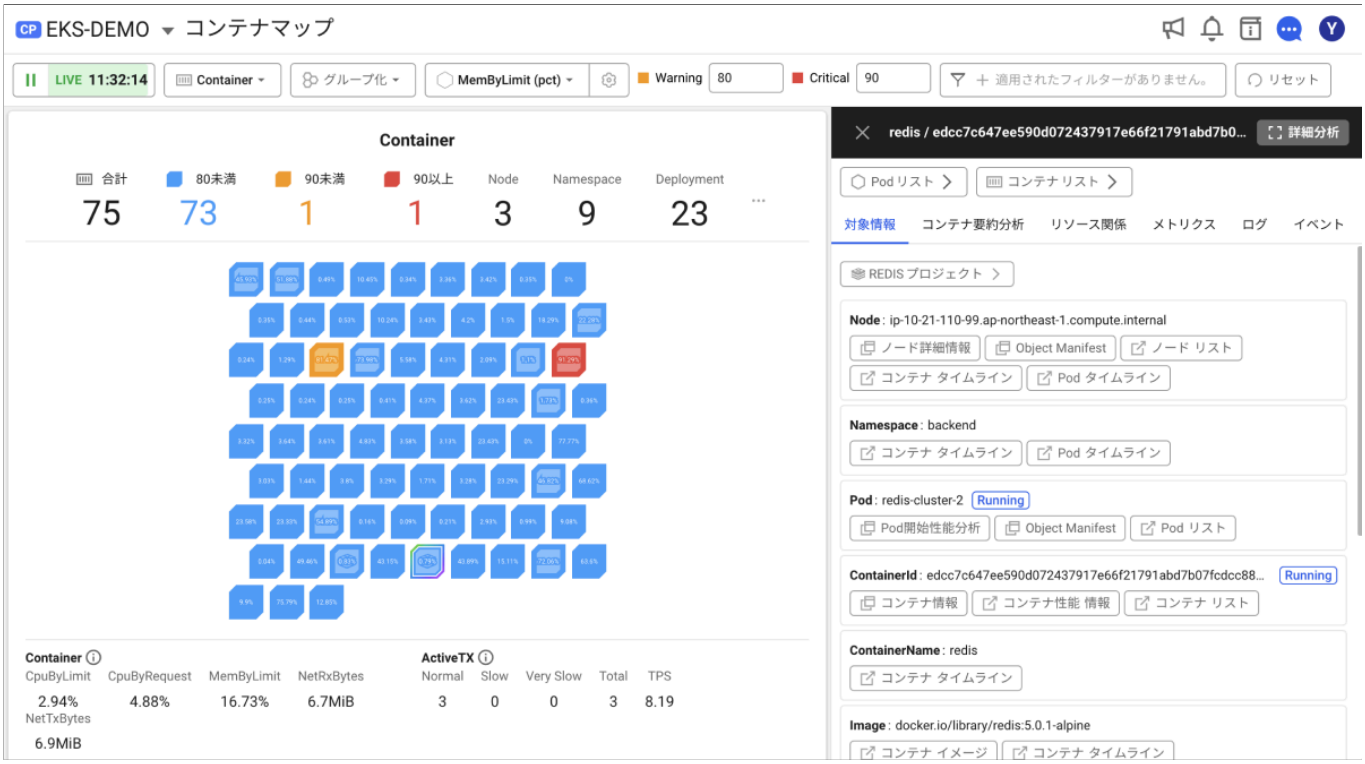
- **呼び出し情報**

  選択したコンテナ内で実行中のアプリケーションの呼び出しの流れを視覚化したチャートを提供します。

これらのデータに基づいて、ユーザはKubernetes環境のObservabilityを確保できます。次の**概要を表示**および**詳細分析**すべての画面が上記の4つの項目を中心に情報を提供します。

概要を表示

コンテナマップで、特定のオブジェクトまたはオブジェクトのセットをクリックします。右側に現れる**概要を表示** 画面を通じてそのオブジェクトの要約情報を確認できます。



- 対象情報

  選択したPodまたはコンテナに関する基本情報を提供します。当該リソースが実行中のノード、所属するネームスペースなど、基本的な対象情報を確認し、一覧メニューに移動できます。

ⓘ
- **Node**: ノード詳細、オブジェクトマニフェスト、Nod要約分析 情報を確認し、ノード一覧 メニューに移動できます。
- **Deployment**: Object Manifest 情報を確認し、Deployment リスト メニューへ移動できます。
- **ReplicaSet**: Object Manifest 情報を確認できます。
- **Pod**：Pod開始解析、オブジェクトマニフェスト、Pod要約分析情報を確認しPod一覧メニューに移動できます。
- **ContainerId**: **コンテナ情報**、コンテナ要約分析 情報を確認し、コンテナリスト メニューへ移動できます。
- **Image**: **コンテナイメージ**情報を確認し、コンテナイメージ メニューへ移動できます。
- **ServiceName**: Object Manifes 情報を確認し、Serviceリスト メニューへ移動できます。

- **Volume**：コンテナが使用するボリュームとマウント経路の情報を確認できます。
  - Podで使用するaボリュームの名前とbボリュームタイプを確認できます。

    例、**Volume** a containerddomainsocket b HostPath
  - **HostPath**ボリュームタイプを使用する場合cホストシステムのファイルパスを確認できます。 このパスは、ファハドとホストシステム間のファイルシステムを接続するパスです。

    例、**Host Path** c /run/containerd/containerd.sock

- リソース関係

  コンテナマップのリソース関係タブを使用して、Kubernetesリソース間の接続関係を視覚的に確認できます。

- トレース

  コンテナマップのトレースタブを使用して、コンテナ上で実行されるアプリケーションの個々のトランザクションを追跡できます。

- メトリクス

  コンテナマップのメトリクスタブを使用して、各コンテナのリソース割当量、使用量などの各種指標データをリアルタイムで確認できます。

- ログ

  コンテナマップのログタブを使用すると、各コンテナの標準出力ログとコンテナ上のアプリケーションが生成するログを管理および表示できます。

- イベント

  コンテナマップのイベントタブを使用して、Kubernetesクラスターで発生したイベントとWhaTapのアラート機能を使用して、ユーザーのしきい値のイベントを表示できます。

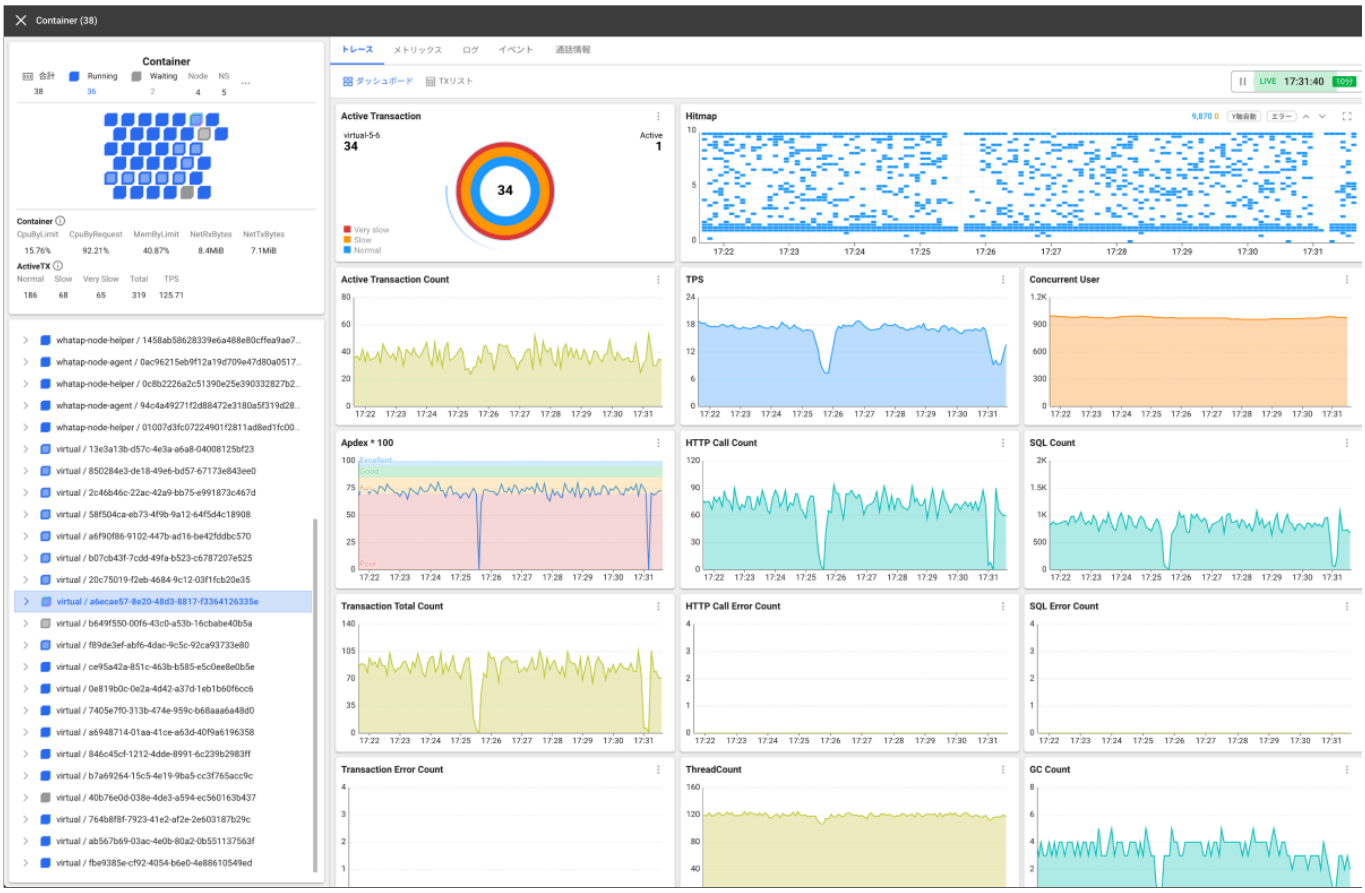
- 呼び出し情報

  コンテナマップの呼び出し情報タブを通じて、Kubernetesコンテナ上で実行中のアプリケーションの呼び出しの流れと外部呼び出し対象の状態を確認できます。

詳細分析

概要を表示画面の右上に**詳細分析**ボタンをクリックします。**詳細分析**画面に切り替えます。**詳細分析**画面構成につ

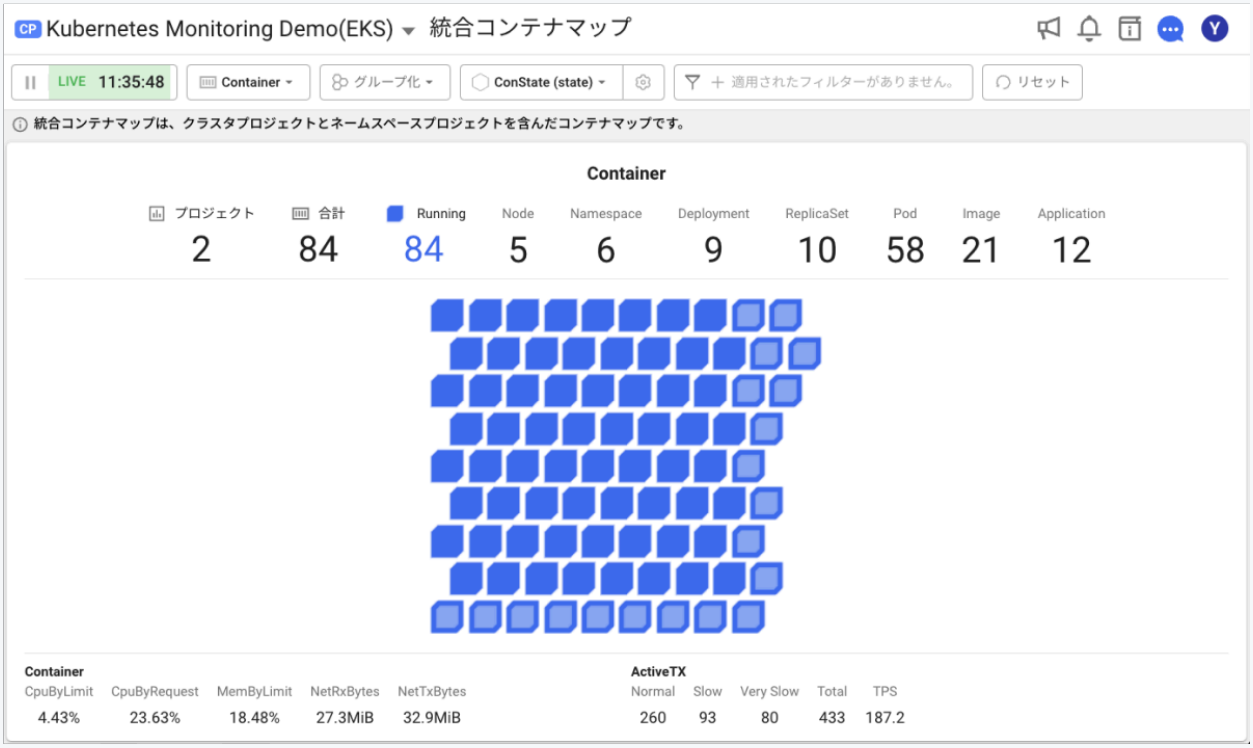
いての詳しい案内は、次の文書を参照してください。



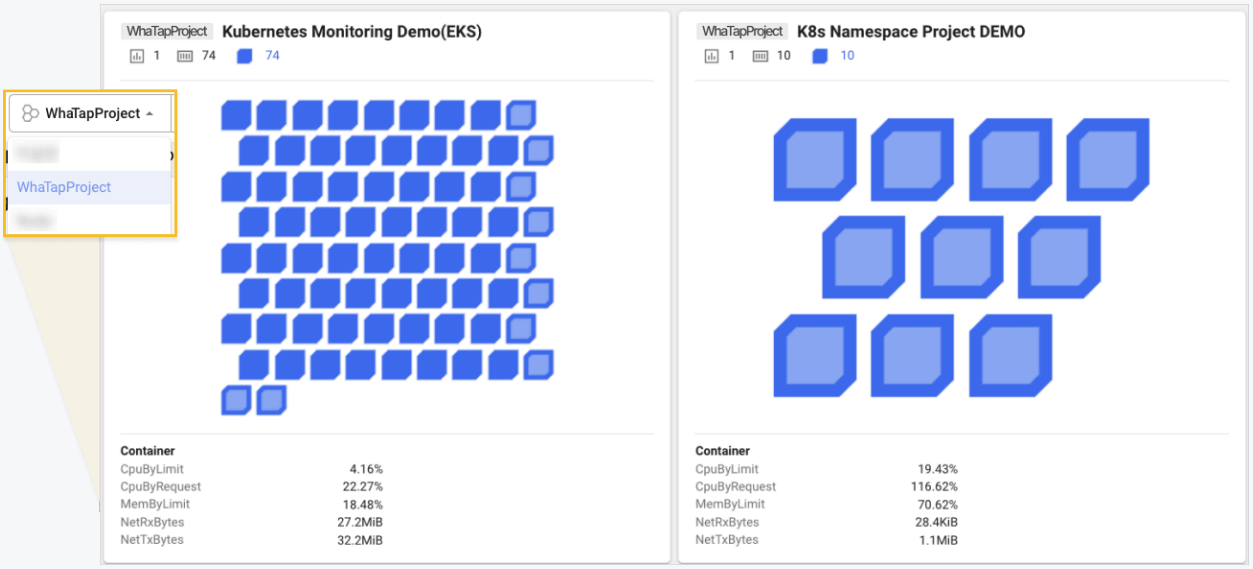
統合コンテナマップ

ホーム画面 > **クラスター** > **統合コンテナマップ**

クラスタープロジェクト(CP)とネームスペースプロジェクト(Namespace)を含むコンテナマップを照会するには、統合コンテナマップを確認してください。



統合コンテナマップは、プロジェクト別グループ化オプションを提供します。グループ化オプションの中でWhaTapProjectグループ化により、一つの画面でクラスタープロジェクト及びネームスペースの個別プロジェクトでのコンテナマップを次のように照会できます。



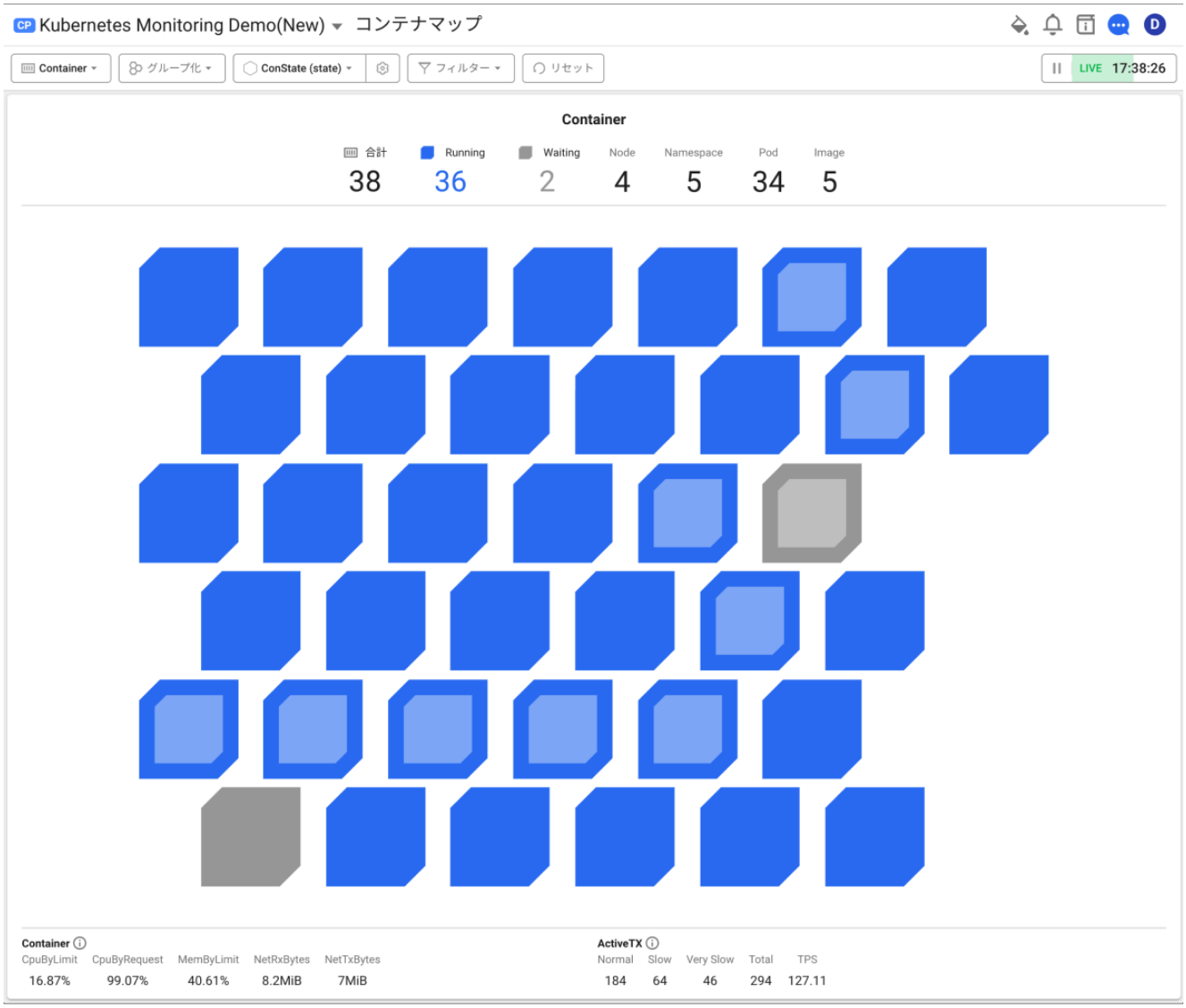
コンテナビュー、Podビュー

カテゴリ選択(コンテナビュー、Podビュー)

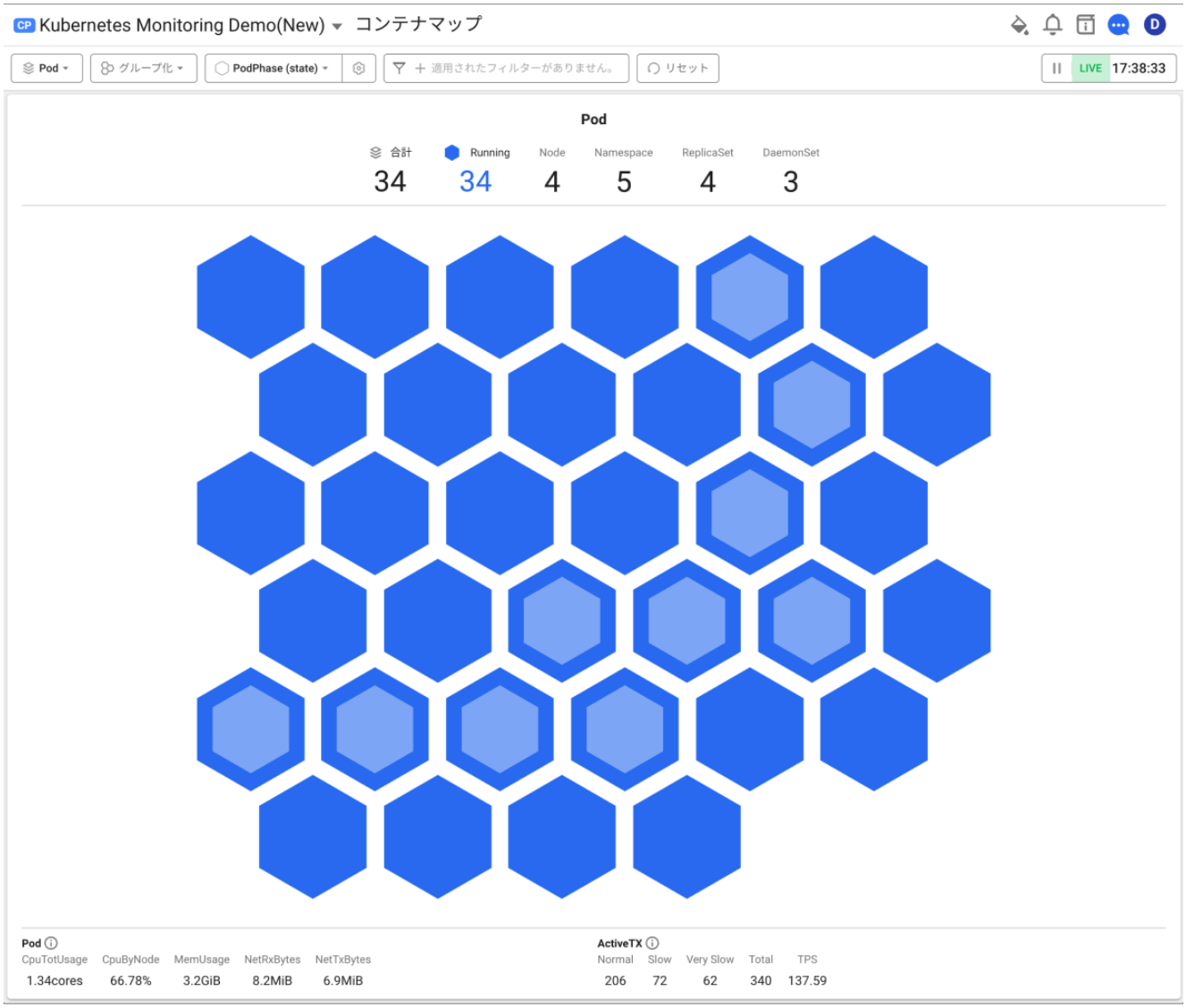
コンテナマップメニューに初めてログインすると、クラスタープロジェクト内の個々のコンテナまたは、Podが展開されていることが確認できます。1つのPod内で2つ以上のコンテナを実行できるため、コンテナ、Podのそれぞれのビューが必要です。

WhaTap Kubernetesのコンテナマップメニューは、**コンテナ基準ビュー**、**Pod基準ビュー**の2つのモードに対応しているため、コンテナ単位のモニタリング、Pod単位のモニタリングなどの状況に応じたモニタリングが可能です。

- **コンテナ基準ビュー**


Kubernetes Monitoring Demo(New)
コンテナマップ
Container
グループ化
ConState (state)
フィルター
リセット
LIVE 17:38:26
Container
合計 Running Waiting Node Namespace Pod Image
38 36 2 4 5 34 5
Container
CpuByLimit CpuByRequest MemByLimit NetRxBytes NetTxBytes
16.87% 99.07% 40.61% 8.2MiB 7MiB
ActiveTX
Normal Slow Very Slow Total TPS
184 64 46 294 127.11

- **Pod基準ビュー**



コンテナ基準ビューと**Pod基準ビュー**は、コンテナマップメニュー画面上のユーザー設定を切り替えることができます。

> ⓘ 旧バージョンのKubernetesエージェントを使用する場合、Pod基準ビューを確認できません。エージェントのアップグレードを実行してください。
>
> Master

```
kubectl set image deployment/whatap-master-agent whatap-master-agent=whatap/kube_mon:{エージェントバージョン} -n whatap-monitoring
```

Node

```
kubectl set image daemonset/whatap-node-agent whatap-node-agent=whatap/kube_mon:{エージェントバージョン} -n whatap-monitoring
```

対象グループ化

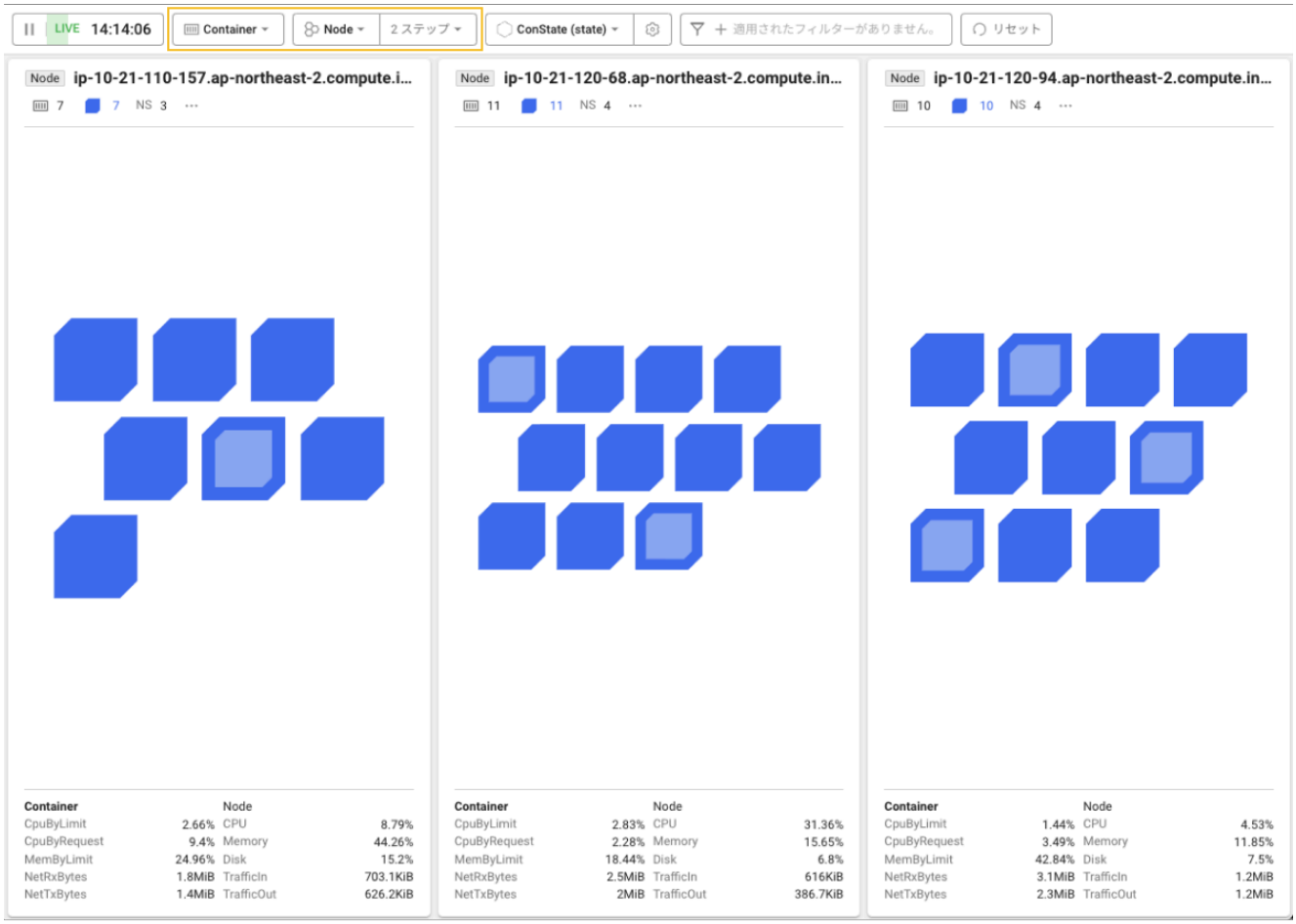
ユーザーの興味に応じた様々なグループ化

基本的なコンテナマップメニューの画面を見ると、すべてのコンテナやPodが展開されます。ただし、Kubernetesのコンテナ/Podを観察する場合、特定の基準に基づいて対象を選択してモニタリングすることが良くあります。もちろん、対象を見る基準は非常に様々です。Kubernetes環境についてユーザーの興味がすべて同じではないからです。

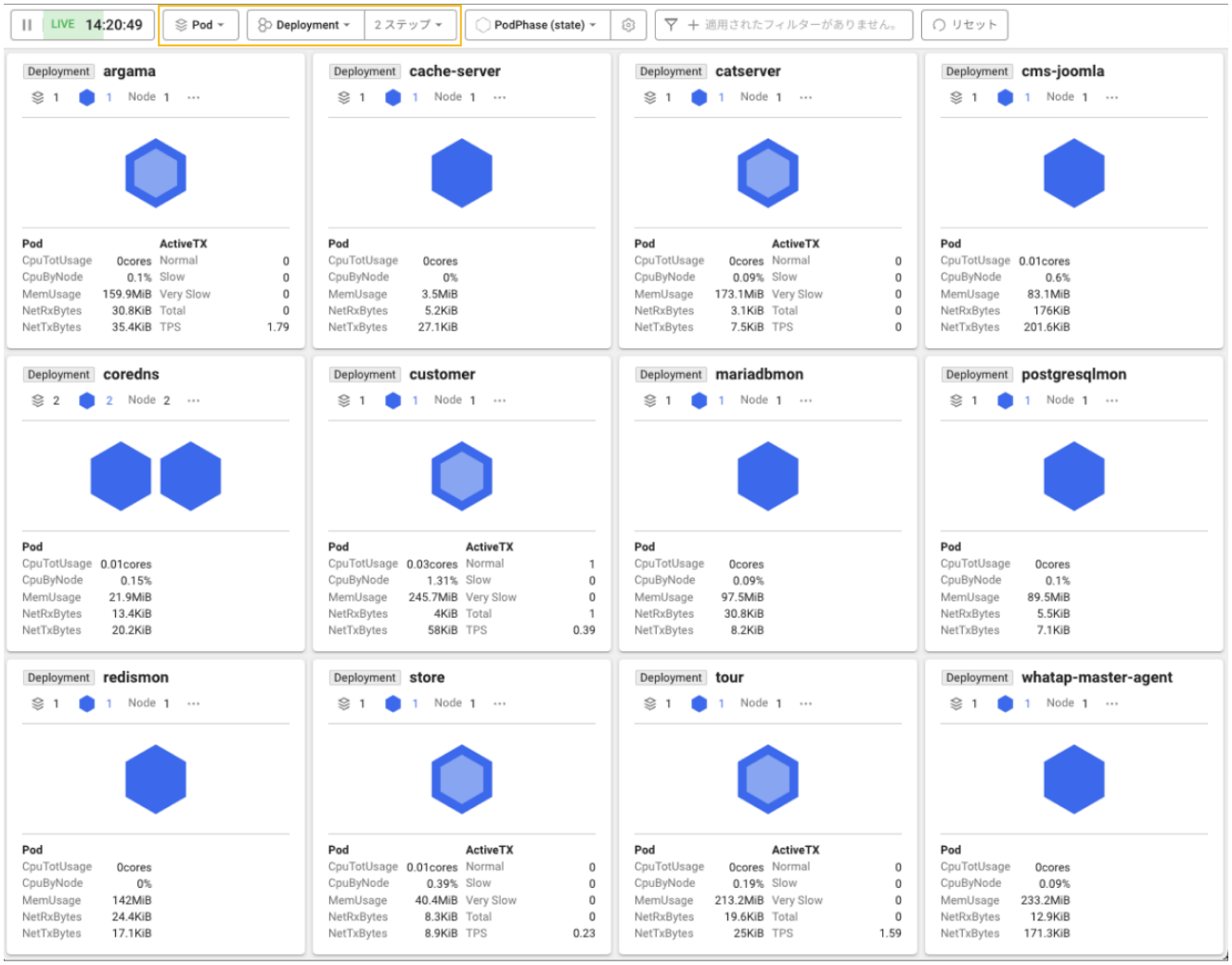
サーバー担当者は、Kubernetesリソースを実行するNodeの状態に関心があります。一方で、特定サービスのアプリケーション担当者は、Deploymentや個別のPod、あるいはコンテナの状態に関心があります。新規バージョンのデプロイが必要な場合は、全体的なDeploymentの状態と新しいイメージを使用するコンテナをモニタリングする必要があります。

そのため、これらの様々なユーザーのニーズを満たすために提供する機能は**グループ化**です。

- **コンテナ基準ビュー Nodeグループ化**



- **Pod基準ビュー + Deploymentグループ化**



主なグループ化の属性

コンテナマップのビューによって対応するグループ化の属性は次のとおりです。

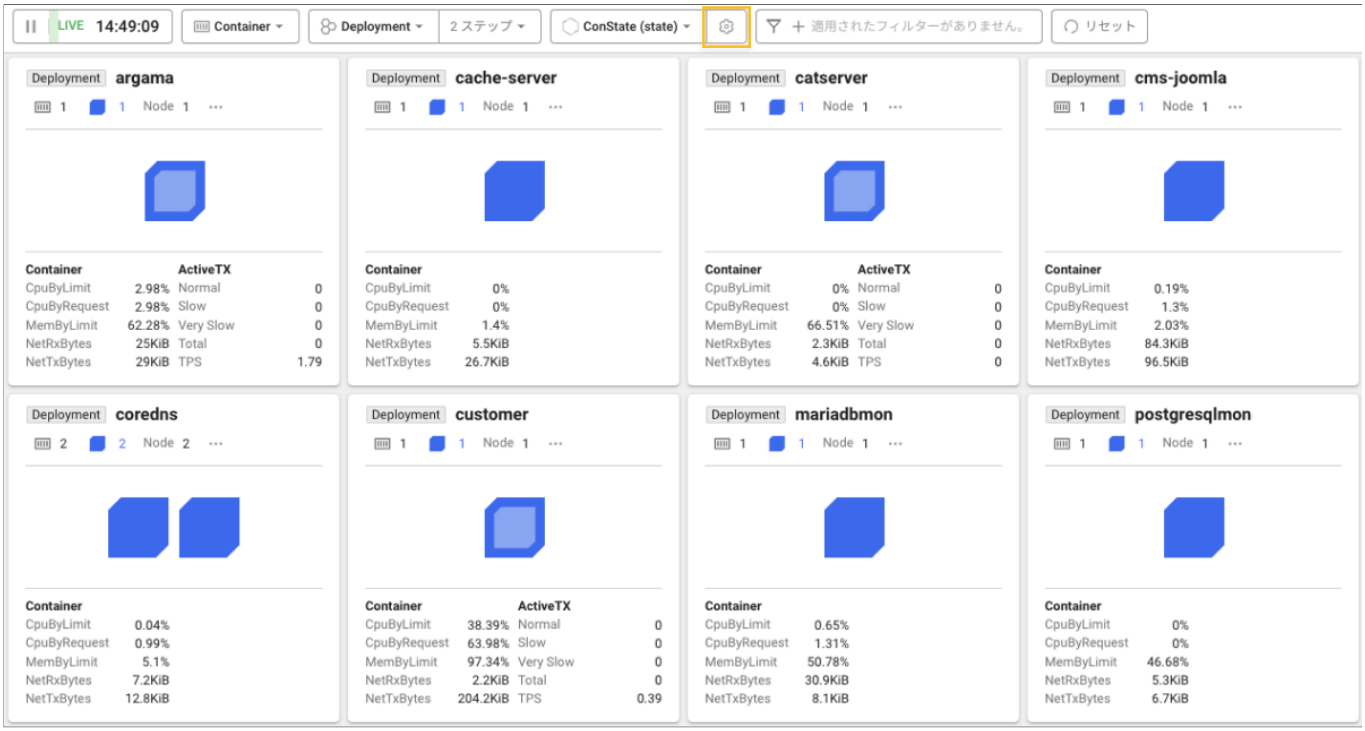
標準ビュー	グループ化属性
Container	未設定(All) Node Namespace Deployment

標準ビュー	グループ化属性
	ReplicaSet ReplicationController Pod ContainerName Image
Pod	未設定(All) Node Namespace Deployment ReplicaSet ReplicationController DemonSet StatefulSet

> 画面で設定したグループ化設定(プロパティフィールドおよびフィルタリングオプションを含む) 情報は、ユーザーのブラウザ内に保存されます。保存された設定を初期化したい場合は、リセットアイコン**リセット**ボタンをクリックします。

グループ化 & カード

コンテナマップは、1つ以上のカードで構成されています。グループ化を適用されていない場合、プロジェクト内の全体コンテナ/Podが1つの大きなカードに表示されます。グループ化を適用する場合は、グループごとにカードを構成します。



上の画面は、Deploymentを基本属性としてグループ化を適用したものです。各Deployment単位でカードが一覧されていることがわかります。

グループ化を適用する場合、基本属性でグループ化できない対象は基本的に表示されません。ただし、画面に表示できる方法があります。画面上部の⚙設定ボタンをクリックします。不明なグループを非表示のチェックボックスを解除するとグループ化に対応しない対象のみを集めて個別のカードとして表示されます。

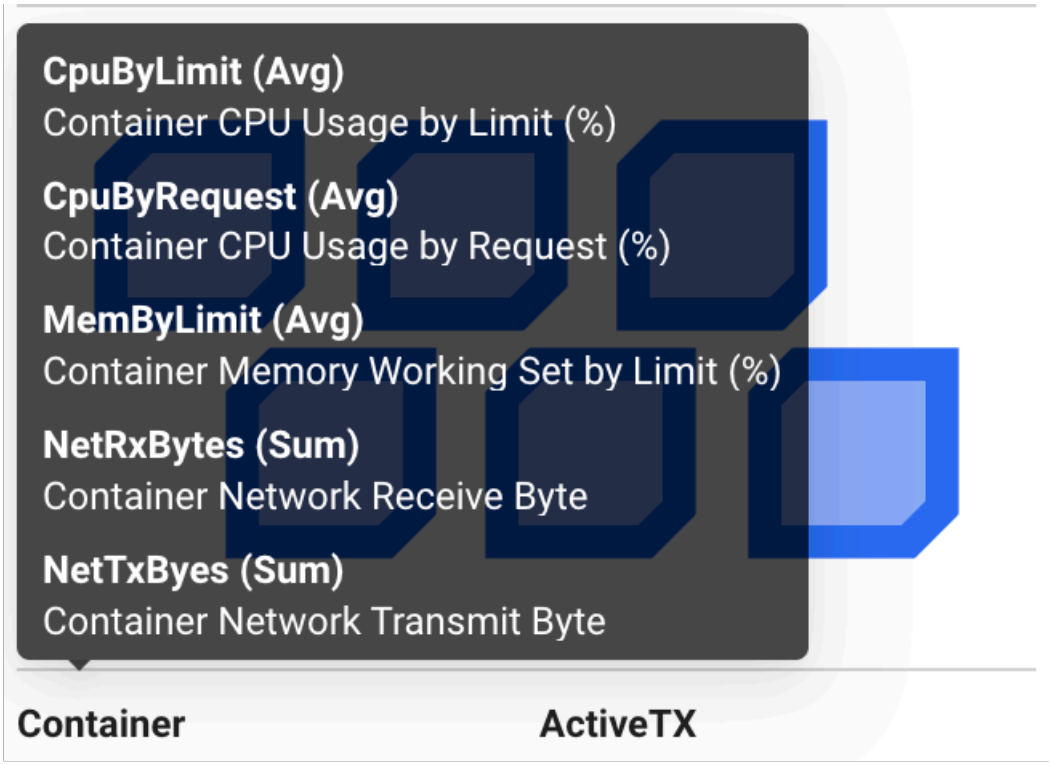
カード別メトリクスデータ

各カードの下には、グループ化された対象に関するメトリクスデータを表示します。以下は、提供する指標の情報です。

標準ビュー	リソース指標	アプリケーションメトリクス
Container	CpuByLimit CpuByRequest MemByLimit NetRxBytes	Normal TX Slow TX Very Slow TX Total TX

標準ビュー	リソース指標	アプリケーションメトリクス
	NetTxBytes	TPS
Pod	CpuTotUsage CpuByNode MemUsage NetRxBytes NetTxBytes	

ユーザーのカーソルがContainer(またはPod)、ActiveTXの上に置かれている場合は、各メトリクスに関する説明をポップアップで確認することができます。

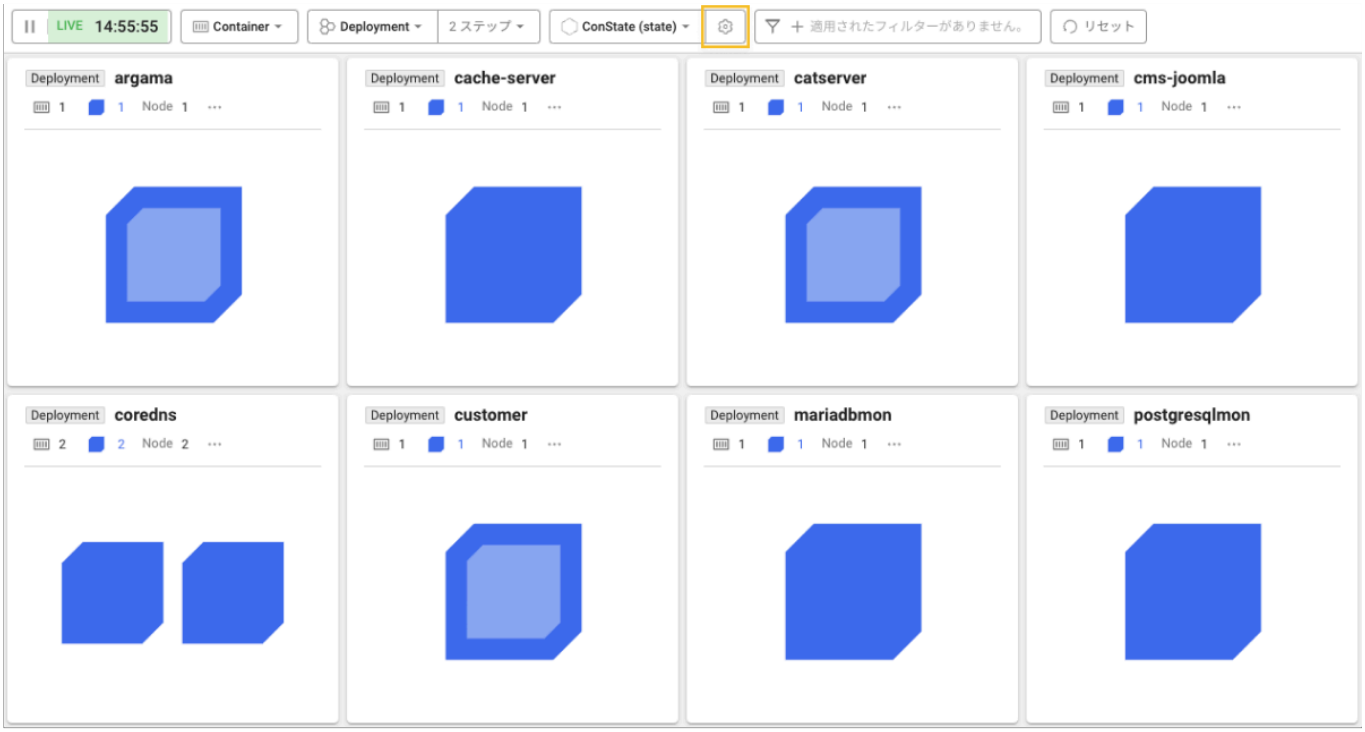


- ポップアップに表示された指標のうち(Avg)が付いている項目は、カードの下に表示された同じ名前の指標の数値がグループ内の個々の対象が持つ値の平均値であることを意味します。
- ポップアップに表示された指標のうち(Sum)が付いている項目は、カードの下に表示された同じ名前の指標の数値がグループ内の個々の対象が持つ値の合算値であることを意味します。

カードのメトリクスデータ非表示

カードの下にメトリクスデータを非表示にしてチャートのみ焦点を当てることができます。画面上部の⚙設定ボタンをクリックします。カードの下部にあるメトリクスを表示チェックボックスを解除してください。この機能は、カード数が多い場合に便利です。

次のように、メトリクス データを非表示にすることができます。

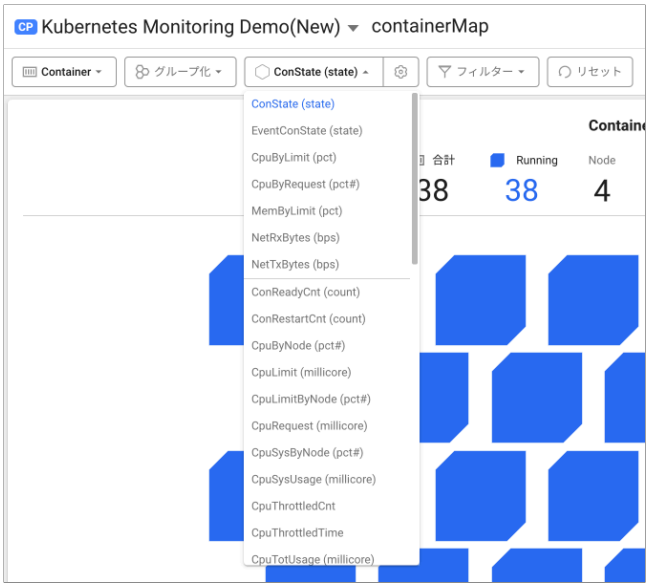


コンテナマップの指標と状態

チャート基準のカストマイズ

WhaTap Kubernetesモニタリングサービスのコンテナマップメニューで提供する基本チャート属性は、コンテナまたはPodの状態(State)です。メトリクス名ではConState(Container State)です。ただし、基本状態以外の基準でマップを構成する必要があります。例えば、基本State状態ではなくCPU使用率(millicore)、メモリ使用率(%)、ネットワーク使用率(byte)に基づいて各対象の状態を判断できます。

WhaTapは、コンテナのチャート基準変更機能を提供します。変更可能なチャートの基準指標は、WhaTap Kubernetesが収集するメトリクス(指標)と同じです。



> 画面で設定したグループ化設定(プロパティフィールドおよびフィルタリングオプションを含む) 情報は、ユーザーのブラウザ内に保存されます。保存された設定を初期化したい場合は、リセットアイコン**リセット**ボタンをクリックします。

基準指標のしきい値設定およびステータス色

対象に割り当てられた基準指標のしきい値を設定することで、各対象の状態を色で分類できます。指標の単位が数値(pct、bps、millicore)の場合、しきい値を設定できます。

Normal、Warning、Critical などステータス別の色や意味は以下の通りです。



状態	説明
Normal	定常間隔の表示(~Warning未満)
Warning	Warningしきい値内にあることを表示(Warning以上~Critical以下)
Critical	Criticalしきい値内にあることを表示(Critical超過~)
Dead	コンテナ終了の進行状態
Wait	コンテナ開始の進行状態
Pause	コンテナの一時停止状態

イベントベースのコンテナ状態

コンテナマップメニューでコンテナビューは、`EventConState`というステータス値を表示します。`EventConState`は`Event-based Container State`でイベントが反映されたコンテナ状態を表します。`EventConState`に反映されたイベントの種類は次のとおりです。

イベント	レベル
Kubernetes Event	Warning
WhaTap Event	Warning, Critical

- `Kubernetes Event`は、Kubernetesが提供するイベント情報です。ユーザーのKubernetes環境で`kubectl get events`コマンドで表示可能なイベント情報と同じです。
- `WhaTap Event`は、ユーザーがWhaTapイベント機能を通じて設定するイベントです。

`EventConState`は、次のような方式で決定されます。

状態	状態値	原因
Critical	50	WhaTapマイクロエージェントでCriticalイベントの発生 WhaTapノードエージェントでCriticalイベントの発生 コンテナ状態がOOMKILLEDに切り替え
Warning	30	KubernetesでWarningイベントが発生 WhaTapマイクロエージェントでWarningイベントが発生 WhaTapノードエージェントでWarningイベントが発生 コンテナ状態がRESTARTINGに切り替わり
Dead	23	コンテナ状態がDEADに転換
Pause	22	コンテナ状態がPAUSEに転換
Waiting	21	コンテナ状態がWAITINGに転換

状態	状態値	原因
Normal	10	コンテナ状態がNormalに転換

WhaTapのイベントには、**イベント状態が解消されたら追加に通知する**を設定したイベントがあり、次のように2種類で区別します。

- 解決を追跡しないイベント：`stateless`イベント
- 解決時に追加の通知を設定したイベント：`stateful`イベント

コンテナで発生した`stateless`、`stateful`イベントは、`EventConState`の状態を次のように判断します。

- `stateless`イベント：コンテナで発生した直近の1分間の`stateless`イベントを`EventConState`状態に反映
- `stateful`イベント：発生から終了までの`stateful`イベントを`EventConState`状態に反映

外部プロジェクトの連携と設定

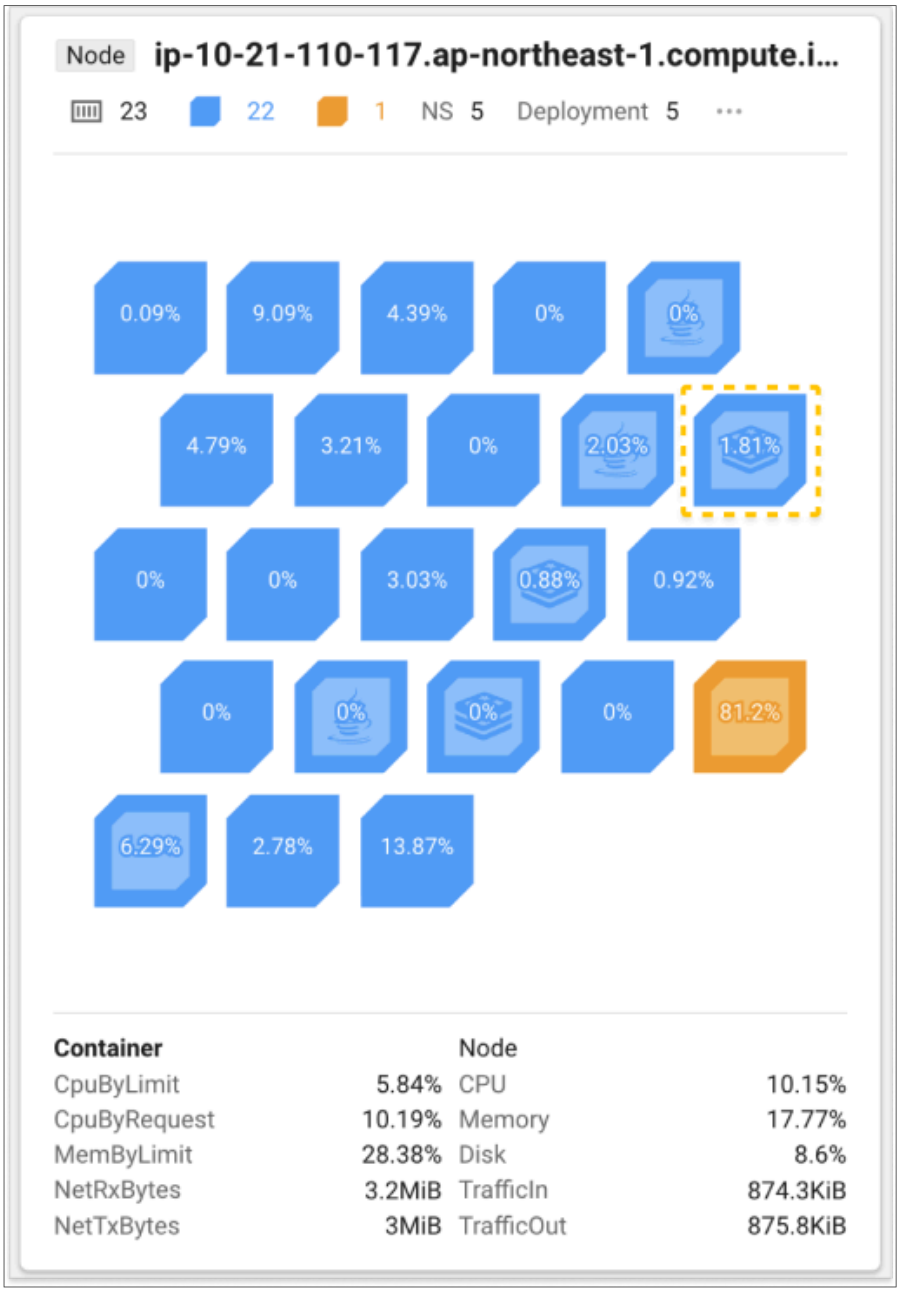
> WhaTap Kubernetesエージェント1.7.16バージョン以降が必要です。

WhaTapは、様々なモニタリングサービスをプロジェクト単位で提供します。 **外部プロジェクト連携**機能を通じてKubernetesプロジェクトのPodと関連した外部プロジェクトを連携できます。

外部プロジェクト連携のためにはKubernetesエージェント設定及び連携対象Pod設定が必要です。 Kubernetesプロジェクトと外部プロジェクトを連携させ、統合的なモニタリングシステムを構築します。

外部プロジェクト連携

外部プロジェクト連携機能を有効にすると、次のように連携されたコンテナ及びPodに当該プロジェクトロゴが表示されます。 これにより、どの外部プロジェクトと連携したのか視覚的に簡単に識別できます。 また、該当するブロックをクリックして、要約表示パネル内のボタンを通じて該当プロジェクトに移動できます。


Node
ip-10-21-110-117.ap-northeast-1.compute.i...
23
22
1
NS 5
Deployment 5
0.09%
9.09%
4.39%
0%
0%
4.79%
3.21%
0%
2.03%
1.81%
0%
0%
3.03%
0.88%
0.92%
0%
0%
0%
0%
81.2%
6.29%
2.78%
13.87%
Container
CpuByLimit 5.84%
CpuByRequest 10.19%
MemByLimit 28.38%
NetRxBytes 3.2MiB
NetTxBytes 3MiB
Node
CPU 10.15%
Memory 17.77%
Disk 8.6%
TrafficIn 874.3KiB
TrafficOut 875.8KiB

外部プロジェクト設定

エージェント設定

KubernetesエージェントのインストールYAMLファイルでwhatap-monitoringネームスペースのwhatap-node-agent DaemonSet設定を変更する必要があります。 特に、whatap-node-agentコンテナのenvセクションに次の環境変数を追加する必要があります。

```
- name: "collect_foreign_project"
  value: "true"
```

例

以下は、環境変数が追加されたYAMLファイルの例です。 実際の運営環境のYAMLファイルと多くの違いがある可能性があるため、例を参考にして必要な部分だけ修正してから使用します。 ハイライトされたラインが追加された環境変数です。

```
apiVersion: "apps/v1"
kind: "DaemonSet"
metadata:
  labels:
    name: "whatap-node-agent"
  name: "whatap-node-agent"
  namespace: "whatap-monitoring"
spec:
  selector:
    matchLabels:
      name: "whatap-node-agent"
  template:
    metadata:
      labels:
        name: "whatap-node-agent"
    spec:
      containers:
      - command:
        - "/data/agent/node/cadvisor_helper"
```

```
    - "-port"
    - "6801"
    env:
    - name: "NODE_NAME"
      valueFrom:
        fieldRef:
          fieldPath: "spec.nodeName"
    image: "whatap/kube_mon_dev:foreign-2"
    name: "whatap-node-helper"
    ports:
    - containerPort: 6801
      name: "helperport"
    resources:
      limits:
        cpu: "200m"
        memory: "350Mi"
      requests:
        cpu: "100m"
        memory: "100Mi"
    volumeMounts:
    - mountPath: "/rootfs"
      name: "rootfs"
      readOnly: true
    - mountPath: "/sys"
      name: "hostsys"
      readOnly: true
    - mountPath: "/dev/disk"
      name: "hostdiskdevice"
      readOnly: true
    - mountPath: "/run/containerd/containerd.sock"
      name: "containerddomainsocket"
  - command:
    - "/bin/entrypoint.sh"
    env:
    - name: "NODE_IP"
      valueFrom:
        fieldRef:
          fieldPath: "status.hostIP"
    - name: "NODE_NAME"
      valueFrom:
        fieldRef:
```

```
        fieldPath: "spec.nodeName"
    - name: "WHATAP_LICENSE"
      value: "x6050rs80f4eg-x1g0egm9rjb3mg-z7p9533a2ea87e"
    - name: "WHATAP_HOST"
      value: "13.124.11.223/13.209.172.35"
    - name: "WHATAP_PORT"
      value: "6600"
    - name: "HOST_PREFIX"
      value: "/rootfs"
    - name: "WHATP_MEM_LIMIT"
      valueFrom:
        resourceFieldRef:
          containerName: "whatap-node-agent"
          resource: "limits.memory"
    - name: "collect_foreign_project"
      value: "true"
    image: "whatap/kube_mon_dev:foreign-2"
    name: "whatap-node-agent"
    ports:
    - containerPort: 6600
      name: "nodeport"
    resources:
      limits:
        cpu: "200m"
        memory: "350Mi"
      requests:
        cpu: "100m"
        memory: "300Mi"
    volumeMounts:
    - mountPath: "/rootfs"
      name: "rootfs"
      readOnly: true
    - mountPath: "/bin/entrypoint.sh"
      name: "start-script-volume"
      readOnly: true
      subPath: "entrypoint.sh"
    - mountPath: "/whatap_conf"
      name: "whatap-config-volume"
  initContainers:
  - command:
    - "/data/agent/tools/whatap_debugger"
```

```
    - "run"
    image: "whatap/kube_mon_dev:foreign-2"
    name: "whatap-node-debug"
    volumeMounts:
    - mountPath: "/rootfs"
      name: "rootfs"
      readOnly: true
  serviceAccount: "whatap"
  tolerations:
  - effect: "NoSchedule"
    key: "node-role.kubernetes.io/master"
  - effect: "NoSchedule"
    key: "node-role.kubernetes.io/control-plane"
  volumes:
  - hostPath:
      path: "/"
    name: "rootfs"
  - hostPath:
      path: "/sys"
    name: "hostsys"
  - hostPath:
      path: "/dev/disk"
    name: "hostdiskdevice"
  - configMap:
      defaultMode: 448
      name: "node-start-script"
    name: "start-script-volume"
  - emptyDir: {}
    name: "whatap-config-volume"
  - hostPath:
      path: "/run/containerd/containerd.sock"
    name: "containerddomainsocket"
```

連携対象Podの設定

連携しようとするPodの設定を修正します。 例えば、Redis Podを連携させる場合、Redisコンテナのenvセクションに次の環境変数を追加する必要があります。

```
- name: "license"
```

```
  value: "here_is_whatap_license"
- name: "whatap_platform"
  value: "redis"
```

- here_is_whatap_license：外部プロジェクトのライセンス情報を入力します。
- whatap_platform：連携しようとしている外部プロジェクトのプラットフォームを入力します。

> ⓘ 現在、外部プロジェクトの連動に対応しているプラットフォームは次のとおりです。 プラットフォーム入力時、大文字と小文字を区別せずに入力できます。
>
> **Redis、Java、Node.js、Python、PHP**

例

以下は、Redis Pod設定に環境変数を追加したYAMLファイルの例です。 ハイライトされたラインが追加された環境変数です。

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: redis-cluster
  namespace: backend
spec:
  serviceName: redis-cluster-svc
  replicas: 6
  selector:
    matchLabels:
      app: redis-cluster
  template:
    metadata:
      labels:
        app: redis-cluster
    spec:
      containers:
      - name: redis
        image: redis:5.0.1-alpine
        ports:
        - containerPort: 6379
```

```
    name: client
  - containerPort: 16379
    name: gossip
  command: ["/conf/update-node.sh", "redis-server", "/conf/redis.conf"]
  env:
  - name: POD_IP
    valueFrom:
      fieldRef:
        fieldPath: status.podIP
  - name: license
    value: x6050s081i709-z2lhh7gpi3bdr4-x7jovr61kumi5k
  - name: whatap_platform
    value: redis
  volumeMounts:
  - name: conf
    mountPath: /conf
    readOnly: false
  - name: data
    mountPath: /data
    readOnly: false
```

外部プロジェクトの連携確認

分析 > メトリクス照会



外部プロジェクトの連携設定後、**メトリクス照会**メニューに移動します。 例の画像のように**カテゴリ選択**ボタンをクリックします。

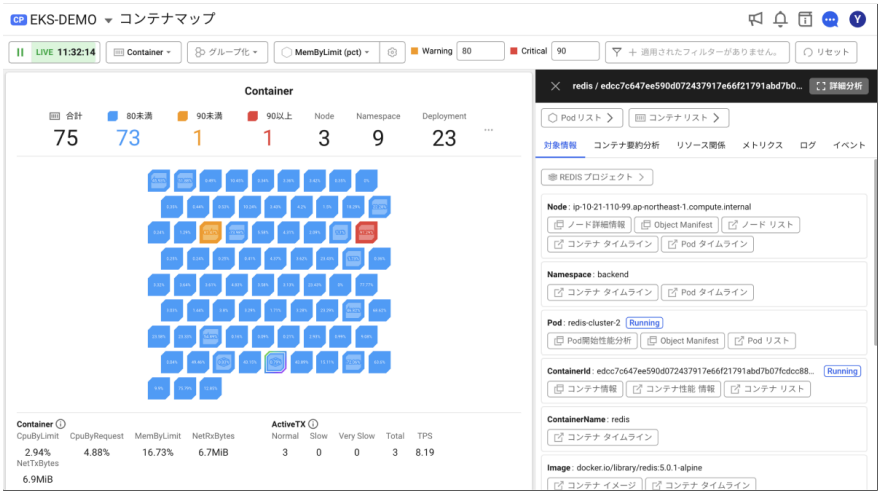
次のカテゴリが表示されたら、連携が正常に適用された状態です。

`container_foreign_project_mapping`

コンテナマップ詳細分析

コンテナマップメニューで選択した対象についての詳細情報を、**詳細分析**画面で確認できます。

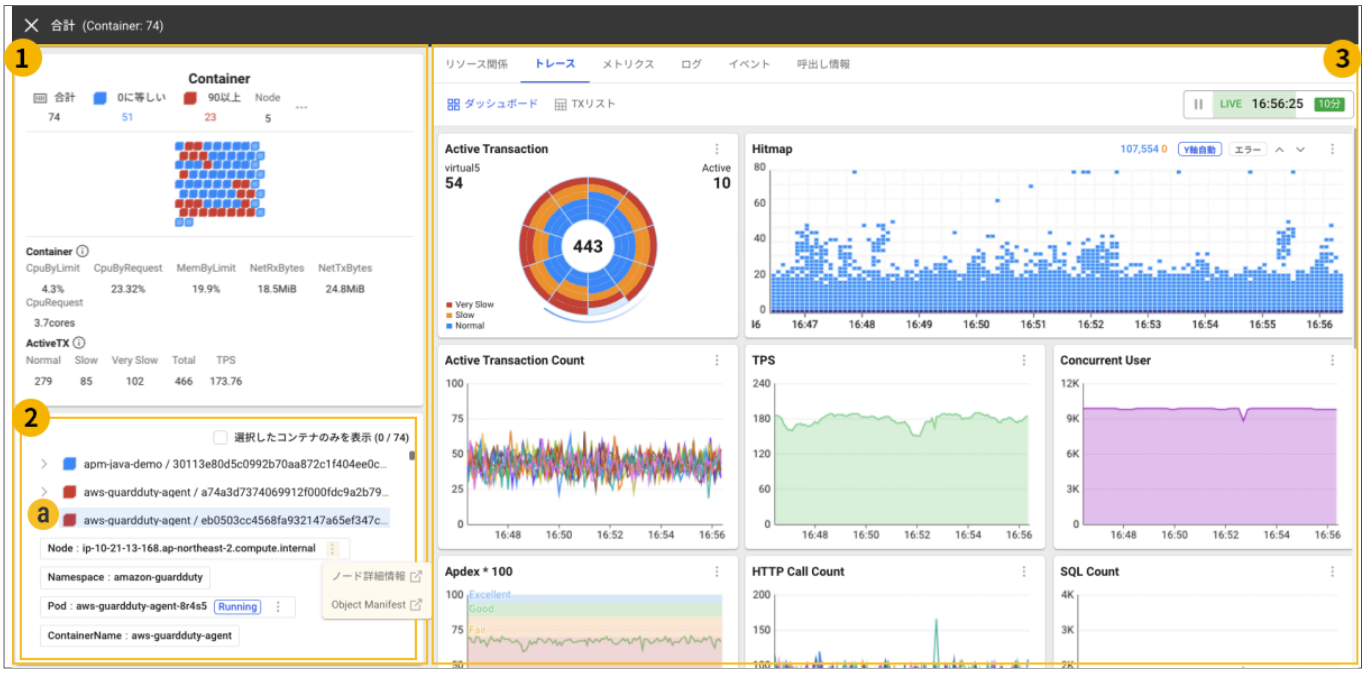
1. コンテナマップメニュー画面で表示する特定のブロック(コンテナまたはPod)を選択します。
2. 画面の右側に、次のように**概要ビュー**パネルが表示されます。



3. **概要を表示**パネル右上の**詳細分析**アイコンを選択すると、より詳細な情報を確認できます。

基本画面ガイド

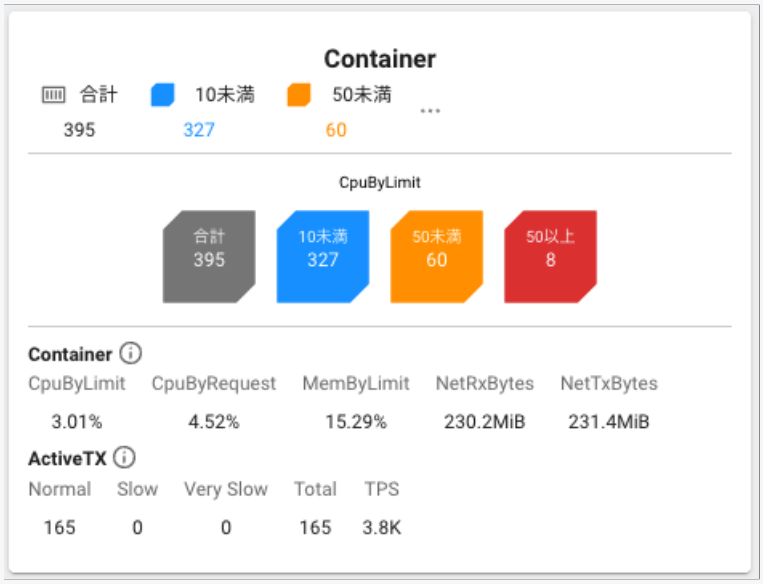
コンテナマップ**詳細分析**画面は、次のように構成されています。基本情報は、カードと一覧形式で確認できます。核心モニタリングデータを照会できるようにトレース、メトリクス、ログ、イベントタブを提供します。



- 1 **概要情報カード** (コンテナまたはPod)

  選択した対象のグループ情報を確認できます。対象を追加で選択すると、3領域から対象の指標が確認できます。コンテナマップで選択したカードと同じビューが表示されます。詳細については、次の文書を参照してください。

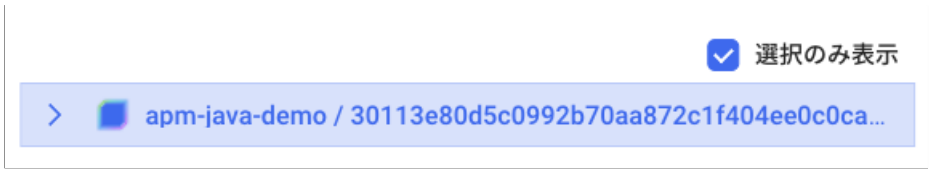
> ⓘ 集計情報カード参照時にブロック数が多い場合、次のように表記方法を変更します。すべてのブロック、正常、アラート、警告状態ブロック数を確認して比較することができます。



- **2 対象情報一覧**

  選択した対象のグループ情報を一覧で確認できます。対象を追加で選択すると、3領域から対象の指標が確認できます。

  - 2一覧の上段の選択項目を見るチェックボックスを選択すると、次のように選択されたコンテナまたはPodのみ表示されます。

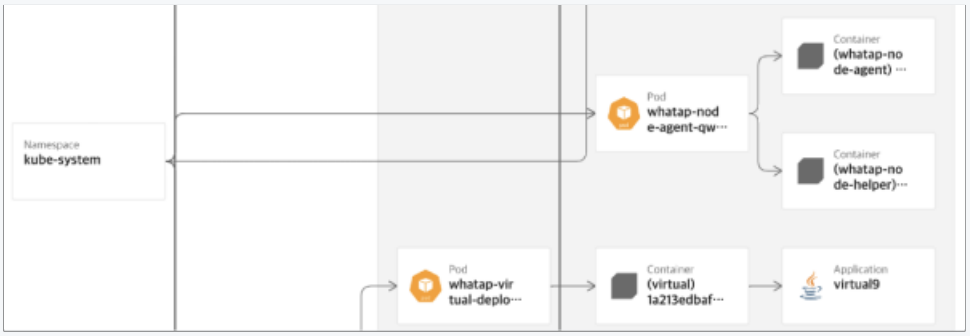
  - 一覧で選択した対象左側の > 詳細を見るアイコンを選択すると、a以下のようにNode、Namespace、Deployment、ReplicaSet、Pod、ContainerName、ContainerId、Image、Agentなどのメトリクス情報を照会できます。詳細については、次の文書を参照してください。

- **3 リソース関係、トレース、メトリクス、ログ、イベント、呼び出し情報**

  - リソース関係

    リソース関係タブを使用して、Kubernetesリソース間の接続関係を視覚的に確認できます。

選択したブロックに関するNamespace、Deployment、ReplicaSet(DaemonSet)、Pod、Container、Applicationなどの関係を表現したチャートを次のように提供します。



- トレース

  トレースタブを使用して、クバネティスコンテナ上で実行中のアプリケーションのトランザクション詳細をモニタリングできます。ダッシュボードビューとトランザクション一覧ビューを提供します。詳細については次の文書を参照してください。

- メトリクス

  メトリクスタブを使用して、選択されたコンテナまたはPodの主要なメトリクス情報とリソース使用パターンを確認できます。これにより、リソース関連の問題をすばやく特定し、最適化ポイントを見つけることができます。詳しい内容は次の文書を参照してください。

- ログ

  ログタブを使用してクバネティスコンテナ及びコンテナアプリケーションで発生するログを表示することができます。ログ機能は、コンテナのリアルタイムログ検索機能と特定の時点での履歴ログ検索機能に分けられます。詳細については次の文書を参照してください。

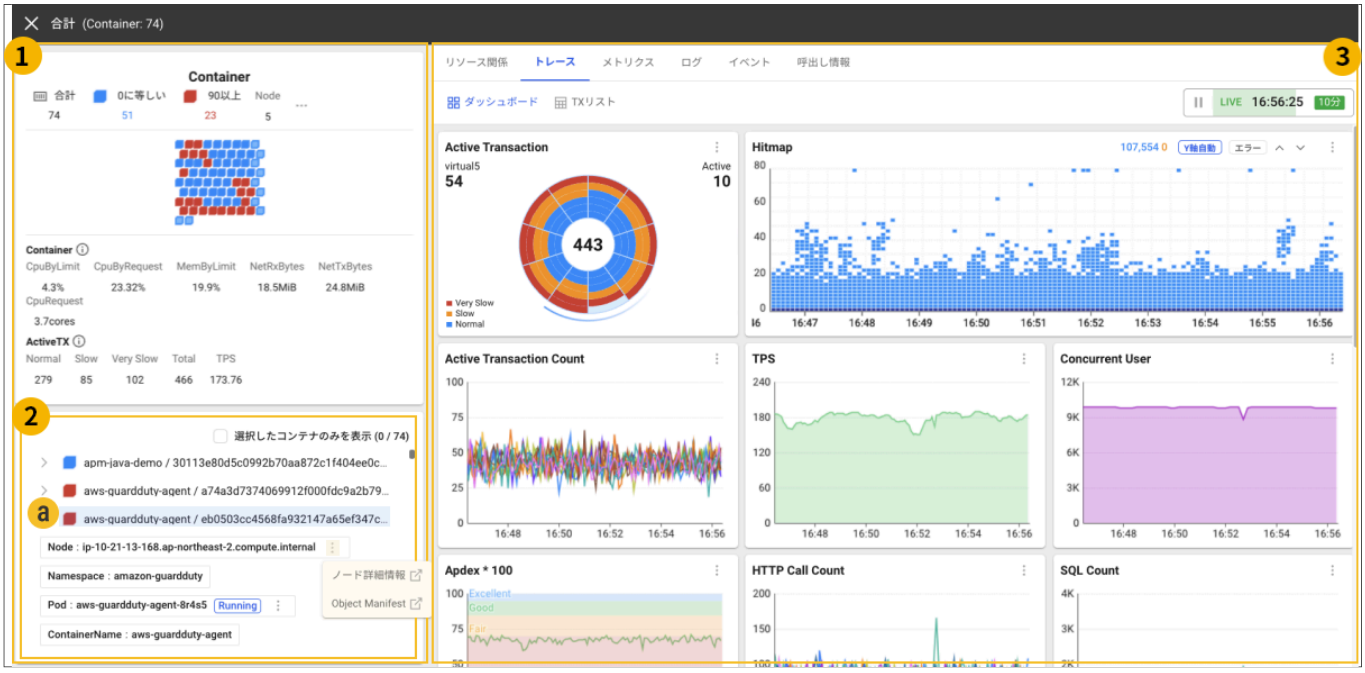
- イベント

  イベントタブでは、クバネティスクラスターで発生するKubernetesイベントとユーザーが設定した条件によるWhaTapイベントを表示することができます。詳細については次の文書を参照してください。

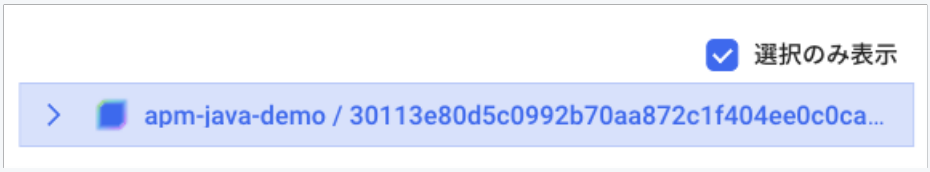
- 呼び出し情報

  呼び出し情報タブを使用してクバネティスコンテナ上で実行中のアプリケーションの呼び出しフローと外部呼び出し対象の状態を確認することができます。詳細については次の文書を参照してください。

Target Information



コンテナビューまたはPodビューでブロックは、各コンテナまたはPodを意味します。 1 **要約情報カード**で対象を追加で選択すると、 2 **対象情報一覧**と 3 領域の各情報タブから、選択した対象に関する情報を確認できます。

> ⓘ 2 **対象情報一覧**の上段にある選択のみ表示チェックボックスを選択すると、次のように選択されたコンテナまたはPodのみ表示されます。
>
> 
>

ブロック選択後、**詳細分析**画面の 2 **対象情報一覧**から、 a のように項目を開くと選択されたコンテナまたはPodの実行中のノード、所属されたネームスペース、Object Manifestなどの詳細な情報を確認したり、一覧メニューに移動できます。

- **Node**: ノード詳細、オブジェクトマニフェスト、Nod要約分析 情報を確認し、ノード一覧 メニューに移動できま

す。

- **Deployment**: Object Manifest 情報を確認し、Deployment リスト メニューへ移動できます。
- **ReplicaSet**: Object Manifest 情報を確認できます。
- **Pod**：Pod開始解析、オブジェクトマニフェスト、Pod要約分析情報を確認しPod一覧メニューに移動できます。
- **ContainerId**: **コンテナ情報**、コンテナ要約分析 情報を確認し、コンテナリスト メニューへ移動できます。
- **Image**: **コンテナイメージ**情報を確認し、コンテナイメージ メニューへ移動できます。
- **ServiceName**: Object Manifes 情報を確認し、Serviceリスト メニューへ移動できます。
- **Volume**：コンテナが使用するボリュームとマウント経路の情報を確認できます。
  - Podで使用するⓐボリュームの名前とⓑボリュームタイプを確認できます。

    例、**Volume** ⓐ `containerddomainsocket` ⓑ `HostPath`
  - **HostPath**ボリュームタイプを使用する場合ⓒホストシステムのファイルパスを確認できます。 このパスは、ファハドとホストシステム間のファイルシステムを接続するパスです。

    例、**Host Path** ⓒ `/run/containerd/containerd.sock`

> ⓘ
> - WhaTap Kubernetesエージェントが提供するボリュームタイプは次のとおりです。
>
> `HostPath` , `ConfigMap` , `Projected` , `EmptyDir` , `PersistentVolumeClaim` (PVC), `Secret`
> - Kubernetesで対応するボリュームのタイプと案内は、次のKubernetes公式文書で確認できます。

クバネティス状態の要約分析

クバネティス環境における各コンテナ、Pod、ノードのパフォーマンスとリソースに関する要約分析レポートを提供します。 クラスターとネームスペースのコンテナ、Pod、ノードのリソース割り当てと使用状況をリアルタイムでモニタリングし、レポート内の提案を参考にして効果的に対応できます。

> ⓘ 現在、ノード要約分析機能は韓国語のみで提供しています。

コンテナ要約分析

- 基本情報: コンテナの起動時間、再起動回数、Podおよびネームスペース情報を提供します。
- コンテナイメージ: 使用中のコンテナイメージと同じイメージを使用している他のコンテナの数およびリソース全体の使用量を確認できます。
- ノード情報: コンテナが実行されるノードのOS、IPアドレス、CPUコア数、メモリサイズなどのハードウェア情報を提供します。
- リソース使用状況: コンテナのCPUとメモリリソースの使用量、設定された限度(Limit)および要請(Request)の数と比較した使用率を提供します。
- サービスネットワーキング情報: 接続されたService及びIngress情報を確認できます。

Pod要約分析

- 基本情報: Podのクラスター名、ネームスペース、起動時間、現在の状況(Phase) などを確認できます。
- Kubernetesイベント: 最近のPodに関連するイベントログから問題の発生有無を確認できます。
- 実行情報: Podを生成したコントローラー(DaemonSet、Deploymentなど)を識別します。
- ノード情報: Podが実行中のノードのOS、カーネルバージョン、CPU、メモリなどの詳細な仕様を提供します。
- リソース使用状況: Pod内のすべてのコンテナのCPUとメモリ使用量、設定されたリソース限度(Limit)と比較した使用率を示しています。
- サービスネットワーキング情報: 接続されたService及びIngress情報を確認できます。

Nodeの要約分析

- 基本情報: ノードのクラスターバージョン、起動時間、現在の状態、スケジューリングの可能有無、内部IPアドレス、そしてリージョンとゾーン情報を提供します。
- オペレーティングシステムおよびハードウェア情報: ノードが実行されるオペレーティングシステム、OSイメージ、CPU アーキテクチャ、コア数、メモリの合計サイズを提供します。 ノードのメモリとCPUへの割り当ての可能有無と現在の使用量を確認できます。
- リソース使用状況: ノードで実行中のすべてのコンテナのリクエスト(Request)と割り当て可能リソース(CPU、メモリ)の違いを通じて、残りのリソースを把握します。 実際のCPU使用率とメモリ使用率を含め、リソースの使用状況を要約します。
- ディスク使用状況: ノードに接続されたディスク情報、ファイルシステムのタイプ、デバイスID、マウントポイン

ト、合計容量と使用中の容量、そして使用率を提供します。

- Podの状態とリソースの余裕: ノードの割り当て可能な全体のPod数と現在実行中のPod数を比較して、追加実行可能なPod数を確認できます。 また、MemoryPressure、DiskPressure、PidPressureなどの状態でノードの安定性をチェックします。

Trace

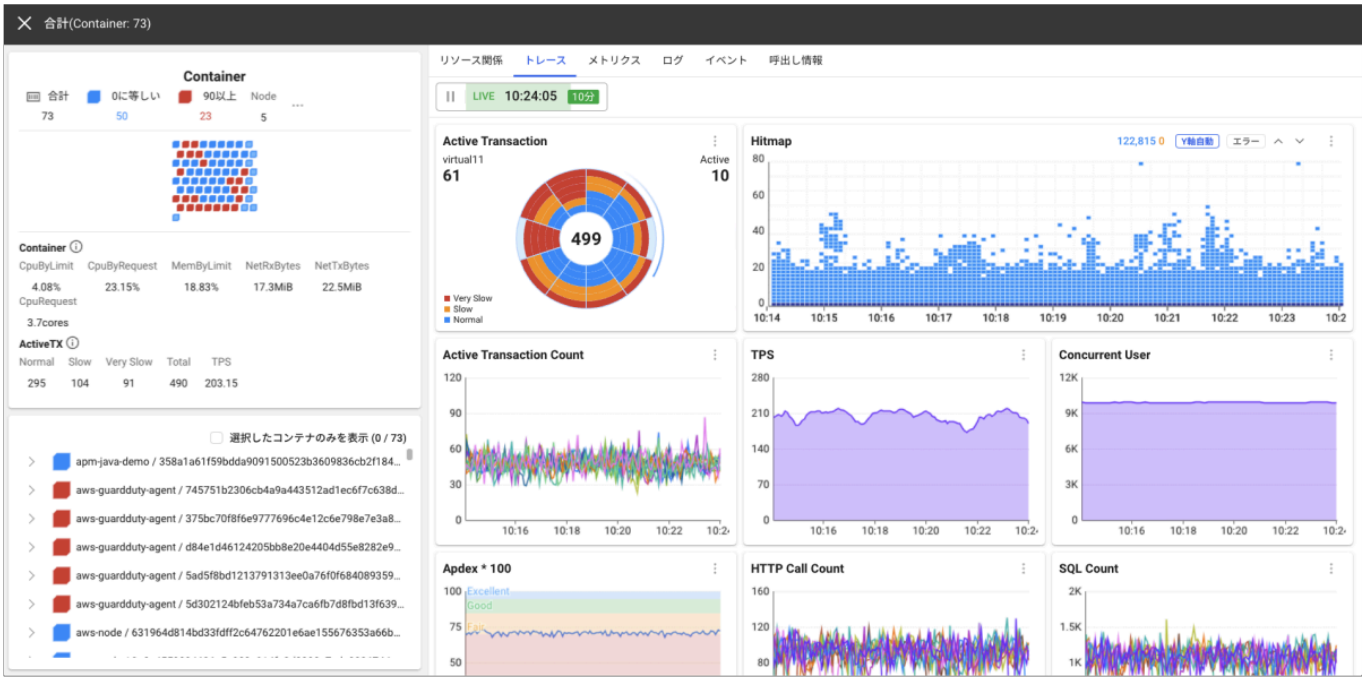
コンテナ化されたアプリケーショントレース

Kubernetesをモニタリングする場合、Kubernetes環境内で実行中のアプリケーションを追跡せずにクラスターをモニタリングするだけでは、現在の状態や問題の原因を特定することは容易ではありません。

問題の原因を追跡する場合、特定のトランザクションのSQL処理の遅延、ビジネスロジックのエラーなど、アプリケーションレベルの問題である場合が多いからです。

WhaTap Kubernetesのトレース機能は、コンテナ上で実行中のアプリケーション状態のモニタリング情報を提供します。

> ⓘ Kubernetes上で実行中のアプリケーションを追跡するには、Java、Node.js、Pythonなどの各アプリケーションコンテナにWhaTapモニタリングエージェントがインストールされている必要があります。エージェントインストールの詳細については、次の文書を参照してください。



ダッシュボード

ダッシュボードは、次のようなアプリケーションのコア情報を一目で確認できます。

- Active Transaction：現在実行中のトランザクションで、トランザクションの実行時間によって色分けされます。
  - 赤色(Very slow): 8秒以上
  - オレンジ色(Slow): 3秒~8秒
  - 青色(Normal): 0秒~3秒
- Active Transaction Count(アクティブTXの推移)
- Htimap(ヒットマップ)：終了したトランザクションレスポンス時間の分布図
  - 横軸：トランザクションの終了時間
  - 縦軸：実行時間
  - 実行時間の長いトランザクションは、分布図の上部にあります。
  - 特定の領域をドラッグすると、トランザクション情報を確認できます
- TPS(Transaction Per Second)：毎秒処理されたトランザクション数
- Conrurrent User：同時ユーザー
- Apdex * 100(Application Performance Index)：ウェブアプリケーションのユーザー満足度の測定
- HTTP Call Count：HTTP呼び出し件数
- HTTP Call Error Count：HTTP呼び出しエラー件数
- SQL Count
- SQL Error Count
- Transaction Total Conunt：トランザクション件数
- Transaction Error Conunt：トランザクションError件数
- Response Time：レスポンス時間
- Heap Memory Usage：Heap Memory使用量、使用可能な最大/現在のメモリ
  - 時間の経過に伴うメモリ使用量の変化をリアルタイムに把握可能
  - PythonのProcess Totalメモリ使用量

- ThreadCount：Thread件数、Node.js、Pythonを除く
- GC Count：GC回数、Node.js、Pythonを除く
- GC Old Generation Count：Old領域のGC回数、Node.js、Pythonを除く

トレース分析

ダッシュボード内Hitmapで照会を希望する区間をドラッグするとトレース分析を確認できます。トランザクション一覧から目的のトランザクションを選択した後、各トランザクションの下位のステップ情報を一度に確認できます。

> ⓘ トレース分析の詳細については 次の文書を参照してください。

Metrics

Kubernetesメトリクス

Kubernetesは、各コンテナのリソース使用量を制御し、継続的にモニタリングするために最適な状態を維持します。そのため、リソース使用量を確認し、そのデータに基づいてコンテナリソース使用量が適切に設定されていることを確認する過程は、Kubernetesモニタリングの基本的なパターンです。

CPUとMemoryメトリクス

代表的なKubernetesリソース設定は、コンテナのCPU Request/Limit、Memory Request/Limitです。以下は設定例です。

```
apiVersion: v1
kind: Pod
metadata:
 name: <Pod名>
spec:
 containers:
 - name: <コンテナ1名>
  image: <コンテナ1イメージ>
  resources:
   requests:
    memory: "64Mi"
    cpu: "100m"
   limits:
    memory: "128Mi"
    cpu: "200m"
 - name: <コンテナ2名>
  image: <コンテナ2イメージ>
  resources:
   requests:
    memory: "128Mi"
    cpu: "250m"
   limits:
```

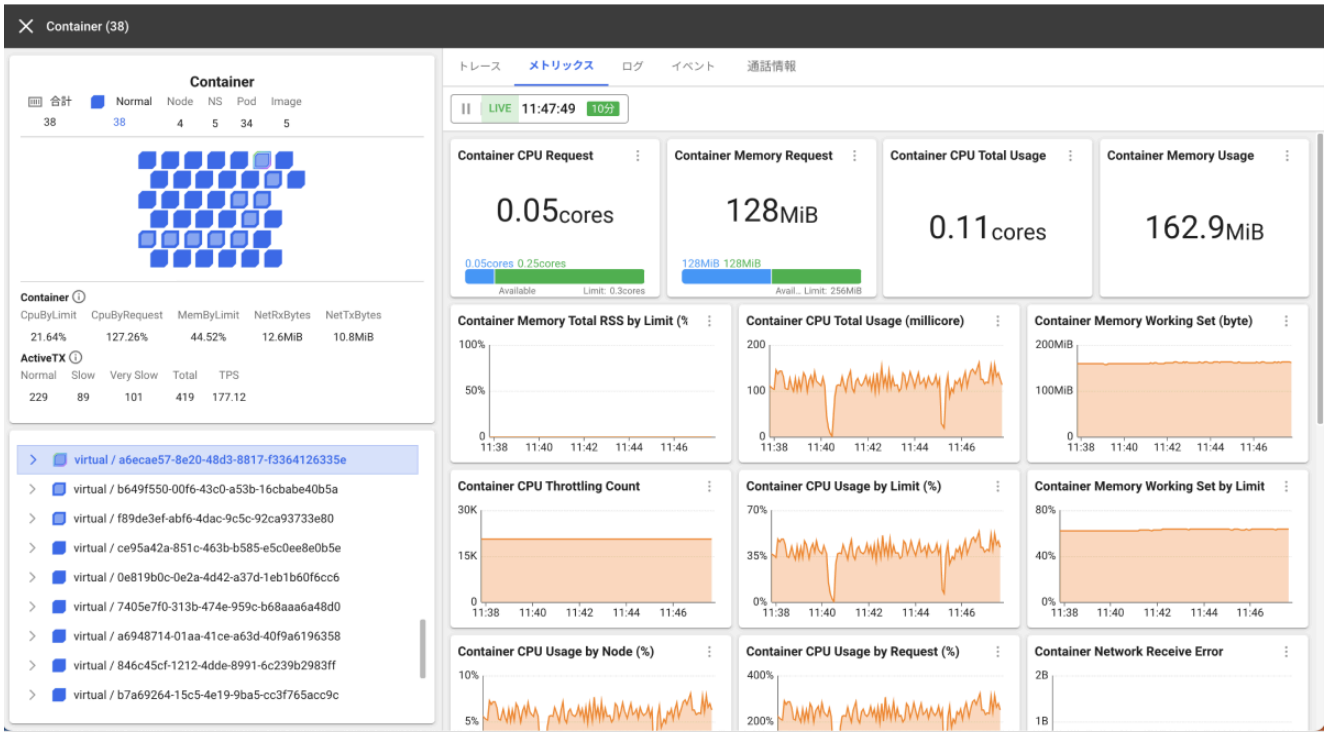
```
23      memory: "256Mi"
24      cpu: "500m"
```

上記の設定により、Pod内には2つのコンテナが存在します。各コンテナのリソース設定は次のとおりです。

対象コンテナ	リソース区分	リソース設定	値
コンテナ1	CPU	Request	100m
		Limit	200m
	メモリ	Request	64Mi
		Limit	128Mi
コンテナ2	CPU	Request	250m
		Limit	500m
	メモリ	Request	128Mi
		Limit	256Mi

様々なコンテナのリソース設定情報や現在の使用量を簡単に把握することは容易ではありません。そのため、WhaTap Kubernetesモニタリングサービスは、特定の単一コンテナ、ユーザーが選択した複数のコンテナ、またはDeployment全体を対象にリソース設定および現在の使用量を簡単に確認できる機能を提供します。

- **詳細を表示** 画面 > **メトリクス** - リソース使用量のモニタリングと分析



- 特定のコンテナを選択して状態を確認できます。



上記のデータを見ると、CPUはRequest 0.2colores(200m)、Limit 0.4colores(400m)に設定されています。現在のCPU使用量は0.02cores(20m)(5秒平均)です。右側のCPU関連の推移グラフを確認すると、全体的には余裕があるものの、断続的にCPU使用量が増加する現象が確認できます。

Memoryは、Request 128MiB、Limit 256MiBに設定されています。ところで現在のMemory使用量は184MiBです。右側のMemory関連推移グラフを見てもRequest基準で100%を超過し、Limit基準でも70%以上を使用しています。

CPU、Memoryに関するの主なWhaTapメトリクスは次のとおりです。

対象リソース	Shortname	Name	説明
CPU	CpuTotUsage	**Container CPU Usage (millicore)**	コンテナCPU使用率
	CpuByRequest	**Container CPU Usage by Request (%)**	コンテナCPU要求の使用率
	CpuByLimit	**Container CPU Usage by Limit (%)**	CPU Limit基準CPUの全体使用率
メモリ	MemWs	**Container Memory Working Set (byte)**	コンテナのメモリworking set
	MemWsByRequest	**Container Memory Working Set by Request (%)**	コンテナメモリの要求基準のworkingset使用量
	MemWsByLimit	**Container Memory Working Set by Limit (%)**	コンテナメモリ割当量基準のworking set使用量

ⓘ Podが実行中のノードに使用可能なリソースが十分である場合、コンテナに設定されたRequest以上のリソースを使用できます。そのためWhaTapのメトリクスデータで、**Container CPU Usage by Request (%)**と**Container Memory Working Set by Request (%)**の値が100%を超える可能性があります。

例えば、コンテナのMemory Requestを128MiBに設定したとします。コンテナ(Pod)が実行中のNodeの合計メモリは4GiBで、そのNodeに他のPodがない場合、コンテナは128MiB以上のメモリを使用できます。

その他のメトリクス

WhaTapはCPU、Memoryおよび、その他のメトリクスを提供します。

- **Container Block I/O Read Byte** (IoReadBytes)
- **Container Block I/O Write Byte** (IoWriteBytes)
- **Container Network Receive Byte** (NetRxBytes)
- **Container Network Transmit Byte** (NetTxByes)

WhaTap Kubernetesメトリクスの詳細については、次の文書を参照してください。

Log

Kubernetesログ

WhaTap Kubernetesモニタリングサービスは、Kubernetesで発生するログを収集して保存します。Kubernetesには次の2種類のログがあります。

- 各コンテナのStandard Out(標準出力) ログ
- 各コンテナの内で実行中のアプリケーションによって出力されたログ

これらのログは、コンテナの状態やメトリクス情報では確認しにくいモニタリングの盲点を補正する役割を果たします。このログを確認する方法は次のとおりです。

- 各コンテナのStandard Outログの場合は、コンテナを入力せずに、Kubernetesが提供する `kubectl logs` コマンドで確認できます。
- 各コンテナ内のアプリケーションログは、コンテナを入力して確認するか、個別のログ収集方法を確率する必要があります。

最近のログメッセージ (コンテナStdout／Stderr)

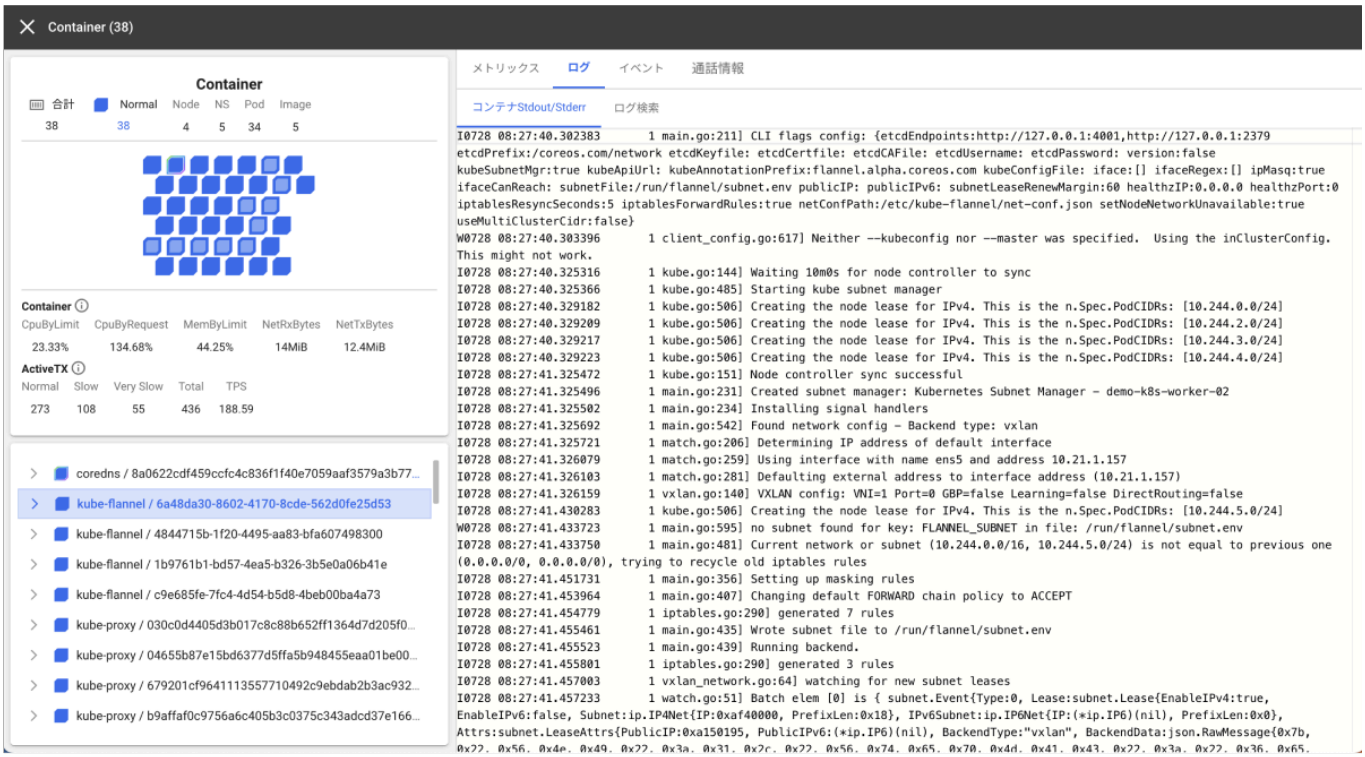
現在のコンテナログをすぐに確認できる機能です。クラスター環境から直接ログを取得して、選択したコンテナのリアルタイムログまたは選択したPod内アプリケーションのリアルタイムログを確認できます。通常、コンテナログを確認するには、次のコマンドを使用します。

```
kubectl logs pod/<Pod名> -n <Namespace名>
```

Pod内に2つ以上のコンテナがある場合は、個々のコンテナを選択してログを確認します。

```
kubectl logs pod/<Pod名> -c <コンテナ名> -n <Namespace名>
```

ログを確認する度にターミナル画面にアクセスしてコマンドラインツールを実行するのは非常に不便です。WhaTap Kubernetesモニタリングの**ログ**機能を使用すると、現在のログを簡単に確認できます。



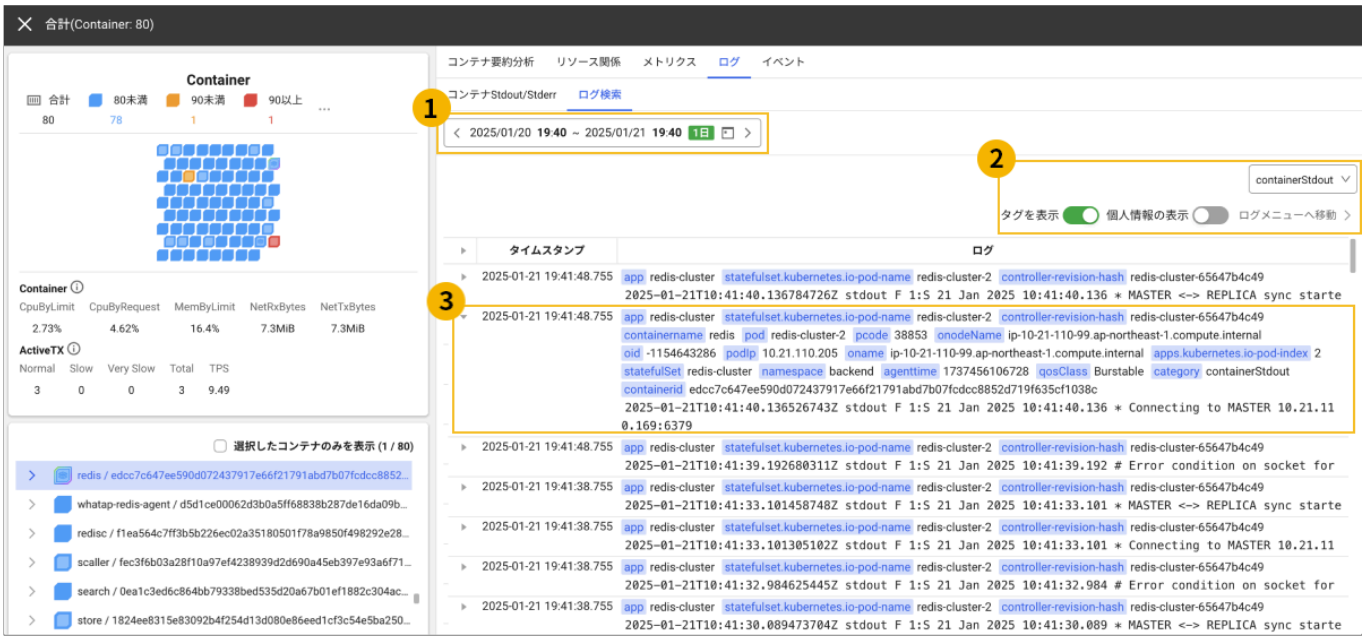
ただし、注意点があります。まず、コンテナを終了すると、ログが確認できなくなります。また、コンテナが実行中であっても、特定の期間または特定の時間のログだけしか確認できないデメリットがあります。WhaTapの最近のログメッセージ(**コンテナStdout/Stderr**) 機能は、Kubernetes環境からのログをWhaTapを通じて確認できる機能であり、ユーザー環境から消えたログは、WhaTapでは確認できません。

したがって、ログデータを安定的かつ継続的に管理するためには、別な管理方法が不可欠です。特にマイクロサービスアーキテクチャの採用が広がり、ビジネスを構成するコンテナ数が増加するにつれてログの検証と管理はますます複雑になっています。

良い方法はありますか？

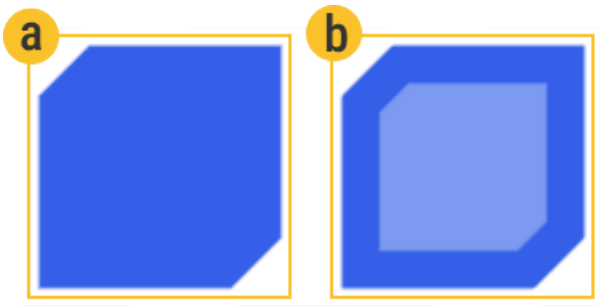
ログ検索

コンテナを終了すると、関連するログはすべて消えますが、**ログ検索**機能を使用して、選択したコンテナまたはPodで過去に発生したログを確認できます。コンテナStandardOutログを保存する場合、希望する保存期間が設定でき、特定の対象固有の情報の確認や相関分析をができます。



- 1時間セレクターを使用すると、データを表示するタイミングを選択できます。
- 2カテゴリオプションでカテゴリを指定して表示することができます。表示できるカテゴリの説明については、次のブロックとグループの手順を参照してください。
- 2 **タグを表示**トグルを無効にすると、ログ収集時のタグ情報を除く**タイムスタンプ**と**メッセージ**を確認できます。
- 2 **個人情報の表示**トグルを有効にすると、マスキングされたデータのソースデータを照会できます。**個人情報の非識別化**機能についての詳細は次の文書を参照してください。
- 2**ログ検索**ボタンをクリックすると、フィルター機能で詳細を確認できる**ログ** > **ログ検索**メニューが表示されます。
- 3 ▶ **詳細**アイコンを選択すると、パーシングされたキーと値のログとログメッセージを確認できます。詳細については、次の文書を参照してください。

一般ブロックとマイクロブロック



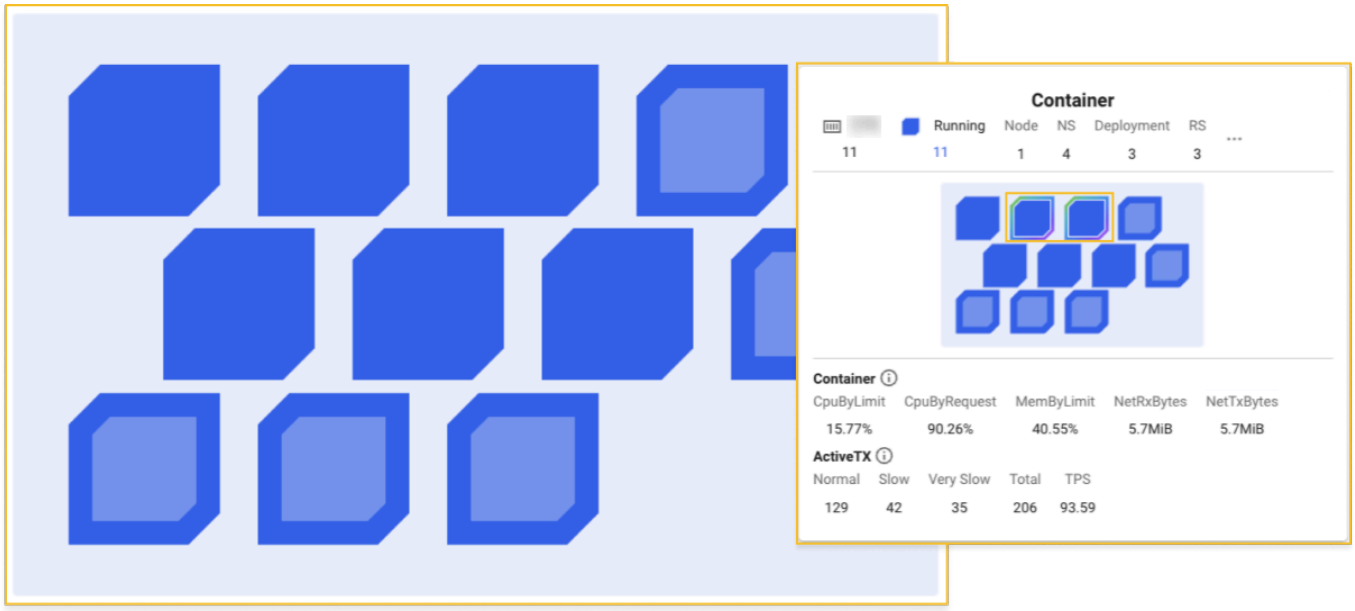
- a **一般ブロック**

  コンテナマップの1つのポリゴンをブロックといいます。ログの `containerStdout` カテゴリでは、 `containerid` に一致する情報を確認できます。

- b **マイクロブロック**

  エージェントがインストールされているブロックをマイクロブロックと呼びます。 `containerStdout` カテゴリだけでなく、 `AppLog` 、 `AppStdOut` 、 `AppStdErr` などの `App*` 関連カテゴリも確認できます。カテゴリを選択する場合、 `microOid` (ブロック内のエージェントid)が一致するログ情報を確認できます。

グループ



コンテナマップ画面で全体ブロックを選択した場合、または**詳細を見る**画面のカードでブロックを複数選択した場合、そのマルチブロックをグループと呼びます。グループを選択すると、**ログ検索**タブで `containerStdout` カテゴリと `App*` カテゴリのすべてのログを確認できます。

Standard OutのNodeエージェント設定の追加

WhaTapログサービスをコンテナのStandard Outputを保存するには、WhaTapクバネティスノードエージェントで `logsink_enabled=true` オプションを設定する必要があります。設定およびWhaTapログモニタリングサービスの詳細については、次の文書を参照してください。

コンテナ内部のアプリケーションログ収集用のアプリケーションエージェント設定の追加

WhaTapログサービスをコンテナ上で実行されるアプリケーションのログを収集するには次の文書を参照してください。Java、Python、Goをサポートしています。

> WhaTapログサービスを有効にして、コンテナのStandard Outputログ保存機能を使用する場合、それに伴う追加費用が請求されます。関連する料金の詳細については、次の文書を参照してください。

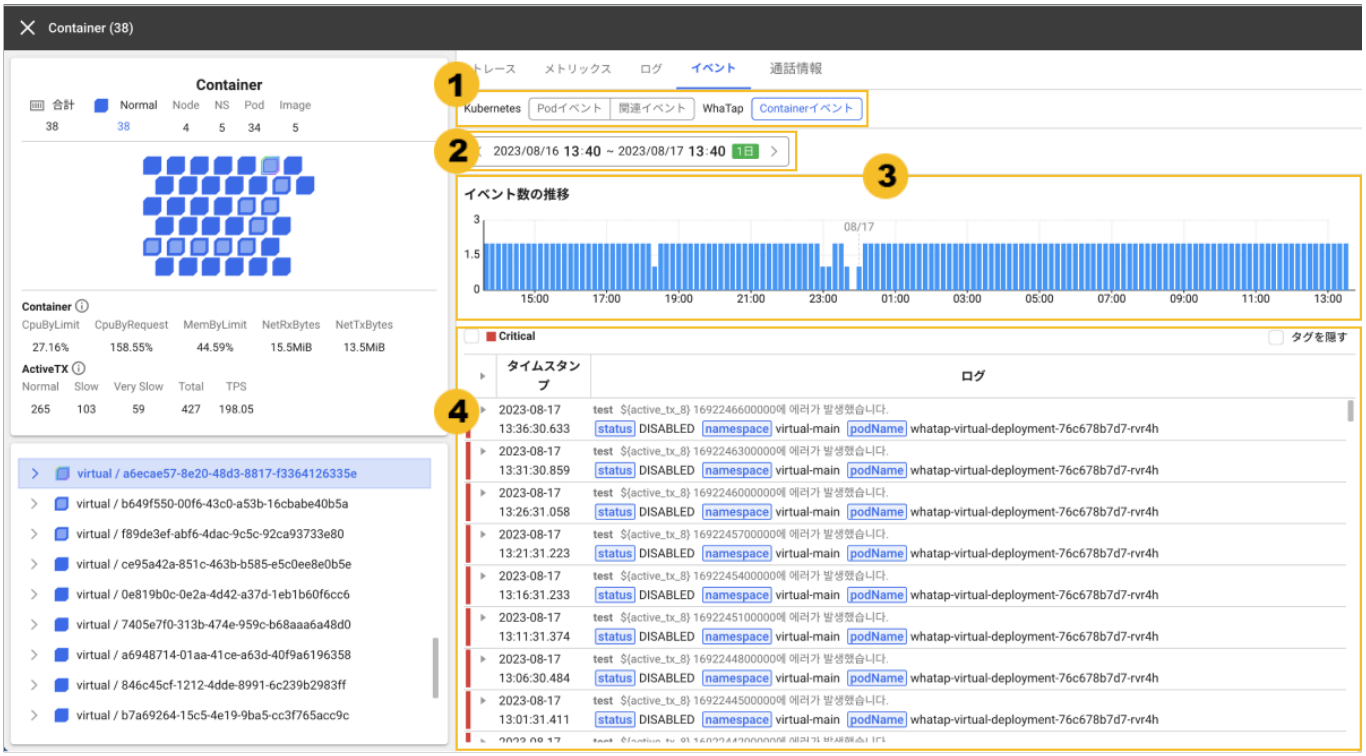
> 一部の古いバージョンのエージェントは、コンテナStandard Outputログ保存機能を提供しません。エージェントのアップグレード方法は、次のとおりです。
>
> Agent update
>
> ```
> kubectl set image daemonset/whatap-node-agent whatap-node-agent=whatap/kube_mon:<エージェントバージョン> -n whatap-monitoring
> ```
>
> WhaTap Kubernetesエージェントバージョンについては、 次の文書で確認できます。

Event

リソースのライフサイクルが発生するすべてのイベントの履歴を提供します。WhaTapクバネティスモニタリングで提供するイベント情報は、次の通りです。

- Kubernetesイベント(Kubernetes)
- WhaTapイベント(WhaTap)

基本画面ガイド



- 1 クバネティスイベント(Kubernetes)または、WhaTapイベント(WhaTap)を選択して、照会することができます。
  - 全体または複数ブロック選択

    コンテナマップおよび左のカードから全体または複数ブロック選択すると、Kubernetes全体イベントまたはWhaTap全体イベントを表示することができます。

- クラスタリングまたは個別ブロック選択

  コンテナマップ上段の２段階グループ化選択または個別ブロックを選択すると、該当イベントを照会できます。**グループ化**(クラスタリング)に関する詳細は次の文書を参照してください。

  > ⓘ **詳細**画面に進入する前コンテナマップ上段で選択したビューおよび対象グループ化によって照会できるイベントが異なります。

- 2時間セレクターを使用して、イベント履歴を表示したいタイムゾーンを指定できます。基本設定は1日です。
- 3選択した時間帯のイベント数推移を棒チャートを通じて確認できます。
- 4一覧形式でイベント履歴情報を確認できます。
  - イベント履歴一覧の右上のタグを非表示を選択すると、イベント履歴からタグ情報以外のタイムスタンプとメッセージが確認できます。
  - Kubernetesイベント
    - NormalまたはWarningの中で、照会をしたいイベントタイプを選択して確認できます。このイベントタイプはクバネティスが提供するイベントタイプと同一です。
    - 各イベントの左帯もまた、イベントタイプを示してます。グレー色はNormal、オレンジ色はWarning状態を意味します。
  - WhaTapイベント
    - Critical,Info,Warningの中で、照会をしたいイベントレベルを選択して確認できます。
    - 各イベントの左帯もまた、イベントレベルを指します。グレー色はNormal、オレンジ色はWarning、赤色はCriticalの状態を意味します。

クバネティスイベント

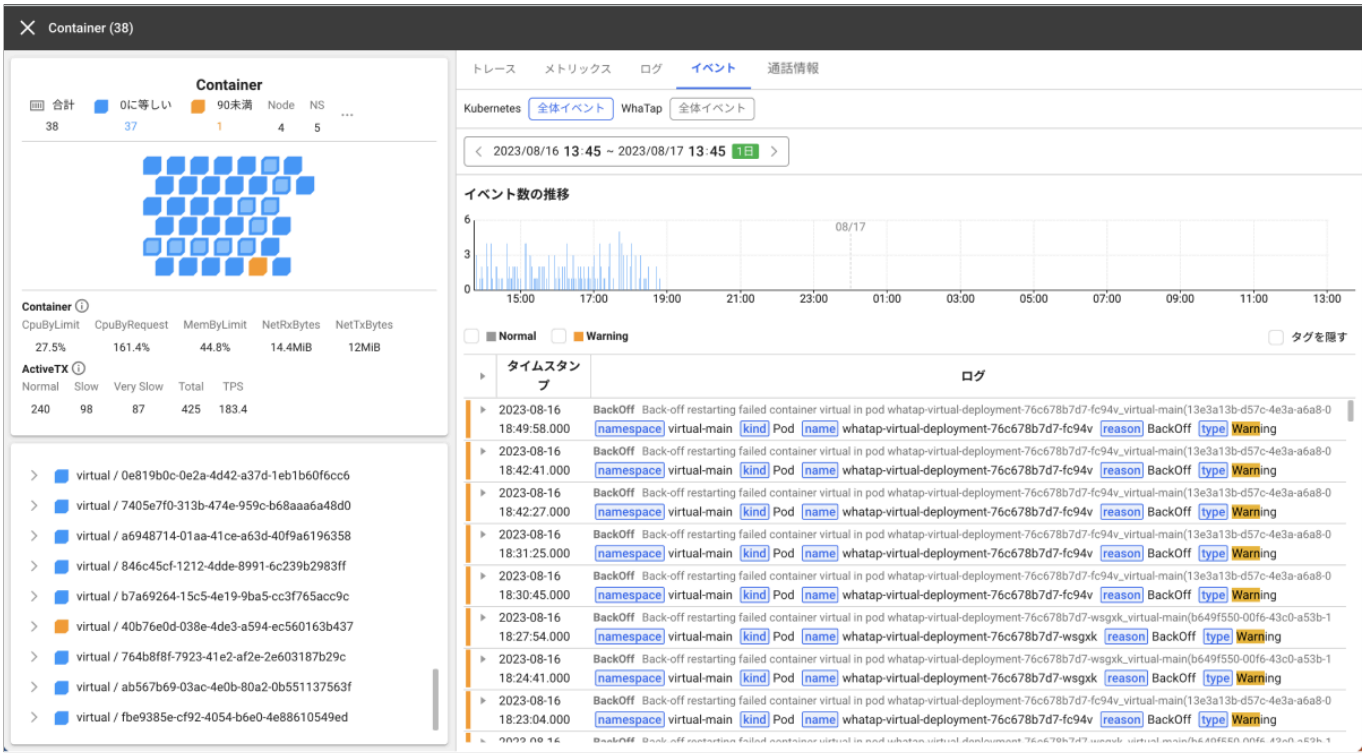
Kubernetesイベントは、Kubernetes自体によって提供されるイベント情報です。Kubernetes環境では、`kubectl get events`コマンドを使用して、Kubernetesクラスターのイベント情報が確認できます。

```
$ kubectl get events -A
NAMESPACE   LAST SEEN   TYPE    REASON          OBJECT              MESSAGE
```

```
 community  15m      Normal   ScalingReplicaSet     deployment/baekdusan          Scaled up replica set
baekdusan-59554d4859 to 5
 community  12m      Normal   ScalingReplicaSet     deployment/suraksan           Scaled down replica set
suraksan-66fc4bf889 to 3
 community  5m18s    Normal   ScalingReplicaSet     deployment/baekdusan          Scaled down replica set
baekdusan-59554d4859 to 3
 community  7m10s    Normal   Created               pod/suraksan-66fc4bf889-h2cw8    Created container suraksan
 community  6m56s    Normal   Pulled                pod/suraksan-66fc4bf889-h2cw8    Container image
"123456789123.dkr.ecr.ap-northeast-2.amazonaws.com/suraksan:0.48" already present on machine
 community  7m9s     Normal   Started               pod/suraksan-66fc4bf889-h2cw8    Started container suraksan
 community  2m7s     Warning  BackOff               pod/suraksan-66fc4bf889-h2cw8    Back-off restarting failed container
 community  7m12s    Normal   Started               pod/suraksan-66fc4bf889-t27rp    Started container suraksan
 community  6m54s    Normal   Pulled                pod/suraksan-66fc4bf889-t27rp    Container image
"123456789123.dkr.ecr.ap-northeast-2.amazonaws.com/suraksan:0.48" already present on machine
 community  7m12s    Normal   Created               pod/suraksan-66fc4bf889-t27rp    Created container suraksan
 community  2m46s    Warning  BackOff               pod/suraksan-66fc4bf889-t27rp    Back-off restarting failed container
 community  7m19s    Normal   ScalingReplicaSet     deployment/mountain           Scaled up replica set
mountain-6795bfbc54 to 5
 community  27m      Normal   ScalingReplicaSet     deployment/mountain           Scaled down replica set
mountain-6795bfbc54 to 4
 community  6m53s    Normal   Pulled                pod/suraksan-66fc4bf889-mjbsj    Container image
"123456789123.dkr.ecr.ap-northeast-2.amazonaws.com/suraksan:0.48" already present on machine
 community  7m10s    Normal   Created               pod/suraksan-66fc4bf889-mjbsj    Created container suraksan
 community  7m10s    Normal   Started               pod/suraksan-66fc4bf889-mjbsj    Started container suraksan
 community  2m10s    Warning  BackOff               pod/suraksan-66fc4bf889-mjbsj    Back-off restarting failed container
 community  5m18s    Warning  FailedToUpdateEndpoint  endpoints/baekdusan-svc       Failed to update endpoint
community/baekdusan-svc: Operation cannot be fulfilled on endpoints "baekdusan-svc": the object has been modified;
please apply your changes to the latest version and try again
```

ただ、これらのイベント情報は継続的に保存されません。したがって、一定時間が経過したイベントは、もう情報を確認できません。

WhaTap Kubernetesは、クラスター環境で発生したKubernetesイベント情報をすべて収集して保存します。次に、ユーザーが確認したい内容に応じて、Pod、ReplicaSet、Deployment、Namespaceなどのイベント情報を提供します。



各イベントの左帯はイベントタイプを示します。グレー色はNormal、オレンジ色はWarning状態を意味します。ユーザーの選択できるクバネティスイベントは、次の通りです。

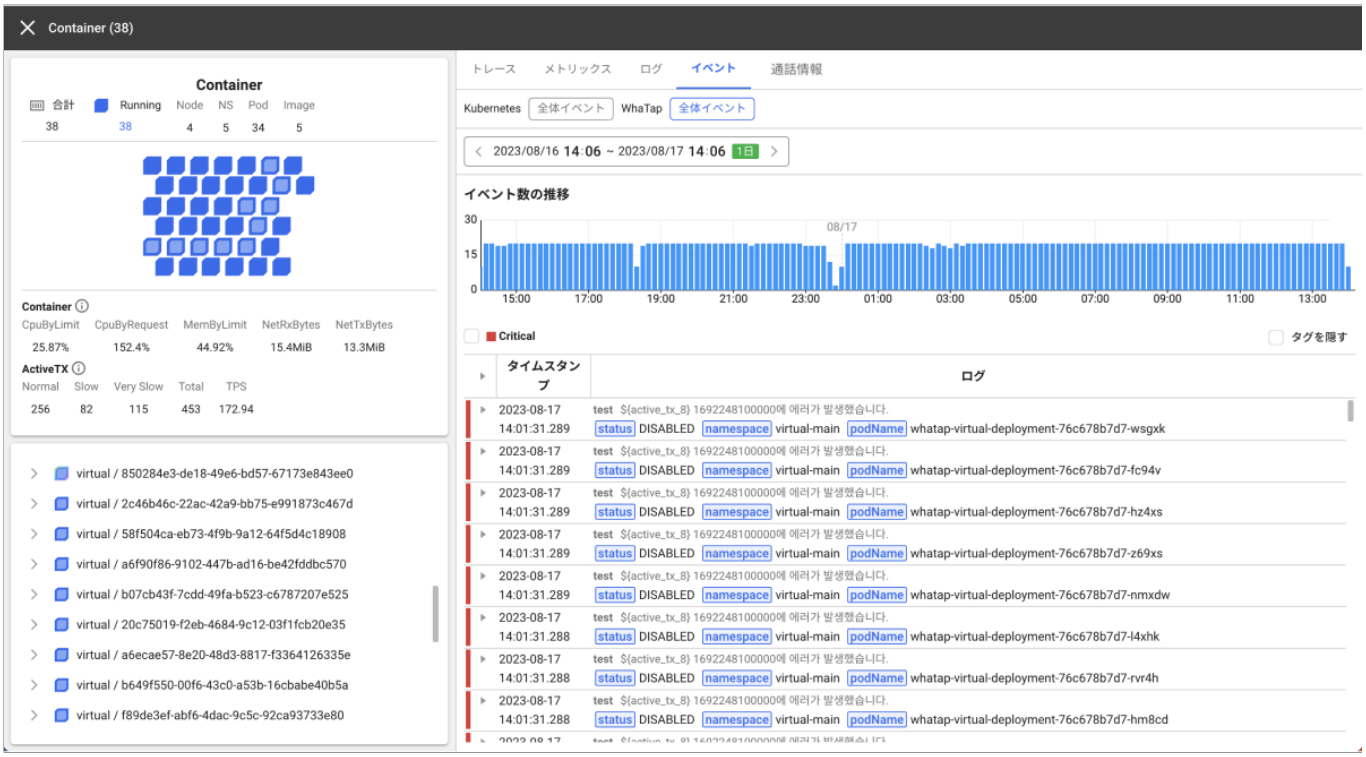
選択対象	提供イベント	提供関連イベント
全体(All)	プロジェクト全体イベント	
Namespace	Namespace下位の全体イベント	
Node	Nodeイベントまたはノード内のPod (コンテナ) 全体イベント	
Deployment	① kind は Deployment ② name は ユーザーの Deployment 名と一致	name がユーザーDeployment名でスタート
ReplicaSet	① kind は ReplicaSet ② name は ユーザーの ReplicaSet 名と一致	name はユーザーReplicaSet名でスタート

選択対象	提供イベント	提供関連イベント
Pod(コンテナ)	① `kind` はPod ② `name` はユーザーのPod名と一致	Podが所属するDeployment名でスタートまたはReplicaSet名でスタート

WhaTapに保存されているKubernetesイベントは、`message`、`namespace`、`kind`、`name`、`reason`, `type` などの項目で構成されています。

WhaTapイベント

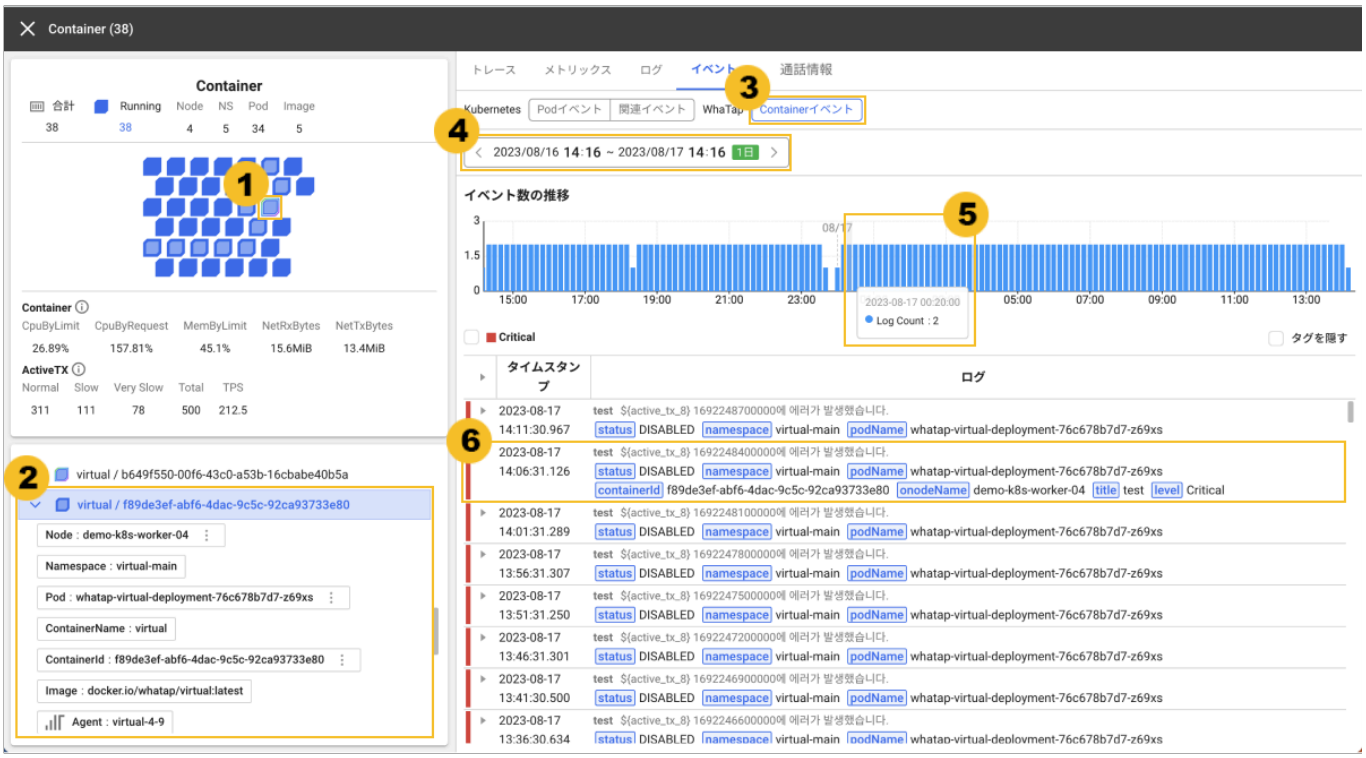
WhaTapイベントは、WhaTapイベント設定で登録したイベント条件によって発生するすべてのイベントです。コンテナマップ詳細画面のイベントタブでユーザーのWhaTapアラート設定によるイベント一覧を確認できます。



各イベントの左側の帯はイベントレベルを示します。グレー色はInfo、オレンジ色はWarning、赤色はCritical状態を意味します。ユーザーの選択によって使用可能なWhaTapイベントは以下の通りです。

選択対象	提供イベント
全体（All）	プロジェクト全体イベント
Namespace	Namespace下位の全体イベント
Node	`onodeName`がユーザーNode名と一致
Deployment	`deployment`はDeployment名と一致
ReplicaSet	`replicaSetName`がユーザーReplicaSet名と一致
Pod	`podName`がユーザーPod名と一致
Container	`containerId`がユーザーContainer IDと一致

WhaTapイベントは`message`、`status`、`title`、`level`などの項目で構成されています。次の例の画面はWhaTap Containterイベント履歴を検索しています。



- 1 **詳細**左側のサマリー情報カードから表示する対象を選択します。
- 2 サマリー情報一覧から選択対象の情報を確認してください。
- 3 イベントタブからWhaTapイベントを選択します。

  詳細画面にアクセスする前に、コンテナマップの上のビューオプションをコンテナビューに設定することで、Containerイベントを表示できます。

- 4 時間セレクターを使用して、表示する時間を選択します。基本設定は1日です。
- 5 棒チャートを使用して指定した時間帯のイベント数の推移を確認してください。

  指定時間帯を基準にイベント数の推移を見ると、イベントの２つがカウントされていることが確認できます。

- 6 イベント履歴一覧でイベント履歴情報を確認してください。リストの左側の帯からイベントのレベルを確認できます。イベントメッセージおよびタグ全体を確認するには ▾ アイコンを選択します。

  イベント履歴一覧で、`コンテナLimitの標準CPU使用量が85%以上`, `GC回数以上` Warningレベルのイベントを２つを表示できます。

Call Information

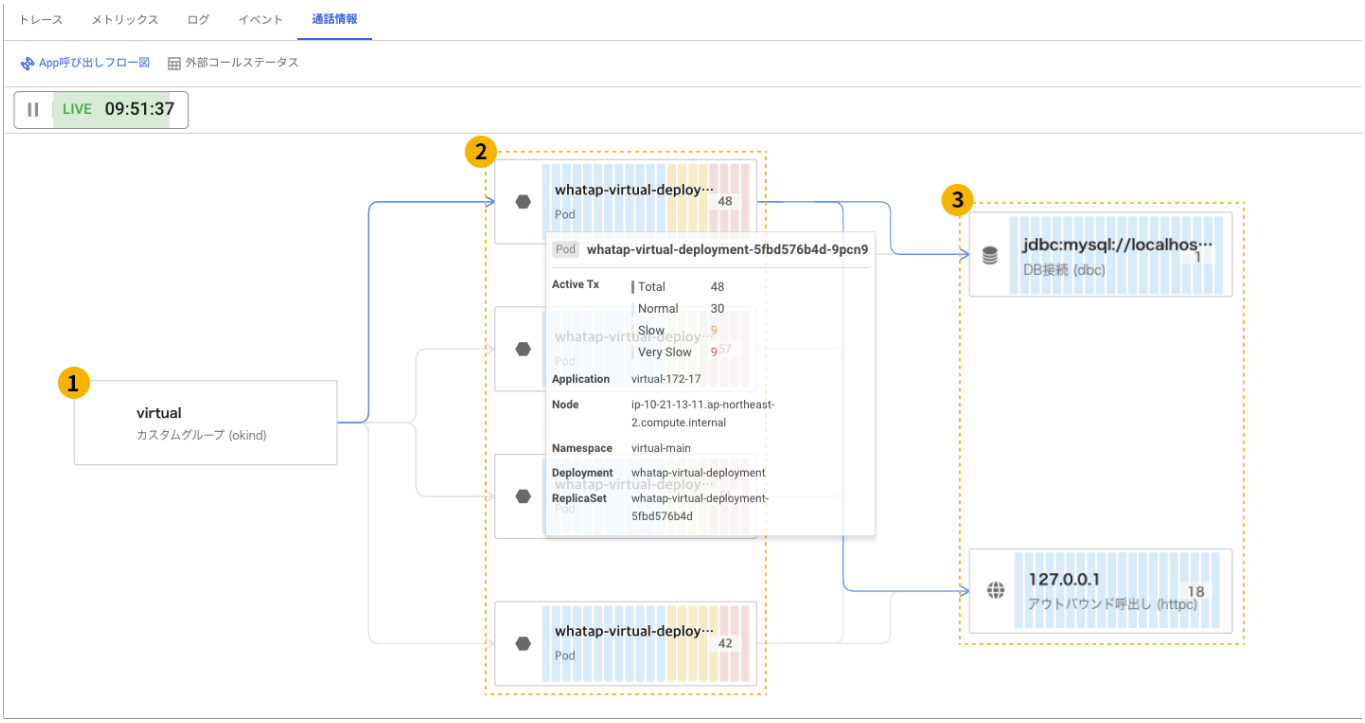
> ⓘ 外部呼び出し状態照会機能は、アプリケーションがJavaの場合のみ対応します。

ホーム画面 > プロジェクト選択 > ダッシュボード > コンテナマップ > 対象選択後**詳細分析**画面へアクセス > 呼び出し情報タブの選択

Kubernetesコンテナ上で実行中のアプリケーションの、呼び出しフローと外部呼び出し対象の状態を確認できます。WhaTap Kubernetesモニタリングで提供する呼び出し情報は以下の通りです。

- App呼び出しフロー図
- 外部呼び出し状態

App呼び出しフロー図



Pod内部で実行中の、コンテナ化したアプリケーションの呼び出しフローを表したチャートを提供します。 ユーザー指

定グループ(Okind)、Pod、DB接続(DBC)および外部呼び出し(HTTPC)の3つのノード間の接続関係を確認できます。 各ノードは、以下の情報を提供します。

1 ユーザー指定グループ(okind)

ユーザーが指定したグループ名を意味します。 グループ(okind)を指定していない場合、該当のノードは表示されません。

2 Pod

コンテナ化したアプリケーションが属しているPod名を意味します。 コンテナ化したアプリケーション内の、アクティブTXの総数と処理速度が確認できます。 選択時、アクティブTXメニューへ移動します。

- Active Tx: コンテナ化したアプリケーション内のアクティブTXを意味します。
  - Total: コンテナ化したアプリケーション内のアクティブTXの総数を意味します。
  - Normal: 0~3秒内に処理されるトランザクションの数を意味します。
  - Slow: 3~8秒内に処理されるトランザクションの数を意味します。
  - Very Slow: 処理に8秒以上かかるトランザクションの数を意味します。
- Application: Pod内部のApplication名を意味します。
- Node: Podが属するノード名を意味します。
- Namespace: Podが属するNamespaceの名を意味します。
- ReplicaSet: Podが属するReplicaSetの名を意味します。
- Deployment: Podが属するDeploymentの名を意味します。

3 DB接続と外部呼び出し

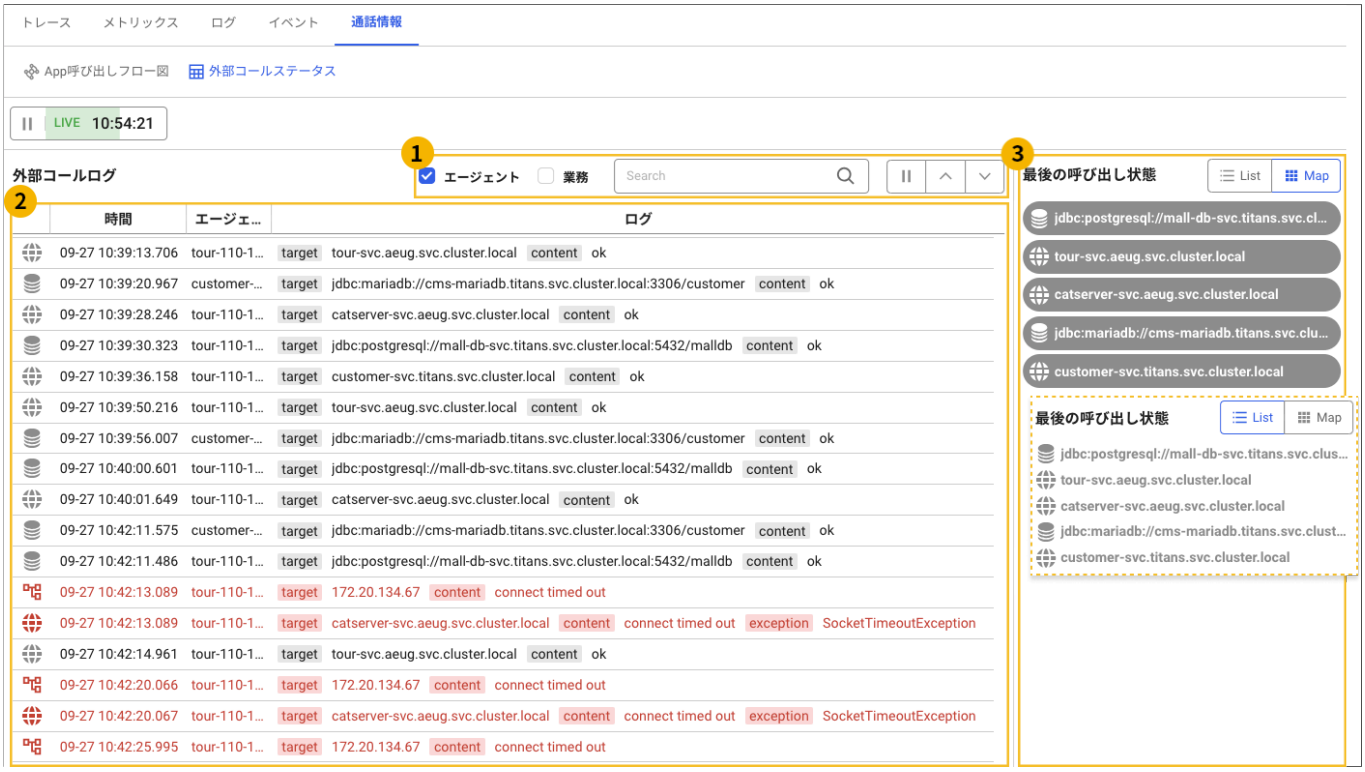
- DB接続(dbc): Pod内部の、コンテナ化したアプリケーションが呼び出したDB Connectionを意味します。 選択時、外部呼び出しの状態照会タブへ移動します。
  - Active Tx: DB Connection中の、コンテナ化したアプリケーション内のアクティブTXを意味します。
  - Total: コンテナ化したアプリケーション内のアクティブTXの総数を意味します。
  - Normal: 0~3秒内に処理されるトランザクションの数を意味します。
  - Slow: 3~8秒内に処理されるトランザクションの数を意味します。

    - Very Slow: 処理に8秒以上かかるトランザクションの数を意味します。
- 外部呼び出し(httpc): Pod内部の、コンテナ化したアプリケーションが呼び出した外部APIを意味します。 選択時、外部呼び出しの状態照会タブへ移動します。
    - Active Tx: 外部API呼び出し中の、コンテナ化したアプリケーション内のアクティブTXを意味します。
    - Total: コンテナ化したアプリケーション内のアクティブTXの総数を意味します。
    - Normal: 0~3秒内に処理されるトランザクションの数を意味します。
    - Slow: 3~8秒内に処理されるトランザクションの数を意味します。
    - Very Slow: 処理に8秒以上かかるトランザクションの数を意味します。

外部呼び出し状態

外部呼び出しログを利用して、モニタリング対象が呼び出す目的地の接続状態情報に基づいて、外部呼び出し対象が正常かどうかの情報を提供します。 外部呼び出し状態照会機能を通じて、接続不可対象向けの可視性を提供します。 これにより、障害発生時の問題個所が容易に把握できます。 また、持続的な接続失敗によるスレッドやコネクションの増加などの、リソース不足現象を事前に感知し、対応することができます。

> **外部呼び出しログ**
>
> アプリケーションエージェント設定により、各モニタリング対象別の外部呼び出しの成功と失敗を記録したログです。
>
> - 外部呼び出しタイプは次のとおりです。
>
>   `httpcall` 、 `db` 、 `socket`
>
> - 外部呼び出しログを収集するには、エージェント設定に次のオプションを追加する必要があります。
>
>   ```
>   logsink_rt_enabled=true
>   logsink_enabled=true
>   ```



外部呼び出し状態タブから、外部呼び出しログを一覧形式で照会することができます。 最近24時間範囲のログを最大500件読み込んで表示します。 リアルタイムで照会すると、データは3秒間隔で更新されます。

1 上段バーオプション

- エージェントまたは業務カラムの表示有無を選択できます。
- 検索語入力画面を通じて、該当の検索語が属するエージェント、業務、ログを含む列がフィルタリングできます。
- ▶ 再生: ログがアップデートされた時に最下段へ移動し、最近のログを確認できます。
- ⏸ 一時停止: 自動スクロール移動を一時的に停止します。
- ▲ ▼: 最上段または最下段へ移動します。

2 外部呼び出しログ一覧

アイコンとタグを通じて、外部呼び出しログの主な内容を簡単に確認できます。 エラーログの場合赤色として表示されます。

- 外部呼び出しタイプはアイコン(db)、(httpcall)、(socket)で表示します。

- target : 外部呼び出しと接続対象を意味します。
- content : ログの内容を意味します。
- exception : エラー発生時に記録されたログの内容を意味します。

3 最後の呼び出し状態

照会されたすべてのログの外部呼び出し対象一覧を確認します。 直近の接続状態によって成功または失敗として表示されます。

- 照会モード
  - List: 基本の一覧ビューです。 一覧全体の長さが画面の高さを超えたらスクロールが生成されます。
  - Map: 一覧が画面を超えたら一覧の長さを短くして、横方向に外部呼び出し対象を並べます。
- フィルター

  外部呼び出し対象を選択すると、該当の外部呼び出し対象のログのみをフィルタリングし、外部呼び出しログ一覧を照会することができます。 再選択すると解除されます。

> **データがない場合**
>
> リアルタイム照会時に、最近24時間の間に外部呼び出しログが収集されたかを確認します。 24時間の間、外部呼び出しログが収集されなかったら、次のような案内メッセージを表示します。


現在、このプロジェクトでは最近24時間以内に外部(リモート)呼び出し関連のログが見つかりません。
外部コールステータスは、アプリケーションエージェントの設定を通じて、
各モニタリング対象毎の外部（リモート）呼び出しの成功/失敗有無を記録したログを視覚化します。
この機能は、Javaアプリケーションの場合にのみ対応します。
この機能を使用するには、「管理 > エージェント設定」メニューから
以下のエージェント設定を追加してください。
logsink_rt_enabled=true
logsink_enabled=true
設定へ →

ノードマップ

> **要求事項**
>
> - WhaTap Kubernetesエージェント1.4.1バージョン以降が必要です。
> - Kubernetesクラスタープロジェクト(CP)の場合、該当メニューを照会できます。

ノードマップメニューは、クラスターの観点からノードの状態をモニタリングする機能を提供します。 ノードの状態、リソース使用量、エージェントの状況を可視化し、ユーザーが直感的に把握できるようにサポートします。 また、過去時点のノード状態を照会でき、以前の状態を簡単に確認できる機能を提供します。

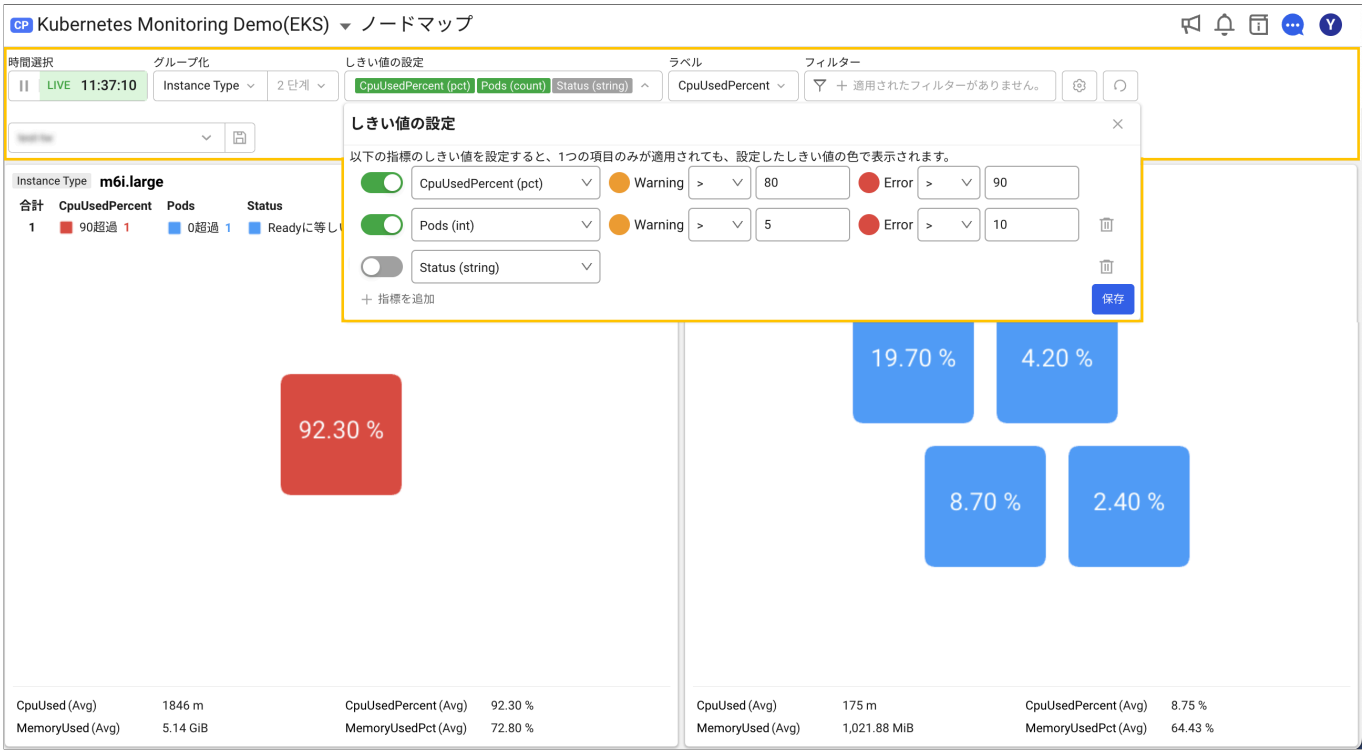


基本画面ガイド

ノードマップメニューは、大きく**ユーザー設定**領域と**ノードマップチャート**領域で構成されています。

ユーザー設定

ノードマップ上段の**ユーザー設定**領域では**グループ化**、**しきい値設定**、**ラベル**、**フィルター**、プリセットおよび⚙追加設定を構成できます。 次の例は、**グループ化**、**しきい値設定**、**ラベル**が適用されています。



グループ化

ユーザーは、様々な条件によってノードを**グループ化**して表示できます。 これにより、各グループ内のノードの関係を簡単に把握できます。 **グループ化**機能は、最大2段階までサポートされます。 **ノードマップ グループ化**を通じて、ユーザーは特定の基準でノードを選別し、カスタマイズされたモニタリング環境を構成できます。

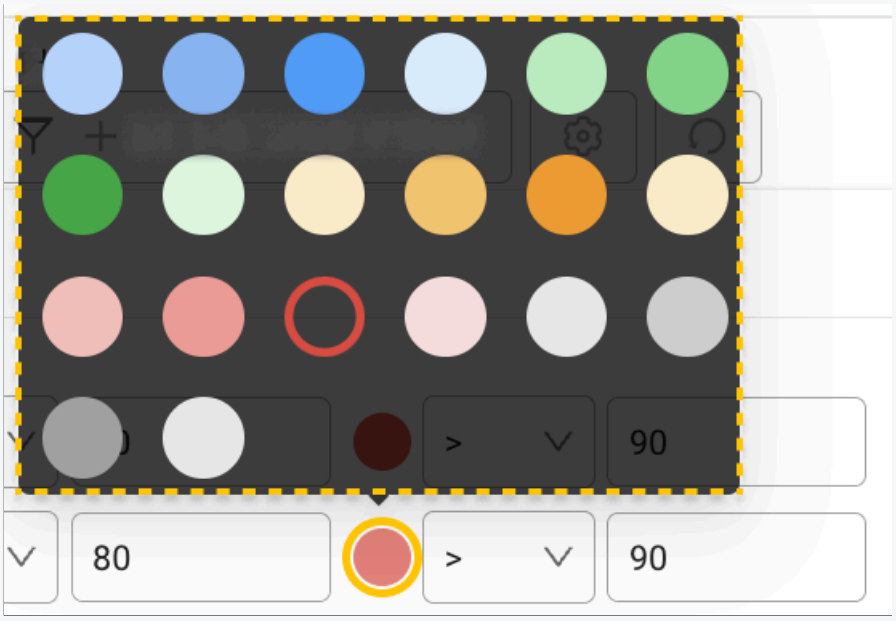
しきい値の設定

ユーザーは、**しきい値設定**を通じてノードの状態を色で表現できます。 メトリクスは1つ以上設定可能で、各メトリクスごとに警告(Warning)とエラー(Error)区間を指定できます。 正常(Normal) 区間は青色で表示します。 設定した範囲によって、その状態に合わせた色が適用されます。

1. **しきい値設定**オプションをクリックします。

2. メトリクスを選択してから、WarningとError区間を設定します。
3. メトリクスを追加する場合は、下段の**+ 指標追加**ボタンをクリックします。
4. メトリクスごとのしきい値を設定した後、下段の**保存**ボタンをクリックします。

異常状態区間であるWarning、Errorは基本的にオレンジ色、赤色に設定されています。 **しきい値設定**で、各状態の色をユーザーが指定できます。



文字列タイプのメトリクス

文字列タイプ(String)のメトリクスの場合、特定の文字列が含まれる時にErrorに指定されます。 例えば**Status (string)**メトリクスに `NotReady` というテキストが含まれる場合、そのノードはErrorに指定され赤色に表示されます。

メトリクス名	値	Error状態
Status	`Ready *` / `NotReady *`	`NotReady *`
Ready	`true` / `false`	`false`

メトリクス名	値	Error状態
PIDPressure	true / false	true
DiskPressure	true / false	true
MemoryPressure	true / false	true
Unschedulable	true / false	true

ノード状態表示の優先順位

状態表示の優先順位はError区間が一番高いです。 様々なメトリクスのうち、1つの項目でもError区間に属する場合、該当ノードは優先的に赤色に表示されます。 次の例の場合、ノードaとノードbすべてErrorに表示されます。



ラベル



ユーザーは、ノードブロックにテキストラベルを設定して、指定した主な情報を一目で確認できます。 **ラベル**を**CpuUsedPercent**メトリクス基準で設定した場合、左の例のようにノードブロックからその情報を確認できます。

フィルター

ユーザーはフィルターを使用して、特定の条件を満たすノードを選択して集中的にモニタリングできます。

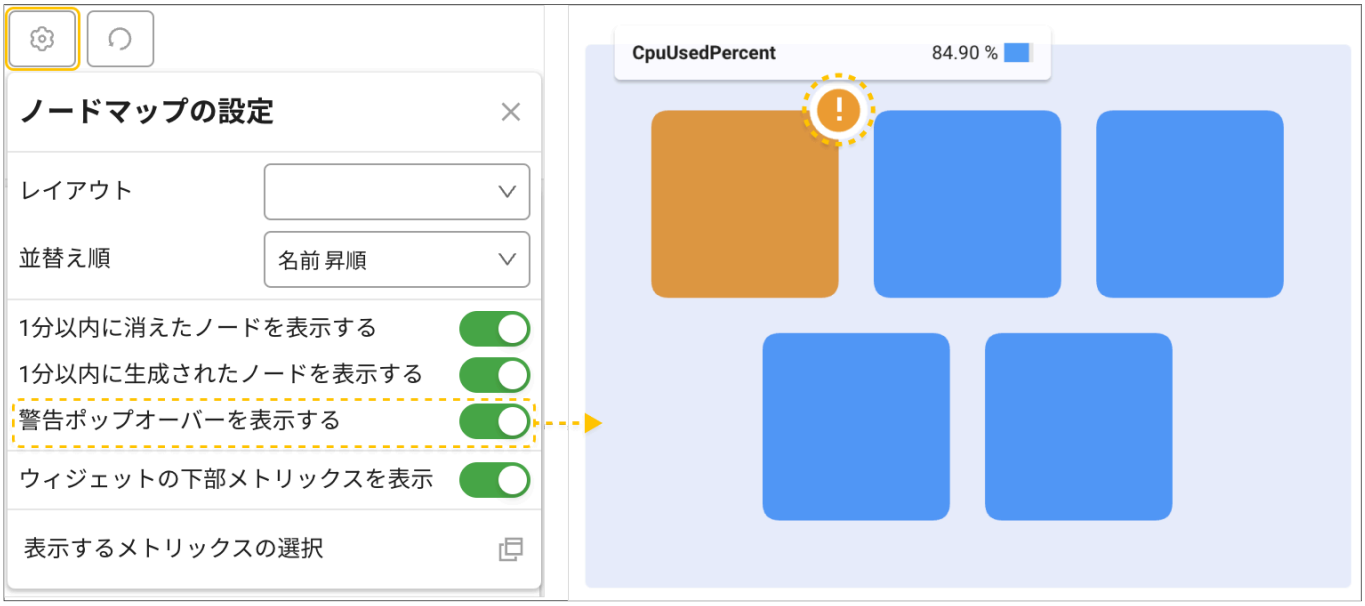
プリセット

ユーザーは、設定したオプションをプリセットに保存し、必要な時読み込みできます。 プリセットはプロジェクト単位で保存されます。 これにより、プロジェクトメンバーは同じ設定でクラスターをモニタリングできます。 基本プリセット(**Default**)を提供しており、基本プリセットは削除できません。

> **基本プリセット**
>
> **ノードマップ**の基本プリセットは、**CpuUsedPercent (pct)**メトリクスのしきい値が次のように設定されています。
>
> - Normal: 値が `80` 以下
> - Warning: 値が `80` 超過 `90` 以下
> - Error: 値が `90` 超過

追加設定

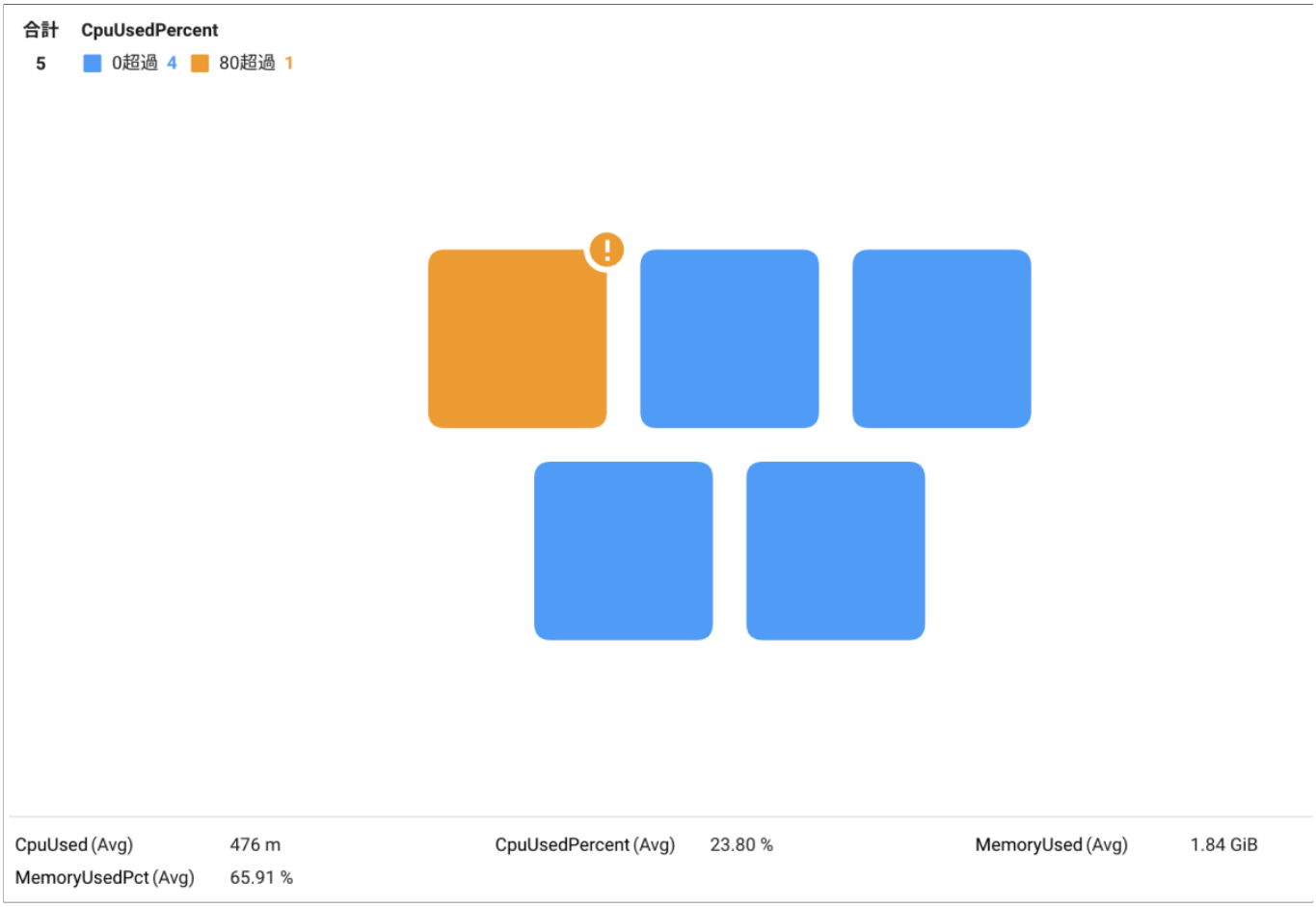
ノードマップ 右上の⚙アイコンを使用して、次のような追加の詳細設定が調整できます。



- **レイアウト**: グループ化されたウィジェットのカラムを固定します。
- **並替え順**: グループ化されたウィジェットをノード数または名前でソートします。
- **1分以内に消えたノードを表示する**: 最近1分以内に消えたノードを表示します。
- **1分以内に生成されたノードを表示する**: 最近1分以内に生成されたノードを表示します。
- **警告ポップオーバーを表示する**: 特定のノードが**しきい値設定**によってWarningまたはErrorステータスの時に警告のポップオーバーを表示します。
- **ウィジェットの下部メトリクスを表示**: ウィジェットの下部に平均、合計、最大値、最小値の表示有無を設定します。
- **表示するメトリクスの選択**: ウィジェットの下部に表示するメトリクスを選択します。 特定のメトリクスを選択し、グループ化されたすべてのノードに対して、データの表示を平均、合計、最大値、最小値から選んで設定します。

ノードマップチャート

ノードマップチャートは、次の3つの領域で構成されています。



1 ウィジェット上段

ノード数またはグループ化された場合、グループ化された項目の数と選択されたメトリクスの状態別の数を表示します。

2 中央可視化領域

ノード全体の状態と数を直感的に確認できる可視化されたダッシュボードを提供します。

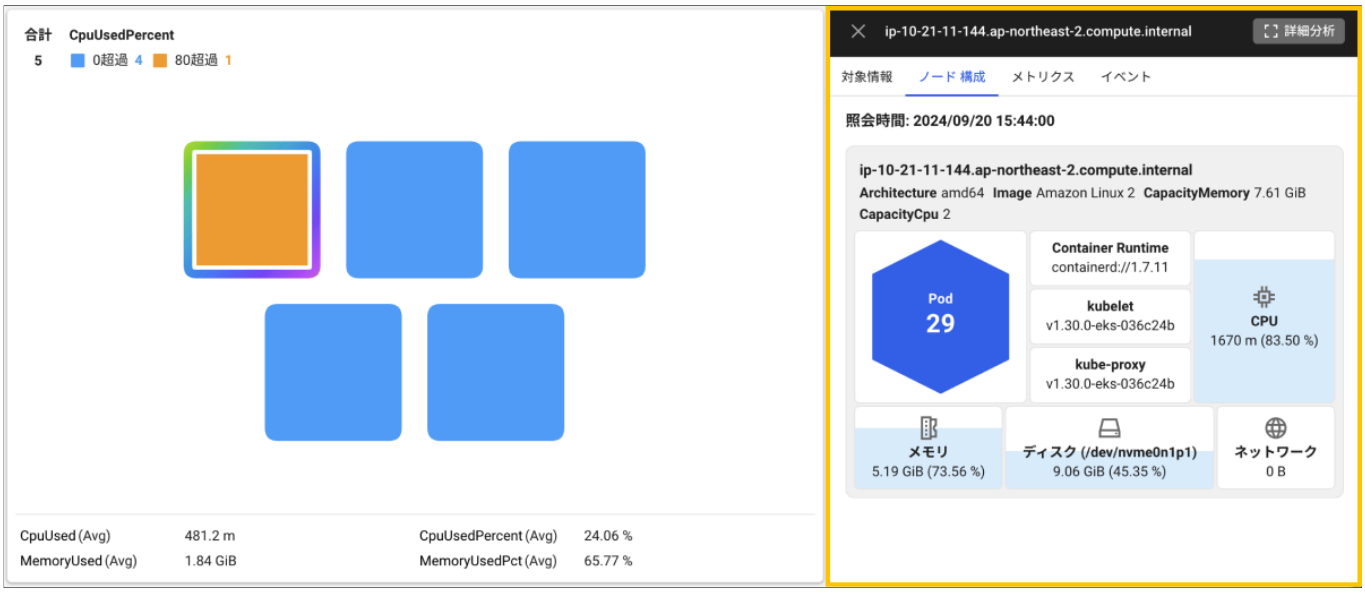
3 ウィジェット下段

選択したメトリクスの統計値を表示します。 追加設定の**表示するメトリクスの選択**で、設定したとおりに平均、合計、最大値、最小値のいずれかでデータを表示します。

> **ノードマップ**では、ノードエージェントがインストールされているノードのみを表示します。

要約の表示と詳細分析

ノードマップで照会するノードを選択します。 右側に表示される概要ビュー画面から該当オブジェクトの概要を確認できます。 要約の表示画面で**詳細分析**ボタンをクリックして詳細分析画面を確認できます。



対象情報

対象情報タブで、ユーザーが選択したノードの主な属性情報を確認できます。 また、ノードリスト およびコンテナマップ メニューに移動できます。

> ⓘ **Labels**指標は、クバネティスエージェントバージョン1.7.9から対応します。

ノード要約分析

ノードのリソース割り当てと使用状況の要約情報を**ノード要約分析**タブで確認できます。

- 基本情報: ノードのクラスターバージョン、起動時間、現在の状態、スケジューリングの可能有無、内部IPアドレス、そしてリージョンとゾーン情報を提供します。
- オペレーティングシステムおよびハードウェア情報: ノードが実行されるオペレーティングシステム、OSイメー

ジ、CPU アーキテクチャ、コア数、メモリの合計サイズを提供します。 ノードのメモリとCPUへの割り当ての可能有無と現在の使用量を確認できます。

- リソース使用状況: ノードで実行中のすべてのコンテナのリクエスト(Request)と割り当て可能リソース(CPU、メモリ)の違いを通じて、残りのリソースを把握します。 実際のCPU使用率とメモリ使用率を含め、リソースの使用状況を要約します。
- ディスク使用状況: ノードに接続されたディスク情報、ファイルシステムのタイプ、デバイスID、マウントポイント、合計容量と使用中の容量、そして使用率を提供します。
- Podの状態とリソースの余裕: ノードの割り当て可能な全体のPod数と現在実行中のPod数を比較して、追加実行可能なPod数を確認できます。 また、MemoryPressure、DiskPressure、PidPressureなどの状態でノードの安定性をチェックします。

ノード 構成

ノード 構成タブで選択したノード構成と全体的なリソース使用量の情報を要約して提供します。 そのノードのハードウェアとソフトウェアの構成要素を総合的に確認できます。 ユーザーはノードのリソース使用状況と各構成要素の状態を把握し、パフォーマンスと運用安定性をモニタリングできます。

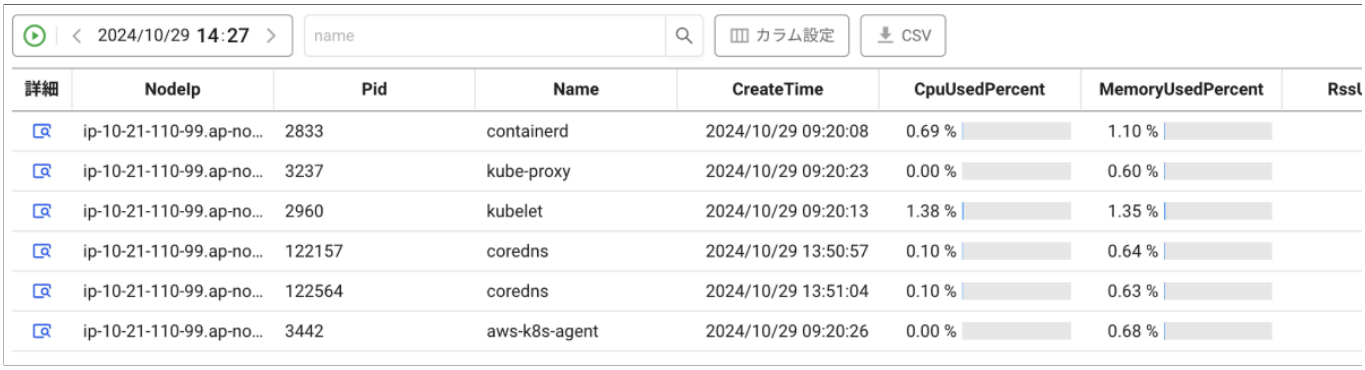

照会時間: 2024/09/20 14:40:30 ~ 2024/09/20 18:11:25
この統計データベースの要約分析は、詳細分析やシステム改善のための参考資料として活用できます。
ip-10-21-140-100.ap-northeast-1.compute.internal は、 CPU 2個、メモリ3.77 GiBのamd64 linuxノードです。 このノードはほとんどの時間にわたって余裕を持っていますが、断続的にCPU, メモリで中程度の負荷が発生する状態です。安定した運用が必要な場合は、
定期的に負荷が発生した時点での監視をお勧めします。
ネットワーク使用量が最大1.31 MiBで、平均使用量54.44 kiBよりも約24.69倍高く、
トラフィックパターンのボラティリティを確認する必要があります。
AVG
MAX
Architecture amd64 Image Amazon Linux 2 CapacityMemory 3.77 GiB CapacityCpu 2
Pod
12.51
Container Runtime
containerd://1.7.11
kubelet
v1.30.2-eks-1552ad0
kube-proxy
v1.30.2-eks-1552ad0
CPU
1380 m (69.00 %)
メモリ
1.04 GiB (30.50 %)
ディスク (/dev/nvme0n1p1)
4.51 GiB (5.64 %)
ネットワーク
1.31 MiB

- **ノードシステム仕様**
  - **Architecture**：ノードが使用しているハードウェアアーキテクチャ
  - **Image**：ノードが実行中のオペレーティングシステムのイメージ情報
  - **CapacityMemory**：ノードの総メモリ容量
  - **CapacityCpu**：ノードの総CPU容量
- **ソフトウェア構成**
  - **Container Runtime**：ノードで使用されるコンテナランタイムのバージョン情報
  - **kubelet**：ノードで実行中のクバネティスkubeletのバージョン情報
  - **kube-proxy**：ネットワーク管理の役割を果たすkube-proxyのバージョン情報
- **リソースの使用状況**
  - **CPU**：ノードのCPU使用量
  - **メモリ**：ノードのメモリ使用量
  - **ディスク**：ノードで使用中のディスク容量と使用比率
  - **ネットワーク**：ネットワークの伝送量

> ⓘ
> - CPU、メモリ、ディスク、ネットワークをクリックすると、照会期間中の使用量の推移が確認できる**メトリクス**タブに移動します。
> - **AVG**、**MAX**ボタンをクリックして、照会期間中の平均使用量と最大使用量を確認できます。

プロセス

プロセスタブで選択したノード内部にある主要なKubernetes管理プロセス情報を確認できます。 プロセスの問題によって引き起こされるクラスター障害を予防し、ダウンタイムを最小限にできます。



詳細	NodeIp	Pid	Name	CreateTime	CpuUsedPercent	MemoryUsedPercent	Rssl
	ip-10-21-110-99.ap-no...	2833	containerd	2024/10/29 09:20:08	0.69 %	1.10 %	
	ip-10-21-110-99.ap-no...	3237	kube-proxy	2024/10/29 09:20:23	0.00 %	0.60 %	
	ip-10-21-110-99.ap-no...	2960	kubelet	2024/10/29 09:20:13	1.38 %	1.35 %	
	ip-10-21-110-99.ap-no...	122157	coredns	2024/10/29 13:50:57	0.10 %	0.64 %	
	ip-10-21-110-99.ap-no...	122564	coredns	2024/10/29 13:51:04	0.10 %	0.63 %	
	ip-10-21-110-99.ap-no...	3442	aws-k8s-agent	2024/10/29 09:20:26	0.00 %	0.68 %	

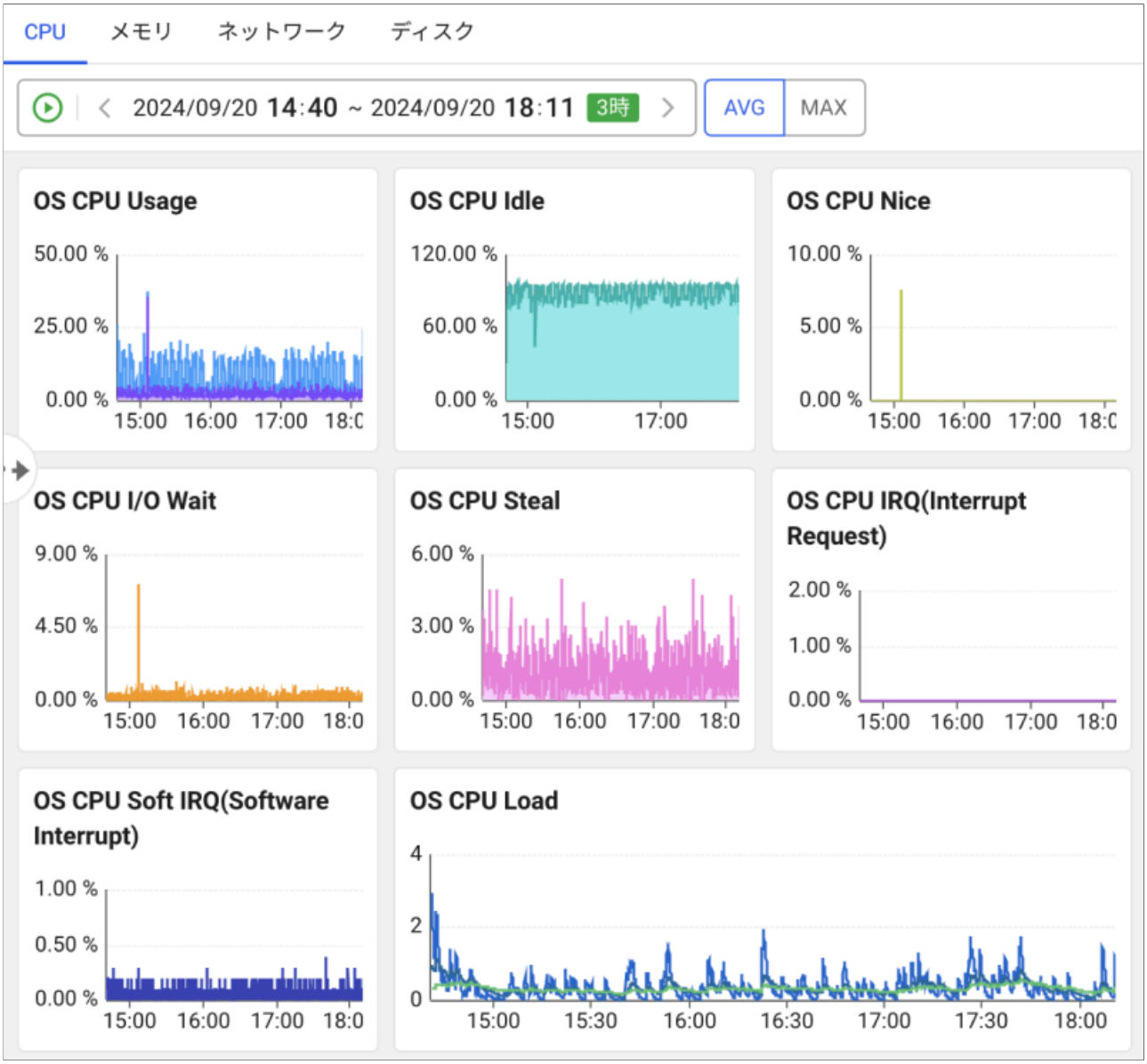
プロセス詳細

- プロセス一覧の左側**詳細**カラムでアイコンを選択すると、そのプロセスに関する詳細情報を確認できます。
- **AVG**または**MAX**ボタンをクリックして、3日以上の長期データ照会時に読み込む統計データの合算方式を選択できます。
- プロセスの主要な情報値をクリックすると、その値がコピーされます。

> **プロセス**タブは、Kubernetesエージェントバージョン1.7.14からサポートします。

メトリクス

メトリクスタブで選択したノードのリソース使用量の推移を確認できます。 照会期間中の平均(**AVG**)、最大値(**MAX**)を基準に**CPU**、**メモリ**、**ネットワーク**、**ディスク**使用量の変化を照会できます。



- **AVG**：照会期間中の平均使用量
- **MAX**：照会期間中の最大使用量
- **Total**：複数のネットワークインターフェイスの全体トラフィック使用量の合計
- **In/Out**：特定のネットワークインタフェースのInbound(受信) およびOutbound(送信) トラフィックの使用量

ⓘ **K8s CPU Request / Allocatable CPU (%)**ウィジェットは、Kubernetesエージェントバージョン1.7.10か

> らサポートします。

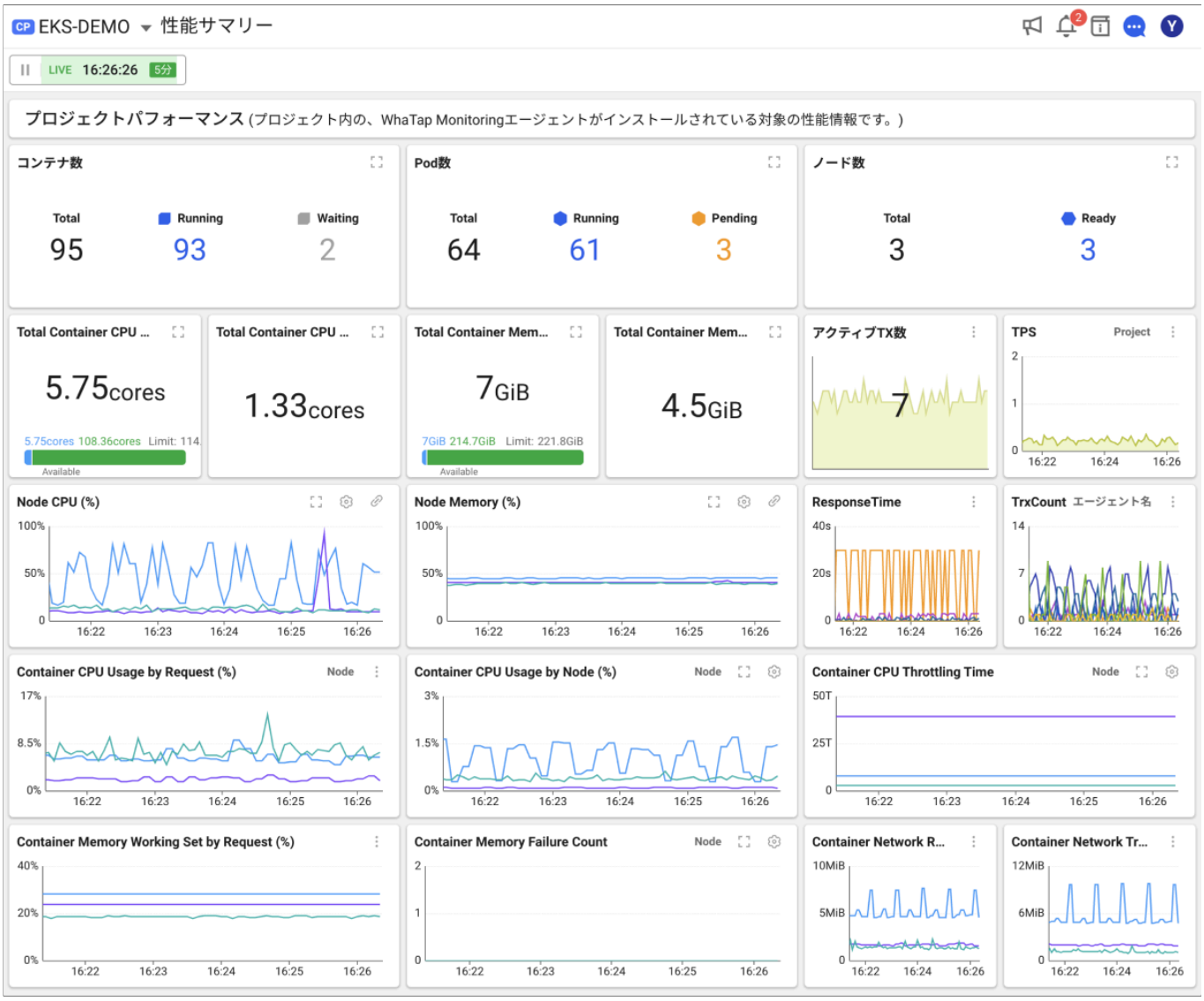
イベント

イベントタブで選択したノードで発生した**K8s Nodeイベント**と**WhaTap Nodeイベント**を確認できます。 **K8s Nodeイベント**はノードの状態変化を含めて、クラスターの重要な作業記録を提供し、**WhaTap Nodeイベント**はユーザーが設定したアラート通知の基準に基づくイベント発生情報を提供します。


K8s Nodeイベント
WhaTap Nodeイベント
2024/09/23 10:04 ~ 2024/09/24 10:04 1日
イベント数の推移
11
5.5
0
09/24
11:00
13:00
15:00
17:00
19:00
21:00
23:00
01:00
03:00
05:00
07:00
09:00
Normal
Warning
タグを隠す
ログメニューへ移動
タイムスタンプ
ログ
2024-09-23 10:34:30.136 NodeReady Node ip-10-21-110-253.ap-northeast-1.compute.internal status is
kind Node name ip-10-21-110-253.ap-northeast-1.compute.internal
2024-09-23 10:34:27.295 Starting
kind Node name ip-10-21-110-253.ap-northeast-1.compute.internal
2024-09-23 10:34:23.842 RegisteredNode Node ip-10-21-110-253.ap-northeast-1.compute.internal ever
kind Node name ip-10-21-110-253.ap-northeast-1.compute.internal
2024-09-23 10:34:21.729 Synced Node synced successfully
kind Node name ip-10-21-110-253.ap-northeast-1.compute.internal
2024-09-23 10:34:21.575 NodeAllocatableEnforced Updated Node Allocatable limit across pods
kind Node name ip-10-21-110-253.ap-northeast-1.compute.internal
2024-09-23 10:34:21.531 NodeHasSufficientPID Node ip-10-21-110-253.ap-northeast-1.compute.intern
kind Node name ip-10-21-110-253.ap-northeast-1.compute.internal

性能サマリー

最近の時間別チャートを通じて、プロジェクト内のWhaTapノードエージェントおよびアプリケーションエージェントがインストールされている対象の情報を表示します。ネームスペースプロジェクトに分離された情報は、ネームスペースプロジェクトで確認できます。



- **ショートカットオプション**

ウィジェッ別の右上に位置する⋮詳細アイコンを選択して🔗ショートカットオプションを確認できます。

- **ラインチャート設定オプション**

  ラインチャートウィジェット別の右上に位置する⋮詳細アイコンを選択した後、またはすぐに⚙設定オプションを確認できます。ラインチャートの場合、設定オプションを通じてidとなる値を変更できます。この設定は保存されません。

  例、Node CPU (%)ウィジェットの場合Nodeごとのデータ照会が可能です。メトリクスを変更してContainerNameごとに照会できます。

> ⓘ エージェントのインストールが正常に完了していない場合、データが表示されない可能性があります。

ノード情報

ウィジェットを提供するNode情報は次のとおりです。

- 直前30分チャート
  - Node CPU (%)、Node Memory (%)など

> ⓘ
> - ノード数量ウィジェットの右上に⋮その他のアイコン>🔗ショートカットアイコンを順番に選択するとクラスター>ノード一覧メニューに移動します。
> - Node CPU (%)ウィジェットとNode Memory (%)ウィジェット右上の⋮詳細を見るアイコン>🔗ショートカットアイコンを順番に選択すると分析>メトリクスチャートメニューに移動します。該当メニューを使用すると、より詳細な条件でチャートを確認できます。

コンテナ情報

ウィジェットによって提供するコンテナ情報は次のとおりです。

- 直前10分チャート
  - Container Status、Container CPU (%)、Container Memory (%)など

アプリケーション情報

ウィジェットによって提供するアプリケーション情報は次のとおりです。

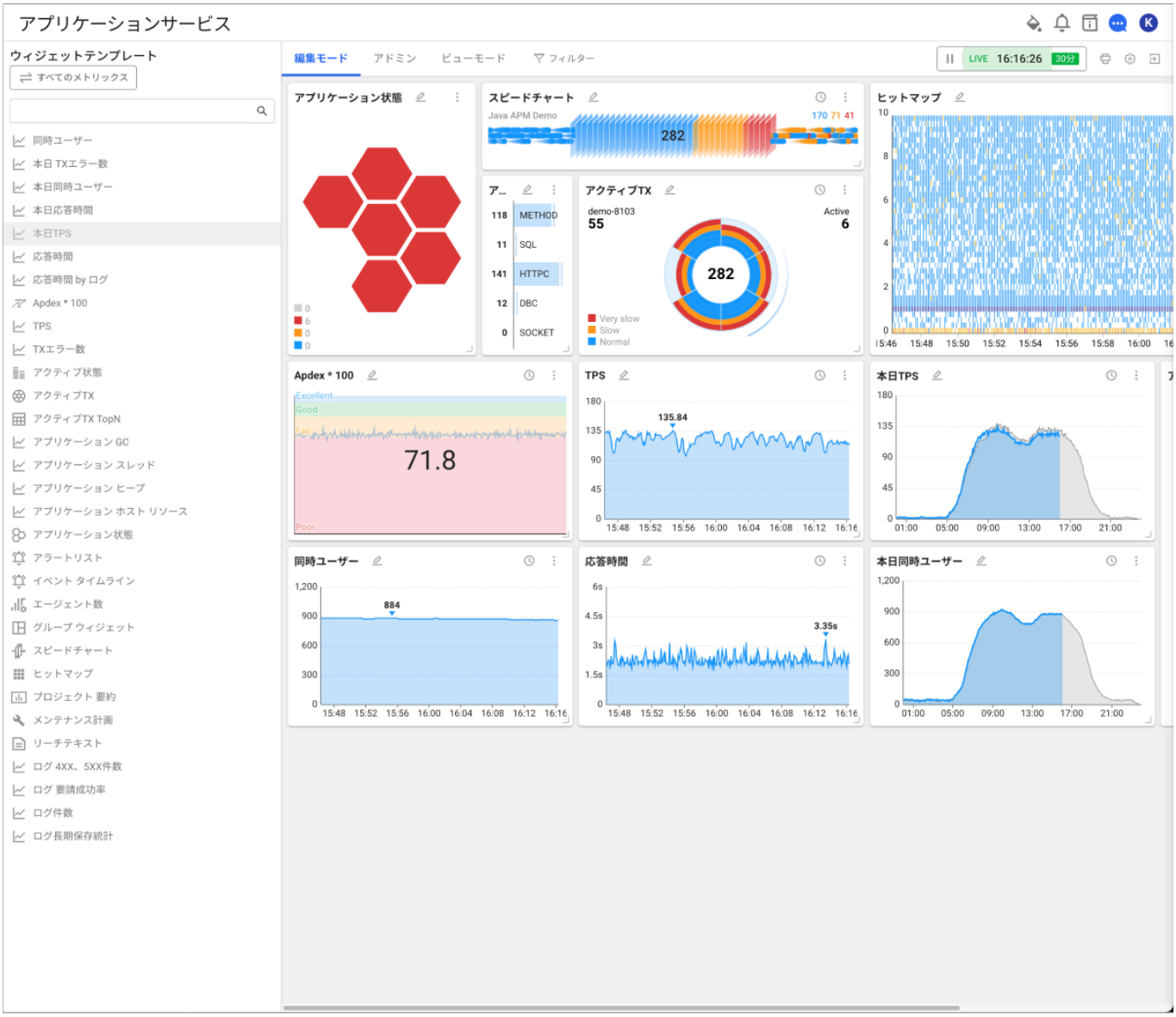
- 直前10分チャート
  - Active TX、TPS、Response Timeなど

> ⓘ アプリケーション情報は、アプリケーションエージェントのインストール対象が存在する場合に表示します。

Flexボード

Flexボードはユーザー定義型の統合ダッシュボードです。アプリケーション、サーバー、データベース、コンテナなどのプロジェクトのデータを使用して、リアルタイムのダッシュボードが作成できます。

事前構成されたテンプレートで初期設定を迅速に完了し、希望する形式のダッシュボードを作成できます。データウィジェットを追加したりプロパティを修正したりして、必要な形式で表示することができ、フィルタリング機能によって監視対象を絞り込むこともできます。時間範囲を指定すれば特定の時点のデータを確認でき、補助チャートを活用することで多様な方法で分析できます。ダッシュボードはお気に入りに登録して簡単にアクセスでき、カスタマイズしたダッシュボードは他のアカウントにコピーして再利用することも可能です。



Flexボードは、次のアクセスパスによってさまざまな用途があります。

- ホーム画面 > **統合Flexボード**
  - ウィジェットを作成すると、表示可能なすべてのプロジェクトを選択オプションとして提供します。
  - ユーザーアカウントにダッシュボードが保存され、他のユーザーにコピーする機能を使用して共有できます。
  - 個人アカウントダッシュボードで、権限による影響はありませんが、読み取り専用として共有されたダッシュボードの場合は修正できません。

- ホーム画面 > プロジェクト選択 > **ダッシュボード** > **Flexボード**
  - ウィジェットを作成すると対応するプロジェクトの情報が自動入力されます。
  - プロジェクトメンバーで作成したFlexボードは、自動共有されます。
  - プロジェクト**修正**権限、または**プロジェクトのFlexボードの修正**権限、**Site Admin**権限を持つユーザーが修正できます。

Flexボードの編集権限を持つユーザーは、以下の機能を利用できます。

- ダッシュボードをJSONファイルとしてエクスポート/インポート
- ダッシュボード内のデータリクエストおよびレスポンス内容を確認
- ウィジェット設定オプションをJSON形式で参照および修正

プロジェクト内の**Flexボード**メニューでは、ダッシュボードの変更権限のあるユーザーのみが**編集モード**および**アドミン**モード、**フィルター**機能にアクセスできます。

- アクセス可能なメンバー権限
  - プロジェクト**修正**権限
  - **プロジェクトのFlexボードの修正**権限
  - **Site Admin**権限
- **ビューモード**および**フィルター**機能の詳細については、次の文書を参照してください。
- メンバー権限の詳細については、次の文書を参照してください。

Flexボードの使用方法

> ダッシュボードはメトリクスウィジェットを含むさまざまなウィジェットテンプレートを活用して構成できます。
> 各テンプレートの特徴と使用方法については、ウィジェットテンプレート ドキュメントで詳しく確認できます。

事前テンプレートを適用する

事前構成テンプレートで初期設定を素早く進め、必要に応じて修正して活用してください。

1. **Flexボード** メニューを選択します。
2. 左側の **テンプレート** 一覧から使用するテンプレートを選択します。
3. **Flexボード管理** ウィンドウが表示されたら、**名前変更** に希望の名前を入力します。
    - **レイアウト** でウィジェットを選択し、ドラッグして希望の位置へ移動できます。
4. **保存** ボタンをクリックします。選択したテンプレートを基準に Flex ボードを作成します。
    - 作成したボードは **Flexボード** のボード一覧で確認できます。

Flex ボードを作成する

Flex ボードを作成し、ウィジェットを配置して自分だけのダッシュボードを作れます。

1. **Flexボード** で右上の **ダッシュボード生成** ボタンをクリックします。
2. **ダッシュボード生成** ウィンドウが表示されたら、入力ボックスにダッシュボードの **名前** を入力します。
3. ウィジェットの配置方式を選択し、**ダッシュボード生成** ボタンをクリックします。
    - **固定レイアウトFlexボード**: ピクセル基準の自由配置
    - **レスポンシブFlexボード** : ブラウザサイズ基準のグリッドレスポンシブレイアウト

（グリッドを設定してから **ダッシュボード生成** ボタンをクリック）

4. **ウィジェットテンプレート** 選択画面で、ダッシュボードに配置するウィジェットを選択します。
   - メトリクスウィジェットを追加する場合は、**ウィジェットテンプレート** の右側にある **すべてのメトリクス** ボタンをクリックします。

   > ⓘ
   > - **ウィジェットテンプレート**: 一般的な監視状況で重要となる指標を要約し、事前定義されたウィジェット一覧から選択できます。
   > - **すべてのメトリクス**: ユーザーのプロジェクトで収集中のすべてのメトリクスデータを基準にウィジェットを作成します。メトリクスウィジェットの詳細は次のドキュメントを参照してください。

5. 配置したウィジェットの位置を移動したりサイズを調整します。
   - ウィジェット上部で十字カーソルが表示されたら、クリックしたままドラッグして位置を移動します。
   - ウィジェット右下で矢印カーソルが表示されたら、クリックしたままドラッグしてサイズを調整します。
6. 追加するウィジェットの配置が完了したら、**ビューモード** を選択してレイアウトを確認します。レイアウトを再編集する場合は、**編集モード** を選択してウィジェット配置を変更します。
7. すべての工程が完了したら、右上の（**リストへ**）アイコンをクリックします。
   - **Flexボード** の **ダッシュボードリスト** で作成したボードを確認できます。作成したボードを選択して新しい Flex ボードを確認してください。

> ⓘ お客様のご要望を反映してダッシュボードテンプレートやウィジェットの追加を検討しています。必要なテンプレートまたはウィジェットがあれば support@whatap.io までご連絡ください。

Flex ボードを編集する

> ⓘ **Flex ボード編集権限**

- **ホーム** > **統合Flexボード**: 個人アカウントのダッシュボードで権限の影響はありませんが、閲覧専用で共有されたダッシュボードは編集できません。
- **ホーム** > **プロジェクト選択** > **ダッシュボード** > **Flexボード**: プロジェクトの編集権限、プロジェクトのフレックスボード編集権限、または Site Admin 権限のユーザーが編集できます。

メンバー権限の詳細は次のドキュメントを参照してください。

1. **Flexボード** の **ダッシュボードリスト** で、編集アイコンをクリックします。画面右側に **Flex ボード管理** ウィンドウが表示されます。
2. **Flex ボード管理** ウィンドウで、名前、プロジェクト、レイアウト、widget.json、option.json を設定します。
   - **名前変更**: ダッシュボード名を編集します。
   - **プロジェクト**: ダッシュボードのウィジェットがデータを取得するプロジェクトを選択します。

     プロジェクト オプションは **ホーム** > **統合Flexボード** メニューでのみ設定できます。
     統合Flexボード から入った場合は、ウィジェットごとにデータ取得元のプロジェクトを選択できます。

     1. ウィジェット右上の ＋ ボタンをクリックし、目的のプロジェクトを検索して単一または複数選択します。
     2. プロジェクトを選択したら **適用** ボタンをクリックします。

   - **レイアウト**
     - ウィジェットをドラッグして位置を移動します。
     - ウィジェット右下をマウスでクリックしたままドラッグしてサイズを調整します。
     - ウィジェットを削除するには、ウィジェット右上の アイコンをクリックします。
   - **ウィジェット.json**: ダッシュボードのウィジェット設定を JSON 形式で読み込みます。 アイコンをクリックすると JSON をコピーします。
   - **オプション.json**: ダッシュボードのオプション設定を JSON 形式で読み込みます。 アイコンをクリックすると JSON をコピーします。

Flex ボードを削除する

1. **Flexボード** > **ダッシュボードリスト** で、削除する項目の アイコンをクリックします。
2. 確認メッセージで **削除** ボタンをクリックします。

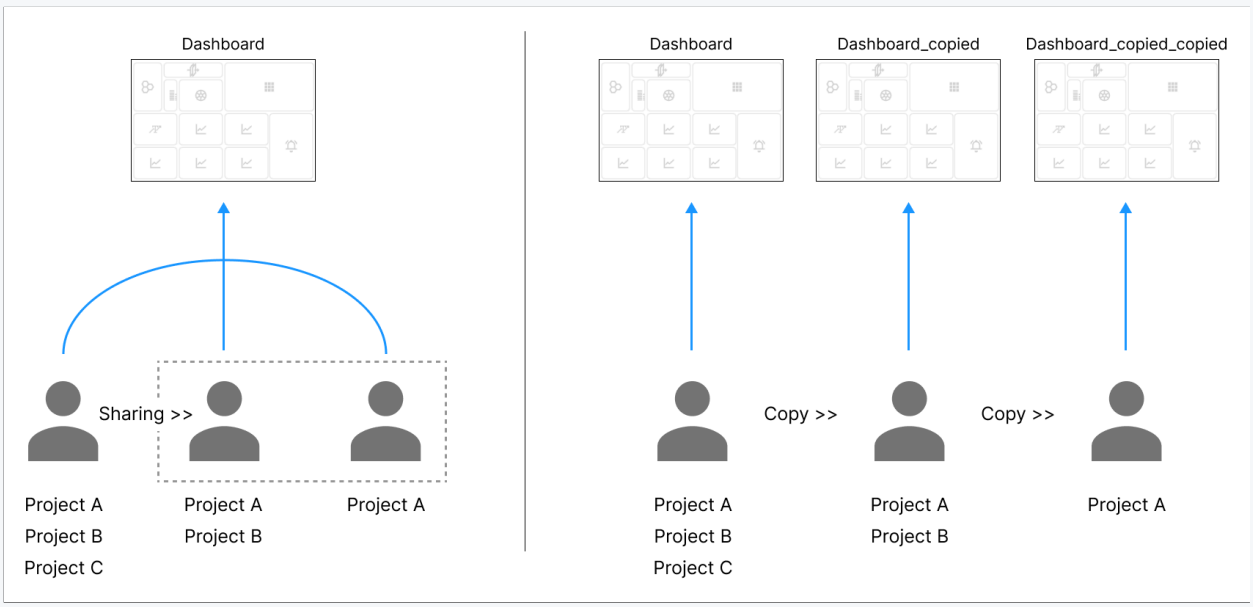
ダッシュボードの削除は所有者と共有を受けたユーザーで動作が異なります。

- ダッシュボード所有者がダッシュボードを削除すると、共有されたすべてのユーザーアカウントでもそのダッシュボードが削除されます。
- 共有を受けたユーザーがダッシュボードを削除すると、そのユーザーの **ダッシュボードリスト** のみから削除されます。元のダッシュボードは維持されます。

Flex ボードを共有する

Flex ボードの共有機能は **ホーム** > **統合Flexボード** メニューでのみ使用できます。
プロジェクトの **Flexボード** メニューでは提供されません。

共有とコピーの違い



- 共有: 共有されたメンバーは同じダッシュボードを閲覧します。
- コピー: ダッシュボードを他のメンバーまたは自分にコピーすると、コピーされた別個のダッシュボードを閲覧します。
- **読取り専用** で共有/コピーされたダッシュボードは編集できません。**編集モード** で共有/コピーされた場合のみ編集できます。
- **読取り専用** で共有を受けたメンバーも、そのダッシュボードを **閲覧専用で再共有** したり **コピー** できます。
- ダッシュボードに含まれるプロジェクトのうち **少なくとも1つ** のプロジェクトに所属しているメンバーにのみ共有できます。共有またはコピー先のメンバーがプロジェクトに所属しているか確認してください。



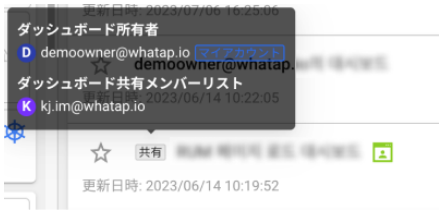
統合Flexボード で作成した Flex ボードを他のメンバーと共有できます。所有者が共有されたダッシュボードを編集すると、共有を受けたすべてのメンバーに同じ変更が反映されます。

1. **統合Flexボード** の **ダッシュボードリスト** で、共有するダッシュボードの アイコンをクリックします。
2. **ダッシュボード共有** ウィンドウが表示されたら、共有するメンバーを1名または複数名選択するか、アカウントを直接入力します。
    - **読取り専用** を選択すると、共有を受けたメンバーはダッシュボードを編集できません。

- 共有対象は、ダッシュボードに含まれるプロジェクトのうち少なくとも1つのプロジェクトに所属している必要があります。

3. **共有** ボタンをクリックします。

- 共有された項目には **共有** タグが表示されます。**共有** タグにマウスを載せると、共有したメンバー情報を確認できます。
- 閲覧専用で共有された項目には **読取り専用** タグが表示されます。

ダッシュボードを共有で受け取った場合	ダッシュボードを共有した場合（ダッシュボード
	

ダッシュボード共有の条件

- ダッシュボード共有機能は **ホーム** > **統合Flexボード** のみで提供します。
- だれでも自分のダッシュボードを共有でき、共有で受けたダッシュボードを再共有やコピーできます。
- **読取り専用** で受けたダッシュボードは編集できませんが、**編集モード** で受けた場合は編集できます。
- **所有者** がダッシュボードを削除すると、共有を受けたすべてのアカウントで該当ダッシュボードが削除されます。
- **共有を受けたユーザー** が削除した場合は、そのユーザーの **ダッシュボードリスト** のみから削除され、元のダッシュボードは維持されます。
- 共有で受けたダッシュボードに含まれるプロジェクトのうち、**ビュー** 権限がないプロジェクトはデータを参照できません。

Flex ボードをコピーする

ユーザーは Flex ボードをコピーして他のメンバーに渡せます。

1. **統合Flexボード** の **ダッシュボードリスト** で、 アイコンをクリックします。
2. コピーオプションを選択します。
   - **自分にコピー**: 自分のアカウントにコピーします。

     a. `_copied` 接尾辞が付いた項目が **ダッシュボードリスト** に追加されます。

   - **他人にコピー**: 他のメンバーのアカウントにコピーします。

     a. ＋ **アカウント** をクリックします。
     b. コピー先メンバーのメールを選択し、**ダッシュボードコピー** ボタンをクリックします。
     c. そのメンバーの **ダッシュボードリスト** に `_copied` 接尾辞が付いた項目が追加されます。

JSON ファイルで共有する

Flex ボードの設定を JSON ファイルとして他のメンバーに渡したり、他のメンバーの設定を取り込めます。

エクスポート

1. **統合Flexボード** またはプロジェクトの **Flexボード** メニューに移動します。
2. **ダッシュボードリスト** で ボタンをクリックします。
3. エクスポートオプションを選択します。
   - **プロジェクト情報を含む**: ダッシュボードに設定されたプロジェクト情報を JSON ファイルに含めて共有
   - **プロジェクト情報を含まない**: ダッシュボードに設定されたプロジェクト情報を JSON ファイルに含めずに共有
4. JSON ファイルをダウンロードし、共有先のメンバーに渡します。

インポート

1. **統合Flexボード** またはプロジェクトの **Flexボード** メニューに移動します。
2. 画面右上の **インポート** ボタンをクリックします。
3. ダウンロードした JSON ファイルを選択します。

> この機能は、プロジェクトの **編集** 権限を持つメンバーのみ利用できます。

Flex ボードの画面モード

Flex ボードの画面モードに応じて提供機能が異なります。

> **アクセス可能なメンバー権限**
>
> プロジェクト内の **Flexボード** メニューでは、**編集権限** のあるユーザーのみが **編集モード**、**アドミン**、**フィルター** 機能にアクセスできます。メンバー権限の詳細は次のドキュメントを参照してください。
>
> - **プロジェクト編集** 権限
> - **プロジェクトのFlexボードの修正** 権限
> - **Site Admin** 権限

モード種類

Flexボード は画面モードにより設定できる機能が異なります。

- **編集モード**: ウィジェットテンプレート一覧からレイアウトにウィジェットを配置し、ウィジェットの編集、削除、位置調整ができます。
- **アドミン**: ウィジェットの詳細プロパティを JSON 形式で管理できます。レイアウトに配置したウィジェット右上の 🔧 ボタンを選択します。
- **ビューモード**: ウィジェットの配置、サイズ調整、詳細オプションなどの設定を完了後、ダッシュボードのように各ウィジェットのデータをリアルタイムで監視できます。

共通機能

- データフィルタリング
- リアルタイムデータの参照範囲設定
- 印刷モード
- Flex ボード管理
- データ結合オプション

データフィルタリング

データフィルタリングで監視対象を絞り、効率的な Flex ボードを構成できます。

1. 画面上部の **フィルター** タブをクリックします。
2. 画面右に表示される **フィルター** ウィンドウでフィルタリング対象を選択します。すべて選択するには **すべて選択** を選択します。
   - **エージェント** オプション: エージェント種別またはアプリケーション一覧を選択、**検索** 機能あり
   - **メトリクス** オプション: プロジェクトまたはエージェント種別を選択、**+ And** 条件の追加や、**含む**、**除外** による文字列ベースの自動選択が可能
3. フィルタリング対象を選択したら、**適用** ボタンをクリックします。
4. よく使う条件は **フィルター保存** ボタンで保存します。

   a. フィルター名を入力し、**保存** を選択します。**保存フィルター** は **フィルター** ウィンドウ下部に表示されます。

フィルタリング条件

フィルタリング条件はエージェントタイプとメトリクスタイプに分かれます。
エージェントは基本的な監視対象のみ選択肢として提供し、メトリクスはプロジェクトで収集したすべてのメトリクスデータのタグを選択肢として提供します。メトリクスタイプは、非エージェント形式でデータを収集し、その後フィルタリングを適用する場合に有用です。メトリクスの詳細は次のドキュメントを参照してください。

タイプ	エージェント	メトリクス
選択オプション	WhaTap プロジェクトの監視対象 (アプリケーション、サーバー、データベース、 コンテナ、Kubernetes の POD と NODE)	エージェントを含むすべてのメトリクスタグ
適用範囲	すべてのウィジェット	メトリクスデータを使用するすべてのウィジェット
フィルタリング方式	チェックボックス	**一致, 含む, 除外**

参照時間を設定する

Flexボード の画面右上の時間セレクターで、ウィジェットのデータ参照時間を設定できます。

- ||: リアルタイムモードを停止して特定の参照時間に設定
- ⊙: リアルタイムモードに戻す

印刷する

Flex ボード画面を PDF で保存して印刷できます。

1. Flex ボード画面の右上にある 🖶 アイコンをクリックして印刷モードに切り替えます。
2. オプションを設定します。
   - **フィルターオプション表示**: 画面のフィルターオプションを表示/非表示に切り替え
   - **{mode}方向に切換え / {mode}方向に切換え**: 画面を横向きまたは縦向きに変更
3. 完了後、**PDFダウンロード** を選択します。
4. 印刷モードを終了するには、右上の ✕ **印刷モード終了** をクリックします。

Flex ボード管理

Flex ボードのレイアウト変更、ウィジェットの位置移動やサイズ調整、名称変更などの管理機能を提供します。 **Flex ボード管理** 機能は、**ダッシュボード** > **Flexボード** メニューの **ダッシュボードリスト** で ✎ アイコンをクリックして入

ることもできます。

1. Flex ボード画面の右上の ⚙ アイコンをクリックすると、**Flexボード管理** ウィンドウが開きます。
   - **名前変更**: Flex ボード名の変更
   - **レイアウト**: ウィジェットの位置・サイズ調整および削除
   - **他のユーザーの修正を許可しない**: 変更制限オプション
     - 編集権限のあるメンバーは編集・削除・変更が可能で、オプションを有効化すると **他のユーザーの修正を許可しない: On** に変更されます
   - **ウィジェット.json**: Flex ボードに含まれるウィジェットを JSON 形式で編集・管理
   - **オプション.json**: Flex ボードの設定を JSON 形式で編集・管理（プロジェクト情報、Flex ボードフィルター、時間選択などの設定値）

     > ⚠ **ウィジェット.json** と **オプション.json** 機能は、ダッシュボード管理者権限のユーザーにのみ提供します。

2. すべての設定を完了し、**保存** ボタンをクリックします。

ダッシュボードをお気に入りに登録する

作成した Flex ボードはお気に入りに登録して素早くアクセスできます。**ダッシュボード** > **Flexボード** の **ダッシュボードリスト** で、お気に入り登録するダッシュボードの ☆ ボタンをクリックすると、★ に変わります。

ウィジェットの管理

Flexボードに配置したウィジェットを管理して、ダッシュボードをユーザーがカスタマイズします。

> ダッシュボードはメトリクスウィジェットを含むさまざまなウィジェットテンプレートを活用して構成できます。
> 各テンプレートの特徴と使用方法については、ウィジェットテンプレート ドキュメントで詳しく確認できます。

ウィジェット追加

作成済みのFlexボードにウィジェットを追加できます。

1. **Flexボード**メニューの**ダッシュボードリスト**で、ウィジェットを追加するダッシュボードを選択します。
2. 画面左上の**編集モード**を選択します。
3. 左側の**ウィジェットテンプレート**一覧から目的のウィジェットを選択します。メトリクスウィジェットを追加するには、**すべてのメトリクス**を選択してから、目的のウィジェットをクリックします。

- ホーム画面の**統合Flexボード**を選択し、ウィジェットを追加した場合は、**プロジェクト選択**ボタンをクリックして、収集対象のプロジェクトを選択します。
- 追加するウィジェットがある場合は、同じプロセスを繰り返してください。

4. すべての手順が完了したら、画面上部の**ビューモード**を選択してレイアウトを確認し、(**リストへ**)ボタンをクリックします。

ウィジェットの属性変更

Flexボードメニューの**ダッシュボードリスト**で、ウィジェットのプロパティを変更するボードを選択します。**編集モード**に入ります。

ウィジェット名の変更

1. 名前を変更するウィジェットの名前の右側の✎ボタンをクリックします。
2. ウィジェットの**題名**、**フォントサイズ**、**色**を設定し、**保存**ボタンを選択します。
   - 保存する際に、**すべてのウィジェットに適用**トグルボタンの機能を活性化すると、フォントサイズと色が他のウィジェットにも適用されます。
   - ウィジェットのタイトルをロックするには👁ボタンを選択し、**保存**ボタンをクリックします。

時間設定

1. 時間を設定するウィジェットの右上の🕓ボタンを選択します。
2. **時間選択**ウィンドウで時間を設定してください。
   - ダッシュボード画面で設定した時間を適用するには、**ダッシュボード設定に従う**をチェックします。ダッシュボードの時間設定は右上にあります。
   - 特定の時間を設定する場合は、**ダッシュボード設定に従う**のチェックを外します。〈または〉ボタンを使用して照会する時間を設定します。
3. 時間設定が終了したら✕ボタンをクリックします。

> ⓘ ウィジェットのデータタイプによっては、時間設定機能をサポートしていない場合があります。

出力データオプションの設定

チャート型ウィジェットの出力データオプションを設定して、ユーザーが必要なデータを選択できます。

1. ウィジェットの右上にある ⋮ アイコンをクリックして、出力データのオプションを設定します。
   - プロジェクト、エージェント、アプリケーションなどを選択してデータをフィルタリングできます。
   - 個々のデータまたはマージされたデータを選択して、チャートに表示されるグラフのタイプを変更できます。
   - ⋮ > ⚙を選択します。**チャート設定**画面で詳細オプションを変更できます。設定できるオプションは次

のとおりです。

- **チャートタイプ**：集計する現在の数値をチャートに表示するオプションです。
- **y軸 チャート最大値**：Y軸にチャートの最大値を**自動**または**固定**で選択できます。
- **水平線設定**：**追加**ボタンを選択し、しきい値または範囲を水平線として表示できます。複数を追加して適用できます。水平線を追加した後は、**適用**ボタンを選択してチャートに反映してください。
- **チャートの最大値を表示**：チャートに表示されるグラフの最大値を表示します。
- **サブチャート**：ウィジェットのメインチャートにサブチャートを追加できます。サブチャートの種類を選択し、レイアウトとサイズ、データのマージ有無を選択します。他のウィジェットにも適用する場合は、**すべてのウィジェットに適用**ボタンを選択します。サブチャートは、データをさまざまな方法で確認するとき便利です。

2. オプション設定がすべて完了したら、**チャート設定**画面の左上の×ボタンをクリックするか、**チャート設定**領域の外側をクリックします。

> ⓘ ウィジェットのデータタイプによって選択できるオプションは異なる場合があります。

メトリクスウィジェットデータの変更

メトリクスウィジェットのデータ照会の条件を変更できます。

1. **Flexボード** > **ダッシュボードリスト**で、ウィジェットデータを変更するダッシュボードを選択します。
2. **編集モード**で、変更したいウィジェットの右上にある ⋮ アイコンをクリックします。
3. <> アイコンをクリックすると、データ設定編集ウィンドウが表示されます。
4. 次を参照して設定を変更するには、適用アイコン**ウィジェットに適用**ボタンをクリックします。
   - ウィジェットを作成すると、デフォルトで生成されるメトリクスのオプションがタブに表示されます。
   - メトリクス オプションには、**カテゴリ**と**フィールド**、**タグ**が含まれます。
   - **カテゴリ**項目は現在、変更を加える機能を制限しています。
   - **フィールド**、**タグ**項目は複数選択できます。選択したタグオプションでチャートデータをグループ化します。
   - 🔍ボタンをクリックすると、元のデータを表示できます。

- **オブジェクトのマージ**および**時間マージ**項目の詳細については、データマージオプションを参照してください。

データマージオプション

Flexボードの**すべてのメトリクス**一覧に配置されたマトリクスウィジェットでは、データのマージオプションを提供します。 データのマージは、**オブジェクトのマージ**と**時間マージ**方法を提供します。

- **オブジェクトのマージ**

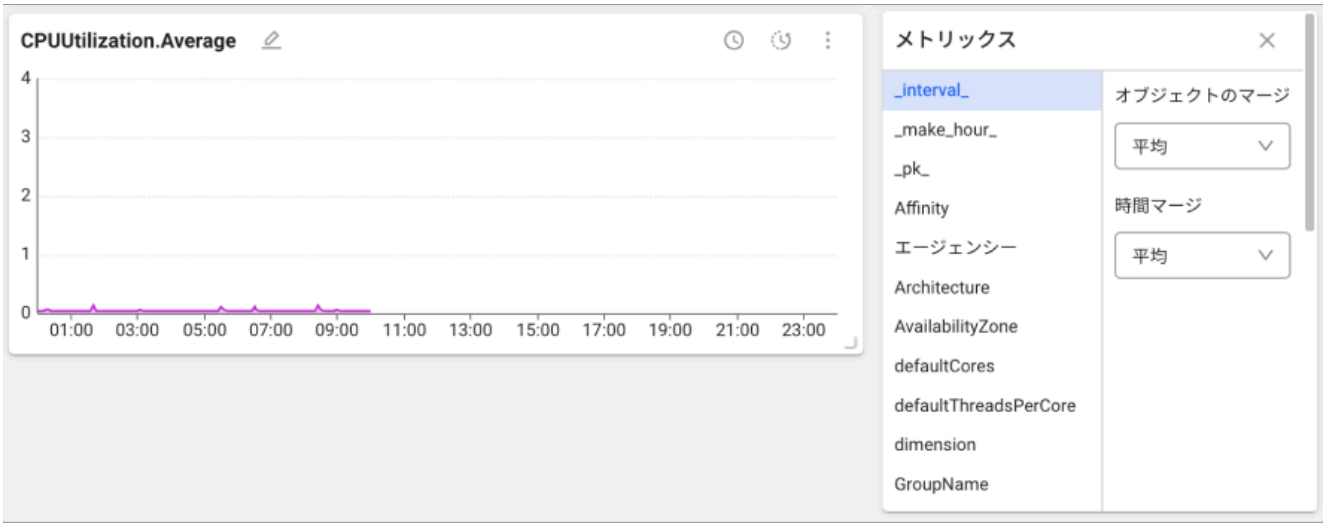
  左側で選択したタグ(例：エージェント名)に基づいてチャートデータをグループ化するために使用します。この場合、異なるフィールド(例：CPU)値を持つデータ間でタグが一致する場合、そのデータは結合されます。

- **時間マージ**

  主に検索時間が長い場合に使用します。ソースデータとフィールド値が同じデータを時間範囲に基づいて一定の時間間隔(5分または1時間)でデータを結合します。この結合方法は、時間結合と呼ばれます。

- **AWS CloudWatchメトリクス**

  AWSモニタリング指標の場合、統計手法はフィールド名の後にサフィックス(接尾辞) として追加されます。データマージの既定値は、このサフィックスを参照しています。



1. **編集モード**で**すべてのメトリクス**ボタンを選択し、**ウィジェットテンプレート**一覧を**すべてのメトリクス**一覧に変更します。
2. メトリクスウィジェットをレイアウトに配置します。

3. メトリクス ウィジェットの右上にある ⋮ ボタンを選択します。
4. タグ(例: エージェント名)オプションを選択すると、データ結合オプションが表示されます。ウィジェットのサイズによっては、メトリクス オプションの位置が異なる場合があります。

ウィジェットのコピー

1. ウィジェットの右上にある ⋮ ボタンをクリックします。
2. ボタンをクリックします。
3. **コピー**画面が表示されたら、＋ **ウィジェット追加**を選択します。
4. 一覧にあるウィジェットの名前を変更します。
5. **適用**ボタンを選択します。
6. **編集モード**画面でコピーしたウィジェットで確認できます。

ウィジェットの削除

1. ウィジェットの右上にある ⋮ ボタンをクリックします。
2. ✕ ボタンをクリックします。
3. 確認メッセージの画面が表示されると、**削除**ボタンを選択します。削除を取り消すには**キャンセル**ボタンをクリックします。

テンプレート一覧

OpenMetrics を初めて使うユーザーが複雑な設定なしで素早くメトリクスを収集・可視化できるよう、デフォルトテンプレートと事前定義テンプレートを提供します。これにより、よく使われるオープンソースメトリクスを簡単に適用でき、初期導入のハードルを下げ、運用者が拡張可能なモニタリング環境を構築できるよう支援します。

テンプレート一覧

テンプレート一覧 は、基本的なオープンソーステンプレートカードを提供します。
各カードには、簡単な説明、テンプレートへのリンク、収集ガイドが含まれています。

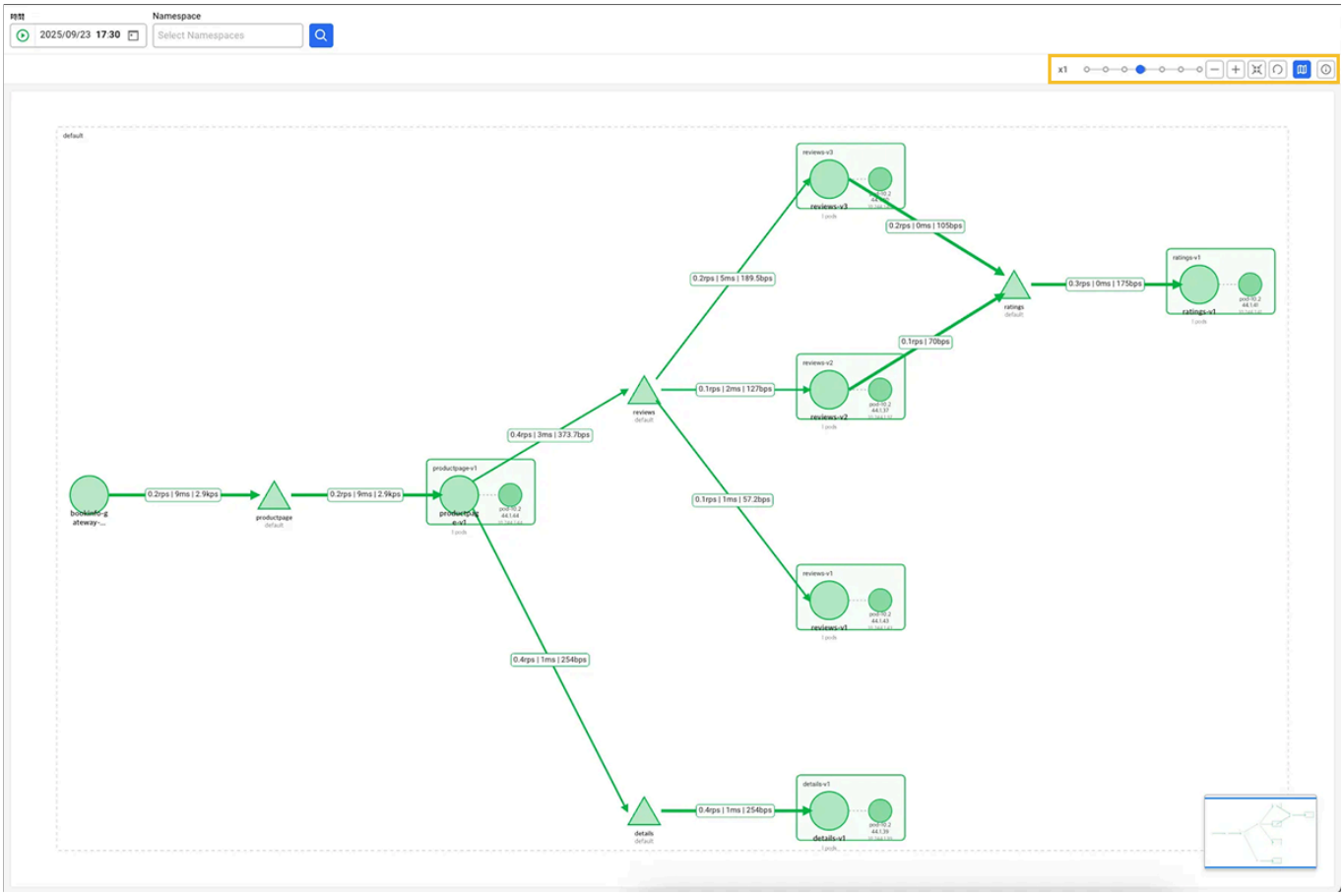
- → **テンプレート**: クリックすると対象テンプレートへ直接移動します。
- ⓘ **収集案内**: クリックするとデータ収集の設定方法を表示します。

Nginx テンプレート

Apache Nginx を監視し、Webサーバーの性能をリアルタイムに追跡・分析できます。

Istio テンプレート

Istio サービスメッシュ環境におけるサービス間の関係とトラフィックフローを可視化し、WhaTap APM データと連携して統合モニタリング環境をテンプレートとして提供します。画面右上のボタンでテンプレートを拡大または縮小できます。



凡例

画面右上の ⓘ ボタンをクリックすると凡例を確認できます。

ノード形状

- 三角形: Kubernetes サービス
- 円: ワークロード
- 四角形: アプリケーション
- 小さな円: Pod

ノード状態

HTTP リクエスト・レスポンス全体におけるエラー比率(4xx, 5xx)

- 緑: エラー率 5% 未満
- 橙: エラー率 5% 以上
- 赤: エラー率 10% 以上
- グレー: リクエストなし

Pod ノード状態

Kubernetes Pod のフェーズ

- 緑: Running 状態の Pod
- 橙: Pending 状態の Pod
- 赤: Failed 状態の Pod

エッジの色

2 つのノード間の HTTP リクエストで発生したエラー比率(4xx, 5xx)

- 緑: エラー率 10% 未満
- 橙: エラー率 10% 以上
- 赤: エラー率 20% 以上
- 青: TCP 通信

OpenMetricsエクスプローラー

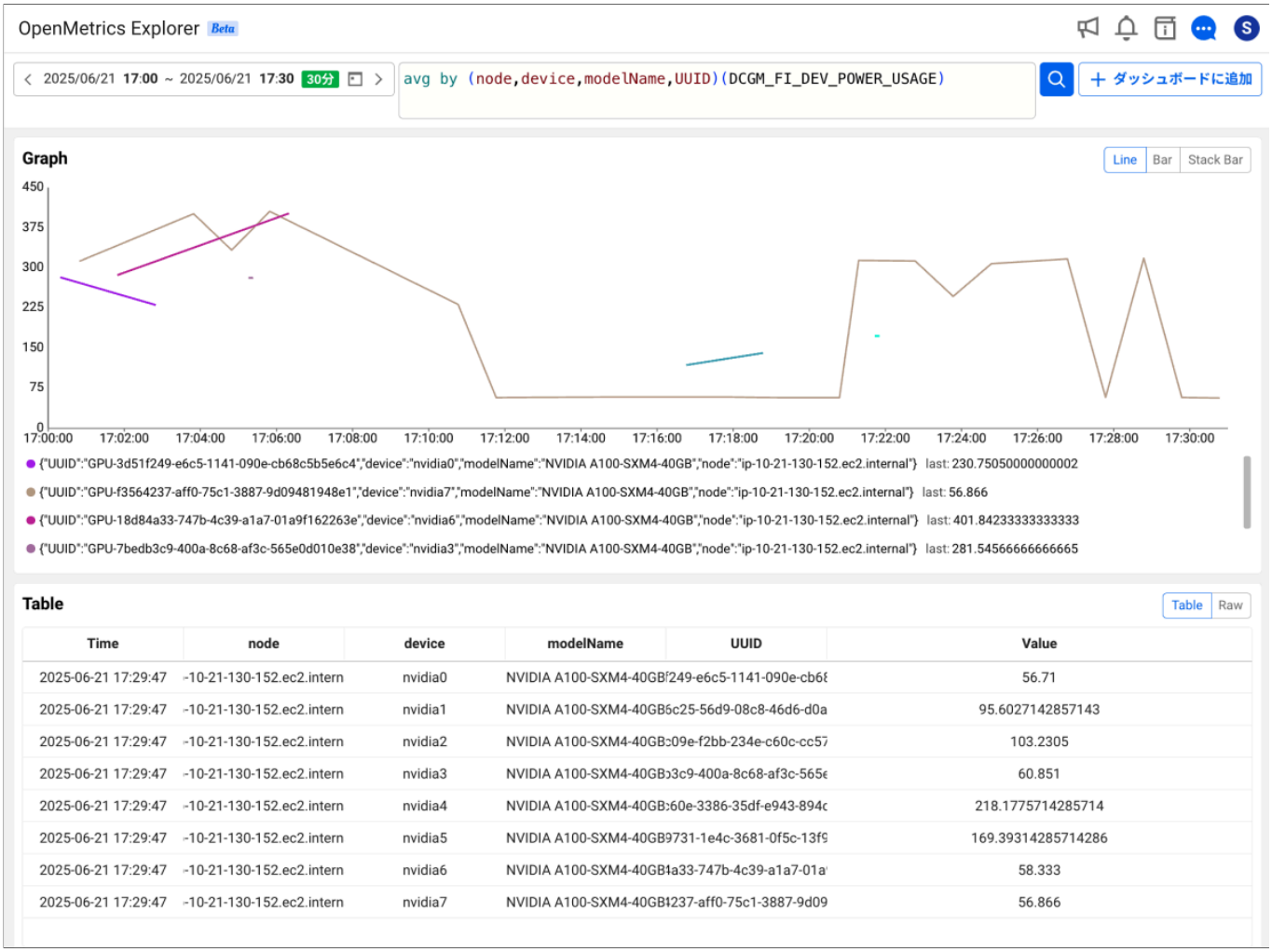
OpenMetrics エクスプローラーは、Prometheusベースのカスタムメトリックを PromQLで直接探索し、さまざまな形式で視覚化してメトリックの状態を確認したり、インサイトを得ることができる機能です。

この機能を使用することで、必要なメトリックを自由に照会し、有意義なデータを抽出したり、ダッシュボードと連携して継続的にモニタリングすることができます。

> ⓘ オープンメトリックエクスプローラーを使用するには、オープンエージェントのインストールが必要です。
>
> - WhaTapオペレーターのインストールドキュメントを参照してください。

PromQLを探索する

OpenMetrics エクスプローラーでPromQLを実行し、さまざまな可視化オプションで結果を確認できます。



Graph

- **Line**: 基本的な時系列のラインチャート
- **Bar**: バーチャート形式での可視化
- **StackBar**: 積み上げられた値を表示するスタック型バーチャート

Table / Raw

- **Table**: 各時系列の値を時間順に並べて表形式で表示
- **Raw**: Prometheus APIのレスポンス（JSON形式）をそのまま確認可能

PromQL文法のサポートおよび自動補完

`node_cpu_seconds_total`、`container_memory_usage_bytes`などのメトリクスを入力すると、自動補完機能によりメトリクスおよび演算子のリストが表示されます。

`{pod="example"}`形式のラベルフィルタリングもサポートされています。

ダッシュボードに追加する

取得したPromQLクエリをダッシュボードウィジェットとして保存することで、希望するメトリクスを継続的にモニタリングできます。

追加方法

1. **オープンメトリック探索機**画面右上の**+ ダッシュボードに追加**ボタンをクリックすると、**ダッシュボードに追加**モーダルウィンドウが表示されます。
2. **ダッシュボードに追加**ウィンドウで、既存のダッシュボードを選択するか、新規のダッシュボードを作成します。
3. **適用**ボタンをクリックすると、該当のウィジェットが指定したダッシュボードに追加されます。
    - 追加されたダッシュボードでウィジェットの名前と説明を入力できます。
    - 追加されたウィジェットはFlexボードのGridレイアウトで自由に配置を調整でき、PromQLの修正も可能です。

クラスターサマリー

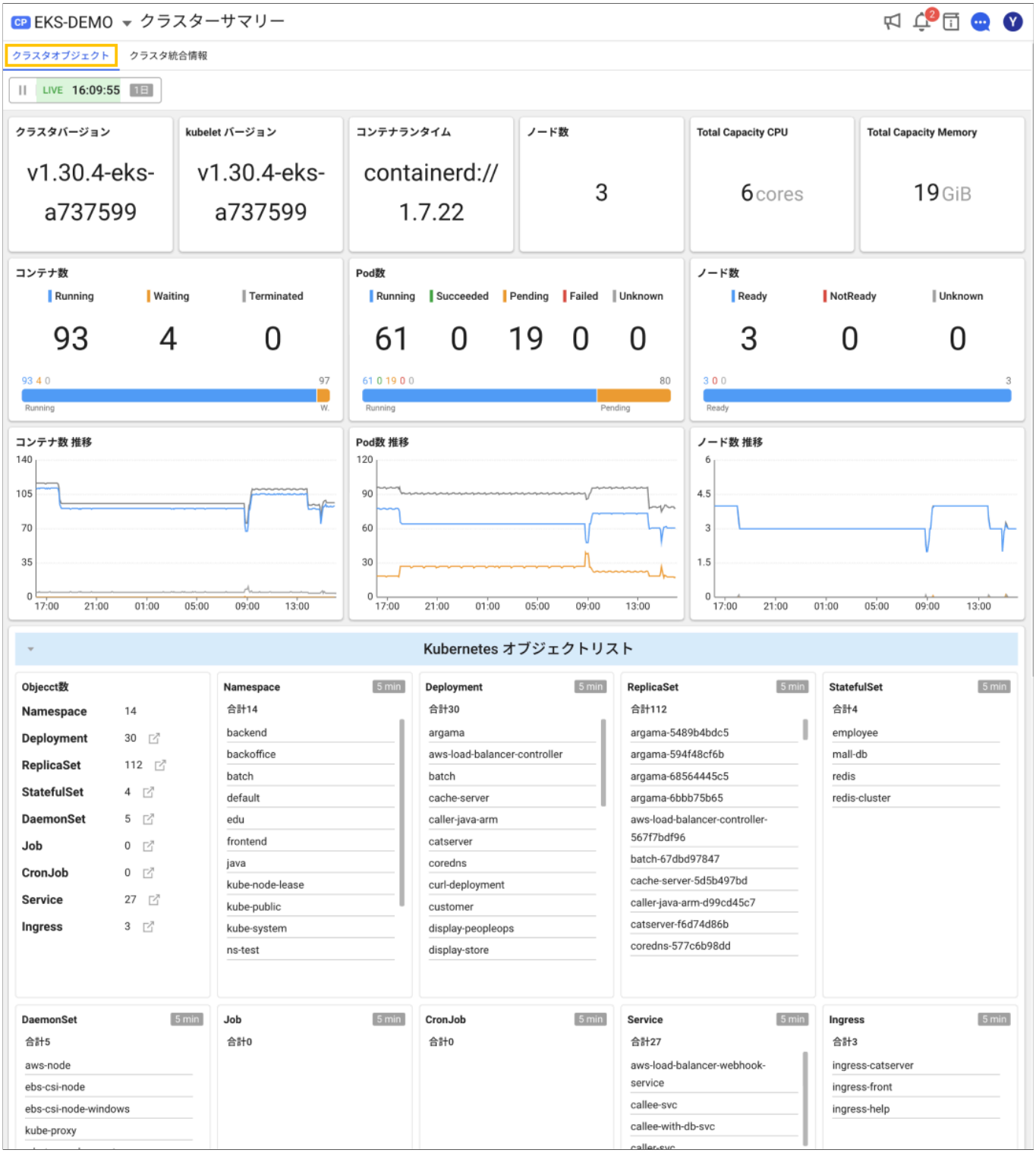
> **要求事項**
>
> クバネティスクラスタープロジェクト(CP)である場合のみ**クラスターサマリー**メニューを確認できます。

クラスター関連のサマリー情報を確認できます。 ネームスペースプロジェクトで分けられた情報を統合して確認することができます。

クラスターオブジェクト

クラスターに関連するバージョン情報やクラスター内部のノード、Pod、コンテナなど、Kubernetesオブジェクトの数を確認できます。

- クバネティスエージェント1.4.1バージョン以降の場合、クラスターバージョンを確認することができます。
- クバネティスエージェント1.3.6バージョン以降の場合、次の事項を確認できます。
  - kubelet バージョン
  - コンテナランタイム


EKS-DEMO クラスターサマリー
クラスタオブジェクト
クラスタ統合情報
LIVE 16:09:55 1日
クラスタバージョン
v1.30.4-eks-a737599
kubelet バージョン
v1.30.4-eks-a737599
コンテナランタイム
containerd://1.7.22
ノード数
3
Total Capacity CPU
6 cores
Total Capacity Memory
19 GiB
コンテナ数
Running 93
Waiting 4
Terminated 0
Pod数
Running 61
Succeeded 0
Pending 19
Failed 0
Unknown 0
ノード数
Ready 3
NotReady 0
Unknown 0
コンテナ数 推移
Pod数 推移
ノード数 推移
Kubernetes オブジェクトリスト
Objecct数
Namespace 14
Deployment 30
ReplicaSet 112
StatefulSet 4
DaemonSet 5
Job 0
CronJob 0
Service 27
Ingress 3
Namespace 5 min
合計14
backend
backoffice
batch
default
edu
frontend
java
kube-node-lease
kube-public
kube-system
ns-test
Deployment 5 min
合計30
argama
aws-load-balancer-controller
batch
cache-server
caller-java-arm
catserver
coredns
curl-deployment
customer
display-peopleops
display-store
ReplicaSet 5 min
合計112
argama-5489b4bdc5
argama-594f48cf6b
argama-68564445c5
argama-6bbb75b65
aws-load-balancer-controller-567f7bdf96
batch-67dbd97847
cache-server-5d5b497bd
caller-java-arm-d99cd45c7
catserver-f6d74d86b
coredns-577c6b98dd
StatefulSet 5 min
合計4
employee
mall-db
redis
redis-cluster
DaemonSet 5 min
合計5
aws-node
ebs-csi-node
ebs-csi-node-windows
kube-proxy
Job 5 min
合計0
CronJob 5 min
合計0
Service 5 min
合計27
aws-load-balancer-webhook-service
callee-svc
callee-with-db-svc
Ingress 5 min
合計3
ingress-catserver
ingress-front
ingress-help

クラスタバージョン

- 5秒サイクルのデータです。 リアルタイムデータ照会時、5秒間隔でデータを読込みします。

kubelet バージョン

- 5秒サイクルのデータです。 リアルタイムデータ照会時、5秒間隔でデータを読込みします。

コンテナランタイム

- 5秒サイクルのデータです。 リアルタイムデータ照会時、5秒間隔でデータを読込みします。

コンテナ数

- 状態別コンテナの数を確認できます。
- 5秒サイクルのデータです。 リアルタイムデータ照会時、5秒間隔でデータを読込みします。

Pod数

- 状態別Podの個数を確認できます。
- 5秒サイクルのデータです。 リアルタイムデータ照会時、5秒間隔でデータを読込みします。

ノード数

- 状態別ノード数を確認できます。
- 5秒サイクルのデータです。 リアルタイムデータ照会時、5秒間隔でデータを読込みします。

Kubernetes オブジェクトリスト

- クバネティスオブジェクトの一覧を確認できます。
- 5分周期のデータです。 リアルタイムデータ照会時に5分間隔でデータを読込みします。

オブジェクト数の推移

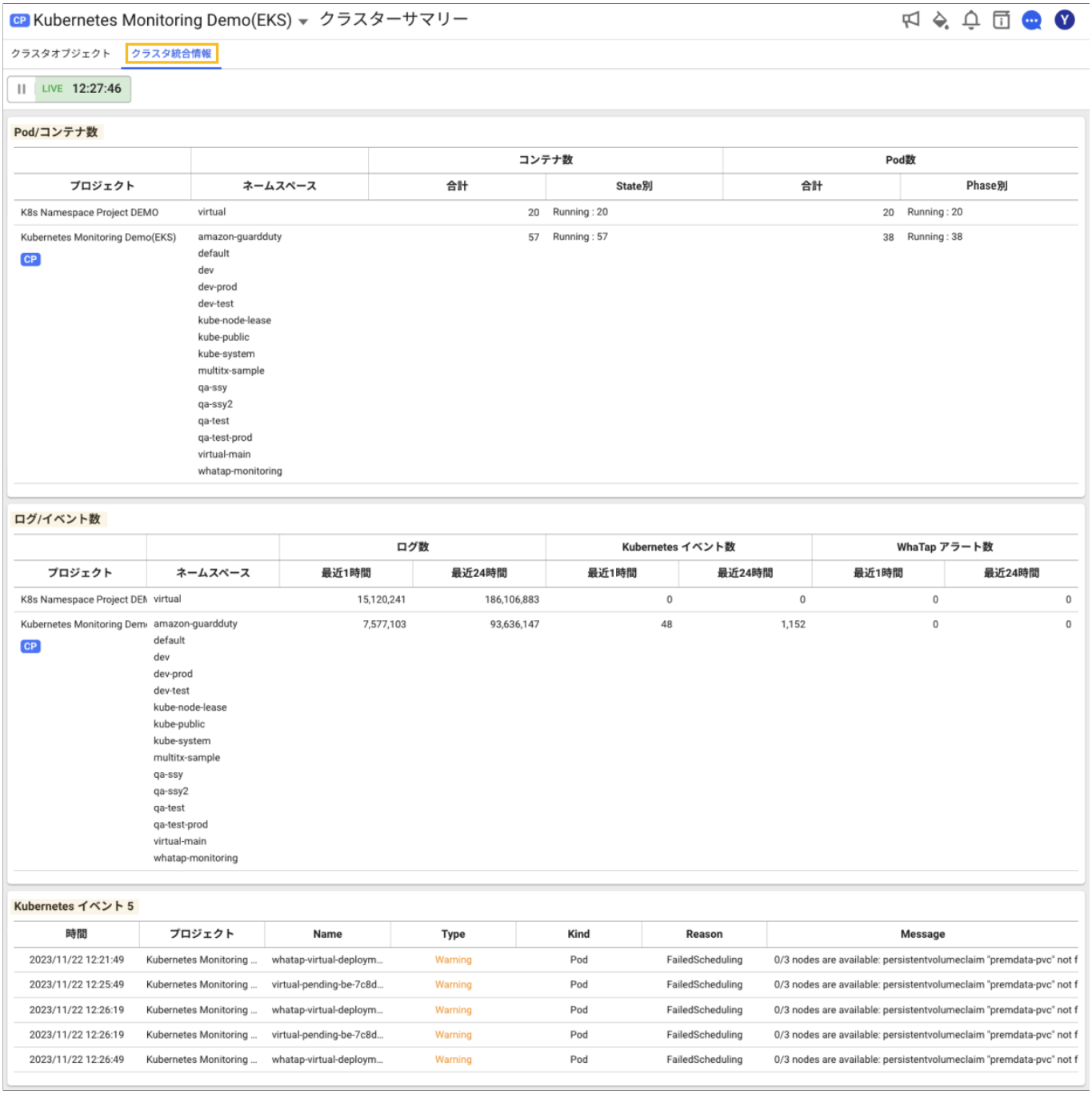
- クバネティスオブジェクトの個数が24時間の間にどのように変化したかを確認できます。

- 5分周期のデータです。 リアルタイムデータ照会時に5分間隔でデータを読込みします。

クラスター統合情報

クラスタープロジェクト情報とネームスペースプロジェクトに分離されたPod、コンテナ、ログ、イベント数、イベント一覧を照会することができます。

> ⓘ データを照会するには、該当ネームスペースプロジェクト**読み込み**権限が必要です。



Pod/コンテナ数

コンテナ数とPod数をプロジェクト単位で確認できます。

- コンテナ数
  - 全体: クラスターに属するコンテナ全体の数を表示します。
  - State別: 状態ごとに進行中のコンテナの数を表示します。
- Pod数
  - 全体: クラスターに属する全てのPod数を表示します。
  - Phase別: 状態別に進行中のPod数を表示します。

ログ/イベント数

ログ、Kubernetesイベント、WhaTapアラートの件数をプロジェクト単位で確認することができます。

- ログ数
  - 最近１時間: 最近１時間の間に追加されたログラインの数を表示します。
  - 最近24時間: 1日間で追加されたログラインの数を表示します。
- Kubernetes イベント数
  - 最近1時間: 最近１時間の間に追加されたイベントラインの数を表示します。
  - 最近24時間: 1日間で追加されたイベントラインの数を表示します。
- WhaTap アラート数
  - 最近１時間: 最近１時間で送信されたアラートの数を表示します。
  - 最近24時間: 一日間で送信されたアラートの数を表示します。

Kubernetesイベント5

クラスタープロジェクトとネームスペースプロジェクトで発生した最近の5つのKubernetesイベントの一覧を確認できます。

ネームスペース状況

> Kubernetesクラスタープロジェクト(CP)の場合、該当メニューを照会できます。

ネームスペース状況メニューを通じてWhaTap Kubernetesクラスタープロジェクトおよび下位ネームスペースプロジェクトの状況を確認できます。ネームスペースプロジェクトではアクセスできないメニューです。



ネームスペース状況メニューは、プロジェクトリスト画面と似ていますが、次のように違いがあります。

- **プロジェクトリスト**

  ネームスペースプロジェクトを作成した後、プロジェクトを別のグループに移動できますが、ネームスペースを一目で確認するのことは困難です。

- **ネームスペース状況**
  - クラスタープロジェクトと同じグループに属していないネームスペースも確認できます。
  - ひと目で関連プロジェクトをすべて確認できます。

クラスタープロジェクトの場合、Kubernetesプラットフォームを表す青色 Kubernetes のタグが表示されます。ネー

ムスペースプロジェクトには、 Namspace タグが表示されます。ウィジェット左上のタイトルはプロジェクト名で、ネームスペース名は一番下の Namspace タグの横に表示されます。

ノードリスト

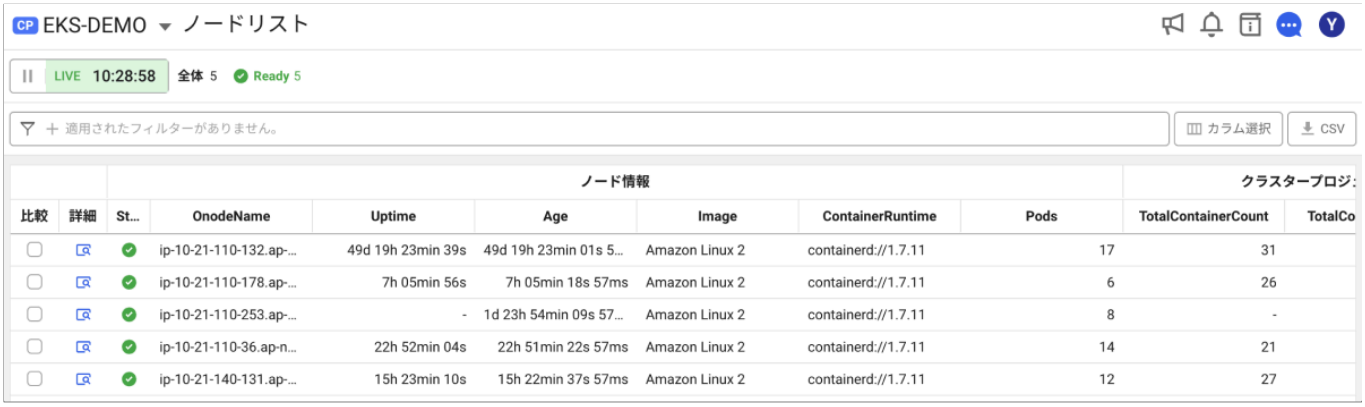
> 特定のカラムのデータが表示されない場合は、クバネティスノードエージェントが旧バージョンである可能性があります。その場合、十分なデータが収集されるように、クバネティスノードエージェントバージョンを1.3.7以降に更新してください。

現在のプロジェクトのノードの情報を一覧形式で確認できます。

- 10秒サイクルでリアルタイムのデータを更新します。
- ノード状態のサマリー、比較、フィルター、および_CSV_ファイルのダウンロード機能を提供します。

  CSV ファイル名、kubernetes_node_list_{pcode}{timestamp}.csv_

基本画面ガイド

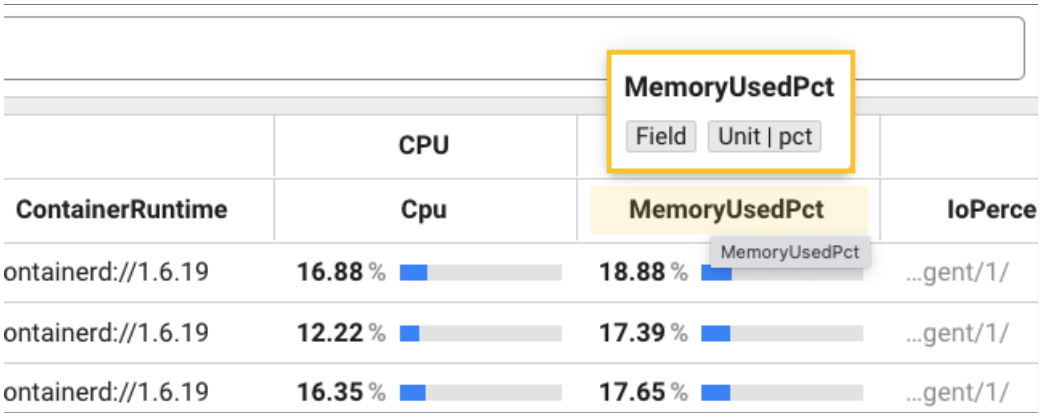


ノードリストの上段から希望するオプションを選択してデータを照会できます。

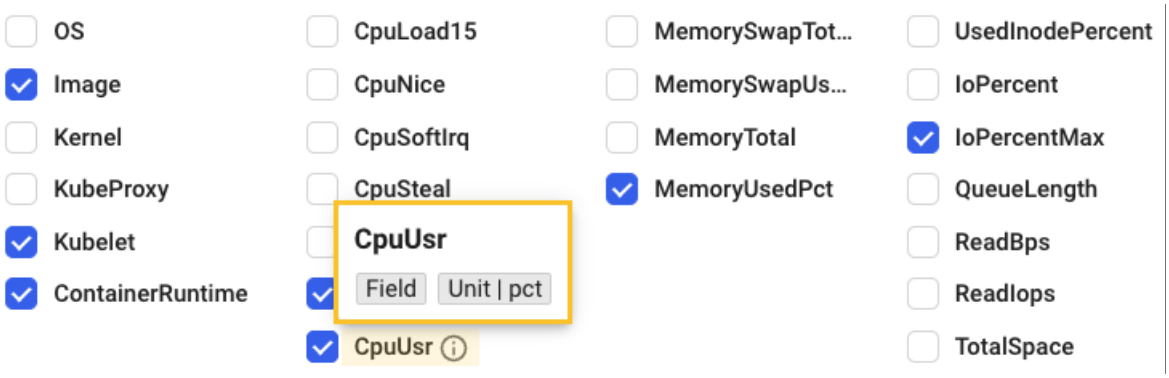
- 全ノード数と状態別ノード数を確認できます。
- 希望するカラムを選択する場合、そのカラムを基準に**ノードリスト**をソートして照会できます。
- **ノードリスト**コラムの左側にある拡張ボタンをクリックすると、ディスク情報を確認できます。

メトリクス情報

- 目的のカラムの上にマウスカーソルを置くと、次のようにカラムが提供するメトリクス情報を確認できます。



- カラム選択オプション画面内のカラム一覧でカラム別メトリクス情報を確認することができます。 ご希望のカラムの上にマウスカーソルを置くと、カラム名の右側にⓘ インフォアイコンが表示されます。 ⓘ インフォアイコン上にマウスカーソルを位置されると、次のようにメトリクス情報を確認できます。



フィルター

フィルターオプションを活用する時、希望するノード一覧を選択して照会できます。 `and` 条件を使用すると、複数のフィルターが適用できます。

1. フィルター入力ウインドウで＋追加アイコンを選択すると、次のようにフィルターを追加するウィンドウが表示されます。

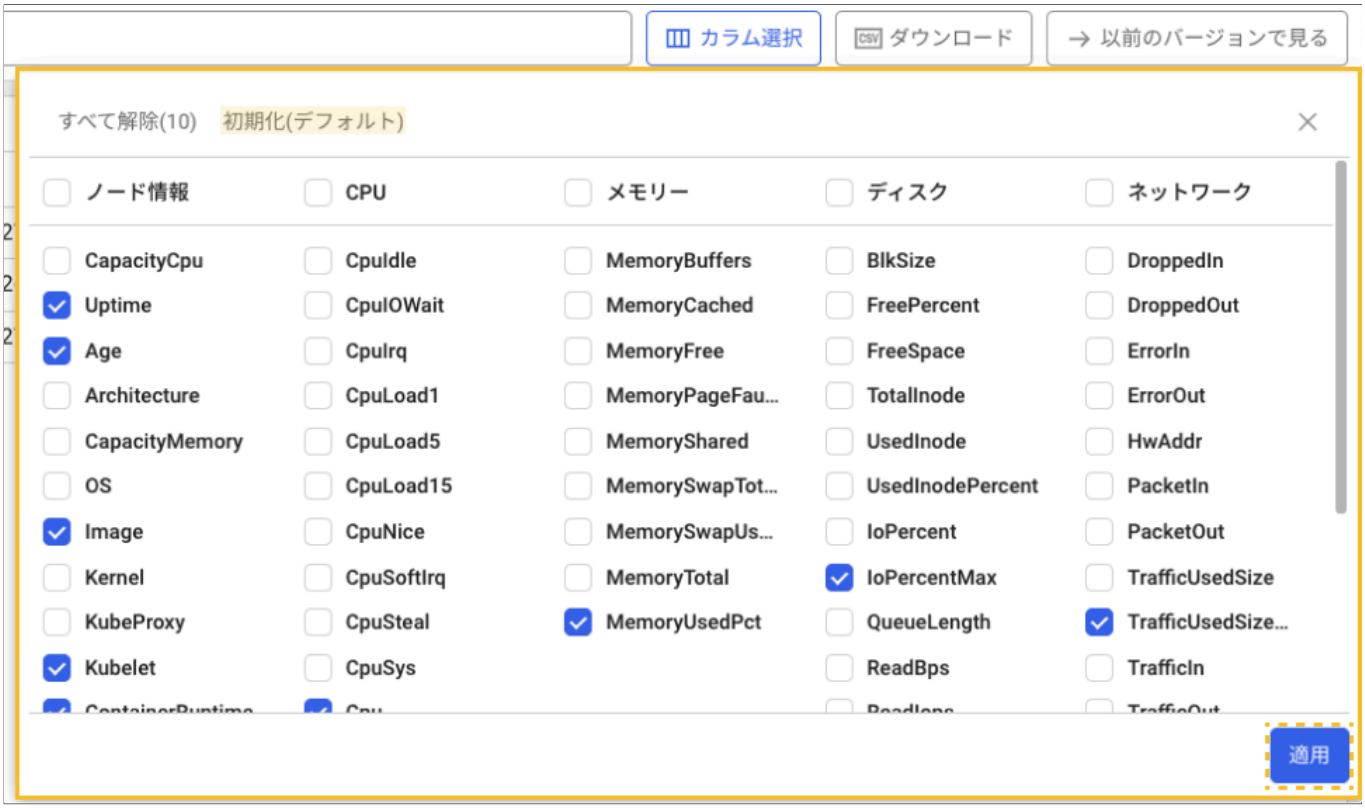


- フィルターキーと条件を選択します。
- 条件に合った値を選択します。

2. フィルター設定後、適用ボタンをクリックし、フィルターを適用します。
3. 複数のフィルターを追加するには、フィルター入力ウインドウまたはフィルターを追加するウィンドウで＋追加アイコンを選択します。

カラム選択

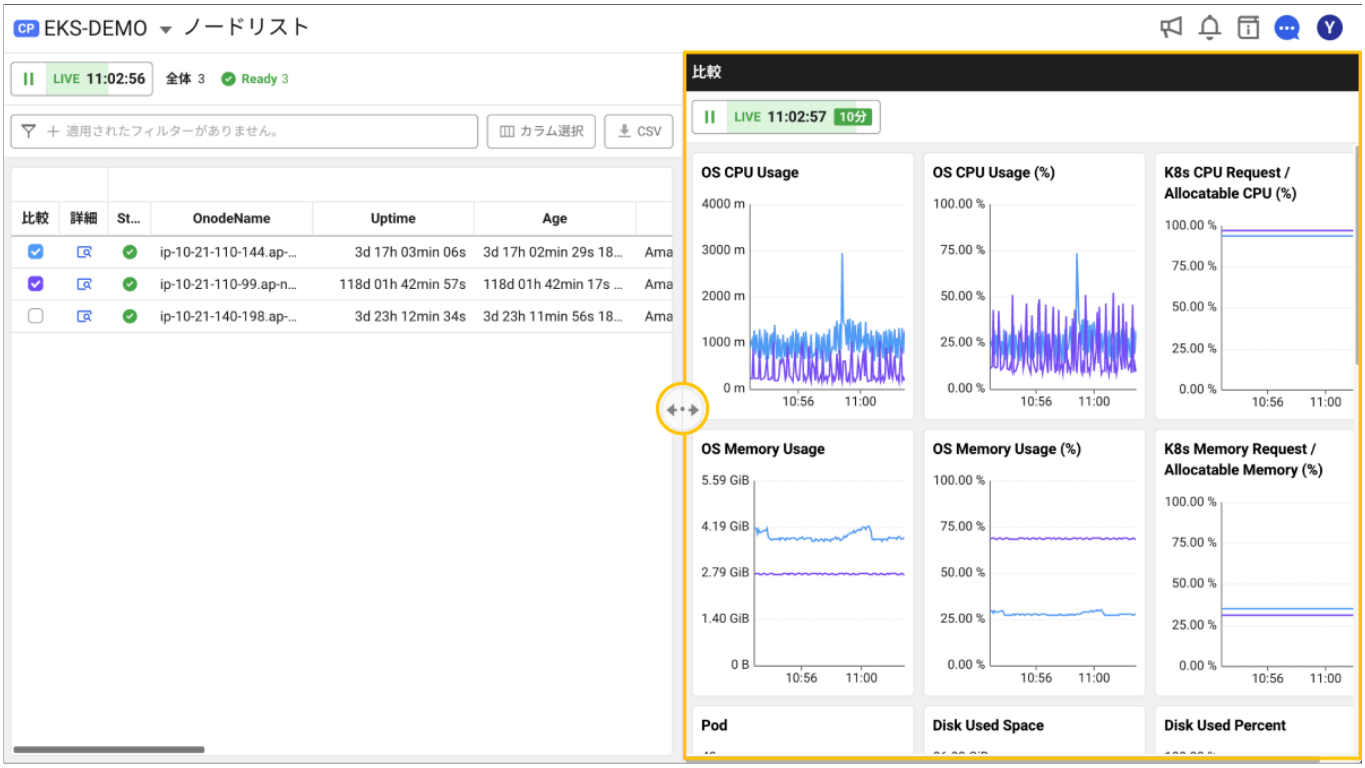
カラム選択オプションを活用する時、希望するノード情報を選択して一覧を照会できます。



- 選択したカラム一覧は、メニューを再入力すると維持されます。
- リセットボタンを選択すると、定義済みの既定の主要指標カラムが選択されます。 この場合、既存のカラム選択は削除されます。
- カラムを選択した後、右下の適用ボタンをクリックして選択履歴を適用することができます。

比較する

比較オプションにより、複数のノードの主な指標に対する比較チャートを確認できます。リストから照会するノードの比較カラムのチェックボックスを選択すると、次のように右側にそのノードの詳細情報を比較する画面が表示されます。



詳細を確認

ノードの詳細は詳細オプションを使用して、選択したノードに関する詳細情報を確認できます。一覧から照会を希望するノード詳細カラムのアイコンを選択すると、右側にそのノードの詳細画面が表示されます。

ノード 構成

ノード 構成タブで選択したノード構成と全体的なリソース使用量の情報を要約して提供します。 そのノードのハードウェアとソフトウェアの構成要素を総合的に確認できます。 ユーザーはノードのリソース使用状況と各構成要素の状態を把握し、パフォーマンスと運用安定性をモニタリングできます。


照会時間: 2024/09/20 14:40:30 ~ 2024/09/20 18:11:25
この統計データベースの要約分析は、詳細分析やシステム改善のための参考資料として活用できます。
ip-10-21-140-100.ap-northeast-1.compute.internal は、 CPU 2個、メモリ3.77 GiBのamd64 linuxノードです。 このノードはほとんどの時間にわたって余裕を持っていますが、断続的にCPU, メモリで中程度の負荷が発生する状態です。安定した運用が必要な場合は、
定期的に負荷が発生した時点での監視をお勧めします。
ネットワーク使用量が最大1.31 MiBで、平均使用量54.44 kiBよりも約24.69倍高く、
トラフィックパターンのボラティリティを確認する必要があります。
AVG
MAX
Architecture amd64 Image Amazon Linux 2 CapacityMemory 3.77 GiB CapacityCpu 2
Pod
12.51
Container Runtime
containerd://1.7.11
kubelet
v1.30.2-eks-1552ad0
kube-proxy
v1.30.2-eks-1552ad0
CPU
1380 m (69.00 %)
メモリ
1.04 GiB (30.50 %)
ディスク (/dev/nvme0n1p1)
4.51 GiB (5.64 %)
ネットワーク
1.31 MiB

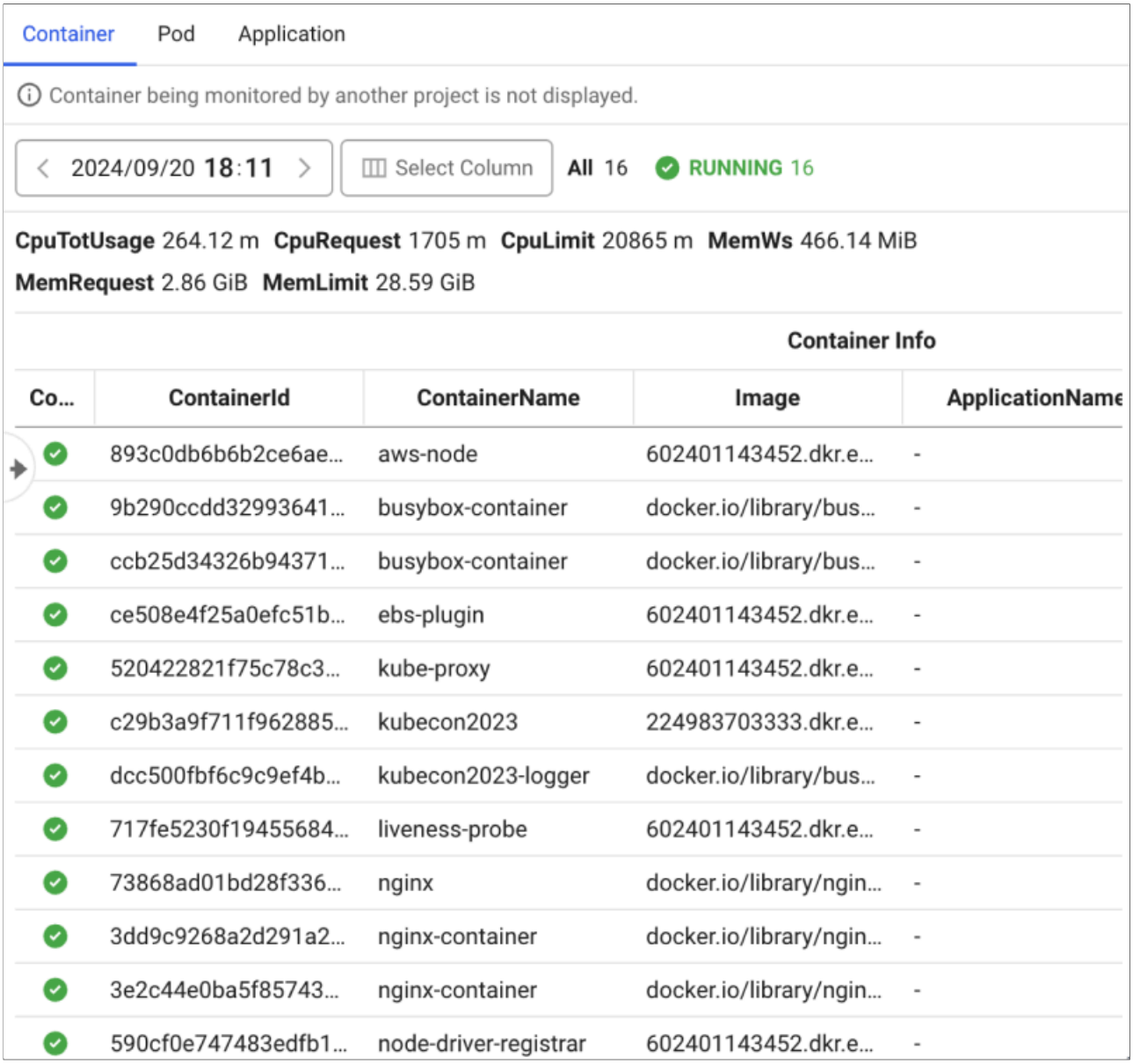
- **ノードシステム仕様**
  - **Architecture**：ノードが使用しているハードウェアアーキテクチャ
  - **Image**：ノードが実行中のオペレーティングシステムのイメージ情報
  - **CapacityMemory**：ノードの総メモリ容量
  - **CapacityCpu**：ノードの総CPU容量
- **ソフトウェア構成**
  - **Container Runtime**：ノードで使用されるコンテナランタイムのバージョン情報
  - **kubelet**：ノードで実行中のクバネティスkubeletのバージョン情報
  - **kube-proxy**：ネットワーク管理の役割を果たすkube-proxyのバージョン情報
- **リソースの使用状況**
  - **CPU**：ノードのCPU使用量
  - **メモリ**：ノードのメモリ使用量
  - **ディスク**：ノードで使用中のディスク容量と使用比率
  - **ネットワーク**：ネットワークの伝送量

> ⓘ
> - CPU、メモリ、ディスク、ネットワークをクリックすると、照会期間中の使用量の推移が確認できる**メトリクス**タブに移動します。
> - **AVG**、**MAX**ボタンをクリックして、照会期間中の平均使用量と最大使用量を確認できます。

> ⓘ **ノード構成**タブはクラスタープロジェクト(CP)の場合、照会できます。

リソース

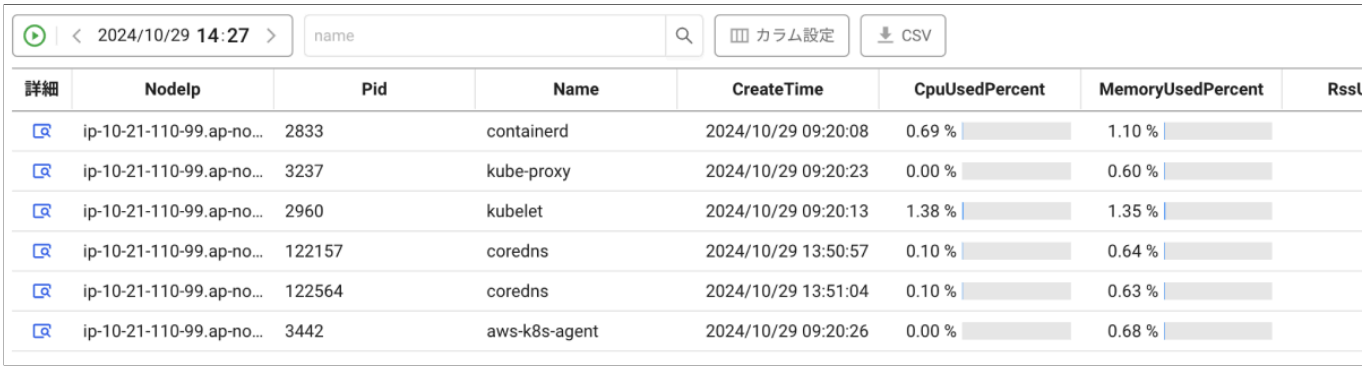
リソースタブで選択したノードにスケジューリングされた**Pod**およびそのPodで実行中の**コンテナ**と**アプリケーション**一覧を確認できます。 その情報は、選択した時間区間の最後の時点を基準に1時間のリソースデータを反映して表示します。 これにより、その時点のノード状態と各Pod、コンテナ、アプリケーションの動作状態をより明確に把握できます。



要約カラムを選択する時、コンテナ及びPod要約分析を確認できます。

プロセス

プロセスタブで選択したノード内部にある主要なKubernetes管理プロセス情報を確認できます。 プロセスの問題によって引き起こされるクラスター障害を予防し、ダウンタイムを最小限にできます。



詳細	NodeIp	Pid	Name	CreateTime	CpuUsedPercent	MemoryUsedPercent	Rssl
	ip-10-21-110-99.ap-no...	2833	containerd	2024/10/29 09:20:08	0.69 %	1.10 %	
	ip-10-21-110-99.ap-no...	3237	kube-proxy	2024/10/29 09:20:23	0.00 %	0.60 %	
	ip-10-21-110-99.ap-no...	2960	kubelet	2024/10/29 09:20:13	1.38 %	1.35 %	
	ip-10-21-110-99.ap-no...	122157	coredns	2024/10/29 13:50:57	0.10 %	0.64 %	
	ip-10-21-110-99.ap-no...	122564	coredns	2024/10/29 13:51:04	0.10 %	0.63 %	
	ip-10-21-110-99.ap-no...	3442	aws-k8s-agent	2024/10/29 09:20:26	0.00 %	0.68 %	

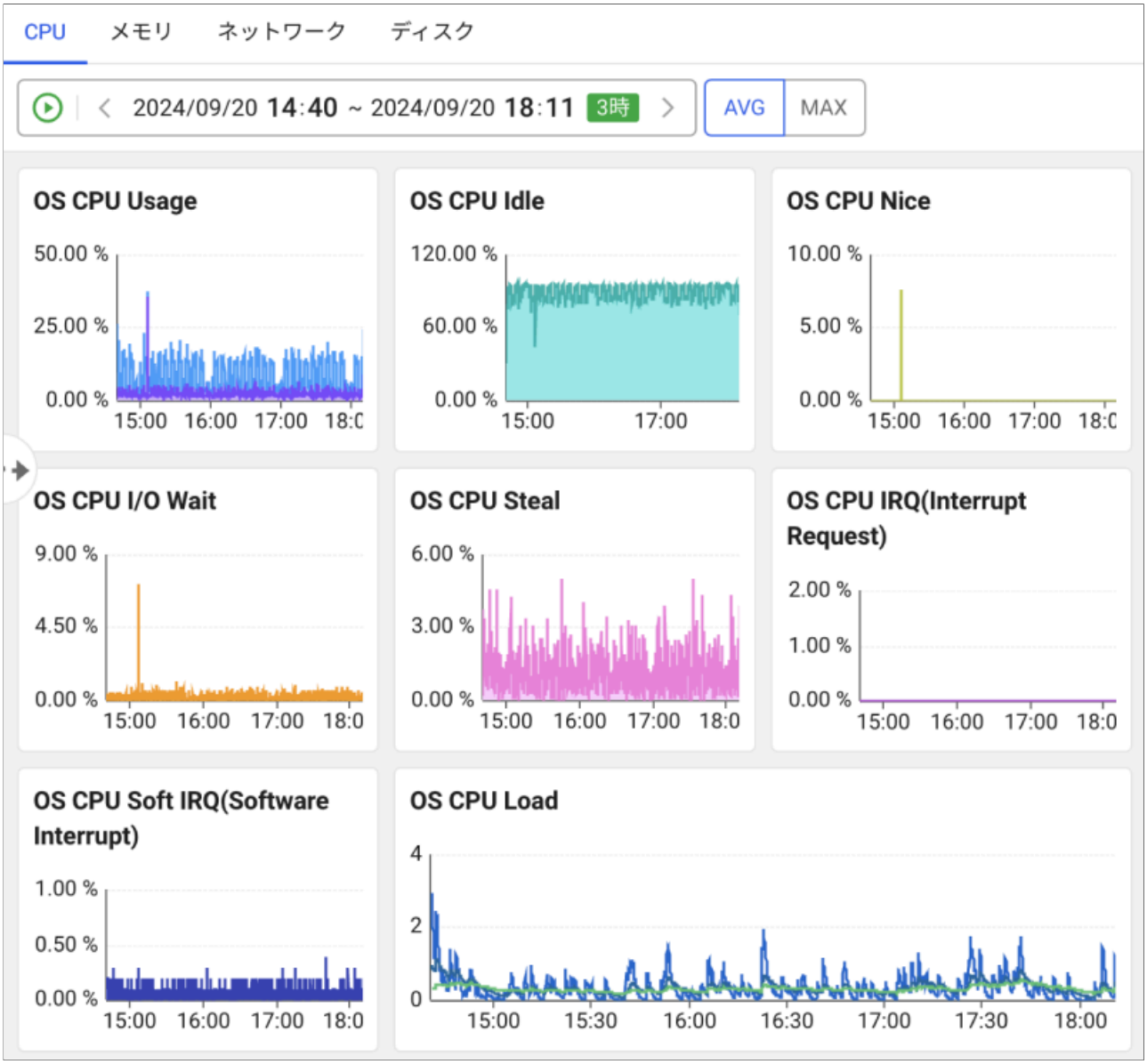
プロセス詳細

- プロセス一覧の左側**詳細**カラムでアイコンを選択すると、そのプロセスに関する詳細情報を確認できます。
- **AVG**または**MAX**ボタンをクリックして、3日以上の長期データ照会時に読み込む統計データの合算方式を選択できます。
- プロセスの主要な情報値をクリックすると、その値がコピーされます。

> **プロセス**タブは、Kubernetesエージェントバージョン1.7.14からサポートします。

メトリクス

メトリクスタブで選択したノードのリソース使用量の推移を確認できます。 照会期間中の平均(**AVG**)、最大値(**MAX**)を基準に**CPU**、**メモリ**、**ネットワーク**、**ディスク**使用量の変化を照会できます。



- **AVG**：照会期間中の平均使用量
- **MAX**：照会期間中の最大使用量
- **Total**：複数のネットワークインターフェイスの全体トラフィック使用量の合計
- **In/Out**：特定のネットワークインタフェースのInbound(受信) およびOutbound(送信) トラフィックの使用量

ⓘ **K8s CPU Request / Allocatable CPU (%)**ウィジェットは、Kubernetesエージェントバージョン1.7.10か

らサポートします。

トレース

トレースタブで選択したノードで実行中のアプリケーショントレース情報を確認できます。



イベント

イベントタブで選択したノードで発生した**K8s Nodeイベント**と**WhaTap Nodeイベント**を確認できます。 **K8s Nodeイベント**はノードの状態変化を含めて、クラスターの重要な作業記録を提供し、**WhaTap Nodeイベント**はユーザーが設定したアラート通知の基準に基づくイベント発生情報を提供します。


K8s Nodeイベント
WhaTap Nodeイベント
2024/09/23 10:04 ~ 2024/09/24 10:04 1日
イベント数の推移
11
5.5
0
09/24
11:00
13:00
15:00
17:00
19:00
21:00
23:00
01:00
03:00
05:00
07:00
09:00
Normal
Warning
タグを隠す
ログメニューへ移動
タイムスタンプ
ログ
2024-09-23 10:34:30.136 NodeReady Node ip-10-21-110-253.ap-northeast-1.compute.internal status is
kind Node name ip-10-21-110-253.ap-northeast-1.compute.internal
2024-09-23 10:34:27.295 Starting
kind Node name ip-10-21-110-253.ap-northeast-1.compute.internal
2024-09-23 10:34:23.842 RegisteredNode Node ip-10-21-110-253.ap-northeast-1.compute.internal ever
kind Node name ip-10-21-110-253.ap-northeast-1.compute.internal
2024-09-23 10:34:21.729 Synced Node synced successfully
kind Node name ip-10-21-110-253.ap-northeast-1.compute.internal
2024-09-23 10:34:21.575 NodeAllocatableEnforced Updated Node Allocatable limit across pods
kind Node name ip-10-21-110-253.ap-northeast-1.compute.internal
2024-09-23 10:34:21.531 NodeHasSufficientPID Node ip-10-21-110-253.ap-northeast-1.compute.intern
kind Node name ip-10-21-110-253.ap-northeast-1.compute.internal

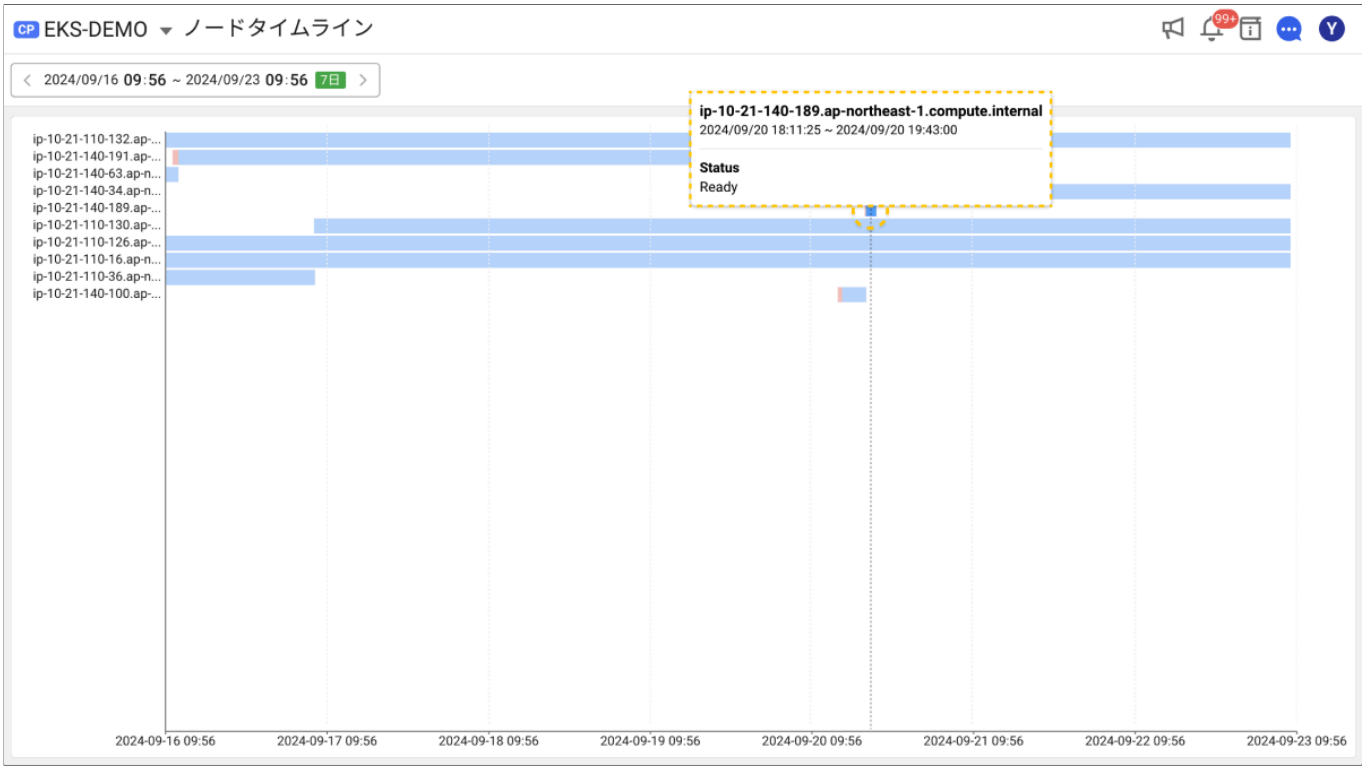
ノードタイムライン

> **要求事項**
>
> - WhaTap Kubernetesエージェント1.7.11バージョン以降が必要です。 タイムラインデータは、1.7.11バージョン以降で収集された時点から提供されます。 つまり、エージェント更新後に収集されたデータから反映されます。
> - Kubernetesクラスタープロジェクト(CP)の場合、該当メニューを照会できます。

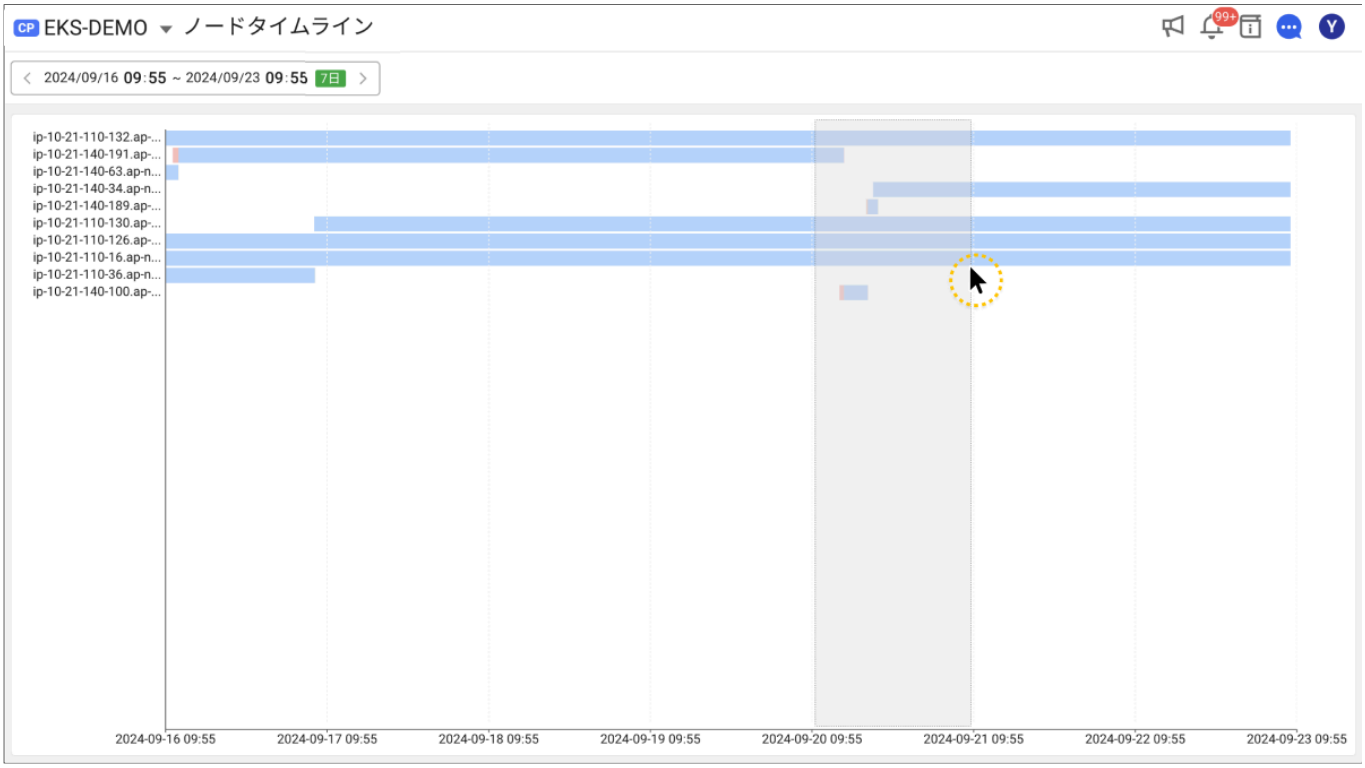
ノードタイムラインメニューは、ノード状態の変化を時間とともに追跡する機能を提供します。 これにより、すでに消えたノードがいつ削除されたのか、その時点でノードがどのような状態だったのかを視覚的に把握できます。 クラスターの運用時に発生したイシューを分析したり、過去の状況を確認するのに役立ちます。

基本画面ガイド

ノードタイムラインメニューに入ると、次のように最近7日間に収集されたノードのリストが表示されます。 時間軸に沿ってノードの状態変化を視覚的に確認できます。 各時間の区間にマウスをかざすと、その区間の開始時刻と終了時刻、そしてノードの状態(**Ready**、**NotReady**など)がツールチップに表示されます。



タイムライン上で区間をドラッグして時間範囲を調整できます。 期間を設定して、過去のデータを確認してみてください。

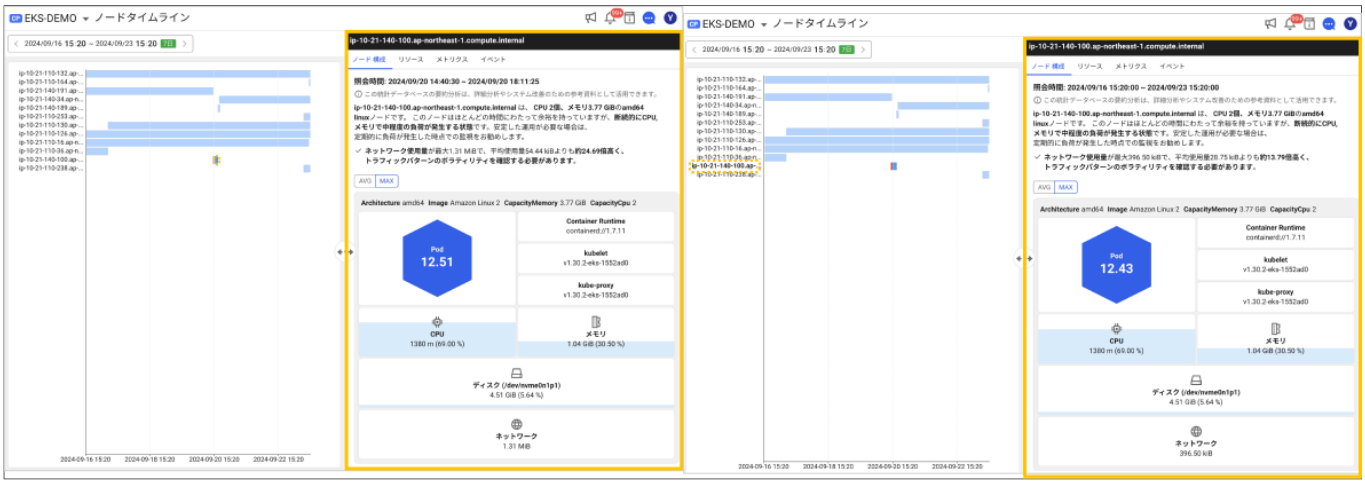


また、特定の期間を選択する時、その期間のノード状態を**詳細画面**で詳しく確認できます。 詳しい内容は、次のノードタイムラインの詳細 案内を参照してください。

> ⓘ • ノード状態についての詳細は、次の文書を参照してください。

ノードタイムラインの詳細

ノードタイムラインの詳細は、次のように特定区間または特定ノードを選択して確認できます。



- 詳細情報を照会したい区間を選択し、右側の**詳細を見る**画面を通じてその時点のノードに対する詳細を照会できます。
- チャート左側のリストから照会するノード名を選択すると、右側の**詳細**画面から照会期間中の該当ノードの全体な状態を確認できます。

ノード 構成

ノード 構成タブで選択したノード構成と全体的なリソース使用量の情報を要約して提供します。 そのノードのハードウェアとソフトウェアの構成要素を総合的に確認できます。 ユーザーはノードのリソース使用状況と各構成要素の状態を把握し、パフォーマンスと運用安定性をモニタリングできます。

照会時間: 2024/09/20 14:40:30 ~ 2024/09/20 18:11:25

ⓘ この統計データベースの要約分析は、詳細分析やシステム改善のための参考資料として活用できます。

ip-10-21-140-100.ap-northeast-1.compute.internal は、 **CPU 2個、メモリ3.77 GiBのamd64 linux**ノードです。 このノードはほとんどの時間にわたって余裕を持っていますが、**断続的にCPU, メモリで中程度の負荷が発生する状態**です。安定した運用が必要な場合は、
定期的に負荷が発生した時点での監視をお勧めします。

✓ **ネットワーク使用量**が最大1.31 MiBで、平均使用量54.44 kiBよりも**約24.69倍高く、トラフィックパターンのボラティリティを確認する必要があります。**

AVG | MAX

Architecture amd64 **Image** Amazon Linux 2 **CapacityMemory** 3.77 GiB **CapacityCpu** 2

Pod
12.51

Container Runtime
containerd://1.7.11

kubelet
v1.30.2-eks-1552ad0

kube-proxy
v1.30.2-eks-1552ad0

CPU
1380 m (69.00 %)

メモリ
1.04 GiB (30.50 %)

ディスク (/dev/nvme0n1p1)
4.51 GiB (5.64 %)

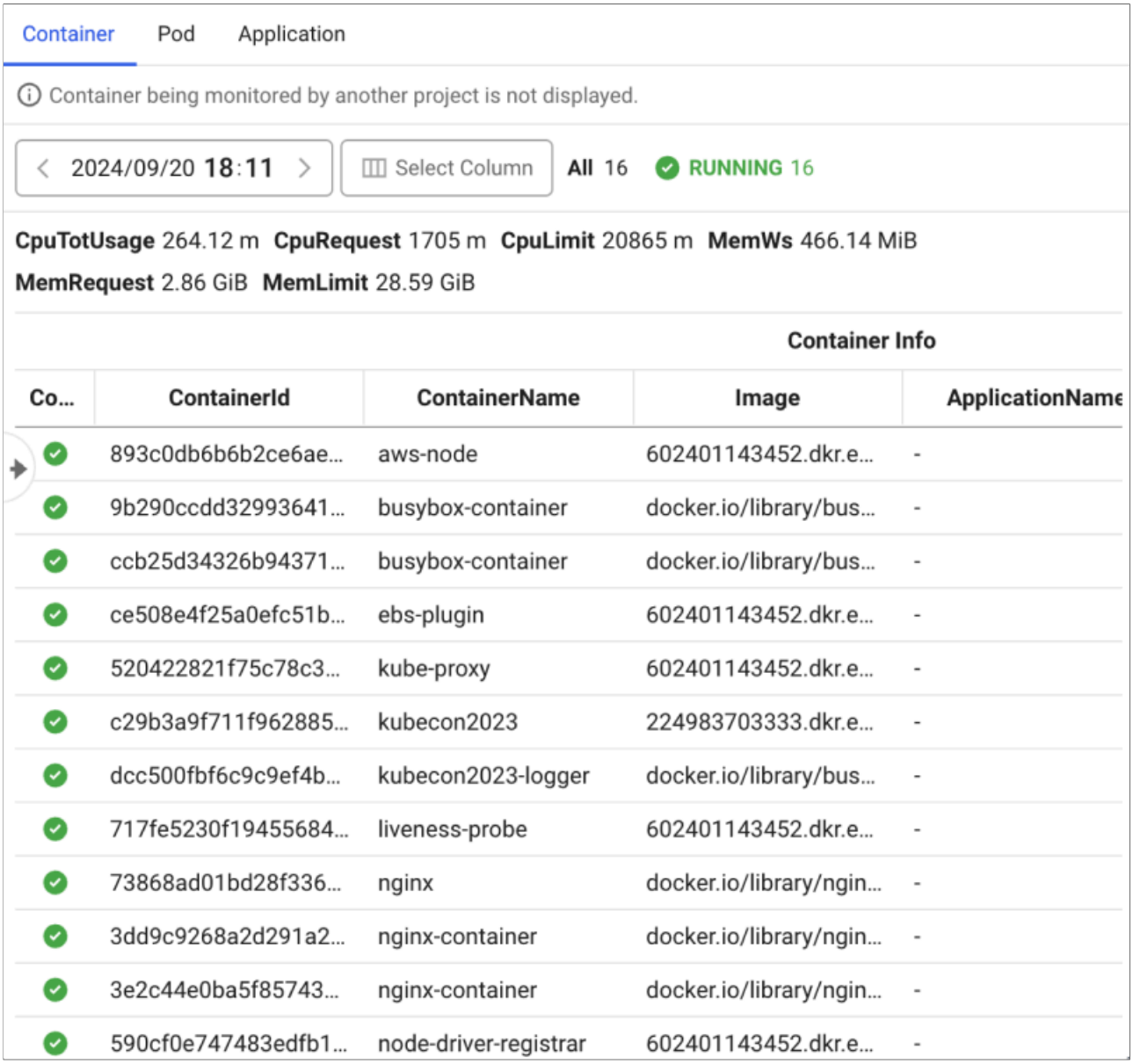
ネットワーク
1.31 MiB

- **ノードシステム仕様**
  - **Architecture**：ノードが使用しているハードウェアアーキテクチャ
  - **Image**：ノードが実行中のオペレーティングシステムのイメージ情報
  - **CapacityMemory**：ノードの総メモリ容量
  - **CapacityCpu**：ノードの総CPU容量
- **ソフトウェア構成**
  - **Container Runtime**：ノードで使用されるコンテナランタイムのバージョン情報
  - **kubelet**：ノードで実行中のクバネティスkubeletのバージョン情報
  - **kube-proxy**：ネットワーク管理の役割を果たすkube-proxyのバージョン情報
- **リソースの使用状況**
  - **CPU**：ノードのCPU使用量
  - **メモリ**：ノードのメモリ使用量
  - **ディスク**：ノードで使用中のディスク容量と使用比率
  - **ネットワーク**：ネットワークの伝送量

> ⓘ **AVG**、**MAX**ボタンをクリックして、照会期間中の平均使用量と最大使用量を確認できます。

リソース

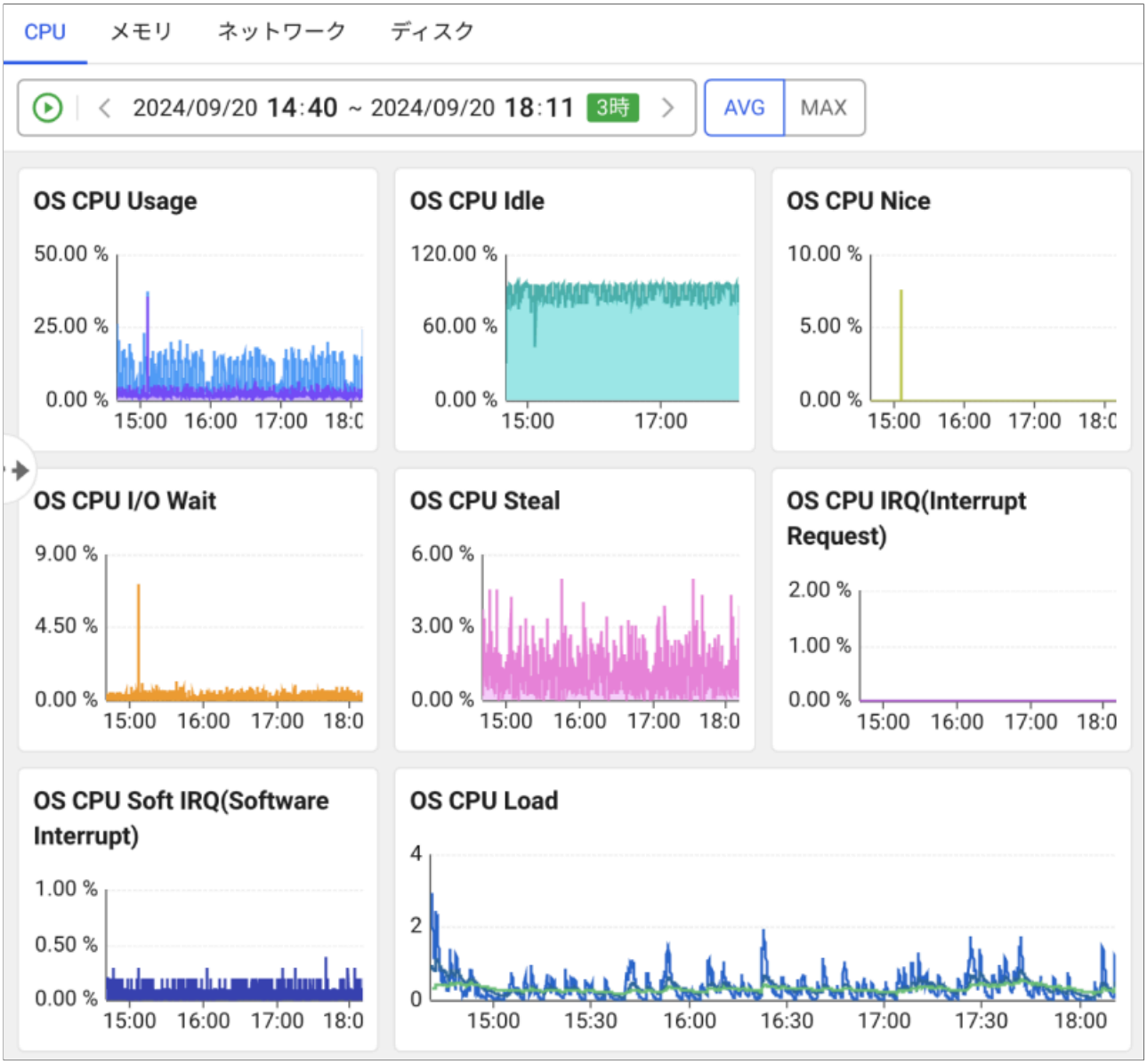
リソースタブで選択したノードにスケジューリングされた**Pod**およびそのPodで実行中の**コンテナ**と**アプリケーション**一覧を確認できます。 その情報は、選択した時間区間の最後の時点を基準に1時間のリソースデータを反映して表示します。 これにより、その時点のノード状態と各Pod、コンテナ、アプリケーションの動作状態をより明確に把握できます。



要約カラムを選択する時、コンテナ及びPod要約分析を確認できます。

メトリクス

メトリクスタブでノードのリソース使用量の推移を提供します。 照会期間中の平均(**AVG**)、最大値(**MAX**)を基準に**CPU**、**メモリ**、**ネットワーク**、**ディスク**使用量の変化を確認できます。



- **AVG**：照会期間中の平均使用量
- **MAX**：照会期間中の最大使用量
- **Total**：複数のネットワークインターフェイスの全体トラフィック使用量の合計
- **In/Out**：特定のネットワークインタフェースのInbound(受信) およびOutbound(送信) トラフィックの使用量

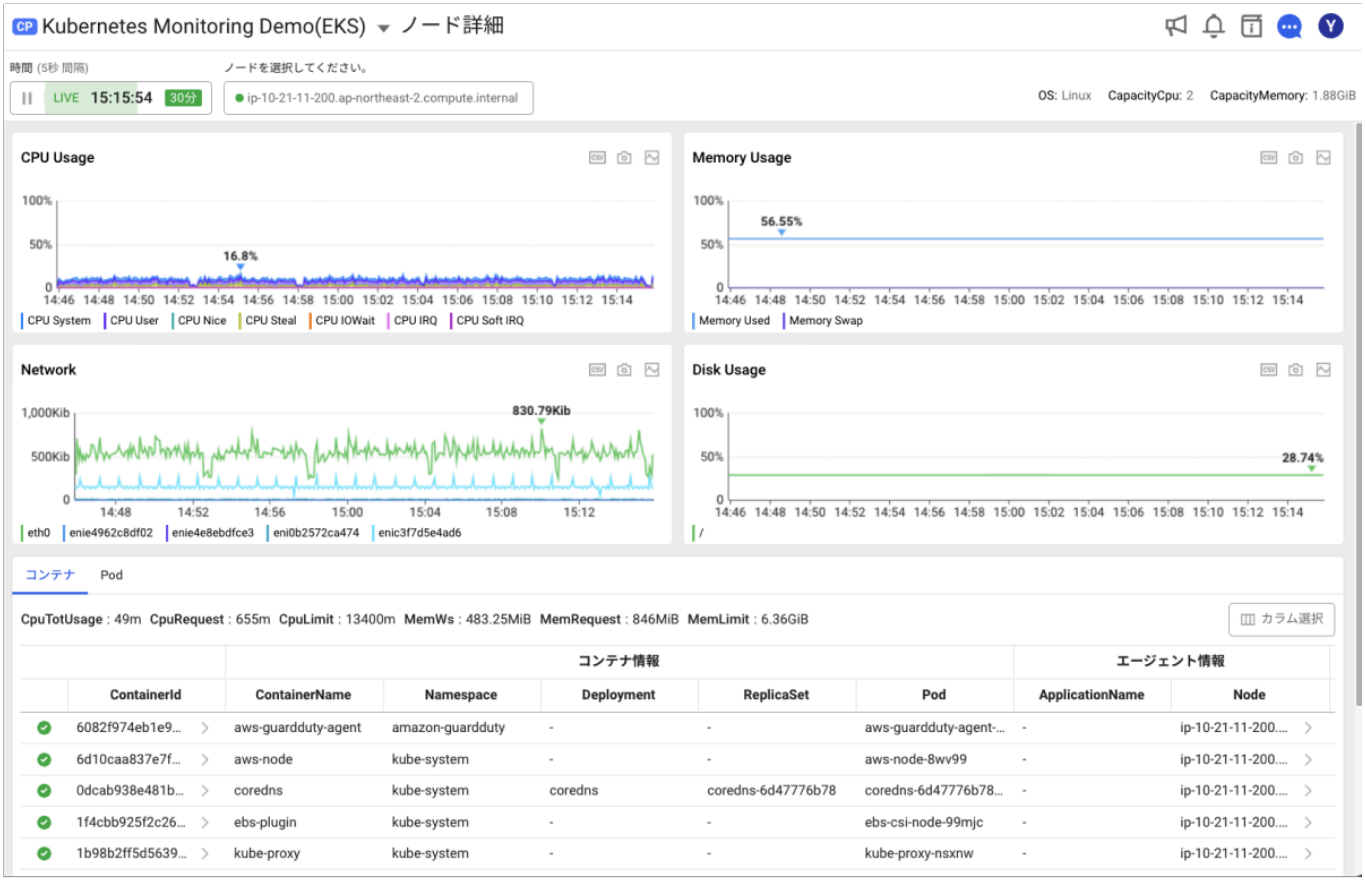
イベント

イベントタブでタイムラインに表示された時間中に発生した**K8s Nodeイベント**と**WhaTap Nodeイベント**を確認できます。 **K8s Nodeイベント**はノードの状態変化を含めて、クラスターの重要な作業記録を提供し、**WhaTap Nodeイベント**はユーザーが設定したアラート通知の基準に基づくイベント発生情報を提供します。



Node詳細情報

ノード詳細情報画面は上段の4つのチャートとノード内の**コンテナ**及び**Pod**詳細情報で構成されています。 ノードセレクタを通じて、照会するノードを選択できます。



> ⓘ 5秒周期で最近の1分のデータを更新します。

トップチャート

次の4つのチャート情報を確認できます。

- **CPU Usage**: ノードのCPU使用量を確認できます。

- **Memory Usage**: ノードのメモリ使用量を確認できます。
- **Network**: ノードのネットワークインターフェイスの入出力トラフィックとエラーパケットを確認できます。
- **Disk Usage**: ノードのDisk使用量を確認することができます。

コンテナ及びPodの詳細情報

コンテナ

コンテナタブの選択時、ノード内のコンテナ詳細情報の上部で、`cpu_quota`、`cpu_request`、`cpu_total_milli`、`mem_limit`、`mem_request`、`mem_working_set`メトリクスの合計を確認できます。

- **コンテナ**タブの選択時、ノード内のコンテナ詳細情報の上段で`cpu_quota`、`cpu_request`、`cpu_total_milli`、`mem_limit`、`mem_request`、`mem_working_set`指標の合計を確認できます。
- **要約**カラムを選択する時**コンテナ要約分析**が確認できます。
- コンテナ選択時に**コンテナー覧**メニューに移動します。 詳細については、次の文書を参照してください。

Pod

Podタブの選択時、Podの詳細情報を照会します。 詳細情報画面の上段から`cpu_quota`、`cpu_request`、`cpu_total_milli`、`mem_limit`、`mem_request`、`mem_working_set`メトリクスの合計を確認できます。

- 希望するカラムを選択すると、そのカラムを基準にPod一覧をソートして照会することができます。
- **要約**カラムを選択する時**Pod要約分析**が確認できます。
- Pod選択時に**Pod一覧**メニューに移動します。 詳細については、次の文書を参照してください。

ノードディスク一覧

> **要求事項**
>
> - WhaTap Kubernetesエージェント1.7.8バージョン以降が必要です。
> - Kubernetesクラスタープロジェクト(CP)の場合、該当メニューを照会できます。

ノードディスクリストを照会します。 WhaTap Kubernetesモニタリングの**ノードディスクリスト**で各ノードにマウントされた個別ディスクの現在の使用量、余裕空間、性能指標をリアルタイムでモニタリングし、ディスクの状態を簡単に確認できます。 これにより、クラスターのディスク リソースを効率的に管理し、潜在的な問題を事前に把握して対応できます。

基本画面ガイド

現在クラスターに存在するノードとそのノードにマウントされたディスクの一覧を次のように確認できます。 照会時間を基準に、最近1分間収集されたディスク情報に基づいて一覧を表示します。



比較	OnodeName	DeviceId	FileSystem	BlkSize	TotalSpace	UsedSpace	UsedPercent	FreeSpace
	ip-10-21-11-144....	/dev/nvme0n1p1	xfs	4.00 kiB	19.99 GiB	9.30 GiB	46.50 %	10.69 GiB
	ip-10-21-11-188....	/dev/nvme0n1p1	xfs	4.00 kiB	19.99 GiB	5.20 GiB	26.03 %	14.79 GiB
	ip-10-21-11-98.a...	/dev/nvme0n1p1	xfs	4.00 kiB	19.99 GiB	4.75 GiB	23.76 %	15.24 GiB
	ip-10-21-13-239....	/dev/nvme0n1p1	xfs	4.00 kiB	19.99 GiB	4.64 GiB	23.21 %	15.35 GiB
	ip-10-21-13-36.a...	/dev/nvme0n1p1	xfs	4.00 kiB	19.99 GiB	8.26 GiB	41.32 %	11.73 GiB

未表示ノードディスク

WhaTap Kubernetesエージェントが起動すると、各ノードにマウントされたディスクの一覧が表示されます。 次のような場合、一覧には表示されません。

- ディスクが特定のパスにマウントされている場合に限って一覧に表示されます。 **マウントポイントが存在しないディスク**は、一覧に表示されません。

- WhaTap KubWhaTap エージェント**起動後に追加されたディスク**は、一覧に表示されません。

フィルター

フィルターオプションを活用する時、希望するを選択して照会できます。 `and` 条件を使用すると、複数のフィルターが適用できます。

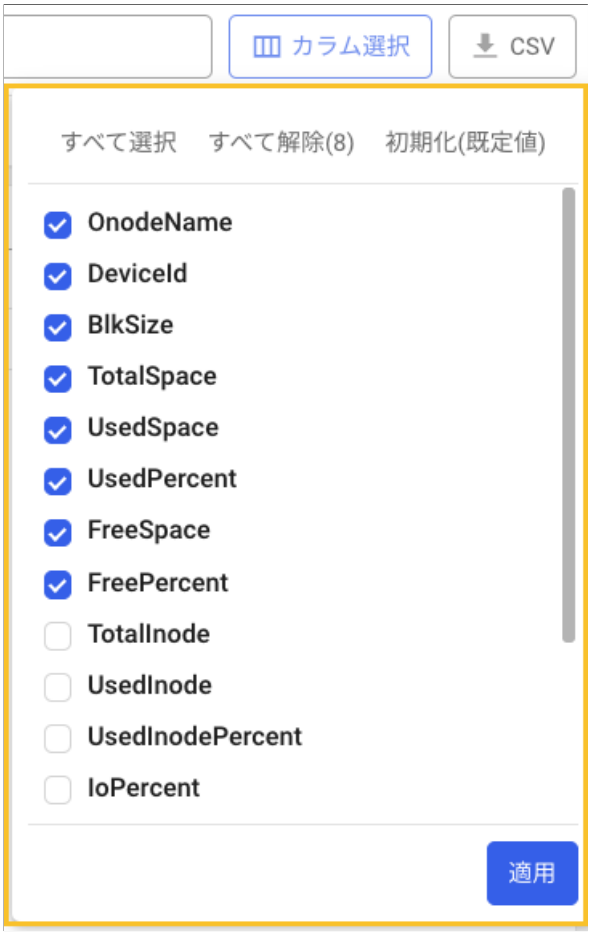
1. フィルター入力ウインドウで＋追加アイコンを選択すると、次のようにフィルターを追加するウィンドウが表示されます。



   - フィルターキーと条件を選択します。
   - 条件に合った値を選択します。
2. フィルター設定後、適用ボタンをクリックし、フィルターを適用します。
3. 複数のフィルターを追加するには、フィルター入力ウインドウまたはフィルターを追加するウィンドウで＋追加アイコンを選択します。

カラム選択

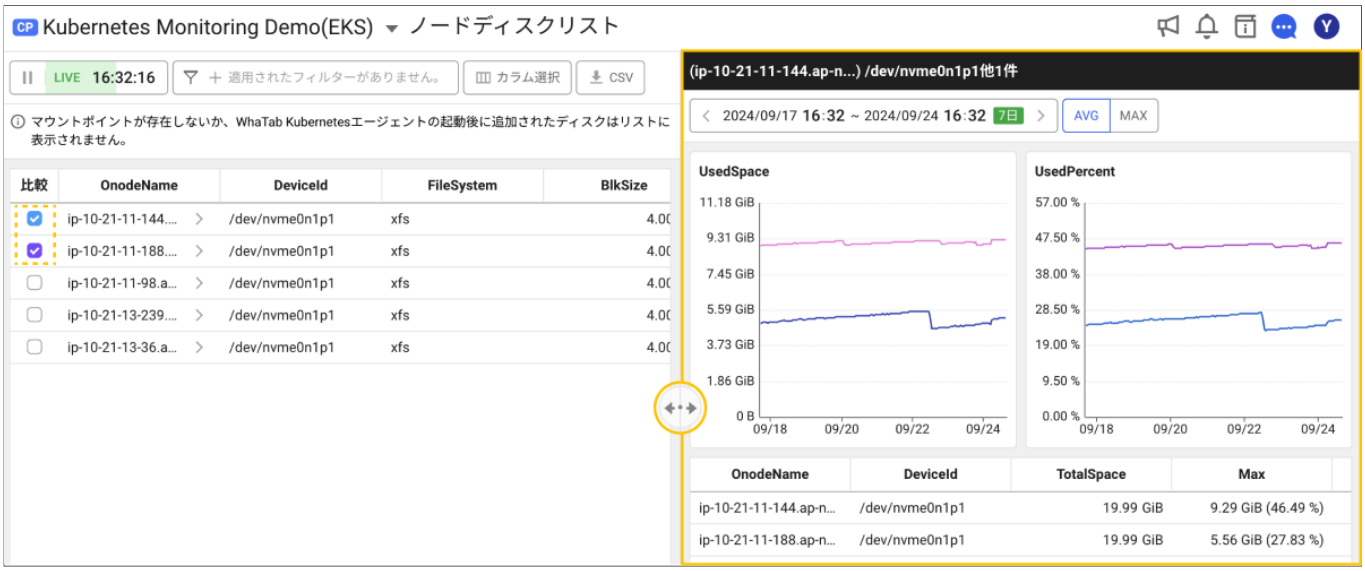
カラム選択オプションを活用する時、希望する情報を選択して一覧を照会できます。



- 選択したカラム一覧は、メニューを再入力すると維持されます。
- リセットボタンを選択すると、定義済みの既定の主要指標カラムが選択されます。 この場合、既存のカラム選択は削除されます。
- カラムを選択した後、右下の適用ボタンをクリックして選択履歴を適用することができます。

比較する

比較オプションにより、ディスク間の使用量を比較できます。 一覧から照会するノードのディスク**比較**カラムのチェックボックスを選択すると、右側に該当ディスクの詳細を比較する画面が表示されます。



例えば、**UsedSpace**および**UsedPercent**グラフを使用すると、各ディスクの時間別使用量と使用率を確認できます。これにより、特定の時間区間でのディスク使用量の変化を直観的に確認できます。 また、下の詳細データ一覧では、各ディスクの最大使用量と最小使用量、増減率などの詳細なデータが表示されます。 カラム情報は次のとおりです。

- **OnodeName**: ディスクが接続されているノードの名前です。
- **DeviceId**: ディスクのデバイスIDを意味し、ディスクの具体的なデバイスを区分できます。
- **Max**: 照会期間中に記録されたディスクの最大使用量と最大使用率を示します。
- **Min**: 照会期間中に記録されたディスクの最小使用量と最小使用率を示します。
- **First / Last**: 照会期間中に最初に収集されたデータと最後に収集されたデータで使用量の変化が分かります。
- **Increase**: 選択された期間中に該当ディスクの使用量の増減率を示します。
- **Increase(Per Day)**: 照会期間中の1日平均の使用量増減率を計算した値で、ディスク使用量の変動傾向を確認できます。 照会期間が1日未満の場合、この値は表示されません。
- **PredictedSpace(N days later)**: 1日平均増減率を基準にN日後の予想使用量を予測する値です。 照会期間が1日未満の場合、この値は表示されません。

  例、照会期間が7日の場合、7日後の予想使用量を表示します。

> - **AVG**、**MAX**ボタンをクリックして、該当区間の平均使用量と最大使用量を確認できます。
> - 左右にドラッグして、画面のサイズを調整できます。

WhaTap GPU モニタリング

WhaTap GPU モニタリングは、サーバーおよびKubernetes環境全体にわたって、GPUリソースの状態と活用状況を統合的に分析できるよう支援します。GPUはサーバーを構成するリソースの一つですが、CPUやメモリと比べてコストが高く、AI/ML、LLM、HPCワークロードのパフォーマンスに直接影響を与える重要なリソースです。

単に**GPUが搭載されていて、正常に動作しているか**を確認するだけでなく、現在どの程度活用されているか、どのタスクが占有しているか、異常な兆候はないか、リソース配置は適切かまでを確認できる必要があります。

WhaTap GPU モニタリング範囲

> **WhaTap GPU モニタリング対応環境**
>
> - Server環境でのGPUモニタリング
> - Kubernetes環境でのGPUモニタリング

サーバー環境

- GPU サーバーモニタリングは、GPU使用率、メモリ使用量、温度、電力、クロック、エラー、PCIe/NVLink通信状態などの主要指標を定期的に収集・保存し、サーバーインフラの観点からGPUデバイスの状態を総合的に把握できるよう支援します。
- GPUを使用しているプロセスの占有状況も合わせて提供するため、どのタスクがどのGPUをどの程度使用しているかを把握できます。これにより、単純なデバイス状態の確認を超えて、サーバー内のGPUリソースの実際の活用方法まで分析できます。
- モニタリングにより、過熱、スロットリング、電力制限、異常エラー、PCIe/NVLink通信異常などの兆候を早期に検出でき、障害予防と迅速な対応体制を整えることができます。これはサービスの安定性確保と運用効率の向上に直接貢献します。

Kubernetes環境

- Kubernetes環境では、GPUは主にAI学習/推論ワークロード、バッチジョブ、高性能演算アプリケーションに割り当てて使用されます。この場合、ノード単位のGPU状態のみを確認するだけでは、実際の運用状況を十分に把握す

ることが難しい場合があります。

- WhaTapのKubernetes GPUモニタリングは、クラスター、ノード、Pod/コンテナ単位でGPUリソースの使用状況を確認できるよう支援します。これにより、どのワークロードがGPUを占有しているか、特定のノードに使用が偏っていないか、GPUリクエスト/割り当てに対して実際の使用効率はどうかなどを把握できます。
- Kubernetesの観点からのGPUモニタリングは、スケジューリングの適切性、リソースの偏り、過少/過大割り当ての有無を分析するうえでも有用です。結果的に、GPUベースのワークロード運用の安定性を高め、クラスター全体のリソース活用効率を改善できます。

エンタープライズ環境

- エンタープライズ環境では、多数のGPUサーバーとKubernetesクラスターが共同で運用されます。単一のサーバーや個別のKubernetes環境単位のGPUモニタリングだけでは、リソース全体の状態と活用状況を一目で把握することが難しいため、統合的な管理体制が必要です。
- 数百～数千枚規模のGPUを運用する企業では、チームや業務ごとに割り当てられたGPUリソースがどれほど効果的に活用されているかを継続的に確認したいと考えています。このようなデータがあってこそ、遊休リソースの再配置、新規リソースの増設、追加購入の要否などに関する合理的な意思決定に役立てることができます。
- 複数のKubernetesベースのGPU環境とベアメタルGPUサーバー環境を単一のモニタリング体制に統合することで、分散したGPUリソースを一つの視点で管理できます。これにより、組織全体のGPUリソース状況を統合的に把握し、Capacity Planningにまで連携できるエンタープライズ型GPUモニタリングを提供します。
- チームやプロジェクト単位でKubernetesポッドラベルをGPUメトリクスに連携して使用量をグルーピングする方法については、GPUサービスグルーピングを参照してください。

WhaTap GPU モニタリングの提供価値

GPUリソースの可視化

サーバー、クラスター、ノード、Pod、プロセスレベルまでGPUの使用状況を多角的に把握できます。

障害の早期検知と迅速な対応

温度上昇、電力制限、クロック低下、エラー発生、通信異常などの兆候を素早く特定し、障害に発展する前に対応できます。

GPU活用率の最適化

遊休GPU、偏った使用、過少活用リソースを確認し、再配置および運用の最適化が可能です。

運用効率の向上

サーバーとKubernetes環境を分離して管理するのではなく、統合的に管理することで運用の複雑さを軽減できます。

合理的な増設・投資判断の支援

実際の使用データをもとにGPU増設の必要性を検討し、予算・購入計画の根拠資料として活用できます。

Capacity Planning支援

短期的な障害対応だけでなく、中長期的なリソース需要予測と運用戦略の策定にまで連携できます。

GPUデバイス異常検知方法

GPUデバイスおよびワークロードで発生し得る異常の兆候を検知する方法です。

1. Xidベースの異常検知

NVIDIA GPUは、障害または異常状態が発生した際にXidイベントをログに記録します。そのため、システムログやドライバーログでXidキーワードおよびコードの発生有無を検知することで、GPUデバイスの異常を比較的迅速に特定できます。

例えば、特定のXidはGPU演算エラー、メモリアクセス問題、ドライバー/ハードウェア異常、リセット発生などの兆候を示す場合があるため、運用環境ではXidイベントを重要な障害シグナルとして活用できます。

2. 状態指標ベースの異常検知

GPU異常は、必ずしもエラーログだけに現れるとは限りません。実際の運用環境では、GPU使用率、メモリ使用量、温度、電力、クロック、PCIe/NVLink通信状態といった主要な状態指標の変化パターンを合わせてモニタリングすることで、異常の兆候をより早期に検知できます。

ただし、これらの指標は**絶対値だけで判断するのではなく、業務特性、タスクの種類、時間帯別パターン、平常時のベースラインを合わせて考慮して解釈することが重要**です。例えば、学習、推論、バッチ処理、データ前処理といったワークロードは、それぞれ正常動作時のGPU使用パターンが異なるため、同じ数値でも業務の文脈によって意味が変わる場合があります。

この観点から、以下のようなパターンの変化は主要な点検対象となります。

- 通常の業務パターンと比較してGPU使用率が過度に高い、あるいは逆に割り当てられたリソースが十分に活用されていない状態が続いている場合
- タスク終了後もメモリ占有が異常に維持されていたり、特定の時点からメモリ使用量が急激に増加している場合
- 温度が継続的に上昇して通常の運用水準を超えたり、長時間にわたって高温状態が維持されている場合
- 電力使用量が制限値付近に長時間とどまり、電力制限による性能制約が疑われる場合
- 負荷水準に対してクロックが期待値より低く維持され、スロットリングまたは異常な制御状態が疑われる場合
- PCIe/NVLink通信量が業務特性に比べて異常に増減したり、InfiniBandにエラー指標が同時に観測される場合

状態指標を個別の数値ではなく、業務の文脈における変化パターンとして解釈することで、単純な使用状況の把握を超えて、パフォーマンス低下、過熱、スロットリング、電力制限、通信ボトルネック、異常動作といった問題をより正確に特定できます。

3. プロセス/ワークロード観点の異常検知

GPUを占有しているプロセス、Pod、Containerなどのワークロード情報を合わせて確認することで、デバイス指標だけでは把握しにくい異常状況をより明確に解釈できます。ただし、これもまた単純な占有有無だけで判断するのではなく、業務の種類、実行段階、繰り返し周期、平常時の占有パターンを合わせて考慮する必要があります。

学習、推論、バッチ処理、データ前処理といったワークロードはGPUの使用方法がそれぞれ異なるため、同じ占有状態であっても、どのような業務かによって正常・異常の基準が変わる場合があります。

この観点から、以下のようなパターンは主要な点検対象となります。

- タスクの特性に比べて特定のプロセスやワークロードがGPUメモリを過度に占有したまま、予想より長い時間解放しない場合
- GPU使用率は低いのにメモリ占有のみが長時間維持され、タスクの停滞・異常待機・メモリリークの可能性が疑われる場合
- 特定のタスクが繰り返し再起動または失敗し、GPUリソースを非効率に占有している場合
- 一部のGPUにのみ特定のワークロードが継続的に集中し、使用の不均衡や局所的な過熱が発生している場合

- 同種の業務と比較して、特定のプロセスのみが過度に長い実行時間、高いメモリ占有、異常な占有パターンを示している場合
- PodまたはContainerの再配置後にGPU占有パターンが急激に変化し、スケジューリングの非効率やリソースの偏りが疑われる場合

これらのパターンは、単純なデバイス異常だけでなく、アプリケーションまたはワークロードの運用上の問題を合わせて特定するのに役立ちます。

GPUダッシュボード

GPUリソースの割り当てから実際の活用までの流れと、デバイスごとの状態をマップで追跡できるKubernetes視点の**GPUダッシュボード**です。

- TotalからEffectiveまで段階別のリソースの流れをファネルで提供し、割り当てに対する実際の活用水準を一目で把握できます。
- デバイス単位のGPU Mapにより、クラスター・ノード・Namespace基準のリソース分布と状態を視覚的に確認できます。
- 無駄なリソースのみをフィルタリングするトグルを提供し、割り当てられているものの活用されていないGPUを素早く識別できます。

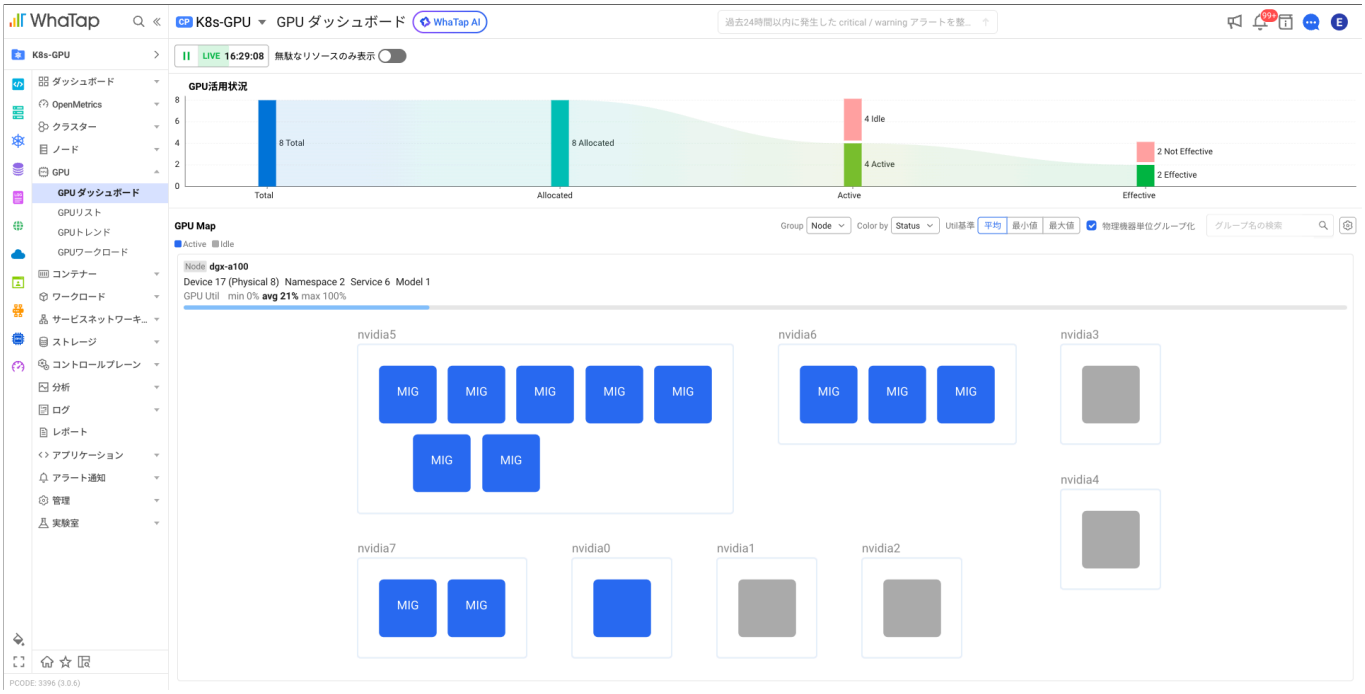
> GPUメニューは**GPUダッシュボード**、**GPU一覧**、**GPUトレンド**、**GPUワークロード**ページで構成されます。本ドキュメントは、リソースの活用状況とデバイスマップを提供する**GPUダッシュボード**ページを扱います。

権限および条件

- **対応環境**：Kubernetesクラスタプロジェクト
- **エージェントバージョン**：Kubernetesエージェント バージョン1.8.7以上
- Open Agentのインストールが必要

基本画面

クラスター内のGPUリソースの活用状況とデバイスごとの状態を一目で把握できる可視化ダッシュボードです。



画面上部にはLIVE時刻と**無駄なリソースのみ表示**トグルがあります。トグルをオンにすると、GPU Mapに無駄なリソースとして分類されたデバイスのみが表示されます。

- **無駄なリソースのみ表示**トグルはGPU Mapの表示対象をフィルタリングするものであり、**GPU活用状況**ファネルの集計値には影響しません。
- 画面上部の時間選択で一時停止ボタンを選択すると、過去の時間のデータを照会できます。

GPU活用状況

GPUリソースが割り当てられてから実際に活用されるまでの流れをファネルチャートで提供します。段階は**Total**、**Allocated**、**Active**、**Effective**の順に絞り込まれ、各段階で活用されなかったリソースは**Idle**、**Not Effective**に分離して表示されます。

Stage	Description
Total	全物理デバイス

Stage	Description
Allocated	プロセスまたはPodが割り当てられたデバイス
Active	5分平均GPU使用率（Utilization）が1%以上のデバイス
Idle	5分平均GPU使用率が1%未満のデバイス
Effective	GPU使用率が50%を超えるデバイス
Not Effective	GPU使用率が50%以下のデバイス

Allocatedの判定基準が、プロセスの実行有無ではなくGPUメトリックのPodラベルの存在有無に変更されました。GPUが割り当てられたPodがCrashLoopなどでプロセスを起動できなかった場合でも、割り当てられたデバイスとして正確に集計されます。これにより、上記の表のAllocated説明は「GPUが割り当てられたPodが存在するデバイス」と理解できます。

上記の画面例では、Total 8個のうち8個がAllocated、そのうちActive 4個・Idle 4個に分かれ、Activeのうちでも Effective 2個・Not Effective 2個に分類されます。

IdleとNot Effectiveは割り当てられているものの十分に活用されていないデバイスであり、リソースの無駄を点検する基準となります。

GPU Map

照会時点で収集されたGPUデバイスをマップで表示します。ノード単位のカード内に各デバイスがボックスで配置され、カード上部で当該ノードのデバイス・Namespace・Service・モデル数とGPU使用率の要約を確認できます。

コントロール

Control	Option	Description
Group	Cluster / Node / Namespace / Service / Model	デバイスをまとめる基準

Control	Option	Description
Color by	Status / Utilization / VRAM(%) / Temp	デバイスボックスの色の基準指標
Util基準	平均 / 最小値 / 最大値	使用率表示に使用する統計値
物理デバイス単位グループ化	オン / オフ	物理デバイス単位でデバイスをまとめて表示
グループ名検索	テキスト入力	グループ名で表示対象をフィルタリング

ノードカードとデバイスボックス

各ノードカードは、当該ノードに搭載されたデバイスをボックスで表示します。カード上部の要約には次の情報が表示されます。

- **Device n (Physical n)**：デバイス数と物理デバイス数
- **Namespace n / Service n / Model n**：ノードに関連するNamespace・Service・モデル数
- **GPU Util (min / avg / max)**：ノードGPU使用率の最小値・平均・最大値

デバイスボックスは状態に応じて異なる表示になります。

Display	Meaning
MIG表記ボックス	MIGインスタンスに分割されたデバイス
色付きボックス	Activeデバイス、Color by基準に応じて色が決定
グレーボックス	Idleデバイス（5分平均GPU使用率1%未満）

凡例では**Active**と**Idle**を色で区別します。

デバイスボックスにマウスを乗せると、デバイス種類（Physical/MIG）、GPU Utilization、Status、Device、Pod、Node、Namespace、Service情報を確認できます。

> ⓘ • **Color by**オプションを変更すると、同じデバイスでも基準指標（状態・使用率・VRAM・温度）に応じて

色が変わります。

- MIGに分割されたデバイスは物理デバイス1個が複数のインスタンスとして表示されるため、デバイス数と物理デバイス数が異なる場合があります。

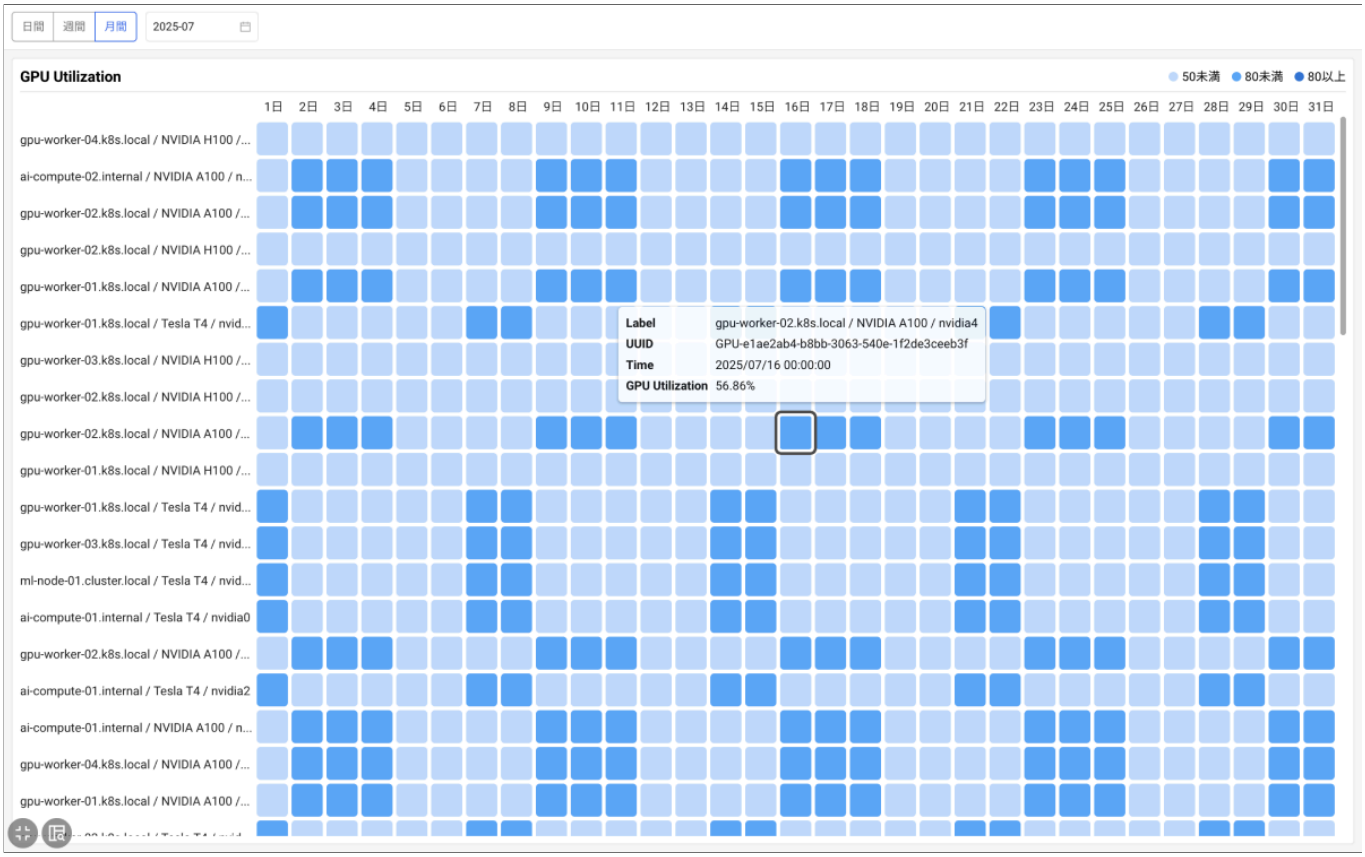
GPUトレンド

GPUトレンドは、GPUデバイスごとの使用量を長期統計データとして可視化し、特定の時点でどのデバイスが非効率的に使用されていたかを迅速に特定できます。

基本画面

GPUの長期使用量統計およびパターンを確認できます。
照会単位は **日次、週次、月次** をサポートします。
デバイスごとのGPU使用率はブロック形式で表示され、色の濃さによって使用レベルを区分します。

- **50%未満**: 薄い青
- **80%未満**: 青
- **80%以上**: 濃い青



詳細情報の確認

各ブロックにマウスを載せると、その時点およびデバイスの詳細情報をツールチップで確認できます。

- **Label**: デバイス名およびGPUモデル
- **UUID**: GPU固有識別子
- **Time**: 基準時刻
- **GPU Utilization**: 当該時点のGPU使用率(%)

GPUサービスグルーピング

> GPUモニタリング全般の概要については、WhaTap GPUモニタリングのドキュメントを参照してください。

WhaTap OperatorのGPUモニタリング機能では、**アプリケーション（GPU使用）ポッドのKubernetesラベルをGPUメトリクスに付与**し、WhaTapでサービス/プロジェクト単位にGPU使用量をグルーピング・照会できます。

- 適用対象: `WhatapAgent` CR (`monitoring.whatap.com/v2alpha1`)
- 関連フィールド: `spec.features.k8sAgent.gpuMonitoring.groupLabel` 、 `groupLabelAllowlistRegex`
- キー結果ラベル: `whatap_kube_label_gpu_group`

> **どのような問題を解決しますか？**
>
> デフォルトのDCGMメトリクス（ `DCGM_FI_DEV_GPU_UTIL` など）にはGPU/ノード/ポッドの識別子はありますが、**ユーザーが運用する「サービス」単位の情報**（例: プロジェクトID、チーム、ワークロード種別）は含まれていません。
>
> その結果:
>
> - 「プロジェクトAが使用する全GPU使用率の合計」のようなクエリが困難です。
> - マルチテナント（複数チームが1クラスターを共有）環境でのコスト/使用量の精算が困難です。
>
> この機能を使用すると、ポッドに付けたラベル（ `prjId=projectA` など）が**DCGMメトリクスラベルに自動伝播**され、WhaTapの共通ラベル `whatap_kube_label_gpu_group` に正規化されて、ダッシュボード/検索でグルーピングキーとしてすぐに使用できます。

クイックスタート

GPUモニタリング + グループラベルの有効化

```
apiVersion: monitoring.whatap.com/v2alpha1
kind: WhatapAgent
```

```
metadata:
  name: whatap
  namespace: whatap-monitoring
spec:
  features:
    k8sAgent:
      gpuMonitoring:
        enabled: true
        # グルーピングキーとして使用するポッドラベルキー
        groupLabel: "prjId"
        # （任意）公開を許可するラベルキーの正規表現。未設定時は ^(prjId)$ が自動適用
        # groupLabelAllowlistRegex: "^(prjId|team)$"
        # （任意）全GPUメトリクスにclusterラベルを付与
        clusterName: "prod-gpu-cluster"
```

GPU使用ポッドへのラベル付与

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ml-train
  namespace: ml
spec:
  replicas: 2
  selector:
    matchLabels:
      app: ml-train
  template:
    metadata:
      labels:
        app: ml-train
        prjId: "projectA"      # ◀ グルーピングキー（groupLabelと同じ）
    spec:
      containers:
        - name: trainer
          image: my-registry/trainer:latest
          resources:
            limits:
              nvidia.com/gpu: 1
```

> ⓘ ラベルはDeploymentラベルではなく、**Podレベル**（`spec.template.metadata.labels`）に付与する必要があります。

使用例（活用シナリオ）

プロジェクト(prjId)単位の使用量

- `groupLabel: prjId`
- ポッドラベル: `prjId: projectA`、`prjId: projectB`
- WhaTapダッシュボードで`whatap_kube_label_gpu_group`でgroup by → プロジェクト別GPU使用率/メモリ使用量の比較

チーム(team)単位の精算

```
gpuMonitoring:
  enabled: true
  groupLabel: "team"
  groupLabelAllowlistRegex: "^(team)$"
```

- ポッドラベル: `team: search`、`team: recsys`、…

複数のラベルを同時に公開したい場合

`groupLabel`で正規化されるラベルは1つ（`whatap_kube_label_gpu_group`）ですが、他のポッドラベルもメトリクスラベルとして同時に公開したい場合は`groupLabelAllowlistRegex`で拡張できます。

```
gpuMonitoring:
  enabled: true
  groupLabel: "prjId"
  groupLabelAllowlistRegex: "^(prjId|team|env)$"
```

- `whatap_kube_label_gpu_group` ← `prjId`の値

- `team`、`env`ラベルもDCGMメトリクスにそのまま公開されます（正規化はされません）

検証方法

dcgm-exporter環境変数の確認

```
kubectl -n whatap-monitoring get ds whatap-node-agent \
 -o jsonpath='{.spec.template.spec.containers[?(@.name=="dcgm-exporter")].env}' | jq
```

出力結果に以下の2つの環境変数が含まれていることで、ポッドラベル収集が有効な状態です。

- `DCGM_EXPORTER_KUBERNETES_ENABLE_POD_LABELS=true`
- `DCGM_EXPORTER_KUBERNETES_POD_LABEL_ALLOWLIST_REGEX=^(prjId)$`

メトリクスの直接照会（ポッドにラベルが必要）

```
# dcgm-exporter Podに接続
POD=$(kubectl -n whatap-monitoring get pod -l name=whatap-node-agent -o name | head -1)
kubectl -n whatap-monitoring exec "$POD" -c dcgm-exporter -- \
 wget -qO- localhost:9400/metrics | grep DCGM_FI_DEV_GPU_UTIL | head
```

```
DCGM_FI_DEV_GPU_UTIL{gpu="0", ..., pod="ml-train-xxxx", namespace="ml", prjId="projectA"} 87
```

OpenAgent scrape設定の確認

```
kubectl -n whatap-monitoring get configmap whatap-open-agent-config -o yaml | grep -A2 whatap_kube_label_gpu_group
```

出力に以下のrelabelブロックが含まれている必要があります。`source_labels`の値は設定した`groupLabel`キーと一致している必要があります。

```
- action: replace
  source_labels: [prjId]
  target_label: whatap_kube_label_gpu_group
```

トラブルシューティング

症状	原因 / 確認ポイント
メトリクスに`prjId`ラベルが表示されない	(1) Podラベルに`prjId`が付与されているか（`spec.template.metadata.labels`）、(2) `dcgm-exporter`が再起動されて新しいenvが反映されているか
`prjId`は見えるが`whatap_kube_label_gpu_group`が見えない	OpenAgent ConfigMap更新後、OpenAgentポッドの再起動が必要。`kubectl rollout restart deploy/whatap-open-agent -n whatap-monitoring`
`groupLabel`の値に`-`、`.`、`/`が含まれる	Prometheusラベルキールール上、`_`に置換されたキーでsource_labelsが構成されます。allowlist正規表現は元のキー基準で設定可能
複数のラベルを公開したい	`groupLabelAllowlistRegex`を`^(prjId|team|env)$`のように明示
全GPUメトリクスにクラスター識別が必要	`clusterName`フィールドを併せて設定（メトリクスに`cluster=<値>`を付与）

関連フィールド一覧

フィールド	説明	デフォルト
`gpuMonitoring.enabled`	GPUモニタリング（dcgm-exporter）の有効化	`false`
`gpuMonitoring.groupLabel`	グルーピングに使用するPodラベルキー。設定時に`whatap_kube_label_gpu_group`に正規化	-
`gpuMonitoring.groupLabelAllowlistRegex`	dcgm-exporterが公開するPodラベルキーの正規表現	`^(<groupLabel>)$`
`gpuMonitoring.clusterName`	全GPUメトリクスに`cluster`ラベルを付与	-

参考

- git repo: https://github.com/whatap/whatap-operator.git
- 実装コード: `internal/controller/install_agents.go`（`addDcgmExporterToNodeAgent`、GPU scrape config生成部）
- CRD定義: `api/v2alpha1/whatapagent_types.go`（`GpuMonitoringSpec`）
- 例: `examples/whatap-agent-all-options.yaml`、`examples/whatap-agent-gpu-custom-env.yaml`

コンテナリスト

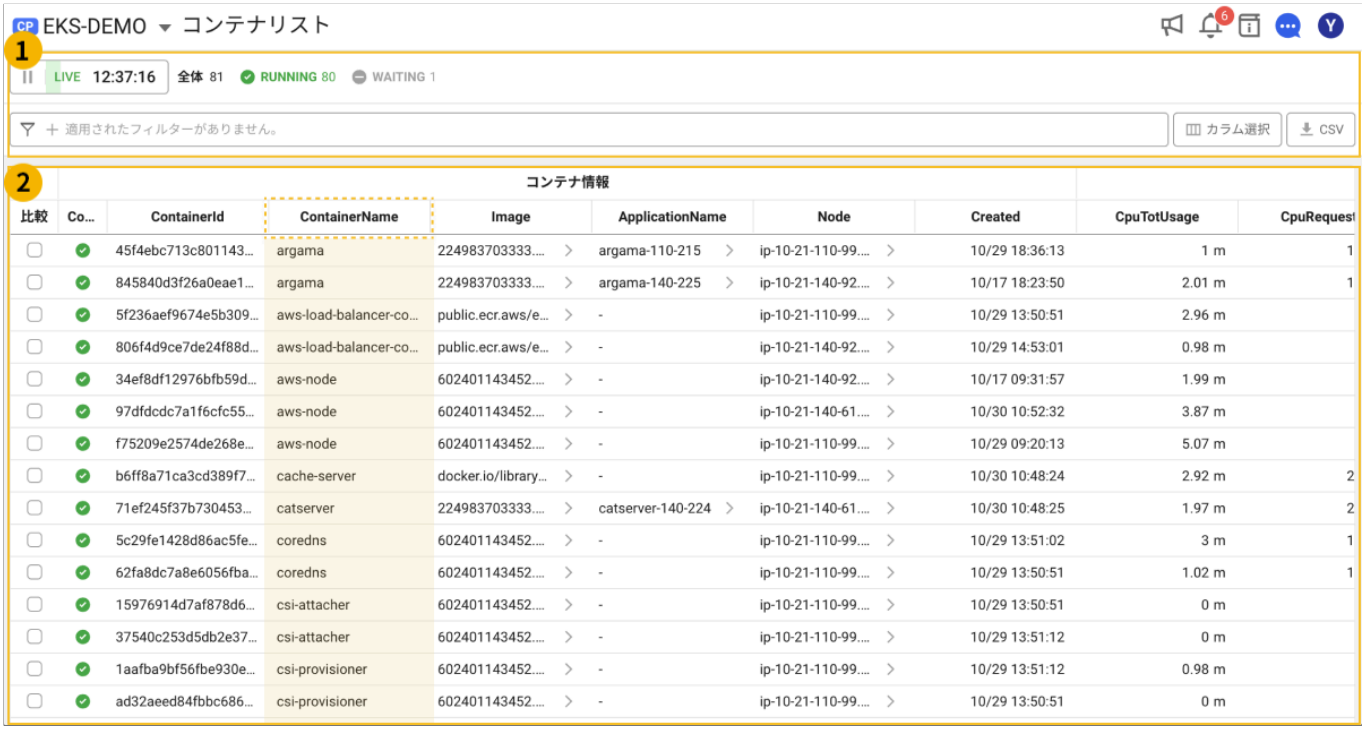
> 特定のカラムのデータが表示されない場合は、Kubernetes Nodeエージェントが古いバージョンである可能性があります。この場合、十分なデータが収集されるようにKubernetesエージェントバージョンを更新してください。

コンテナリストメニューを通じて現在プロジェクトのコンテナの詳細情報をリスト形式で確認することができます。

- 10秒サイクルでリアルタイムのデータを更新します。
- コンテナのステータス要約、比較、フィルター、および_CSV_ファイルのダウンロード機能を提供します。

  CSV ファイル名、kubernetes_contianer_list_ {pcode} {timestamp} .csv_

基本画面ガイド

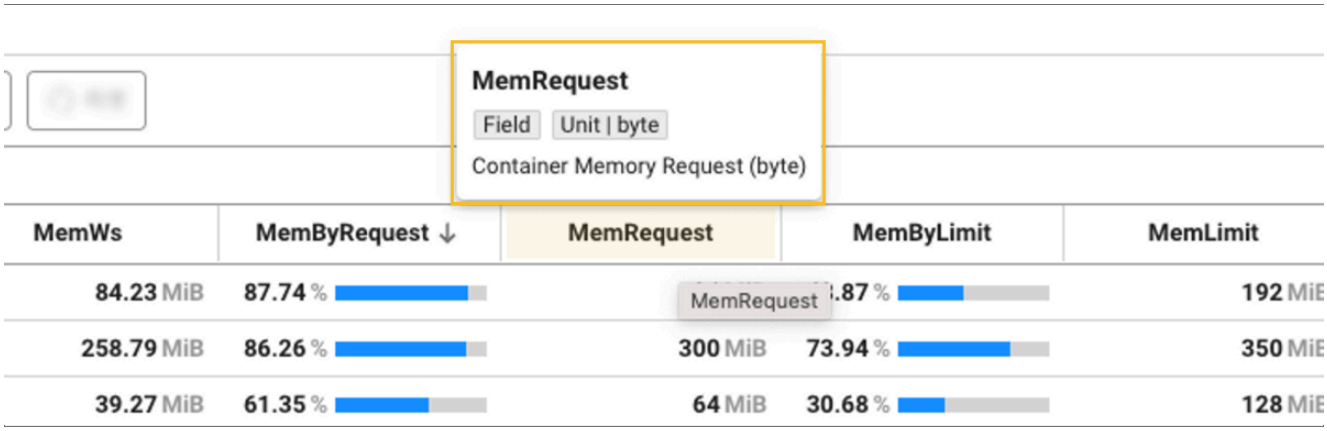


1領域から必要なオプションを選択し2領域でデータを参照できます。

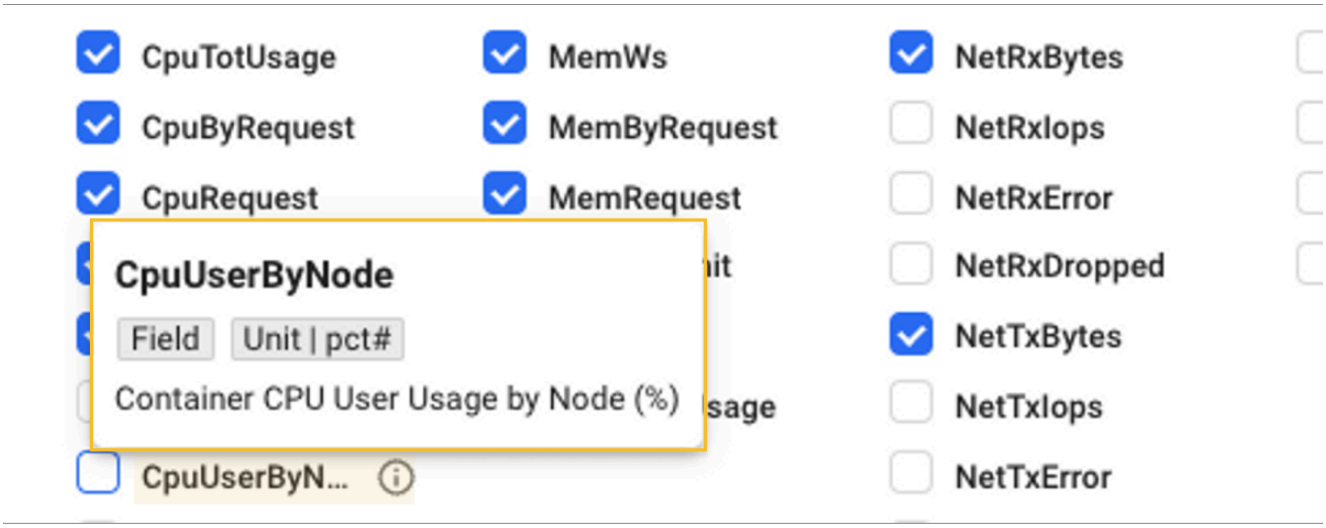
- 1領域の左上で、コンテナ全体の数とステータス別コンテナの数を確認できます。
- 2領域で目的のカラムを選択すると、そのカラムを基準にコンテナー覧を並べ替えて表示することができます。
- 2 領域Node カラムで ノード情報の表示 アイコンを選択すると、 ノードリスト メニューに移動します。

メトリクス情報

- 目的のカラムの上にマウスカーソルを置くと、次のようにカラムが提供するメトリクス情報を確認できます。



- カラム選択オプション画面内のカラム一覧でカラム別メトリクス情報を確認することができます。 ご希望のカラムの上にマウスカーソルを置くと、カラム名の右側にインフォアイコンが表示されます。 インフォアイコン上にマウスカーソルを位置されると、次のようにメトリクス情報を確認できます。



フィルター

フィルターオプションを活用する時、希望するコンテナー一覧を選択して照会できます。 and 条件を使用すると、複数のフィルターが適用できます。

1. フィルター入力ウインドウで＋追加アイコンを選択すると、次のようにフィルターを追加するウィンドウが表示されます。

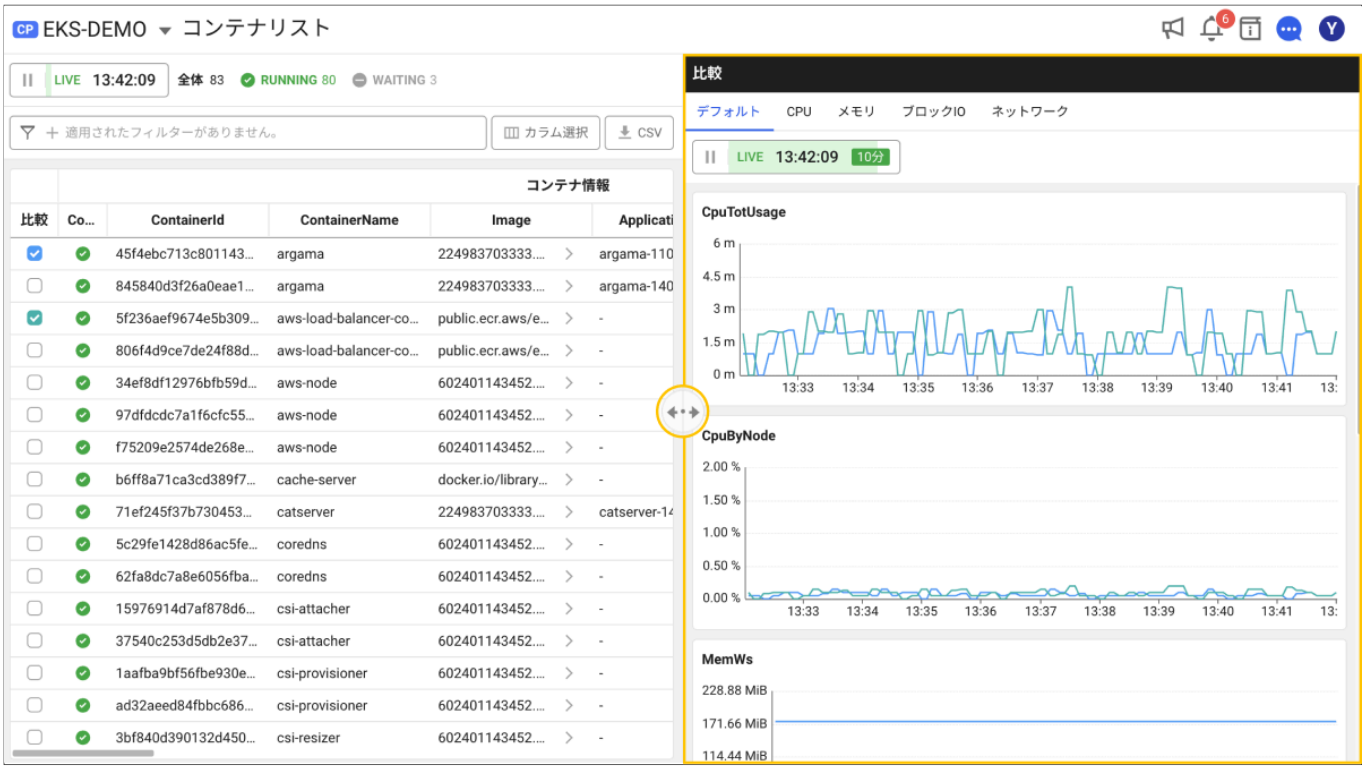


   - フィルターキーと条件を選択します。
   - 条件に合った値を選択します。
2. フィルター設定後、適用ボタンをクリックし、フィルターを適用します。
3. 複数のフィルターを追加するには、フィルター入力ウインドウまたはフィルターを追加するウィンドウで＋

追加アイコンを選択します。

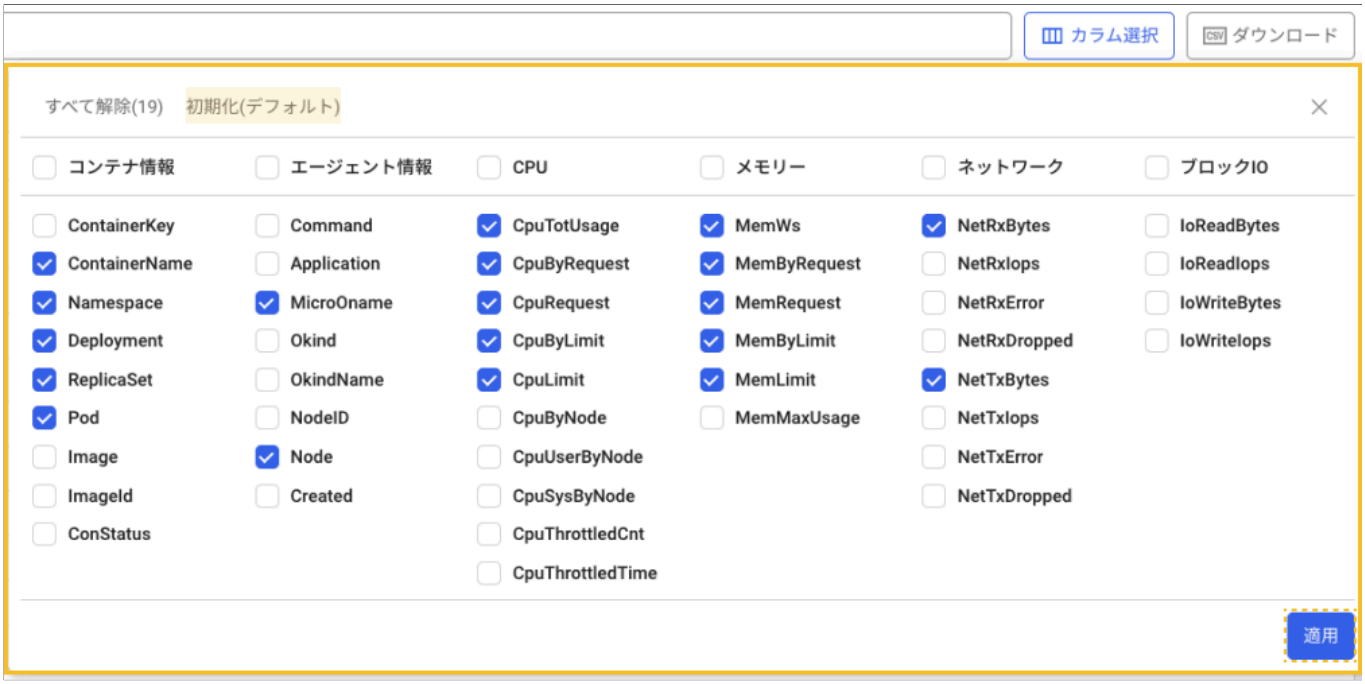
比較する

比較オプションにより、複数のコンテナの主な指標に対する比較チャートを確認できます。一覧で照会を希望するコンテナの比較カラムのチェックボックスを選択すると、次のように右側にそのコンテナの詳細情報を比較するウィンドウが表示されます。



カラム選択

カラム選択オプションを活用する時、希望するコンテナ情報を選択して一覧を照会できます。



- 選択したカラム一覧は、メニューを再入力すると維持されます。
- リセットボタンを選択すると、定義済みの既定の主要指標カラムが選択されます。 この場合、既存のカラム選択は削除されます。
- カラムを選択した後、右下の適用ボタンをクリックして選択履歴を適用することができます。

コンテナタイムライン

ホーム画面 > プロジェクト選択 > **コンテナ** > **コンテナタイムライン**

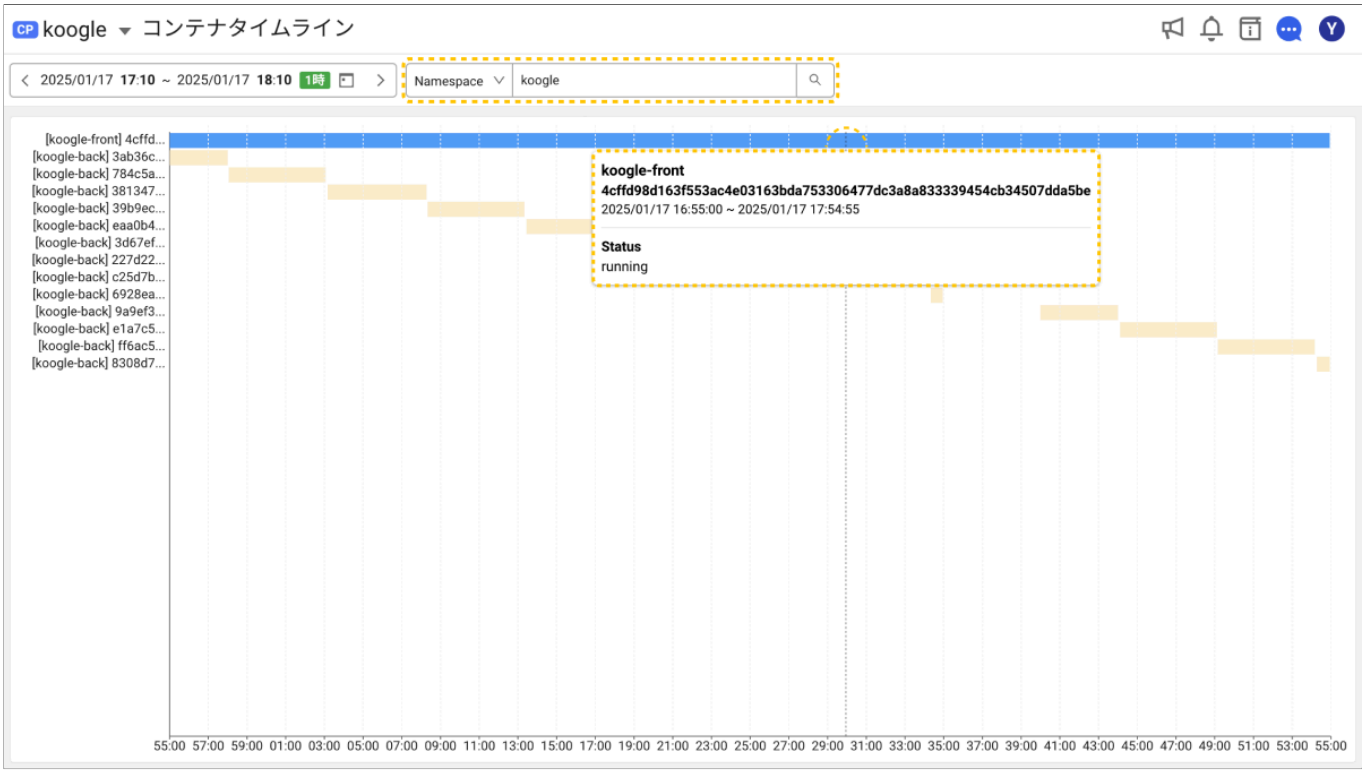
> **要求事項**
>
> WhaTap Kubernetesエージェント1.7.6バージョン以降が必要です。 タイムラインデータは、1.7.6バージョン以降で収集された時点から提供されます。

コンテナタイムラインメニューは時間の経過に伴うコンテナ の状態変化を視覚的に追跡できる機能を提供します。 既に終了されたコンテナ がいつ終了したのか、特定の時点でコンテナ がどのような状態であったかを視覚的に捉えることができます。 これにより、ユーザーは過去の状態を分析し、クラスターで発生した問題の原因把握および運営の安定性をモニタリングすることができます。

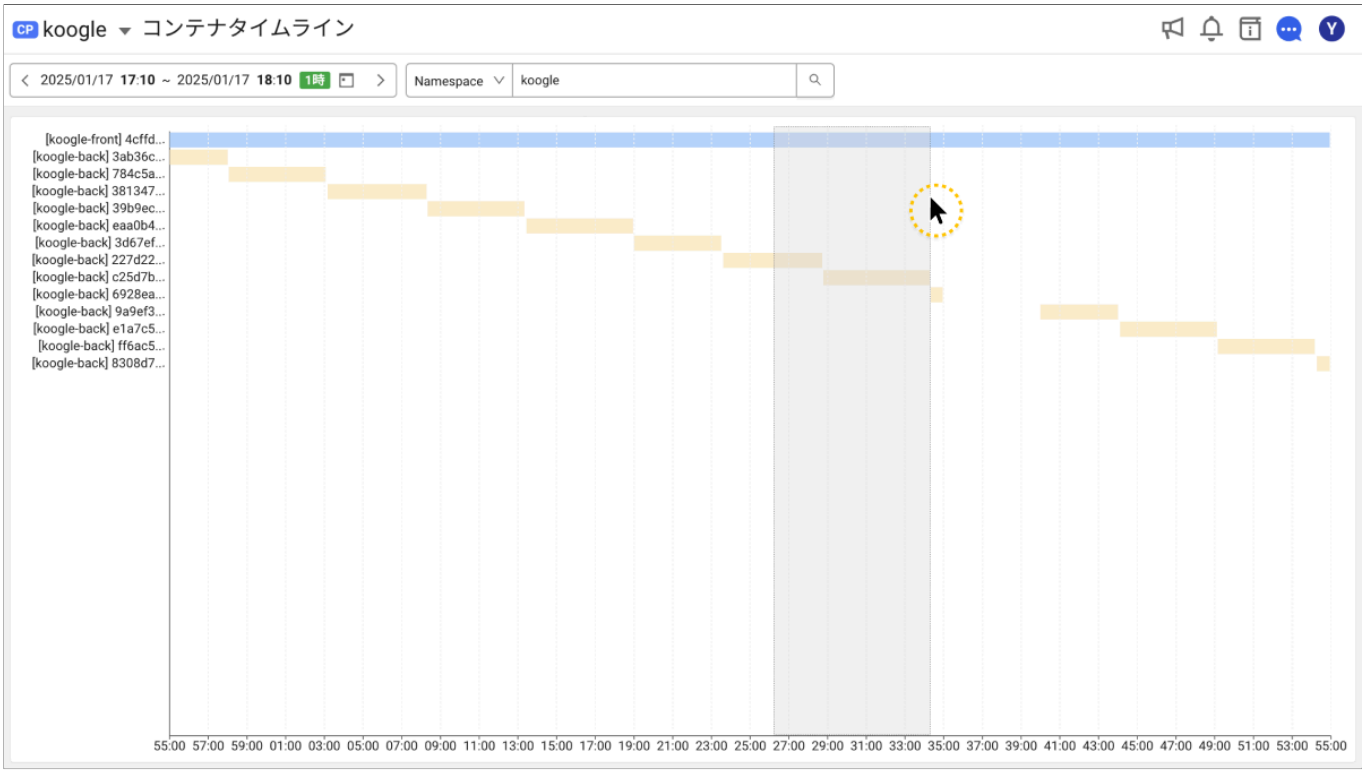
基本画面ガイド

コンテナタイムラインメニューに入った後**フィルターキー**と**フィルタ値**を選択してデータを照会します。 最近1時間の間に収集されたコンテナ 一覧を確認することができます。 時間軸に沿ってコンテナ の状態変化を視覚的に確認できます。

各時間の区間にマウスを乗せると、当該区間の開始時間と終了時間、そしてコンテナの状態(**running**、**waiting**、**terminated**)が表示されます。



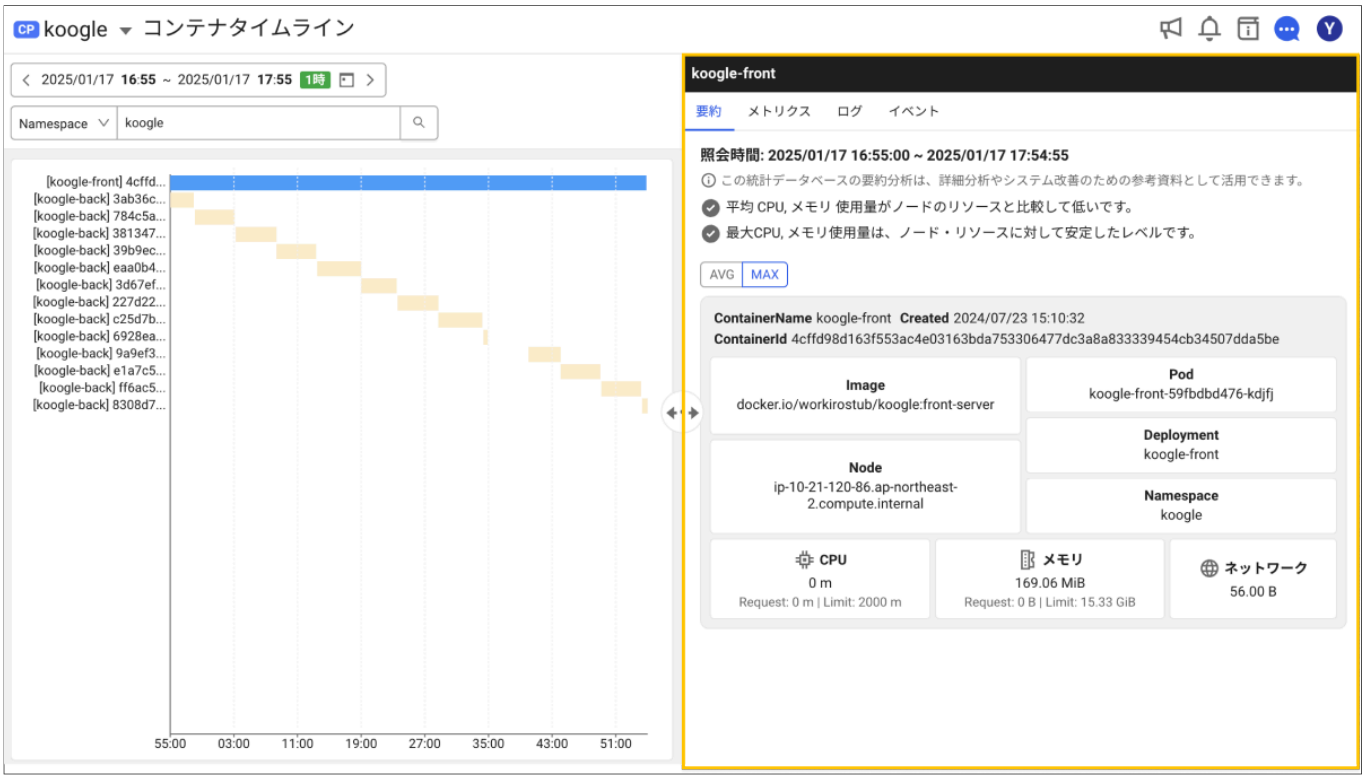
タイムライン上で区間をドラッグして時間範囲を調整できます。 期間を設定して、過去のデータを確認してみてください。



また、特定の期間を選択した場合、当該期間中のコンテナ 状態を**詳細画面**を通じて詳しく確認できます。 詳細は以下コンテナ タイムラインの詳細案内を参照してください。

コンテナ タイムライン詳細

コンテナ タイムラインの詳細は、特定区間又は特定コンテナ を選択して確認できます。



- 詳細情報を照会したい区間を選択し、右側の **詳細を見る** 画面を通じて当該時点のコンテナ に対する詳細を照会できます。
- チャートの左側の一覧で照会したいコンテナ 名前を選択して右側の**詳細を見る** 画面から該当コンテナ の全体の状態を見ることができます。

> ⓘ 詳細ビューのメトリクス、イベント、ログタブの場合、コンテナマップの詳細ビューと同様です。

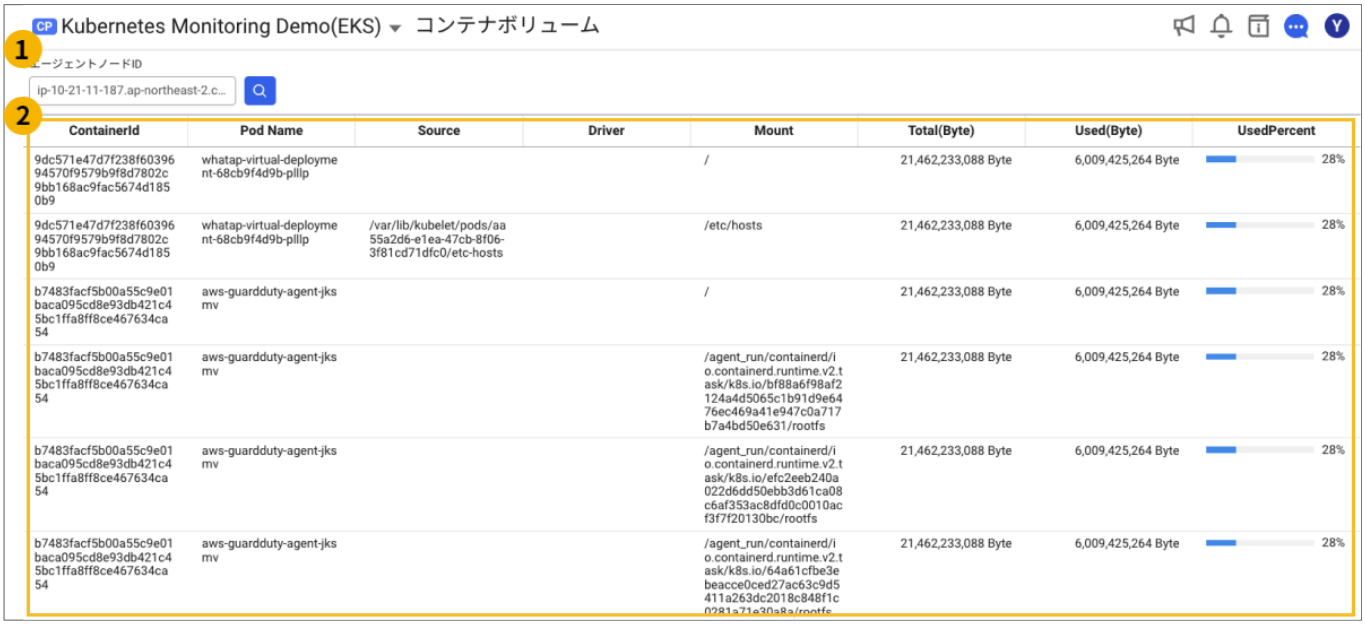
要約

要約タブで選択したコンテナ情報と全体的なリソース使用量情報を要約して提供します。 当該コンテナの画像、コンテナが実行されているノード、コンテナが実行されているPodの `ownerReference` などのコンテナ情報を総合的に確認できます。 ユーザーは、コンテナのリソース使用状況と状態を把握し、性能と運用の安定性をモニタリングできます。 また、**AVG**、**MAX**ボタンをクリックして、照会期間中の平均使用量と最大使用量を確認できます。

コンテナボリューム

コンテナボリュームメニューを通じてコンテナにマウントされたボリュームとその容量状態を確認できます。

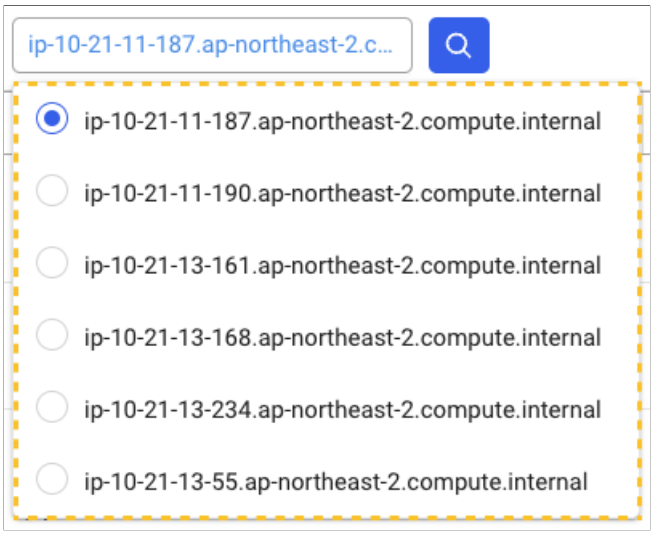
基本画面ガイド



1領域で照会したいエージェントノードを選択し2領域でデータを確認できます。

ノード別照会

入力ウィンドウを選択すると、次のようにノードリストウィンドウが表示されます。照会したいエージェントノードを選択した後、🔍 虫眼鏡ボタンをクリックします。



コンテナボリューム情報

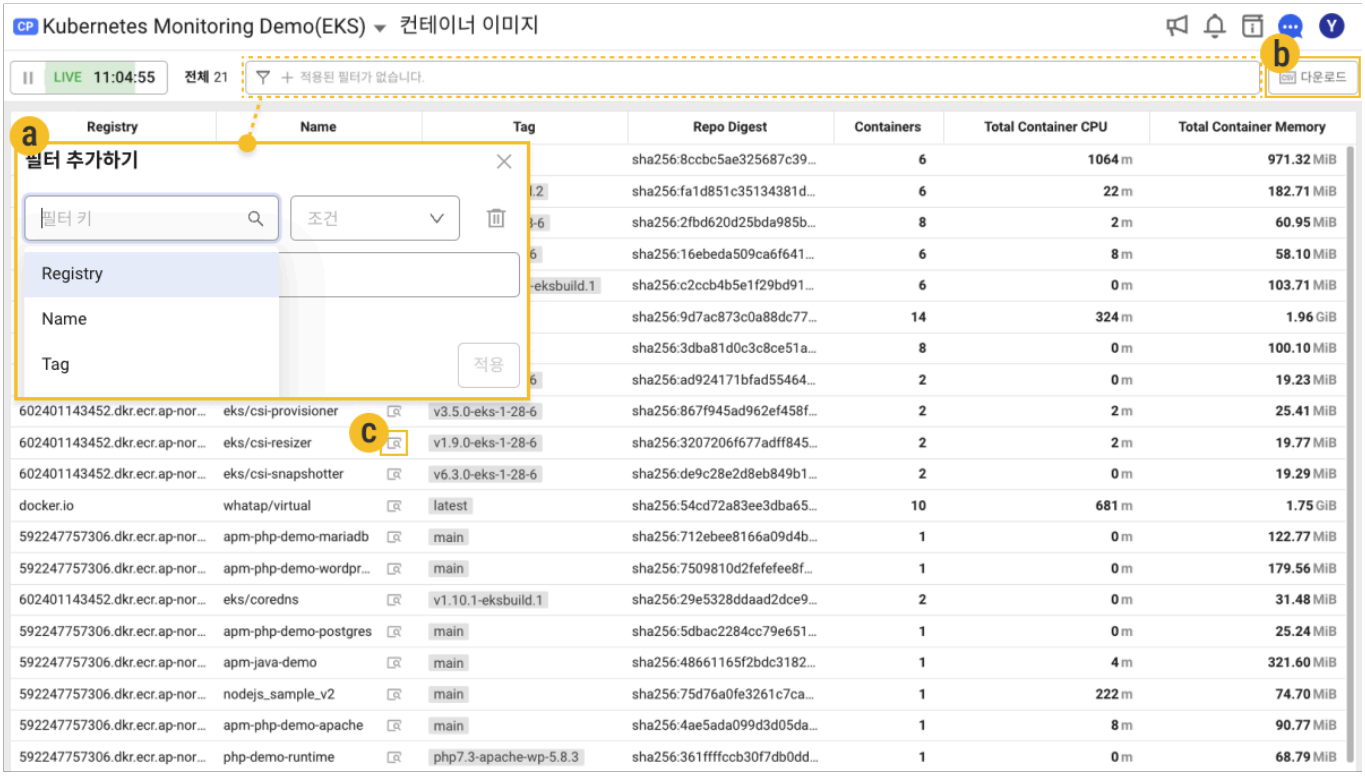
コンテナボリュームメニューで提供する情報項目を次のように案内します。

項目	説明
ContainerId	コンテナの固有識別子であるコンテナIDです。このIDは、コンテナのランタイムによって生成されます。クラスター内でコンテナを識別・管理するために不可欠です。
Pod Name	コンテナが属するPod名です。Kubernetesクラスターでは、複数のコンテナが一つのPod内で実行されることができ、この名前を通じて当該コンテナがどのPodに属しているかを識別できます。
MountType	ボリュームマウントされた方式を示します。ただ、`tmpfs`は、メモリを使用する一時的なファイルシステムであるため、当該コンテナボリューム情報リストに **表示されません。**
	- `bind`：ホスト上の特定のディレクトリがマウントされたことを意味します。 - `volume`: ホスト上のDockerが管理する特定のファイルシステム(例：/var/lib/docker/volumes/)がマウントされたことを意味します。
Source	ホスト上での当該ボリュームのパスです。このパスは、ボリュームデータが実際に保存される場所を意味します。

項目	説明
Driver	当該ボリューム作成時に使用されたボリュームドライバーです。ボリュームドライバーは、データの保存方法とボリュームの管理方法を決定します。現在、この情報は主にドッカーランタイムで提供されます。containerdとCRI-Oのような、より低いレベルのコンテナランタイムでは、高レベルの詳細情報を直接提供しませんが、モニタリングダッシュボードは、ランタイムAPIを通じてこのようなボリュームデータを制限的に提供します。
Mount	コンテナ内でボリュームがマウントされたパスです。このパスにより、コンテナ内のアプリケーションがボリュームに保存されているデータにアクセスできます。
Total(Byte)	ボリュームの全体容量をバイト単位で示します。これは、ボリュームに保存できるデータの最大サイズを意味します。
Used(Byte)	現在ボリュームで使用中の容量をバイト単位で示します。これは、ボリューム内に実際に保存されているデータのサイズを意味します。
UsedPercent	ボリューム容量の使用率をパーセンテージで示します。この比率は、ボリュームの全体容量に対する現在使用中の容量の比率を示しています。使用率が高くなるにつれて、ボリュームスペースが不足していることを意味します。これは、性能の低下やデータ損失のリスクを知らせる重要な指標になりえます。

コンテナイメージ

コンテナイメージメニューを通じてコンテナで使用しているイメージ情報とそのイメージに基づいて実行中のコンテナの個数及びCPU、Memory使用総量をリスト形式で確認できます。



a 필터

フィルターのウィンドウで**フィルターキー**選択時、Registry、Name、Tag、Repo Digestに対するフィルター機能を提供します。

b CSVダウンロード

上段右側の**CSV**ボタンを選択すると、元のデータを_CSV_ファイルでダウンロードできます。 ファイル名は、`kubernetes_container_images_` 後にプロジェクトコードとデータ照会時点のタイムスタンプを加えて作成します。

例、kubernetes_container_images_{pcode}_{timestamp}.csv

Ⓒコンテナの詳細を見る

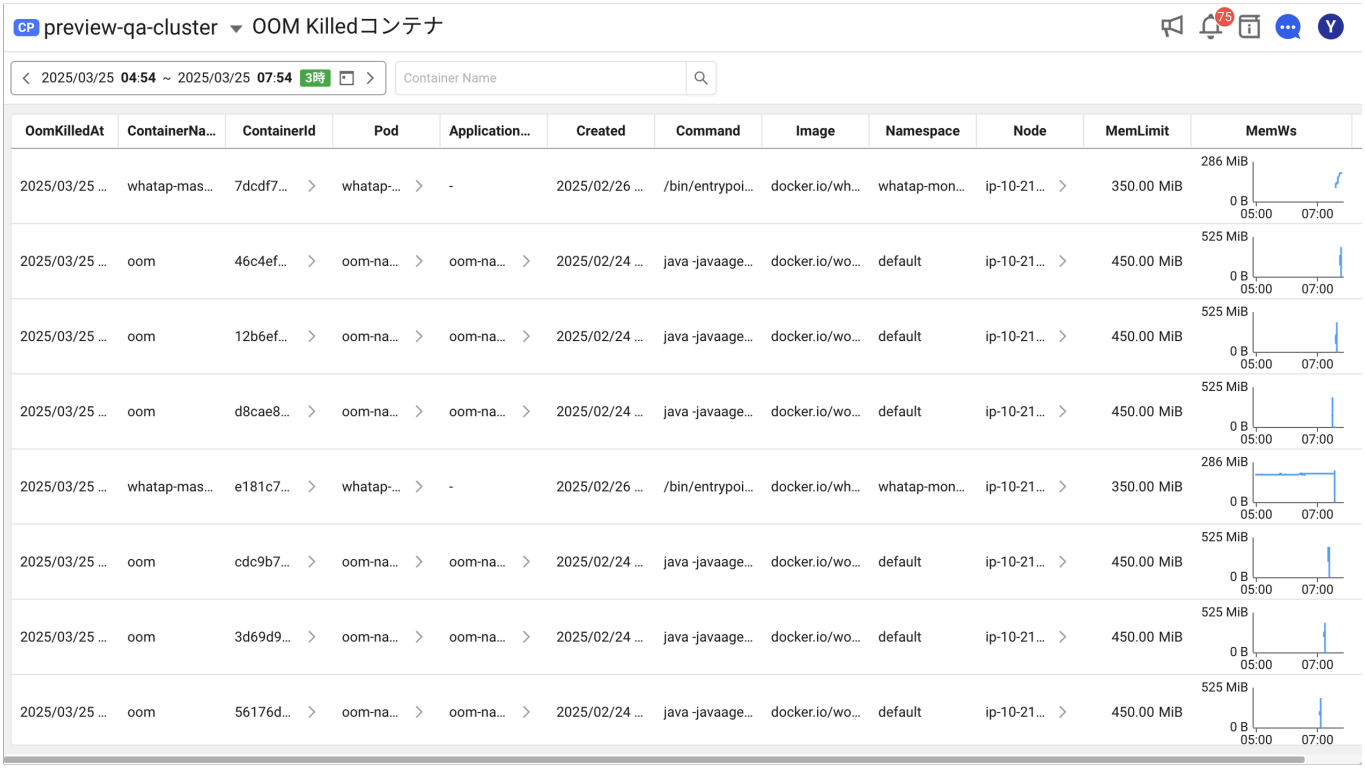
Name カラム右の **コンテナの詳細を表示します。**アイコンを選択すると、そのイメージが検索された**コンテナリスト**メニューに移動します。

Out Of Memory Killedコンテナ

OOM Killedコンテナメニューを通じて**Out Of Memory**(**OOM**)が発生したコンテナをすばやく検索し、その時点でのメモリ使用量の変化の推移を確認できます。変化の推移は急激または漸進的である可能性があります。これを確認し、適切なPOD数とメモリの臨界点に対する客観的な判断が可能です。

> **OOM発生例**
>
> コンテナのメモリ使用量がメモリ制限を超えたり、コンテナが実行中のノードのメモリが不足している場合、Out Of Memoryが発生します。もしコンテナがコード `137` で終了したらOut Of Memoryによって `SIGKILL` を受信したものです。

基本画面ガイド



- 左上のタイムセレクターをクリックして照会したい時間帯を選択できます。最大2日まで照会できます。

- 右上の検索入力画面でコンテナ名を入力すると、照会したい特定のコンテナ情報をすばやく確認できます。
- リストの右の詳細アイコンをクリックすると、選択したコンテナのメモリ使用の推移を確認できます。
- OOMが発生したコンテナ情報をリスト形式で提供します。コンテナの詳細については、次のガイドを参照してください。

OOM Killedコンテナ情報

OOM Killedコンテナメニューで提供するカラム情報を次のように案内します。

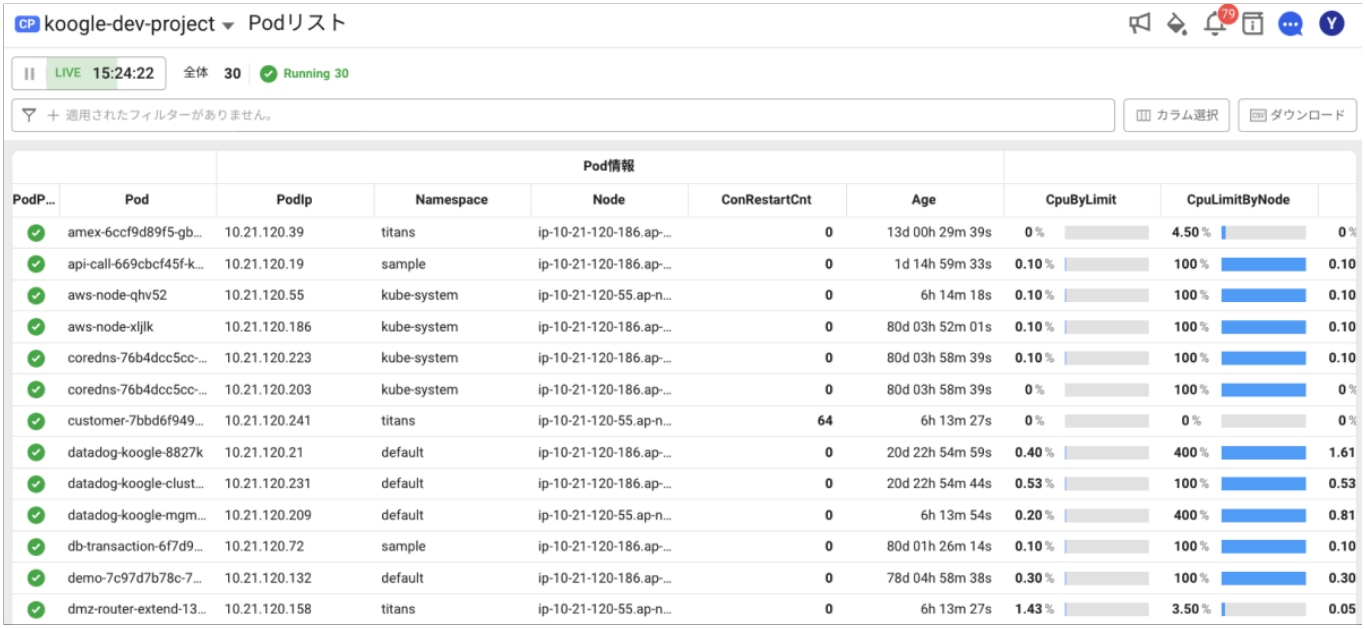
カラム	意味
OomKilledAt	Out Of Memory Killedが発生したTimeStamp
ContainerName	コンテナ名
ContainerId	コンテナのID
Pod	コンテナが属するPod名
ApplicationName	コンテナで実行中のアプリケーション名
Created	コンテナが生成されたTimeStamp
Command	コンテナの実行コマンド
Image	コンテナのイメージ名
Namespace	コンテナのPodが属するネームスペースの名前
Node	コンテナが動作しているノード名
MemLimit	コンテナのメモリLimitサイズ
MemWs	コンテナで非活性化されているメモリを除いて実際に使用中のメモリ

> ⓘ • ContainerIdをクリックするとコンテナ一覧メニューに移動します。

- Podをクリックすると Pod一覧メニューに移動します。
- ApplicationNameクリックすると、該当アプリケーションが検索されたコンテナアプリケーションリストメニューに移動します。
- Nodeをクリックするとノード一覧メニューに移動します。

Podリスト

Pod一覧を照会します。 希望するカラムを選択すると、そのカラムを基準にPod一覧をソートして照会することができます。



	Pod情報							
PodP...	Pod	PodIp	Namespace	Node	ConRestartCnt	Age	CpuByLimit	CpuLimitByNode
	amex-6ccf9d89f5-gb...	10.21.120.39	titans	ip-10-21-120-186.ap-...	0	13d 00h 29m 39s	0 %	4.50 % 0 %
	api-call-669cbcf45f-k...	10.21.120.19	sample	ip-10-21-120-186.ap-...	0	1d 14h 59m 33s	0.10 %	100 % 0.10
	aws-node-qhv52	10.21.120.55	kube-system	ip-10-21-120-55.ap-n...	0	6h 14m 18s	0.10 %	100 % 0.10
	aws-node-xljlk	10.21.120.186	kube-system	ip-10-21-120-186.ap-...	0	80d 03h 52m 01s	0.10 %	100 % 0.10
	coredns-76b4dcc5cc-...	10.21.120.223	kube-system	ip-10-21-120-186.ap-...	0	80d 03h 58m 39s	0.10 %	100 % 0.10
	coredns-76b4dcc5cc-...	10.21.120.203	kube-system	ip-10-21-120-186.ap-...	0	80d 03h 58m 39s	0 %	100 % 0 %
	customer-7bbd6f949...	10.21.120.241	titans	ip-10-21-120-55.ap-n...	64	6h 13m 27s	0 %	0 % 0 %
	datadog-koogle-8827k	10.21.120.21	default	ip-10-21-120-186.ap-...	0	20d 22h 54m 59s	0.40 %	400 % 1.61
	datadog-koogle-clust...	10.21.120.231	default	ip-10-21-120-186.ap-...	0	20d 22h 54m 44s	0.53 %	100 % 0.53
	datadog-koogle-mgm...	10.21.120.209	default	ip-10-21-120-55.ap-n...	0	6h 13m 54s	0.20 %	400 % 0.81
	db-transaction-6f7d9...	10.21.120.72	sample	ip-10-21-120-186.ap-...	0	80d 01h 26m 14s	0.10 %	100 % 0.10
	demo-7c97d7b78c-7...	10.21.120.132	default	ip-10-21-120-186.ap-...	0	78d 04h 58m 38s	0.30 %	100 % 0.30
	dmz-router-extend-13...	10.21.120.158	titans	ip-10-21-120-55.ap-n...	0	6h 13m 27s	1.43 %	3.50 % 0.05

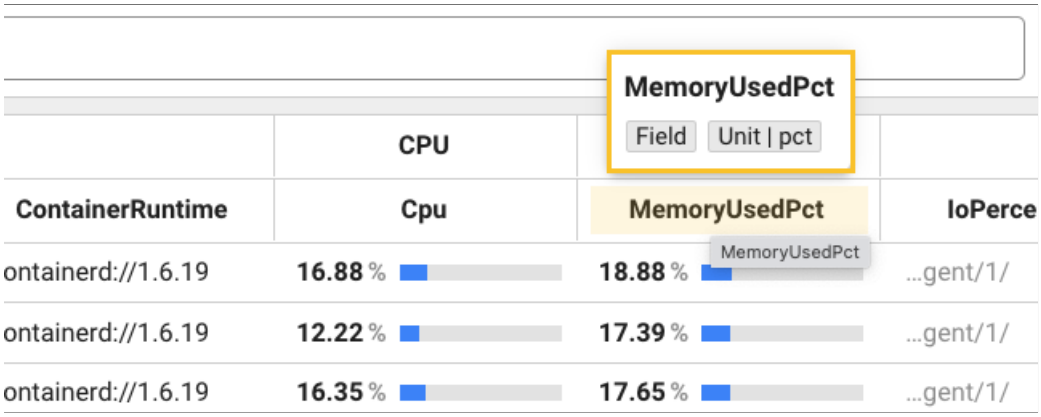
- 10秒サイクルでリアルタイムのデータを更新します。
- 左上で全体のPod数と状態別のPod数を確認することができます。
- 検索、カラム整列、詳細を見る、_CSV_ファイルのダウンロードを提供します。

  CSV ファイル名、kubernetes_pod_list_ {pcode} _ {timestamp} .csv

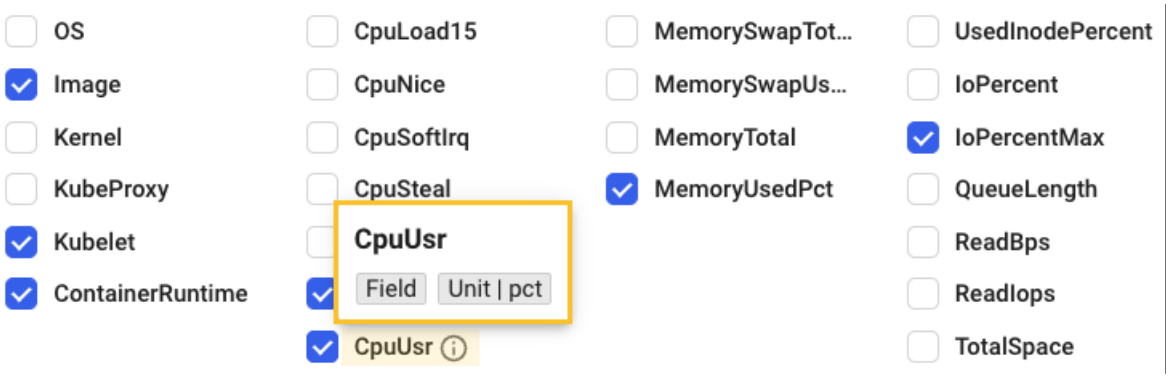
> ⓘ ノードリストとコンテナリストおよびPodリストとコンテナアプリケーションリストなどリスト型メニューの**メトリクス情報**、**フィルター**、**カラム選択**案内は同じです。

メトリクス情報

- 目的のカラムの上にマウスカーソルを置くと、次のようにカラムが提供するメトリクス情報を確認できます。



- カラム選択オプション画面内のカラム一覧でカラム別メトリクス情報を確認することができます。 ご希望のカラムの上にマウスカーソルを置くと、カラム名の右側にⓘ インフォアイコンが表示されます。 ⓘ インフォアイコン上にマウスカーソルを位置されると、次のようにメトリクス情報を確認できます。



フィルター

フィルターオプションを活用する時、希望するPod一覧を選択して照会できます。 `and` 条件を使用すると、複数のフィルターが適用できます。

1. フィルター入力ウインドウで＋追加アイコンを選択すると、次のようにフィルターを追加するウィンドウが表示されます。

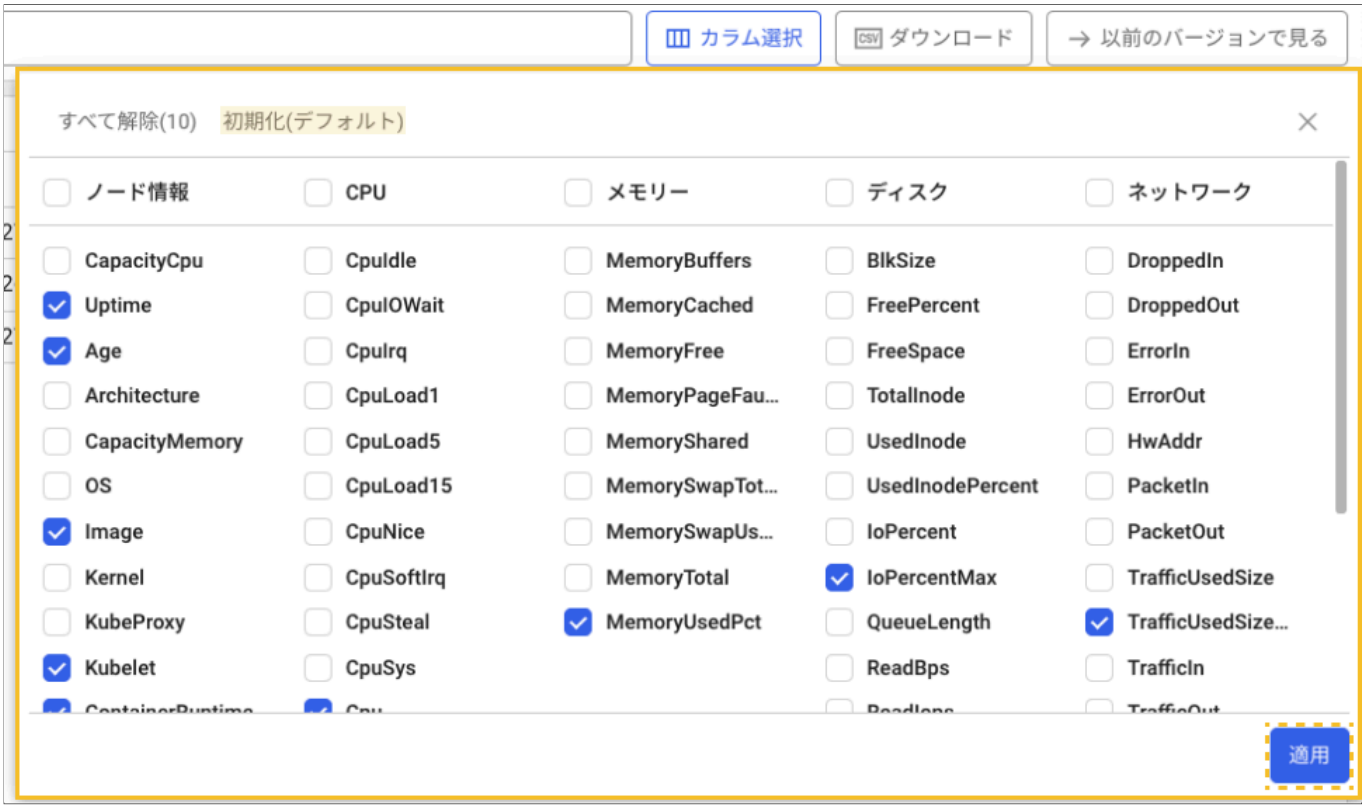


- フィルターキーと条件を選択します。
- 条件に合った値を選択します。

2. フィルター設定後、適用ボタンをクリックし、フィルターを適用します。
3. 複数のフィルターを追加するには、フィルター入力ウインドウまたはフィルターを追加するウィンドウで＋追加アイコンを選択します。

カラム選択

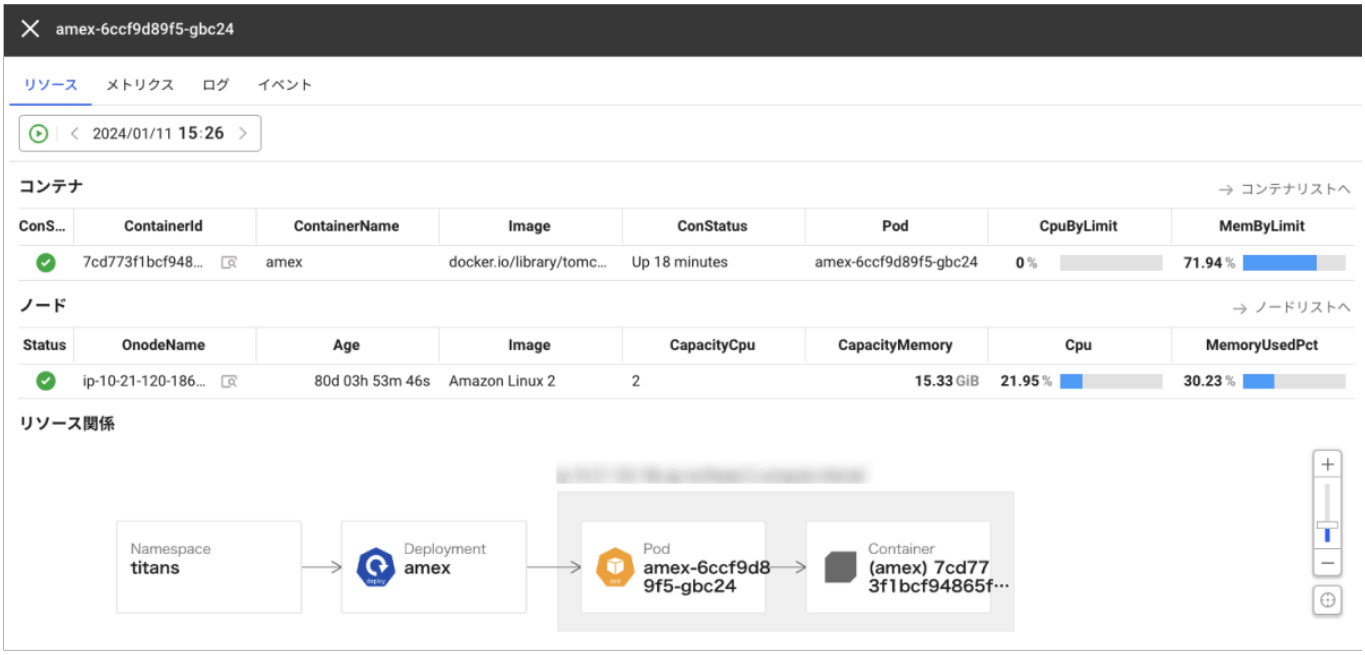
カラム選択オプションを活用する時、希望するPod情報を選択して一覧を照会できます。



- 選択したカラム一覧は、メニューを再入力すると維持されます。
- リセットボタンを選択すると、定義済みの既定の主要指標カラムが選択されます。 この場合、既存のカラム選択は削除されます。
- カラムを選択した後、右下の適用ボタンをクリックして選択履歴を適用することができます。

詳細を確認

一覧で照会を希望するPodの🔍 詳細を見るアイコンを選択する時に、該当Podに関する詳細情報を確認できます。



詳細画面のトレース、メトリクス、ログ、イベント、Pod要約分析 タブの場合、コンテナ マップの詳細ビューと同じです。

リソース

対象を選択した後、詳細表示の**リソース**タブから選択したリソースに関連するリソースの一覧と関係図を照会できます。

要約情報

選択したリソースに関連付けられているリソース一覧の要約情報を提供します。 リソース名の横にある詳細ビュー アイコンを選択すると、そのリソース一覧の画面に移動します。

- **コンテナ**：当該コンテナのみフィルタリングされた状態の**コンテナリスト**に移動します。
- **Pod**：該当するPodのみフィルタリングされた**Podリスト**に移動します。
- **ノード**：該当ノード名が選択された状態の**ノード詳細情報**に移動します。

リソース関係図

リソース間の接続関係と状態を視覚的に確認できます。 一覧と同様に、Pod、コンテナ、アプリケーションにマウスオーバーすると、詳細指標を確認できます。 リソースを選択すると、そのリソース一覧ページに移動します。

Pod要約分析

- 基本情報: Podのクラスター名、ネームスペース、起動時間、現在の状況(Phase) などを確認できます。
- Kubernetesイベント: 最近のPodに関連するイベントログから問題の発生有無を確認できます。
- 実行情報: Podを生成したコントローラー(DaemonSet、Deploymentなど)を識別します。
- ノード情報: Podが実行中のノードのOS、カーネルバージョン、CPU、メモリなどの詳細な仕様を提供します。
- リソース使用状況: Pod内のすべてのコンテナのCPUとメモリ使用量、設定されたリソース限度(Limit)と比較した使用率を示しています。
- サービスネットワーキング情報: 接続されたService及びIngress情報を確認できます。

> ⓘ 現在、要約分析機能は韓国語のみで提供しています。

Pod タイムライン

ホーム画面 > プロジェクト選択 > **ワークロード** > **Pod タイムライン**

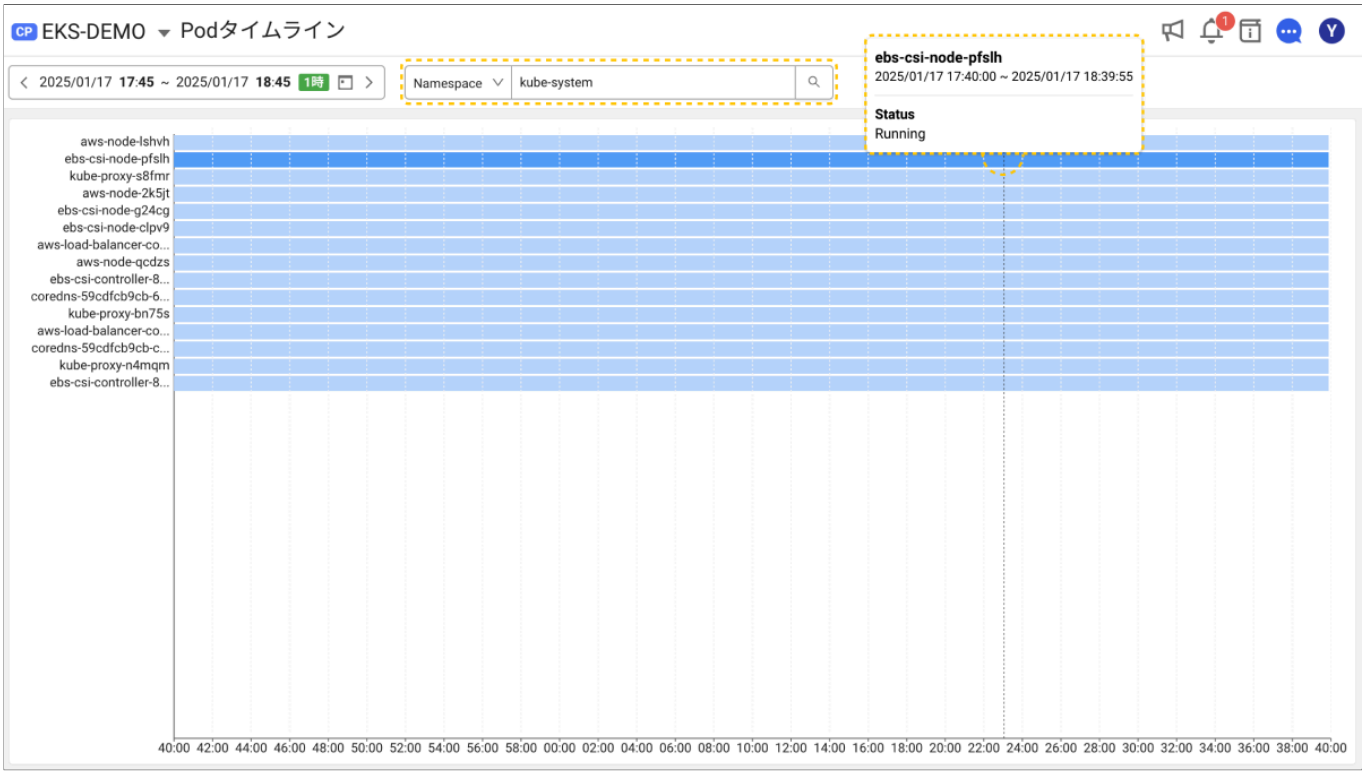
> **要求事項**
>
> WhaTap Kubernetesエージェント1.7.6バージョン以降が必要です。 タイムラインデータは、1.7.6バージョン以降で収集された時点から提供されます。

Pod タイムラインメニューは時間の経過に伴うPodの状態変化を視覚的に追跡できる機能を提供します。 既に終了されたPodがいつ終了したのか、特定の時点でPodがどのような状態であったかを視覚的に捉えることができます。 これにより、ユーザーは過去の状態を分析し、クラスターで発生した問題の原因把握および運営の安定性をモニタリングすることができます。

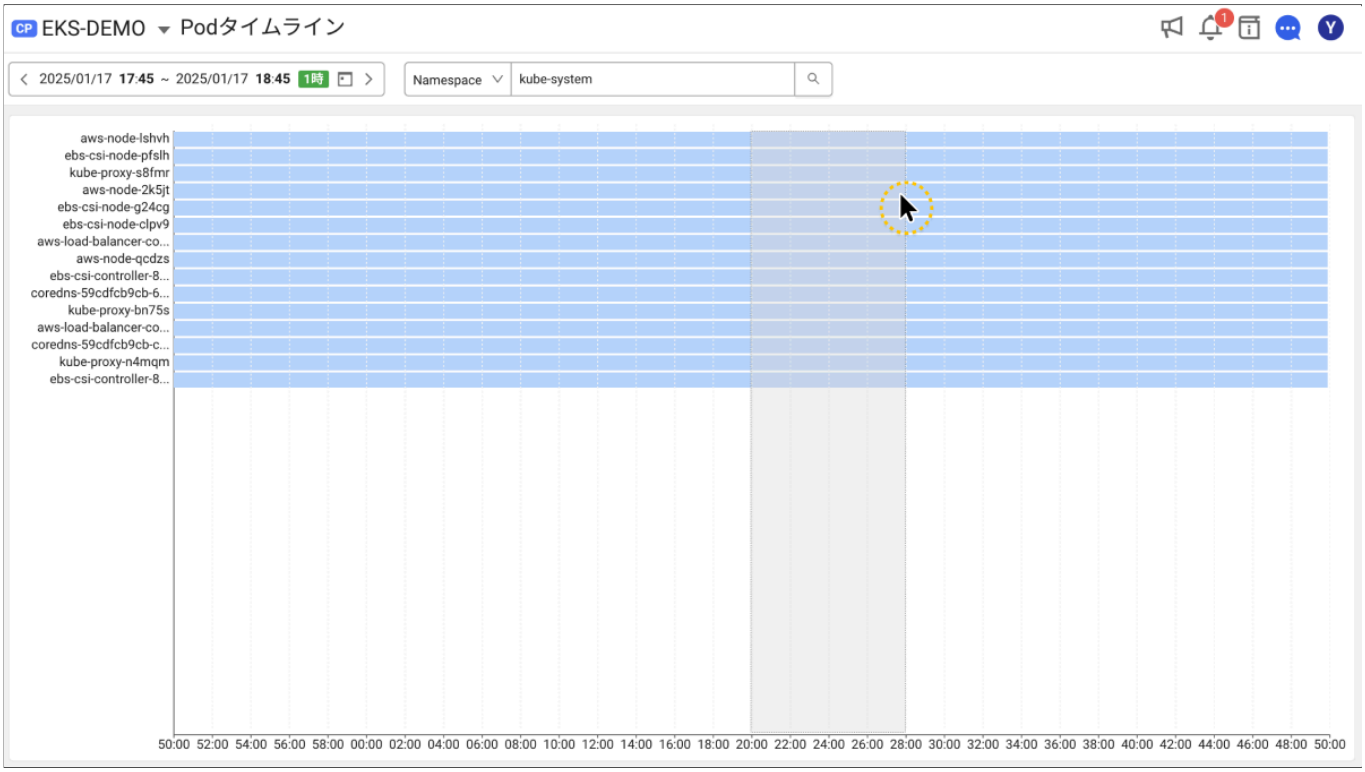
基本画面ガイド

Pod タイムラインメニューに入った後**フィルターキー**と**フィルタ値**を選択してデータを照会します。 最近1時間の間に収集されたPod一覧を確認することができます。 時間軸に沿ってPodの状態変化を視覚的に確認できます。

各時間の区間にマウスを乗せると、当該区間の開始時間と終了時間、そしてPodの状態(**Running**、**Pending**、**Succeeded**、**Failed**)が表示されます。



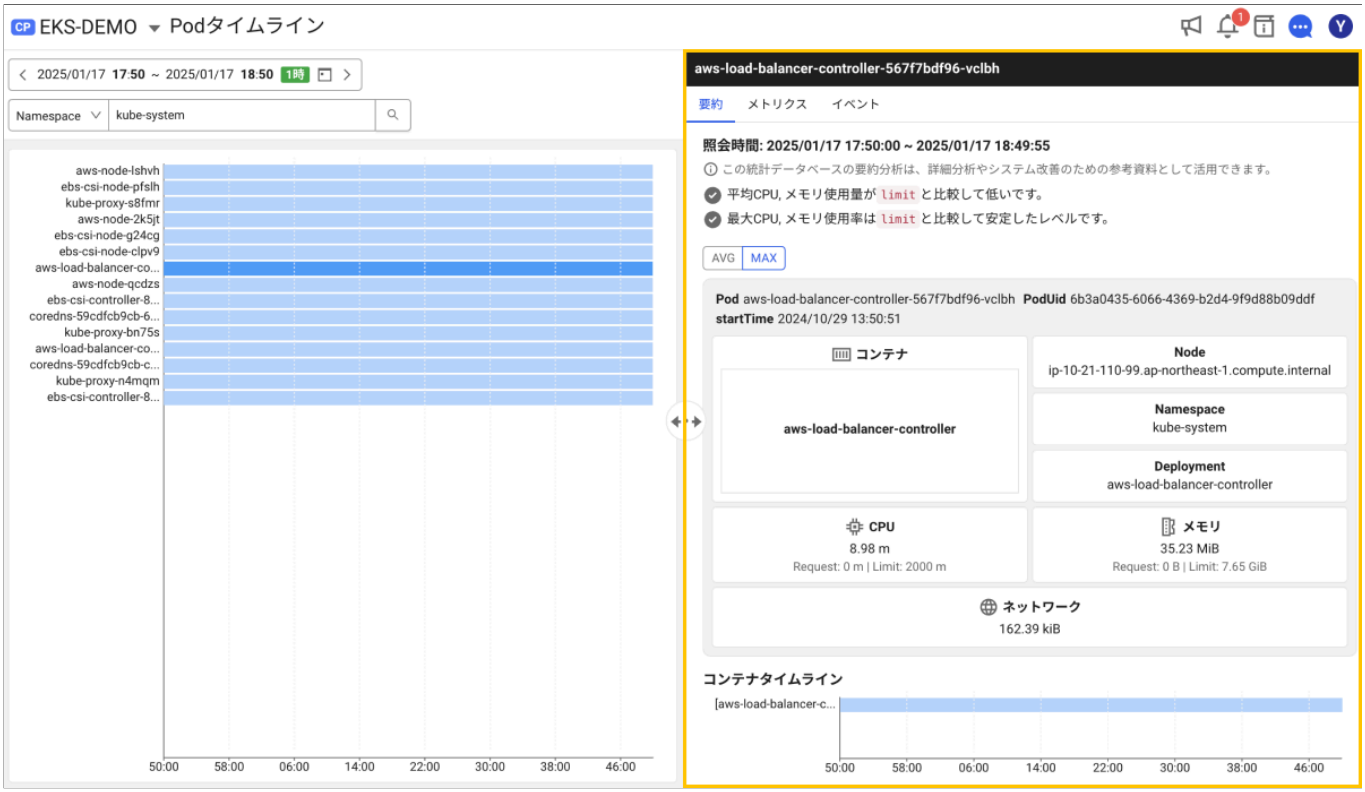
タイムライン上で区間をドラッグして時間範囲を調整できます。 期間を設定して、過去のデータを確認してみてください。



また、特定の期間を選択した場合、当該期間中のPod状態を**詳細画面**を通じて詳しく確認できます。 詳細は以下Podタイムラインの詳細案内を参照してください。

Pod タイムライン詳細

Podタイムラインの詳細は、特定区間又は特定Podを選択して確認できます。



- 詳細情報を照会したい区間を選択し、右側の **詳細を見る** 画面を通じて当該時点のPodに対する詳細を照会できます。
- チャートの左側の一覧で照会したいPod名前を選択して右側の**詳細を見る** 画面から該当Podの全体の状態を見ることができます。

> ⓘ 詳細ビューのメトリクス、イベントタブの場合、コンテナマップの詳細ビューと同様です。

要約

要約タブで選択したPod情報及び全般的なリソース使用量情報を要約して提供します。 当該Podで実行中のコンテナ、Podが実行されているノード、 `ownerReference` などPod情報を総合的に確認できます。 ユーザーはPodのリソース使用状況と各構成要素の状態を把握し、パフォーマンスと運用安定性をモニタリングできます。

- **AVG**、**MAX**ボタンをクリックして、照会期間中の平均使用量と最大使用量を確認できます。
- Pod内で実行されたコンテナのタイムラインを提供します。 コンテナのタイムラインを選択した場合、当該タイム

ラインが選択された**コンテナタイムライン**メニューが新しいタブで開きます。

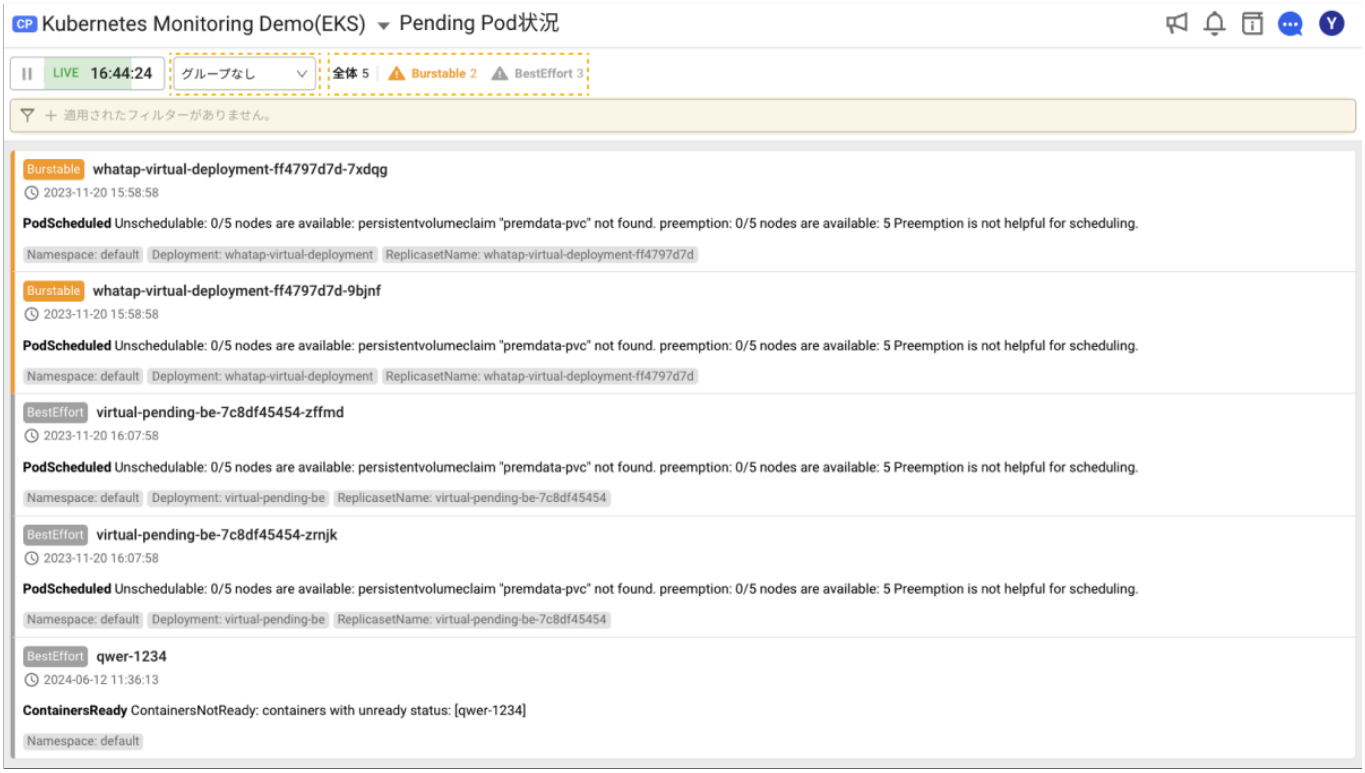
Pending Podの状況

> **要求事項**
>
> WhaTap Kubernetesエージェント1.7.0バージョン以降が必要です。

Pending状態のPod状況を照会して各Podの待ち原因を総合的に分析できます。 クラスタ内に多くのPending Podがあるほど、システムの効率と安定性が低下する可能性があります。 Pending状態のPodをモニタリングし、これらの問題を早期に発見して解決することで、クラスタの安定性を維持します。

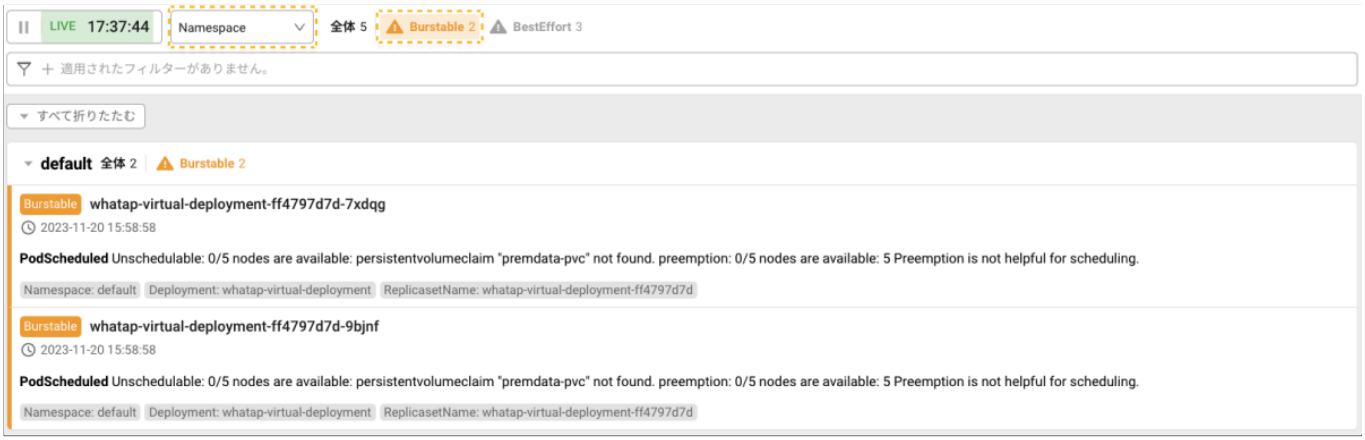
Pending Pod一覧

現在Pending状態であるPod一覧を次のように照会できます。 各Pending PodごとにQoS Class、Pod名、タイムスタンプ、イベントメッセージ、ネームスペースおよびそのPodのデプロイ情報を簡単に確認できます。 Pod選択時、Pending Podの詳細を照会できます。 詳細については次の文書を参照してください。



グループ化とQoS Classの照会

オブジェクト別**グループ化**機能を提供します。 特定のネームスペース、ノード、またはレプリカセットに関連するすべてのPeding Pod状態の情報を簡単に確認し、問題の識別と分析に容易です。 また**QoS Class別のPending状態のPod数**を確認できます。 QoS Class選択時に当該QoS Classに属するPodを一覧で照会できます。



> **QoS Class**(Quality of Service Class)についての詳細は、次の文書を参照してください。

フィルター

フィルター機能により、追加の条件別検索が可能です。 上段のフィルター選択画面の他に、一覧の各Pending Pod下段のタグを通じてフィルター機能を活用できます。 タグの選択時、そのオブジェクトに対する一致条件が追加され、条件に合うタグは青色で表示されます。



Pending Podの詳細

Pending Podの一覧でPodを選択すると、そのPodに関する詳細情報をSuggestions、 Event、Logタブを通じて次のように確認できます。



Suggestions

現在PodがPending状態にある原因と、問題解決のための様々な情報を次のように提供します。

- **QoS Classの分析**

  QoS Classを分析してスケジューリング推奨事項を確認できます。

- **詳細データ収集時刻**

  その詳細情報が収集された時間を確認できます。

- **Pending Podのイベントメッセージと問題解決の提案**

イベントメッセージを確認し、これに基づいて問題の識別と解決策を提案します。

- **問題関連の情報**

  発生した問題に関する情報を提供します。 例えば、ボリュームマウントのプロセスに問題がある場合は、ボリューム関連の設定を案内し、ノードリソースに問題がある場合は、そのリソースを確認できる関連情報を案内します。

Event

選択したPodで実行中のコンテナのイベント情報を提供します。 コンテナマップのイベントタブと同じです。 詳細については、次の文書を参照してください。

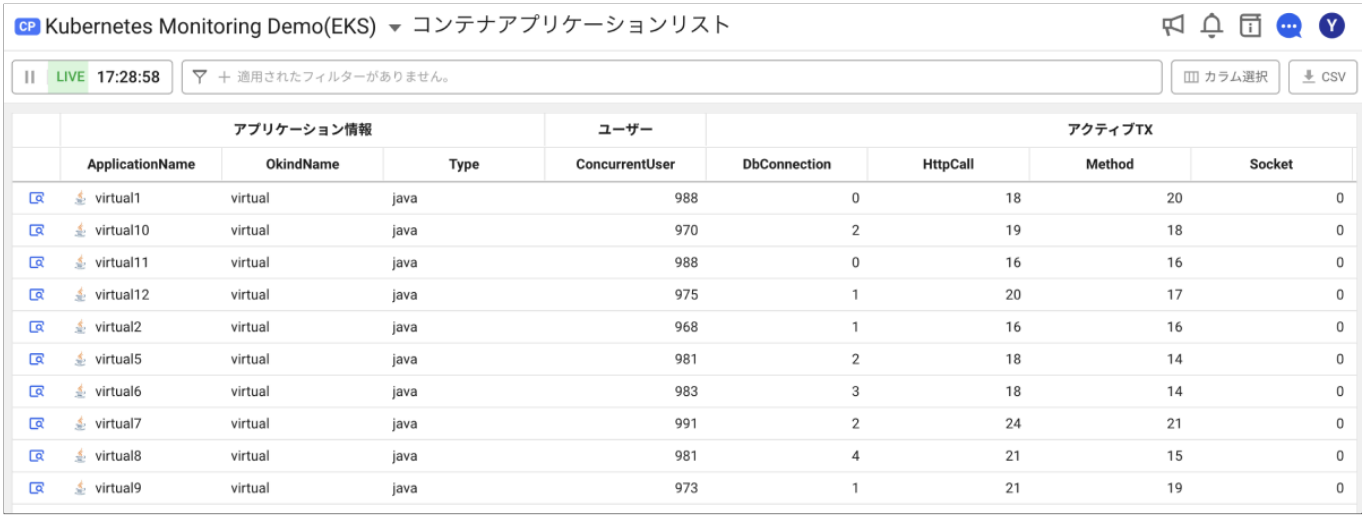
Log

選択したPodで実行中のコンテナのログ情報を提供します。 コンテナマップのLogタブと同じです。 詳細については、次の文書を参照してください。

> ⓘ コンテナログは、コンテナが実行された後にのみ確認できます。 もし、Podがコンテナ実行前のPending状態にとどまっている場合は、そのPodのコンテナログは確認できません。

コンテナアプリケーション一覧

コンテナ化されたアプリケーションの一覧を照会します。 希望するカラムを選択すると、そのカラムを基準にアプリケーション一覧をソートして照会することができます。



	アプリケーション情報			ユーザー	アクティブTX			
	ApplicationName	OkindName	Type	ConcurrentUser	DbConnection	HttpCall	Method	Socket
	virtual1	virtual	java	988	0	18	20	0
	virtual10	virtual	java	970	2	19	18	0
	virtual11	virtual	java	988	0	16	16	0
	virtual12	virtual	java	975	1	20	17	0
	virtual2	virtual	java	968	1	16	16	0
	virtual5	virtual	java	981	2	18	14	0
	virtual6	virtual	java	983	3	18	14	0
	virtual7	virtual	java	991	2	24	21	0
	virtual8	virtual	java	981	4	21	15	0
	virtual9	virtual	java	973	1	21	19	0

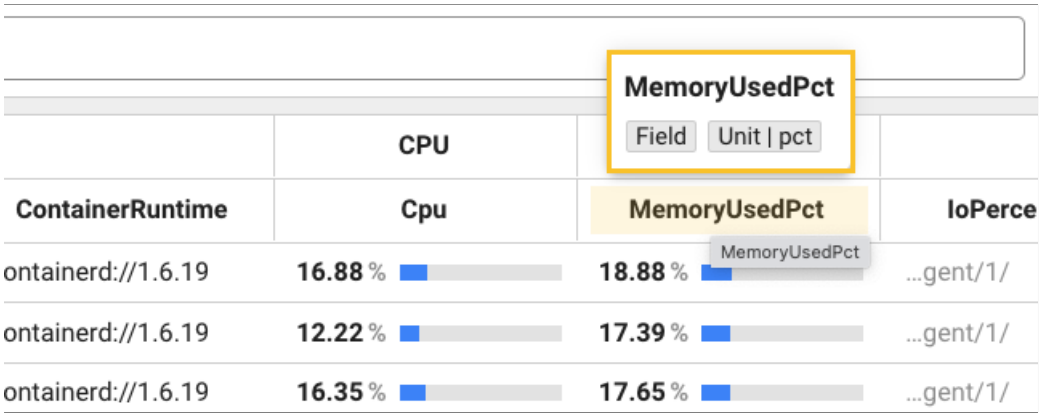
- 10秒サイクルでリアルタイムのデータを更新します。
- 検索、カラム整列、詳細を見る、_CSV_ファイルのダウンロードを提供します。

  CSV ファイル名、kubernetes_application_list_ {pcode} _ {timestamp} .csv

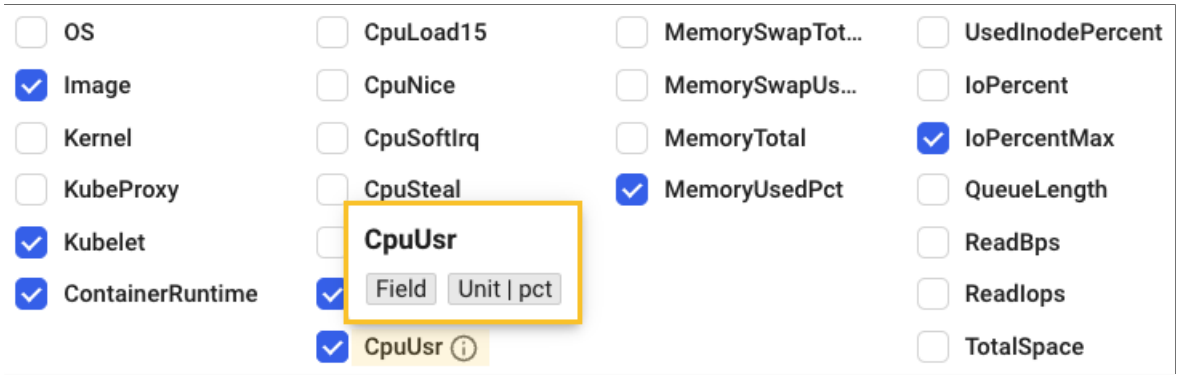
> ⓘ ノードリストとコンテナリストおよびPodリストとコンテナアプリケーションリストなどリスト型メニューの**メトリクス情報**、**フィルター**、**カラム選択**案内は同じです。

メトリクス情報

- 目的のカラムの上にマウスカーソルを置くと、次のようにカラムが提供するメトリクス情報を確認できます。



- カラム選択オプション画面内のカラム一覧でカラム別メトリクス情報を確認することができます。 ご希望のカラムの上にマウスカーソルを置くと、カラム名の右側にⓘ インフォアイコンが表示されます。 ⓘ インフォアイコン上にマウスカーソルを位置されると、次のようにメトリクス情報を確認できます。



フィルター

フィルターオプションを活用する時、希望するアプリケーション一覧を選択して照会できます。 `and` 条件を使用すると、複数のフィルターが適用できます。

1. フィルター入力ウインドウで＋追加アイコンを選択すると、次のようにフィルターを追加するウィンドウが表示されます。

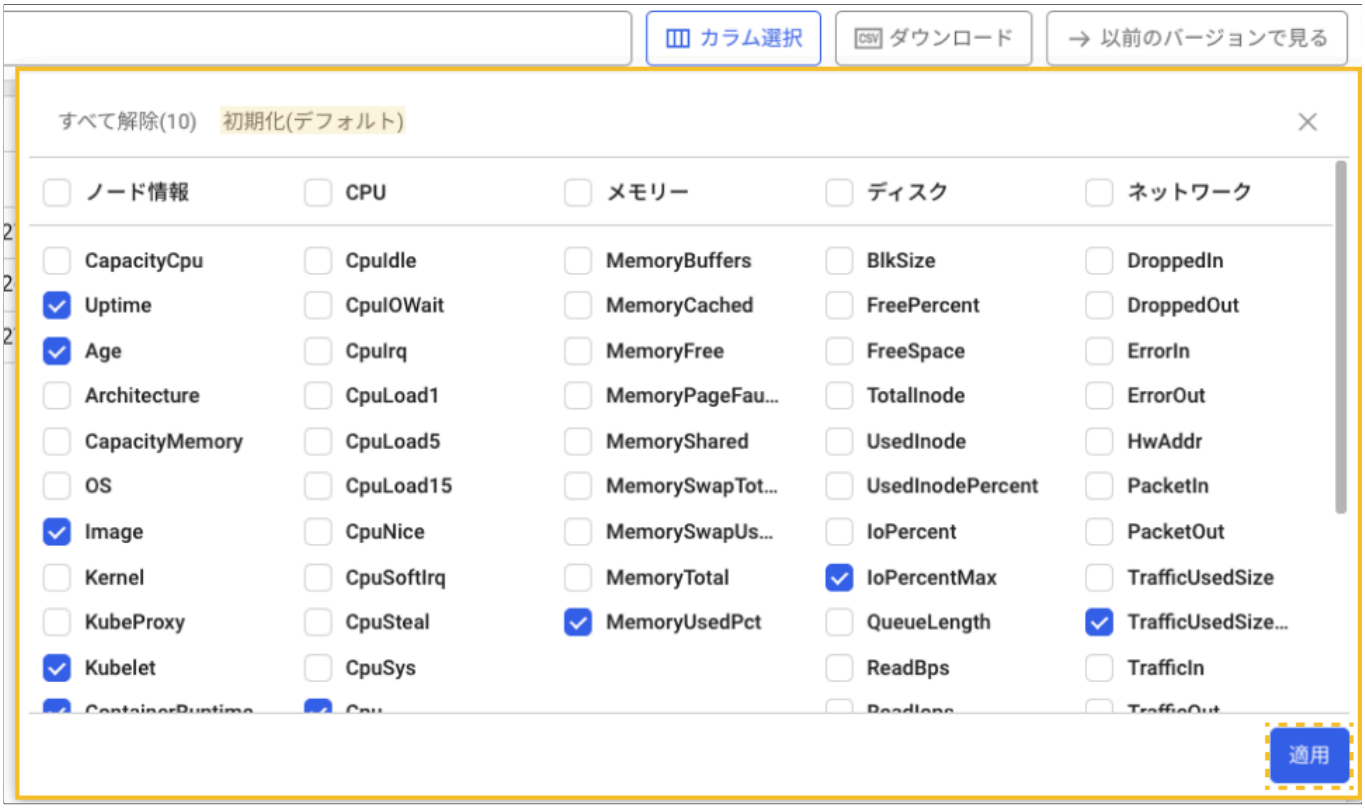


- フィルターキーと条件を選択します。
- 条件に合った値を選択します。

2. フィルター設定後、適用ボタンをクリックし、フィルターを適用します。
3. 複数のフィルターを追加するには、フィルター入力ウインドウまたはフィルターを追加するウィンドウで＋追加アイコンを選択します。

カラム選択

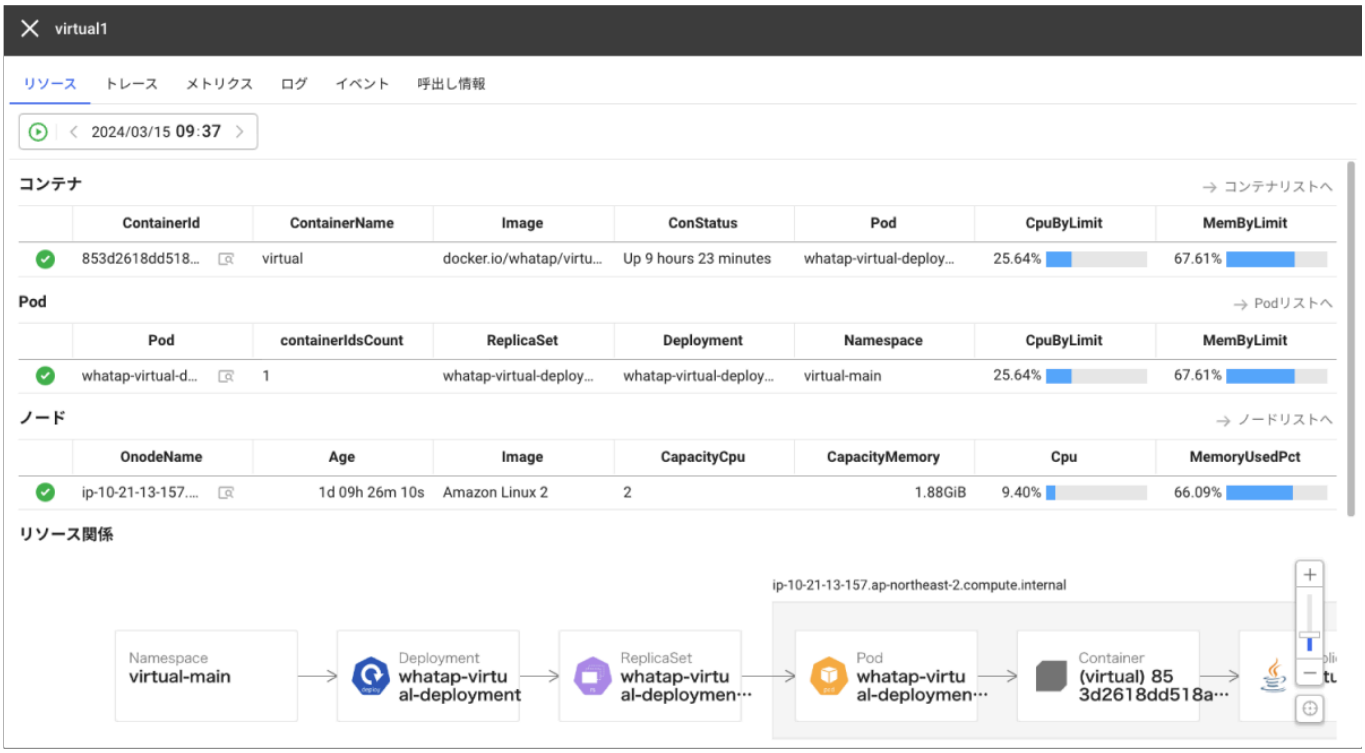
カラム選択オプションを活用する時、希望するアプリケーション一覧情報を選択して一覧を照会できます。



- 選択したカラム一覧は、メニューを再入力すると維持されます。
- リセットボタンを選択すると、定義済みの既定の主要指標カラムが選択されます。 この場合、既存のカラム選択は削除されます。
- カラムを選択した後、右下の適用ボタンをクリックして選択履歴を適用することができます。

詳細を確認

一覧で照会を希望するアプリケーションの🔍 詳細を見るアイコンを選択する時に、該当アプリケーションに関する詳細情報を確認できます。



詳細ビューのコンテナ要約分析、トレース、メトリクス、ログ、イベント、呼び出し情報タブの場合、コンテナマップの詳細ビューと同じです。

リソース

対象を選択した後、詳細表示の**リソース**タブから選択したリソースに関連するリソースの一覧と関係図を照会できます。

要約情報

選択したリソースに関連付けられているリソース一覧の要約情報を提供します。 リソース名の横にある詳細ビュー アイコンを選択すると、そのリソース一覧の画面に移動します。

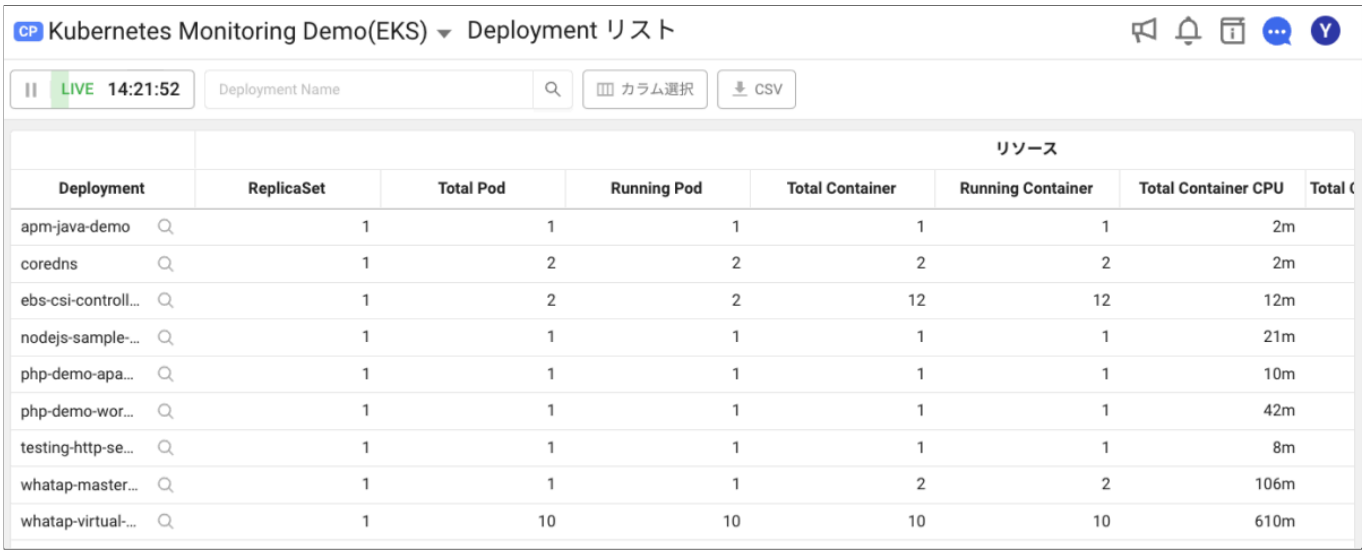
- **コンテナ**：当該コンテナのみフィルタリングされた状態の**コンテナリスト**に移動します。
- **Pod**：該当するPodのみフィルタリングされた**Podリスト**に移動します。
- **ノード**：該当ノード名が選択された状態の**ノード詳細情報**に移動します。

リソース関係図

リソース間の接続関係と状態を視覚的に確認できます。 一覧と同様に、Pod、コンテナ、アプリケーションにマウスオーバーすると、詳細指標を確認できます。 リソースを選択すると、そのリソース一覧ページに移動します。

Deploymentリスト

Deploymentのリストを照会します。 複数のPodで構成されたアプリケーションの全体的な運営状態をモニタリングし、システムの安定性と性能を維持するのに役立ちます。 Deploymentリスト内のカラムを選択する時に、そのカラムを基準にしてリストを整列して照会できます。



- 10秒サイクルでリアルタイムのデータを更新します。
- 検索、カラム整列、詳細を見る、_CSV_ファイルのダウンロードを提供します。

  CSV ファイル名、kubernetes_deployment_list_ {pcode} _ {timestamp} .csv

カラム選択

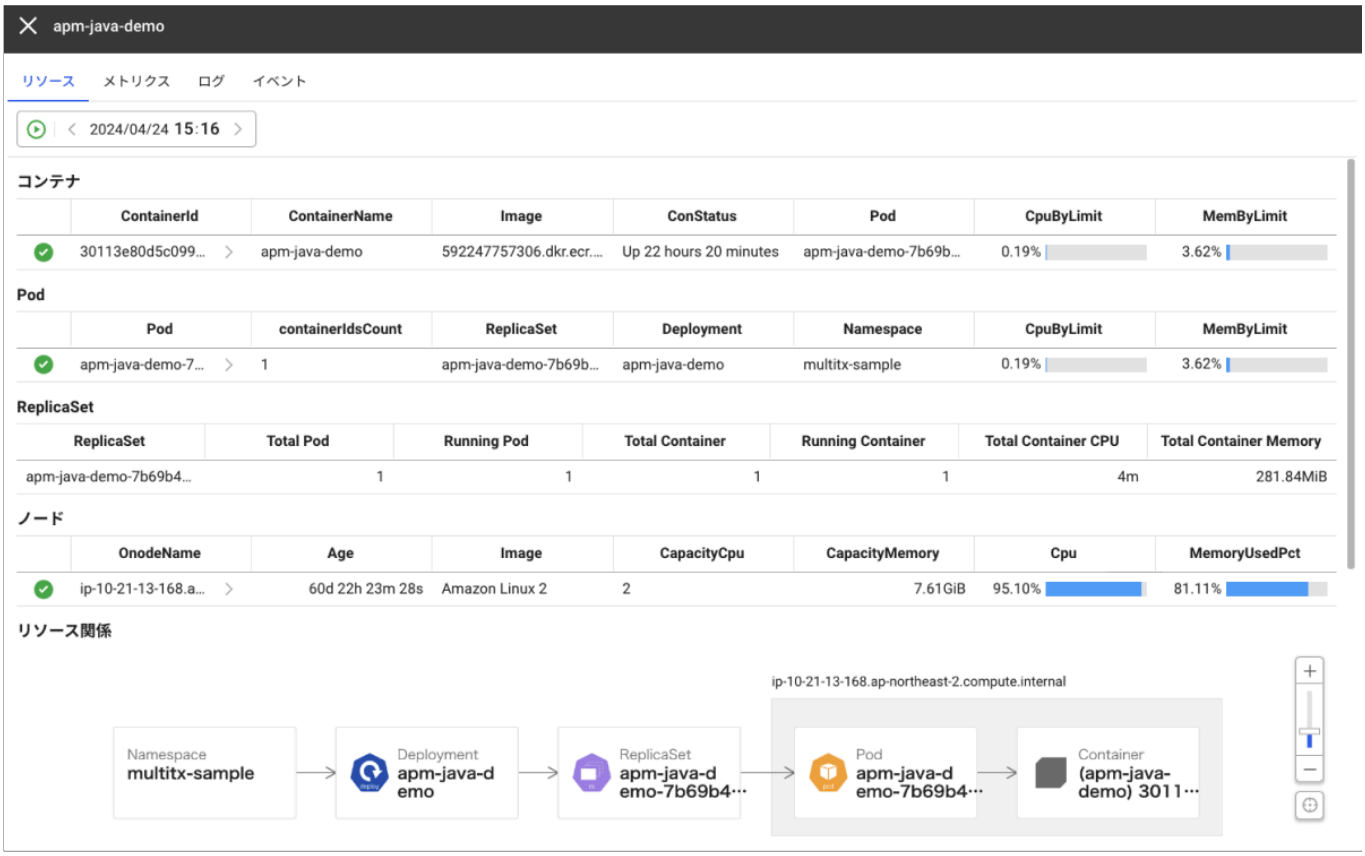
カラム選択オプションを活用する時、希望するDeployment情報を選択して一覧を照会できます。



- 選択したカラム一覧は、メニューを再入力すると維持されます。
- リセットボタンを選択すると、定義済みの既定の主要指標カラムが選択されます。 この場合、既存のカラム選択は削除されます。
- カラムを選択した後、右下の適用ボタンをクリックして選択履歴を適用することができます。

詳細を確認

一覧で照会を希望するDeploymentの 詳細を見るアイコンを選択する時に、該当Deploymentに関する詳細情報を確認できます。



詳細ビューのトレース、メトリクス、ログ、イベントタブの場合、コンテナマップの詳細ビューと同様です。

リソース

対象を選択した後、詳細表示の**リソース**タブから選択したリソースに関連するリソースの一覧と関係図を照会できます。

要約情報

選択したリソースに関連付けられているリソース一覧の要約情報を提供します。 リソース名の横にある詳細ビュー アイコンを選択すると、そのリソース一覧の画面に移動します。

- **コンテナ**：当該コンテナのみフィルタリングされた状態の**コンテナリスト**に移動します。

- **Pod**：該当するPodのみフィルタリングされた**Podリスト**に移動します。
- **ノード**：該当ノード名が選択された状態の**ノード詳細情報**に移動します。
- ReplicaSet：ReplicaSetを基準に実行中のPodとコンテナの要約情報を提供します。

リソース関係図

リソース間の接続関係と状態を視覚的に確認できます。 一覧と同様に、Pod、コンテナ、アプリケーションにマウスオーバーすると、詳細指標を確認できます。 リソースを選択すると、そのリソース一覧ページに移動します。

Service リスト

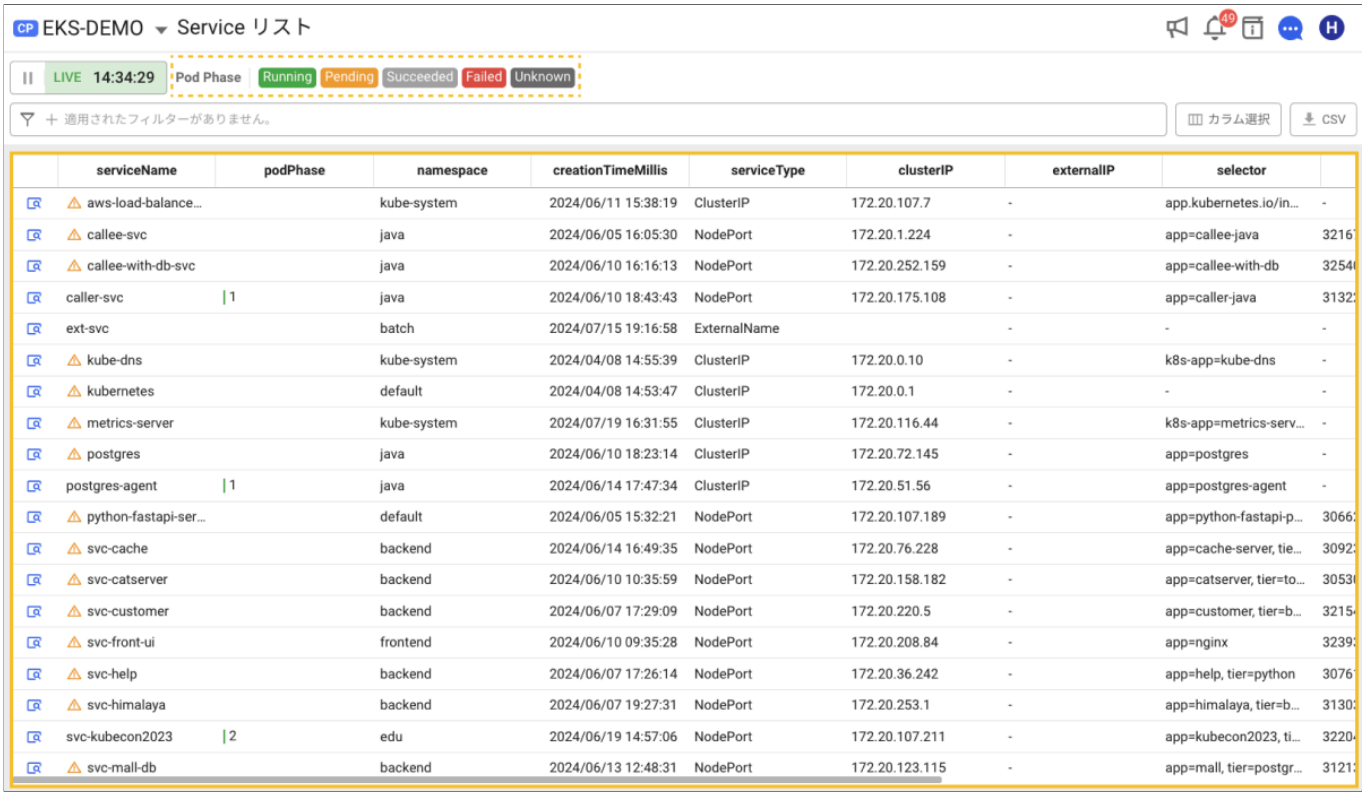
> WhaTap Kubernetesエージェント1.7.8バージョン以降が必要です。

Service リストを照会します。 WhaTap Kubernetesモニタリングの**Service リスト**これにより、クラスター内のネットワーク構成を容易に把握し、各サービスの構成と現状を効率的にモニタリングできます。

Kubernetesで**Service**は、クラスター内部と外部からPodに接続するためのオブジェクトです。 **Service リスト**メニューで各サービスの詳細構成を一目で確認でき、接続されたPodの状態をリアルタイムでモニタリングできます。 これにより、サービスとPod間の異常な接続状態を迅速に把握して解決できます。

基本画面ガイド

現在クラスターに存在する**Service リスト**情報を次のように確認できます。 照会時間を基準に、最近1分間収集されたService情報に基づいて一覧を表示します。 ネームスペース、ラベルセレクターなど様々な基準でリソースを区分・管理でき、サービスの多様な属性を明確に確認できます。



> **Pod Phase**
>
> **podPhase**カラムは、当該サービスに接続されたPodの状態別個数を表示します。 Podの状態は色で表記します。 Pod状態については、上段の**Pod Phase**案内を参照してください。

> リスト型メニューの**フィルター**、**カラム選択**案内は同じです。

フィルター

フィルターオプションを活用する時、希望するを選択して照会できます。 `and` 条件を使用すると、複数のフィルターが適用できます。

1. フィルター入力ウインドウで＋追加アイコンを選択すると、次のようにフィルターを追加するウィンドウが表示されます。

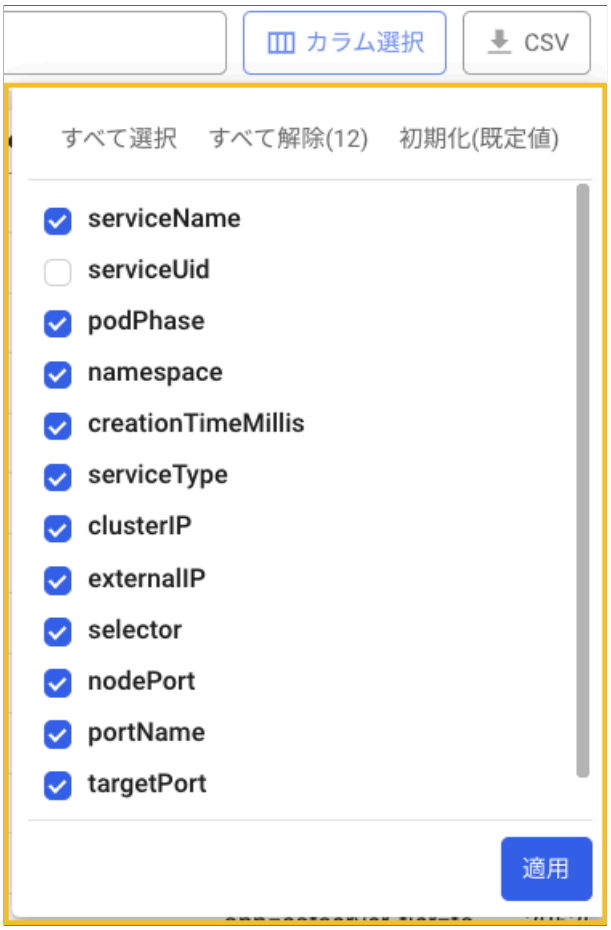


- フィルターキーと条件を選択します。
- 条件に合った値を選択します。

2. フィルター設定後、適用ボタンをクリックし、フィルターを適用します。
3. 複数のフィルターを追加するには、フィルター入力ウインドウまたはフィルターを追加するウィンドウで＋追加アイコンを選択します。

カラム選択

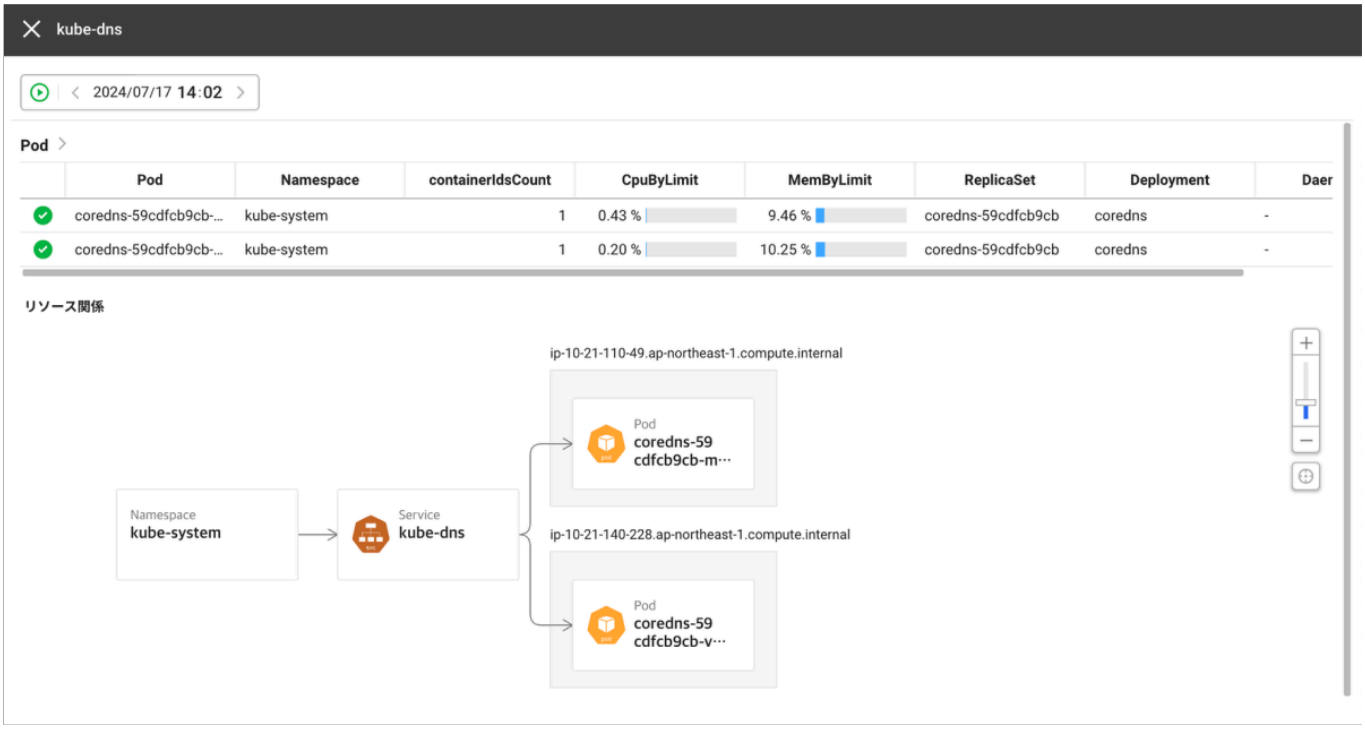
カラム選択オプションを活用する時、希望する情報を選択して一覧を照会できます。



- 選択したカラム一覧は、メニューを再入力すると維持されます。
- リセットボタンを選択すると、定義済みの既定の主要指標カラムが選択されます。 この場合、既存のカラム選択は削除されます。
- カラムを選択した後、右下の適用ボタンをクリックして選択履歴を適用することができます。

詳細を確認

リストから照会を希望するServiceの **詳細**アイコンを選択すると、該当Serviceに接続されたPodリストを確認できます。


kube-dns
2024/07/17 14:02
Pod
Pod
Namespace
containerIdsCount
CpuByLimit
MemByLimit
ReplicaSet
Deployment
Daer
coredns-59cdfcb9cb-...
kube-system
1
0.43 %
9.46 %
coredns-59cdfcb9cb
coredns
-
coredns-59cdfcb9cb-...
kube-system
1
0.20 %
10.25 %
coredns-59cdfcb9cb
coredns
-
リソース関係
ip-10-21-110-49.ap-northeast-1.compute.internal
Pod
coredns-59
cdfcb9cb-m···
Namespace
kube-system
Service
kube-dns
ip-10-21-140-228.ap-northeast-1.compute.internal
Pod
coredns-59
cdfcb9cb-v···

Ingress一覧

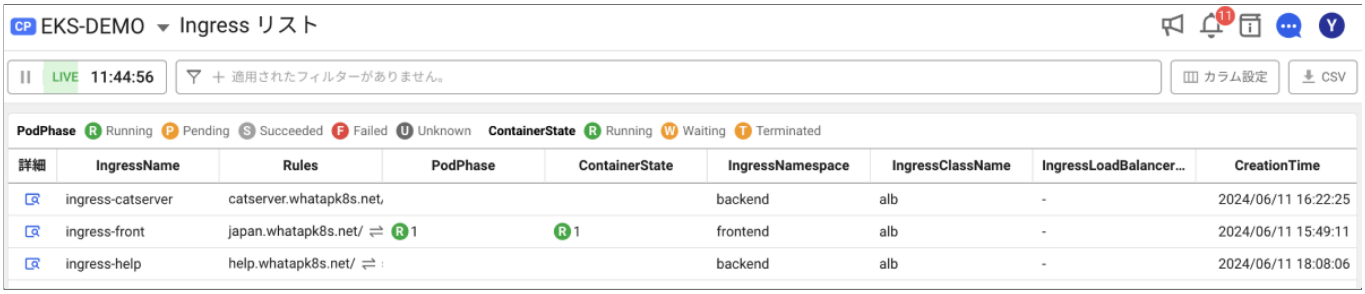
> WhaTap Kubernetesエージェント1.7.13バージョン以降が必要です。

Ingress リストを照会します。 クバネティス外部から流入する要請は、**Ingress**を通じてコンテナ内部のアプリケーションまで最終的にルーティングされ、Ingress設定によって特定のServiceやPodにつながります。 Ingressモニタリングは、クバネティスクラスターに入る外部トラフィックの流れを把握し、管理する上で重要な役割を果たします。

WhaTapの**Ingress リスト**は、Ingress別のルーティングポリシーを視覚化した一覧ビューを提供します。 これを通じて、各Ingressがどのようにトラフィックを分配しているかを一目で把握できます。

基本画面ガイド

Ingressポリシーを中心に主なIngress情報とそのポリシーに接続されたPodおよびコンテナの状態別個数を次のように照会できます。 一覧と詳細を見ることで、Ingressポリシー別の接続状態と主要オブジェクトの情報を総合的に調べることができ、また一覧の上段でPodとコンテナのステータスごとのアイコン情報を確認できます。



> リスト型メニューの**フィルター**、**カラム選択**案内は同じです。

フィルター

フィルターオプションを活用する時、希望するを選択して照会できます。 `and` 条件を使用すると、複数のフィルターが適用できます。

1. フィルター入力ウインドウで＋追加アイコンを選択すると、次のようにフィルターを追加するウィンドウが表示されます。



   - フィルターキーと条件を選択します。
   - 条件に合った値を選択します。
2. フィルター設定後、適用ボタンをクリックし、フィルターを適用します。
3. 複数のフィルターを追加するには、フィルター入力ウインドウまたはフィルターを追加するウィンドウで＋追加アイコンを選択します。

カラム選択

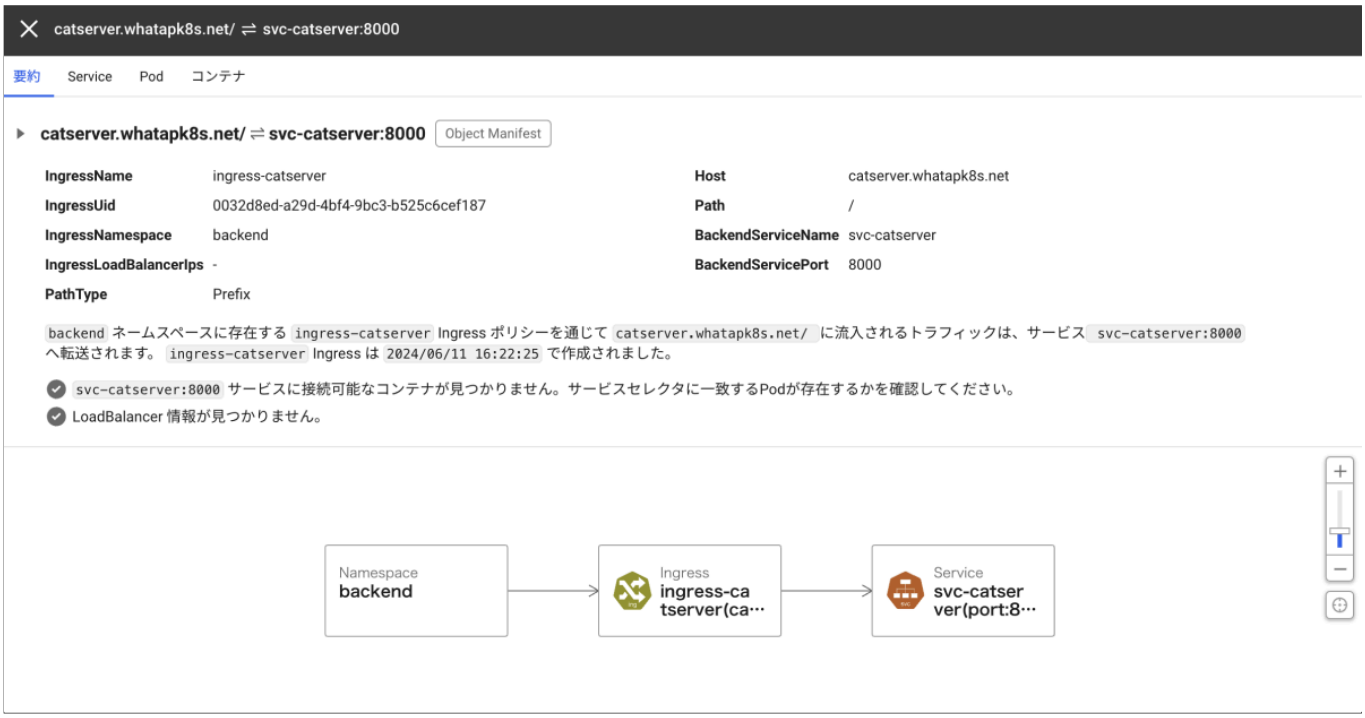
カラム選択オプションを活用する時、希望する情報を選択して一覧を照会できます。

カラム選択

- 選択したカラム一覧は、メニューを再入力すると維持されます。
- リセットボタンを選択すると、定義済みの既定の主要指標カラムが選択されます。 この場合、既存のカラム選択は削除されます。
- カラムを選択した後、右下の適用ボタンをクリックして選択履歴を適用することができます。

詳細を確認

一覧から照会を希望するIngressポリシー項目の**詳細**アイコンを選択すると、そのIngressポリシーに関する詳細情報を確認できます。



要約

Ingressの基本情報と選択したIngressポリシーに対する**要約**分析を提供します。 Ingressのネームスペース、接続されたホストとパスの設定、トラフィックが転送されるバックエンドサービスの情報などを確認できます。 これにより、外部トラフィックの流れとIngressポリシーの仕組みを確認できます。 また、このポリシーを使用して接続されている主要オブジェクトの関係図を視覚化することで、ポリシー内のリソース関係を一目で把握できます。

> ⓘ **要約**タブの上段でオブジェクトマニフェスト 情報を確認できます。

Service

選択したIngressポリシーに接続した**Service**一覧を提供します。 これを通じて、Ingressがどのようなサービスとつながっているかを確認できます。 各サービスの接続状態とサービスに関する詳細を照会できます。

Pod

選択したIngressポリシーに接続した**Pod**一覧を提供します。 接続したPodの状態と基本情報及び要約分析情報を確認できます。

コンテナ

選択したIngressポリシーに接続した**コンテナ**一覧を提供します。 Ingressポリシーを介したトラフィックが転送されるコンテナの状態と性能及び要約分析情報を確認できます。

PV/PVC リスト

> **要求事項**
>
> - WhaTap Kubernetesエージェント1.8.1バージョン以降が必要です。
> - Kubernetesクラスタープロジェクト(CP)の場合、該当メニューを照会できます。

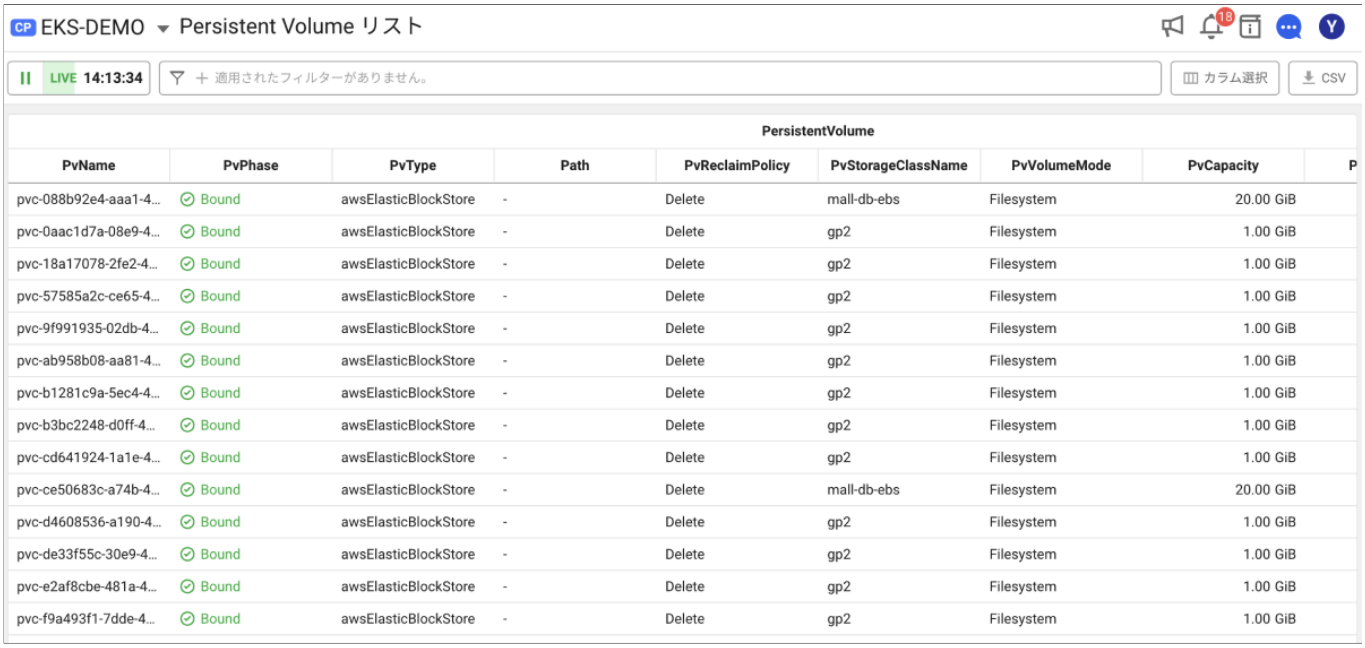
Persistent Volume(PV)はKubernetesクラスターで使用可能な物理的ストレージリソースを意味します。 **Persistent Volume Claim**(PVC)はPodのストレージ要件を宣言的に定義するオブジェクトです。 PVCはPVと接続され、これを通じてPodがデータを読み書きできるアクセス経路を提供します。

WhaTapのKubernetesは**Persistent Volume一覧**と**Persistent Volume Claim一覧**メニューからクラスターのストレージ状態をモニタリングできます。 このメニューは、リポジトリリソースの使用状態と接続関係を直観的に提供し、リポジトリ容やアクセスモード、イベント状態などの情報を一目で確認できます。

Persistent Volume リスト

ホーム画面 > プロジェクト選択 > **ストレージ** > Persistent Volume 一覧

Persistent Volume一覧メニューで現在クラスターに存在するPVを照会することができます。 照会時間を基準に、最近1分間収集されたPV情報に基づいて一覧を表示します。 PVCとの接続状態及びPVのバインディング状態を直観的に確認することができます。 また、該当PVと接続されたPVCの詳細情報を簡単に確認できます。



PersistentVolume

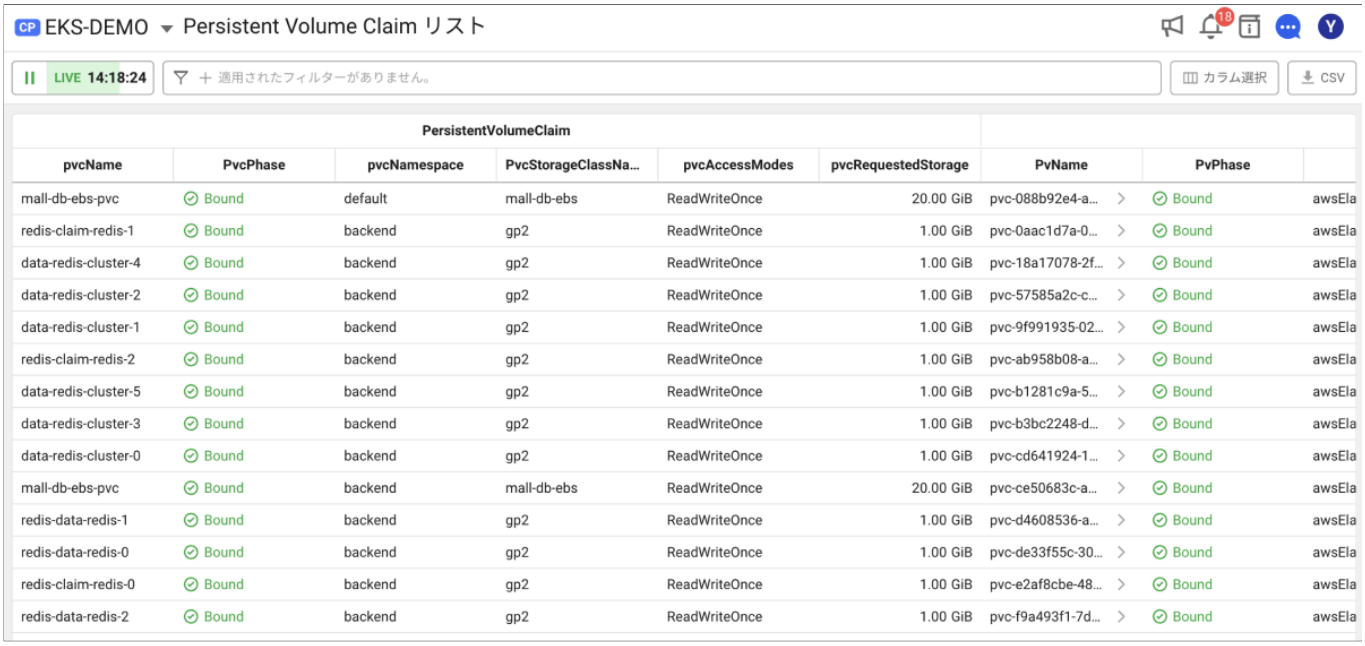
PvName	PvPhase	PvType	Path	PvReclaimPolicy	PvStorageClassName	PvVolumeMode	PvCapacity
pvc-088b92e4-aaa1-4...	Bound	awsElasticBlockStore	-	Delete	mall-db-ebs	Filesystem	20.00 GiB
pvc-0aac1d7a-08e9-4...	Bound	awsElasticBlockStore	-	Delete	gp2	Filesystem	1.00 GiB
pvc-18a17078-2fe2-4...	Bound	awsElasticBlockStore	-	Delete	gp2	Filesystem	1.00 GiB
pvc-57585a2c-ce65-4...	Bound	awsElasticBlockStore	-	Delete	gp2	Filesystem	1.00 GiB
pvc-9f991935-02db-4...	Bound	awsElasticBlockStore	-	Delete	gp2	Filesystem	1.00 GiB
pvc-ab958b08-aa81-4...	Bound	awsElasticBlockStore	-	Delete	gp2	Filesystem	1.00 GiB
pvc-b1281c9a-5ec4-4...	Bound	awsElasticBlockStore	-	Delete	gp2	Filesystem	1.00 GiB
pvc-b3bc2248-d0ff-4...	Bound	awsElasticBlockStore	-	Delete	gp2	Filesystem	1.00 GiB
pvc-cd641924-1a1e-4...	Bound	awsElasticBlockStore	-	Delete	gp2	Filesystem	1.00 GiB
pvc-ce50683c-a74b-4...	Bound	awsElasticBlockStore	-	Delete	mall-db-ebs	Filesystem	20.00 GiB
pvc-d4608536-a190-4...	Bound	awsElasticBlockStore	-	Delete	gp2	Filesystem	1.00 GiB
pvc-de33f55c-30e9-4...	Bound	awsElasticBlockStore	-	Delete	gp2	Filesystem	1.00 GiB
pvc-e2af8cbe-481a-4...	Bound	awsElasticBlockStore	-	Delete	gp2	Filesystem	1.00 GiB
pvc-f9a493f1-7dde-4...	Bound	awsElasticBlockStore	-	Delete	gp2	Filesystem	1.00 GiB

- **CSV**ボタンをクリックするとCSVファイルをダウンロードできます。

Persistent Volume Claim リスト

ホーム画面 > プロジェクト選択 > **ストレージ** > **Persistent Volume Claim 一覧**

Persistent Volume Claim一覧メニューで現在クラスターに存在するPVCを照会することができます。 照会時間を基準に、最近1分間収集されたPVC情報に基づいて一覧を表示します。 PVとの接続状態およびPVCのバインディング状態を直観的に確認できます。 また、該当PVCと接続されたPV詳細情報を簡単に確認することができます。



PersistentVolumeClaim

pvcName	PvcPhase	pvcNamespace	PvcStorageClassNa...	pvcAccessModes	pvcRequestedStorage	PvName	PvPhase	
mall-db-ebs-pvc	Bound	default	mall-db-ebs	ReadWriteOnce	20.00 GiB	pvc-088b92e4-a...	Bound	awsEla
redis-claim-redis-1	Bound	backend	gp2	ReadWriteOnce	1.00 GiB	pvc-0aac1d7a-0...	Bound	awsEla
data-redis-cluster-4	Bound	backend	gp2	ReadWriteOnce	1.00 GiB	pvc-18a17078-2f...	Bound	awsEla
data-redis-cluster-2	Bound	backend	gp2	ReadWriteOnce	1.00 GiB	pvc-57585a2c-c...	Bound	awsEla
data-redis-cluster-1	Bound	backend	gp2	ReadWriteOnce	1.00 GiB	pvc-9f991935-02...	Bound	awsEla
redis-claim-redis-2	Bound	backend	gp2	ReadWriteOnce	1.00 GiB	pvc-ab958b08-a...	Bound	awsEla
data-redis-cluster-5	Bound	backend	gp2	ReadWriteOnce	1.00 GiB	pvc-b1281c9a-5...	Bound	awsEla
data-redis-cluster-3	Bound	backend	gp2	ReadWriteOnce	1.00 GiB	pvc-b3bc2248-d...	Bound	awsEla
data-redis-cluster-0	Bound	backend	gp2	ReadWriteOnce	1.00 GiB	pvc-cd641924-1...	Bound	awsEla
mall-db-ebs-pvc	Bound	backend	mall-db-ebs	ReadWriteOnce	20.00 GiB	pvc-ce50683c-a...	Bound	awsEla
redis-data-redis-1	Bound	backend	gp2	ReadWriteOnce	1.00 GiB	pvc-d4608536-a...	Bound	awsEla
redis-data-redis-0	Bound	backend	gp2	ReadWriteOnce	1.00 GiB	pvc-de33f55c-30...	Bound	awsEla
redis-claim-redis-0	Bound	backend	gp2	ReadWriteOnce	1.00 GiB	pvc-e2af8cbe-48...	Bound	awsEla
redis-data-redis-2	Bound	backend	gp2	ReadWriteOnce	1.00 GiB	pvc-f9a493f1-7d...	Bound	awsEla

- **CSV**ボタンをクリックするとCSVファイルをダウンロードできます。
- **PvName**カラムの右側**>**ボタンをクリックすると**Persistent Volume一覧**メニューに移動します。

フィルター

フィルターオプションを活用する時、希望するを選択して照会できます。 `and` 条件を使用すると、複数のフィルターが適用できます。

1. フィルター入力ウインドウで＋追加アイコンを選択すると、次のようにフィルターを追加するウィンドウが表示されます。



- フィルターキーと条件を選択します。
- 条件に合った値を選択します。

2. フィルター設定後、適用ボタンをクリックし、フィルターを適用します。
3. 複数のフィルターを追加するには、フィルター入力ウインドウまたはフィルターを追加するウィンドウで＋追加アイコンを選択します。

カラム

グループ	カラム名	説明
PersistentVolume	**PvUid**	Persistent Volumeの一意識別子です。
	PvName	Persistent Volumeの名前です。
	PvPhase	Persistent Volumeの現在状態を表示します。
	PvType	Persistent Volumeのストレージタイプです。
	Path	Persistent Volumeがマウントされた経路を意味します。
	PvReclaimPolicy	Persistent Volumeの返還ポリシーを意味します。
	PvStorageClassName	該当PVに割り当てられたストレージクラス名です。
	PvVolumeMode	Persistent Volumeのボリュームモードプロパティを意味します。
	PvCapacity	Persistent Volumeの総保存容量を表示します。
	PvMessage	現在、Persistent Volumeが特定のPhaseを持つようになった原因を説明するメッセージです。
	PvReason	Persistent Volumeに関わるエラーを表すキーワードで、問題の原因や現在の状態を簡単に表示します。

グループ	カラム名	説明
PersistentVolumeClaim	**PvcUid**	Persistent Volume Claimの一意識別子です。
	PvcName (PVC)	Persistent Volume Claimの名前です。
	PvcNamespace	Persistent Volume Claimが属するネームスペースを意味します。
	PvcStorageClassName	Persistent Volume Claimに設定されたストレージのクラス名です。
	PvcAccessModes	Persistent Volume Claimのアクセスモードを表示します。
	PvcRequestedStorage	Persistent Volume Claimが要求したストレージ容量を示します。

- Persistent Volumeの返還ポリシーに関する詳細は次の文書を参照してください。

  Retain, **Delete**

- Persistent Volumeのボリュームモードの詳細は次の文書を参照してください。

  Filesystem, **Block**

- Persistent Volume Claimアクセスモードの詳細は次の文書を参照してください。

  ReadWriteOnce, **ReadOnlyMany**, **ReadWriteMany**

Phase

PvPhase

PvPhase項目はPersistent Volumeの状態を表示します。

Phase	説明
Available	PVがまだどのPVCともバインディングされていない状態です。
Bound	PVが特定のPVCにバインドされて使用中の状態です。
Released	PVCは削除されましたが、PVがまだ返却されていない状態です。
Failed	PVが自動返却に失敗し、使用できない状態です。

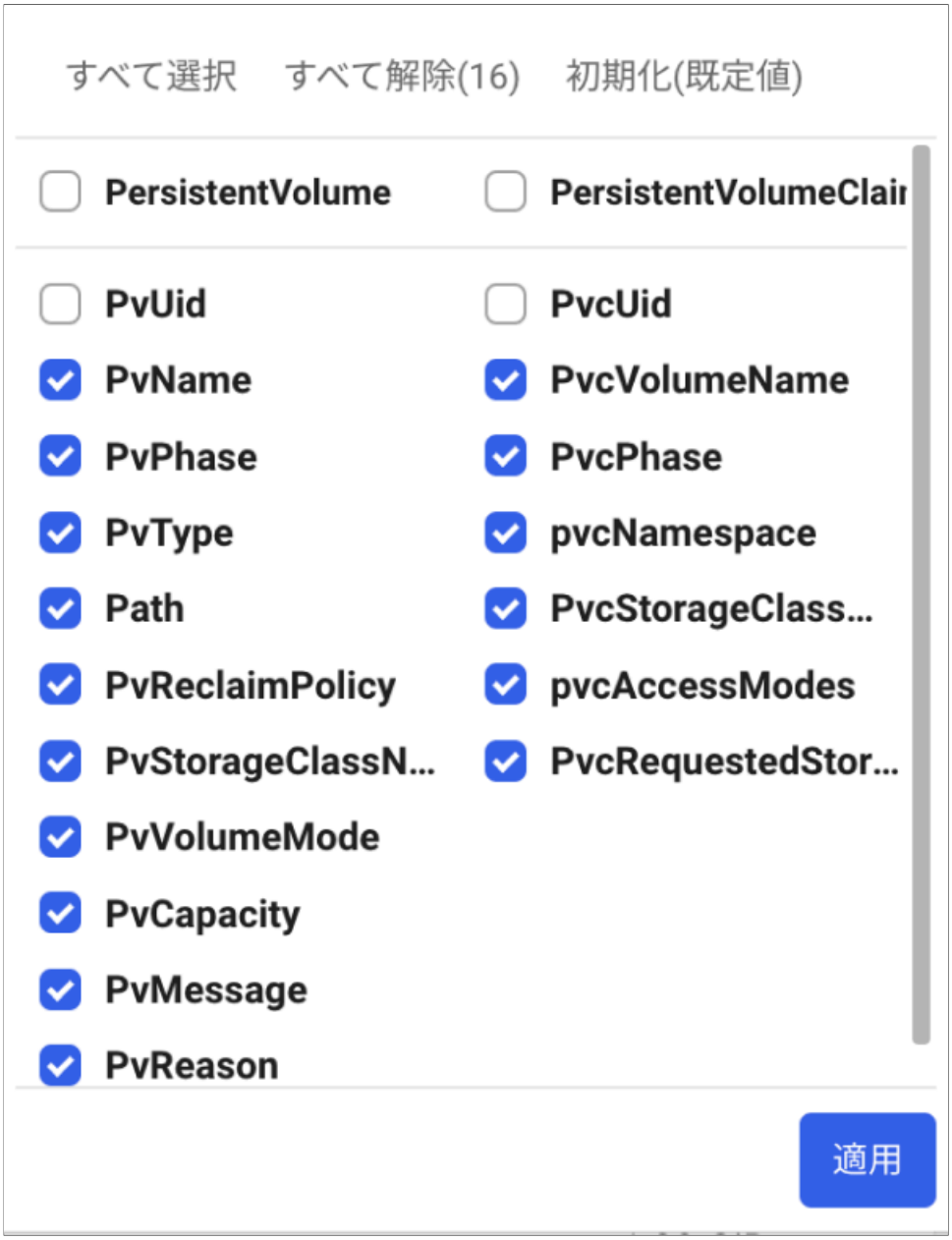
PvcPhase

PvcPhase項目はPersistent Volume Claimの状態を表示します。

Phase	説明
Pending	PVCがPVとまだバインディングされていないか、PV割り当て待機中の状態です。
Bound	PVCがPVとバインディングされて使用中の状態です。
Lost	接続されたPVが削除されたり、非アクティブになったため使用できない状態です。

カラム選択

カラム選択オプションを活用する時、希望する情報を選択して一覧を照会できます。



- 選択したカラム一覧は、メニューを再入力すると維持されます。
- リセットボタンを選択すると、定義済みの既定の主要指標カラムが選択されます。 この場合、既存のカラム選択は削除されます。
- カラムを選択した後、右下の適用ボタンをクリックして選択履歴を適用することができます。

PersistentVolumeまたは**PersistentVolumeClaim**のカラムグループを一括選択および解除することができます。

kube-apiserver ダッシュボード

> **要求事項**
>
> - WhaTap Kubernetesエージェント1.5.6バージョン以降が必要です。
> - Kubernetesクラスタープロジェクト(CP)の場合、該当メニューを照会できます。

kube-apiserverは、**コントロールプレーン**(Control Plane) 内外部のリクエストを処理するコンポーネントです。**kube-apiserver ダッシュボード**メニューを通じてkube-apiserverの状態と性能メトリックを収集した後、kube-apiserverのレスポンスタイム、処理量、リクエスト失敗などを追跡し、クラスターの可用性と性能を簡単にモニタリングできます。

基本画面ガイド



kube-apiserver ダッシュボードメニューは最近1時間のkube-apiserver主要指標を1分周期で反映します。 以下の指標を確認することができます。 指標詳細は、下段のメトリクス案内を参考にしてください。

- クラスタの負荷と使用量をモニタリングするために、メソッド、リソース、リスポンスコードごとのリクエスト回数の状況と推移を提供します。
- クラスターの負荷状態を把握できるように、現在処理中のAPIリクエスト数を提供します。
- 生成されたOSスレッドと現在存在するゴルーチンの数を提供します。

kube-apiserverモニタリング活性化

WhaTap Kubernetesエージェントのインストール時にダウンロードしたyamlファイルを以下の案内に従って修正し、kube-apiserverモニタリングを活性化します。

1. `whatap-master-agent` deploymentのspec下位にkube-apiserverモニタリングのためのエージェントコンテナspecを追加します。

```
containers:
 - name: whatap-control-plane-helper
   image: whatap/kube_mon
   imagePullPolicy: Always
   ports:
    - containerPort: 9496
   command: ["/data/agent/master/whatap_control_plane_helper"]
   args: ["-collect_control_plane_monitoring_enabled=true"]
```

次の例を参照してください。

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: whatap-master-agent
 namespace: whatap-monitoring
spec:
 replicas: 1
 selector:
  matchLabels:
   name: whatap-master-agent
```

```
template:
 metadata:
  labels:
   name: whatap-master-agent
 spec:
  containers:
   - name: whatap-control-plane-helper
     image: whatap/kube_mon
     imagePullPolicy: Always
     ports:
      - containerPort: 9496
     command: ["/data/agent/master/whatap_control_plane_helper"]
     args: ["-collect_control_plane_monitoring_enabled=true"]
   - name: whatap-master-agent
     image: whatap/kube_mon
     imagePullPolicy: Always
     resources:
      requests:
       memory: 300Mi
       cpu: 100m
      limits:
       memory: 350Mi
       cpu: 200m
     ports:
      - containerPort: 6600
     command: ['/bin/entrypoint.sh']
     env:
      - name: WHATAP_LICENSE
        value: "xxxxxxxx-12345xxxxxx-x00e000000xxx8"
      - name: WHATAP_HOST
        value: "13.124.11.223/13.209.172.35"
      - name: WHATAP_PORT
        value: "6600"
      - name: WHATP_MEM_LIMIT
        valueFrom:
         resourceFieldRef:
          containerName: whatap-master-agent
          resource: limits.memory
     volumeMounts:
```

```
        - name: start-script-volume
          mountPath: /bin/entrypoint.sh
          readOnly: true
          subPath: entrypoint.sh
        - mountPath: /whatap_conf
          name: whatap-config-volume
    volumes:
      - name: start-script-volume
        configMap:
          defaultMode: 0700
          name: master-start-script
      - name: whatap-config-volume
        emptyDir: {}
    serviceAccount: whatap
```

2. `whatap-master-agent` deploymentの `whatap-master-agent` コンテナspecのenv設定に次の内容を追加します。

```
env:
  - name: collect_control_plane_monitoring_enabled
    value: "true"
```

次の例を参照してください。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: whatap-master-agent
  namespace: whatap-monitoring
spec:
  replicas: 1
  selector:
    matchLabels:
      name: whatap-master-agent
  template:
    metadata:
      labels:
        name: whatap-master-agent
```

```
spec:
 containers:
  - name: whatap-control-plane-helper
    image: whatap/kube_mon
    imagePullPolicy: Always
    ports:
     - containerPort: 9496
    command: ["/data/agent/master/whatap_control_plane_helper"]
    args: ["-collect_control_plane_monitoring_enabled=true"]
  - name: whatap-master-agent
    image: whatap/kube_mon
    imagePullPolicy: Always
    resources:
     requests:
      memory: 300Mi
      cpu: 100m
     limits:
      memory: 350Mi
      cpu: 200m
    ports:
     - containerPort: 6600
    command: ['/bin/entrypoint.sh']
    env:
     - name: collect_control_plane_monitoring_enabled
       value: "true"
     - name: WHATAP_LICENSE
       value: "xxxxxxxx-12345xxxxxx-x00e000000xxx8"
     - name: WHATAP_HOST
       value: "13.124.11.223/13.209.172.35"
     - name: WHATAP_PORT
       value: "6600"
     - name: WHATP_MEM_LIMIT
       valueFrom:
        resourceFieldRef:
          containerName: whatap-master-agent
          resource: limits.memory
    volumeMounts:
     - name: start-script-volume
       mountPath: /bin/entrypoint.sh
```

```
          readOnly: true
          subPath: entrypoint.sh
        - mountPath: /whatap_conf
          name: whatap-config-volume
    volumes:
     - name: start-script-volume
       configMap:
        defaultMode: 0700
        name: master-start-script
     - name: whatap-config-volume
       emptyDir: {}
    serviceAccount: whatap
```

3. コントロールプレーンモニタリング性能指標を収集するために `whatap` ClusterRoleに次の権限を追加します。

```
rules:
 - nonResourceURLs: ["/metrics"]
   verbs: ["*"]
```

次の例を参照してください。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 annotations:
   rbac.authorization.kubernetes.io/autoupdate: "true"
 name: whatap
 namespace: whatap-monitoring
rules:
 - apiGroups: ["*"]
   resources: ["namespaces", "pods", "pods/log", "configmaps","services",
"endpoints","daemonsets","ingresses","deployments","nodes","persistentvolumes","persistentvolumeclaims",
"events", "replicasets", "roles", "rolebindings", "clusterroles", "clusterrolebindings", "jobs", "cronjobs",
"statefulsets", "serviceaccounts", "configmaps", "storageclasses", "horizontalpodautoscalers" ]
   verbs: ["get","list","watch"]
 - apiGroups: [""]
   resources: ["pods/exec"]
```

```
  verbs: ["create"]
- apiGroups: [""]
  resources: ["configmaps"]
  verbs: ["*"]
- nonResourceURLs: ["/metrics"]
  verbs: ["*"]
---
```

ⓘ `kubectl apply` または `kubectl edit` コマンドを通じてkube-apiserverモニタリングを活性化できます。

kube-apiserverメトリクス

分類	メトリクス	説明
API GETリクエスト	1分当たりのGETリクエスト状況	kube-apiserverのGETリクエストに対する1分当たりのリクエスト数
	1秒当たりのGETリクエスト数の推移	kube-apiserverのGETリクエストに対する1秒当たりのリクエスト数の推移
	Resource & Subresource別の1分当たりのGETリクエスト数の推移	kube-apiserverに入ってきたGETリクエストのResourceとSubresourceごとの1分当たりのリクエスト数の推移
API POSTリクエスト	1分あたりのPOSTリクエストの状況	kube-apiserverのPOSTリクエストに対する1分当たりのリクエスト数
	1秒当たりのPOSTリクエスト数の推移	kube-apiserverのPOSTリクエストに対する1秒当たりのリクエスト数の推移
	Resource & Subresource別のPOST1分当たりのリクエスト数の推移	kube-apiserverに入ってきたPOSTリクエストに対してResourceとSubresourceごとにグループ化した1分当たりのリクエスト数の推移

分類	メトリクス	説明
APIレスポンスコード別のリクエスト	[2xx] 1分あたりのレスポンスのリクエスト状況	kube-apiserverリクエストのうち、200以上300未満のレスポンスコードが出されたリクエスト数に対する1分当たりの現況
	[2xx] 1分当たりのレスポンスコードごとのリクエスト数の推移	kube-apiserverリクエストのうち、200以上300未満のレスポンスコードが出されたリクエスト数に対するレスポンスコードごとの1分当たりの現況
	[2xx] 1分当たりのResource & Subresourceごとのリクエスト数の推移	kube-apiserverリクエストのうち、200以上300未満のレスポンスコードが出されたリクエストに対しResourceとSubresourceごとにグループ化した1分当たりのリクエスト数の推移
	[3xx] 1分あたりのレスポンスのリクエスト状況	kube-apiserverリクエストのうち、300以上400未満のレスポンスコードが出されたリクエスト数に対する1分当たりの現況
	[3xx] 1分当たりのレスポンスコードごとのリクエスト数の推移	kube-apiserverリクエストのうち、300以上400未満のレスポンスコードが出されたリクエスト数に対するレスポンスコードごとの1分当たりの現況
	[3xx] 1分当たりのResource & Subresourceごとのリクエスト数の推移	kube-apiserverリクエストのうち、300以上400未満のレスポンスコードが出されたリクエストに対しResourceとSubresourceごとにグループ化した1分当たりのリクエスト数の推移
	[400~] 1分あたりのレスポンスのリクエスト状況	kube-apiserverリクエストのうち、400以上のレスポンスコードが出されたリクエスト数に対する1分当たりの現況
	[400~] 1分当たりのレスポンスコードごとのリクエスト数の推移	kube-apiserverリクエストのうち、400以上のレスポンスコードが出されたリクエスト数に対するレスポンスコードごとの1分当たりの現況
	[400~] 1分当たりのResource & Subresourceごとのリクエスト数の推移	kube-apiserverリクエストのうち、400以上のレスポンスコードが出されたリクエストに対しResourceとSubresourceごとにグループ化した1分当たりのリクエスト数の推移
Go Metrics	Goルーチン数	現在存在するGoルーチンの数
	スレッド数	現在存在するOSスレッド数

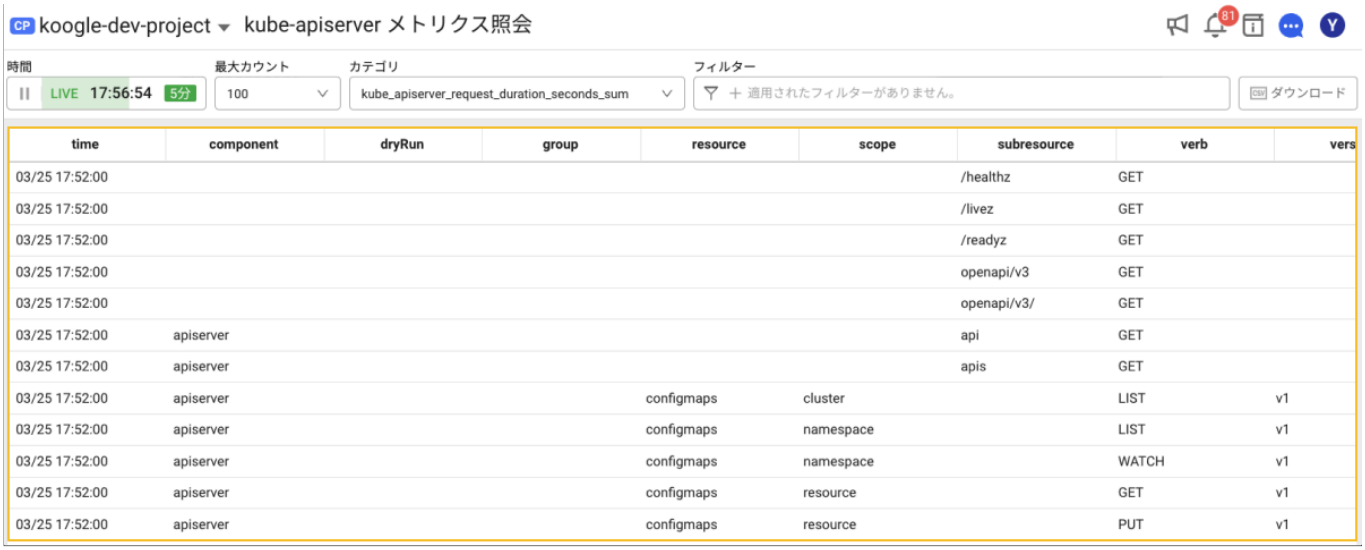
分類	メトリクス	説明
現在実行中のリクエスト数(1秒)	直近1秒以内に実行中のリクエスト数	データ収集時にkube-apiserverで1秒以内に行われていたリクエスト数

kube-apiserverメトリクス照会

> **要求事項**
>
> - WhaTap Kubernetesエージェント1.5.6バージョン以降が必要です。
> - Kubernetesクラスタープロジェクト(CP)の場合、該当メニューを照会できます。

kube-apiserverは、**コントロールプレーン**(Control Plane) 内外部のリクエストを処理するコンポーネントです。**kube-apiserver メトリクス照会**メニューからkube-apiserverで収集される指標の原本データ整合性(Data Consistency)が確認できます。

基本画面ガイド



time	component	dryRun	group	resource	scope	subresource	verb	vers
03/25 17:52:00						/healthz	GET	
03/25 17:52:00						/livez	GET	
03/25 17:52:00						/readyz	GET	
03/25 17:52:00						openapi/v3	GET	
03/25 17:52:00						openapi/v3/	GET	
03/25 17:52:00	apiserver					api	GET	
03/25 17:52:00	apiserver					apis	GET	
03/25 17:52:00	apiserver			configmaps	cluster		LIST	v1
03/25 17:52:00	apiserver			configmaps	namespace		LIST	v1
03/25 17:52:00	apiserver			configmaps	namespace		WATCH	v1
03/25 17:52:00	apiserver			configmaps	resource		GET	v1
03/25 17:52:00	apiserver			configmaps	resource		PUT	v1

kube-apiserver メトリクス照会メニューは、kube-apiserver対象に収集したモニタリング原本データを**カテゴリ**別に照会する機能を提供します。 最近5分間に収集したデータが1分おきに反映されます。

- 上段左のタイムセレクターを通じて、過去の特定の時点のデータを照会できます。 過去のデータ照会時に希望するオプションを選択した後、**検索**アイコンをクリックします。
- 照会する**最大カウント**を設定できます。

- メトリクスをフィルタリングして、ソースデータを必要に応じて加工できます。
- 上段右側の**ダウンロード**アイコンを選択し、希望するデータを_CSV_形式のファイルでダウンロードできます。

フィルター

フィルターオプションを活用する時、希望するを選択して照会できます。 `and` 条件を使用すると、複数のフィルターが適用できます。

1. フィルター入力ウインドウで＋追加アイコンを選択すると、次のようにフィルターを追加するウィンドウが表示されます。



   - フィルターキーと条件を選択します。
   - 条件に合った値を選択します。
2. フィルター設定後、適用ボタンをクリックし、フィルターを適用します。
3. 複数のフィルターを追加するには、フィルター入力ウインドウまたはフィルターを追加するウィンドウで＋追加アイコンを選択します。

kube-apiserverモニタリング活性化

WhaTap Kubernetesエージェントのインストール時にダウンロードしたyamlファイルを以下の案内に従って修正し、kube-apiserverモニタリングを活性化します。

1. `whatap-master-agent` deploymentのspec下位にkube-apiserverモニタリングのためのエージェントコンテナspecを追加します。

```
containers:
 - name: whatap-control-plane-helper
   image: whatap/kube_mon
   imagePullPolicy: Always
   ports:
    - containerPort: 9496
   command: ["/data/agent/master/whatap_control_plane_helper"]
   args: ["-collect_control_plane_monitoring_enabled=true"]
```

次の例を参照してください。

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: whatap-master-agent
 namespace: whatap-monitoring
spec:
 replicas: 1
 selector:
  matchLabels:
   name: whatap-master-agent
 template:
  metadata:
   labels:
    name: whatap-master-agent
  spec:
   containers:
    - name: whatap-control-plane-helper
      image: whatap/kube_mon
      imagePullPolicy: Always
      ports:
       - containerPort: 9496
      command: ["/data/agent/master/whatap_control_plane_helper"]
      args: ["-collect_control_plane_monitoring_enabled=true"]
    - name: whatap-master-agent
      image: whatap/kube_mon
      imagePullPolicy: Always
      resources:
       requests:
```

```
        memory: 300Mi
        cpu: 100m
      limits:
        memory: 350Mi
        cpu: 200m
    ports:
     - containerPort: 6600
    command: ['/bin/entrypoint.sh']
    env:
     - name: WHATAP_LICENSE
       value: "xxxxxxxx-12345xxxxxx-x00e000000xxx8"
     - name: WHATAP_HOST
       value: "13.124.11.223/13.209.172.35"
     - name: WHATAP_PORT
       value: "6600"
     - name: WHATP_MEM_LIMIT
       valueFrom:
         resourceFieldRef:
           containerName: whatap-master-agent
           resource: limits.memory
    volumeMounts:
     - name: start-script-volume
       mountPath: /bin/entrypoint.sh
       readOnly: true
       subPath: entrypoint.sh
     - mountPath: /whatap_conf
       name: whatap-config-volume
volumes:
  - name: start-script-volume
    configMap:
      defaultMode: 0700
      name: master-start-script
  - name: whatap-config-volume
    emptyDir: {}
serviceAccount: whatap
```

2. `whatap-master-agent` deploymentの `whatap-master-agent` コンテナspecのenv設定に次の内容を追加します。

```
env:
 - name: collect_control_plane_monitoring_enabled
   value: "true"
```

次の例を参照してください。

```
apiVersion: apps/v1
kind: Deployment
metadata:
 name: whatap-master-agent
 namespace: whatap-monitoring
spec:
 replicas: 1
 selector:
  matchLabels:
   name: whatap-master-agent
 template:
  metadata:
   labels:
    name: whatap-master-agent
  spec:
   containers:
    - name: whatap-control-plane-helper
      image: whatap/kube_mon
      imagePullPolicy: Always
      ports:
       - containerPort: 9496
      command: ["/data/agent/master/whatap_control_plane_helper"]
      args: ["-collect_control_plane_monitoring_enabled=true"]
    - name: whatap-master-agent
      image: whatap/kube_mon
      imagePullPolicy: Always
      resources:
       requests:
        memory: 300Mi
        cpu: 100m
       limits:
        memory: 350Mi
        cpu: 200m
```

```
      ports:
       - containerPort: 6600
      command: ['/bin/entrypoint.sh']
      env:
       - name: collect_control_plane_monitoring_enabled
         value: "true"
       - name: WHATAP_LICENSE
         value: "xxxxxxxx-12345xxxxxx-x00e000000xxx8"
       - name: WHATAP_HOST
         value: "13.124.11.223/13.209.172.35"
       - name: WHATAP_PORT
         value: "6600"
       - name: WHATP_MEM_LIMIT
         valueFrom:
           resourceFieldRef:
             containerName: whatap-master-agent
             resource: limits.memory
      volumeMounts:
       - name: start-script-volume
         mountPath: /bin/entrypoint.sh
         readOnly: true
         subPath: entrypoint.sh
       - mountPath: /whatap_conf
         name: whatap-config-volume
    volumes:
     - name: start-script-volume
       configMap:
         defaultMode: 0700
         name: master-start-script
     - name: whatap-config-volume
       emptyDir: {}
    serviceAccount: whatap
```

3. コントロールプレーンモニタリング性能指標を収集するために `whatap` ClusterRoleに次の権限を追加します。

```
rules:
 - nonResourceURLs: ["/metrics"]
```

```
  verbs: ["*"]
```

次の例を参照してください。

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
 annotations:
  rbac.authorization.kubernetes.io/autoupdate: "true"
 name: whatap
 namespace: whatap-monitoring
rules:
 - apiGroups: ["*"]
  resources: ["namespaces", "pods", "pods/log", "configmaps","services",
"endpoints","daemonsets","ingresses","deployments","nodes","persistentvolumes","persistentvolumeclaims",
"events", "replicasets", "roles", "rolebindings", "clusterroles", "clusterrolebindings", "jobs", "cronjobs",
"statefulsets", "serviceaccounts", "configmaps", "storageclasses", "horizontalpodautoscalers" ]
  verbs: ["get","list","watch"]
 - apiGroups: [""]
  resources: ["pods/exec"]
  verbs: ["create"]
 - apiGroups: [""]
  resources: ["configmaps"]
  verbs: ["*"]
 - nonResourceURLs: ["/metrics"]
  verbs: ["*"]
---
```

ⓘ `kubectl apply` または `kubectl edit` コマンドを通じてkube-apiserverモニタリングを活性化できます。

kube-apiserverメトリクス

分類	メトリクス	説明
API GETリクエスト	1分当たりのGETリクエスト状況	kube-apiserverのGETリクエストに対する1分当たりのリクエスト数
	1秒当たりのGETリクエスト数の推移	kube-apiserverのGETリクエストに対する1秒当たりのリクエスト数の推移
	Resource & Subresource別の1分当たりのGETリクエスト数の推移	kube-apiserverに入ってきたGETリクエストのResourceとSubresourceごとの1分当たりのリクエスト数の推移
API POSTリクエスト	1分あたりのPOSTリクエストの状況	kube-apiserverのPOSTリクエストに対する1分当たりのリクエスト数
	1秒当たりのPOSTリクエスト数の推移	kube-apiserverのPOSTリクエストに対する1秒当たりのリクエスト数の推移
	Resource & Subresource別のPOST1分当たりのリクエスト数の推移	kube-apiserverに入ってきたPOSTリクエストに対してResourceとSubresourceごとにグループ化した1分当たりのリクエスト数の推移
APIレスポンスコード別のリクエスト	[2xx] 1分あたりのレスポンスのリクエスト状況	kube-apiserverリクエストのうち、200以上300未満のレスポンスコードが出されたリクエスト数に対する1分当たりの現況
	[2xx] 1分当たりのレスポンスコードごとのリクエスト数の推移	kube-apiserverリクエストのうち、200以上300未満のレスポンスコードが出されたリクエスト数に対するレスポンスコードごとの1分当たりの現況
	[2xx] 1分当たりのResource & Subresourceごとのリクエスト数の推移	kube-apiserverリクエストのうち、200以上300未満のレスポンスコードが出されたリクエストに対しResourceとSubresourceごとにグループ化した1分当たりのリクエスト数の推移
	[3xx] 1分あたりのレスポ	kube-apiserverリクエストのうち、300以上400未満のレスポンスコード

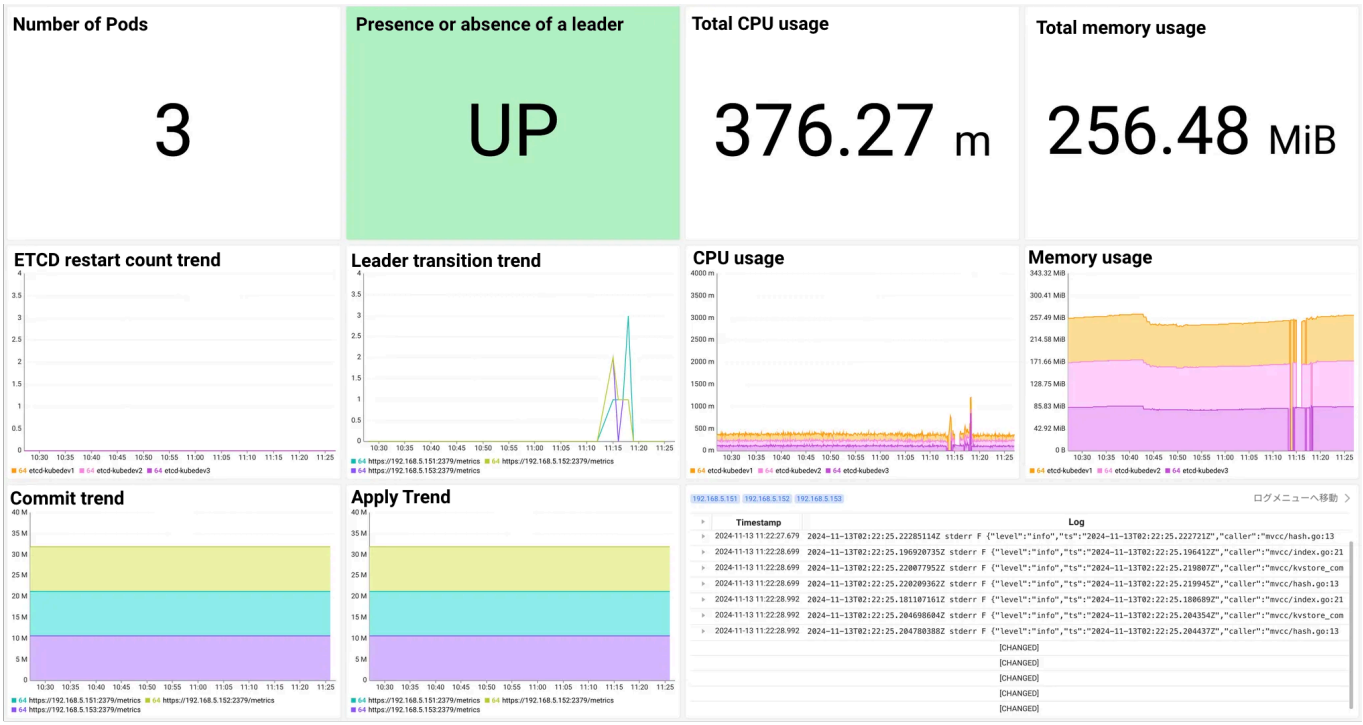
分類	メトリクス	説明
	ンスのリクエスト状況	が出されたリクエスト数に対する1分当たりの現況
	[3xx] 1分当たりのレスポンスコードごとのリクエスト数の推移	kube-apiserverリクエストのうち、300以上400未満のレスポンスコードが出されたリクエスト数に対するレスポンスコードごとの1分当たりの現況
	[3xx] 1分当たりのResource & Subresourceごとのリクエスト数の推移	kube-apiserverリクエストのうち、300以上400未満のレスポンスコードが出されたリクエストに対しResourceとSubresourceごとにグループ化した1分当たりのリクエスト数の推移
	[400~] 1分あたりのレスポンスのリクエスト状況	kube-apiserverリクエストのうち、400以上のレスポンスコードが出されたリクエスト数に対する1分当たりの現況
	[400~] 1分当たりのレスポンスコードごとのリクエスト数の推移	kube-apiserverリクエストのうち、400以上のレスポンスコードが出されたリクエスト数に対するレスポンスコードごとの1分当たりの現況
	[400~] 1分当たりのResource & Subresourceごとのリクエスト数の推移	kube-apiserverリクエストのうち、400以上のレスポンスコードが出されたリクエストに対しResourceとSubresourceごとにグループ化した1分当たりのリクエスト数の推移
Go Metrics	Goルーチン数	現在存在するGoルーチンの数
	スレッド数	現在存在するOSスレッド数
現在実行中のリクエスト数(1秒)	直近1秒以内に実行中のリクエスト数	データ収集時にkube-apiserverで1秒以内に行われていたリクエスト数

etcd ダッシュボード

> **要求事項**
>
> - WhaTap Kubernetesエージェント1.7.16バージョン以降が必要です。
> - Kubernetesクラスタープロジェクト(CP)の場合、該当メニューを照会できます。

etcdは、Kubernetes環境でクラスター状態に関連するデータを保存する重要なコンポーネントです。 WhaTapはetcdの性能と状態をモニタリングできる**etcd ダッシュボード**を提供します。 これにより、クラスターの安定性を高め、問題が発生した場合の迅速な対応が可能になります。

基本画面ガイド



etcd ダッシュボードは、1時間の間に収集された主要なetcdメトリクスを1分周期で提供します。 ダッシュボードで確認できる主メトリクスは次の通りです。 メトリクス詳細は、下段のメトリクス案内を参照してください。

- etcdが使用するリソース、Podの再起動数、詳細ログを提供します。
- etcd内部のcommit及びapply回数を提供します。
- etcdリーダーの存在有無及びリーダーの変更履歴を提供します。

etcdモニタリング有効化

WhaTap Kubernetesエージェントインストール時にダウンロードしたYAMLファイルの修正とmTLS認証書を登録してエージェントを有効にします。

認証書の確認とSecretを作成する

etcd認証書の確認

etcdのmTLS認証書が必要です。 etcd認証書は一般的にControl Planeノードの*/etc/kubernetes/pki/etcd*経路に保存されます。 ファイル所有者はrootアカウントであり、必要な証明書は以下の通りです。

- ca.crt
- server.crt
- server.key

`kubectl` コマンドを実行するマシンで、次のコマンドを使用して証明書が存在することを確認します。

```
sudo ls /etc/kubernetes/pki/etcd
```

Secret生成

証明書が存在する場合、次のコマンドを使用してSecretを生成します。

```
sudo kubectl create secret generic whatap-etcd-cert -n whatap-monitoring --from-file=ca.crt=/etc/kubernetes/pki/etcd/ca.crt
--from-file=server.crt=/etc/kubernetes/pki/etcd/server.crt --from-file=server.key=/etc/kubernetes/pki/etcd/server.key
```

正常にSecret生成時、次のようなメッセージが出力されます。

```
secret/whatap-etcd-cert created
```

rootアカウントが使用できない場合

rootアカウントで `kubectl` コマンドが使用できない場合は、次のガイドを参照してください。

1. 証明書の複製後、アクセス可能なパスに移動します。
2. 複製された証明書ファイルの所有者を、現在の所有者に変更します。

ネームスペースがない場合

KubernetesエージェントYAMLの適用前にSecretを作る場合、 `whatap-monitoring` ネームスペースが存在しないため、Secret作成に失敗する可能性があります。 次のコマンドを使用してネームスペースを作成し、再度Secretを生成します。

```
kubectl create namespace whatap-monitoring
```

WhaTapのKubernetesエージェントYAMLを修正

WhaTapのKubernetesエージェントYAMLファイルを修正してコンテナ設定を追加し、Secretを接続します。

コンテナ設定

1. Deploymentの `whatap-control-plane-helper` コンテナ内のargsセクションにetcd関連設定を追加します。

   ```
   -collect_etcd_monitoring_enabled=true
   -etcd_hosts=ユーザーの_etcd_ip、ユーザーの_etcd_ip
   ```

   次の例を参照してください。

   Example

   ```
   -etcd_hosts=192.168.1.45,192.168.1.46,192.168.1.47
   ```

2. Deploymentの `whatap-master-agent` コンテナの環境変数(env)に次の内容を追加します。

   ```
   - name: "collect_etcd_monitoring_enabled"
     value: "true"
   ```

Secret接続

1. Deploymentの `whatap-master-agent` 内のvolumesセクションにSecretを追加します。

```
- name: "whatap-etcd-cert"
  secret:
   secretName: "whatap-etcd-cert"
```

2. `whatap-control-plane-helper` コンテナに当該Secretをマウントします。

```
volumeMounts:
 - name: "whatap-etcd-cert"
    readOnly: true
    mountPath: "/etc/kubernetes/pki/etcd"
```

全体YAMLの例

以下は、すべての設定が適用されたYAMLファイルの例です。 ハイライトされた部分を参照してください。

```
apiVersion: "apps/v1"
kind: "Deployment"
metadata:
  name: "whatap-master-agent"
  namespace: "whatap-monitoring"
spec:
  replicas: 1
  selector:
    matchLabels:
      name: "whatap-master-agent"
  template:
    metadata:
      labels:
        name: "whatap-master-agent"
    spec:
      containers:
      - command:
        - "/bin/entrypoint.sh"
```

```
    env:
    - name: "WHATAP_LICENSE"
      value: "x208gnc8i1kko-z2p19shk7npc4t-z788sgbj3n123a"
    - name: "WHATAP_HOST"
      value: "59.13.101.109"
    - name: "WHATAP_PORT"
      value: "61574"
    - name: "WHATP_MEM_LIMIT"
      valueFrom:
        resourceFieldRef:
          containerName: "whatap-master-agent"
          resource: "limits.memory"
    - name: "collect_control_plane_monitoring_enabled"
      value: "true"
    - name: "collect_etcd_monitoring_enabled"
      value: "true"
    image: "whatap/kube_mon"
    name: "whatap-master-agent"
    ports:
    - containerPort: 6600
    resources:
      limits:
        cpu: "200m"
        memory: "350Mi"
      requests:
        cpu: "100m"
        memory: "300Mi"
    volumeMounts:
    - mountPath: "/bin/entrypoint.sh"
      name: "start-script-volume"
      readOnly: true
      subPath: "entrypoint.sh"
    - mountPath: "/whatap_conf"
      name: "whatap-config-volume"
  - args:
    - "-collect_control_plane_monitoring_enabled=true"
    - "-collect_etcd_monitoring_enabled=true"
    - "-etcd_hosts=192.168.1.45,192.168.1.46,192.168.1.47"
    command:
    - "/data/agent/master/whatap_control_plane_helper"
    image: "whatap/kube_mon"
```

```
    imagePullPolicy: "Always"
    name: "whatap-control-plane-helper"
    ports:
    - containerPort: 9496
    volumeMounts:
     - name: "whatap-etcd-cert"
       readOnly: true
       mountPath: "/etc/kubernetes/pki/etcd"
  serviceAccount: "whatap"
  volumes:
  - configMap:
      defaultMode: 448
      name: "master-start-script"
    name: "start-script-volume"
  - emptyDir: {}
    name: "whatap-config-volume"
  - name: "whatap-etcd-cert"
    secret:
      secretName: "whatap-etcd-cert"
```

etcdメトリクス

区分	説明
Pod数	現在存在するetcd Podの数
etcd再起動数の推移	etcd Podの再起動回数の合計推移
リーダー変更の推移	etcd Podリーダーの変更回数推移
リーダーの存在有無	リーダーの役割をするetcd Podの存在有無
CPUの合計使用量	etcd Podの合計CPU使用量
CPU使用量	etcd PodのCPU使用量の推移
Memoryの合計使用量	etcd Podの合計メモリ使用量

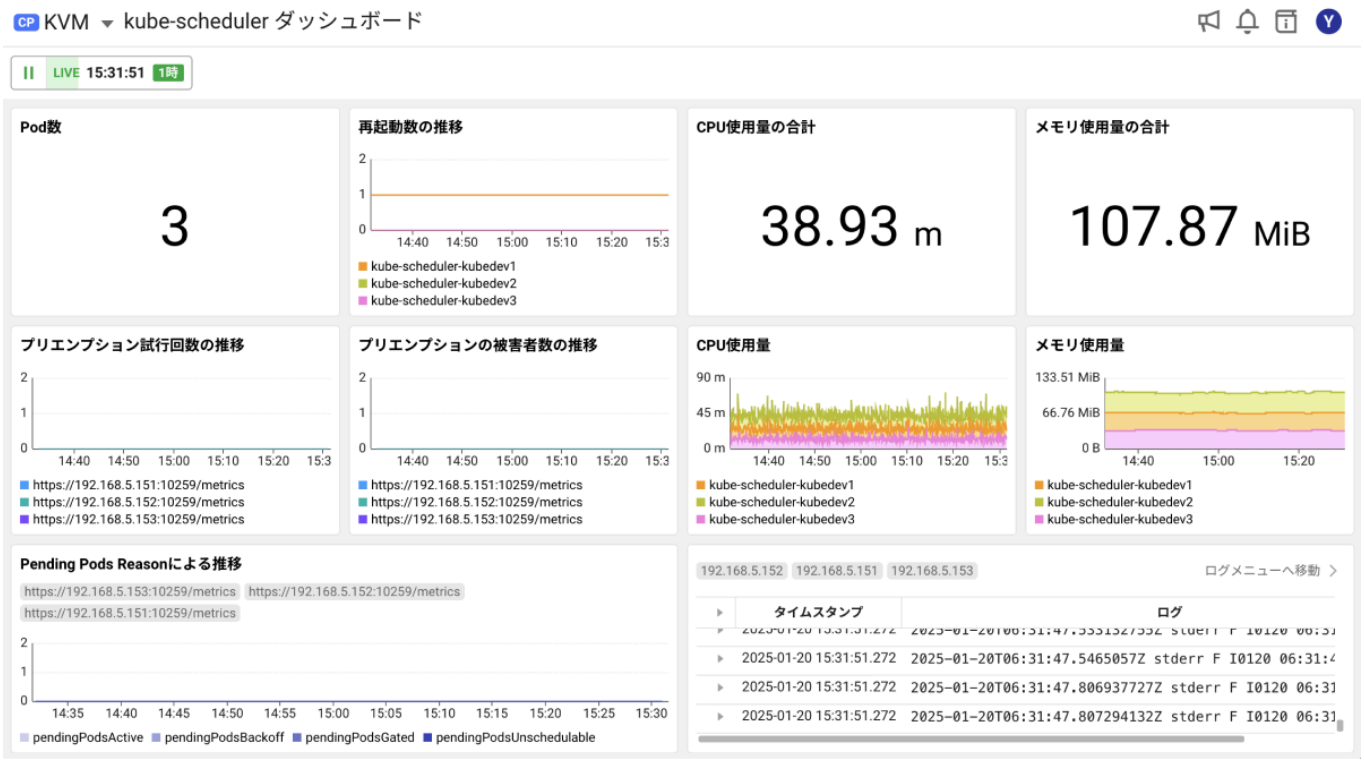
区分	説明
Memory使用量	etcd Podのメモリ使用量の推移
Commit推移	etcd変更事項commit回数
Apply推移	commit以後applyされた回数
etcdログ	etcd Podで発生したログ

kube-scheduler ダッシュボード

> **要求事項**
>
> - WhaTap Kubernetesエージェント1.8.1バージョン以降が必要です。
> - Kubernetesクラスタープロジェクト(CP)の場合、該当メニューを照会できます。
> - Vanilla Kubernetesまたは `kube-scheduler` Podを観測できるクラスターの場合は、そのメニューを使用できます。

`kube-scheduler`はKubernetes環境でPodのスケジューリングを担当する主要なコンポーネントです。 WhaTapは`kube-scheduler`の性能、スケジューリング状態をリアルタイムでモニタリングできる**kube-scheduler ダッシュボード**を提供します。 **kube-scheduler ダッシュボード**これにより、クラスターがスケジューリング不能に陥ったり、スケジューラの負荷による問題を事前に回避できます。

基本画面ガイド



kube-scheduler ダッシュボードは、最近1時間収集された主要な指標を1分単位で視覚化して提供します。 ダッシュボードで確認できる主メトリクスは次の通りです。 メトリクス詳細は、下段のメトリクス案内を参照してください。

- kube-scheduler が使用するリソースの推移、Pod再起動の回数、そして詳細ログを提供します。
- kube-scheduler の先取り試行の回数と先取りによる被害者数の推移を提供します。
- kube-scheduler Pod内部のPending Queueに割り当てられたPending PodのReason別推移を提供します。

kube-schedulerモニタリングの活性化

活性化の要求条件

- kubectl get pods -n kube-system -l component=kube-scheduler コマンドを通じて kube-scheduler Podを観測

できるようにする必要があります。

- `kube-scheduler` とのセキュリティ通信のため `--bind-address` の設定を外部からアクセスできるように変更する必要があります。 一般的に `0.0.0.0` を使います。

活性化の過程

> **活性化の順序**
>
> 1. whatap service accountのRBACに `secrets` に対する `GET` 権限を追加します。
> 2. whatap service accountをもとに `secrets` を作成します。
> 3. `whatap-master-agent` deploymentに `kube-scheduler` モニタリングの活性化オプションを適用してくだ活性化

1. **権限設定**

   `whatap-monitoring` のネームスペース内でwhatap ClusterRoleの `rules` にアクセス可能なリソースに `secrets` リソースを追加します。 次の内容をyamlファイルに作成した後、 `kubectl apply -f <yamlファイル名>.yaml` を実行して適用します。

   ClusterRole.yaml

```yaml
apiVersion: "rbac.authorization.k8s.io/v1"
kind: "ClusterRole"
metadata:
 annotations:
  rbac.authorization.kubernetes.io/autoupdate: "true"
 name: "whatap"
 namespace: "whatap-monitoring"
rules:
- apiGroups:
 - "*"
 resources:
 - "namespaces"
 - "pods"
 - "pods/log"
 - "configmaps"
```

```
  - "services"
  - "endpoints"
  - "daemonsets"
  - "ingresses"
  - "deployments"
  - "nodes"
  - "persistentvolumes"
  - "persistentvolumeclaims"
  - "events"
  - "secrets"
  - "replicasets"
  - "roles"
  - "rolebindings"
  - "clusterroles"
  - "clusterrolebindings"
  - "jobs"
  - "cronjobs"
  - "statefulsets"
  - "serviceaccounts"
  - "configmaps"
  - "storageclasses"
  - "horizontalpodautoscalers"
  - "replicationcontrollers"
  verbs:
  - "get"
  - "list"
  - "watch"
- apiGroups:
  - ""
  resources:
  - "pods/exec"
  verbs:
  - "create"
- apiGroups:
  - ""
  resources:
  - "configmaps"
  verbs:
  - "*"
```

```
- nonResourceURLs:
  - "/metrics"
  verbs:
  - "*"
```

2. **Secret生成**

   次の内容をyamlファイルに作成した後、 `kubectl apply -f yamlファイル名.yaml` を実行して適用します。

   secrets.yaml

```
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  name: whatap-scheduler-monitoring-token
  namespace: whatap-monitoring
  annotations:
    kubernetes.io/service-account.name: "whatap"
```

3. **Deployment修正**

   `whatap-master-agent` deploymentの内容を修正した後、 `kubectl apply -f <yamlファイル名>.yaml` のコマンドを使って適用します。

   - **環境変数の追加**

     `containers` 項目下位の `whatap-master-agent` コンテナの `env` セクションに次の内容を追加します。

```
env:
- name: "collect_kube_scheduler_monitoring_enabled"
  value: "true"
```

   - **コマンド因子の追加**

     `containers` 項目下位の `whatap-control-plane-helper` コンテナの `args` セクションに次の内容を追加します。

```
 - args:
  - "-collect_kube_scheduler_monitoring_enabled=true"
```

環境変数およびコマンド因子を追加する場合は、次の例を参照してください。

```
apiVersion: "apps/v1"
kind: "Deployment"
metadata:
 name: "whatap-master-agent"
 namespace: "whatap-monitoring"
spec:
 replicas: 1
 selector:
  matchLabels:
   name: "whatap-master-agent"
 template:
  metadata:
   labels:
    name: "whatap-master-agent"
  spec:
   priorityClassName: high-priority
   containers:
   - command:
    - "/bin/entrypoint.sh"
    env:
    - name: "WHATAP_LICENSE"
     value: "c1024123i1kko-241245127abdbe-612632344235"
    - name: "WHATAP_HOST"
     value: "192.168.1.0"
    - name: "WHATP_MEM_LIMIT"
     valueFrom:
      resourceFieldRef:
       containerName: "whatap-master-agent"
       resource: "limits.memory"
    - name: "collect_kube_scheduler_monitoring_enabled"
     value: "true"
    image: "whatap/kube_mon"
    name: "whatap-master-agent"
```

```
    ports:
    - containerPort: 6600
    resources:
      limits:
        cpu: "200m"
        memory: "350Mi"
      requests:
        cpu: "100m"
        memory: "300Mi"
    volumeMounts:
    - mountPath: "/bin/entrypoint.sh"
      name: "start-script-volume"
      readOnly: true
      subPath: "entrypoint.sh"
    - mountPath: "/whatap_conf"
      name: "whatap-config-volume"
  - args:
    - "-collect_kube_scheduler_monitoring_enabled=true"
    command:
    - "/data/agent/master/whatap_control_plane_helper"
    image: "whatap/kube_mon"
    imagePullPolicy: "Always"
    name: "whatap-control-plane-helper"
    ports:
    - containerPort: 9496
  serviceAccount: "whatap"
  volumes:
  - configMap:
      defaultMode: 448
      name: "master-start-script"
    name: "start-script-volume"
  - emptyDir: {}
    name: "whatap-config-volume"
```

kube-schedulerのメトリック

区分	説明
Pod数	現在実行中の `kube-scheduler` Podの数
kube-schedulerの再起動数の推移	`kube-scheduler` Podの再起動回数の合計推移
CPUの合計使用量	`kube-scheduler` Podの総CPU使用量
CPU使用量	`kube-scheduler` PodのCPU使用量推移
Memoryの合計使用量	`kube-scheduler` Podの総メモリ使用量
Memory使用量	`kube-scheduler` Podのメモリ使用量の推移
kube-scheduler ログ	`kube-scheduler` Podで発生したログ
先取り試みの回数推移	Podスケジューリングのために先取りを試みた回数の推移
先取り被害者数の推移	先取りによって追い出されたPod数の推移
Pending Pod Reason別の推移	スケジューリングされずにPending状態で待機中のPodの原因別推移

Object Manifest情報

> **要求事項**
>
> WhaTap Kubernetesエージェント1.5.5バージョン以降が必要です。 1.5.5バージョン未満の場合は、既存のマスターメタ情報画面が提供されます。

Kubernetesマスターエージェントは、クラスターにあるPod、サービス、Replicasetなどのオブジェクトに対するマニフェストを1日に1回ずつ自動的に収集します。 該当データは日付別に保存され、ユーザーが特定の日付基準のクラスター状態を簡単に確認できます。 収集したデータは日付別の比較が可能で、時間によるクラスターの変化を把握しやすいです。

また、ユーザーは、KubernetesマスターエージェントのDeployment配布前にコンテナ環境変数を設定することで、収集時間のタイムゾーンを自由に指定できます。 当該設定方法は、次のコンテナ環境変数 の案内を参考にしてください。

コンテナ環境変数

- **store_k8s_object_manifest_enabled**

  既定値 `true`

  オブジェクトマニフェスト収集機能を活性化します。 既定値が `true` であるため、別途の設定なしで収集機能が活性化されます。

- **store_k8s_object_manifest_timezone**

  既定値 `UTC`

  オブジェクトマニフェスト収集のためのタイムゾーンを設定します。 別途設定がない場合、既定値の `UTC` を基準にして、午前0時に収集を開始します。

  - リージョンベースのタイムゾーンまたはGMTオフセット形式で設定できます。
  - 分単位の精密な時間指定に対応します。

    例、 `GMT+0810` のように設定することで、特定地域の非標準時間帯の指定が可能

- **store_k8s_object_manifest_max_fetch_limit**

  既定値 `50`

  クラスターが一度にリクエストするオブジェクト数を設定します。 別途設定がない場合、既定値の `50` を基準にして1回のリクエストにつき50個ずつリクエストします。

  例、全体のPod数が90個のクラスタで該当オプションを `30` に設定すると、Pod情報を30個ずつ3回リクエスト

Example

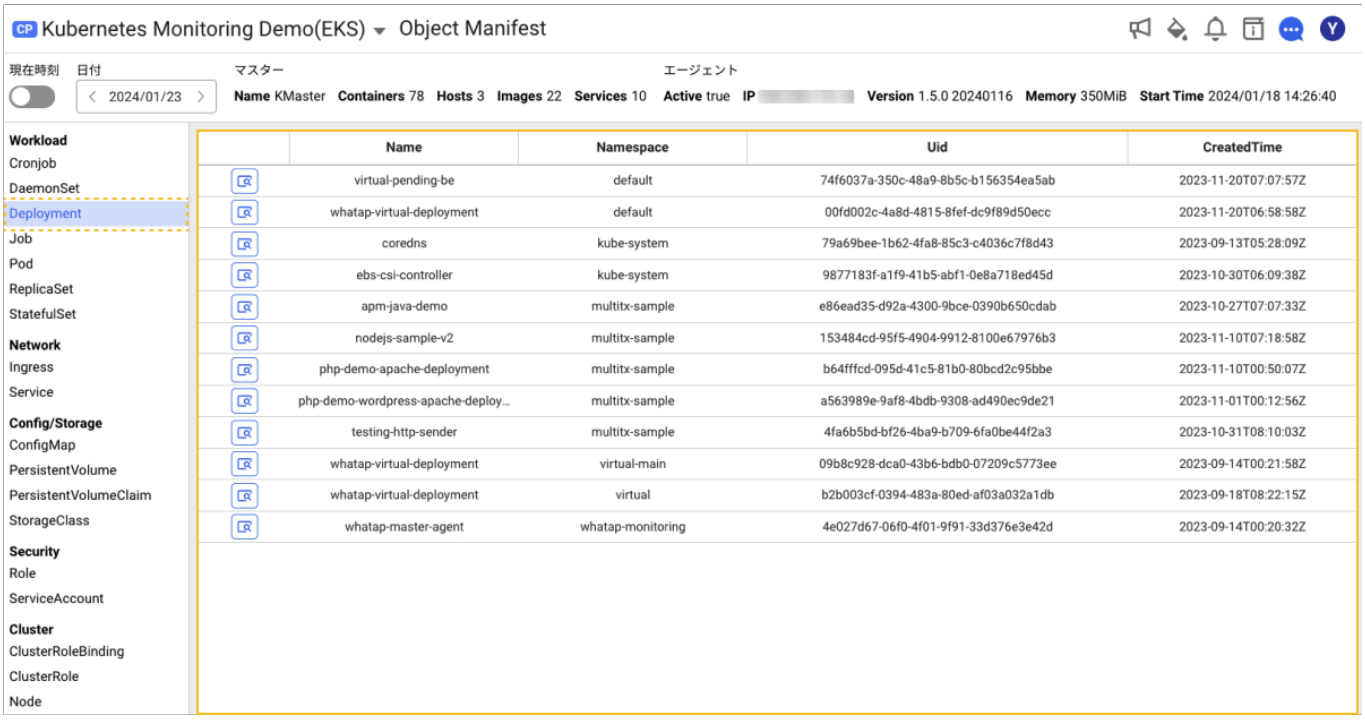
```
containers:
    - name: whatap-master-agent
      image: whatap/kube_mon
      resources:
        requests:
          memory: 300Mi
          cpu: 100m
        limits:
          memory: 350Mi
          cpu: 200m
      ports:
        - containerPort: 6600
      command: ['/bin/entrypoint.sh']
      env:
        - name: WHATAP_LICENSE
          value: {PROJECT_LICENSE_VALUE}
        - name: WHATAP_HOST
          value: "13.124.11.223/13.209.172.35"
        - name: WHATAP_PORT
          value: "6600"
        - name: WHATP_MEM_LIMIT
          valueFrom:
            resourceFieldRef:
              containerName: whatap-master-agent
              resource: limits.memory
        # 追加された環境変数
        - name: store_k8s_object_manifest_enabled
          value: "true"
        - name: store_k8s_object_manifest_timezone
          value: "GMT+9"
        - name: store_k8s_object_manifest_max_fetch_limit
```

```
    value: "50"
```

基本画面ガイド

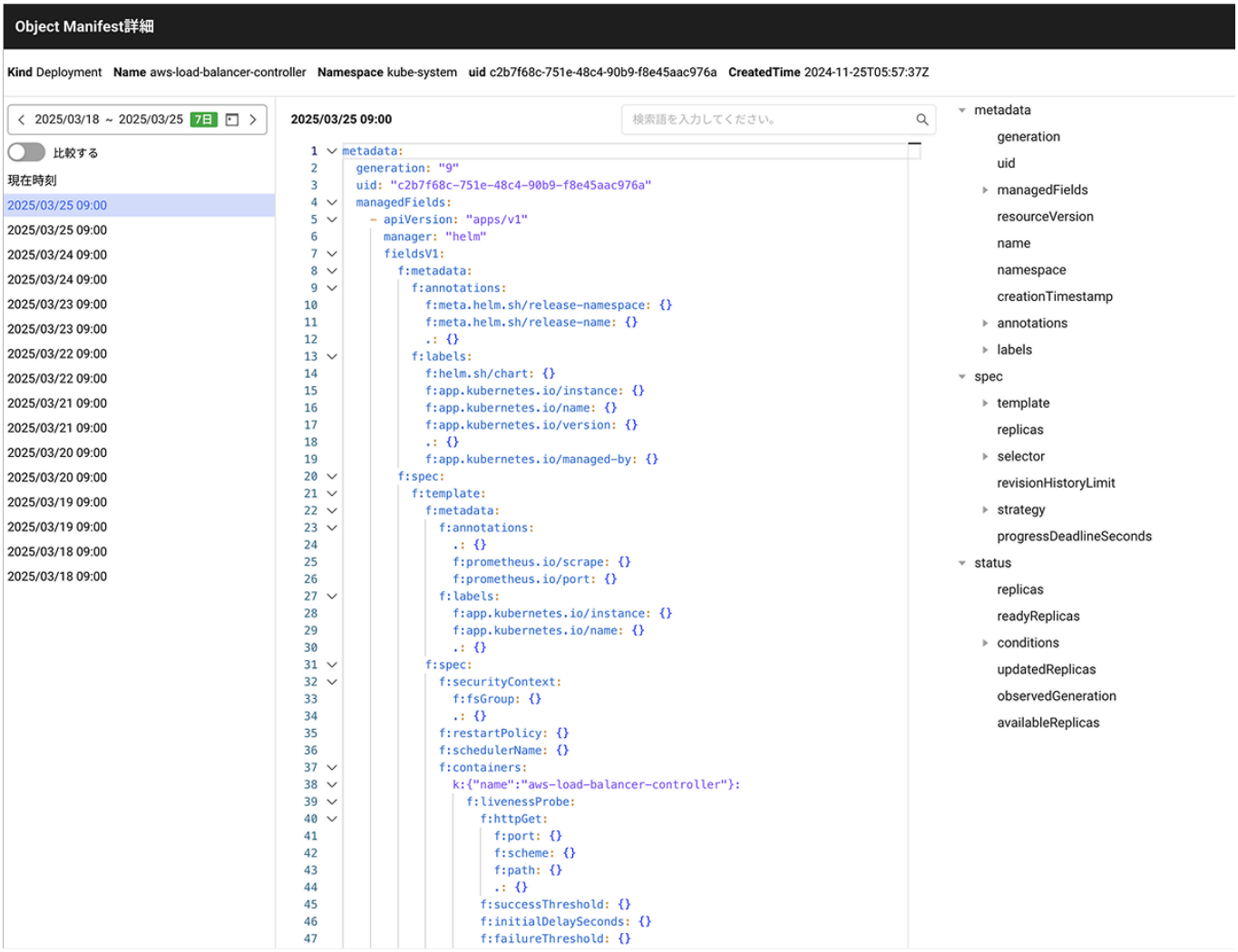
オブジェクトリスト

オブジェクトの種類の一覧からオブジェクトを選択すると、そのオブジェクトの一覧を確認できます。 Name、Namespace、Uid、CreatedTimeなどのオブジェクト情報が表示されます。 次の画面は、オブジェクト種類の一覧からDeploymentを選択する時の例です。



Object Manifest情報の詳細

オブジェクト一覧で 詳細を見るボタンの選択時オブジェクトマニフェスト詳細ウィンドウで選択したオブジェクトのマニフェストを照会できます。



オブジェクトマニフェストの保存時間一覧

- そのオブジェクトのマニフェストが保存されている時間の一覧を左側で確認できます。 オブジェクトマニフェストは、毎日決まった時間に自動的に保存されます。
- オブジェクトマニフェストの保存時間は、マスター エージェントのコンテナ環境変数 `store_k8s_object_manifest_timezone` 値を修正して変更できます。
- オブジェクトマニフェストは、毎日決められた保存時間に保存されるだけでなく、警告イベントが発生した場合にも追加で保存されます。

> ⓘ オブジェクトマニフェストが保存される警告イベントは、WarningタイプのKubernetesイベントオブジェク

トを意味します。

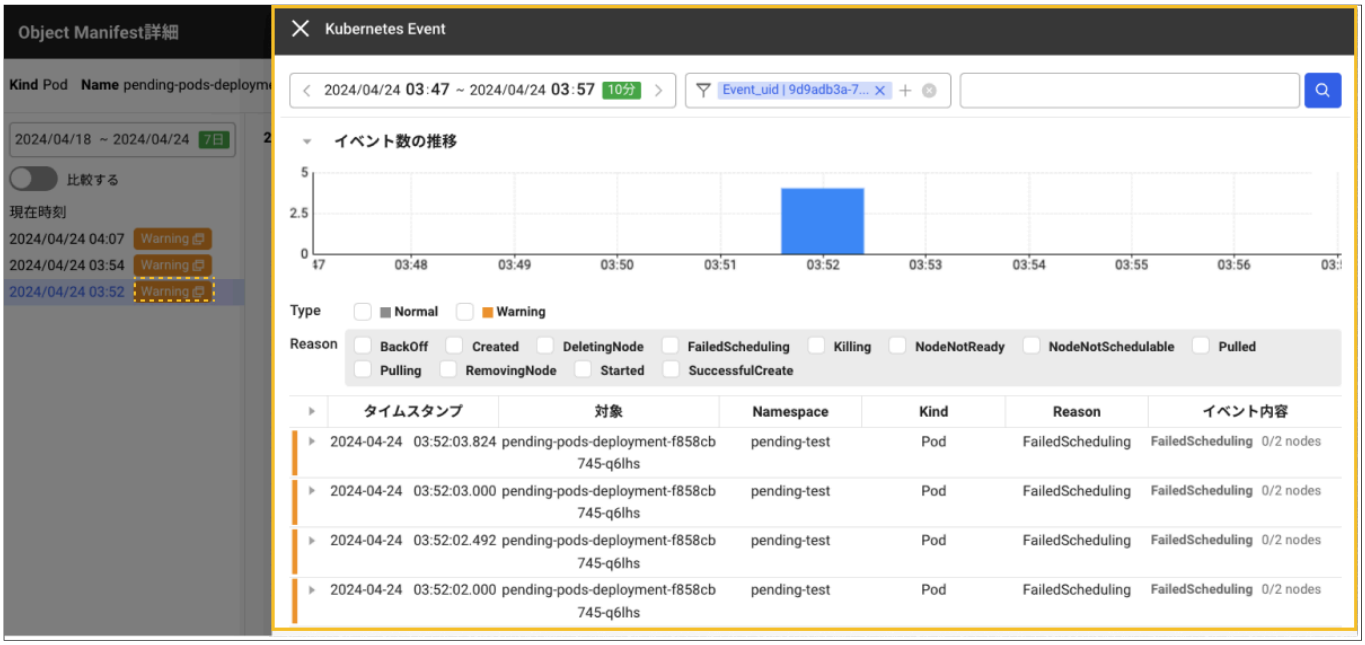
オブジェクトマニフェスト検索と探索

オブジェクトマニフェスト詳細画面でオブジェクトマニフェスト内の特定のキーワードを見つけるための検索機能を提供します。 検索バーに入力した文字列は、オブジェクトマニフェスト内でハイライトされます。 これにより、複雑なマニフェスト構造で特定の設定やフィールドをすばやく確認できます。

またオブジェクトマニフェスト詳細画面の右側に、マニフェスト全体の構造をツリー形式で要約したナビゲーションパネルを提供します。 これにより、大容量マニフェスト文書を扱う際に、別途のスクロールなしに主要項目を確認できます。

Kubernetesイベント連携

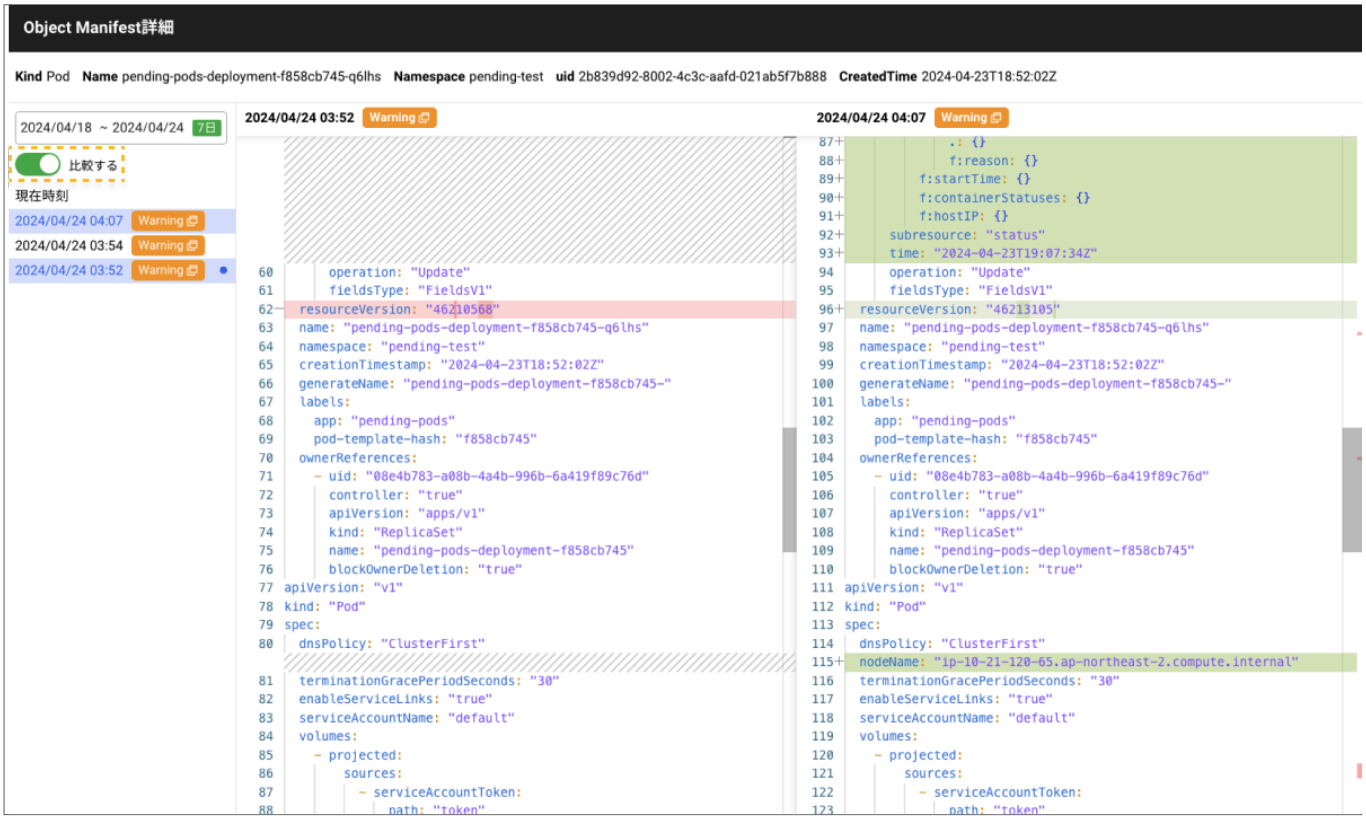
オブジェクトマニフェスト詳細画面でWarningボタンをクリックすると、次のようにKubernetesイベントを確認できます。



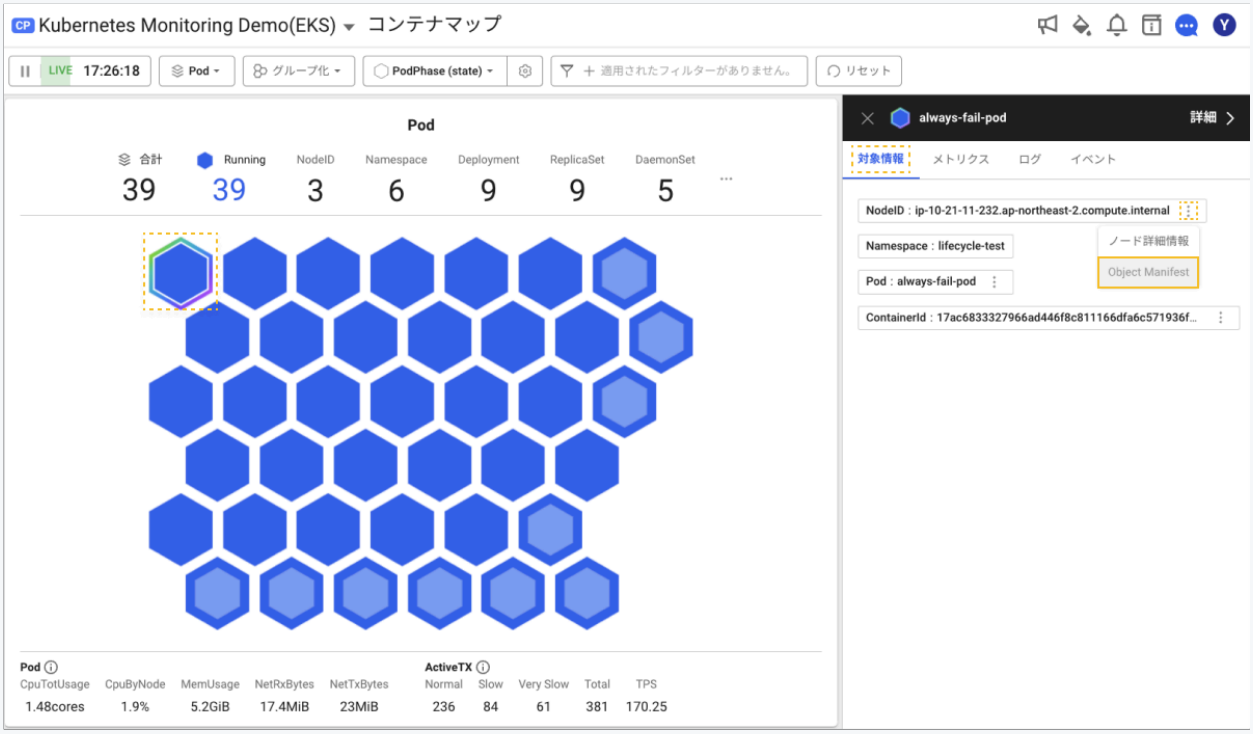
Kubernetesイベントの連携機能は、WhaTapマスターエージェントのバージョン1.6.1以降に対応します。

Object Manifest情報の比較

比較するトグル活性化時に保存時間リストから選択し、次のように2つのオブジェクトマニフェストを比較できます。 変更事項がある場合、ハイライトされます。



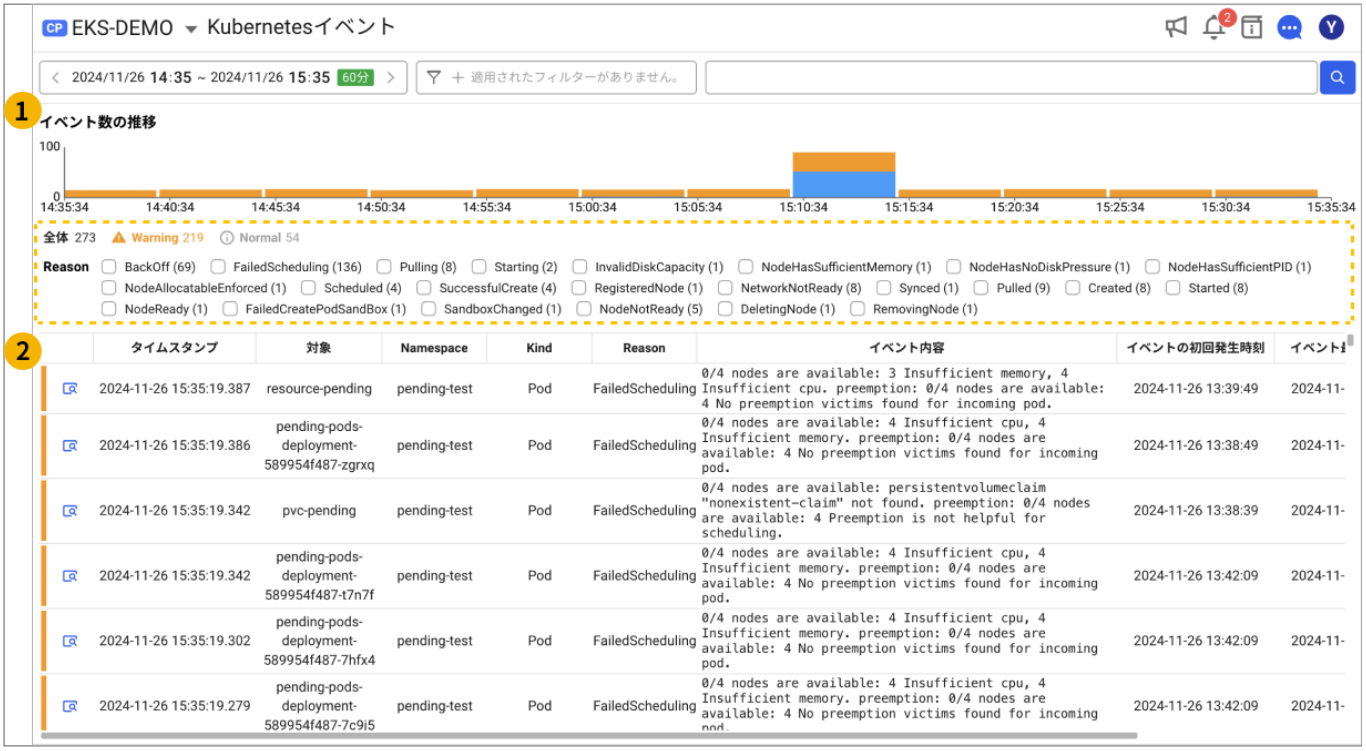
> **コンテナマップでのオブジェクトマニフェスト照会**
>
> コンテナ·マップでオブジェクトマニフェストを確認できます。



- ブロックを選択します。
- 要約情報の画面で対象情報タブを選択し、Podまたはコンテナ情報を確認します。
- Pod、Deployment、ReplicaSet、Nodeタグの右の ⋮ 詳細を見るアイコンをクリックしてオブジェクトマニフェストを選択します。

クバネティスイベント

WhaTap**Kubernetesイベント**メニューを通じて、コマンドなしでKubernetesイベントに対する様々な照会と検索が可能です。 ユーザーのクラスターイベントを確認でき、時間セレクタを通じて過去のイベントも照会できます。



- フィルターオプションと検索ウィンドウで希望するイベントを確認できます。
- NormalまたはWarningのようなイベントタイプ及び**Reason**基準に素早くフィルタリングできます。

> ⓘ • Kubernetesイベントについての詳細は、次の文書を参照してください。

イベント数の推移

バーチャート形式で提供される1 **イベント数の推移**これにより、選択した時間範囲内で発生したイベント数の変化を視覚的に確認できます。

照会を希望する時間区間を選択し、該当区間で発生したイベントをイベント一覧ですぐに確認できます。 これにより、イベントが集中的に発生した時間帯を素早く識別し、詳細を分析できます。

イベント一覧

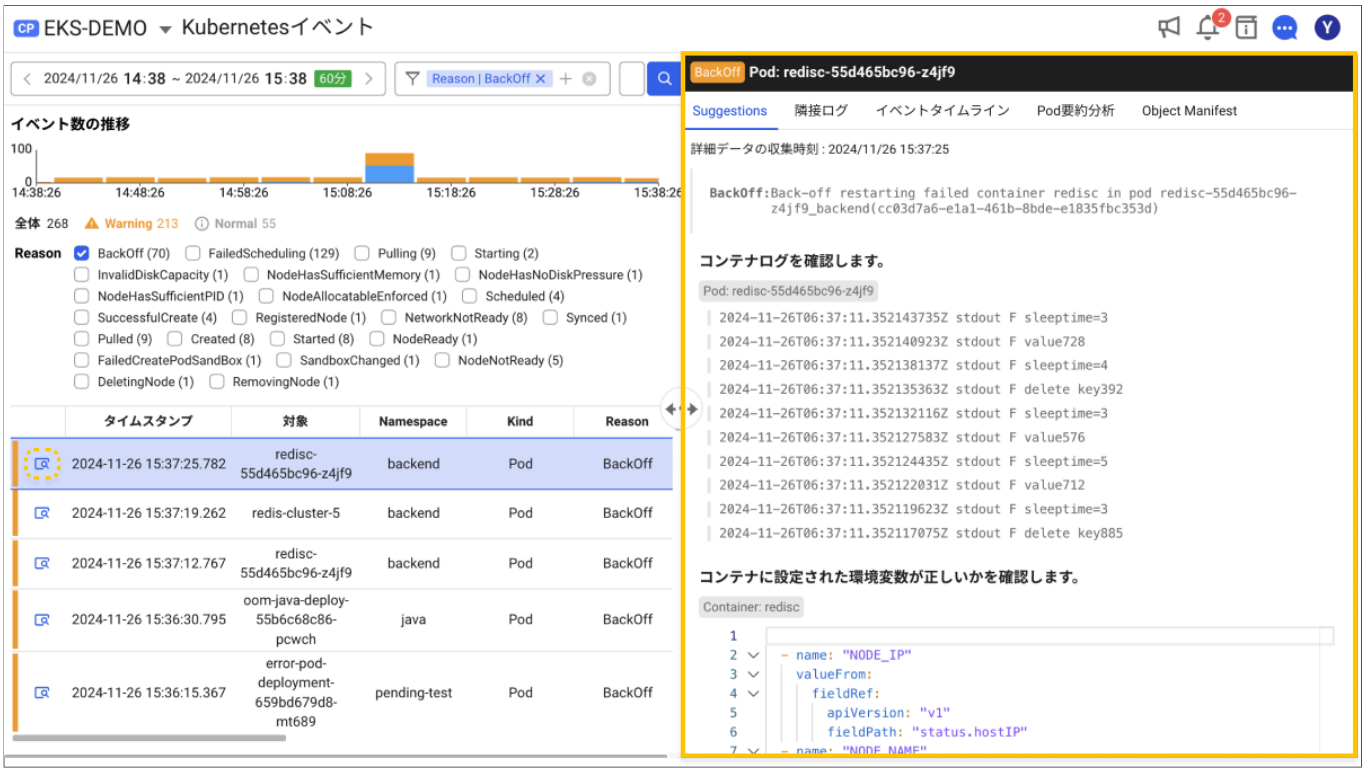
特定の時間帯に発生したKubernetesイベント一覧を2領域で確認できます。 WhaTapは最大30日間のイベントデータを保存し、これによって過去のイベントを照会して分析できます。

また、一覧上段でイベントType(Warning、Normal) またはReasonを選択して、希望するイベントを素早くフィルタリングできます。

項目	説明
タイムスタンプ	イベントが発生した時間です。
対象	イベントが発生した対象です。 KindがPodの場合はPod名、Nodeの場合はNode名を表示します。
Namespace	イベントが発生した対象が属するネームスペースです。
Kind	イベントが発生した対象の種類です。
Reason	イベントが発生した理由です。
イベント初回発生時間	イベントが最初に観察された時間です。
イベント発生件数	一定の時間継続的に発生したイベントの回数です。
イベント内容	具体的なイベントの詳細情報です。

イベント詳細

イベント一覧で照会を希望するイベントの左側詳細アイコン選択時**イベント詳細**を確認できます。



Suggestions

Suggestionsタブは、発生したイベントメッセージに基づいてイベントの発生原因を分析し、問題を追跡するための方向性を提示します。

> ⓘ 現在Warningタイプの${Field}**BackOff, NodNotReady**イベントのみが提供される機能です。

隣接ログ

選択したイベント発生時点を基準に隣接したログ照会を通じてイベント発生前後の脈絡を確認できます。 時間選択ボタンにより、希望の隣接範囲を設定できます。

> ⓘ **ログ照会権限**
>
> - **ログ照会権限**がある場合は、隣接するすべてのログを確認できます。

- **ログ照会権限**がない場合は、隣接するKubernetesイベントのみ確認できます。

イベントタイムライン

選択したイベントの対象(Kind)が生成してから現在までに発生したすべてのイベントをタイムライン形式で提供します。 これにより、対象のイベント履歴を一目で把握できます。 これにより、対象リソースの生成、正常動作、警告発生、復旧など、ライフサイクル全般を一目で把握できます。

要約分析

イベントが発生した対象であるPodまたはNodeの性能と資源状態を一目で把握できるように要約情報を提供します。

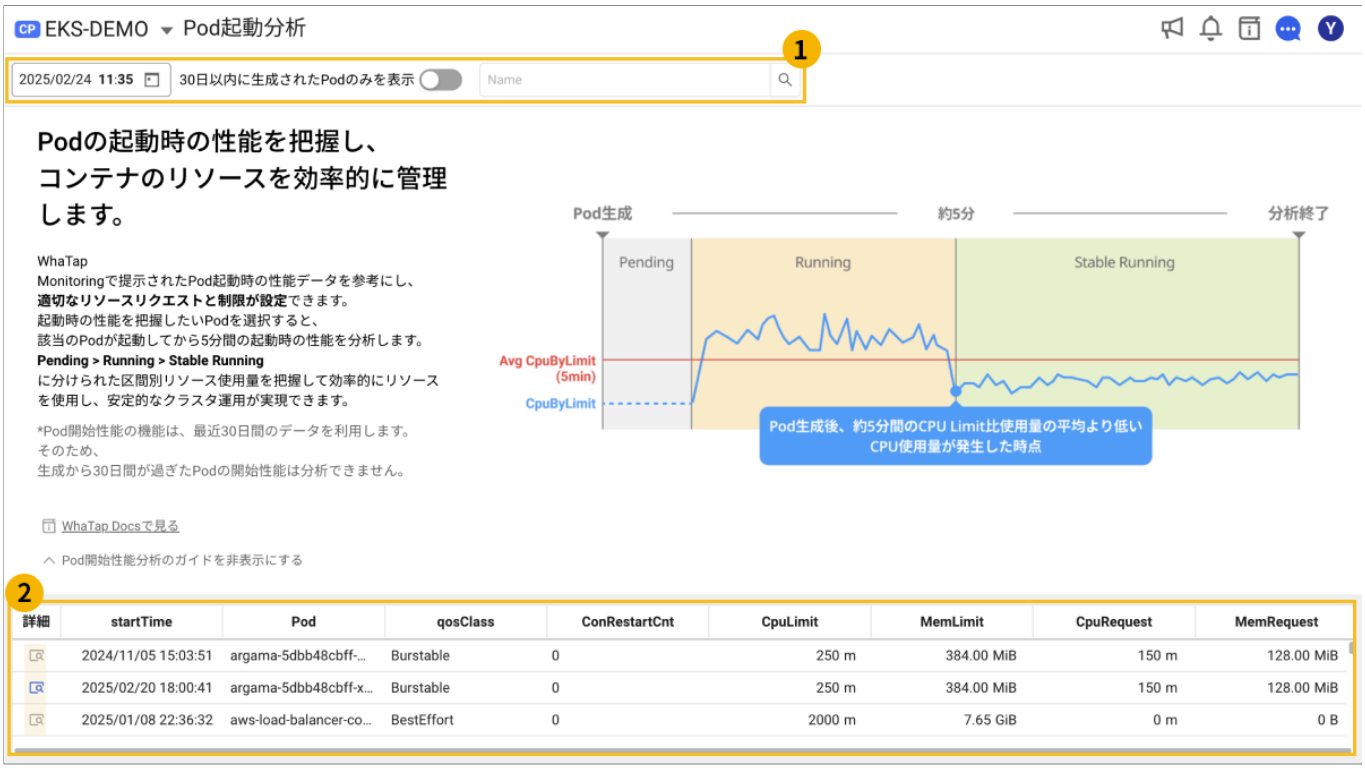
Pod起動分析

> WhaTapノードエージェントバージョン**1.4.2以降**に対応します。 当該バージョン未満の場合、既存メニュー(旧**Pod初期化性能**) 画面を提供します。

円滑なクラスター運用のために、Podが安定的に動作するまでの性能を分析し、Podに割り当てられたリソースのサイズを最適化する必要があります。 **Pod起動分析**メニューは、Podが起動してから約5分間の変化を追跡します。**Pending**状態から**Running**状態に、そして安定的な**Stable Running**状態につながる3段階のリソース使用量を分析することにより、リソース割当量を最適化できます。

これにより、運用者は、リソース効率の向上とコストの削減、スケーラビリティの向上、スケールの容易性などの利点を得ることができます。

基本画面ガイド



上部オプション領域

上段の1領域で以下のオプションを活用できます。

- タイムセレクターを通じて開始時刻を選択できます。 照会を希望する開始時間を選択した場合2領域のPodリストからデータを確認できます。
- 30日以内に生成されたPodのみ表示 トグルを活性化して2領域のPod一覧で30日以内のPodだけを照会できます。
- Pod検索の入力欄に文字を入力してPodを検索できます。

> ⓘ
> - **Pod起動分析**メニューは、タイムセレクター機能をWhaTap Kubernetesエージェント 1.7.6バージョンから提供します。
> - **Pod起動分析**メニューはPod生成日から最近30日以内の範囲内で5分間のPod開始データを分析します。 つまり、生成後30日が過ぎたPodの場合、開始性能を分析できません。

Pod一覧領域

2領域でPod一覧を確認できます。 Pod一覧が開いたテーブルのヘッダーカラムには、次の情報が含まれています。

- QoS Class：PodのQuality of Serviceを意味します。 QoSクラスは、リソース不足時のPodの優先順位を決定します。 優先順位の高いPodが優先順位の低いPodより先に終了します。
  - Guaranteed: ノードで使用可能なリソースが保証され、他のPodよりも優先的に割り当てます。 最後に終了します。
    - PodのすべてのコンテナがCPUおよびメモリのrequestを持っている場合です。
    - PodのすべてのコンテナがCPUおよびメモリに limitを持っており、それらの値がrequestと同じである場合です。
  - Burstable: Podはノードのリソースが不足している場合は、一部のリソースを割り当てます。
    - Guaranteedクラスの条件を満たしていない場合です。
    - Podのコンテナのいずれかがが CPUおよびメモリに request または limitが設定されている場合です。
  - BestEffort: リソース requestやlimitがないため、他のPodより優先度が低くなります。 最初に終了します。

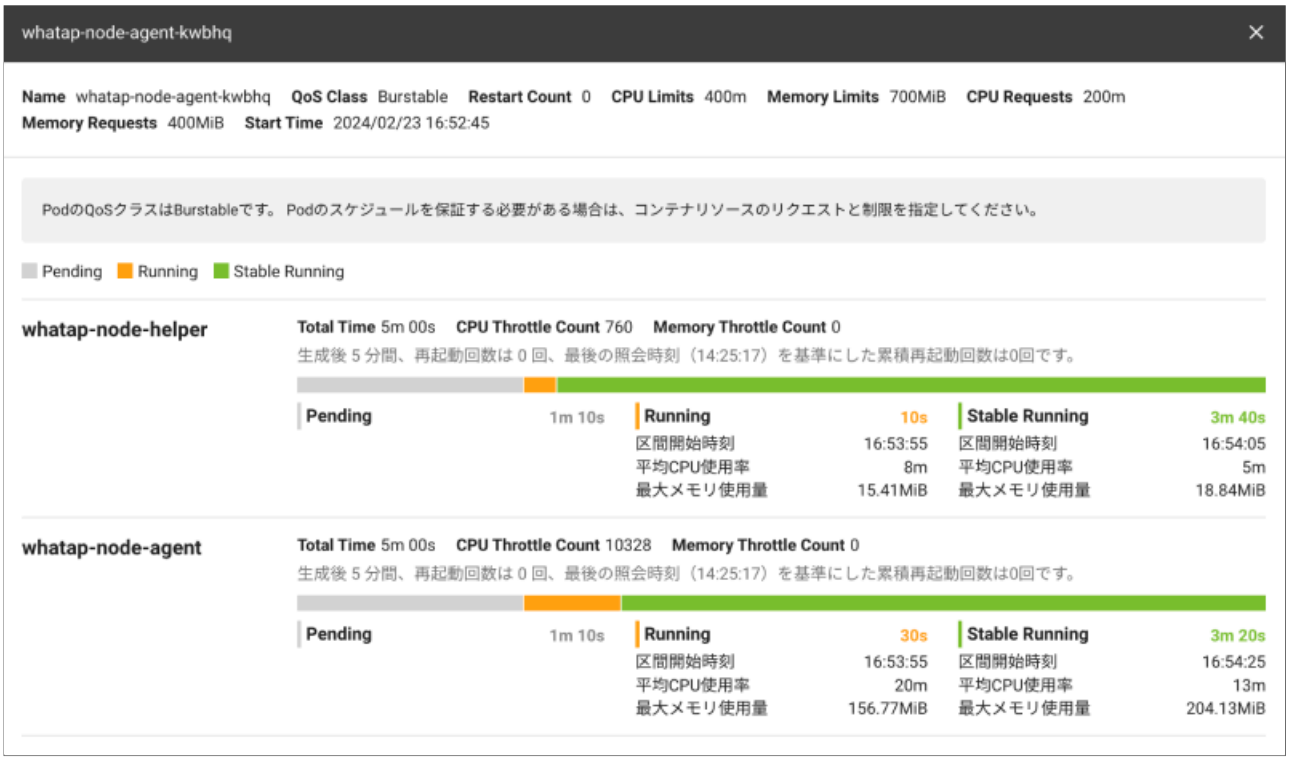
    - Podのすべてのコンテナには、CPUおよびメモリの requestと limitはありません。

- Restart Count：Pod内コンテナの総再起動回数です。
- CPU/Memory Limits: Pod内コンテナに設定されたCPU/Memory Limitsの総合(resources.limits)です。
- CPU/Memory Requests: Pod内コンテナに設定されたCPU/Memory Requestsの総合(resources.requests)です。
- Start Time: Pod生成時刻(metadata.startTime)です。

> ⓘ
> - QoS Classの詳細については、次のリンクを参考にしてください。
> - Pod及びコンテナリソースの管理の詳細については、次のリンクを参考にしてください。

詳細表示の案内

2領域のPod一覧で分析を希望する項目の一番左の 詳細を見るボタンを選択します。 選択したPodのコンテナ別の開始性能を区間別に分析した内容を次のように確認できます。

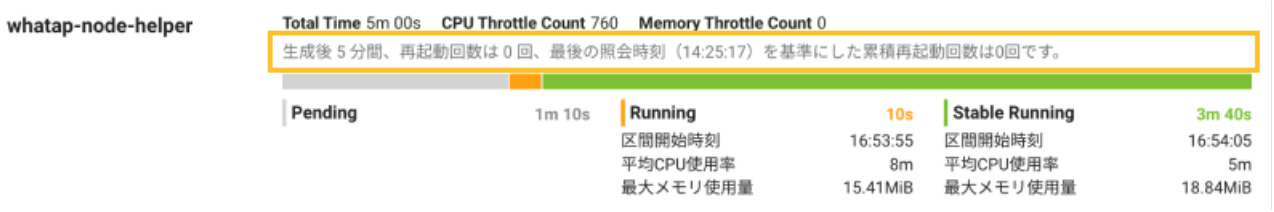


- Pending: Podを生成した後Running状態に進入する区間です。
- Running: PodがPending状態を経て安定化するまでの区間です。
- Stable Running: PodがRunning状態を経て安定化した区間です。

> ⓘ **安定化した状態(Stable Running)**
>
> WhaTapでは、Pod生成直後の約5分間、コンテナのCPU Limit比使用量の平均を計算します。 この平均値よりCPU使用量が少なくなる時点を安定化の開始状態と定義します。

詳細分析する

- 5分間に1つのコンテナが何度も再起動した場合は、最初に開始したコンテナの情報のみを表記します。 コンテナごとの再起動回数を確認してください。 再起動の回数が多い場合は、そのコンテナが正常に実行されていることを確認してください。



- Pending区間が長いなら、PodがRunning状態に進入するまで長い時間がかかったという意味です。 PodのPending状態を誘発する要因があるか点検してみてください。

> ⓘ Podの**pending**ステータスの詳細については、次のリンクを参考にしてください。

- Running区間が長いなら、Pod内のコンテナが初めて実行される時、リソース使用量が多い区間があるという意味です。 コンテナの実行過程に問題がないか確認してください。
- 全体的な初期リソース使用量が多い場合は、コンテナリソースのリクエストと制限を増やすことを検討してください。

Pod起動分析

> **WhaTapノードエージェントバージョン1.4.2未満**
>
> - 新規Pod開始分析 メニューの代わりに既存のメニュー(旧**Pod初期化性能**) 画面を提供します。
> - クラスタープロジェクト(CP)の場合のみアクセスできます。

Springbootを含むJVMベースのアプリケーションは、初期化時に大量のリソースを使用して、Livenessチェックに失敗する場合があります。もし、一定回数以上Livenessチェックに失敗した場合、Podが削除され、無制限に再起動現象が発生し、デプロイが失敗する可能性があります。

WhaTap KubernetesのPod起動分析機能を活用すると、Pod起動時の最大使用量と初期化時間を分析し、最適なInitial Delay(Livenessチェック除外時間)設定に効果的です。

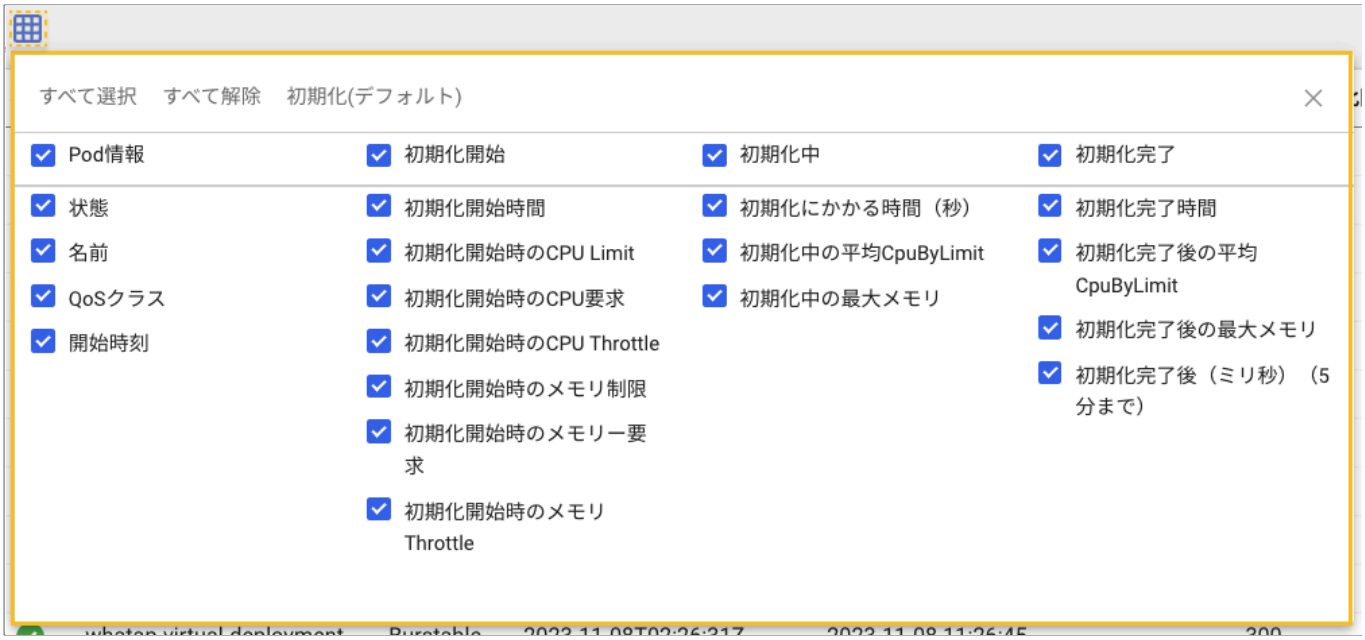
基本画面ガイド



状態	名前	QoSクラス	開始時刻	初期化開始時間	初期化開始時...	初期化開始時...	初期化開始時...	初期化開始時...	初期化開始時...	初期化開始時...
	whatap-node-agent-lvlhx	Burstable	2023-11-08T02:28:16Z	2023-11-08 11:29:30	400	200	31	700 M	400 M	0
	whatap-node-agent-hqcg2	Burstable	2023-11-02T05:09:16Z	2023-11-02 14:09:55	400	200	69	700 M	400 M	0
	whatap-node-agent-4tbg5	Burstable	2023-11-08T02:28:16Z	2023-11-08 11:29:30	400	200	40	700 M	400 M	0
	whatap-master-agent-7c76...	Burstable	2023-11-08T02:27:39Z	2023-11-08 11:28:35	200	100	501	350 M	300 M	0
	whatap-virtual-deployment-...	Burstable	2023-11-08T02:26:13Z		300	50	0	256 M	256 M	0
	whatap-virtual-deployment-...	Burstable	2023-11-08T02:26:13Z		300	50	0	256 M	256 M	0
	whatap-virtual-deployment-...	Burstable	2023-11-08T02:26:13Z		300	50	0	256 M	256 M	0
	whatap-virtual-deployment-...	Burstable	2023-11-08T02:26:19Z		300	50	0	256 M	256 M	0
	whatap-virtual-deployment-...	Burstable	2023-11-08T02:26:18Z		300	50	0	256 M	256 M	0
	whatap-virtual-deployment-...	Burstable	2023-11-08T02:26:24Z	2023-11-08 11:26:45	300	50	23	256 M	256 M	0
	whatap-virtual-deployment-...	Burstable	2023-11-08T02:26:31Z	2023-11-08 11:26:45	300	50	12	256 M	256 M	0
	whatap-virtual-deployment-...	Burstable	2023-11-08T02:26:24Z	2023-11-08 11:26:45	300	50	9	256 M	256 M	0
	whatap-virtual-deployment-...	Burstable	2023-11-08T02:26:31Z	2023-11-08 11:26:45	300	50	31	256 M	256 M	0
	whatap-virtual-deployment-...	Burstable	2023-11-08T02:26:30Z	2023-11-08 11:26:45	300	50	14	256 M	256 M	0
	whatap-virtual-deployment-...	Burstable	2023-11-08T02:26:31Z	2023-11-08 11:26:45	300	50	31	256 M	256 M	0
	whatap-virtual-deployment-...	Burstable	2023-11-08T02:26:24Z	2023-11-08 11:26:45	300	50	12	256 M	256 M	0
	whatap-virtual-deployment-...	Burstable	2023-11-08T02:26:24Z	2023-11-08 11:26:45	300	50	11	256 M	256 M	0
	whatap-virtual-deployment-...	Burstable	2023-11-08T02:26:31Z	2023-11-08 11:26:45	300	50	17	256 M	256 M	0
	whatap-virtual-deployment-...	Burstable	2023-11-08T02:26:24Z	2023-11-08 11:26:45	300	50	20	256 M	256 M	0
	testing-http-sender-58d44d...	BestEffort	2023-11-02T05:21:29Z	2023-11-02 14:24:40	0	0	0	0	0	0

- 1領域の左上にある状態別のPod数を確認することができます。
- 1領域の右上にある 更新アイコンを選択して更新できます。
- 1領域の左側にある カラム選択アイコンを選択してカラムを選択できます。
- 1領域の右側にある入力ウィンドウで、Pod**名前**または**QoSクラス**を検索できます。
- 2領域では、目的のカラムを選択すると、そのカラム一覧を並べ替えて表示することができます。

カラムガイド



Pod起動分析のリストは次のとおり、**Pod情報**とPod初期化のタイムラインによりカラムを分類します。

Pod情報

Podの**名前**と**状態**、**QoSクラス**、**開始時刻**に関する情報を確認することができます。

- **状態**: Podのライフサイクルを意味します。
  - ⏸ Pending: クバネティスクラスター内でPod開始が承認されているが、コンテナが生成されていない状態を意味します。
  - ✅ Running: Podがノードにバインディングされ、すべてのコンテナが生成され、1 つ以上のコンテナがまだ実行中であるか、開始または再起動中である状態を意味します。
  - ☑Succeeded: Podのすべてのコンテナが正常に終了し、再起動されていない状態を意味します。
  - ⚠Failed: Podのコンテナの1 つ以上で障害が発生し、終了した状態を意味します。
  - ⑦Unknown: 何らかの理由でPodと通信できない状態を意味します。
- **QoSクラス**: PodのQuality of Serviceを意味します。QoSクラスは、リソース不足時のPodの優先順位を決定しま

す。優先度の高いPodが優先度の低いPodより先に終了します。

- Guaranteed: ノードで使用可能なリソースが保証され、他のPodよりも優先的に割り当てます。最も最後に終了します。
  - Podのすべてのコンテナが CPU および メモリの request を持っている場合です。
  - Podのすべてのコンテナが CPU および メモリに limit を持っており、それらの値が request と同じである場合です。
- Burstable: Podはノードのリソースが不足している場合は、一部のリソースを割り当てます。
  - Guaranteed クラスの条件を満たしていない場合です。
  - Podのコンテナのいずれかが CPU および メモリに request または limit が設定されている場合です。
- BestEffort: リソース requestやlimitがないため、他のPodより優先度が低くなります。 最初に終了します。
  - Podのすべてのコンテナには、CPU および メモリの request と limit はありません。

- **開始時刻**: Podが開始または生成された時間を意味します。

初期化開始

初期化開始は、最初のメトリクスデータが収集された日時を意味します。

- **初期化開始時刻**: 初期化が開始された時間に関するデータで、最初のデータ収集時間を意味します。
- **初期化開始時のCPU Limit**: 初期化の開始時のCPU limitを意味します。
- **初期化開始時のCPU Request**: 初期化の開始時のCPU requestを意味します。
- **初期化開始時のCPU Throttle Count**: 初期化の開始時にコンテナでCPU制限を超える使用試行が発生した場合、増加する値を意味します。
- **初期化開始時のメモリ Limit**: 初期化の開始時に設定したメモリ limitを意味します。
- **初期化開始時のメモリ Request**: 初期化の開始時に要求したメモリ requestを意味します。
- **初期化開始時のメモリ Throttle Count**: 初期化の開始時にコンテナでメモリ制限を超える使用試行が発生した場合、増加する値を意味します。

初期化中

初期化中は、初期化の開始後5分間のデータのCpuByLimitの平均値より低いCpuByLimit値のデータが発生した以前の

時点を意味します。

- **初期化にかかる時間(秒)**: Pod初期化にかかった時間を意味します。

  初期化開始時の5分間のデータのうち、CpuByLimitの平均値より低いCpuByLimit値が収集された時間 - 最初のデータ収集時間

- **初期化中の平均CpuByLimit**: 初期化中に使用したCpuByLimitの平均値を意味します。
- **初期化中の最大メモリ**: 初期化中にそのコンテナで使用されたメモリの最大使用量を意味します。

> ⓘ CpuByLimit(`cpu_per_quota`)は、CPU Limitに基づくCPU全体の使用率を意味します。

初期化完了

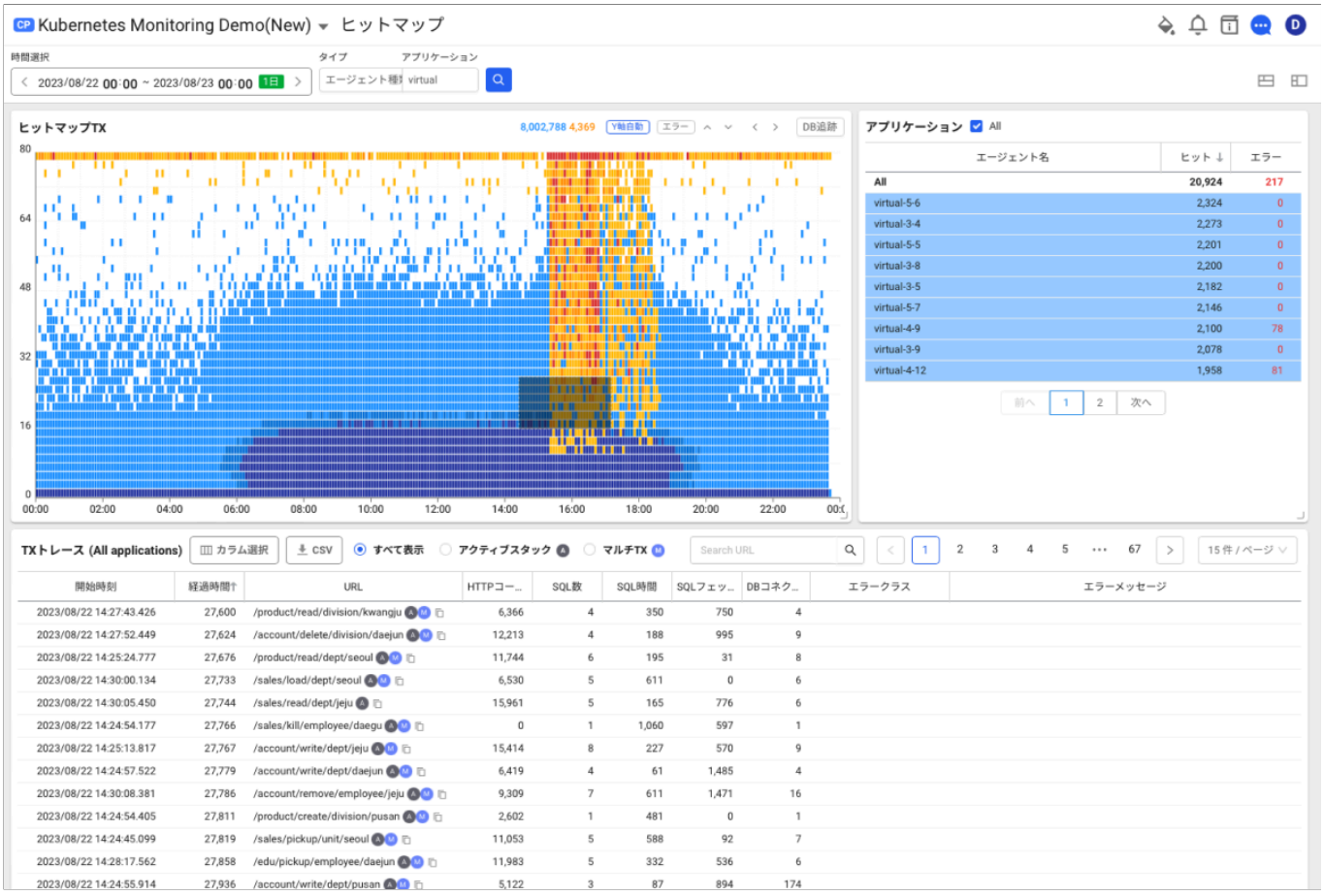
初期化完了とは、初期化開始後5分間のデータのCpuByLimit平均値より低いCpuByLimit値のデータが発生した時間を意味します。

- **初期化完了時刻**: 初期化完了した時間を意味します。
- **初期化完了後の平均CpuByLimit**: 初期化完了後に使用したCpuByLimitの平均値を意味します。
- **初期化完了後の最大メモリ**: 初期化が完了した後、そのコンテナで使用したメモリの最大使用量を意味します。
- **初期化完了後(秒)(5分まで)**: 初期化完了後5分間のミリ秒単位時間を意味します。

アプリケーション ヒットマップ

コンテナ内のアプリケーションのヒットマップ情報を確認できます。ヒットマップは、トランザクション実行件数マップを含むレスポンス時間分布図を表示します。検索条件は、時間選択、分類およびアプリケーションに分けられます。

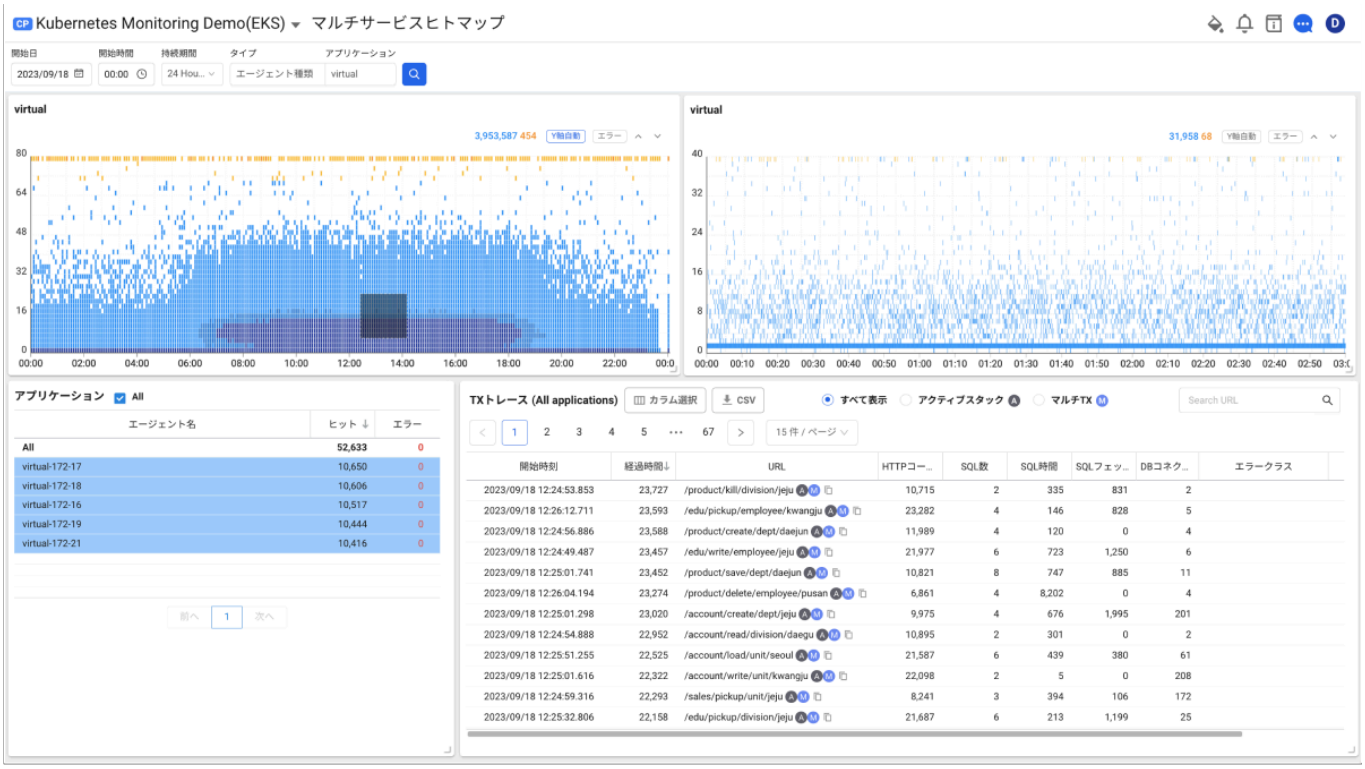
分類	説明
エージェント	コンテナ内部にインストールされているアプリケーションエージェント名
エージェント種類	エージェントがグループ化されたエージェントの種類(OKIND)
エージェントサーバー	エージェントが実行されているKubernetesノード名



ヒットマップの特定区分をドラッグすると、トレース情報が確認できます。トレースデータは、URLベースの検索が可能であり、アクティブスタックが存在するトレースは、マルチトランザクションで行うトレース情報をフィルタリングして表示できます。

> ⓘ ヒットマップの詳細については、次の文書を参照してください。
>
> - Java
> - Node.js
> - Python

マルチサービスヒットマップ



複数のサービスを対象にヒットマップチャートを確認し、サービス別の状態を比較分析できます。

検索条件は次のとおりです。

項目	説明
開始日	検索日
開始時間	3日、7日、1か月の検索では使用できません。
期間	• 5分/10分/30分 • 1時間/3時間/24時間 • 3日/7日/1か月

項目	説明
区分	エージェント種類
アプリケーション	アプリケーション

ヒットマップの詳細については、次の文書を参照してください。

- Java
- Node.js
- Python

MSA分析

MSA分析とは？

MSA分析機能は、WhaTapの**マルチTX追跡**機能を使用して収集されたトランザクション間の呼び出しの割合を分析します。 これによって、マイクロサービス間の依存性をインスタンス中心ではなくURLレベルで確認できます。



例えば、A、B、C、D、Eという5つのマイクロサービスがあります。ユーザーがAを呼び出すと、ロジックに従って次のような3つの呼び出しパターンが発生すると仮定します。

1. AからBとCを呼び出し
2. AからBとCを呼び出し + またBからDを呼び出し
3. AからBとCを呼び出し + またBからDとEを呼び出し

この3つのパターンが一度ずつ実行された場合、MSA分析では基準URLに従って関連度分析のデータが確認できます。

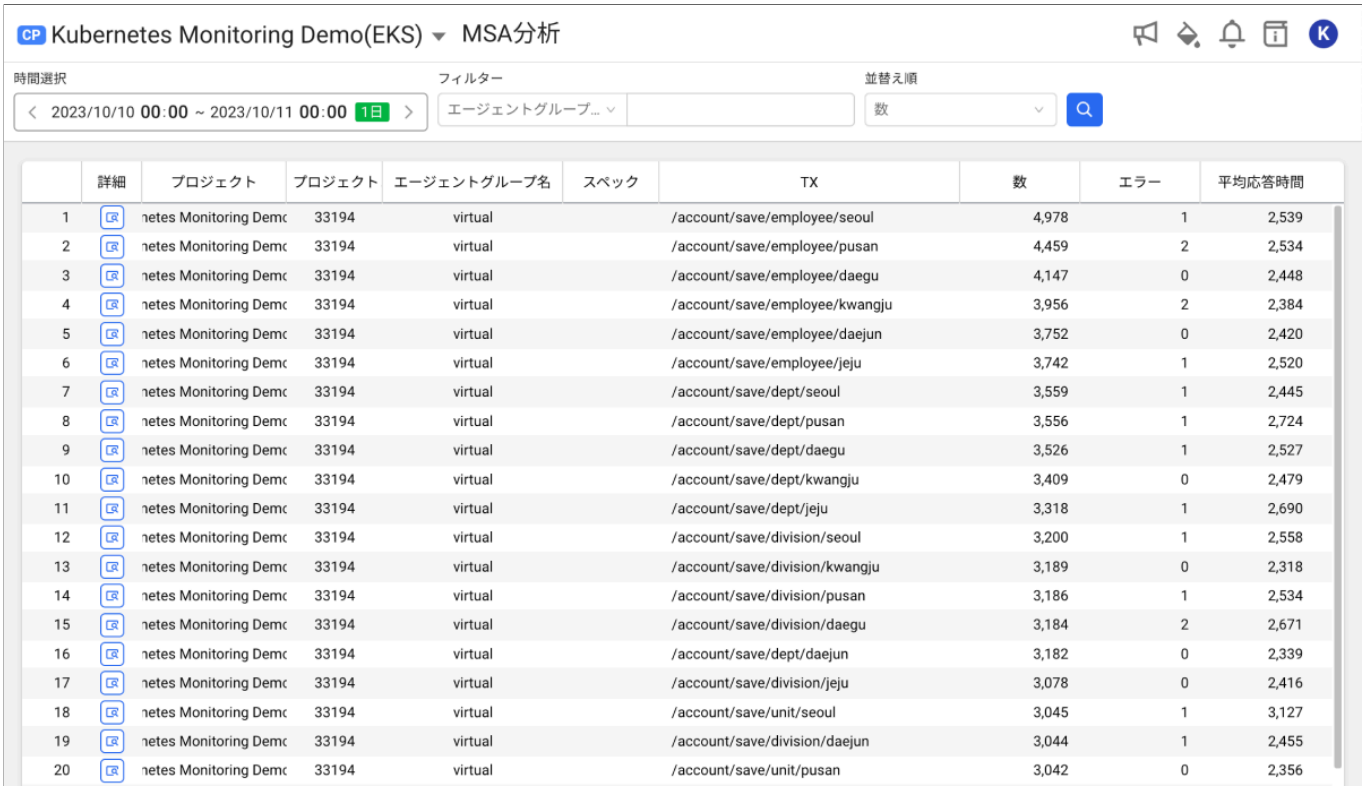


> **マルチトランザクション追跡**に関する詳細は、次の文書を参照してください。

MSAトランザクションの統計

ホーム画面 > プロジェクト選択 > **分析** > **MSA分析**

MSA分析メニューは、マイクロサービス環境で各サービスの性能と安全性を効果的に分析できるツールです。 トランザクションの詳細な呼び出し関係を分析し、これを基づいてシステムの性能を最適化することで、潜在的な問題を事前に識別できます。



	詳細	プロジェクト	プロジェクト	エージェントグループ名	スペック	TX	数	エラー	平均応答時間
1		netes Monitoring Demc	33194	virtual		/account/save/employee/seoul	4,978	1	2,539
2		netes Monitoring Demc	33194	virtual		/account/save/employee/pusan	4,459	2	2,534
3		netes Monitoring Demc	33194	virtual		/account/save/employee/daegu	4,147	0	2,448
4		netes Monitoring Demc	33194	virtual		/account/save/employee/kwangju	3,956	2	2,384
5		netes Monitoring Demc	33194	virtual		/account/save/employee/daejun	3,752	0	2,420
6		netes Monitoring Demc	33194	virtual		/account/save/employee/jeju	3,742	1	2,520
7		netes Monitoring Demc	33194	virtual		/account/save/dept/seoul	3,559	1	2,445
8		netes Monitoring Demc	33194	virtual		/account/save/dept/pusan	3,556	1	2,724
9		netes Monitoring Demc	33194	virtual		/account/save/dept/daegu	3,526	1	2,527
10		netes Monitoring Demc	33194	virtual		/account/save/dept/kwangju	3,409	0	2,479
11		netes Monitoring Demc	33194	virtual		/account/save/dept/jeju	3,318	1	2,690
12		netes Monitoring Demc	33194	virtual		/account/save/division/seoul	3,200	1	2,558
13		netes Monitoring Demc	33194	virtual		/account/save/division/kwangju	3,189	0	2,318
14		netes Monitoring Demc	33194	virtual		/account/save/division/pusan	3,186	1	2,534
15		netes Monitoring Demc	33194	virtual		/account/save/division/daegu	3,184	2	2,671
16		netes Monitoring Demc	33194	virtual		/account/save/dept/daejun	3,182	0	2,339
17		netes Monitoring Demc	33194	virtual		/account/save/division/jeju	3,078	0	2,416
18		netes Monitoring Demc	33194	virtual		/account/save/unit/seoul	3,045	1	3,127
19		netes Monitoring Demc	33194	virtual		/account/save/division/daejun	3,044	1	2,455
20		netes Monitoring Demc	33194	virtual		/account/save/unit/pusan	3,042	0	2,356

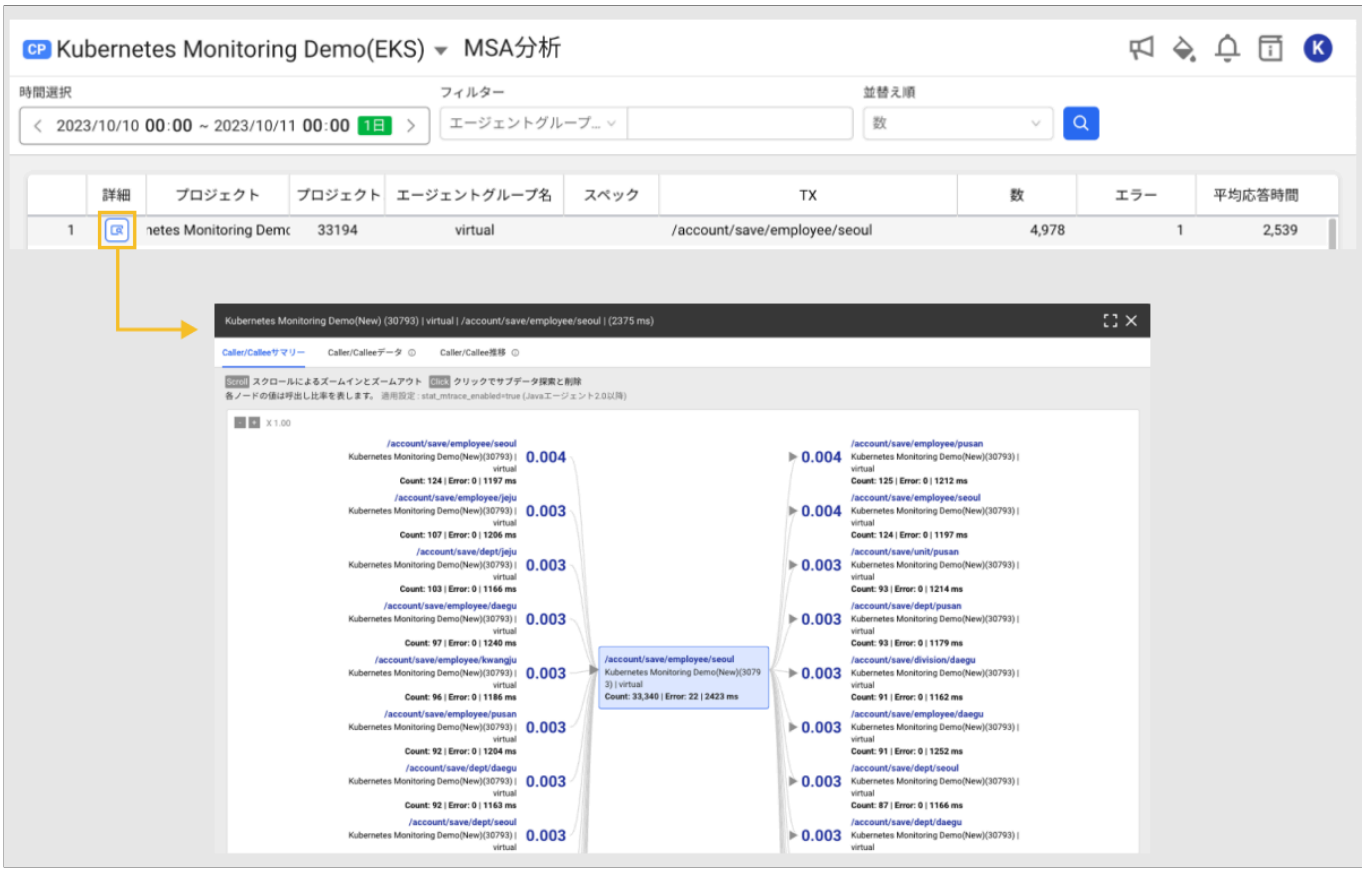
- **時間選択**: 照会期間を選択します。 基本1日を基準に、Caller/Calleeを含むトランザクションの一覧を照会できます。
- **フィルター**: **エージェントグループ名**, **スペック**, **TX** 基準でフィルタリングし、検索できます。
- **並替え順**: **数**, **エラー**, **平均時間**(平均の応答時間) 基準で一覧をソートできます。

> ⓘ
> - **時間選択**で照会できる時間範囲は、最大1日(24時間)となります。 **時間選択**オプションの利用方法の詳細については。
> - **エージェントグループ名**は、エージェント設定オプションの'whatap.okind'に分類されたグループ名です。
> - 条件を設定してからトランザクション一覧を更新するには、ボタンを選択します。

詳細分析する

トランザクション一覧で、照会する項目のボタンを選択すると、詳細表示画面が表示されます。 詳細表示画面で、

該当トランザクションに対する呼び出し元(Caller)、呼び出し先(Callee) 情報を照会できます。



各タブでは、次の分析機能を提供します。

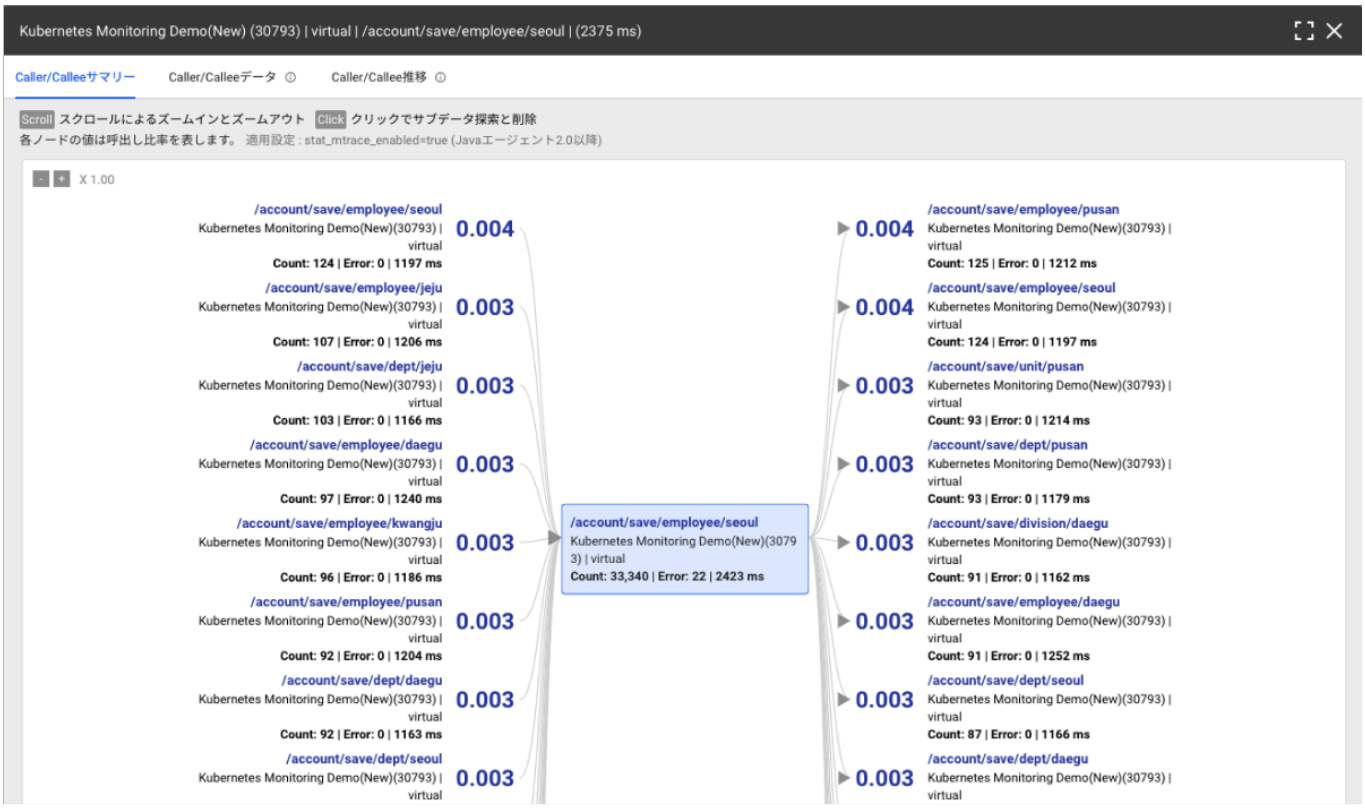
- **Caller/Calleeサマリー**: トランザクションの呼び出し元と呼び出し先の間の関係を、視覚化されたダイアグラムで提供します。
- **Caller/Calleeデータ**: トランザクションの呼び出し回数、エラー数、応答時間を詳細に提供します。
- **Caller/Callee推移**: 時間によるトランザクションの呼び出し回数とレートの性能変化を、時系列チャートで提供します。

> ⓘ
> - 詳細画面の右上にある⛶ボタンを選択すると、全画面で確認できます。
> - 詳細画面を閉じるには、右下にある**確認**ボタンを選択するか、ESCキーを押します。

Caller/Calleeサマリー

Caller/Calleeサマリータブでは、トランザクション一覧で選択したトランザクションのCaller、Callee間の依存性情報を視覚化されたダイアグラムで表示します。 サービス間の呼び出し関係を、直観的に理解できます。 呼び出し経路と従属性が視覚的に把握でき、問題の原因分析に便利です。

呼び出し関係の分析



CallerとCalleeノードには、最大0.004、0.003など最大1人の数字情報を表示しますが、これは全体のCallerの中で占める割合を意味します。 同様に各Calleeの比重は、右側に一緒に表示します。

上の画面から_/account/save/employee/seoul_というトランザクションを基準に、CallerとCallee間の依存関係を分析できます。 トランザクションがどのサービスから呼び出され、どのサービスに呼び出されるかを視覚的に確認できます。

性能指標の確認

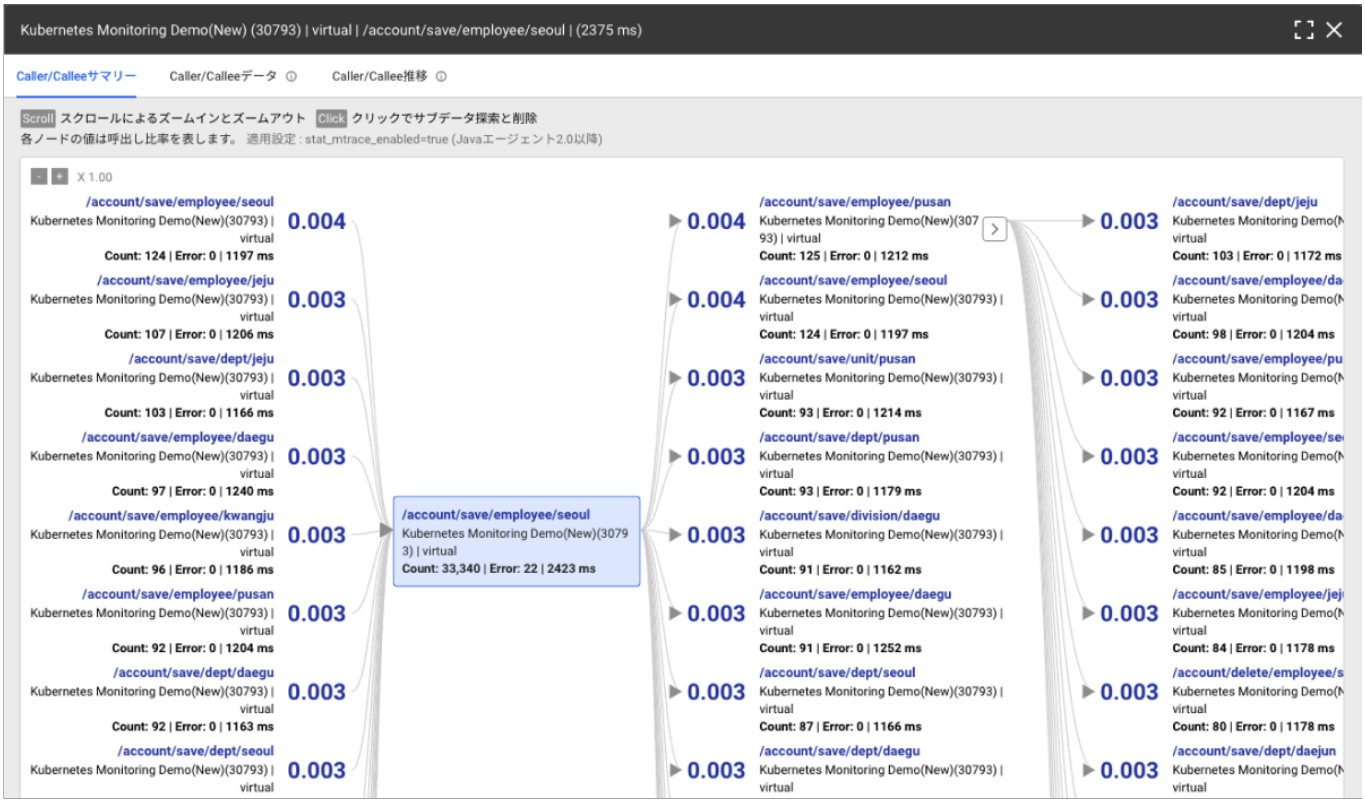


各ノードはトランザクションURL、プロジェクト名、エージェントのグループ名、そして次の性能情報で構成されています。

- **Count**: 当該トランザクションを呼び出した回数
- **Error**: 呼び出し中に発生したエラー数
- 平均応答時間: 当該トランザクションの平均の応答時間、ミリ秒(ms) 単位

トランザクションの流れを把握

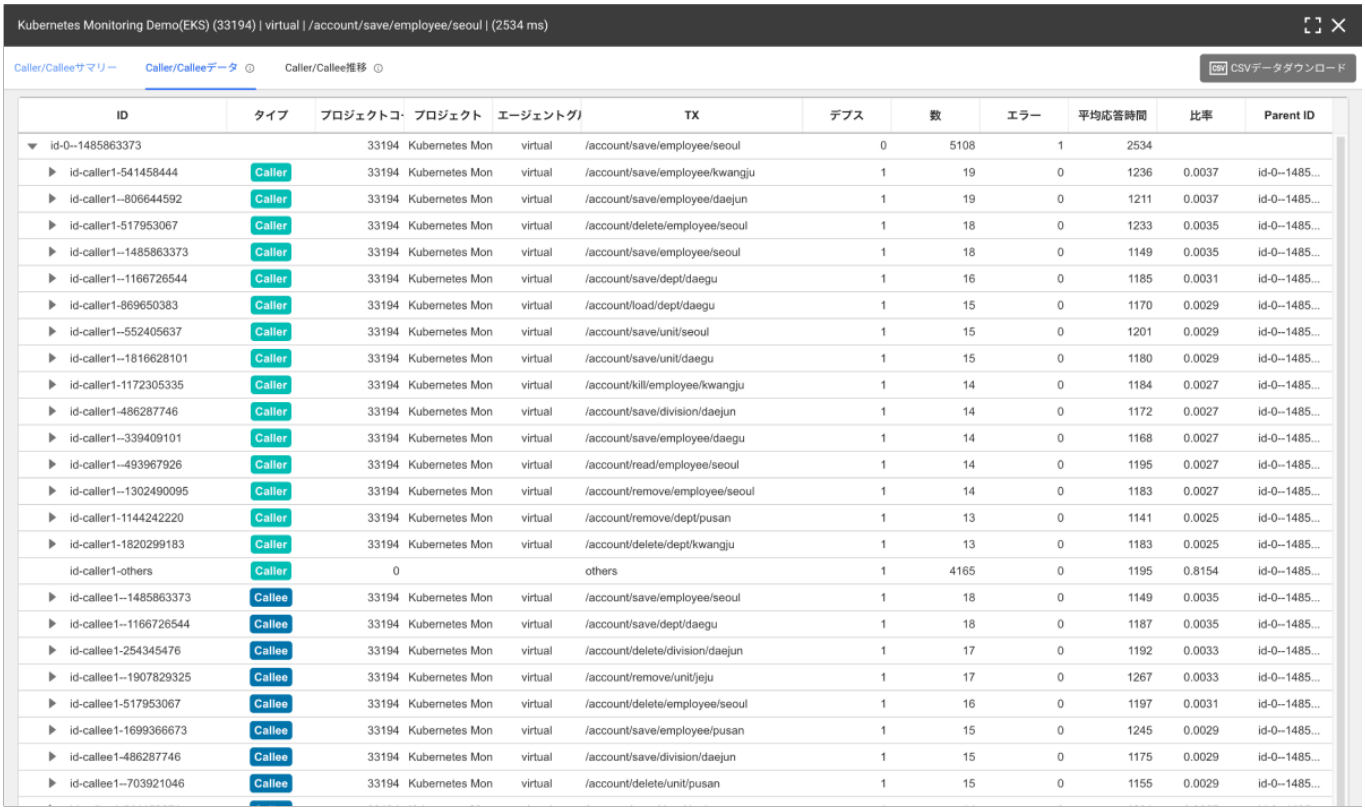
CallerやCalleeノードにマウスをオーバーすると、 > ボタンが表示されます。 このボタンを選択すると、下位ノードの詳細に展開できます。 トランザクションの全体的な流れを把握して、問題の発生元を見つけるのに役に立ちます。



- 下位ノードを閉じるには、もう一度 > ボタンを選択します。
- マウスのスクロール機能により、画面を拡大または縮小できます。 または画面左上の+ / -ボタンをクリックします。

Caller/Calleeデータ

Caller/Calleeデータタブでは、基準URL(例: /account/save/employee/seoul)に対する呼び出し元(Caller)と呼び出し先(Callee)の相関関係を、表の形式で確認できます。 各トランザクションの詳細情報(呼び出し回数、エラー数、平均応答時間など)が具体的に確認でき、データ基盤で精密に分析できます。

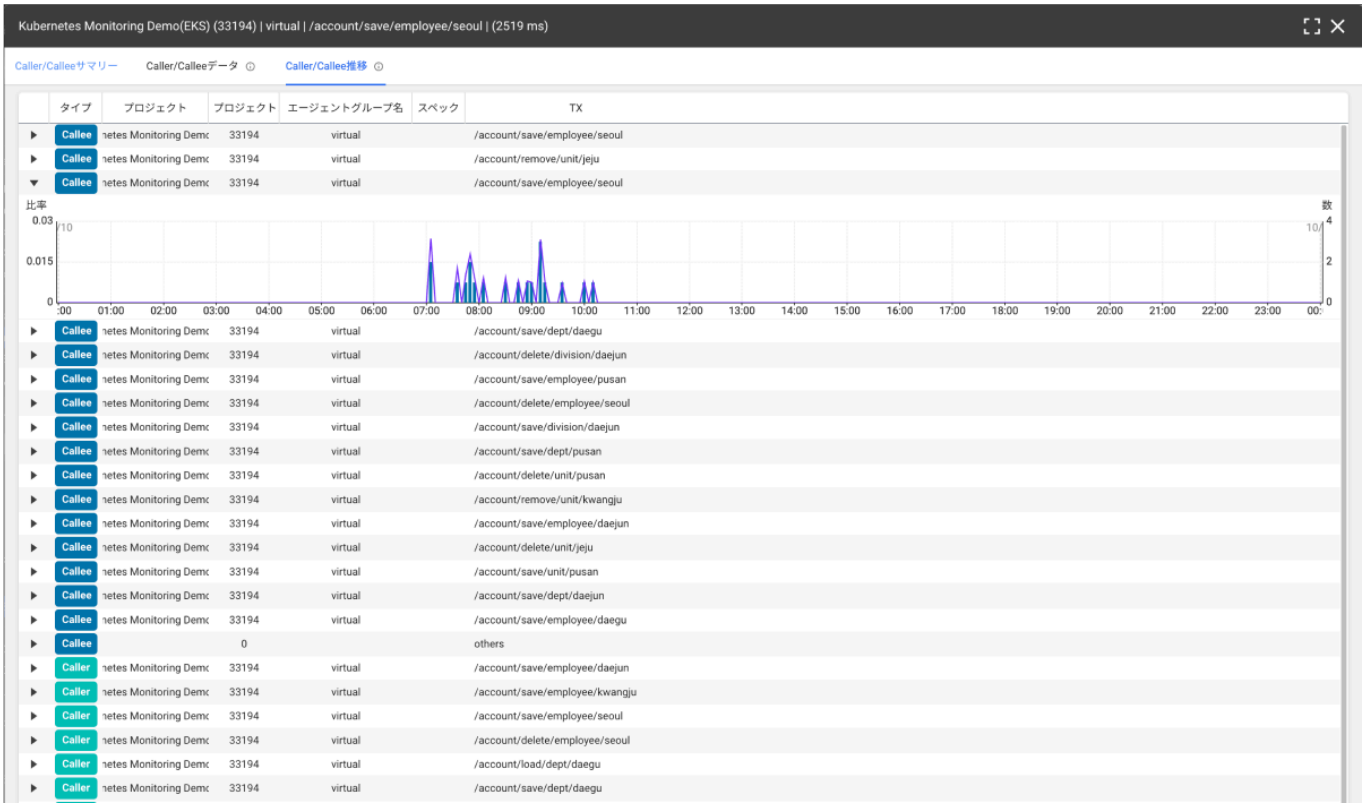


ID	タイプ	プロジェクトコ	プロジェクト	エージェントグ	TX	デプス	数	エラー	平均応答時間	比率	Parent ID
id-0--1485863373		33194	Kubernetes Mon	virtual	/account/save/employee/seoul	0	5108	1	2534		
id-caller1-541458444	Caller	33194	Kubernetes Mon	virtual	/account/save/employee/kwangju	1	19	0	1236	0.0037	id-0--1485...
id-caller1--806644592	Caller	33194	Kubernetes Mon	virtual	/account/save/employee/daejun	1	19	0	1211	0.0037	id-0--1485...
id-caller1-517953067	Caller	33194	Kubernetes Mon	virtual	/account/delete/employee/seoul	1	18	0	1233	0.0035	id-0--1485...
id-caller1--1485863373	Caller	33194	Kubernetes Mon	virtual	/account/save/employee/seoul	1	18	0	1149	0.0035	id-0--1485...
id-caller1--1166726544	Caller	33194	Kubernetes Mon	virtual	/account/save/dept/daegu	1	16	0	1185	0.0031	id-0--1485...
id-caller1-869650383	Caller	33194	Kubernetes Mon	virtual	/account/load/dept/daegu	1	15	0	1170	0.0029	id-0--1485...
id-caller1--552405637	Caller	33194	Kubernetes Mon	virtual	/account/save/unit/seoul	1	15	0	1201	0.0029	id-0--1485...
id-caller1--1816628101	Caller	33194	Kubernetes Mon	virtual	/account/save/unit/daegu	1	15	0	1180	0.0029	id-0--1485...
id-caller1-1172305335	Caller	33194	Kubernetes Mon	virtual	/account/kill/employee/kwangju	1	14	0	1184	0.0027	id-0--1485...
id-caller1-486287746	Caller	33194	Kubernetes Mon	virtual	/account/save/division/daejun	1	14	0	1172	0.0027	id-0--1485...
id-caller1--339409101	Caller	33194	Kubernetes Mon	virtual	/account/save/employee/daegu	1	14	0	1168	0.0027	id-0--1485...
id-caller1--493967926	Caller	33194	Kubernetes Mon	virtual	/account/read/employee/seoul	1	14	0	1195	0.0027	id-0--1485...
id-caller1--1302490095	Caller	33194	Kubernetes Mon	virtual	/account/remove/employee/seoul	1	14	0	1183	0.0027	id-0--1485...
id-caller1-1144242220	Caller	33194	Kubernetes Mon	virtual	/account/remove/dept/pusan	1	13	0	1141	0.0025	id-0--1485...
id-caller1-1820299183	Caller	33194	Kubernetes Mon	virtual	/account/delete/dept/kwangju	1	13	0	1183	0.0025	id-0--1485...
id-caller1-others	Caller	0			others	1	4165	0	1195	0.8154	id-0--1485...
id-callee1--1485863373	Callee	33194	Kubernetes Mon	virtual	/account/save/employee/seoul	1	18	0	1149	0.0035	id-0--1485...
id-callee1--1166726544	Callee	33194	Kubernetes Mon	virtual	/account/save/dept/daegu	1	18	0	1187	0.0035	id-0--1485...
id-callee1-254345476	Callee	33194	Kubernetes Mon	virtual	/account/delete/division/daejun	1	17	0	1192	0.0033	id-0--1485...
id-callee1--1907829325	Callee	33194	Kubernetes Mon	virtual	/account/remove/unit/jeju	1	17	0	1267	0.0033	id-0--1485...
id-callee1-517953067	Callee	33194	Kubernetes Mon	virtual	/account/delete/employee/seoul	1	16	0	1197	0.0031	id-0--1485...
id-callee1-1699366673	Callee	33194	Kubernetes Mon	virtual	/account/save/employee/pusan	1	15	0	1245	0.0029	id-0--1485...
id-callee1-486287746	Callee	33194	Kubernetes Mon	virtual	/account/save/division/daejun	1	15	0	1175	0.0029	id-0--1485...
id-callee1--703921046	Callee	33194	Kubernetes Mon	virtual	/account/delete/unit/pusan	1	15	0	1155	0.0029	id-0--1485...

- 各トランザクションの呼び出し回数と割合、エラー数、平均応答時間などを確認し、性能問題を分析するのに活用できます。
- **タイプ**カラムを通じてCallerとCalleeを簡単に区別することができ、呼び出し関係を把握できます。
- 各トランザクションの ▶ ボタンを選択すると、そのトランザクションに関連するトランザクションを、追加で確認できます。
- データ一覧をCSV形式で保存するには、右上の**CSVデータダウンロード**ボタンを選択します。

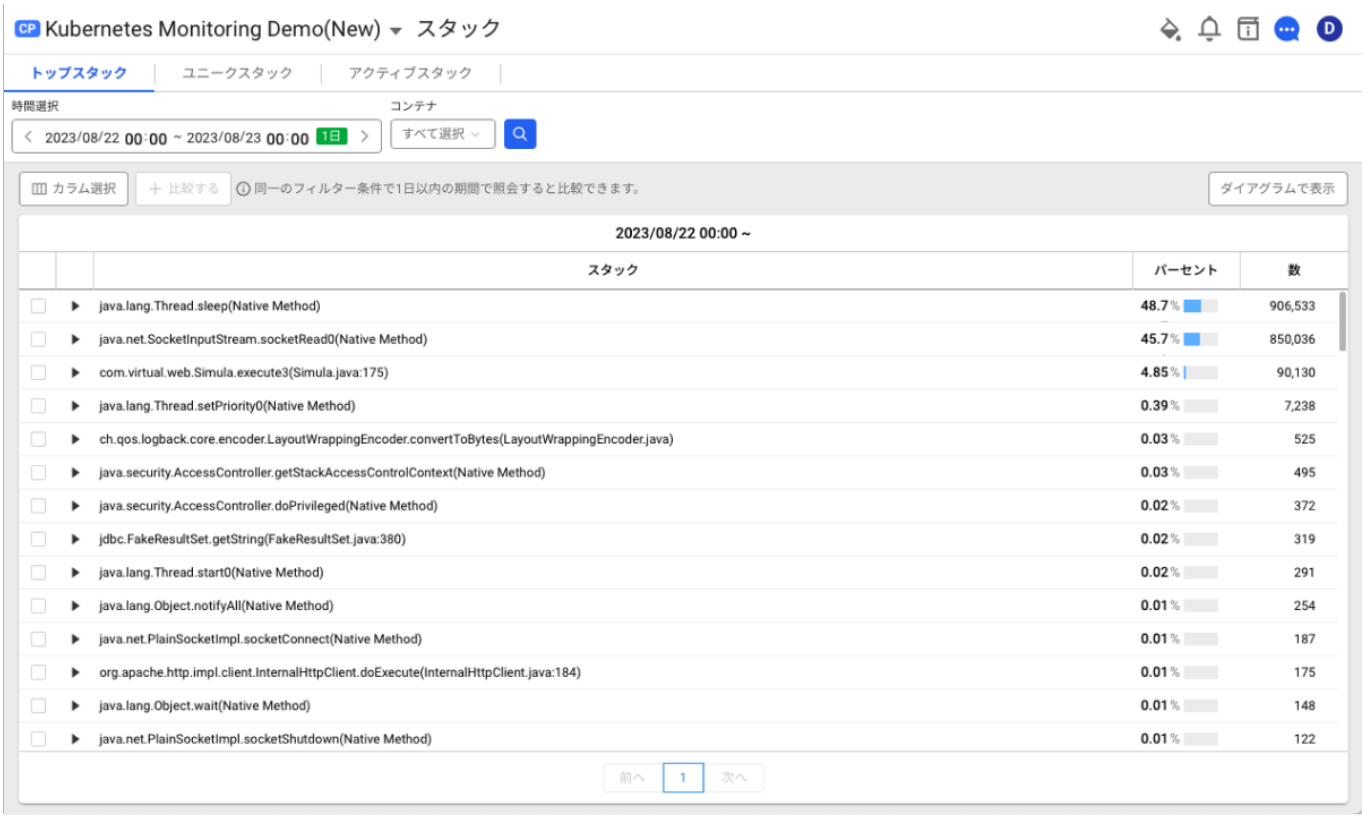
Caller/Callee推移

Caller/Callee推移タブでは、基準URL(/account/save/employee/seoul)の呼び出し関係と性能データを時間の経過とともに視覚的に分析できます。 これにより、トランザクションの性能変動パターンを把握できます。



- 特定のトランザクションの呼び出し回数と割合を、時間帯別グラフで可視化し、性能の変化を一目で把握できます。
- **タイプ**カラムを通じてCallerとCalleeを簡単に区別することができ、呼び出し関係を把握できます。
- 時系列チャートにマウスをオーバーすると、ツールチップを通じて数値情報を確認できます。
- チャートの特定の時間帯をクリックすると、**分析** > **TX検索**メニューへ移動します。 選択した時間帯のトランザクション一覧を照会できます。

アプリケーションスタック



WhaTapは、10秒(既定値) 間隔で収集したThread Stackを利用してメソッドレベルの性能遅延間隔を分析します。この機能を使用するには、各コンテナのトップスタック、ユニークスタック、アクティブスタックを確認することができます。

> ⓘ トップスタック、ユニークスタック、アクティブスタックの詳細については、次の文書を参照してください。
>
> - Java
> - Python

メトリクス

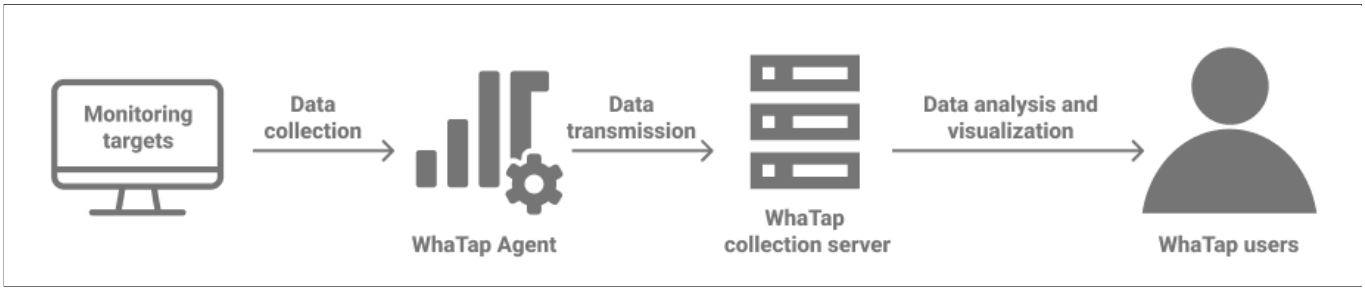
メトリクスとは?

メトリクスはモニタリング対象の状態を数値で表現した時系列データです。WhaTapエージェントがモニタリング対象から収集した性能データを加工した結果です。

メトリクスを通じてサーバー別メモリ使用率、DB接続時間などの主要な性能指標を一目で把握できます。問題を発見した後はログとトレースを通じて詳細分析ができます。

リソース使用量の統計を基に必要なリソース量を算定でき、コスト効率化にも活用できます。

WhaTapのメトリクス収集方式



メトリクスを収集するにはモニタリング対象にWhaTapエージェントをインストールする必要があります。エージェントが送信したメトリクスはWhaTap収集サーバーに保存されます。

例えば**Kubernetes**モニタリングを開始するにはまずエージェントをインストールしてください。収集可能なメトリクスはKubernetesメトリクスで確認できます。

WhaTapのメトリクス構成要素

WhaTapの**メトリクス**は以下の要素で構成されています。

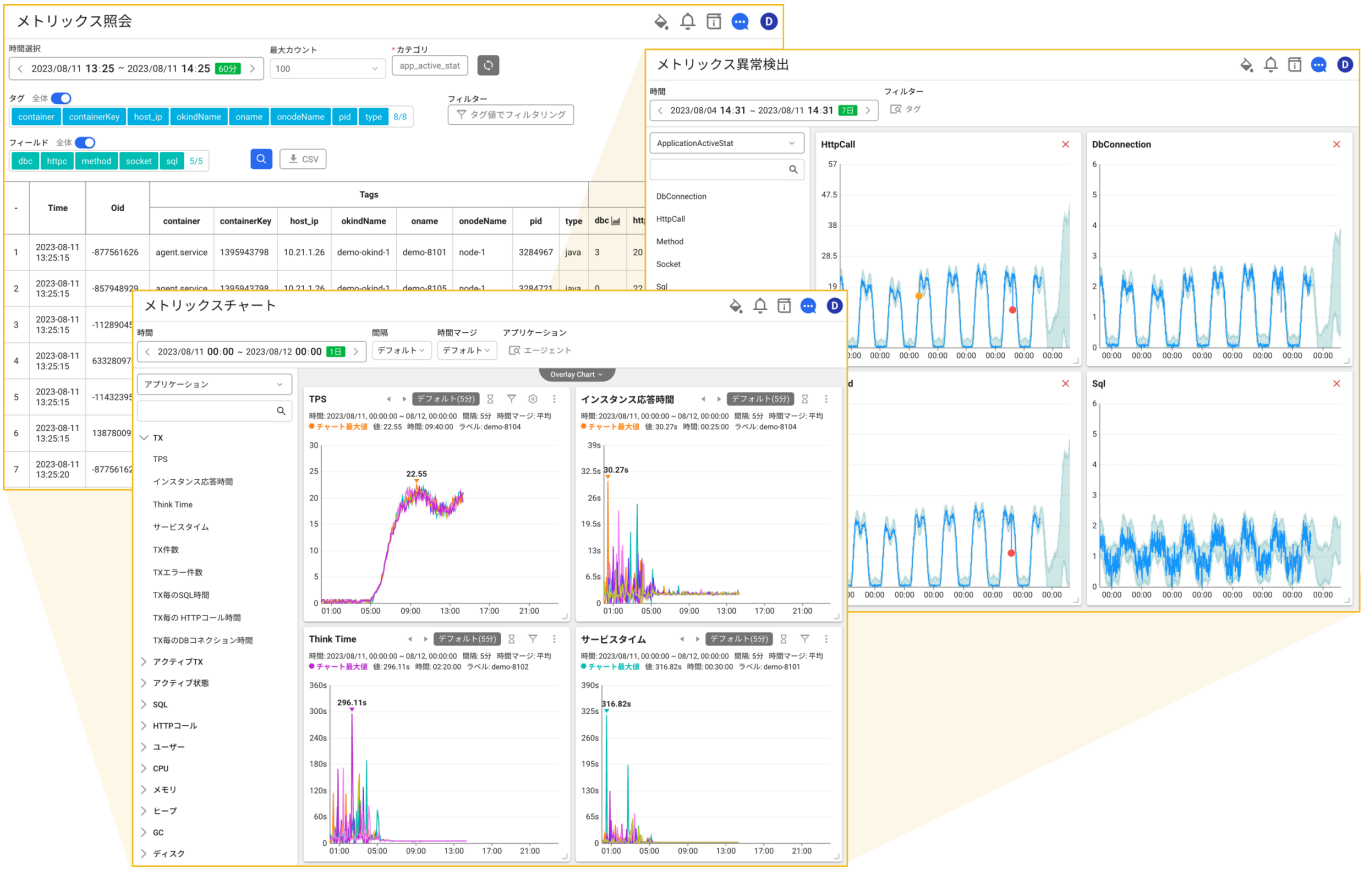
- **Category**: 関連メトリクスを1つのグループにまとめる名前です。

- **Tags**: 収集対象を識別するラベルです。IP、Oname、Hostなど変更の少ない固有情報を保存します。1つのメトリクスに複数のタグが含まれることがあります。
- **Fields**: 実際の測定値を保存します。1つのメトリクスに複数のフィールドが含まれることがあります。
- **Time**: メトリクスを収集した時間です。
- **Oid**: メトリクスを収集したエージェントの固有番号です。
- **Oname**: メトリクスを収集したエージェントの名前です。

メトリクスデータの照会と可視化

WhaTapは収集したメトリクスをさまざまな方式で照会できるメニューを提供します。

- **メトリクス照会**: メトリクスの原本データを照会します。
- **メトリクスチャート**: 可視化したチャートを通じてメトリクスデータを照会します。
- **メトリクス異常検出**: AIが学習したメトリクスパターンと比較して予想パターンを外れた異常を探知します。


メトリックス照会
メトリックス異常検出
メトリックスチャート
TPS
インスタンス応答時間
Think Time
サービスタイム
HttpCall
DbConnection
Sql

Kubernetesメトリクス

Kubernetesモニタリングが収集するメトリクスをカテゴリ別に整理しました。各カテゴリのTags(ディメンション)とFields(測定値)を確認できます。

- 一部のメトリクスはサーバーモニタリング、アプリケーションモニタリングと共通です。
- メトリクスを直接検索したり、チャートで可視化するにはメトリクス照会、メトリクスチャートページを参照してください。

コンテナ(container) メトリクス

container カテゴリは、コンテナのPODに設定されているすべてのユーザー定義ラベルをタグとして収集します。

- 対象：クラスタープロジェクト、ネームスペースプロジェクト
- 収集間隔：5秒
- 統計データ：5分

Tags

Tag	Type	Unit	Description
agentOid	-	-	ノードエージェント固有ID(固有値)
agentPcode	-	-	プロジェクトコード(固有値)
command	-	-	実行コマンド
containerId	-	-	コンテナID(固有値)
containerKey	-	-	コンテナKey
created	-	-	コンテナが生成されたTimeStamp

Tag	Type	Unit	Description
image	-	-	コンテナイメージ名
imageHash	-	-	イメージハッシュ値
imageId	-	-	イメージID
k8s-app	-	-	Podのラベルk8s-appに対する値
microOid	-	-	コンテナにインストールされたWhaTap APMエージェントの固有ID
name	-	-	コンテナ名
namespace	-	-	コンテナが所属するネームスペース
namespaceHash	-	-	コンテナが所属するネームスペースのハッシュ値
okind	-	-	コンテナにインストールされているWhaTap APMエージェントに指定したOKINDの固有ID
okindName	-	-	コンテナにインストールされているWhaTap APMエージェントに指定したOKIND名
oname	-	-	コンテナにインストールされているWhaTap APMエージェント名
onode	-	-	コンテナが実行されているノードエージェントの固有ID
onodeName	-	-	コンテナが動作しているノード名
podHash	-	-	コンテナのPodハッシュ値
podName	-	-	コンテナのPod名
replicaSetHash	-	-	コンテナのレプリカセットハッシュ値
replicaSetName	-	-	コンテナのレプリカセット名

Tag	Type	Unit	Description
whatap_project	-	-	コンテナが属するWhaTapプロジェクト名

Fields

Field	Type	Unit	Shortname/Name/Description
blkio_rbps	-	バイト	IoReadBytes
			Container Block I/O Read Byte
			コンテナ全体ブロックデバイスの1秒当たり読み込んだバイトの合計
blkio_riops	-	count	IoReadIops
			Container Block I/O Read IOPS
			コンテナ全体ブロックデバイスの1秒当たり読み込んだ数の合計
blkio_wbps	-	バイト	IoWriteBytes
			Container Block I/O Write Byte
			コンテナ全体ブロックデバイスの1秒当たり書き込んだバイトの合計
blkio_wiops	-	count	IoWriteIops
			Container Block I/O Write IOPS
			コンテナ全体ブロックデバイスの1秒当たり書き込んだ数の合計
cpu_per_quota	-	percent	CpuByLimit

Field	Type	Unit	Shortname/Name/Description
			Container CPU Usage by Limit(%)
			CPU Limit基準CPUの全体使用率
cpu_quota	-	millicores	CpuLimit
			Container CPU Limit (core)
			コンテナのCPU Limit割当量 - Limitが未設定の場合、コンテナが実行中のノードのCPU全体コアがミリコア単位で表示
cpu_quota_percent	-	percent	CpuLimitByNode
			Container CPU Limit by Node(%)
			ノードCPUに対するコンテナCPUリミットの割り当て量 - リミットが設定されていない場合、コンテナが動作しているノードのCPU全コアがパーセントで表示されます。
cpu_sys	-	percent	CpuSysByNode
			Container CPU Sys Usage by Node(%)
			ノードCPUに対するコンテナCPU System使用率
cpu_throttledperiods	-	count	CpuThrottledCnt
			Container CPU Throttling Count
			コンテナCPU Throttled数
cpu_throttledtime	-	nanosecond(ns)	CpuThrottledTime
			Container CPU Throttling Time

Field	Type	Unit	Shortname/Name/Description
			コンテナCPU Throttled時間
cpu_total	-	percent	CpuByNode
			Container CPU Usage by Node(%)
			ノードCPUに対するコンテナCPU使用率
cpu_total_milli	-	millicores	CpuTotUsage
			Container CPU Usage (millicore)
			コンテナCPU使用量
cpu_user	-	percent	CpuUserByNode
			Container CPU User Usage by Node(%)
			ノードCPUに対するコンテナCPU User使用率
cpu_request	-	millicores	CpuRequest
			Container CPU Request (core)
			コンテナCPU要求
cpu_per_request	-	percent	CpuByRequest
			Container CPU Usage by Request(%)
			コンテナCPUリクエストに対する使用率 = cpu_total_milli / cpu_request * 100
mem_failcnt	-	count	MemFailCnt
			Container Memory Failure Count

Field	Type	Unit	Shortname/Name/Description
			コンテナメモリLimit到達件数
mem_limit	-	バイト	MemLimit
			Container Memory Limit (byte)
			コンテナのメモリLimitサイズ
mem_maxusage	-	バイト	MemMaxUsage
			Container Memory Max Usage (byte)
			コンテナメモリの最大使用量の記録値 - 詳細は下記案内を参照してください。
mem_percent	-	percent	MemWsByLimit
			Container Memory Working Set by Limit(%)
			コンテナメモリLimit基準working set使用量 = mem_working_set / mem_limit * 100
mem_totalcache	-	バイト	MemTotCache
			Container Memory Total Cache (byte)
			コンテナ全体のキャッシュサイズ
mem_totalpgfault	-	count	MemTotPageFaultCnt
			Container Memory Total Page Fault Count
			コンテナPage Fault回数
mem_totalrss	-	バイト	MemTotRss

Field	Type	Unit	Shortname/Name/Description
			Container Memory Total RSS (byte)
			コンテナRSSメモリ全体サイズ
mem_totalrss_percent	-	percent	MemTotRssByLimit
			Container Memory Total RSS by Limit (%)
			コンテナRSSメモリ全体使用率
mem_totalunevictable	-	バイト	MemTotUnevictable
			Container Memory Total Unevictable (byte)
			コンテナUnevictable Memory全体サイズ
mem_usage	-	バイト	MemUsage
			Container Memory Usage (byte)
			コンテナメモリ使用量
mem_working_set	-	バイト	MemWs
			Container Memory Working Set (byte)
			コンテナメモリworking set = mem_usage - inactive file
mem_working_set_percent	-	percent	MemWsByLimit
			Container Memory Working Set by Limit (%)
			コンテナメモリLimit基準working set使用量 = mem_working_set / mem_limit * 100

Field	Type	Unit	Shortname/Name/Description
mem_request	-	バイト	MemRequest
			Container Memory Request (byte)
			コンテナのメモリRequestサイズ
mem_per_request	-	percent	MemWsByRequest
			Container Memory Working Set by Request (%)
			コンテナメモリRequest基準working set使用量 = mem_working_set / mem_request * 100
network_rbps	-	byte	NetRxBytes
			Container Network Receive Byte
			コンテナ全体ブロックデバイスの1秒当たり読み取りバイトの合計
network_rdropped	-	バイト	NetRxDropped
			Container Network Receive Dropped
			コンテナネットワーク受信dropped数
network_rerror	-	バイト	NetRxError
			Container Network Receive Error
			コンテナのネットワーク受信エラー数
network_riops	-	バイト	NetRxIops
			Container Network Receive IOPS

Field	Type	Unit	Shortname/Name/Description
			コンテナのネットワーク受信数
network_wbps	-	バイト	NetTxByes
			Container Network Transmit Byte
			コンテナのネットワーク送信データサイズ
network_wdropped	-	count	NetTxDropped
			Container Network Transmit Dropped
			コンテナのネットワーク送信dropped数
network_werror	-	count	NetTxError
			Container Network Transmit Error
			コンテナのネットワーク送信エラー数
network_wiops	-	count	NetTxIops
			Container Network Transmit IOPS
			コンテナのネットワーク送信数
node_cpu	-	percent	ConNodeCpu
			Container Work Node CPU Usage (%)
			コンテナ実行中のノードのCPU使用量
node_mem	-	percent	ConNodeMem
			Container Work Node Memory Usage (%)

Field	Type	Unit	Shortname/Name/Description
			コンテナ実行中のノードのメモリ使用量
phase	string	-	Podライフサイクル ① PENDING ② RUNNING ③ SUCCEEDED ④ FAILED ⑤ UNKNOWN
restart_count	integer	-	ConRestartCnt
			Container Restart Count
			コンテナの再起動回数
state	integer	-	ConState
			Container Current State
			コンテナステータスコード ① RUNNING = 114 ② PAUSE = 112 ③ RESTARTING = 101 ④ OOMKILLED = 111 ⑤ DEAD = 100 ⑥ WAITING = 119
status	string	-	ConStatus
			Container Current Status
			コンテナステータス情報 ① runningステータス：uptime情報表示 ② waiting/terminatedステータス：ステータスに対するreason情報の表示

> mem_maxusage は、コンテナの実行中に記録されたメモリの最大使用量を示します。しかし、Linuxカーネルバージョンが5.19より低い場合、このメトリクスの原本データが対応できない場合があります。このような場合、その値が0になる可能性があります。該当メトリクスを正常に収集するためには、Linuxカーネルバージョンを5.19以上にアップデートする必要があります。

Kubernetesノード(kube_node) メトリクス

kube_node カテゴリは、ノードに設定されているすべてのユーザー定義ラベルをタグとして収集します。

- 対象：クラスタープロジェクト、ネームスペースプロジェクト
- 収集間隔：5秒
- 統計データ：5分、1時間

Tags

Tag	Type	Unit	Description
nodeName	-	-	ノード名

次のタグが追加されました。

Tag	Type	Unit	Description
infrastructureType	string	-	ノードのインフラ種別 - baremetal、vm、unknown

Fields

Field	Type	Unit	Description
allocatable_cpu	-	millicores	ノード割り当て可能なCPU量

Field	Type	Unit	Description
allocatable_memory	-	バイト	ノード割り当て可能なメモリ量
allocatable_pods	integer	-	ノード割り当て可能なPod数
limit_cpu	-	millicores	ノードCPU Limit合計
limit_memory	-	バイト	ノードメモリLimit合計
pods	integer	-	ノードPodの総個数
request_cpu	-	millicores	ノードCPU Request合計
request_memory	-	バイト	ノードメモリRequest合計

ノードのGPU数量は、デバイスタイプ別の並列リストフィールドで収集します。`*_types`と`*_counts`はインデックスが1:1で対応し、`nvidia.com/gpu`(whole-GPU)とすべてのMIGプロファイル(`nvidia.com/mig-*`)を個別の項目として保持します。

Field	Type	Unit	Description
gpu_capacity_types	list	-	ノードGPU capacityのデバイスタイプ一覧
gpu_capacity_counts	list	count	上記タイプ別capacity数量(インデックス1:1対応)
gpu_allocatable_types	list	-	ノードGPU allocatableのデバイスタイプ一覧
gpu_allocatable_counts	list	count	上記タイプ別allocatable数量(インデックス1:1対応)

`v1.9.14`から、whole-GPU専用のスカラーフィールド`capacity_gpu`、`allocatable_gpu`は削除されました。MIG環境でスカラーフィールドが`nvidia.com/gpu`のみを保持しMIGスライスを欠落させることで、集計が誤った合計になる問題を防ぐためです。whole-GPU数量もリストの`nvidia.com/gpu`項目で確認し、全体・タイプ別の集計は`*_counts`を合算します。

Category	Removed scalar field	Replacement list field
kube_node	capacity_gpu, allocatable_gpu	gpu_capacity_types / gpu_capacity_counts, gpu_allocatable_types / gpu_allocatable_counts

---

Kubernetesノード性能(kube_node_perf) メトリクス

kube_node_perf カテゴリは、ノードにスケジューリングされたすべてのPod/Containerのリソース使用量と割り当て量を集計し、ノード単位の性能メトリクスを収集します。

- 対象：クラスタープロジェクト、ネームスペースプロジェクト
- 収集間隔：5秒
- 統計データ：5分、1時間

Tags

Tag	Type	Unit	Description
onode	-	-	ノードハッシュ値
onodeName	-	-	ノード名
agentPcode	-	-	エージェントプロジェクトコード
agentOid	-	-	エージェント固有ID

次のタグが追加されました。

Tag	Type	Unit	Description
cgroup_version	-	-	ノードホストのcgroupマウントモード - `unified`(cgroup v2)、`hybrid`、`legacy`(cgroup v1)、`unknown`(アクセス不可)

ノードホストの `/sys/fs/cgroup` マウントモードを直接検知してタグとして収集します。ノード別のcgroupバージョン状況をサーバーで集計できます。

ノードの仮想化判別タグが追加されました。

Tag	Type	Unit	Description
`virtualization`	`string`	-	ノードの仮想化の有無 - `baremetal`、`vm`、`unknown`
`hypervisor`	`string`	-	ハイパーバイザーのベンダー名 - ベアメタルの場合や判別できない場合は`-`

ノードホストのDMI情報（ `/sys/class/dmi/id/sys_vendor` 、 `product_name` ）でベンダーを先に照合し、一致するベンダーがない場合は `/proc/cpuinfo` のhypervisorフラグで仮想マシンかどうかのみを判定します。DMI情報がない非x86ノードは `unknown` として収集します。

QEMU/KVM、VMware、Hyper-V、Xen、VirtualBox、Parallels、Amazon EC2、Google GCE、OpenStack、Nutanix AHV、oVirt/RHV、Alibaba Cloudなどのベンダーを認識します。値はノードの稼働中に変わらないため、最初の1回のみ読み取ってキャッシュします。

制約：Amazon EC2の `.metal` インスタンスのようにクラウド事業者が提供するベアメタルは、DMI情報だけでは一般インスタンスと区別できないため `vm` として判別されます。

ノードエージェント1.9.21バージョン以上で収集します。

Fields

Field	Type	Unit	Description
cpuRequest	long	millicores	ノード内全コンテナのCPU Request合計

Field	Type	Unit	Description
cpuLimit	long	millicores	ノード内全コンテナのCPU Limit合計
cpuTotal	long	millicores	ノード内全コンテナのCPU使用量合計
cpuPerRequest	float	percent	ノードCPU Request対比の実際のCPU使用率
cpuPerLimit	float	percent	ノードCPU Limit対比の実際のCPU使用率
memoryRequest	long	byte	ノード内全コンテナのMemory Request合計
memoryLimit	long	byte	ノード内全コンテナのMemory Limit合計
memTotal	long	byte	ノード内全コンテナのMemory使用量合計
memoryPerRequest	float	percent	ノードMemory Request対比の実際のMemory使用率
memoryPerLimit	float	percent	ノードMemory Limit対比の実際のMemory使用率

次のフィールドが追加されました。

Field	Type	Unit	Description
`conntrack_count`	`long`	`count`	現在のconntrackエントリ数
`conntrack_max`	`long`	`count`	conntrackエントリの上限（`nf_conntrack_max`）
`conntrack_pct`	`float`	`percent`	上限に対する使用率

- エントリ数は、ホストのnetwork namespace基準の値を読み取るため `/proc/1/net/stat/nf_conntrack` パスから収集します。
- `nf_conntrack` モジュールがロードされていないノードはフィールドを送信しません。値 `0` と未収集を区別するための動作です。
- ノードエージェント1.9.18バージョン以上で収集します。 `collect_conntrack_enabled` オプションで収集をオフにできます。

Kubernetesイベント(kube_event) メトリクス

kube_event カテゴリは、クラスタープロジェクトの場合は、クラスター全体を対象としてデータを収集し、ネームスペースプロジェクトの場合は、ネームスペースで発生したイベントのみ収集します。

- 対象：クラスタープロジェクト、ネームスペースプロジェクト
- 収集間隔：5秒
- 統計データ：5分、1時間

Tags

Tag	Type	Unit	Description
field_path	-	-	Field Path
kind	-	-	イベントが発生したオブジェクトの種類
name	-	-	イベントが発生したKubernetesオブジェクト名
namespace	-	-	イベントが発生したネームスペース名
reason	-	-	イベント発生理由
type	-	-	イベントタイプ - WarningまたはNormal
uid	-	-	UID - イベントが発生したオブジェクト

Fields

Field	Type	Unit	Description
action	string	-	アクション名

Field	Type	Unit	Description
count	-	count	イベント発生件数
event_time	integer	-	イベント初期発生TimeStamp
first_timestamp	integer	-	イベント初期発生時間
last_timestamp	integer	-	イベント最終発生時間
message	string	-	イベントメッセージ
reasonFiled	string	-	イベントReason
reporting_component	string	-	現在のイベントを報告するコンポーネント
reporting_instance	string	-	現在のイベントを報告するインスタンス
series_last_observed_time	integer	-	series last observed time

---

Kubernetesクラスター(`kube_stat`) メトリクス

`kube_stat` カテゴリは、クラスタープロジェクトのクラスター全体を対象として収集し、ネームスペースプロジェクトは、そのネームスペースに関連するオブジェクトのみ対象に収集します。

- 対象：クラスタープロジェクト、ネームスペースプロジェクト
- 収集間隔：5秒
- 統計データ：5分、1時間

Tags

Tag	Type	Unit	Description
name	-	-	kube_stat(固定値)

次のタグが追加されました。

Tag	Type	Unit	Description
namespace	-	-	収集対象のネームスペース名

Fields

Field	Type	Unit	Description
alloctable_cpu	-	millicores	クラスタ全体のコア数(クラスタプロジェクトOnly)
alloctable_ephemeral-storage	-	バイト	クラスター全体に割り当てられるephemeral storage(クラスタープロジェクトOnly)
alloctable_hugepages-1gi	-	バイト	クラスター全体に割り当てられるhugepages-1Gi(クラスタープロジェクトOnly)
alloctable_hugepages-2mi	-	バイト	クラスター全体に割り当てられるhugepages-2Mi(クラスタープロジェクトOnly)
alloctable_memory	-	バイト	クラスター全体に割り当てることができるメモリ(クラスタープロジェクトOnly)
alloctable_pods	integer	-	割り当て可能なPod数
available_pod	integer	-	PhaseがRunning状態のPodの数
desired_pod	integer	-	`metadata.ownerReferences`なしで配布されたPod数とクバネティスオブジェクト(ReplicaSet, Daemonset, StatefulSet)に定義されたDesired Pod数の合計
			`kubectl get pods -A`で参照したPod数と同じ
nodes	integer	-	ノード数
pod_phase_Pending	integer	-	PendingステータスPod数

Field	Type	Unit	Description
pod_phase_Running	integer	-	Running状態Pod数
running_containers	integer	-	Runningコンテナ数
stopped_containers	integer	-	Stoppedコンテナ数
total_available_cpu	integer	-	割り当て可能なCPUの合計
total_available_memory	integer	-	割り当て可能なMemory合計
total_limit_cpu	-	millicores	Limit CPU合計
total_limit_memory	-	バイト	Limit Memory合計
total_request_cpu	-	millicores	Request CPU合計
total_request_memory	-	バイト	Request Memory合計
unavailable_pod	integer	-	PhaseがRunning状態ではない(Pending, Failed, Succedded) Pod数
waiting_containers	integer	-	Waiting Container数

ネームスペースに設定されたResourceQuotaの上限と消費量を収集します。次のフィールドが追加されました。

Field	Type	Unit	Description
quota_count	long	count	ネームスペースに設定されたResourceQuota数
quota_request_cpu_hard	long	millicores	CPU Requestクォータの上限
quota_request_cpu_used	long	millicores	CPU Requestクォータの消費量
quota_request_memory_hard	long	byte	Memory Requestクォータの上限
quota_request_memory_used	long	byte	Memory Requestクォータの消費量

Field	Type	Unit	Description
quota_limit_cpu_hard	long	millicores	CPU Limitクォータの上限
quota_limit_cpu_used	long	millicores	CPU Limitクォータの消費量
quota_limit_memory_hard	long	byte	Memory Limitクォータの上限
quota_limit_memory_used	long	byte	Memory Limitクォータの消費量
quota_pods_hard	long	count	Pod数クォータの上限
quota_pods_used	long	count	Pod数クォータの消費量

- Kubernetesが算出した `status.hard` と `status.used` の値をそのまま使用します。
- 1つのネームスペースにスコープの異なるResourceQuotaが複数ある場合は、項目ごとに合算します。
- クォータが設定されていないネームスペースは `quota_count` のみを `0` で送信し、残りのフィールドは送信しません。クォータ未設定と上限 `0` を区別するための動作です。
- マスターエージェント1.9.17バージョン以上で収集します。

権限：ClusterRoleにcore APIの `resourcequotas` に対する `list` 権限が必要です。権限がない場合、ResourceQuotaの収集のみが省略され、残りの `kube_stat` 収集は継続されます。

Pod(`kube_pod`) メトリクス

`kube_pod` カテゴリは、Podに設定されているすべてのユーザー定義ラベルをタグとして収集します。

- 対象：マスター(クラスター) プロジェクト、ネームスペースプロジェクト
- 収集間隔：5秒
- 統計データ：5分

Tags

Tag	Type	Unit	Description
agentOid	-	-	ノードエージェント固有ID(固有値)
agentPcode	-	-	プロジェクトコード(固有値)
command	-	-	実行コマンド
containerIds	-	-	Podに属しているコンテナID
containerIdsCount	-	-	containerIdsの個数
containerKeys	-	-	Podに属しているコンテナIDのハッシュ値
containerKeysCount	-	-	containerKeysの個数
DaemonSet	-	-	PodのDaemonSet名
Deployment	-	-	Deployment
k8s-app	-	-	Podのラベルk8s-appに対する値
microOid	-	-	Podのコンテナ内部アプリケーションで実行中のエージェントのID
microOids	-	-	Podの複数コンテナ内部のアプリケーションで実行中のエージェントの複数ID
microOidsCount	-	-	microOidsの個数
name	-	-	Pod名
onames	-	-	Podのコンテナ内部アプリケーションで実行中のエージェントの名称
onamesCount	-	-	onamesの個数
podName	-	-	Pod名

Tag	Type	Unit	Description
namespace	-	-	Podが所属するネームスペース
namespaceHash	-	-	Podが所属するネームスペースハッシュ値
replicaSetHash	-	-	PodのReplicaSetハッシュ値
replicaSetName	-	-	PodのReplicaSet名
whatap_project	-	-	Podが属するWhaTapプロジェクト名

Fields

Field	Type	Unit	Shortname、 Name、 Description
blkio_rbps	-	バイト	IoReadBytes
			Pod Block I/O Read Byte
			Pod全体ブロックデバイスの1秒あたり読み込みバイトの合計
blkio_riops	-	count	IoReadIops
			Pod Block I/O Read IOPS
			Pod全体ブロックデバイスの1秒当たり読み込み件数の合計
blkio_wbps	-	バイト	IoWriteBytes
			Pod Block I/O Write Byte
			Pod全体ブロックデバイスの1秒あたり書き込みバイトの合計
blkio_wiops	-	count	IoWriteIops
			Pod Block I/O Write IOPS

Field	Type	Unit	Shortname、 Name、 Description
			Pod全体ブロックデバイスの1秒当たり書き込み件数の合計
cpu_per_limit	-	percent	CpuByLimit
			Pod CPU Usage by Limit (%)
			CPU Limit基準CPUの全体使用率
cpu_per_request	-	percent	CpuByRequest
			Pod CPU Usage by Request (%)
			CPU Request基準CPUの全体使用率
cpu_quota_percent	-	percent	CpuLimitByNode
			Pod CPU Limit by Node (%)
			ノードCPUに対するPod CPU Limit割当量 - Limit未設定の場合、Podが実行中のノードのCPU全体コアがパーセントで表示
cpu_sys	-	percent	CpuSysByNode
			Pod CPU Sys Usage by Node (%)
			ノードCPUに対するPod CPU System使用率
cpu_throttledperiods	-	count	CpuThrottledCnt
			Pod CPU Throttling Count
			Pod CPU Throttled件数
cpu_throttledtime	-	nanosecond(ns)	CpuThrottledTime

Field	Type	Unit	Shortname、 Name、 Description
			Pod CPU Throttling Time
			Pod CPU Throttled時間
cpu_total	-	percent	CpuByNode
			Pod CPU Usage by Node (%)
			ノードCPUに対するPod CPU使用率
cpu_total_milli	-	millicores	CpuTotUsage
			Pod CPU Usage (millicore)
			Pod CPU使用量
cpu_user	-	percent	CpuUserByNode
			Pod CPU User Usage by Node (%)
			ノードCPUに対するPod CPU User使用率
cpu_request	-	millicores	CpuRequest
			Pod CPU Request (core)
			Pod CPUリクエスト
cpu_per_request	-	percent	CpuByRequest
			Pod CPU Usage by Request (%)
			Pod CPUリクエストに対する使用率 = cpu_total_milli / cpu_request * 100
mem_totalcache	-	バイト	MemTotCache

Field	Type	Unit	Shortname、 Name、 Description
			Pod Memory Total Cache (byte)
			Pod全体のキャッシュサイズ
mem_totalpgfault	-	count	MemTotPageFaultCnt
			Pod Memory Total Page Fault Count
			Pod PageFault回数
mem_totalrss	-	バイト	MemTotRss
			Pod Memory Total RSS (byte)
			Pod RSSメモリ全体サイズ
mem_totalrss_percent	-	percent	MemTotRssByLimit
			Pod Memory Total RSS by Limit (%)
			Pod RSSメモリ全体使用率
mem_totalunevictable	-	バイト	MemTotUnevictable
			Pod Memory Total Unevictable (byte)
			Pod Unevictable Memory全体サイズ
mem_usage	-	バイト	MemUsage
			Pod Memory Usage (byte)
			Podメモリ使用量
mem_working_set	-	バイト	MemWs

Field	Type	Unit	Shortname、 Name、 Description
			Pod Memory Working Set (byte)
			Podメモリworking set = mem_usage - inactive file
memory_request	-	バイト	MemRequest
			Pod Memory Request (byte)
			Podメモリー要求量
memory_limit	-	バイト	MemLimit
			Pod Memory Limit (byte)
			PodメモリLimit割当量
memory_per_request	-	percent	MemByRequest
			Pod Memory Working Set By Request (%)
			Podメモリの要求基準のWorking Set使用量
memory_per_limit	-	percent	MemByLimit
			Pod Memory Working Set By Limit (%)
			PodメモリのLimit基準のWorking Set使用量
network_rbps	-	バイト	NetRxBytes
			Pod Network Receive Byte
			Pod全体ブロックデバイスの1秒あたり読み込みバイトの合計
network_rdropped	-	バイト	NetRxDropped

Field	Type	Unit	Shortname、 Name、 Description
			Pod Network Receive Dropped
			Podネットワーク受信のdropped件数
network_rerror	-	バイト	NetRxError
			Pod Network Receive Error
			Podネットワーク受信エラー件数
network_riops	-	バイト	NetRxIops
			Pod Network Receive IOPS
			Podネットワーク受信件数
network_wbps	-	バイト	NetTxByes
			Pod Network Transmit Byte
			Podネットワーク送信データサイズ
network_wdropped	-	count	NetTxDropped
			Pod Network Transmit Dropped
			Podネットワーク送信dropped件数
network_werror	-	count	NetTxError
			Pod Network Transmit Error
			Podネットワーク送信エラー件数
network_wiops	-	count	NetTxIops

Field	Type	Unit	Shortname、 Name、 Description
			Pod Network Transmit IOPS
			Podネットワーク送信件数
phase	string	-	Phase
			Pod Current Phase
			Podライフサイクル ① PENDING ② RUNNING ③ SUCCEEDED ④ FAILED ⑤ UNKNOWN

次のフィールドは内部用に予約されています。

Field	Type	Unit	Description
kube_sless_normal	-	-	Kubernetes情報イベント発生件数
kube_sless_warning	-	-	Kubernetes警告イベント発生件数
micro_sful_critical	-	-	apm重要ステータスベースのイベント発生件数
micro_sful_info	-	-	apm情報ステータスベースのイベント発生件数
micro_sful_warning	-	-	apm警告ステータスベースのイベント発生件数
micro_sless_critical	-	-	apm深刻状態イベント発生件数
micro_sless_info	-	-	apm情報状態イベント発生件数
micro_sless_warning	-	-	apm警告状態イベント発生件数
sful_critical	-	-	メトリクス重要状態ベースのイベント発生件数

Field	Type	Unit	Description
sful_info	-	-	メトリクス情報ステータスベースのイベント発生件数
sful_warning	-	-	メトリクス警告ステータスベースのイベント発生件数
sless_critical	-	-	メトリクス重要状態イベント発生件数
sless_info	-	-	メトリクス情報状態イベント発生件数
sless_warning	-	-	メトリクス警告状態イベント発生件数

PodのGPU Request・Limit数量は、デバイスタイプ別の並列リストフィールドで収集します。`*_types` と `*_counts` はインデックスが1:1で対応し、`nvidia.com/gpu` (whole-GPU)とすべてのMIGプロファイル(`nvidia.com/mig-*`)を個別の項目として保持します。

Field	Type	Unit	Description
gpu_request_types	list	-	Pod GPU Requestのデバイスタイプ一覧
gpu_request_counts	list	count	上記タイプ別Request数量(インデックス1:1対応)
gpu_limit_types	list	-	Pod GPU Limitのデバイスタイプ一覧
gpu_limit_counts	list	count	上記タイプ別Limit数量(インデックス1:1対応)

> ⓘ `v1.9.14` から、whole-GPU専用のスカラーフィールド `gpu_request` 、 `gpu_limit` は削除されました。MIG環境でスカラーフィールドが `nvidia.com/gpu` のみを保持しMIGスライスを欠落させることで、集計が誤った合計になる問題を防ぐためです。whole-GPU数量もリストの `nvidia.com/gpu` 項目で確認し、全体・タイプ別の集計は `*_counts` を合算します。

Kubernetes Pod統計(`kube_pod_stat`) メトリクス

`kube_pod_stat` カテゴリのクラスタープロジェクトは、クラスター全体のデータを収集し、ネームスペースプロジェ

クトは、ネームスペースに所属するPodのデータのみ収集します。

- 対象：クラスタープロジェクト、ネームスペースプロジェクト
- 収集間隔：5秒
- 統計データ：5分、1時間

Tags

Tag	Type	Unit	Description
kind	-	-	タイプ - クラスタープロジェクトの場合は、clusterに固定値を持ち、ネームスペースプロジェクトの場合は、DeploymentまたはReplicaSet のみ収集
name	-	-	Kubernetes資源名 -クラスタープロジェクトにはname値がなく、ネームスペースプロジェクトはDeploymentまたはReplicaSetの名前

Fields

Field	Type	Unit	Description
available_pod	integer	-	PhaseがRunning状態のPodの数
desired_pod	integer	-	`metadata.ownerReferences`なしで配布されたPod数とクバネティスオブジェクト(ReplicaSet, Daemonset, StatefulSet)に定義されたDesired Pod数の合計
			`kubectl get pods -A`で参照したPod数と同じ
limit_cpu	-	millicores	CPU Limit使用量
limit_memory	-	バイト	Memory Limit使用量
request_cpu	-	millicores	CPU Request使用量

Field	Type	Unit	Description
request_memory	-	バイト	Memory Request使用量
running_container	integer	-	Running Container数
stopped_container	integer	-	Stopped Container数
waiting_container	integer	-	Waiting Container数

Kubernetes Podボリューム(`kube_pod_volume`) メトリクス

`kube_pod_volume` カテゴリは、ノードで実行中のPodにマウントされたボリュームの容量・i-node使用量を収集します。アプリPodのPVCボリュームはディレクトリ名がPV名であるため、 `volumeName` タグでPVCの `spec.volumeName` と結合できます。

Tags

Tag	Type	Unit	Description
hostDevice	-	-	ボリュームが位置するホストデバイス
hostFileSystem	-	-	ボリュームのファイルシステム
hostMountPath	-	-	ホスト上のボリュームマウントパス
volumeName	-	-	ボリューム名 - PVCボリュームの場合はPV名(例: `pvc-...`)。PVCの `spec.volumeName` と結合可能
volumePlugin	-	-	ボリュームプラグイン - 例: `kubernetes.io~csi` 、 `kubernetes.io~nfs` 、 `volume-subpaths`

Fields

Field	Type	Unit	Description
capacity	long	byte	ボリューム全体の容量
available	long	byte	ボリュームの使用可能容量
availablePercent	float	percent	ボリュームの使用可能容量の割合
usedPercent	float	percent	ボリューム使用率
free	long	byte	ボリュームの空き容量
freePercent	float	percent	ボリュームの空き容量の割合
totalInode	long	count	ボリュームの全i-node数
usedInode	long	count	ボリュームで使用中のi-node数
usedInodePercent	float	percent	ボリュームのi-node使用率

emptyDirボリュームの収集が追加されました。デフォルト（ディスク）mediumのemptyDirは別途のマウントではなくノードのルートファイルシステム上のディレクトリであるため、`df`ベースの収集では捕捉できませんでした。ノードエージェントがkubeletのボリュームディレクトリを直接照会し、同じ`kube_pod_volume`カテゴリーに送信します。

- 照会パス：`{HOST_PREFIX}{kubelet_root_dir}/pods/{podUid}/volumes/kubernetes.io~empty-dir/{volumeName}`
- 収集周期：60秒。ディレクトリ照会のコストが大きいため、5秒の収集周期とは分離します。
- emptyDirボリュームの`volumePlugin`タグの値は`kubernetes.io~empty-dir`です。

次のタグが追加されました。

Tag	Type	Unit	Description
volumeType	string	-	ボリュームの種類 - emptyDirボリュームは EmptyDir
emptyDirMedium	string	-	emptyDirの保存媒体 - Memory または Disk。Podのspecの emptyDir.medium 基準で、未指定は Disk
emptyDirSpecConfirmed	string	-	Podのspecを確認したかどうか - true はspecで確認した値、false はkubeletのパスとファイルシステムの種類から推定した値

次のフィールドが追加されました。

Field	Type	Unit	Description
usedSpace	long	byte	ボリューム使用量の合計
fileCount	long	count	ボリューム内のファイル数
sizeLimit	long	byte	Podのspecの emptyDir.sizeLimit - 未設定の場合は送信しません
truncated	integer	-	収集上限を超過したかどうか - 1 の場合は上限に達し、値が実際より小さいことを示します

capacity 、 available 、 usedPercent は、ボリューム容量を確定できる場合にのみ送信します。

- sizeLimit がある場合は、その値をボリューム容量として使用します。
- sizeLimit がなく、mediumが Memory の場合は、kubeletが作成したtmpfsの総量を容量として使用します。tmpfs総量のデフォルト値はノードメモリの半分です。
- sizeLimit がない Disk mediumは上限がないため容量を作成しません。この場合はPodの container.ephemeral_storage_limit と比較してください。

ノードエージェント1.9.23バージョン以上で、上記のタグとフィールドをすべて送信します。収集の有効化とコスト上限のオプションは、Kubernetesエージェントのオプション設定を参照してください。

Kubernetes Horizontal Pod Autoscaler(HPA)(kube_hpa_stat) メトリクス

WhaTapで使用するCluster RoleにHPAが追加されると、メトリクス収集が開始されます。

- 対象：クラスタープロジェクト
- 収集間隔：5秒
- 統計データ：5分、1時間

Tags

Tag	Type	Unit	Description
name	-	-	HPA名

次のタグが追加されました。

Tag	Type	Unit	Description
scaleTargetKind	string	-	スケール対象オブジェクトの種類（spec.scaleTargetRef.kind）
scaleTargetName	string	-	スケール対象オブジェクトの名前（spec.scaleTargetRef.name）
scaleTargetApiVersion	string	-	スケール対象オブジェクトのAPIバージョン
creationTimeMillis	long	-	HPAの作成時刻（epoch ms） - 値がない場合は送信しません

HPAのスケール対象は spec.scaleTargetRef にあります。 ownerReferences ベースの owner_kind 、 owner_name 、 owner_id タグはHPAではほとんど空の値になるため、スケール対象は scaleTarget* タグで確認してください。既存のタグは下位互換のために維持します。

マスターエージェント1.9.24バージョン以上で収集します。

Fields

Field	Type	Unit	Description
currentReplicas	integer	count	現在のレプリカ数
desiredReplicas	integer	count	Desiredレプリカ数
lastScaleTime	integer	count	最後にスケールが変更されたTimeStamp
maxReplicas	integer	count	最大レプリカ数
minReplicas	integer	count	最小レプリカ数

Kubernetes HPAリスト(kube_hpa) メトリクス

kube_hpa カテゴリーは、 kubectl get hpa -o wide の結果と同等のHPAリストデータを収集します。時系列指標の性格を持つ kube_hpa_stat とは別のカテゴリーです。

- 対象：クラスタープロジェクト
- 収集間隔：30秒
- 要求バージョン：マスターエージェント1.9.24以上

Tags

Tag	Type	Unit	Description
hpaUid	string	-	HPAのUID（固有値）
hpaName	string	-	HPAの名前
namespace	string	-	HPAが所属するネームスペース

Tag	Type	Unit	Description
creationTimeMillis	long	-	HPAの作成時刻（epoch ms）
scaleTargetKind	string	-	スケール対象オブジェクトの種類
scaleTargetName	string	-	スケール対象オブジェクトの名前
scaleTargetApiVersion	string	-	スケール対象オブジェクトのAPIバージョン

Fields

Field	Type	Unit	Description
minReplicas	integer	count	最小レプリカ数
maxReplicas	integer	count	最大レプリカ数
currentReplicas	integer	count	現在のレプリカ数
desiredReplicas	integer	count	Desiredレプリカ数
targetMetricType	list	-	スケール基準となるメトリックの種類リスト
targetValue	list	-	メトリック別の目標値リスト
currentValue	list	-	メトリック別の現在値リスト
conditionAbleToScale	string	-	AbleToScaleコンディションの状態
conditionScalingActive	string	-	ScalingActiveコンディションの状態
conditionScalingLimited	string	-	ScalingLimitedコンディションの状態

targetMetricType、targetValue、currentValueは、同じインデックスが1つのメトリックに対応する並列リストです。HPAはCPU、メモリ、ユーザー定義メトリックを同時に持つことができるため、1つの値では表現できません。

- 値は `kubectl` の表記と同じ文字列です（例： `50%` 、 `100m` ）。
- metrics-serverがなく現在値を取得できない場合は `<unknown>` として収集します。
- Resourceメトリック以外のPods、External、Objectメトリックは、それぞれ `pods:` 、 `external:` 、 `object:` の接頭辞を付けて格納します。
- `autoscaling/v2` を優先して使用し、未対応のクラスターは `autoscaling/v1` で代替します。

ClusterRoleに `horizontalpodautoscalers` 権限がすでに含まれているため、追加の権限設定は必要ありません。

プロセス(`kube_process`) メトリクス

> ⓘ Kubernetesエージェント1.7.12バージョン以降が必要です。エージェントのアップデートの詳細については、次の文書を参照してください。

ノードに存在するKubernetes関連のプロセスをモニタリングする時に収集されます。

- 対象：クラスタープロジェクト、ネームスペースプロジェクト
- 収集間隔：5秒
- 統計データ：5分

Tags

Tag	Type	Unit	Description
ppid	string	-	親プロセスID `/proc/[pid]/status::PPid`
pid	string	-	プロセスID `/proc/[pid]/status::Pid`
cmd1	string	-	コマンド名 `/proc/[pid]/status::Name`

Tag	Type	Unit	Description
cmd2	string	-	コマンドライン(全コマンドと印字) `/proc/[pid]/cmdline`
ユーザー	string	-	ユーザーIDまたはユーザー名 `/proc/[pid]/status::Uid`
onodeName	string	-	プロセスのノード名 コンテナシステム環境変数 (`NODE_IP`)
createTime	TimeStamp	-	プロセス開始時間 `/proc/uptime` によって計算されたフィールド

Fields

Field	Type	Unit	Description
cpu	float	percent(%)	CPU使用率 - `/proc/[pid]/stat` によって計算されたフィールド
memory	float	percent(%)	メモリ使用率 - `/proc/[pid]/statm` によって計算されたフィールド
rss	long	バイト	実際のメモリ使用量(Resident Set Size) - `/proc/[pid]/status` の VmRSS
uid	string	-	ユーザーIDまたはユーザー名 - `/proc/[pid]/status` のUid
state	string	-	プロセス状態 - `/proc/[pid]/status` のState
sharedMemory	long	バイト	共有メモリサイズ - `/proc/[pid]/statm` によって計算されたフィールド
openFileDescriptors	integer	-	プロセスが開いたファイルディスク リプター数

Field	Type	Unit	Description
			- `/proc/[pid]/fd`によって計算されたフィールド
vmSize	long	バイト	仮想メモリサイズ(Virtual Memory Size) - `/proc/[pid]/status`のVmSize
threads	integer	-	プロセスが作成したスレッド数 - `/proc/[pid]/status`のThreads

Kubernetes環境のLinuxプロセスの状態

Linuxで_/proc/[pid]/status_ファイルの`State`フィールドがプロセスの現在の状態を表示します。各ステータス値の意味は次の通りです。

Code	Description
R (Running)	実行中 - プロセスが実行中または実行される準備ができている状態
S (Sleeping)	待機中 - 割り込み可能な停止状態で、待機しながらイベントを待っている状態
D (Disk Sleep)	ディスク睡眠 - 割り込み不可能なスリープ状態で、通常はI/O作業を待っている状態
R (Zombie)	ゾンビ状態 - プロセスが終了したが、まだ親プロセスが終了状態を収集していない状態
T (Stopped)	中止 - プロセスが作業制御信号によって停止された状態(SIGSTOPなど) またはデバッガにより中止された状態
t (Tracing stop)	追跡中止 - デバッガによって追跡中の状態(小文字tで表示される)
X (Dead)	死んだ状態

Code	Description
	- プロセスが死んだ状態(通常は発生しない)
x (Dead)	死んだ状態 - カーネルスレッドが死んだ状態(通常は発生しない)
K (WakeKill)	強制終了 - 起こしの信号を無視して直ちに死んだ状態
W (Waking)	起こし中 - 起こし信号を受けて起こされる最中の状態
I (Idle)	遊休状態 - カーネルスレッドがアイドル状態(一般的にユーザーのプロセスには見えない)

> Kubernetesは、コンテナとノードのリソースを効率的に管理するため、コンテナ内部で実行される多数のプロセスが実際に待機状態を維持します。これにより、ほとんどのプロセスは `Sleeping` 状態の可能性があります。

エージェント状態(`agent_status_summary`) メトリクス

エージェントの状態に関するメトリクスを10秒間隔で収集したカテゴリです。

Fields

Field	Type	Unit	Description
`inActTime`	-	milliseconds(ms)	エージェントが無効になったままの時間
`isActive`	Boolean	-	現在のエージェントの活性状態有無(`true` / `false`)
`isRestart`	Boolean	-	エージェントが再起動したかどうか (`true` / `false`)

Field	Type	Unit	Description
lastActTime	-	milliseconds(ms)	最後にエージェントが活性化された状態の時刻 - 0 : 非活性化された場合
oid	-	-	プロジェクトに含まれる各エージェントの固有の識別子
oType	-	-	エージェント種類 - 1 : アプリケーションエージェント - 2 : subType 参照
startTime	-	milliseconds(ms)	エージェントが開始された時点を示すタイムスタンプ
subType	-	-	エージェント種類 - 9 : ノードエージェント - 10 : マスターエージェント

Ingress(kube_ingress)メトリクス

> Kubernetesエージェント1.7.13バージョン以降が必要です。エージェントのアップデートの詳細については、次の文書を参照してください。

Ingressリソースに関するメタデータと関連情報をモニタリングする際に収集されます。

- 対象：クラスタープロジェクト、ネームスペースプロジェクト
- 収集間隔: 30秒
- 統計データ：5分

Tags

Tag	Type	Unit	Description
ingressUid	string	-	Ingressリソースの固有ID
ingressName	string	-	Ingressリソースの名前
ingressNamespace	string	-	Ingressリソースのネームスペース
creationTimeMillis	long	milliseconds(ms)	Ingressリソース生成時間
ingressClassName	string	-	Ingressクラス名
ingressLoadBalancerIps	List	-	IngressロードバランサーIP

Fields

Field	Type	Unit	Description
host	List	-	Ingressリソースが受信するホスト名 (`*`の場合、すべてのホストに対して適用される)
path	List	-	特定のホストサブでのリクエストパス
backendServiceName	List	-	バックエンドで配信されるサービスの名前
backendServicePort	List	-	バックエンドで配信されるポート番号
backendServiceUid	List	-	バックエンドで配信されるサービスのuid
pathType	List	-	パス一致方式(例、`Prefix`、`Exact`)

Kubernetesクロンジョブ(kube_cronjob)メトリクス

デプロイメントリソースに関するメタデータとステータス情報をタグ付けして収集します。

Tags

Tag	Type	Unit	Description
cronJobName	string	-	CronJobの名前(metadata.name)
namespace	string	-	CronJobが属するネームスペース (metadata.namespace)
cronJobUid	string	-	CronJobのUID(metadata.uid)

Fields

Field	Type	Unit	Description
lastScheduleTime	string	-	CronJobが最後にスケジュールされた時間 (status.lastScheduleTime)
lastSuccessfulTime	string	-	CronJobが最後に成功した時間 (status.lastSuccessfulTime)
cronJobSpecSchedule	string	-	CronJobスケジュール周期(spec.schedule , 例: "0 */5 * * *")
successfulJobsHistoryLimit	integer	count	成功したJobを保存する最大個数 (spec.successfulJobsHistoryLimit)
failedJobsHistoryLimit	integer	count	失敗したJobを保存する最大個数 (spec.failedJobsHistoryLimit)
concurrencyPolicy	string	-	同時実行ポリシー (Allow , Forbid , Replace) (spec.concurrencyPolicy)

Field	Type	Unit	Description
startingDeadlineSeconds	integer	time(seconds)	スケジュールされた時間が過ぎた後でもJobを実行できる最大遅延時間(spec.startingDeadlineSeconds)

Kubernetesデプロイメント(kube_deployment)メトリクス

デプロイメントリソースに関するメタデータとステータス情報をタグ付けして収集します。

Tags

Tag	Type	Unit	Description
deployName	string	-	デプロイメント名(metadata.name)
namespace	string	-	デプロイメントが属するネームスペース(metadata.namespace)
deployUid	string	-	デプロイメントのUID(metadata.uid)

Fields

Field	Type	Unit	Description
deployReadyReplicas	integer	count	準備(Ready)状態のPod数(status.readyReplicas)
deployTotalReplicas	integer	count	設定されたPod数(spec.replicas)
deployUpdatedReplicas	integer	count	最新バージョンに更新されたPod数(status.updatedReplicas)
deployAvailableReplicas	integer	count	使用可能な状態のPod数(status.availableReplicas)
deployUnavailableReplicas	integer	count	使用不可能な状態のPod数

Field	Type	Unit	Description
			(status.unavailableReplicas)
deployCreationTime	integer	milliseconds(ms)	デプロイメント生成時刻(Unix Epoch milliseconds, metadata.creationTimestamp)
deploySelector	string	-	デプロイメントの選択者ラベル (spec.selector.matchLabels)
deployStrategy	string	-	デプロイメントの配布戦略(spec.strategy.type, 例: RollingUpdate, Recreate)

Kubernetesデーモンセット(kube_daemonset)メトリクス

デモンセットリソースに関するメタデータとステータス情報をタグ付けして収集します。

Tags

Tag	Type	Unit	Description
daemonSetName	string	-	DaemonSetの名前(metadata.name)
namespace	string	-	DaemonSetが属するネームスペース(metadata.namespace)
daemonSetUid	string	-	DaemonSetのUID(metadata.uid)
creationTime	integer	milliseconds(ms)	DaemonSetの生成時刻(metadata.creationTimestamp)
age	string	-	DaemonSetの生成後の経過時間

Fields

Field	Type	Unit	Description
currentNumberScheduled	integer	count	現在スケジュールされているPod数 (status.currentNumberScheduled)
desiredNumberScheduled	integer	count	期待するPod数(status.desiredNumberScheduled)
numberReady	integer	count	Ready状態のPod数(status.numberReady)
numberAvailable	integer	count	使用可能なPod数(status.numberAvailable)
numberMisscheduled	integer	count	誤ってスケジュールされたPod数(status.numberMisscheduled)
updatedNumberScheduled	integer	count	更新されたPod数(status.updatedNumberScheduled)
daemonSetSelector	string	-	Selectorラベル(spec.selector.matchLabels)

Kubernetesレプリカセット(kube_replicaset)メトリクス

レプリカセット(ReplicaSet)リソースに関するメタデータとステータス情報をタグ付けして収集します。

Tags

Tag	Type	Unit	Description
replicaSetName	string	-	ReplicaSetの名前(metadata.name)
namespace	string	-	ReplicaSetが属するネームスペース(metadata.namespace)
replicaSetUid	string	-	ReplicaSetのUID (metadata.uid)

Tag	Type	Unit	Description
ownerKind	string	-	上位オブジェクトの種類(例: Deployment) (metadata.ownerReferences.kind)
ownerName	string	-	上位オブジェクト名(metadata.ownerReferences.name)
ownerUid	string	-	上位オブジェクトUID (metadata.ownerReferences.uid)
creationTime	integer	milliseconds(ms)	ReplicaSetの生成時刻(metadata.creationTimestamp)
age	string	-	ReplicaSetの生成後の経過時間

Fields

Field	Type	Unit	Description
replicaSetReplicas	integer	count	設定されたPod数(spec.replicas)
replicaSetReadyReplicas	integer	count	Ready状態のPod数(status.readyReplicas)
replicaSetAvailableReplicas	integer	count	使用可能なPod数(status.availableReplicas)
replicaSetSelector	string	-	Selectorラベル(spec.selector.matchLabels)

Kubernetesステートフルセット(kube_statefulSet)メトリクス

ステートフルセット(StatefulSet)リソースに関するメタデータとステータス情報をタグ付けして収集します。

Tags

Tag	Type	Unit	Description
statefulSetName	string	-	StatefulSetの名前(metadata.name)
namespace	string	-	StatefulSetが属するネームスペース(metadata.namespace)
statefulSetUid	string	-	StatefulSetのUID (metadata.uid)
creationTime	integer	milliseconds(ms)	StatefulSetの生成時刻(metadata.creationTimestamp)
age	string	-	StatefulSetの生成後の経過時間

Fields

Field	Type	Unit	Description
replicas	integer	count	設定されたPod数(spec.replicas)
readyReplicas	integer	count	Ready状態のPod数(status.readyReplicas)
currentReplicas	integer	count	現在作成されているPod数(status.currentReplicas)
updatedReplicas	integer	count	更新されたPod数(status.updatedReplicas)
statefulSetSelector	string	-	Selectorラベル(spec.selector.matchLabels)

Kubernetesジョブ(kube_job)メトリクス

デプロイメントリソースに関するメタデータとステータス情報をタグ付けして収集します。

Tags

Tag	Type	Unit	Description
jobName	string	-	Jobの名前(metadata.name)
namespace	string	-	Jobが属するネームスペース(metadata.namespace)
jobUid	string	-	JobのUID (metadata.uid)
ownerKind	string	-	上位オブジェクトの種類(例: CronJob)
ownerName	string	-	上位オブジェクトの名前
ownerUid	string	-	上位オブジェクトのUID

Fields

Field	Type	Unit	Description
succeededPods	integer	count	成功したPod数(status.succeeded)
failedPods	integer	count	失敗したPod数(status.failed)
completionTime	string	-	Jobが完了した時間(status.completionTime.toString())
parallelism	integer	count	同時に実行されるように設定されたPod数(spec.parallelism)
completions	integer	count	Jobが完了するために必要なPod実行回数(spec.completions)
backoffLimit	integer	count	失敗時に再試行可能な最大回数(spec.backoffLimit)
activeDeadlineSeconds	integer	seconds(sec)	Jobが実行できる最大時間(spec.activeDeadlineSeconds)

Kubernetes NetworkPolicy(kube_network_policy) メトリクス

kube_network_policy カテゴリーは、クラスターのNetworkPolicyオブジェクトをイベントベースで収集します。周期収集ではなく、オブジェクトが追加、変更、削除されるたびに、その時点のスナップショットを送信します。

- 対象：クラスタープロジェクト、ネームスペースプロジェクト
- 収集方式：イベントベース

Tags

Tag	Type	Unit	Description
networkPolicyUid	string	-	NetworkPolicyのUID（固有値）
networkPolicyName	string	-	NetworkPolicyの名前
namespace	string	-	NetworkPolicyが所属するネームスペース
resourceVersion	string	-	オブジェクトのリビジョン（metadata.resourceVersion）

Fields

Field	Type	Unit	Description
generation	long	-	オブジェクトの世代（metadata.generation）
creationTimeMillis	long	-	オブジェクトの作成時刻（epoch ms）
policyTypes	list	-	ポリシー種別のリスト - Ingress、Egress。空のリストはAPIのデフォルト値であるIngress
selectedPodUids	list	-	ポリシーが選択したPod UIDのリスト
podSelector	map	-	ポリシー対象のPodセレクター（matchLabels、matchExpressions）

Field	Type	Unit	Description
ingressRules	list	-	Ingressルールのリスト - ルールごとにfrom、portsを含む
egressRules	list	-	Egressルールのリスト - ルールごとにto、portsを含む
specSchemaVersion	integer	-	specフィールドのスキーマバージョン
specHash	long	-	正規化したspecのハッシュ値 - 内容の変更有無の判別用

ルールは入れ子構造で送信し、Kubernetes APIの意味をそのまま保持します。

- 1つのpeer内のnamespaceSelectorとpodSelectorはAND関係です。
- peerのリストとルールのリストはOR関係です。
- ルール内の空のfrom、toはすべてのpeerを、空のportsはすべてのポートを意味します。
- peerのipBlockはcidrとexceptを格納します。
- ポートはprotocol、portType、portNumberまたはportName、endPortを格納します。

ClusterRoleにnetworkpoliciesの照会権限が必要です。collect_network_policy_enabledオプションで収集をオフにできます。

GPUメトリクス

DCGM Exporterを通じて収集されるNVIDIA GPUメトリクスの一覧です。GPU収集方式とサイドカーパターンについては、動作環境 > NVIDIA GPUドキュメントを参照してください。

Metric	Type	Unit	Description
DCGM_FI_DEV_GPU_UTIL	gauge	%	GPUコアが演算処理を実行していた時間の割合。
DCGM_FI_PROF_GR_ENGINE_ACTIVE	gauge	ratio	グラフィックスエンジン(3D/Compute)がアクティブだった時間の割合。
DCGM_FI_PROF_SM_ACTIVE	gauge	ratio	1つ以上のワープ(Warp)がアクティブだったサイクルの割合。
DCGM_FI_PROF_SM_OCCUPANCY	gauge	ratio	SM(Streaming Multiprocessor)の占有率(SMあたりの常駐ワープ数の割合)。
DCGM_FI_PROF_PIPE_TENSOR_ACTIVE	gauge	ratio	テンソルコア(HMMA)パイプがアクティブだったサイクルの割合。
DCGM_FI_PROF_DRAM_ACTIVE	gauge	ratio	メモリインタフェースがアクティブ(データの送受信)だったサイクルの割合。
DCGM_FI_DEV_FB_USED	gauge	MiB	使用中のフレームバッファ(VRAM)メモリ。
DCGM_FI_DEV_FB_FREE	gauge	MiB	使用可能なフレームバッファ(VRAM)メモリ。
DCGM_FI_DEV_FB_TOTAL	gauge	MiB	フレームバッファ(VRAM)メモリの総量。
DCGM_FI_DEV_FB_USED_PERCENT	gauge	%	使用中のフレームバッファ(VRAM)メモリの割合。
DCGM_FI_DEV_FB_RESERVED	gauge	MiB	予約済みのフレームバッファメモリ。
DCGM_FI_PROF_PCIE_TX_BYTES	counter	byte	PCIeバスを通じて送信(Host → Device)した総データ量。

Metric	Type	Unit	Description
DCGM_FI_PROF_PCIE_RX_BYTES	counter	byte	PCIeバスを通じて受信(Device → Host)した総データ量。
DCGM_FI_DEV_POWER_USAGE	gauge	W	現在のGPUの消費電力。
DCGM_FI_DEV_GPU_TEMP	gauge	°C	GPUチップの現在の温度。
DCGM_FI_DEV_SM_CLOCK	gauge	MHz	SM(Streaming Multiprocessor)のクロック周波数。
DCGM_FI_DEV_MEM_CLOCK	gauge	MHz	メモリのクロック周波数。
DCGM_FI_DEV_PSTATE	gauge	P-State	現在のGPUの性能状態(P0が最高)。
DCGM_FI_DEV_ECC_SBE_AGG_TOTAL	counter	count	累積したシングルビット(Single-Bit)永久ECCエラー数。
DCGM_FI_DEV_ECC_DBE_AGG_TOTAL	counter	count	累積したダブルビット(Double-Bit)永久ECCエラー数。

次のメトリックが追加されました。

Metric	Type	Unit	Description
DCGM_FI_DEV_WEIGHTED_GPU_UTIL	gauge	ratio	GPU使用率を、デバイスが占有するスライスの割合で加重した値。

MIG環境と非MIG環境のGPU使用率を同じ基準で合算するためのメトリックです。MIGプロファイルは物理GPUの一部のみを占有するため、使用率をそのまま加算するとGPU単位の総量と一致しません。

算出方式は calculation_method ラベルで区分します。

Value	Description
weighted_sum	MIGインスタンス。DCGM_FI_PROF_GR_ENGINE_ACTIVE にスライスの割合

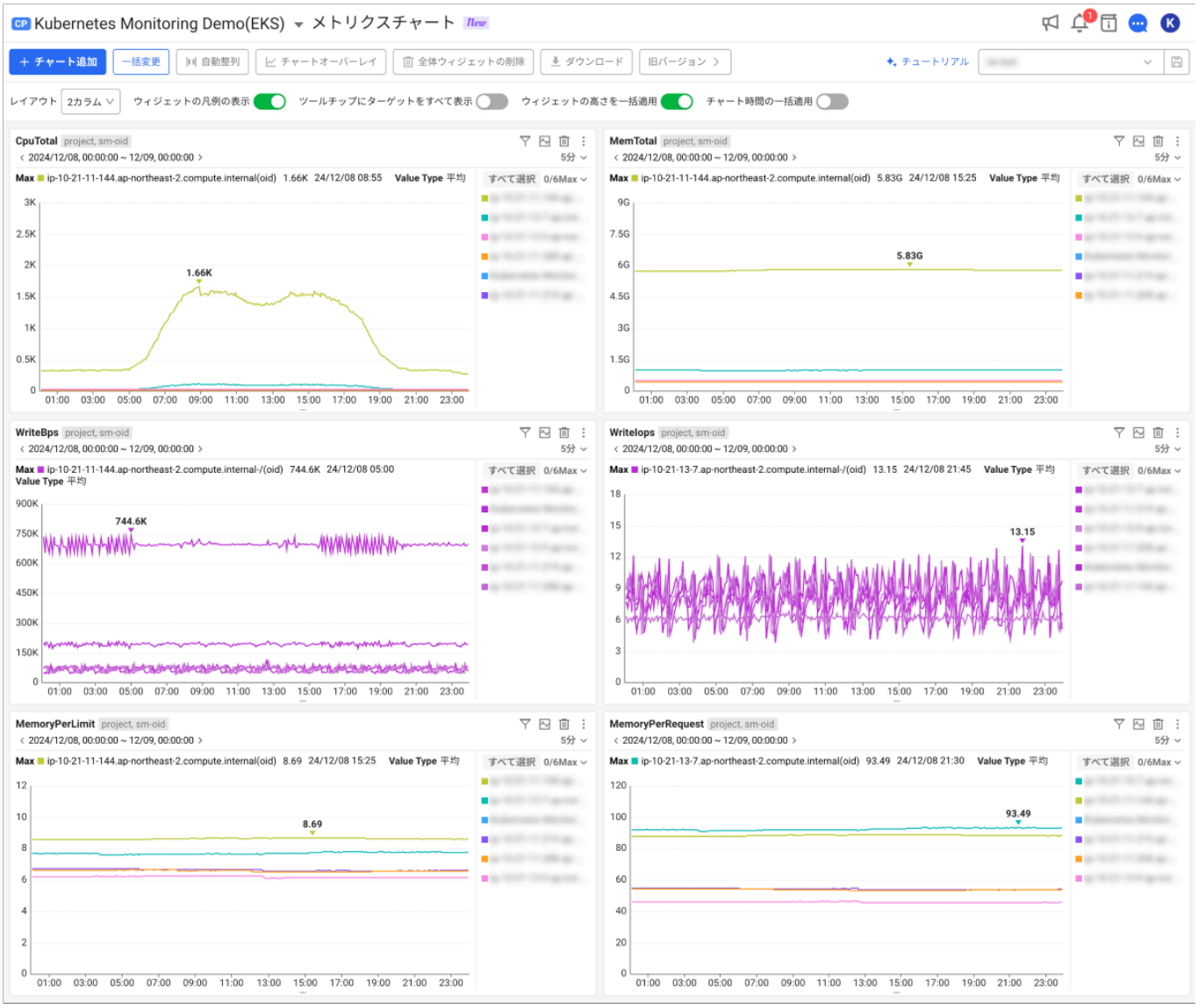
Value	Description
	（compute_slices / DCGM_FI_DEV_MIG_MAX_SLICES）を掛けて合算
direct	非MIG GPU。DCGM_FI_DEV_GPU_UTILを100で割った値
prof_gr_engine_active	H100、H200、H800、H20系列の非MIG GPU。DCGM_FI_PROF_GR_ENGINE_ACTIVEの値をそのまま使用

- 値の範囲は0から1までです。
- DCGM_FI_DEV_GPU_UTILが100を超える異常値は、100に制限してから計算します。
- GPU UUIDはDCGM_FI_DEV_UUIDラベルで一緒に提供します。

メトリクスチャート

ホーム画面 > プロジェクト選択 > **分析** > **メトリクスチャート** *New*

メトリクスチャート *New* メニューは、WhaTapモニタリングソリューションの主要機能であり、様々なモニタリング対象から収集したデータを時系列チャートで視覚化して提供します。 これにより、ユーザーはシステムの性能を効果的にモニタリングし、異常をすばやく識別して、問題解決に必要なインサイトを確認できます。



- 様々なメトリクスを時系列チャートを通じて視覚的に確認でき、障害発生時点の様々なメトリクスを一度に分析できます。
- 特定の時間範囲を選択して過去のデータを照会し、その期間の性能を分析できます。
- 特定の時間帯や条件に合ったデータをフィルタリングすることで、より正確な分析ができます。
- ユーザーは、目的のチャートを画面に配置し、レイアウトを自由に調整できます。
- ユーザーが希望するメトリクスを配置し、プリセットとして保存し、必要に応じて読み込みできます。

> ⓘ **既存のメトリクスチャートメニューとの違い**
>
> Service 2.8.0 アップデート以降の変更点は以下の通りです。 アップデート後の変更事項を画面で確認するには、右上の**チュートリアル**ボタンを選択します。
>
> - ユーザーが時間、対象、インターバルオプションを設定した後、モニタリングする対象のメトリクスを特定し、チャートを追加できるようにユーザー経験(UX)を改善しました。
> - ダッシュボードに配置されたウィジェットを削除することなく、チャートの照会時間の範囲を変更できます。
> - ダッシュボードに配置したすべてのウィジェットのオプションを一括で適用できる**一括変更**機能を提供します。
> - プリセット機能により、ダッシュボードに配置されたウィジェットの情報を保存し、読み込みできます。
> - 提供する様々なオプションにより、ダッシュボードのレイアウトやウィジェットの機能を調整できます。

チャートウィジェットを追加する

ダッシュボードにメトリクスをウィジェットとして追加するには、画面左上の**チャート追加**ボタンを選択します。

照会時間の設定

時間でメトリクスを照会する日付と時間を設定します。

対象を選択する

プロジェクトに含まれるエージェントの中から、対象を選択します。 個々のエージェントを選択するか、エージェン

ト設定で分類されたグループ単位で選択できます。

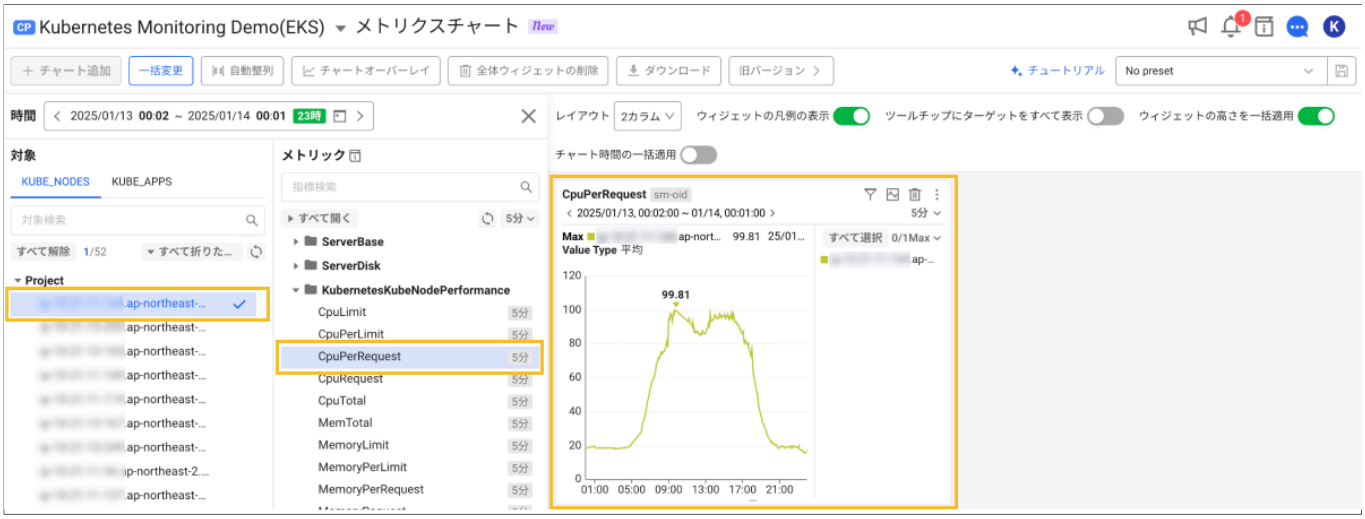


- **エージェント**: 個別エージェントまたはすべてを選択できます。
- **種類別**: エージェント設定で `whatap.okind` に分類されたグループ単位でエージェントをモニタリングします。
- **サーバー別**: エージェント設定で `whatap.onode` に分類されたグループ単位でエージェントをモニタリングします。

- 設定した時間と日付によって、選択できる対象が変更される場合があります。
- エージェントの一覧からマウスをクリックして、対象を選択または解除できます。 `Shift` キーを使用して複数のエージェントを選択できます。
- **すべて選択**または**すべて解除**ボタンを選択すると、関連する下位対象をすべて選択または解除できます。
- 希望のエージェントが一覧に表示されない場合は、**リフレッシュ**ボタンを選択します。
- **すべて折りたたむ**または**すべて開く**ボタンを選択してグループ単位で一覧を隠したり、展開したりできます。

指標を選択する

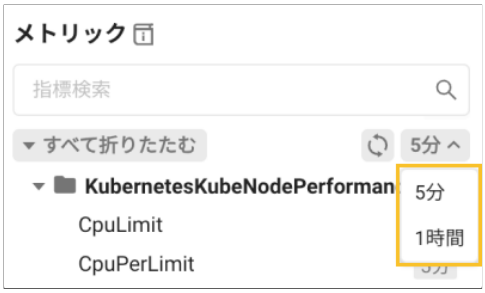
モニタリングする対象エージェントを選択した後、**メトリック**パネルに選択できる一覧が表示されます。 一覧からご希望のメトリクスを選択します。



選択したメトリクスに対応するチャートウィジェットが右側のダッシュボードに配置されます。 他のメトリクスを選択し、チャートウィジェットを追加できます。 チャート領域を拡大して見るには、**時間**オプションの右にある×ボタンを選択します。 **対象**と**メトリック**パネルが消え、チャート領域のみが画面に表示されます。

- **指標検索**入力欄に文字列を入力すると、希望のメトリクスをすばやく検索できます。
- メトリクスのカテゴリとフィールド名に基づいても検索できます。 カテゴリおよびフィールドの詳細については、次の文書を参照してください。
- **すべて折りたたむ**または**すべて開く**ボタンを選択して、カテゴリ単位でメトリクス一覧を非表示または表示できます。
- 希望するメトリクス項目が一覧に表示されない場合は、**リフレッシュ**ボタンを選択します。

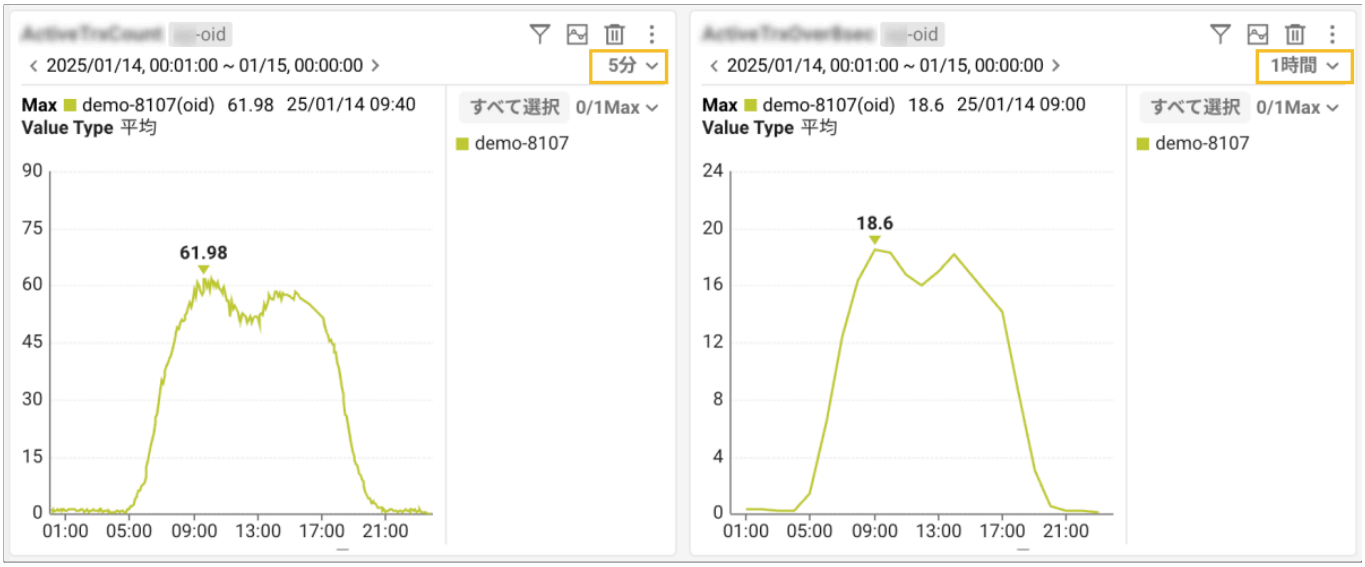
インターバルを設定する



チャートごとにインターバルを設定でき、データの集計単位を調整して、様々な時間間隔のデータを可視化できます。 **メトリック**パネルの右上にあるボタンを選択すると、インターバルオプションを選択できます。

メトリクスごとにデフォルトで設定されたインターバル時間が異なり、照会時間の範囲によってインターバル時間が変更されます。

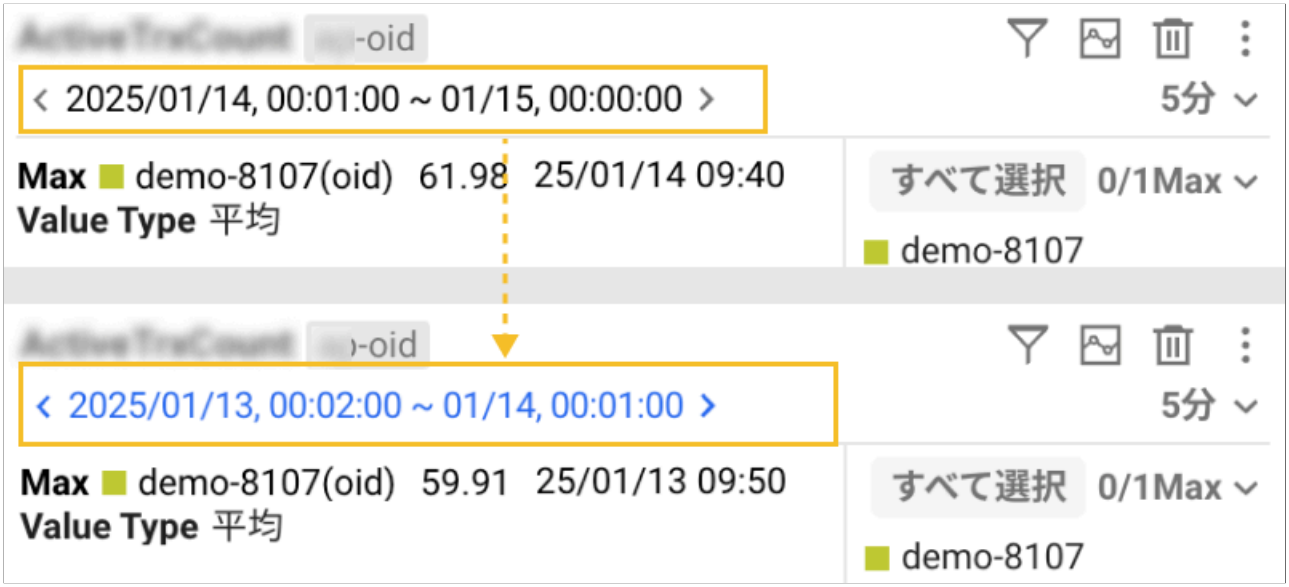
例えば、**5分間**インターバルは5分単位の平均値を表示し、**1時間**インターバルは、1時間単位の平均値を表示します。



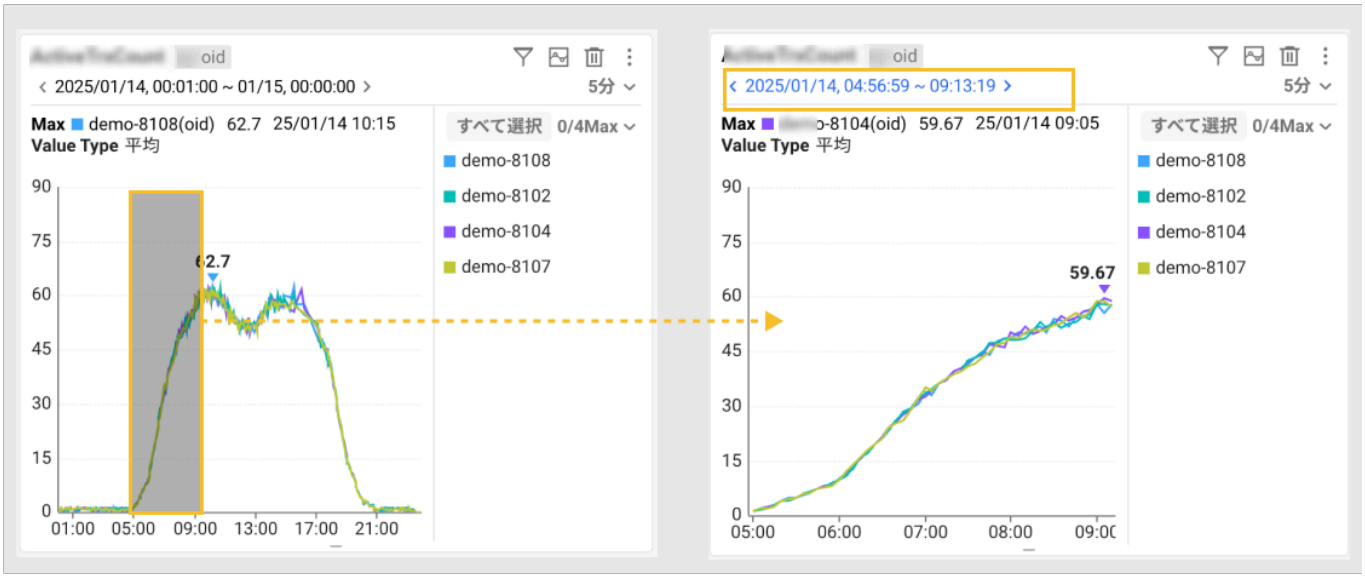
インターバルオプションを調整し、短い間隔の詳細データから長い間隔の全体的な推移まで、多様な観点からデータを分析できます。

照会時間を変更する

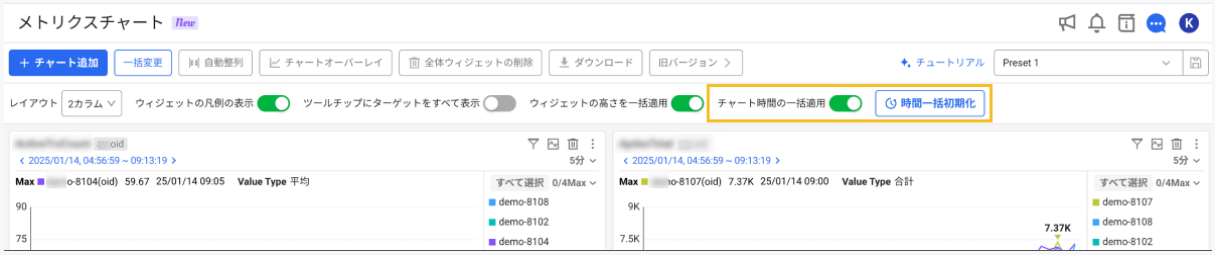
ウィジェットのチャート照会時間の範囲を変更するには、ウィジェット上部の < または > ボタンを選択します。 時間変更範囲は、**時間**オプションに設定された時間基準と同じです。 例えば、**時間**オプションで**1日間**に設定されている時、 < または > ボタンを選択すると、時間を1日単位で変更できます。



またはチャートの特定領域をドラッグすると、ドラッグした範囲に時間を変更して照会できます。



- 画面の上部にある**チャート時間の一括適用**オプションを有効にして、ウィジェットの時間を変更すると、すべてのウィジェットの時間を変更できます。



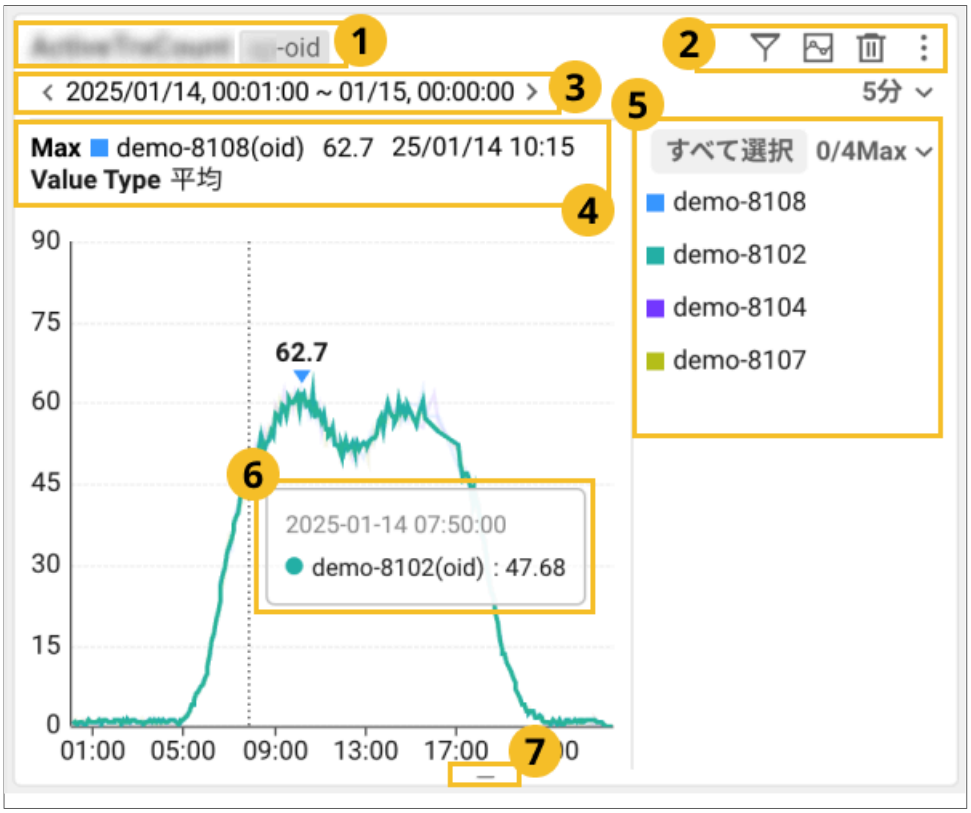
- 照会時間の範囲を変更した後、直近の時間に再度変更するには、ボタンを選択します。

  もし画面の上部にある**チャート時間の一括適用**オプションを有効にした場合、**時間バッチ初期化**ボタンが表示されます。 このボタンを選択すると、すべてのウィジェットの時間を初期化できます。

一括変更機能を利用してウィジェット全体の時間を変更し、モニタリング対象とインターバルも修正できます。 詳細については、次の文書を参照してください。

チャートウィジェットの詳細

チャートウィジェットに表示される各アイテムの詳細を確認します。



- 1 **メトリクス名とカテゴリ**: チャートウィジェットのメトリクス名を表示します。 メトリクスの右側には、照会対象のカテゴリをタグ形式で表示します。 ご利用の商品によりカテゴリ名称は異なる場合がございます。
  - *oid : 個々のエージェントを選択した時に表示されます。
  - project : プロジェクト単位で対象を選択した時に表示されます。
  - okind : エージェント設定で whatap.okind で分類されたグループを選択した時に表示されます。
  - onode : エージェント設定で、 whatap.onode で分類されたグループを選択した時に表示されます。
- 2 **ウィジェットオプション**: チャートウィジェットのオプションメニューです。 詳細については、次の文書を参照してください。 インターバルオプションの詳細については、次の文書を参照してください。
- 3 **時間**: チャートのデータを照会する時間範囲を表示します。 < または > ボタンを選択して照会時間範囲を調整できます。
- 4 **データ**: モニタリング対象のうち照会したデータの最大値(**Max**)とデータ結合方式(**Value Type**)を確認できます。
- 5 **凡例**: モニタリング対象を凡例として表示します。 凡例表示の詳細については次の文書を参照してください。
- 6 **チャート**: 照会したデータを時系列グラフで表示します。 マウスをグラフの上にかざすと、詳細な情報を確認できます。 チャートの特定領域をドラッグすると、ドラッグした範囲に時間を変更して照会できます。

- 7 **サイズ調整**: チャートウィジェットの真ん中の下部領域をドラッグすると、ウィジェットのサイズを調整できます。

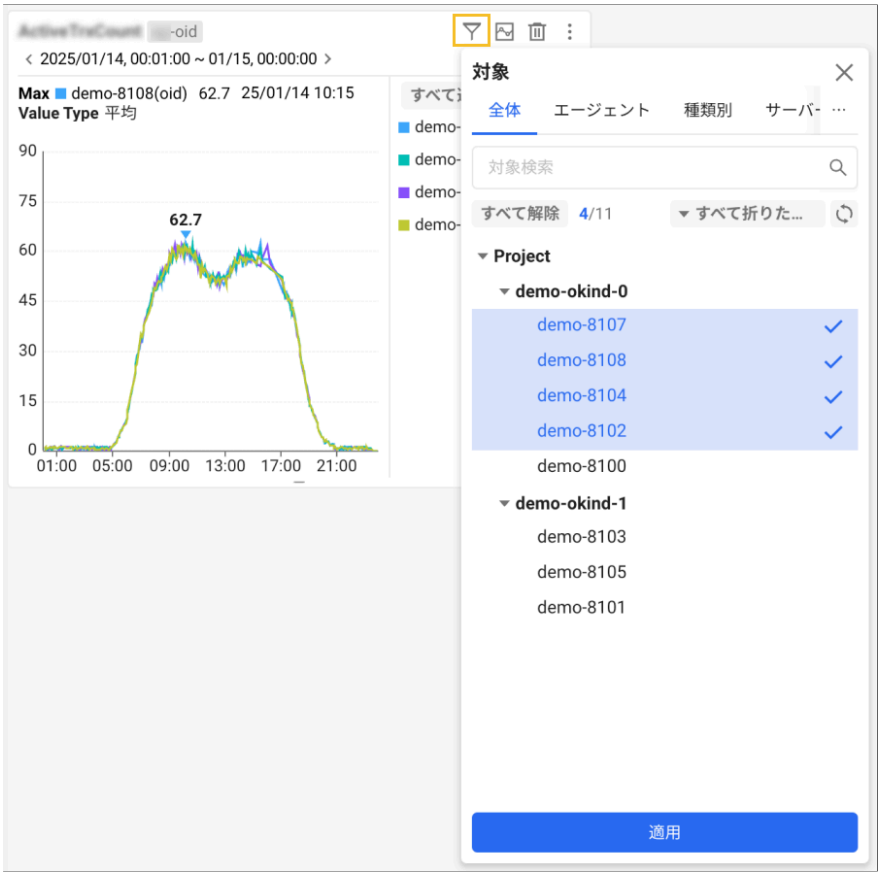
チャートウィジェットオプションを利用する

チャートウィジェットの右上にあるオプションボタンを使用すると、様々な機能が利用できます。

対象をフィルタリングする

画面に追加したチャートウィジェットに、モニタリング対象を追加または除外できます。

1. ボタンを選択すると、モニタリング対象を選択できる画面メニューが表示されます。



2. 画面の案内に従って希望する対象を選択します。

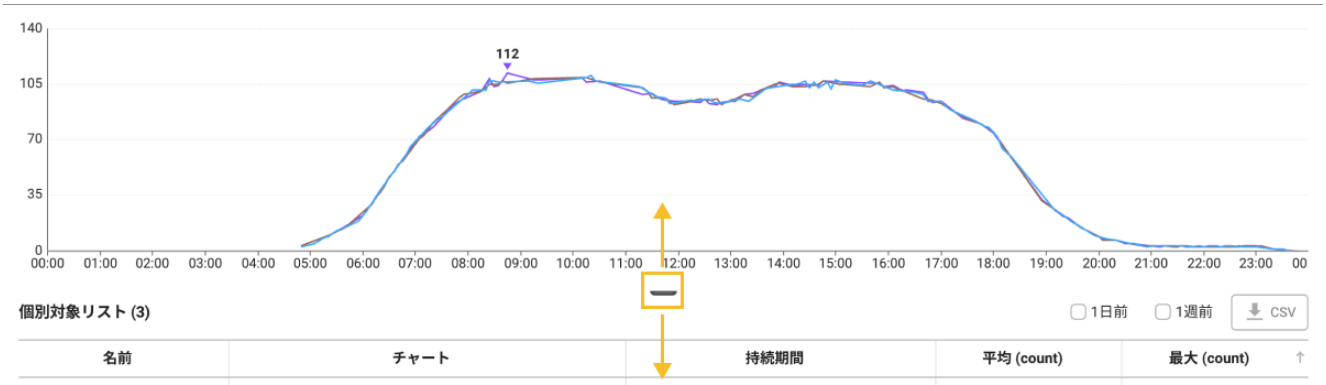
3. **適用**ボタンを選択します。

詳細を照会する

チャートのデータをより広い画面で詳細情報を確認するには🖼ボタンを選択します。 **詳細検索**画面が表示されます。個々の対象に対するチャートと平均および最大値を照会できます。



- **1日前**: 1日前のデータと比較して確認できます。
- **1週前**: 1週間前のデータと比較して確認できます。
- **CSV**: 照会したデータをCSV形式のファイルとしてダウンロードできます。
- グラフチャートの下部にあるアイコンをドラッグすると、チャートのサイズを調整できます。

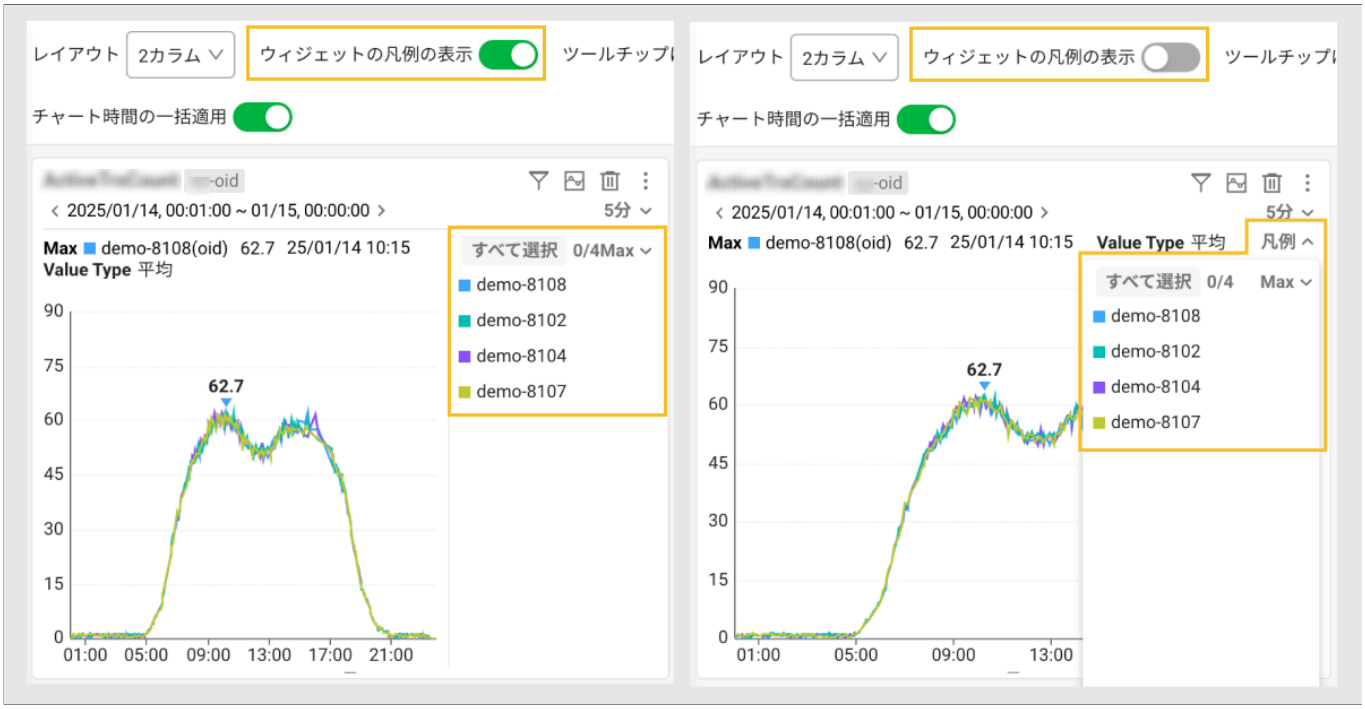


チャートウィジェットを削除する

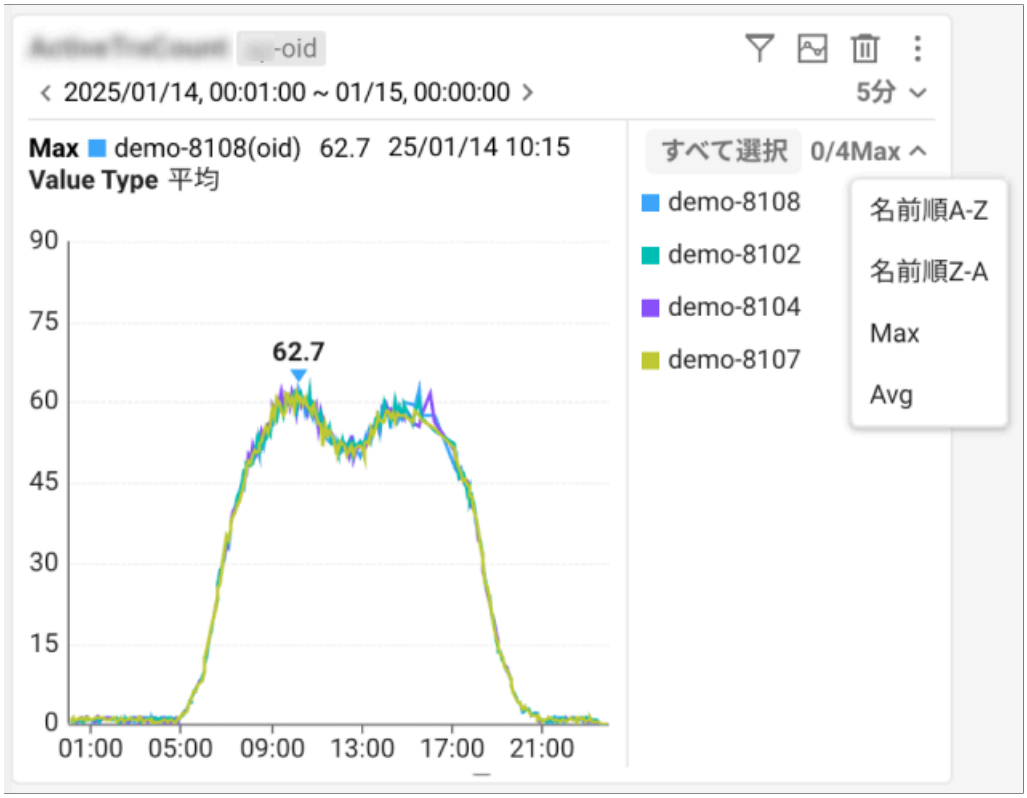
画面に配置したチャートウィジェットを削除するには、ボタンを選択します。

凡例を活用する

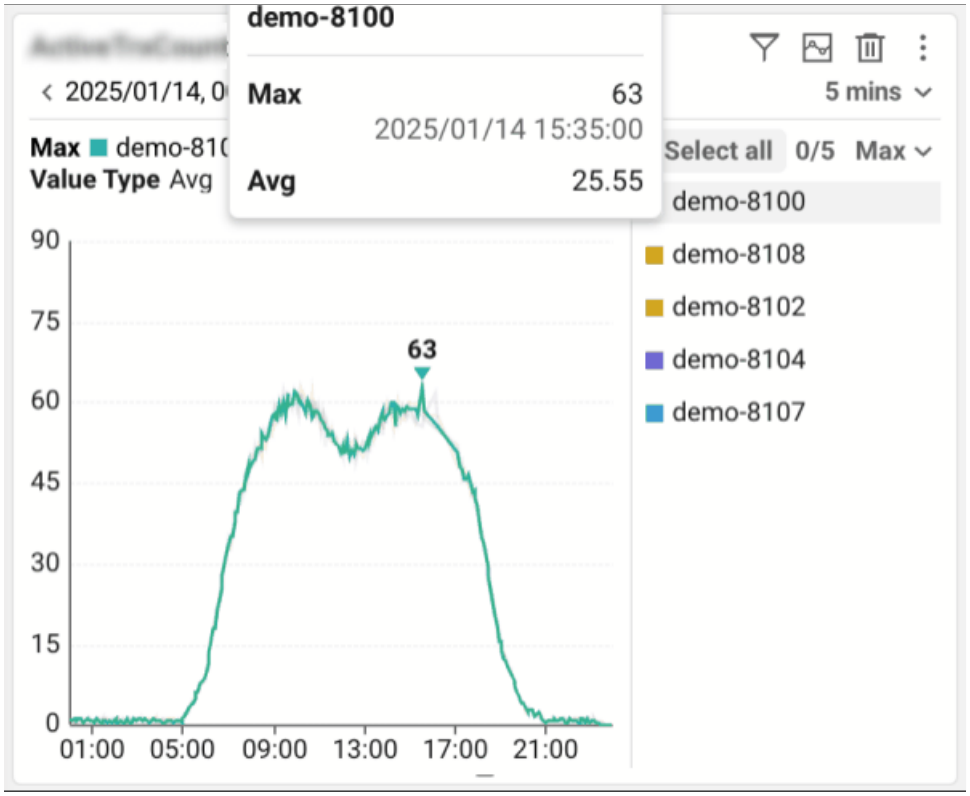
凡例を通じてチャートに表示されるモニタリング対象を確認し、希望する基準で整列したりフィルタリングできます。画面上部の**ウィジェットの凡例の表示**オプションをオンまたはオフにすると、ウィジェットのチャートの右側に凡例を表示・非表示できます。 オプションをオフにした状態では**凡例**をクリックしてモニタリング対象を確認して選択できます。



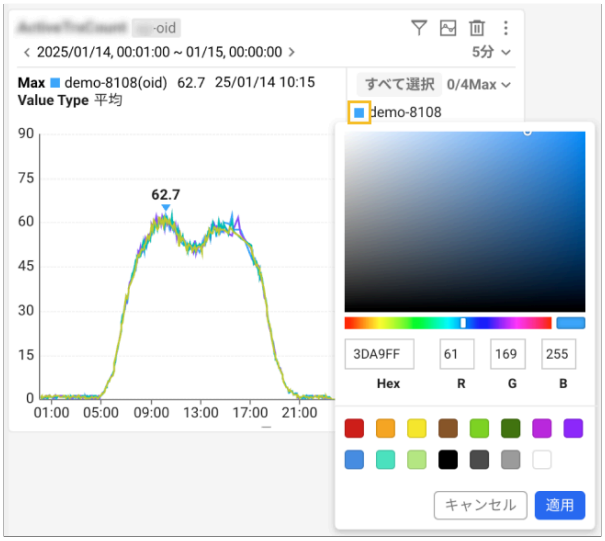
- **すべて選択/すべて解除**ボタンを選択してモニタリング対象を選択または解除できます。
- 凡例一覧では、モニタリング対象を名前順、最大値、平均値に基づいてソートできます。



- 凡例一覧の各項目にマウスをオーバーすると、そのモニタリング対象に対するグラフを強調して確認でき、Max値が記録された特定時間をツールチップで表示します。 または、凡例一覧の対象を選択してもフィルタリングできます。



凡例色を変更する



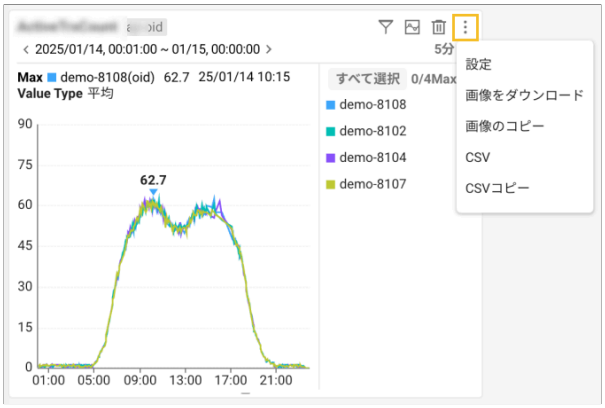
モニタリング対象に該当するグラフの色を変更できます。 凡例一覧から色を変更したい対象の左側にある四角い箱

を選択します。 希望する色を選択した後**適用**ボタンをクリックします。

> ⓘ 設定した色は、プリセット機能によって保存されません。 この機能は、今後のアップデートで提供される予定です。

追加オプションを利用する

チャートウィジェット右上にある ⋮ ボタンを選択すると、次の追加オプションが利用できます。



- **設定**
  - **自動Y軸を使用**: 照会したデータ範囲内でY軸の大きさを自動調整できます。
  - **最大値表示**: チャートのグラフ領域に最大値を表示できます。
- **画像をダウンロード** : チャートを画像形式のファイルとしてダウンロードできます。
- **画像のコピー**: チャートを画像形式でクリップボードにコピーできます。
- **CSV**: チャートで照会したデータをCSV形式のファイルにダウンロードできます。
- **CSVコピー**: チャートで照会したデータをCSV形式でクリップボードにコピーできます。

プリセットを設定する

ダッシュボードからユーザーが設定したウィジェットの設定とレイアウト状態を保存して読み込むことができます。ウィジェットのサイズを調整し、希望する場所に配置して新しいプリセットを作成できます。

> **プロジェクト修正**権限のあるメンバーのみがプリセットを保存または変更できます。 メンバー権限の詳細については、次の文書を参照してください。

新しいプリセットを作成する

1. ダッシュボードから、希望する形にウィジェットを配置してみてください。 サイズを変更して、頻繁に確認するウィジェットのみを配置することもできます。
2. 画面の右上に保存ボタンを選択します。
3. 新しいプリセット名を入力します。



4. **保存**ボタンを選択します。

プリセット一覧から新しく保存したプリセットを確認できます。

> - 新しく作成したプリセットに変更が加えられた場合は、再度プリセットを保存してください。 保存ボタンを選択し、同じ名前でプリセットを保存します。 既存のプリセットに変更を上書きします。
> - ダッシュボードの変更を保存せずに別のメニューに移動すると、変更は保存されません。
> - プリセットはプロジェクト単位で保存され、他のユーザーと共有できます。
> - プリセットに保存される対象は、ダッシュボードのレイアウトとダッシュボードに配置したウィジェット、ウィジェットで修正したオプション(**対象**, **間隔**)です。 画面上部の**時間**オプションは保存されません。

- プリセットを読み込んでからウィジェットの照会時間を変更するには、**一括変更**機能を利用します。 詳細については、次の文書を参照してください。

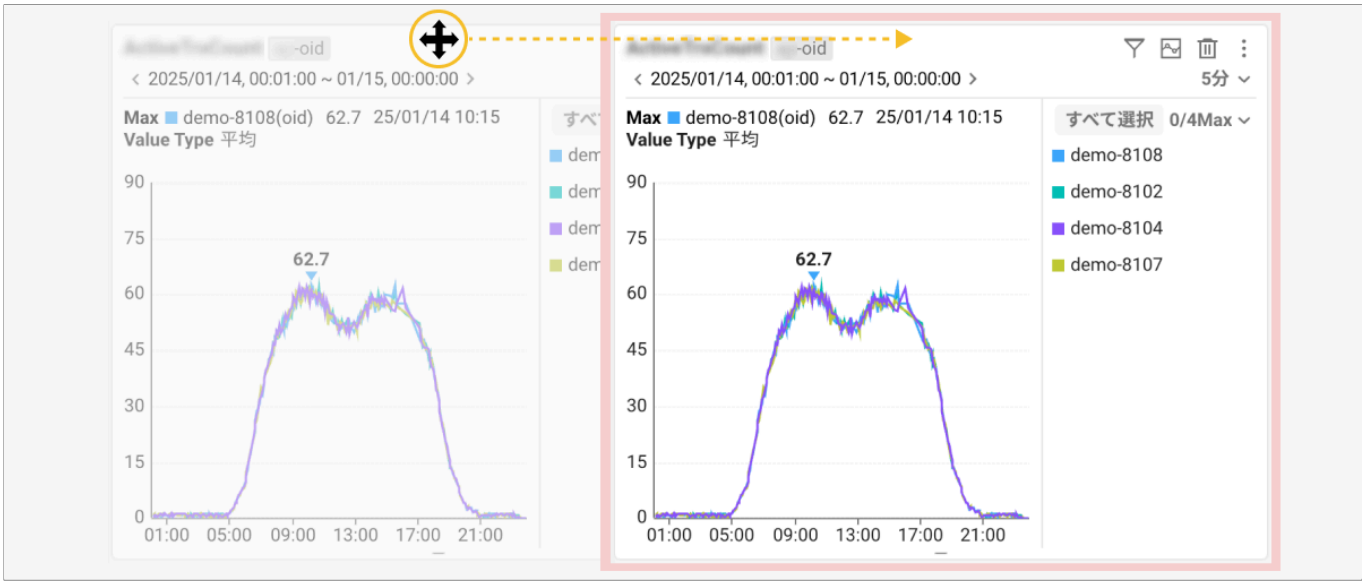
プリセットを削除する

使用していないプリセットは、プリセット一覧から削除できます。 プリセット一覧から削除する項目の右側にある ✕ ボタンを選択します。

ダッシュボードウィジェットの編集

ダッシュボードに配置したウィジェットをユーザーが希望する場所に配置したり、サイズを調整できます。

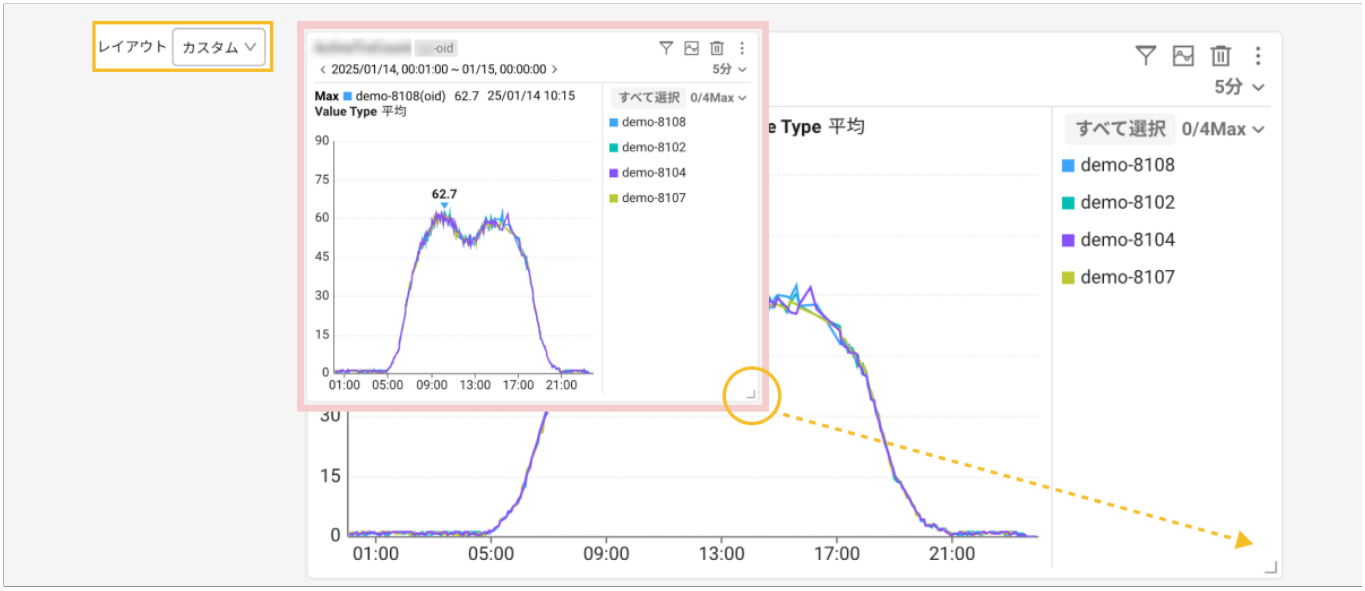
ウィジェットの移動



ウィジェットの上部にマウスをかざすと、✛に変更されます。 この時、クリックしたまま希望の位置にドラッグするとウィジェットを移動できます。

ウィジェットのサイズ変更

ダッシュボードオプションの**レイアウト**が**カスタム**に設定されていると、ウィジェットのサイズを調整できます。



ウィジェットの右下にある⌟をマウスでクリックした状態で希望するサイズにドラッグします。

> **レイアウト**が**カスタム**に設定されていない場合は、ウィジェットの高さだけを調整できます。 ウィジェットの真ん中の下部領域をマウスでクリックしたままでサイズを調整します。

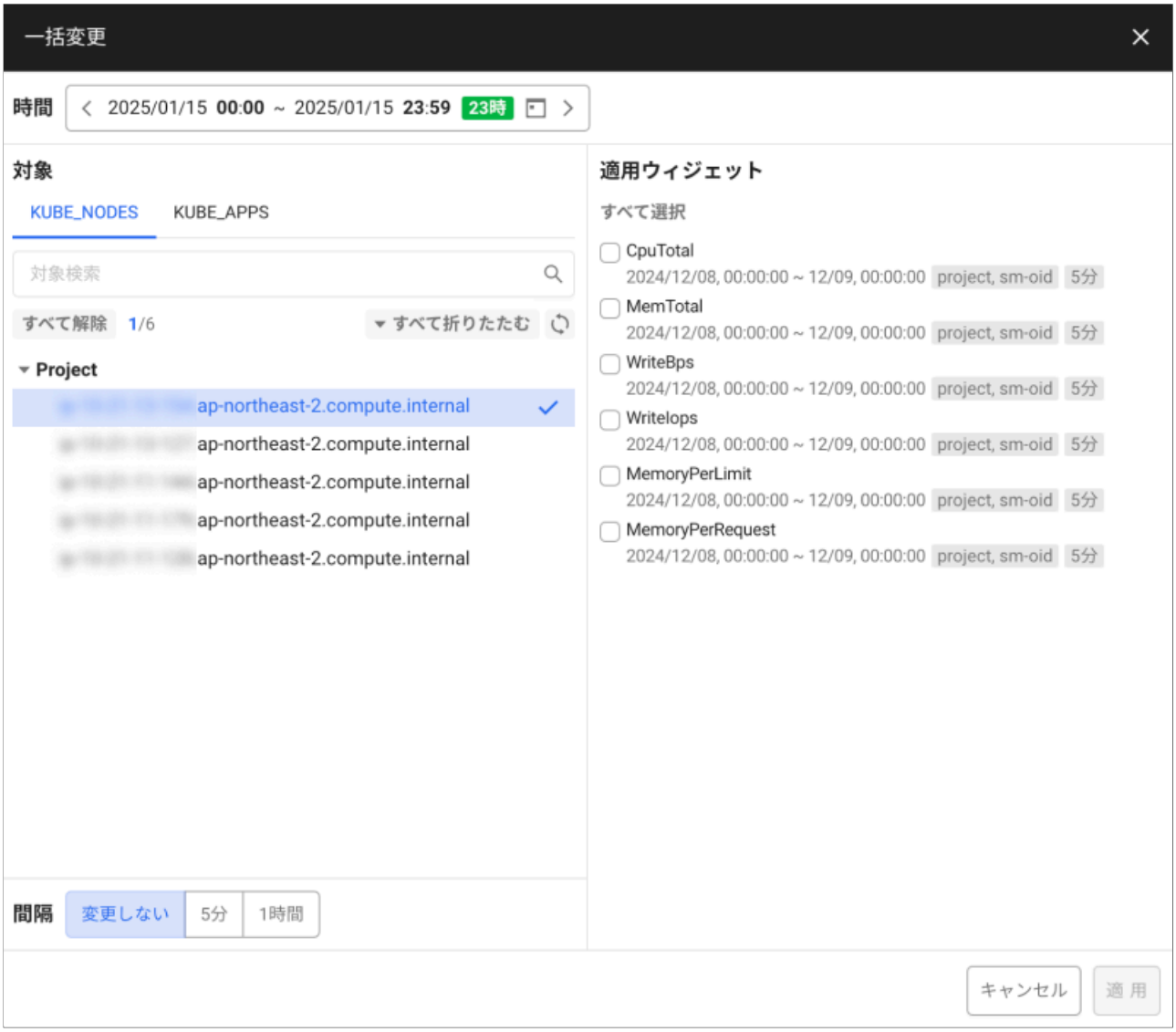
ダッシュボードオプションを利用する

画面上部にあるオプションメニューを利用して、チャートウィジェットに適用された様々な設定を一度に適用できます。 また、ダッシュボードのレイアウトを希望する形に設定できます。



チャートウィジェットオプションを一括変更する

画面に配置されたチャートウィジェットに、詳細オプションを一度に適用できる機能を提供します。 画面左上に**一括変更**ボタンを選択します。 **一括変更**画面が表示されたら、オプションを設定し、**適用**ボタンを選択します。



- **時間:** チャートのデータ照会範囲を設定できます。
- **対象:** モニタリング対象を選択します。 個々のエージェントを選択したり、エージェント設定を通じて分類したグループ単位で選択できます。

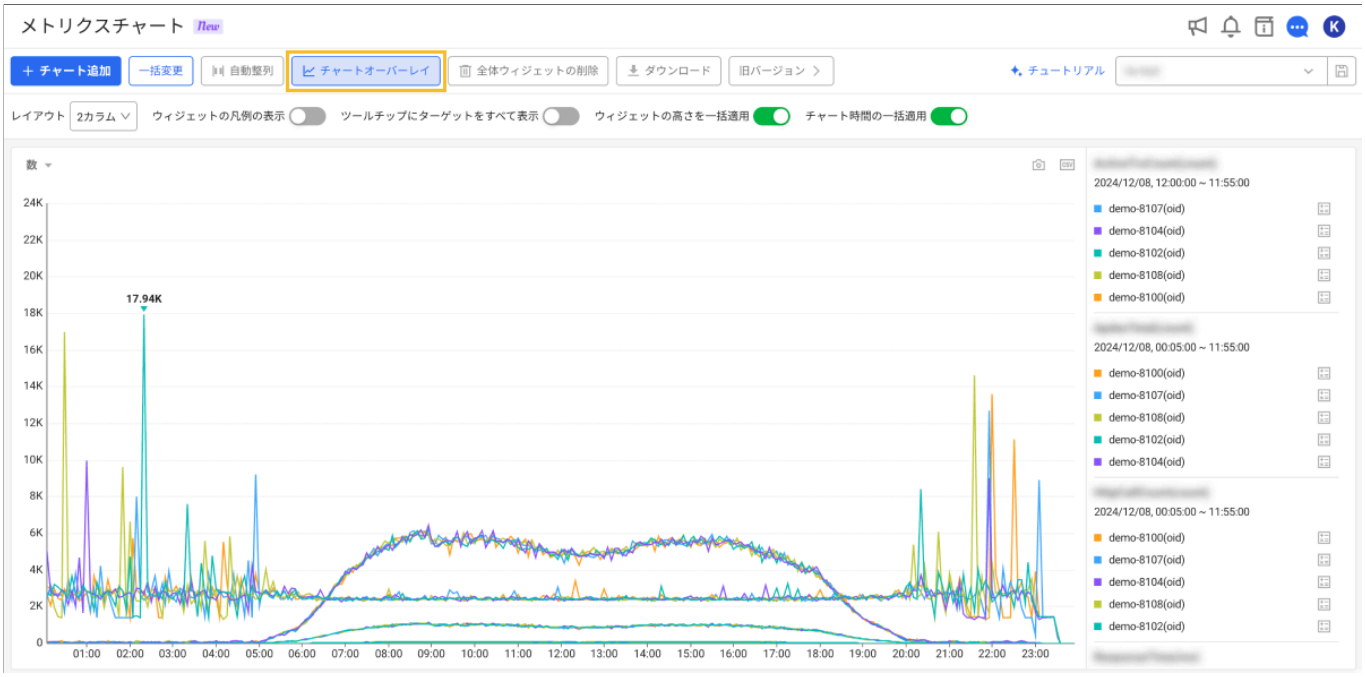
  - **エージェント**: 個別エージェントまたはすべてを選択できます。
  - **種類別**: エージェント設定で `whatap.okind` に分類されたグループ単位でエージェントをモニタリングします。
  - **サーバー別**: エージェント設定で `whatap.onode` に分類されたグループ単位でエージェントをモニタリングします。
- **間隔**: 様々な時間間隔でデータの集計単位を調整できます。
- **適用ウィジェット**: 変更するオプションを適用するチャートウィジェットを選択します。 選択したウィジェットに変更したオプションを一度に適用できます。

ダッシュボードを自動ソートする

ダッシュボードに配置したウィジェットの位置を自動的にソートするには**自動整列**ボタンを選択します。 ウィジェットの幅、高さが小さい順にウィジェットの配置を自動的にソートします。 幅が同じであれば、高さが小さい順にソートします。

複数のチャートを比較する

ダッシュボードに配置した複数のチャートデータを1つのチャートで確認できる機能です。 この機能により、同時間帯の様々なメトリクス情報を他のメトリクスと比較しながら確認できます。 **チャートオーバーレイ**ボタンを選択します。 画面にオーバーレイチャートを配置できます。



- オーバーレイチャート左上の**数**を選択すると、x軸チャートのデータ基準を変更できる一覧が表示されます。 希望する項目を選択して、チャートのデータを比較できます。
- : 照会中のデータに基づいて、チャートを画像形式のファイルとしてダウンロードできます。
- CSV: 照会中のデータに基づいて、チャートをCSV形式のファイルとしてダウンロードできます。
- 右側のセクションでは、メトリクスごとのエージェントの一覧を確認できます。 個々のエージェントの色は、チャートのグラフの色と一致します。

配置されたウィジェットを一括削除する

ダッシュボードに配置された複数のウィジェットを一度に削除できます。 **全体ウィジェットの削除**ボタンを選択します。 確認メッセージが表示されたら**削除**ボタンを選択します。

CSVファイルでダウンロードする

ダッシュボードに配置したウィジェットのデータをCSV形式のファイルとしてダウンロードできます。 **ダウンロード** > **CSV**を選択します。 時間帯のメトリクスのデータを確認できます。

イメージファイルでダウンロードする

ダッシュボードに配置されたすべてのウィジェットを1つのイメージファイルでダウンロードできます。 **ダウンロード** > **画像をダウンロード** を選択します。 ダッシュボード画面をそのままイメージで保存できます。

以前のバージョンを利用する

以前のバージョンの**メトリクスチャート**メニューを利用するには、**旧バージョン**ボタンを選択します。 または **サイトマップ** > **分析** > **メトリクスチャート** *Old* メニューパスで確認できます。

> **メトリクスチャート** *Old* メニューの詳細は、次の文書を参照してください。

その他のオプションを利用する

次のその他のオプションで、ダッシュボードのレイアウトを調整したり、ウィジェットに適用されたオプションを簡単に一括設定できます。



- **レイアウト**: ダッシュボードで列の数、つまり1行に配置できるウィジェットの数を設定できます。 最大4列まで設定できます。 **カスタム**を選択すると、列の数に関係なくウィジェットを配置できます。

  > **カスタム**では、ウィジェットの幅と高さを自由に調整できます。 他のオプションを選択すると、ウィジェットの高さだけを調整できます。

- **ウィジェットの凡例の表示**: オプションをオンまたはオフにすると、ウィジェットのチャートの右側に凡例を表示・非表示できます。 オプションをオフにした状態では**凡例**をクリックしてモニタリング対象を確認して選択できます。 凡例の活用についての詳細は次の文書を参照してください。
- **ツールチップにターゲットをすべて表示**: チャートにマウスかざすと、そのグラフの位置に該当する値を確認できます。 マウスが位置した時間帯のすべての対象のデータを確認するにはオプションをオンにします。
- **ウィジェットの高さを一括適用**: オプションをオンにすると、すべてのウィジェットの高さを同じく設定できます。

- **チャート時間の一括適用**: チャートウィジェットのデータ照会時間を変更する時、他のウィジェットのチャート時間も一緒に変更するにはオプションをオンにします。

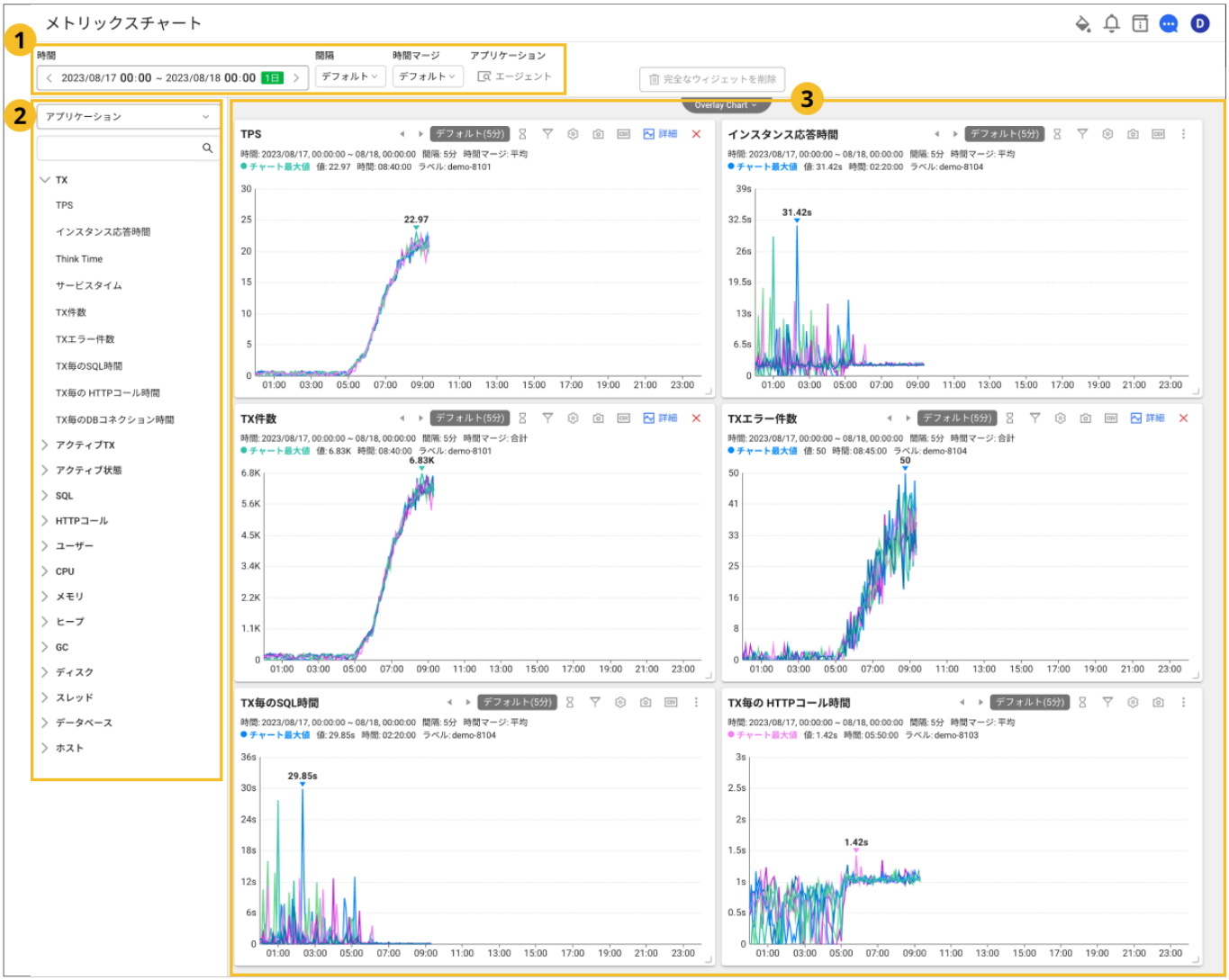
チュートリアルを利用する

画面の右上の**チュートリアル**ボタンを選択すると、Service 2.8.0 アップデート以降の変更を確認できます。

メトリクスチャート

サイトマップ > 分析 > メトリクスチャート *Old*

メトリクスチャートメニューでは、モニタリング対象から収集されたメトリクスデータを次のようにチャートで照会できます。時間と指標の指定が必須です。



上部オプション

1領域の**メトリクスチャート**の上部オプションからチャートの時間範囲とモニタリング対象エージェントを指定できます。

- **時間**：時間の範囲でX軸の開始と終了を指定できます。

  > **時間**オプションの利用方法の詳細については。

- **間隔**：時間間隔でX軸データの間隔を指定できます。
- **時間マージ**：データマージ方式の1つを選んで、指定した時間間隔内のデータをマージできます。

  例）**平均**は、1時間のデータ平均値を意味します。

- **エージェント**：照会するエージェントを指定できます。指定しない場合は、すべてを対象に照会します。

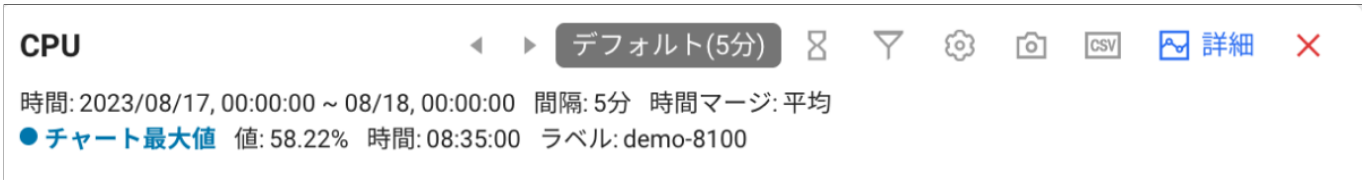
指標一覧

2領域は、オプションを検索する指標のリストです。まず、**カテゴリ**を選択します。 **カテゴリ**を選択すると、そのカテゴリに含まれる指標が表示されます。目的の指標を選択してください。**カテゴリ**と指標を選択すると、1領域の上のメニューで指定した時間範囲のデータから3領域でチャートウィジェットを照会できます。

> **データ結合**
>
> データのマージは、**オブジェクトのマージ**と**時間マージ**方法を提供します。
>
> - **時間マージ**は、ソースデータからフィールド値が同じデータを一定の間隔でマージします。
> - **オブジェクトのマージ**は、異なるフィールド値を持つデータの中からタグが一致する場合、そのデータをマージします。

チャートウィジェット

3領域のチャートウィジェットの左上で指標名を確認できます。チャートウィジェットの右上で次のオプションを確認できます。



- タイムシフト：◀ ▶左矢印または右矢印ボタンを使用して、選択した時間範囲だけ-1または、+1ずつ移動することができます。

  例えば、時間範囲が2月13日00:00~2月14日00:00の場合、◀左矢印を選択すると、2月12日00:00~2月13日00:00までのデータが表示できます。

- 間隔/時間のマージ：1トップメニューで指定した間隔と時間のマージを修正できます。
- モニタリング対象：アイコンを選択して、モニタリング対象を指定できます。選択しない場合は、すべて表示されます。
- 時間比較：アイコンを選択すると、同じ指標の以前の時間帯の推移を比較できます。
- スナップショット：アイコンを選択すると、ウィジェットのオプションを除くチャートのスナップショットを取得できます。
- CSV：CSVアイコンを選択して、チャートを描画したデータをCSVファイルとしてダウンロードできます。
- **詳細**：アイコンを選択して詳細を表示します。モニタリング対象が多い場合、モニタリング対象の指標推移を特定して確認できるため便利です。

> ⓘ メトリクスチャートウィジェットの上部にオプションが表示されない場合は、⋮アイコンを選択します。

ウィジェットを削除する

画面に配置したすべてのウィジェットを削除するには、**全体ウィジェットの削除**ボタンを選択します。

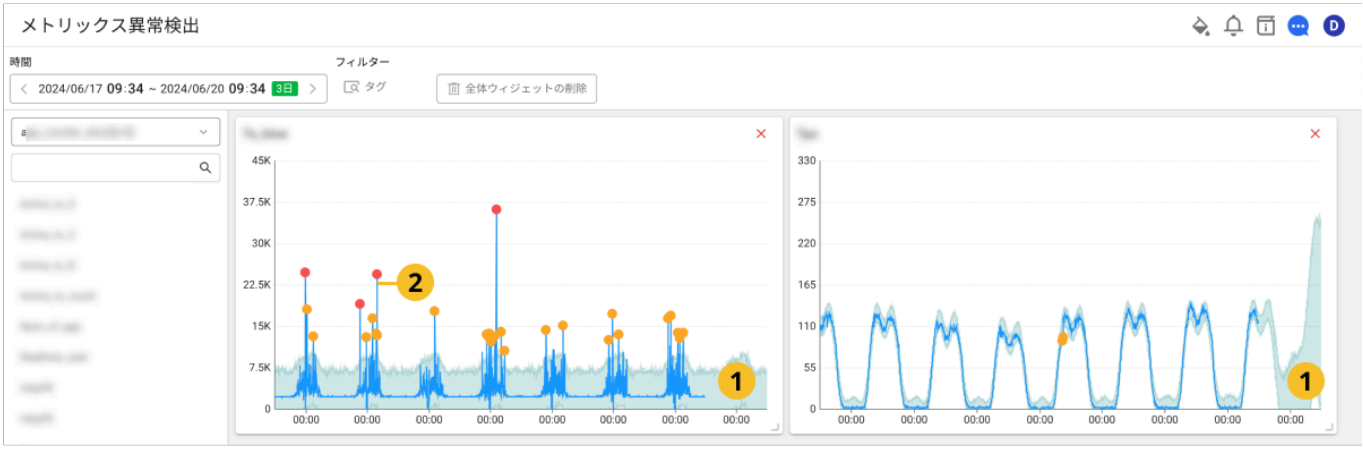
メトリクス異常検出

ホーム画面 > プロジェクト選択 > **分析** > **メトリクス異常検出**

さまざまなメトリクスのパターンをAIが学習した予想パターンと比較できます。予想パターンを外れた異常探知をグラフチャートで確認できます。過去のデータに基づいて繰り返されるパターンを確認し、今後の指標値の予測に活用できます。

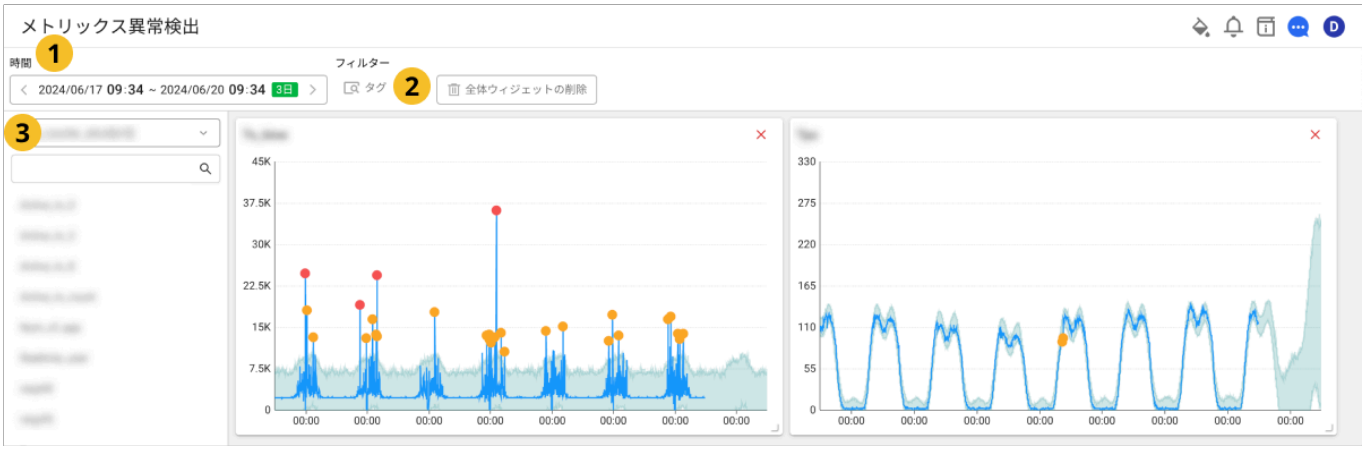
- 画面に配置したウィジェットは、別のメニューに移動すると保存されずに初期化されます。
- パターン分析と異常値検出機能を除くと**分析** > **メトリクスチャート**メニューに似ています。
- **異常値**(Anomaly Detection)アラート通知機能の技術的根拠は、**メトリクス異常値**です。**異常値**アラート通知設定の詳細については、次の文書を参照してください。

ウィジェットを確認する



- 1 明るい色のグラフ領域は、AIによって分析した予想されるパターンです。
- 2 青色グラフは、プロジェクトのメトリクスの推移です。
- AIによって分析された予想パターンから外れるとオレンジ色と、赤色のステップでグラフにドットが表示されます。予想されるパターンの範囲から大幅に外れた値を赤色として表示します。

ウィジェットを配置する



1. 1 **時間**で、目的の時間間隔を設定してください。最大1か月間隔まで設定できます。
2. 2 **フィルター**でメトリクスの範囲を選択します。
3. 以下の一覧でモニタリングするメトリクスを選択します。

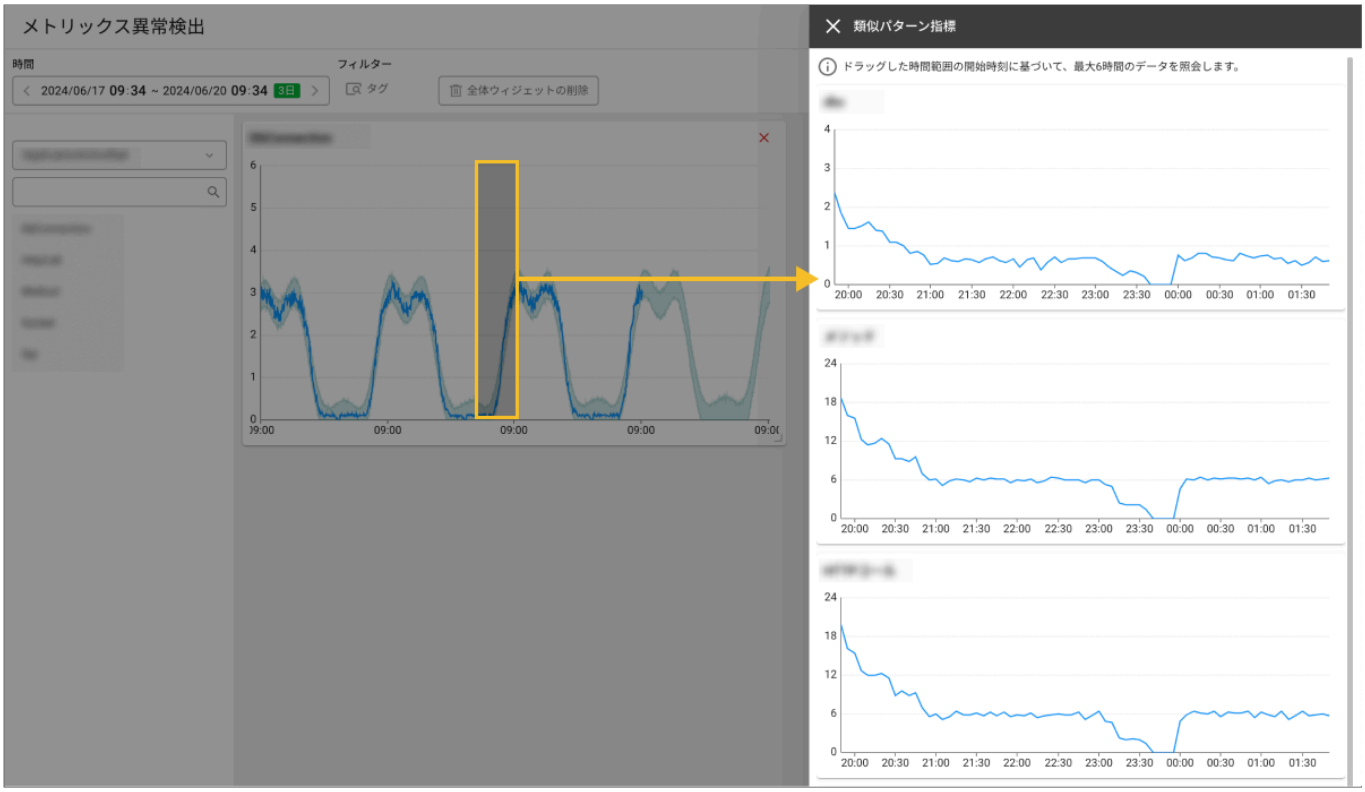
選択したメトリクスを画面の右側に配置します。

- モニタリング対象を選択してメトリクスを絞り込んで確認するには、2 **タグ**を選択します。画面右側にタグ選択リストが表示されます。目的の項目を選択し、ウィジェットを追加します。
- 画面に配置したウィジェットは、**時間**または**タグ**値を変更してもチャートに反映されません。
- 画面に配置したウィジェットを削除するには、ウィジェット右上の✕ボタンを選択します。
- ウィジェットの左上を選択した状態でドラッグすると、ウィジェットの位置を変更できます。
- ウィジェットの右下を選択した状態でドラッグすると、ウィジェットのサイズを変更できます。

ウィジェットを削除する

画面に配置したすべてのウィジェットを削除するには、**全体ウィジェットの削除**ボタンを選択します。

関連メトリクスを確認する

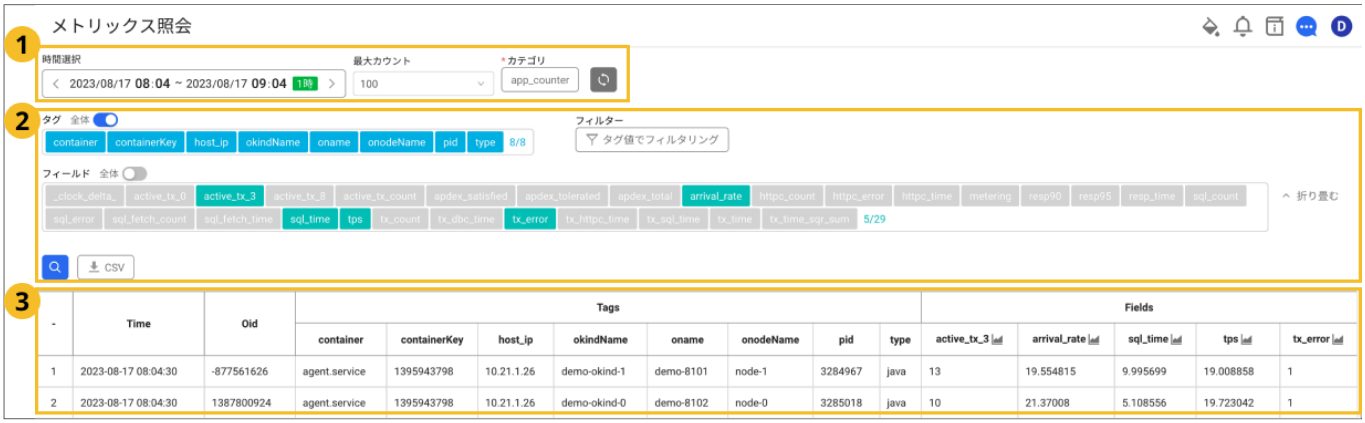


ウィジェットでチャートの一部の領域をドラッグします。画面の右側に**類似パターン指標**画面が表示されます。ドラッグした時間範囲の開始時間に基づいて、最大6時間の関連メトリクスを照会できます。

メトリクス照会

ホーム画面 > プロジェクト選択 > **分析** > **メトリクス照会**

メトリクス照会メニューからタグベースで特定のメトリクスを照会できます。



1 時間とカテゴリの選択

1 領域でメトリクスが収集された**時間選択**と**最大カウント**および**カテゴリ**を指定できます。 **時間選択**と**カテゴリ**は、必ず指定してください。

- **時間選択**：メトリクスが収集された時間を指定して照会できます。 既定値は `1時間` です。 基本オプションとして提供する照会時間以外に、ユーザーが**時間選択**タブを選択して日付と時間を指定できます。

  > ⓘ **時間選択**オプションの利用方法の詳細については。

- **最大カウント**：3 一覧から参照するメトリクスの最大数を指定できます。 `10` 、 `50` 、 `100` 、 `200` 、 `300` 、 `1000` 、 `2000` 、 `3000` 個まで設定できます。
- **カテゴリ**：関連指標の分類単位です。 **カテゴリ**タブを選択し、**カテゴリ**を指定できます。
- **カテゴリ、タグ、フィールドオプションの更新**：**カテゴリ**、**タグ**および**フィールド**オプションを再読み込みできます。

2 タグとフィールドの選択

2 領域で**タグ**と**フィールド**を選択します。 ユーザーが個別に指定しない場合、デフォルト設定が完全な選択になります。

- **タグ**：収集された対象を区分できる固有情報データです。
- **フィールド**：モニタリング対象から収集された指標です。
- **フィルター**：**タグ値でフィルタリング**ボタンを選択し、タグ値を設定してフィルタリングできます。

  例） `oname` の値を `demo-8101` に設定すると、フィルタリングしたデータを表示することができます。

- **検索**：条件を設定して**検索**アイコンを選択すると、3 領域でそのメトリクスの原本データを検索できます。
- CSV：メトリクスの原本データをCSVファイルとしてダウンロードできます。

3 メトリクステーブル

収集されるメトリクスを事前に特定できないため、収集されるすべてのメトリクスの原本データを確認することが重要です。 上記の条件領域で目的の条件を設定した後3 領域で、そのメトリクスの原本データをテーブル形式で検索できます。 ユーザーが**タグ**と**フィールド**の各条件を指定することによって、テーブルのカラムが変更されます。

> - メトリクスの参照時、**時間選択**と**カテゴリ**を指定する必要があります。
> - メトリクス参照時、**タグ**と**フィールド**の指定はオプションです。

ログ

Kubernetesに関する様々なログが確認できます。WhaTap Kubernetesが提供するモニタリングカテゴリは次のとおりです。

カテゴリ	説明
`#K8sEvent`	• Kubernetesで発生したイベントがデフォルトで保存および作成されるログ • ユーザー設定に関係なく基本作成
`#WhatapEvent`	• WhaTapイベント設定によって発生したイベントが格納されるログ • ユーザー設定に関係なく、デフォルトで作成されます
`containerStdout`	• コンテナ標準出力ログ • ユーザー設定時に作成 • ノードエージェントに`logsink_enabled=true`設定を追加したした場合
`AppLog`	• コンテナ内アプリケーションログ • ユーザー設定時に作成 • アプリケーションエージェントに`logsink_enabled=true`設定の追加時

> ⓘ 設定によって重複したログ内容が保存される場合があるため、重複がないか必ず確認してください。

containerStdOutログフィルターの設定

ホーム画面 > プロジェクト選択 > **管理** > **エージェント設定**

Kubernetesログで必要な情報を選別するためのフィルタリングオプションを案内します。 フィルタリングにより、許可または除外するログを設定できます。

1. **管理** > **エージェント設定** に移動してください。
2. **プロジェクトデフォルト** タブで、**オプション作成** 方式から **直接入力** をクリックしてください。



ログネーミングのルール

Kubernetes containerStdOutログはノードの*/var/log/containers/*パスの下位に生成されます。 次のようなネーミングルールに従っています。

```
<podName>_<namespace>_<containerName>-<containerId>.log
```

ログフィルターオプション

- **log_filter_enabled** bool

  既定値 false

  ログフィルターの使用有無を設定します。

- **log_filter_allow_list** list

  許可するログの一覧を設定します。 フィルタリングの適用時に含める項目を意味します。

- **log_filter_deny_list** list

  除外するログの一覧を設定します。 フィルタリングの適用時に除外する項目を意味します。

ログフィルター動作

`log_filter_enabled` オプションが有効になっている場合にのみ、ログフィルターが動作します。 該当オプションの値が `true` に設定された状態で、 `log_filter_allow_list` および `log_filter_deny_list` を通じてログを選択的に収集できます。

ログフィルターはブラックリストベースで動作します。 例えば、 `log_filter_enabled` 活性化した後 `log_filter_allow_list` オプションに追加する項目を設定しない場合、デフォルトではすべてのコンテナのログ収集がブロックされます。 また、ログフィルターはallowルールよりdenyルールの優先順位が高いです。 同一の項目がallowとdenyの両方に設定されている場合、denyルールが優先的に適用され、そのログは収集されません。

ログフィルターの例

単一ネームスペースログの収集

- 許容 **log_filter_allow_list** `*_infra_*`

`infra` ネームスペースに存在するすべてのコンテナが収集対象として特定されます。

マルチネームスペースログの収集

- 許容 **log_filter_allow_list** `*_infra_*`、`*_monitoring_*`

`infra` 、 `monitoring` ネームスペースに存在するすべてのコンテナが収集対象として特定されます。

特定の単語を含むPodを除く

- 許容 **log_filter_allow_list** `*_*_*`
- 除外 **log_filter_deny_list** `*prod*_*_*`

すべてのコンテナが収集対象として特定されますが、 `podName` に `prod` が含まれているログは除外されます。

特定の単語を含むPodログの収集

- 許容 **log_filter_allow_list** `*prod*_*_*`

`podName` に `prod` が含まれるPodのログのみが収集されます。

マルチネームスペースログの収集と特定の単語を含むPodを除く

- 許容 **log_filter_allow_list** `*_infra_*、*_monitoring_*`
- 除外 **log_filter_deny_list** `*prod*_*_*`

`infra`、`monitoring`ネームスペースのログが収集され、`podName`に`prod`が含まれているログは除外されます。

コンテナ名内の特定の単語で始まるコンテナログを除外

- 許容 **log_filter_allow_list** `*_*_db-*`

すべてのネームスペースで`containerName`が`db`で始まるコンテナのログは除外されます。

ライブテール

ホーム画面 > プロジェクト選択 > ログ > ライブテール

WhaTap Kubernetesモニタリングカテゴリの内容をリアルタイムで確認できます。ライブテールの詳細については、次の文書を参照してください。

ログトレンド

ホーム画面 > プロジェクト選択 > ログ > ログトレンド

時間範囲を設定することで、保存されたログの推移を確認できます。ログトレンドの詳細については、次の文書を参照してください。

ログの検索

ホーム画面 > プロジェクト選択 > ログ > ログ検索

時間範囲を設定し、保存されたログ内容が検索できます。例えば、特定のコンテナのStandard Outログの確認したり、Deploymentに属するすべてのコンテナのStandard Outログをまとめて確認できます。特定のキーワードをハイライトして表示することもできます。

ログトレンドの詳細については、次の文書を参照してください。

ログエクスプローラー

ホーム画面 > プロジェクト選択 > ログ > ログ検索

この機能では、ログデータを迅速に検索、フィルタリング、分析できます。ログエクスプローラーの詳細については、次のドキュメントを参照してください。

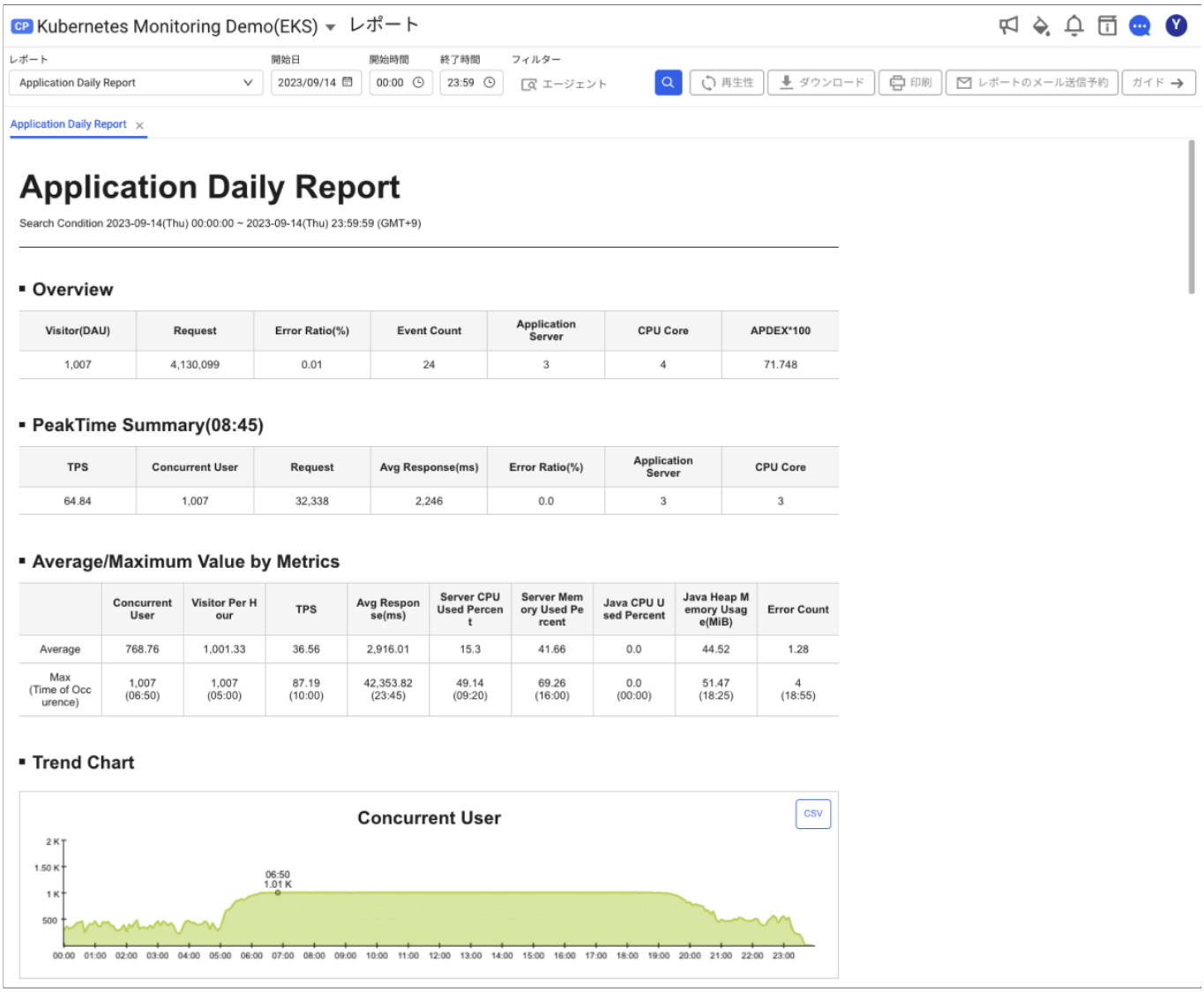
ログ設定

ホーム画面 > プロジェクト選択 > ログ > 設定

ログの有効化など、ログに関するさまざまな設定を行うことができます。ログ設定の詳細については、次の文書を参照してください。

レポート

画面の上に**レポート**で ボタンをクリックすると、日刊、週刊、月刊に分類された各種分析レポートを確認できます。希望するレポート様式を選択し、照会したい日付と時間、フィルターオプションを設定してください。 ボタンを選択すると、選択したレポートを確認できます。



レポートは、個々のプロジェクトに対してサービスの利用状況とトラブル発生の履歴を報告する文書です。サービスモニタリング担当者は、レポートを通じて関連部署の担当者と状況を共有します。モニタリングデータの分析は、サー

ビスの改善方向を定める指標となるため、非常に重要です。但し、複数のダッシュボードのデータの収集と文書化は面倒な場合があります。

WhaTapの**レポート**メニューは、**レポート作成業務の自動化**をサポートします。毎週報告書を作成すること、決まった時間に報告書を共有すること、様々な書式を管理することなど、すべて**レポート**メニューから解決できます。

- レポートをダウンロードするには、ダウンロードボタンを選択します。htmlファイル形式でダウンロードできます。
- レポートを印刷するには、印刷ボタンを選択します。
- レポートを送信予約するには、レポートメール送信予約ボタンを選択します。

レポートと統合レポートの違い

レポートは、1つのプロジェクトのレポートを生成するメニューです。したがって、プロジェクトを選択した後、左側のレポートメニューからアクセスできます。統合レポートは、複数のプロジェクトのレポートを生成するメニューです。プロジェクトを選択せずに、ホーム画面の基本メニューからアクセスできます。統合レポートの詳細については、次の文書を参照してください。

レポート種類

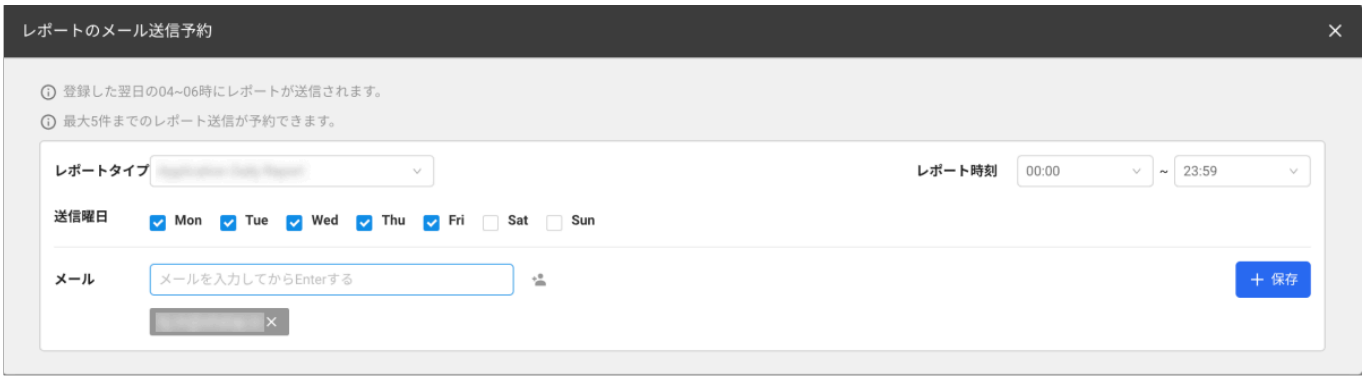
Kubernetesに関連する様々なレポートを確認できます。WhaTap Kubernetesが提供するレポートのタイプは次のとおりです。

- 日次アプリケーションレポート
- 日次アプリケーションの種類別しきい値超過トランザクション一覧
- 日次メトリクス推移
- 日次アプリケーションエラー統計レポート
- 週次アプリケーションレポート
- 月間アプリケーションレポート
- Pod別Cpu／Memoryの日次レポート
- Pod別Cpu／Memoryの週次レポート

- 日次サーバーレポート
- 週次サーバーレポート
- 月間サーバーレポート

メール送信予約

定期的にレポートをメールで受け取りたい場合は、レポートのメール送信予約を選択します。出社直後に実行していた様々なサービス点検に関する手続きをメール確認に置き換えることができます。



統合レポート

ホーム画面 > **統合レポート**

ユーザーは、複数のプロジェクトを統合したレポートを作成および共有できます。レポートの種類と含めるプロジェクトを選択してレポートを生成することができ、生成一覧で作業の進行状況を確認できます。特定のプロジェクトでエラーが発生すると、エラーの内容を確認できます。作成が完了したレポートは、すぐに照会するか、ダウンロードおよび印刷機能で共有できます。



> **統合レポート**メニューは**Application**、**Server**、**Kubernetes**、**Database**商品のプロジェクトに関するレポートのみ作成できます。

主な機能の案内

統合レポートメニューで提供する機能は次のとおりです。

- **マルチプロジェクトサポート**

  複数のプロジェクトを選択して一度にレポートを生成できるので便利です。プロジェクトで利用できる**レポート**メニューは、そのプロジェクトに関するレポートのみ作成できます。

- **タスクごとのレポート作成**

  レポート生成機能はタスクごとにプロジェクトを選択し、テンプレートとして保存できます。通常タスクごとにプロジェクトを分割して使用するマイクロサービスアーキテクチャ(Micro Service Architecture、MSA)の環境で有効な機能です。

- **大容量データの最適化**

  レポートの生成時間が大幅に短縮されます。また、一部のプロジェクトレポートの生成中にエラーが発生した場合でも、レポートの結果を表示できます。

- **レポート作成中に同時作業が可能**

  レポートを作成すると同時に、他の業務を進めます。プロジェクトで利用できる**レポート**メニューは、レポート作成タスクを完了するまで待つ必要があります。

  統合レポートメニューからレポートの生成を開始した後、別のページに移動できます。また、レポートの結果一覧から進捗状況と生成されたレポートを照会できます。

- **レポート作業の共有**

  同じプロジェクト権限を持つユーザーが作成したレポートを表示できます。関連するレポートがすでに作成されているかを確認できるため、重複の作成を回避できます。

レポートと統合レポートの違い

プロジェクトで利用できる**レポート**メニューでは、そのプロジェクトに関するレポートを作成できます。初期画面でプロジェクトを選択し、サイドメニューの**レポート**または**統計/レポート**メニューでアクセスできます。

統合レポートメニューでは、複数のプロジェクトに関するレポートを作成できます。プロジェクトを選択することなく、初期画面でアクセスできます。

レポート作成

レポートの種類を選択

レポートタイプ一覧で希望するレポートタイプを選択すると**レポート生成**ボタンが有効になります。レポート名の前の**アイコン**は、レポート作成に対応するプロジェクトプラットフォームを意味します。

- **Application**プロジェクトプラットフォームを示します。
- **Database**プロジェクトプラットフォームを示します。
- **Kubernetes**プロジェクトのプラットフォームを示します。
- **Server**プロジェクトプラットフォームを示します。



詳細設定を行う

レポート生成画面が表示されたら、各項目を設定します。

- **題名**: 同じ種類のレポートが複数ある場合、名前を指定するとレポートの結果を区別するときに役に立ちます。
- **時間**: レポートで使用するデータの期間を選択します。期間はレポ

ートの種類(日、週、月)によって異なります。

- **プロジェクト**：レポートを作成するプロジェクトを選択します。グループおよび個別のプロジェクト単位で選択できます。**検索**フィールドを使用して、目的のプロジェクトを検索できます。
- **レポートフォーマット保存**：現在入力したレポートの設定をテンプレートとして保存できます。今後、新しい報告書を作成する際**タイプ**一覧で読み込めます。

設定を完了したら、**レポート生成**ボタンを選択します。



選択したレポートの様式に従って照会期間を設定する方法は次のとおりです。

- **日刊報告書**: 開始と終了の日付、時間を選択できます。
- **週間報告書**: 開始日のみを選択すると、終了日を計算して自動的に選択します。

- **月間報告書**: 開始日のみを選択すると、終了日を計算して自動的に選択します。
  - 開始日が1日であれば、その月の月末までに自動的に選択されます。
  - 開始日が1日でなければ、来月の前日までに自動選択されます。

例1、開始日が**1月1日**であれば、**1月1日00時00分00秒**から**1月31日23時59分59秒**を自動選択します。

例2、開始日が**1月10日**であれば、**1月10日00時00分00秒**から**2月9日23時59分59秒**を自動選択します。

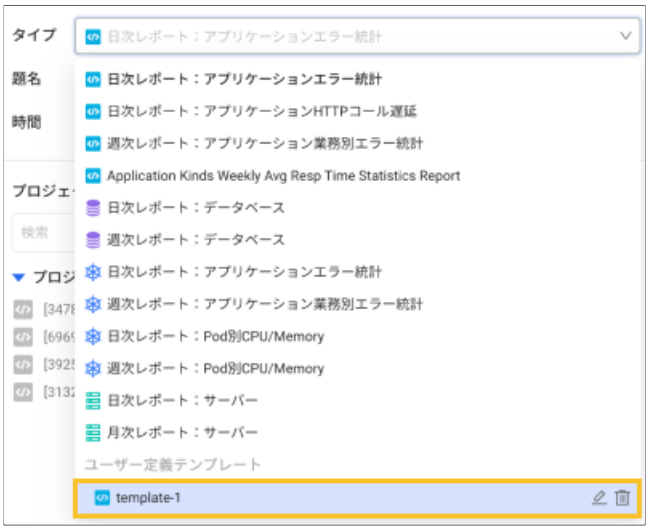
- **時間**オプションの利用方法の詳細については、次の文書を参照してください。
- 保存したレポート様式を読み込み方法の詳細は次の文書を参照してください。

レポートフォームの読み込み

新しいレポートを作成するとき**レポートフォーマット保存**機能を使用して保存した様式は、その後のレポート作成時に**タイプ**項目で確認できます。

業務単位でプロジェクトを頻繁に使用するユーザーは、アクティブに活用できます。

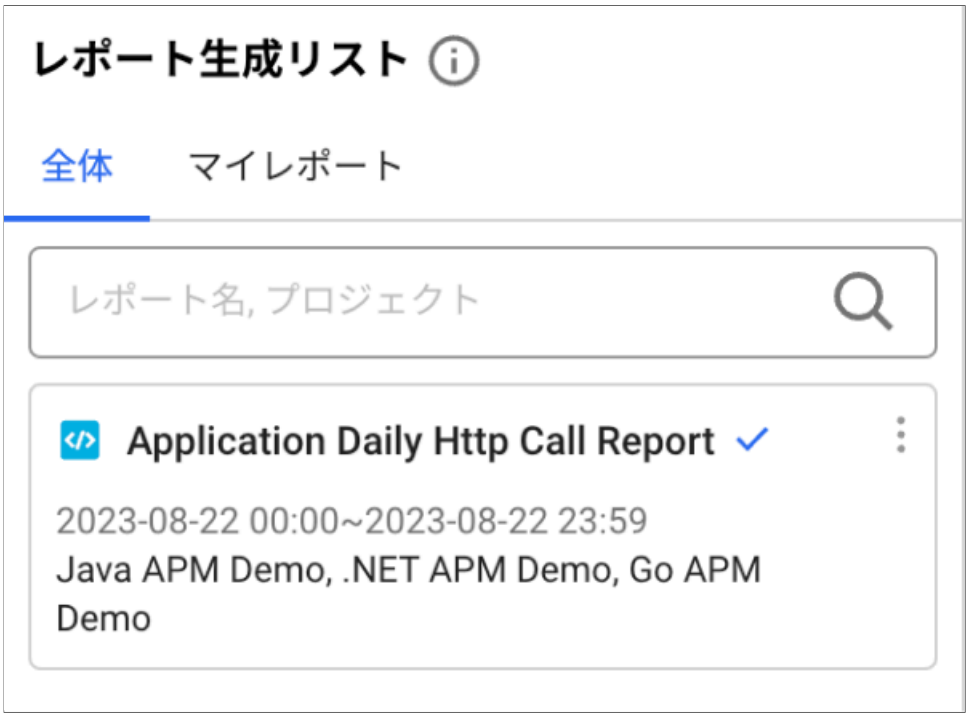
- ：保存したレポートフォームを修正できます。
- ：保存したレポートフォームを削除できます。



レポート一覧を確認する

レポート生成リストで以前に作成したレポートと現在作業中のレポートを確認できます。

- **全体**タブでは、ユーザー本人が作成したレポートの他にも、同じ権限を持つユーザーが作成したレポートを一緒に確認できます。
- **マイレポート**タブでは、ユーザー本人が作成したレポートのみ確認できます。
- レポート項目の右上に ⋮ ボタンを選択すると、そのレポートをコピーまたは削除できる機能を利用できます。



- エラーなく作成されたレポートは✓アイコンで表示され、エラーが含まれている場合は⚠アイコンで表示されます。⚠アイコンを選択すると、どのプロジェクトでエラーが発生したかを確認できます。
- 作成したレポートは最大100個まで7日間保管できます。

表示と共有

- **画面照会**

  レポート生成リスト一覧で作成したレポートを選択すると、レポートの詳細を確認できます。

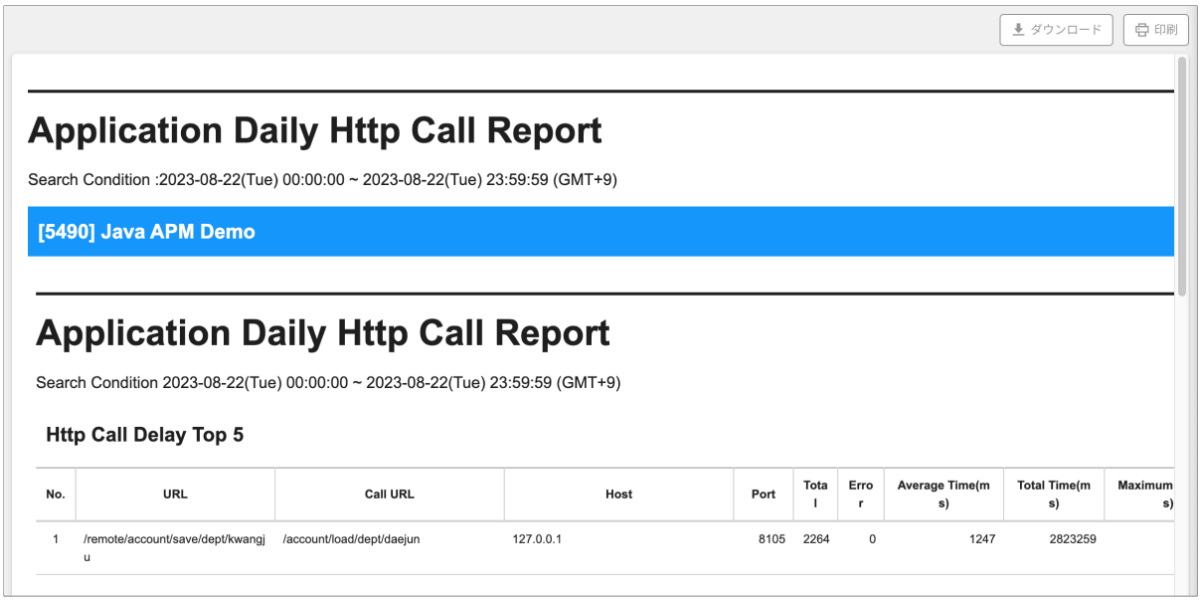
- **ダウンロード**

  現在照会中のレポートをhtml形式のファイルでダウンロードするには**ダウンロード**ボタンを選択します。モニタリングの管理

権限のない人にレポートを共有できます。

- **印刷およびPDF保存**

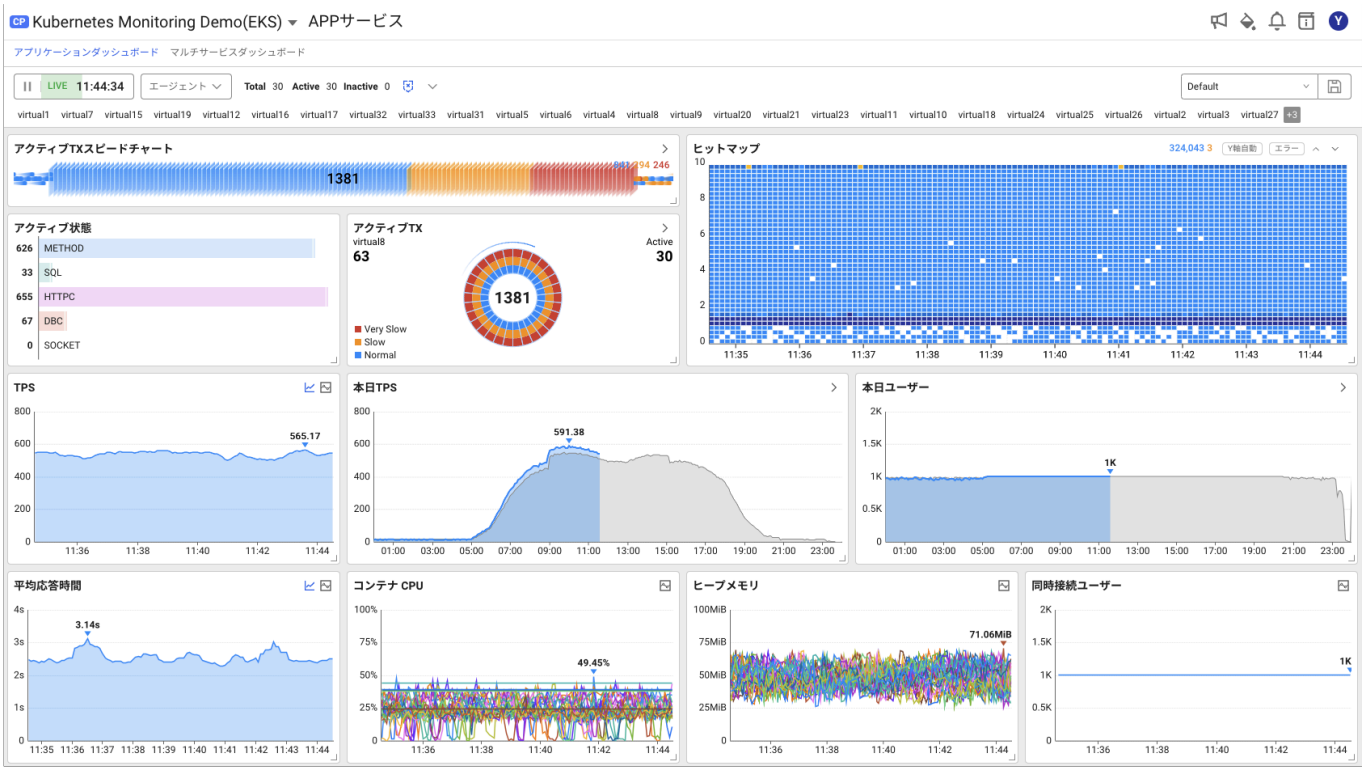
  レポートをプリンターで印刷するか、PDF形式のファイルとして保存するには**印刷**ボタンを選択します。



アプリケーションサービスダッシュボード

コンテナ化されて実行中のアプリケーションの状態をリアルタイムで確認できるダッシュボードです。 各ウィジェットの正常のデータ出力のためにはKubernetes環境にノードエージェントをインストールする必要があります。また、コンテナ化されたアプリケーションにもWhaTapエージェントをインストールする必要があります。

- Java, Node.js, Python, PHPで開発したアプリケーション向けのWhaTapエージェントのインストールをサポートします。
- ダッシュボードメニューまたはアプリケーションメニュー下位のダッシュボードメニューからアクセスできます。



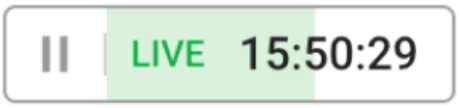
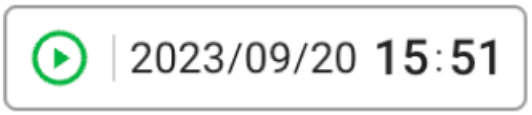
アプリケーションダッシュボードでは、アプリケーションの主な指標をエージェントを通じて収集します。また、チャートを構成してリアルタイムでモニタリングし、過去のデータを表示することができます。 ダッシュボードに配置されたウィジェットを通じて、アプリケーションの問題を素早く特定して調査することができます。

CPU、Memoryを除く一般的なチャートの場合、安定したデータは青色の系列で表現され、問題として識別される要素は赤色の系列で表示されるため、利用状況を簡単に認識できます。

アプリケーションダッシュボード

過去のデータ参照

アプリケーションダッシュボードは主にリアルタイムモニタリング機能を提供しますが、過去のデータも照会できます。

リアルタイムモード	非リアルタイムモード
	

1. 画面左上の時間選択から II ボタンを選択します。
2. 日付と時間領域をクリックして、目的の時間帯を設定してください。



3. **適用**ボタンを選択します。

ユーザーが設定した時間に基づいて、ダッシュボードに配置したウィジェットのデータを更新します。 リアルタイムモードに切り替えるには▶ボタンを選択します。

エージェントの確認

エージェント接続状態を確認する



画面の左上のタイムピッカーの右側には、プロジェクトに関連したエージェントの状態を確認できる情報が表示されます。 これにより、アプリケーションサーバーの動作有無がすぐに確認できます。

- Total: プロジェクトに関連付けられているすべてのエージェントの数
- Active: 活性化されたエージェントの数
- Inactive: 非活性化状態のエージェント数
- : 非活性化されたエージェントを表示または非表示します。

エージェント別モニタリング

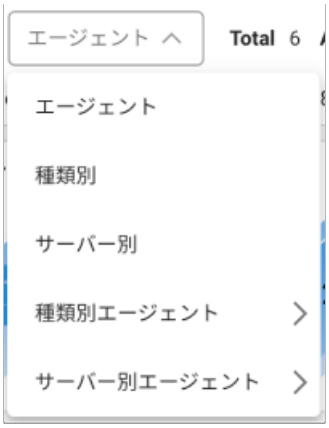


基本的に、ダッシュボードにはすべてのエージェントから収集した指標をチャートとして表示しますが、エージェントを選択してデータを表示することもできます。 タイムピッカーの下にあるエージェントを1つ以上選択します。 選択したエージェントの指標でダッシュボードに配置されたウィジェットのデータを更新します。

> エージェントを1つ以上選択した状態で再度すべてのエージェントを選択するには、選択を解除するか、Totalを選択します。

> プロジェクトに接続されたエージェントの数が多い場合は、エージェント名を短く設定することをお勧めします。 エージェント名の設定については、次の文書を参照してください。

分類別エージェントモニタリング



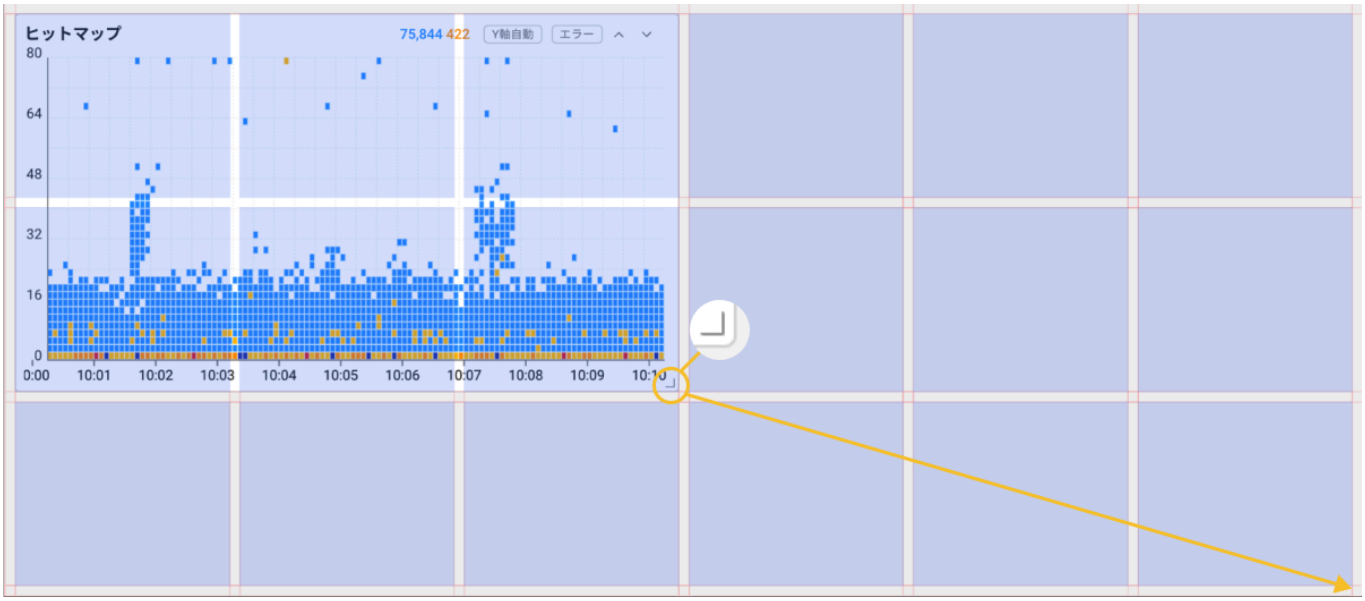
エージェント設定で分類されたグループ単位でエージェントを選択してモニタリングできます。

- **エージェント**: 個別エージェントまたはすべてを選択できます。
- **種類別**: エージェント設定で `whatap.okind` に分類されたグループ単位でエージェントをモニタリングします。
- **サーバー別**: エージェント設定で `whatap.onode` に分類されたグループ単位でエージェントをモニタリングします。
- **種類別エージェント**: エージェント設定で `whatap.okind` に分類されたエージェントの一覧のみをモニタリングします。
- **サーバー別エージェント**: エージェント設定で `whatap.onode` に分類されたエージェントの一覧のみをモニタリングします。

ダッシュボードウィジェットの編集

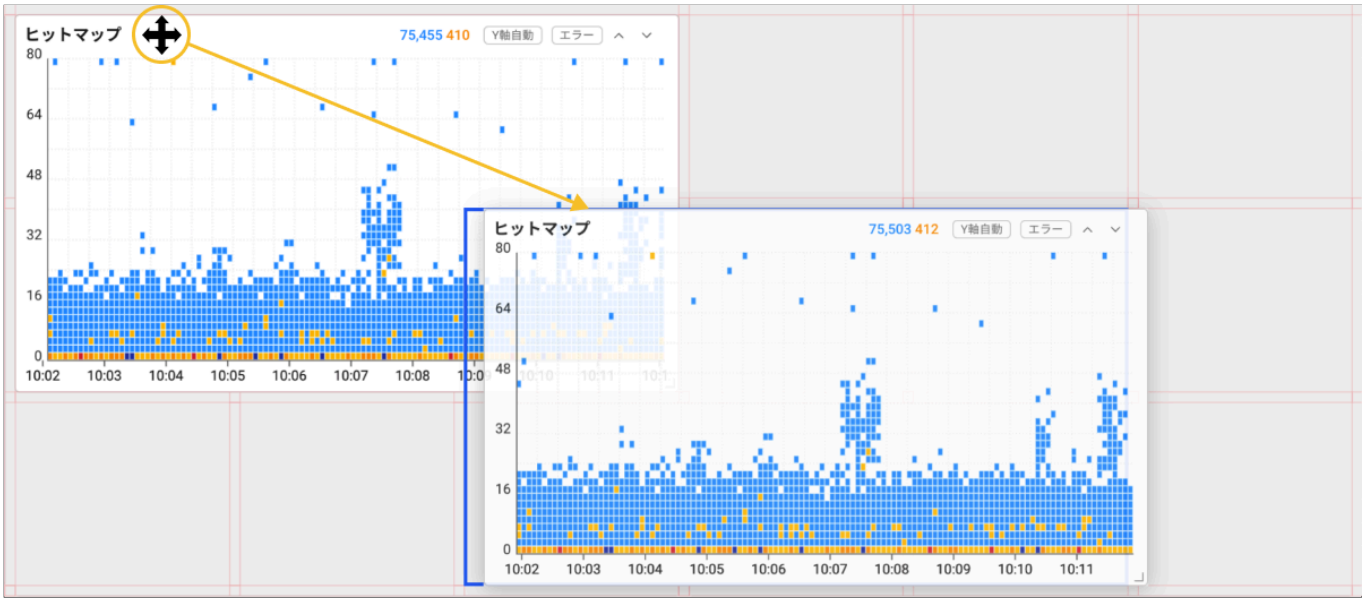
ダッシュボードに配置したウィジェットは、ユーザーが希望するサイズに調整でき、任意の位置に配置できます。 るウィジェットは削除したり、再度追加したりすることもできます。

ウィジェットのサイズ変更



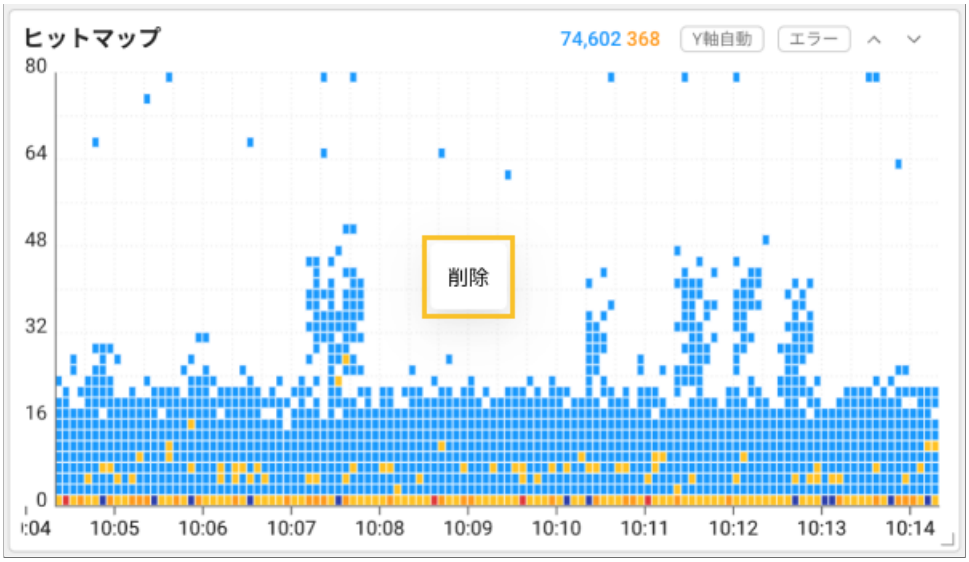
ウィジェットの右下にある⌟をマウスでクリックした状態で希望するサイズにドラッグします。 幅と高さの比率が均一なグリッドが表示され、グリッド単位でウィジェットのサイズを調整できます。

ウィジェットの移動



ウィジェットの上部にマウスカーソルを位置すると、カーソルの形が✢の形に変更されます。 この時、マウスの左ボタンをクリックした状態で希望する位置にドラッグすると、ウィジェットを移動することができます。

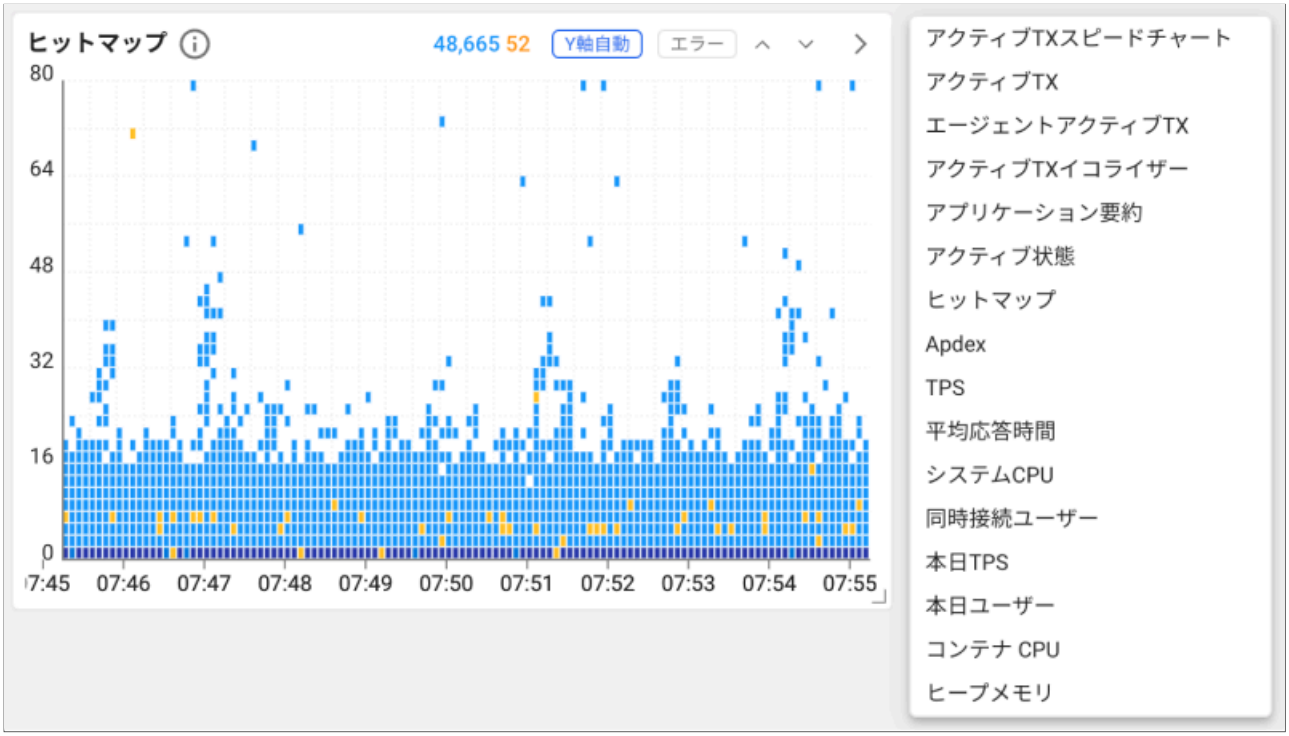
ウィジェットを削除する



削除するウィジェットで右クリックします。 **削除**ボタンをクリックすると、ウィジェットがダッシュボードから削除

されます。

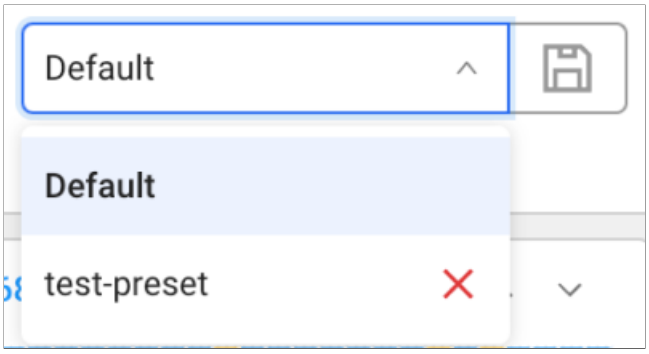
ウィジェットを追加する



ダッシュボードから空白エリアにマウスカーソルを移動し、マウスの右ボタンをクリックします。 ポップアップメニューから追加するウィジェットを選択します。 希望する場所にウィジェットを配置してサイズを変更します。

> ⓘ 現在追加できるウィジェットは固定されていますが、今後のアップデートでウィジェットサポートを追加していく計画です。

プリセット



ダッシュボードからユーザーが設定したウィジェットの設定とレイアウト状態を保存して読み込むことができます。基本プリセット(**デフォルト**)は変更できず、希望する形でウィジェットのサイズを変更、配置して新しいプリセットが作成できます。

新しいプリセットを作成する

1. ダッシュボードから、希望する形にウィジェットを配置してみてください。 サイズを変更して、頻繁に確認するウィジェットのみを配置することもできます。
2. 画面の右上に保存ボタンを選択します。
3. 新しいプリセット名を入力します。



エージェント選択履歴を一緒に保存するには、**エージェント選択履歴を含む**を選択します。

4. **保存**ボタンを選択します。

プリセット一覧から新しく保存したプリセットを確認できます。

- 新しく作成したプリセットに変更が加えられた場合は、再度プリセットを保存してください。 ボタンを選択し、同じ名前でプリセットを保存します。 既存のプリセットに変更を上書きします。
- ダッシュボードの変更を保存せずに別のメニューに移動すると、変更は保存されません。
- プリセットはユーザーアカウントごとに保存され、他のユーザーと共有されません。 より良い機能が提供できるように更新していく予定です。

プリセット削除

使用していないプリセットは、プリセット一覧から削除できます。 プリセット一覧から削除したい項目の右側に✕ボタンをクリックします。

ウィジェットオプションを利用する

ウィジェットの上に表示されているアイコンボタンの機能は次のとおりです。

- ：主要ウィジェットに関する機能と情報が確認できます。
- **マージ/個別に表示** : ウィジェット項目のエージェント データを個別またはマージしてグラフで表示します。
- **詳細** : ウィジェット項目のデータをエージェント別に区分して照会できるポップアップが表示されます。

ウィジェットによって使用可能なオプションは異なる場合があります。

ウィジェットの詳細

この文書では、**アプリケーションダッシュボード**に配置されたウィジェットについて説明します。 **アプリケーションダッシュボード**を使用したトランザクション、ユーザー、サービス及びリソース分析の方法についての詳細は、次の文書を参照してください。

- アプリケーションダッシュボードを確認する
- ヒットマップTX
- DB接続遅延とコネクションプール

エージェント関連ウィジェット

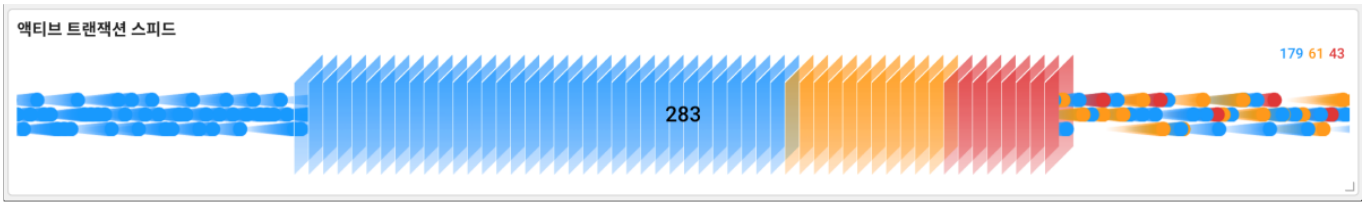
アプリケーションにインストールしたエージェントを使用して、アプリケーションの情報を確認できます。

トランザクション関連ウィジェット

トランザクションを通じて確認できるトラブルの事象は、まず応答時間で把握できます。 また、進行中のトランザクションが終了しない場合は、これもトラブルとして認識すべきです。 WhaTap Monitoringでは、進行中のトランザクションの遂行時間に基づいて状態を表示します。 青色(Normal)は応答時間が正常なトランザクション、オレンジ色(Slow)は応答時間が8秒までの遅いトランザクション、赤色(Very slow)は、応答時間が8秒を超えた非常に遅いトランザクションを意味します。 これにより、ユーザーはトラブル状況が直感的にすぐ認識できるようになります。

> Javaプラットフォームでは、**連携プロジェクト管理**メニューでデータベースプロジェクトを追加すると、次のウィジェットで連携したプロジェクトの性能指標を照会できます。
>
> - **アクティブTXスピードチャート**, **アクティブTX**: 現在進行中のトランザクションと連携されたデータベースのSQL性能を照会できます。
> - **ヒットマップ**: 過去の時点でのトランザクションに連携されたデータベースの性能メトリクスを照会できます。
>
> 連携プロジェクトの追加と分析の詳細については、次の文書を参照してください。

アクティブTXスピードチャート



現在進行中のトランザクション(中央領域)と、終了したトランザクション(右領域)の状態を確認できます。 左から右に流れる水玉模様のアイコンは、トランザクションを意味します。 チャート領域をクリックすると、進行中のトランザクション一覧を確認できる画面が表示されます。 トランザクション一覧のいずれかを選択すると、詳細なトランザクション情報とSQLクエリ、コールスタック情報を確認できます。

- 左領域

リクエストTXの数を意味し、TXの数に比例してより多く、より速く表示します。

- 中央領域

  アクティブTXの状態を速度で色分けし、右上に個数を表示します。 チャートの数字は、アクティブTX全体の数です。 チャートの色の意味は次のとおりです。

  - 青色(Normal)： 0秒~3秒
  - オレンジ色(Slow)：3秒~8秒
  - 赤色(Very Slow)：8秒以上

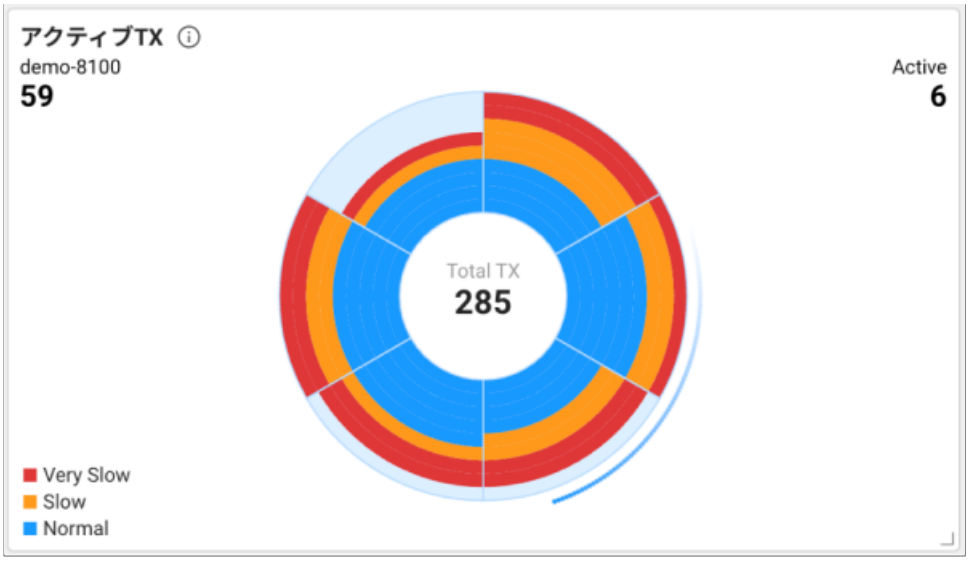
- 右領域

  アプリケーションの性能指標(Application Performance Index, Apdex)を意味します。 ウェブアプリケーションの顧客満足度を測定した指標であり、0と1の間の数です。 水滴が飛んでいく速度はApdexのトランザクションの合計数に比例し、水滴の色の意味は次のとおりです。

  - 青色:Apdex満足(Satisfied)数
  - オレンジ色:Apdex許容(Tolerating)数
  - 赤色:Apdex不満(Frustrated)数= `APDEXトランザクション総数` - (`Apdex満足数` + `Apdex許容数`)

> ⓘ アプリケーションの性能指標(Application Performance Index、Apdex)についての詳細は次のリンクを参照してください。

アクティブTX



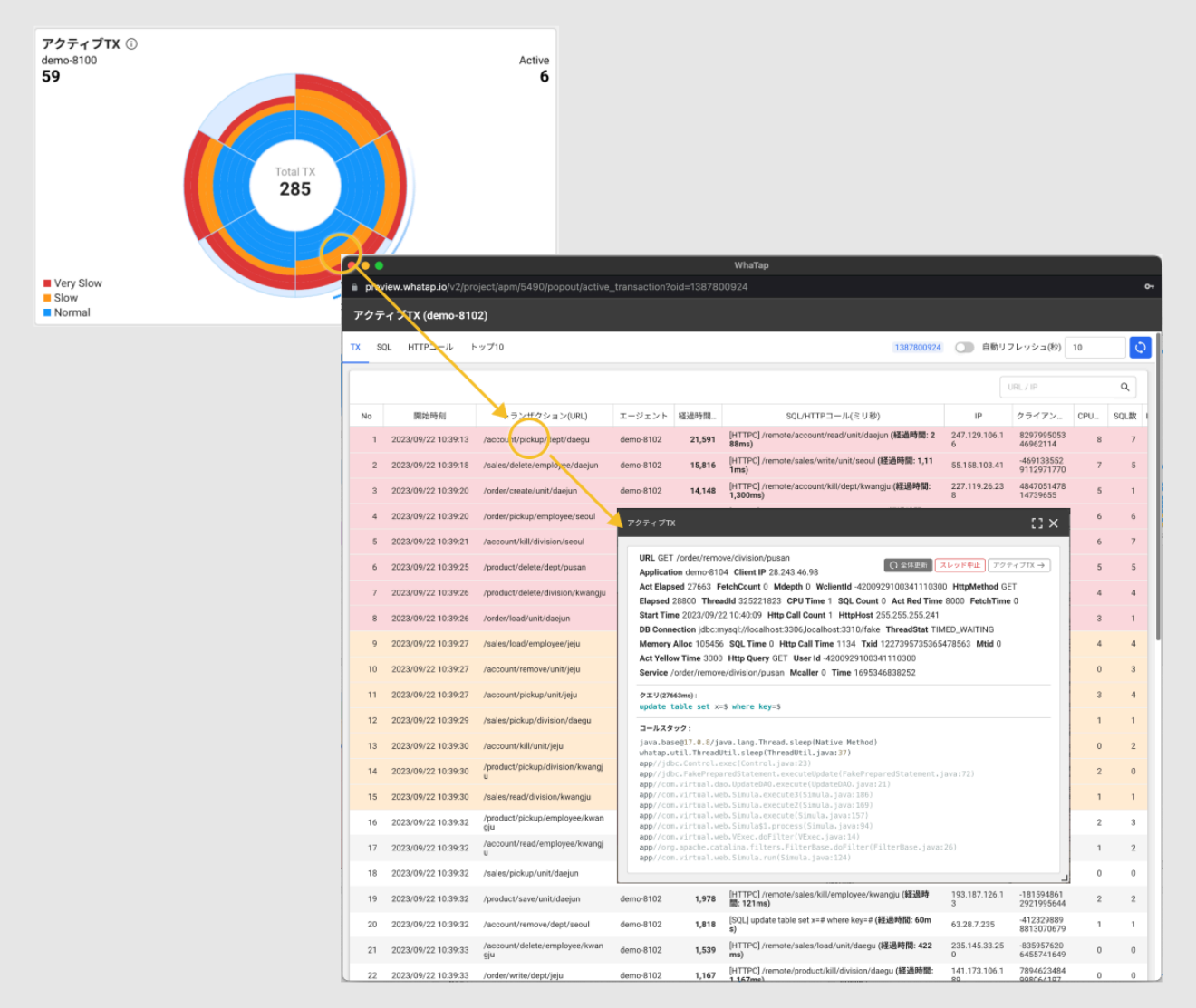
アクティブTXウィジェットでは、進行中のトランザクションを**アークイコライザー**チャートとして表示します。 遅延の発生状況をエージェントごとに確認できます。 エージェントの数程にアークが分割されます。 5秒ごとに現在サーバーで処理中のリクエストの数を表示するため、そのリクエストがそれぞれどのくらいの時間で処理されているのかが把握できます。 5秒間隔で検知されたリクエストは、遅延状況が直感的に把握できるよう、次のように色で分けられます。

- 赤色(Very Slow)：8秒以上
- オレンジ色(Slow)：3秒~8秒
- 青色(Normal)： 0秒~3秒

青色が多い状況は問題ありません。 これらの一部がオレンジ色や赤色に変わっていくかを監視する必要があります。 トラブルが発生すると、進行中のトランザクション数が増加し、赤色の割合が高くなります。 一方で、レスポンスが非常に速いシステムの場合、処理されるトランザクション数(TPS)が多くても、進行中のトランザクション数は少ないことがあります。

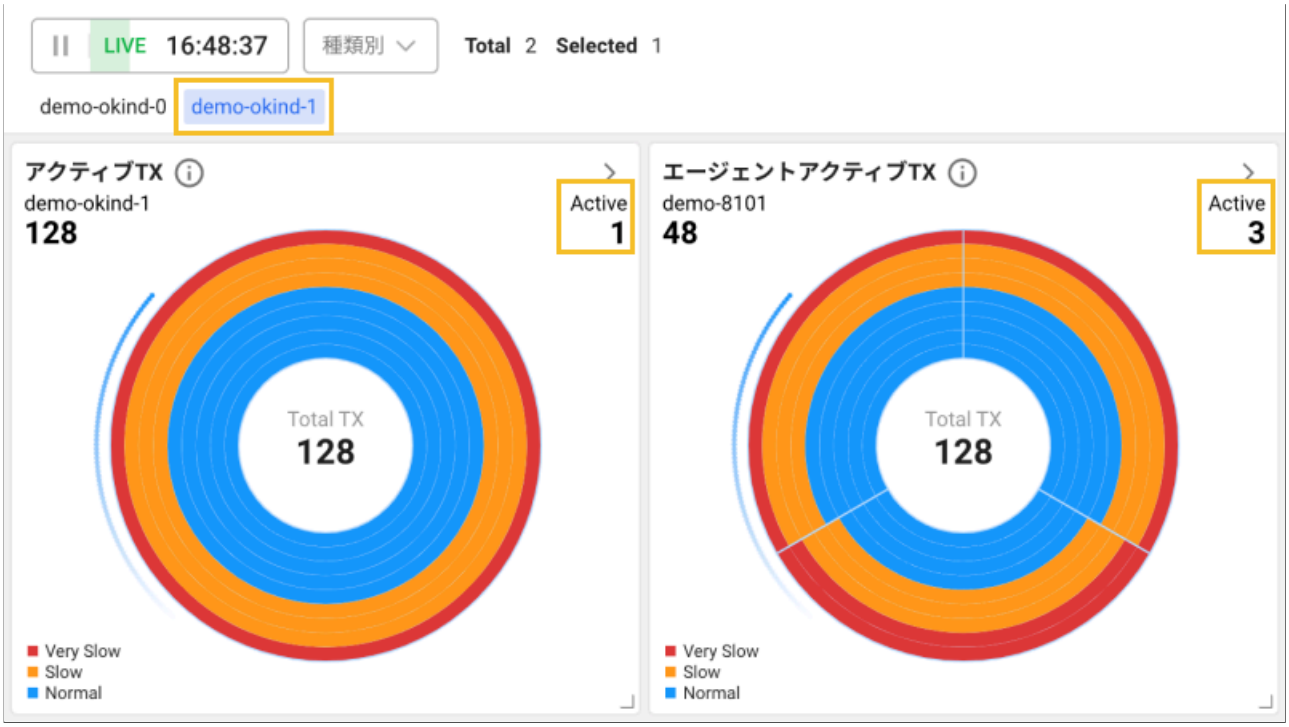
- 左上には、進行中のトランザクションが最も多いエージェント名を表示します。
- 真ん中の数字は、エージェント全体のアクティブTXの合計数です。
- アクティブTXを囲む2本のバーは、処理量(TPS)に基づいて3段階の速度でグラフの周りを回転します。
- 分割されたアーク領域を選択すると、該当エージェントが監視するアプリケーションから進行中のトランザクション一覧がポップアップ画面に表示されます。 詳細なトランザクション情報を確認するには、トランザクション一覧

から任意の項目を1つ選択します。



ⓘ
- トランザクション一覧で選択した項目が既に終了しているトランザクションの場合は、**トレース表示**ボタンを選択します。 **トランザクション情報**画面が表示されます。 **トランザクション情報**画面の詳細については次の文書を参照してください。
- トランザクションの詳細については、次の文書を参照してください。

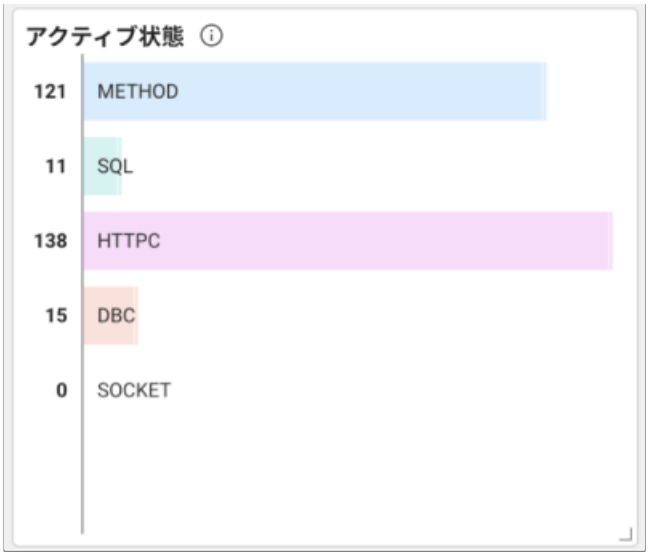
エージェントアクティブTX



アクティブTXウィジェットは、エージェント選択オプションで選択した基準に従って、マージされたデータをチャートに表示します。 一方、**エージェントアクティブTX**ウィジェットでは選択したエージェント分類の基準に含まれたエージェントの状況を確認できます。

ダッシュボードでマウスの右ボタンをクリックし、ウィジェット一覧で**エージェントアクティブTX**ウィジェットを選択します。

アクティブ状態



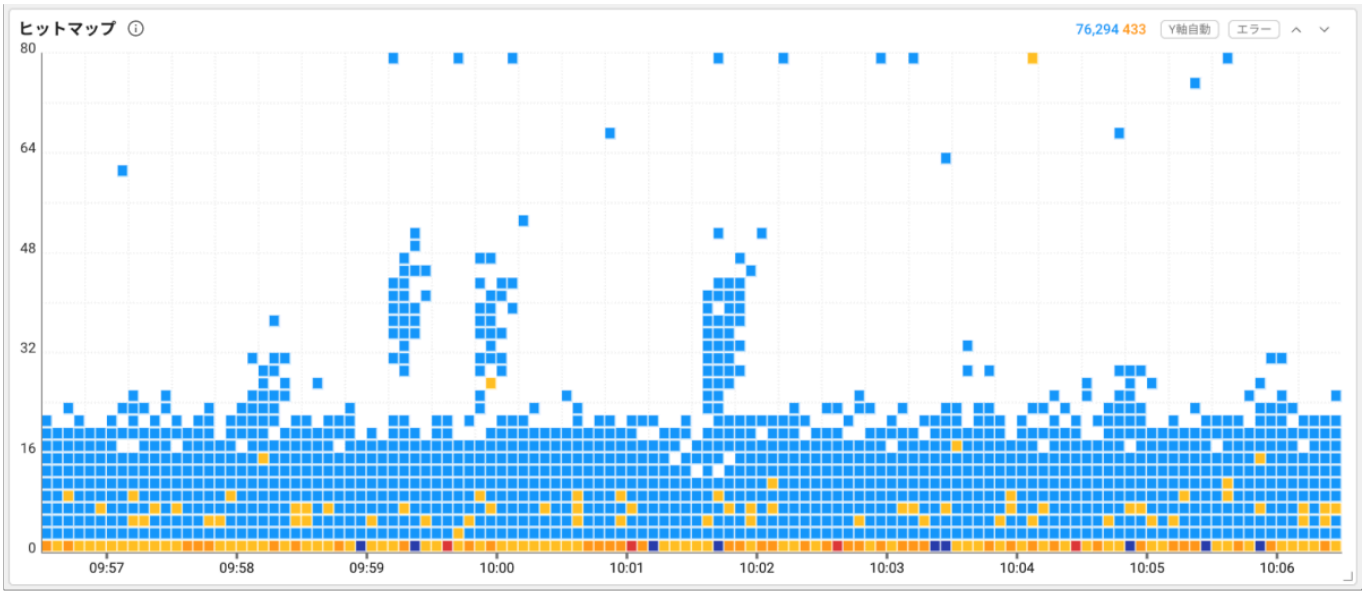
プロジェクト内のすべてのエージェントは、5秒ごとにアクティブTX数を収集すると同時に、トランザクションのステータスを収集します。 トランザクションのステータスは**METHOD**、**SQL**、**HTTPC**、**DBC**、**SOCKET**に分けられます。

- **METHOD**：メソッドを実行している状態です。 一般的な状況として青色で表示されます。
- **SQL**：SQLを実行している状態です。 **SQL**数の増加が見られた場合は、データベースサーバーとの接続を確認してください。
- **HTTPC**：外部APIを呼び出し中の状態です。 **HTTPC** 数が増加した場合は、外部と接続されたサーバーの応答が正常ではない状況として解釈できます。
- **DBC**：トランザクションがConnection Poolから新しいConnectionを獲得(get)しようと待機している状態です。 DB Connection Poolの数が不足すると、新しい接続リクエストが発生するたびに遅延が発生し、性能トラブルの原因となります。 この場合、**DBC**の数値が増加します。

  > ⓘ DB接続遅延に関する詳細については次の文書を参照してください。

- **SOCKET**：外部システムとのTCP接続試行を意味します。 **SOCKET** 数値が持続的に増加する場合は、外部システムとの接続ができないため障害が発生している可能性があります。

ヒットマップ



ヒットマップチャートは、終了したトランザクションの応答時間を分布図で表現します。 横軸はトランザクションの終了時間です。 縦軸は実行時間です。 応答時間が伸びたトランザクションは、分布図の上段に位置します。 **ヒットマップ**チャートの中の領域をドラッグすると、詳細なトランザクション情報が確認できる**トレース分析**画面が表示されます。 **トレース分析**の詳細は、次の文書を参照してください。

- 通常のトランザクションは水色→青色→紺色の順に表現されます。
- エラートランザクションは黄色→オレンジ色→赤色の順に表現されます。
- 画面の右上の数字はチャート内のトランザクション数/エラー数を意味します。
- **Y軸自動**ボタンを選択すると、実行時間を示すY軸を自動調整します。
- **エラー**ボタンを選択すると、エラートランザクションのみ表示します。
- ∧または∨ボタンを選択すると、5秒~80秒でチャートを拡大／縮小することができます。

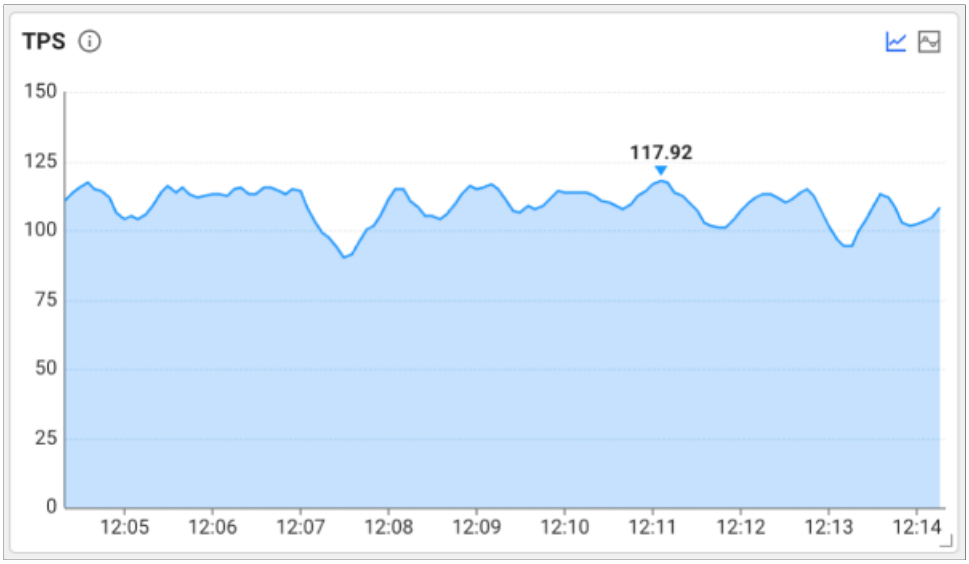
ⓘ
- エラーが発生せず、実行時間が500ms以下のトランザクションの詳細は、URLあたり5分に1回のみ収集されます。
- TPSや平均応答時間などの統計情報には影響しません。
- エージェント設定により、ポリシーを設定できます。 詳細については、次の文書を参照してください。
- ヒットマップウィジェットを使用してトランザクションを分析するプロセスについては、次の文書を参照

してください。

サービス関連ウィジェット

サービス関連ウィジェットは、アプリケーションのパフォーマンスチューニングの指標として使用できます。

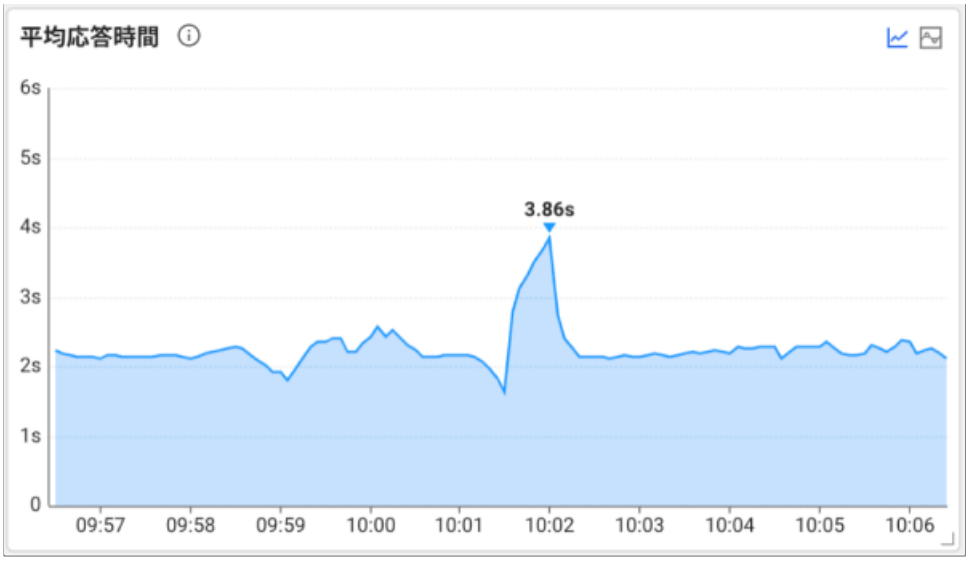
TPS



Tansaction Per Second(**TPS**)は毎秒処理されるトランザクション数を意味し、サービスパフォーマンス指標の基準となります。 5秒間に処理されたトランザクション数を1秒あたりの数として換算してチャートで表現します。 直近10分間のTPSを表示します。

TPS、応答時間、CPU 使用率分析の詳細については次の文書を参照してください。

平均応答時間

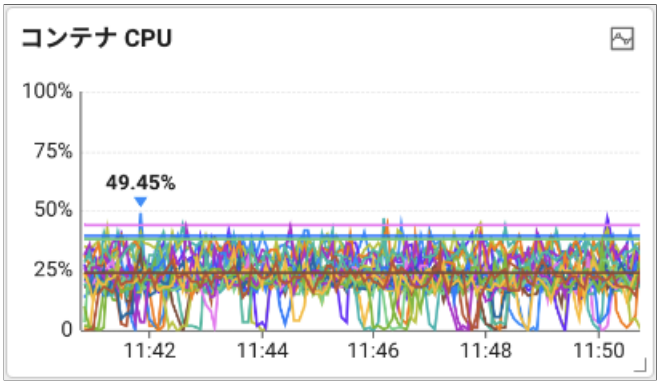


アプリケーションサーバーがユーザーにリクエスト結果を返すのに掛かる時間をリアルタイムでモニタリングします。**平均応答時間**は、チューニング指標として意味があります。 WhaTapのモニタリングサービスは、5秒間隔でトランザクションの平均応答時間を計算します。

> TPS、応答時間、CPU 使用率分析の詳細については次の文書を参照してください。

リソース関連ウィジェット

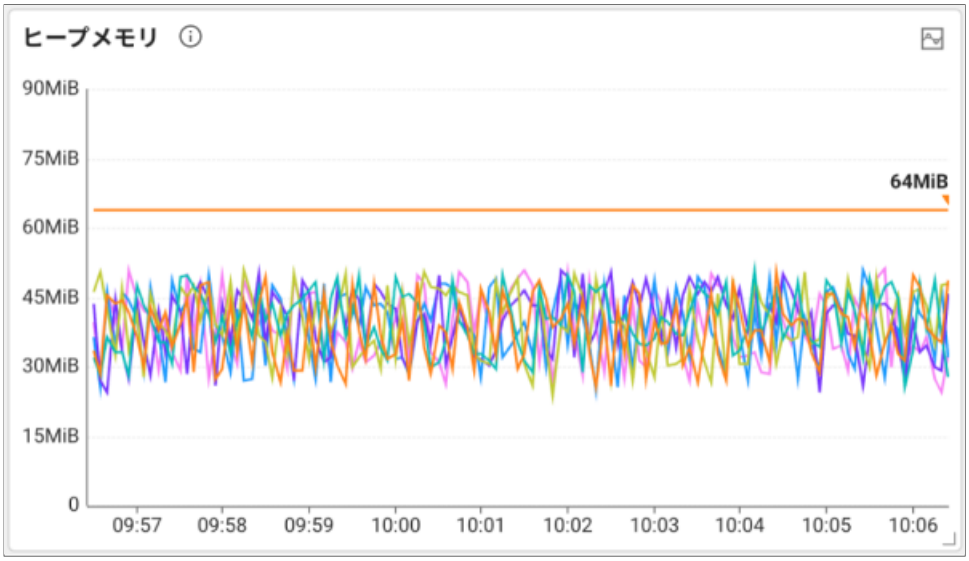
コンテナCPU



コンテナCPU使用量です。 リアルタイムでCPU使用量の変化の推移を把握できます。

> TPS、応答時間、CPU 使用率分析の詳細については次の文書を参照してください。

ヒープメモリ



サーバー毎に使用できる最大メモリと現在のメモリを表示して、負荷がかっかているサーバーが容易に把握できます。 時間の経過に伴うメモリ使用量の変化をリアルタイムで確認できます。

メモリ折れ線チャートは通常、波線を続けます。 アプリケーションサーバーがリクエストを処理するためにメモリを使用するとメモリ折れ線チャートは増加します。 GC(Garbage Collection)によってメモリをクリーンアップすると減少します。 **ヒープメモリ**指標は平均値を使って表記します。

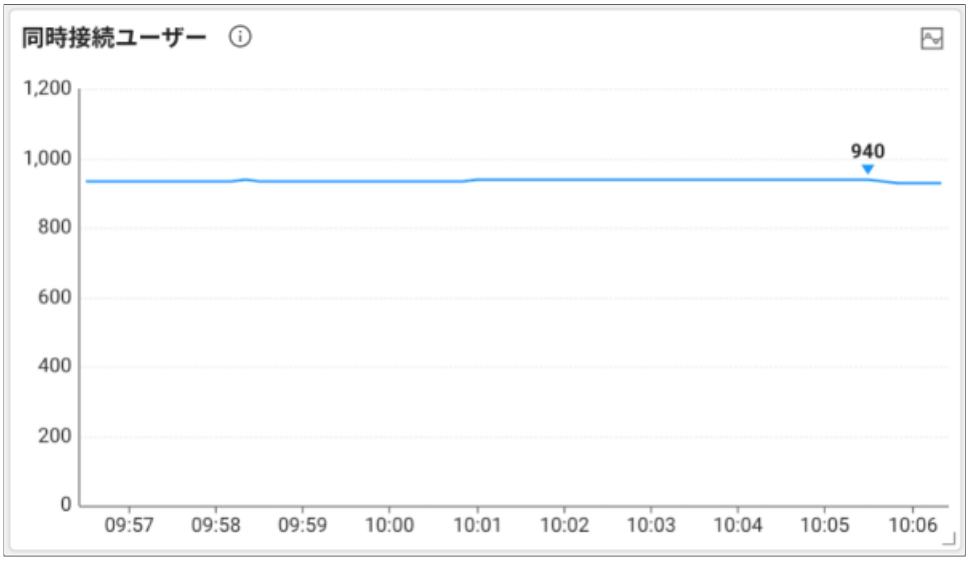
> - JavaとNode.jsでは、ヒープメモリの合計量と使用量が表示されます。
> - PHPおよびPython, .Netでは、プロセスのメモリの使用量が表示されます。

> ヒープメモリおよびヒープメモリチャートの分析については、次のリンクを参照してください。
> - 学習する: ヒープメモリ メトリクスおよびメトリクス通知設定
> - Javaヒープメモリーチャートの分析：Ch.1ヒープチャート観察

- JAVAヒープメモリーチャートの分析：Ch.2 メモリリークとヒープダンプ分析

ユーザー関連ウィジェット

同時接続ユーザー



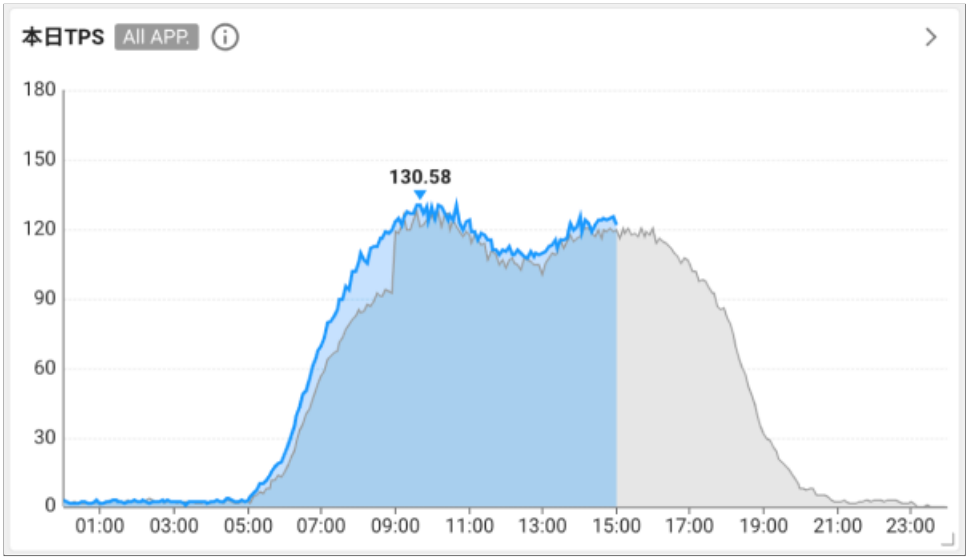
リアルタイムブラウザのユーザー数を表示します。 10秒ごとに、直近5分以内にトランザクションを発生させたユーザーを重複なしでカウントして表示します。 ユーザーブラウザのIP情報でカウントします。

- 通常、接続ユーザーはネットワークに接続しているユーザーを意味します。 ただし、ウェブシステムは未接続のネットワークを使用するため異なります。 接続されているという意味ではなく、最近のリクエストを送信したユーザーが測定の対象になる可能性があります。
- リアルタイム(同時接続)ユーザーとは、同じ時間帯にシステムを使用しているユーザーを意味します。 リアルタイム(同時接続)ユーザーを計算または測定する方式は多数あります。
- ユーザー数は、単に合計されるのではなく、** HyperLogLog**によって計算されます。 **HyperLogLog**は、非常に少ないメモリ内の要素の数を推定する確率的なデータ構造を意味します。
- ユーザー数の集計のためのエージェントオプションの設定方法は次の文書を参照してください。

1日基準比較ウィジェット

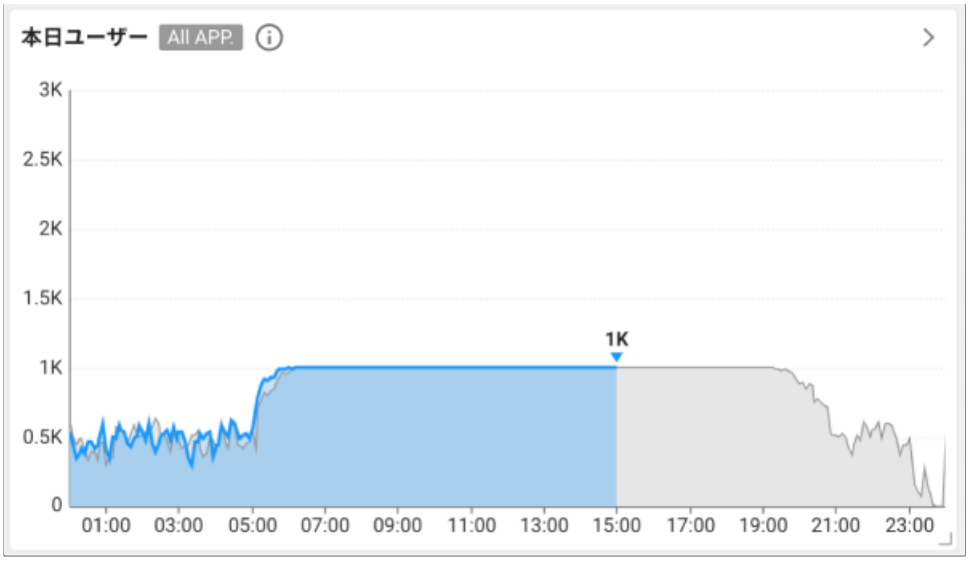
> All APPS. タグ付きウィジェットは、常にすべてのエージェントのデータを表示します。

本日TPS



1日のTPSを時間帯別に表示します。 昨日の同時間帯のTPS(グレー)と比較して確認できます。 > ボタンをクリックすると、**統計/レポート** > **日別APP現況**メニューに移動します。

本日ユーザー



一日あたりの**同時接続ユーザー**の数を時間帯別に表した指標です。 昨日の同時ユーザー数(グレー)と比較して確認できます。 > ボタンをクリックすると、**統計/レポート** > **日別APP現況**メニューに移動します。

リアルタイムアラートの確認

画面の右上に**リアルタイムアラート**ボタンをクリックすると、最近発生したイベントが確認できます。 トグルメニューをクリックして、ブラウザ通知をオン、オフできます。

> ⓘ 画面上段の固定メニュー領域にある基本項目については、次の文書を参照してください。

マルチサービスダッシュボード

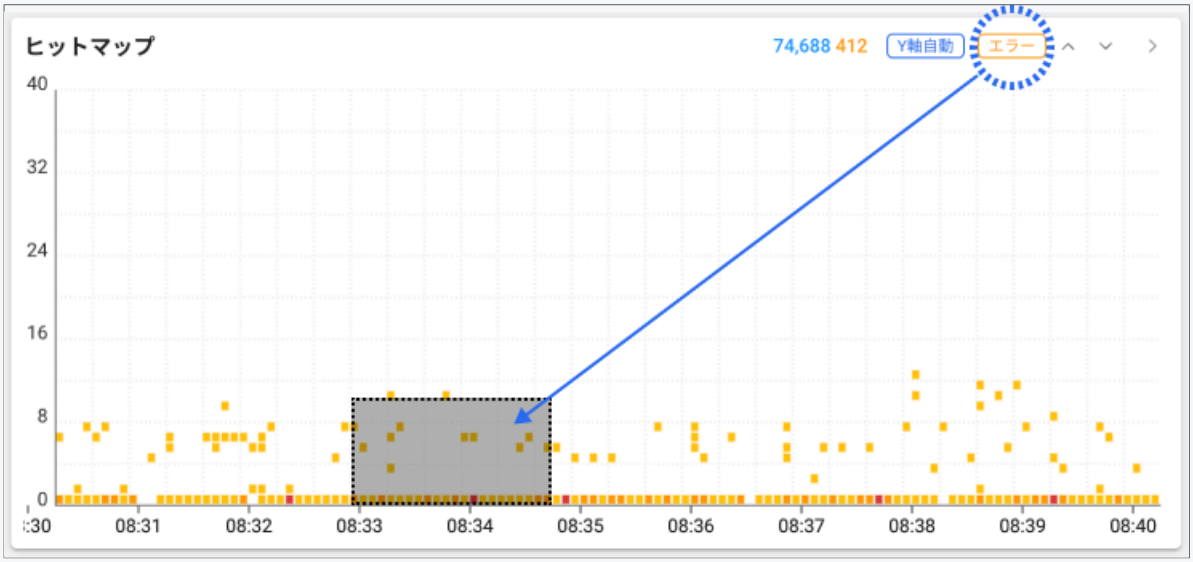
選択したアプリケーションを個々のカード形式で表示および比較できます。


WhaTap
K8s Namespace Project DEMO
APPサービス
全プロジェクト
Demo Project
Amazon CloudWatch ...
Browser Monitoring De...
ECS Monitoring Demo
Go APM Demo
Java APM Demo
K8s Namespace Proje...
プロジェクト メニュー
サイトマップ
ダッシュボード
コンテナマップ
APPサービス
APP TXマップ
アクティブTX
Flexボード
リソース
アプリケーション
分析
ログ
レポート
アラート通知
管理
利用内訳
すべて選択
virtual
virtual 2
virtual 3
アプリケーションダッシュボード
マルチサービスダッシュボード
virtual
virtual-172-20
58
Active
4
203
Very slow
Slow
Normal
TPS
89.73
応答時間
3.21s
virtual
virtual-172-19
62
Active
4
198
TPS
88.72
応答時間
2.88s
virtual
virtual-172-21
60
Active
5
246
TPS
111.35
応答時間
4.4s

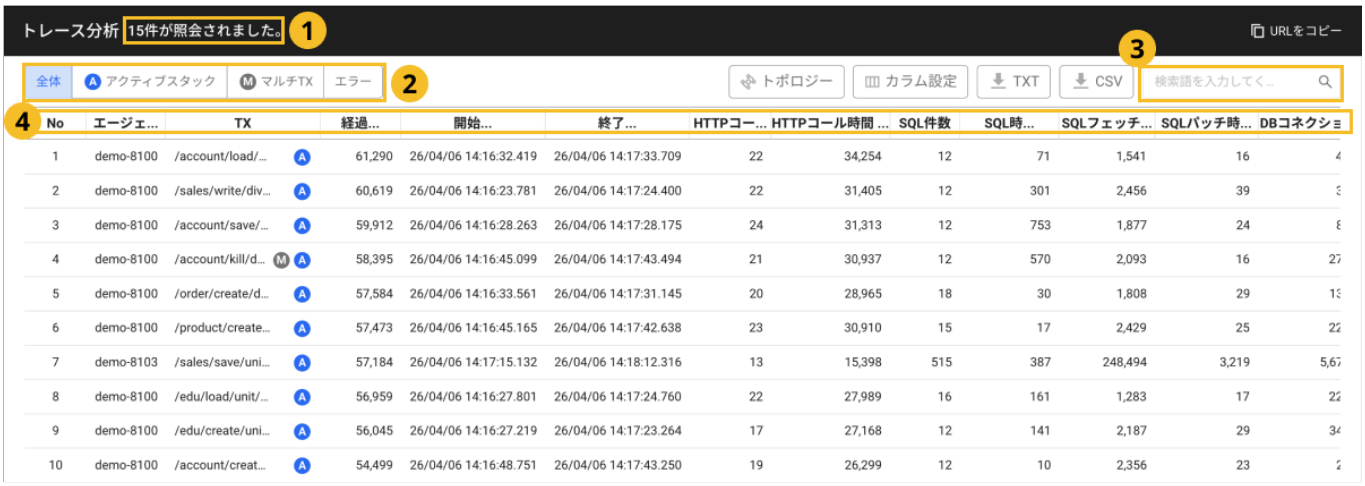
ヒットマップ - トレース分析

アプリケーションの**ヒットマップ**ウィジェットから**トレース分析**画面を開くと、トランザクション一覧と各トランザクションの下位ステップ情報が一緒に確認できます。 ヒットマップウィジェットで照会する領域をドラッグします。 選択した領域のトランザクション一覧と下位ステップ情報が**トレース分析**画面に表示されます。

エラーが発生したトランザクション情報のみを分類して確認するには、**アプリケーション**の**ヒットマップ**ウィジェットの右上にある**エラー**ボタンをクリックし、チャート領域をドラッグしてください。

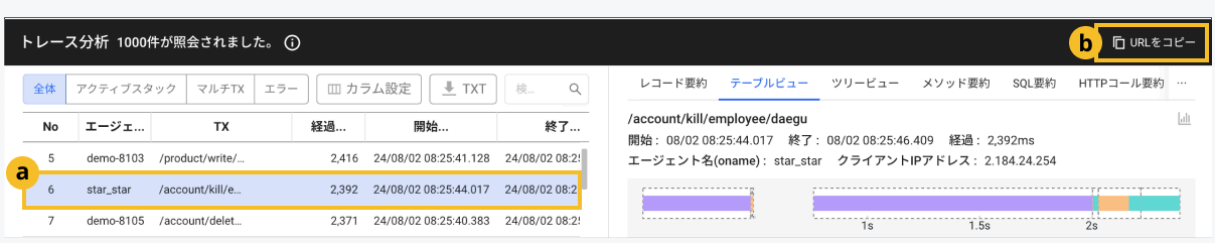


基本画面ガイド



- 1 **トランザクション照会件数**:ヒットマップウィジェットからドラッグした領域のトランザクション一覧の数を確認できます。 最大1000件まで照会できます。
- 2 **トランザクションの分類**：A **アクティブスタック**とM **マルチTX**、**エラー**項目を区分して一覧を確認できます。
- 3 **トランザクション検索**：トランザクション名、エージェント名、エラー メッセージに基づいてトランザクション一覧を検索できます。
- 4 **一覧ソート**：テーブルのヘッダーカラム項目をクリックすると、選択した項目に基づいてトランザクション一覧をソートできます。
- エラーが発生したトランザクション項目は、一覧に赤色で表示されます。

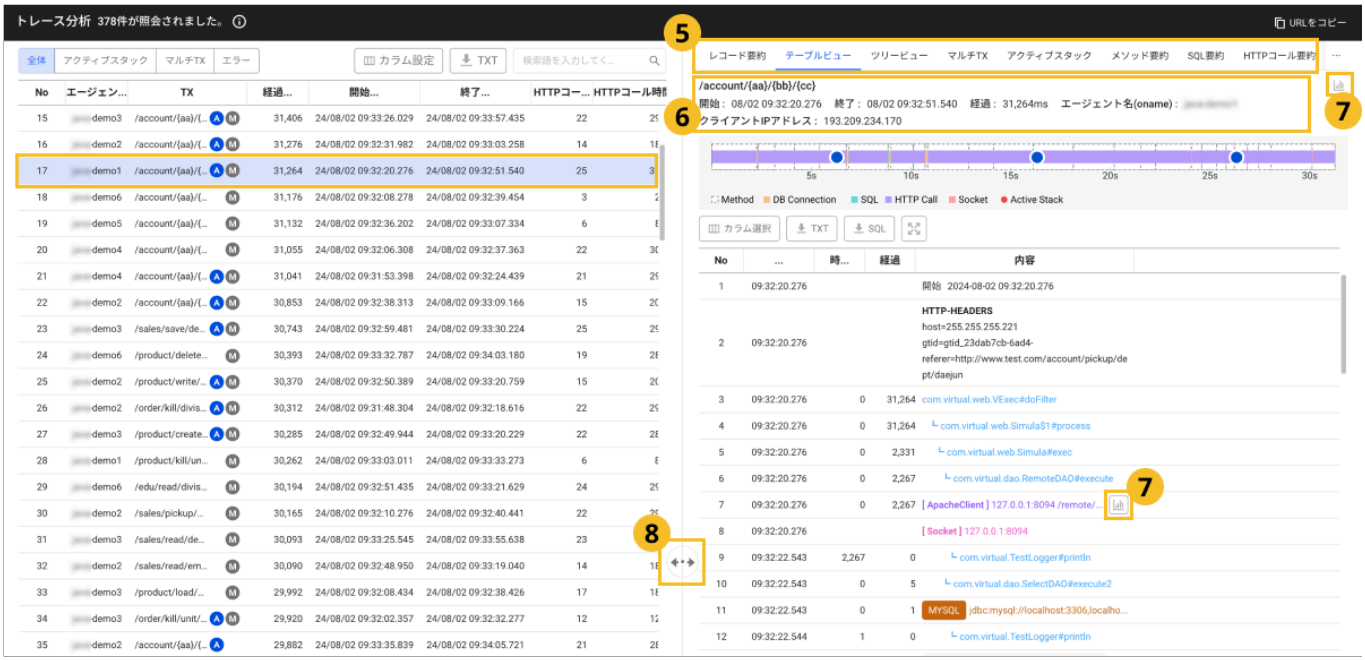
- テーブルでよく参照するカラムを前方に配置したり、カラムの追加および非表示などができます。 詳細は次の文書を参照してください。
- **ヒットマップ**ウィジェットから別の領域をドラッグすると、新たなトレースの分析画面を表示できます。既存のトレース分析画面と比較して確認できます。
- 表示されているトランザクション項目を他のユーザーに共有できます。 トランザクション項目を選択した状態(a)で、トレース分析画面b **URLをコピー** ボタンを選択します。 コピーしたURLアドレスを他のユーザーに共有します。 URLアドレスを受け取ったユーザーは、ブラウザアドレスバーにURLアドレスを入力すると、同じ項目をすばやく確認できます。



トレース分析

トランザクション一覧から詳細内容を確認する項目を選択します。 画面右側の選択したトランザクションに関しての詳細分析内容を表示します。 この状態で別のトランザクション項目を選択すると、選択したトランザクション項目の詳細分析内容に変更されます。

WhaTapは、トランザクションの性能を分析するために、名前やクライアント情報などの属性、トランザクションの処理性能、そして各区間別の詳細な実行履歴を収集して表示します。



- 5領域では、ステップ情報をさまざまな形式で確認できます。 希望する形式のタブを選択します。
- 6領域では、トランザクションに関する基本情報を確認できます。 左一覧のカラム項目と一致する情報です。

- 7 📊：該当URLまたは各実行区間の統計データの画面が表示されます。 統計データ画面のグラフチャートで、希望する時間をクリックすると**統計**または**TX検索**メニューに移動します。 選択した時間に基づいて統計データを照会することができます。
- 8 ↔左右にドラッグして、画面のサイズを調整できます。 より多くの情報を確認する際に便利な機能です。

> ⓘ
> - **統計**メニューの詳細は次の文書を参照してください。
> - **TX検索**メニューの詳細は、次の文書を参照してください。
> - ◀または▶ボタンをクリックすると、トランザクション一覧または詳細な分析画面をフルサイズに変更できます。

収集する**ステップ**(**Step**)の種類は次のとおりです。 ダイアグラムとテキストの色を区分して表示します。

- DB Connectionステップ START-END

  RDBへの接続のパフォーマンスが含まれます。 ステップ情報には、名前、レスポンス時間、エラーが含まれます。

- SQLステップ START-END

  JDBC SQLのパフォーマンスが含まれます。 ステップ情報には、接続情報、SQL文、エラーが含まれています。

- HTTP Callステップ START-END

  外部HTTPサービス呼び出しのパフォーマンスが含まれます。 ステップ情報には、url、host、port、レスポンス時間、エラーが含まれます。

- Messageステップ ADD

  トレースを収集する過程で、非構造化のすべての間隔の履歴を収集する際にメッセージステップを使用します。

- Socketステップ ADD

  Socketオープンを表現するステップです。

- Methodステップ START-END

  メソッド応答時間を追跡します。

- Active Stackステップ START-END

  アクティブスタックに関する情報を含みます。 別のスレッドが作成し、トレースに追加する方法で収集します。

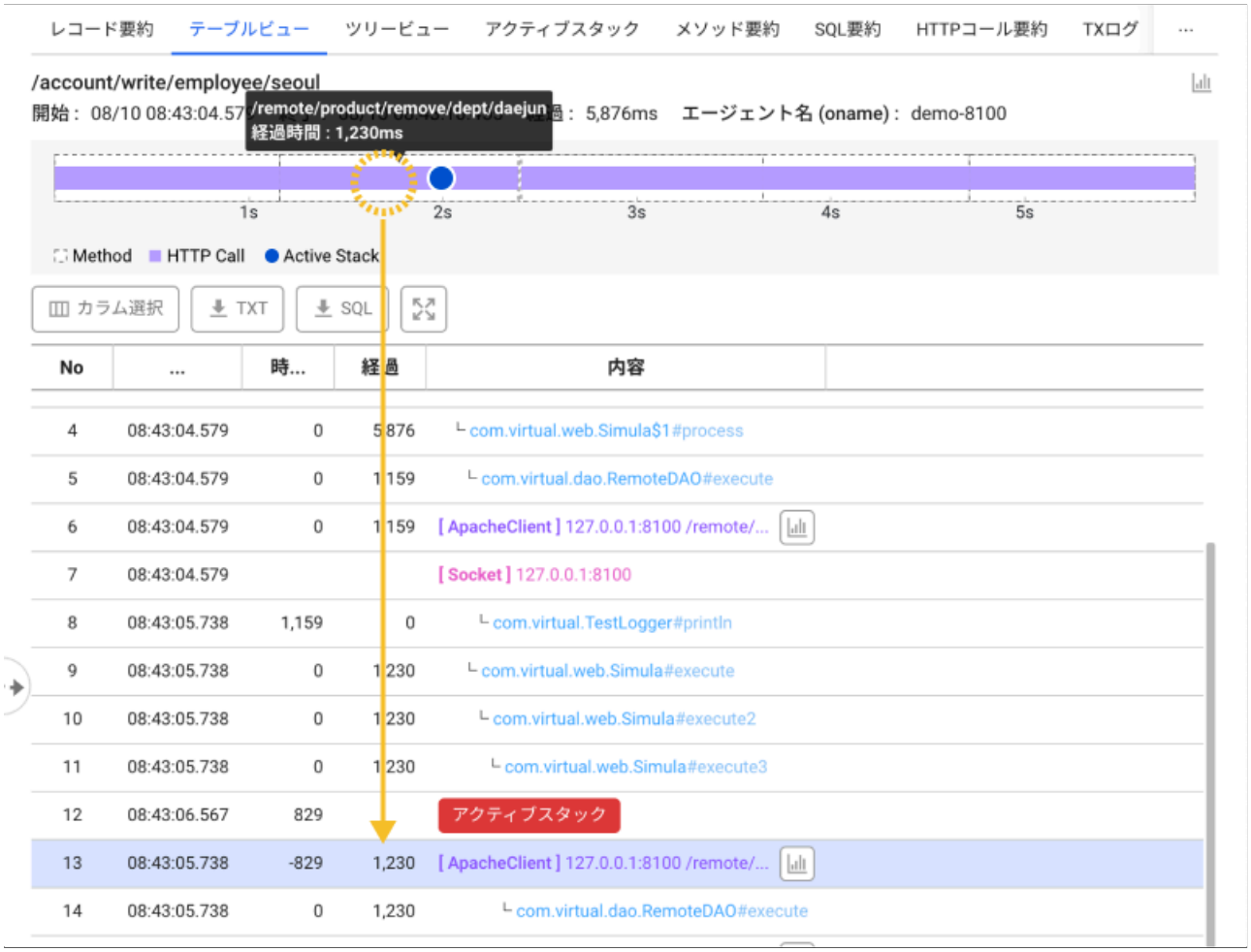
Javaプラットフォームでは、**連携プロジェクト管理**メニューでデータベースプロジェクトを追加すると、連携したプロジェクトの性能メトリクスを照会できます。 連携プロジェクトの追加と分析の詳細については、次の文書を参照してください。

- WhaTapモニタリングサービスでは、**ステップ**(**Step**)は**スパン**(**Span**)と同じ意味で使用されます。
- 2023年7月25日にリリースされた**トレース分析**機能の変更点を確認するには 次の文書を参照してください。

テーブルビュー

テーブルビュータブでは、トランザクションの実行過程を時間順に確認できます。

- トランザクションを実行区間別に分類し、ダイアグラムで確認できます。 全経過時間のうち、各区間別に所要された時間、最も長くかかった区間を素早く把握できます。
- ダイアグラムの各区間を選択すると、そのステップが位置するテーブルの一覧に移動します。



- 赤い丸(●)が位置する領域は、アクティブスタックが収集された時点です。 赤い丸を選択すると、**アクティブスタック**ボタンがあるテーブル一覧に移動できます。 **アクティブスタック**ボタンをクリックすると、その区間で行われたステップ情報を確認できます。

  > ◦ トランザクション一覧からⒶアイコンが表示された項目で確認できます。
  > ◦ アクティブスタックの詳細については、次の文書を参照してください。

- **カラム選択**：テーブル一覧にカラムでメモリ累積情報およびCPU累積情報を追加または非表示できます。 カラム情報は次のとおりです。
  - **No**：ステップの実行順序
  - **時間**：各ステップの開始時刻

- **時間差**：直前ステップの開始時刻から現在のステップに移るまでの待ち時間、外部要因により遅延される場合、経過時間と異なる場合があります。
- **経過**：各メソッドの開始から終了までの所要時間
- **内容**：該当ステップの詳細な実行内容

- **TXT**：トランザクションの基本情報および区間ごとの実行情報をTXTファイルでダウンロードできます。
- **SQL**：トランザクションの基本情報およびSQL実行情報をTXTファイルでダウンロードできます。
  - **置換値を含むダウンロード**: バインド変数値を元の値に置き換え、SQL統計データをダウンロードします。 セキュリティキー(`paramkey`)が設定されている場合は、セキュリティキー入力画面が表示されます。 セキュリティキーを入力しないとダウンロードできません。 セキュリティキーに関するエージェント設定の詳細については、次の文書を参照します。
  - **置換値を含まないダウンロード**: バインド変数値を置き換えない状態でSQL統計データをダウンロードします。
- :トランザクションの要約情報とダイアグラムを非表示にし、テーブル一覧のみを確認できます。 ボタンをクリックすると、隠された情報を再表示します。 テーブル一覧が長い場合は、この機能を利用すると便利です。
- SQLステップを選択すると、パラメータを照会できる**SQL**画面が表示されます。 詳細については、次の文書を参照してください。

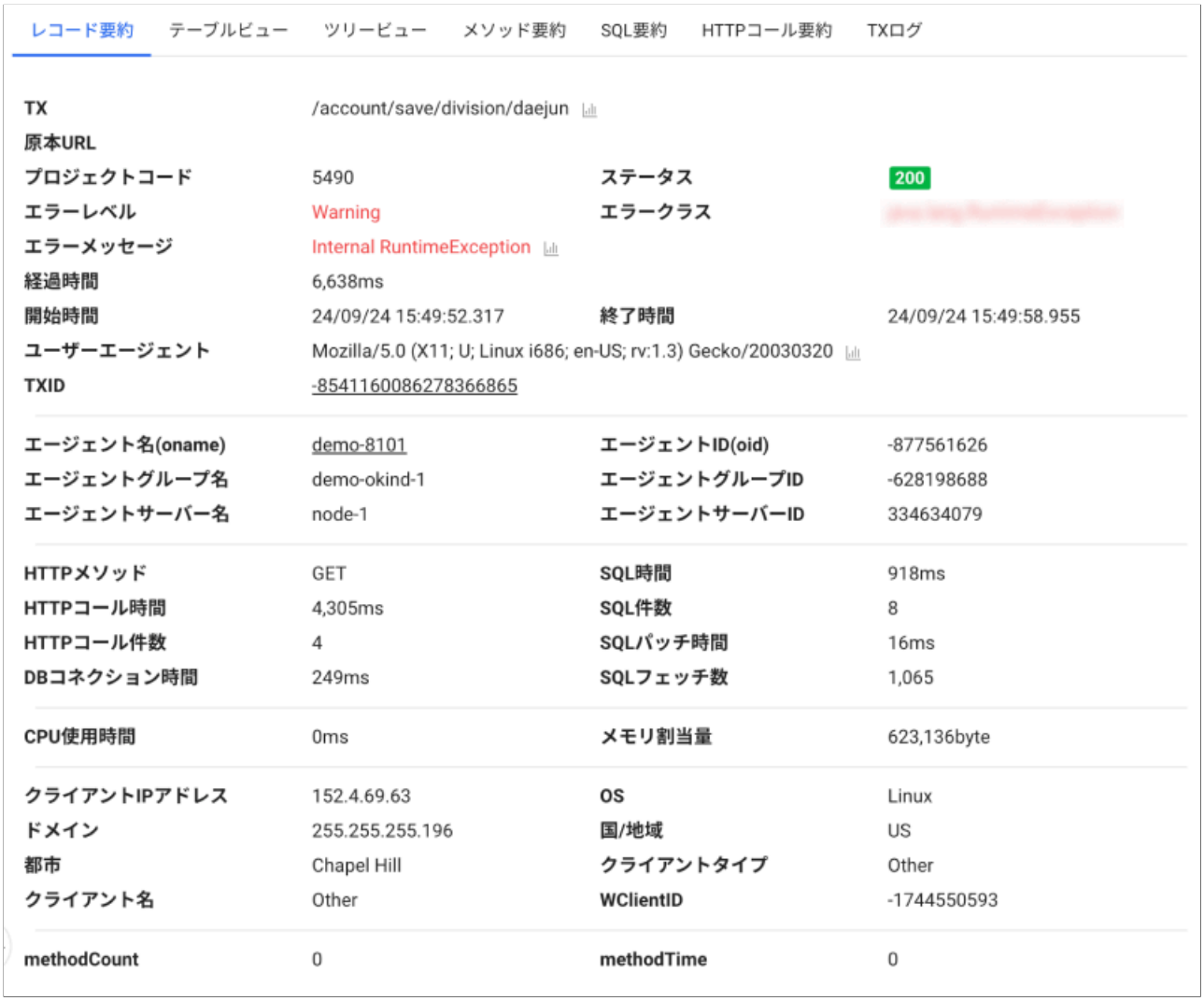
Javaプラットフォームでは**連携プロジェクト管理**メニューを通じてデータベースプロジェクトを追加すると、**DB Connectionステップ**で連携したプロジェクトの性能メトリクスを照会できます。



連携プロジェクトの追加と分析の詳細については、次の文書を参照してください。

レコード要約

トランザクションの基本情報を確認するには、**レコード要約**タブを選択します。 トランザクションの基本情報およびそのトランザクションが実行されたエージェント関連情報、メモリ割り当て使用量、クライアント関連情報を確認できます。



分類	プロパティ	説明
トランザクション	TX	アプリケーションで実行されたAPIまたはURL呼び出し、そのトランザクションのパスです。 正規表現式を使用して一定の形式に変換します。

分類	プロパティ	説明
	原本URL	実際に呼び出されたURLパスです。 変数を含まない具体的なリクエストパスです。 元のURLがある場合は元のURL表示、元のURLがない場合は表示しません。
	プロジェクトコード	WhaTapモニタリングサービスに登録したプロジェクトの識別コードです。
	ステータス	そのトランザクションのHTTP応答ステータスのコードです。
	エラーレベル	そのトランザクションで発生したエラー レベルです。 WarningまたはCriticalと表示されます。
	エラークラス	そのトランザクションで発生したエラー関連クラスです。
	エラーメッセージ	そのトランザクションで発生したエラーメッセージです。
	経過時間	トランザクションの実行時間です。
	開始時間	トランザクションの開始時刻です。
	終了時間	トランザクションの終了時刻です。
	ユーザーエージェント	クライアント関連情報の抽出に利用するブラウザ情報です。
	リファラー	クライアントがブラウザで以前に利用したページアドレスまたは流入パスです。
	TXID	該当トランザクションの固有識別子(ID)です。
	マルチTXID	マルチトランザクションの固有識別子(ID)です。 他のアプリケーションのトランザクションと並列に実行された場合に表示されます。
エージェント	エージェント名(oname)	エージェント名です。
	エージェント	エージェントの固有識別子(ID)です。

分類	プロパティ	説明
	ID(oid)	
	エージェントグループ名	`okind`オプションで設定されたエージェントグループ名です。
	エージェントグループID	エージェントグループの固有識別子(ID)です。
	エージェントサーバー名	`ondoe`オプションで設定したエージェントサーバー名です。
	エージェントサーバーID	エージェント サーバーの固有識別子(ID)です。
ステップ (Step)	HTTPメソッド	HTTPメソッドです。 (GET、POST、PUT、HEADなど)
	HTTPコール時間	外部Http Call時間です。
	HTTPコール件数	外部HTTP Call件数です。
	DBコネクション時間	データベースに接続するまでにかかる時間です。
	SQL時間	SQLの実行時間です。
	SQL件数	SQLの実行件数です。
	SQLパッチ時間	レコードの照会にかかった時間です。 ネストが発生するか、他のロジックが含まれる場合があります。
	SQLフェッチ数	SQLでデータを照会したレコードの件数です。

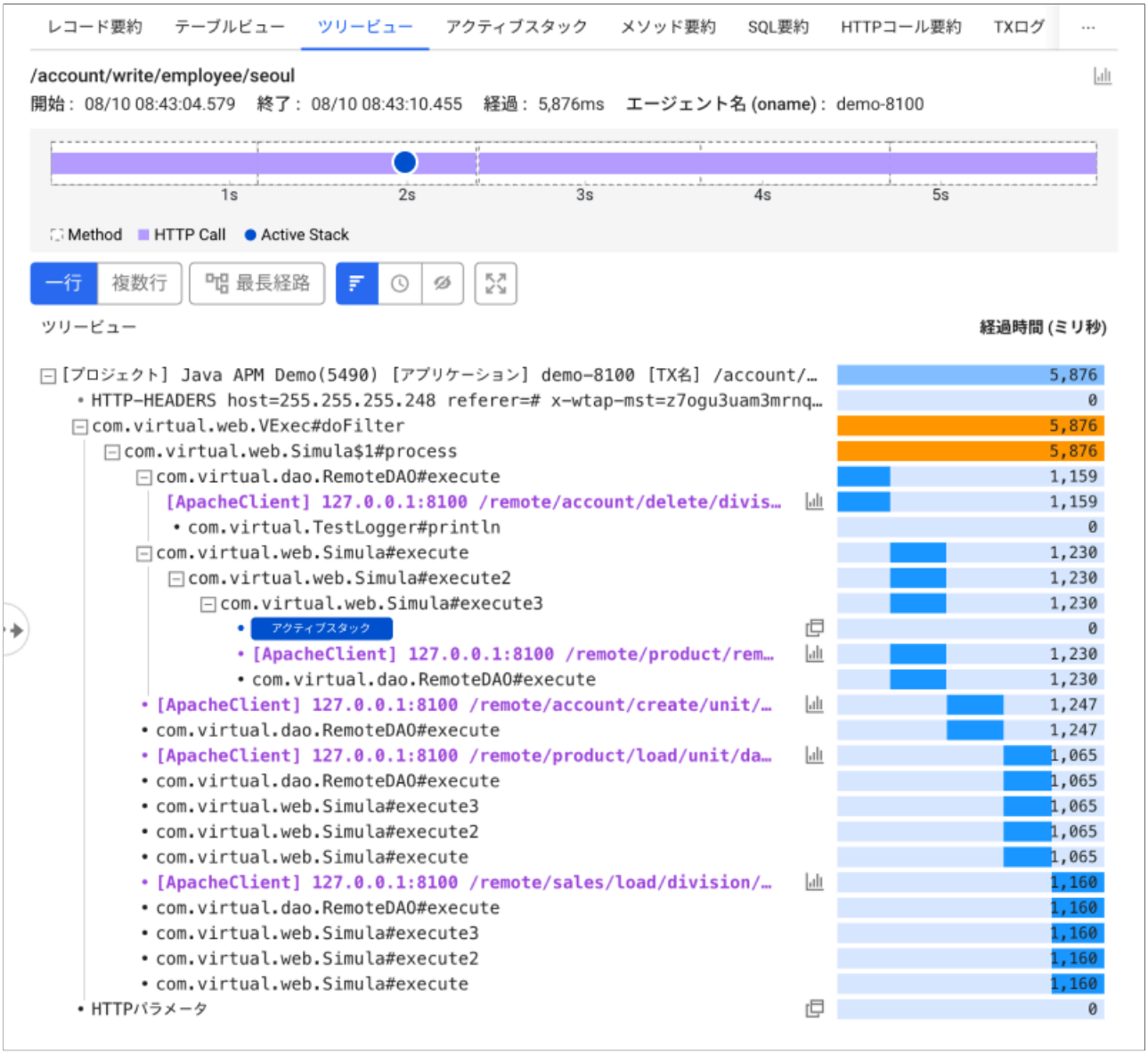
分類	プロパティ	説明
リソース (Resource)	CPU使用時間	トランザクションの処理に使用されたCPU時間です。
	メモリ割当量	そのトランザクションで使用されたメモリ量です。
コンテナ	Pod名	該当トランザクションを処理したKubernetes Podの名前です。
	コンテナID	Pod内で実行されている特定のコンテナの固有識別子です。
クライアント	クライアントIPアドレス	クライアントのIPアドレスです。
	OS	ブラウザが実行されるOS環境です。
	ドメイン	クライアントが接続したIPアドレスに指定されたインターネットアドレスです。
	国/地域	クライアントの国の情報です。
	都市	クライアントの国内の都市情報です。
	クライアントタイプ	クライアントが利用したブラウザの種類です。
	クライアント名	クライアントが利用したデバイス名です。
	WClientID	クライアントの固有識別子です。
Method	methodCount	トランザクションで呼び出されたメソッドの数です。
	methodTime	メソッドの実行にかかった時間です。

- アプリケーションの種類や設定、ステップの種類によって収集する情報は異なる場合があります。 **コンテナ**項目は、アプリケーションがコンテナ環境で実行されている場合にのみ表示されます。
- エラー関連項目は、そのトランザクションでエラーが発生した場合に表示されます。

- トランザクション属性の一部は、統計機能を提供します。 ボタンを選択すると、時系列チャートを通じて照会時間中に発生したトランザクション状況を把握できる画面が表示されます。 チャート内の棒グラフを選択すると、**アプリケーション** > **分析** > **TX検索**メニューに移動して詳細照会できます。
- WhaTapは、クライアントに関する情報をデフォルトで保存します。 ユーザーデータの収集に関するエージェント設定の詳細については、次の文書を参照してください。

ツリービュー

トランザクションの実行過程をツリー形式で確認するには、**ツリービュー**タブを選択します。 各トランザクションとそれに属するトレースの詳細、トレースの開始時刻と所要時間、呼び出し関係を確認できます。 ダイアグラムの各区間を選択すると、そのステップが位置するツリービューに移動します。



- **一行**：各区間別の実行情報に表示されたテキストを一行で表示し、ツリー形式をシンプルに整理できます。
- **複数行**：各区間別の実行情報に表示されたテキストを改行してすべて表示します。
- **最長経路**：最長の経路で移動できます。
- **タイムバーの表示**：経過時間を棒チャートで表示します。
- **時間を表示する**：各区間ごとのタイムスタンプ、ギャップ、経過時間をテキスト形式で表示します。
  - 8秒以上：超過遅延状態で赤色で表現します。

- 3秒以上8秒未満：遅延状態でオレンジで表現します。
- 3秒未満：正常な状態で青色で表示します。

- **時間を表示しない**：時間情報を非表示にします。
- ：トランザクションの要約情報とダイアグラムを非表示にし、ツリービューのみを確認できます。 ボタンをクリックすると、隠された情報を再表示します。 ツリービュー一覧が長い場合は、この機能を利用すると便利です。
- ：SQL変数とHTTPクエリを照会できる画面が表示されます。 詳細については、次の文書を参照してください。

> 開始および所要時間の場合、トランザクション呼び出し環境によって生じる時差を上位トランザクションのトレースやマッピングで修正および表示することで、実際に収集された時間データとの差異が生じる可能性があります。

アクティブスタック



アクティブスタックタブを選択すると、該当トランザクションから収集したアクティブスタックの一覧を確認できます。

- 左側の**アクティブスタック**一覧から収集時点（No、時間）を選択すると、右側に該当時点のスタック情報を表示し

ます。

- **スタック**：アクティブスタックの詳細情報をTXTファイルでダウンロードできます。

> - トランザクション一覧でAアイコンが表示された項目からアクティブスタックを確認できます。
> - アクティブスタックの詳細については、次の文書を参照してください。

AIアクティブスタック分析

Beta

アクティブスタック分析ボタンをクリックすると、収集したアクティブスタックをAIが自動的に分析します。進行中のトランザクションのスタックダンプを基にパフォーマンスの問題を特定し、改善方法を案内します。

分析結果は次の4つのセクションで構成されます。

- **全体要約**：分析したスタック数と全体の実行フローを要約します。
- **共通パターン**：繰り返し現れる実行パターンを表示します。パターン名、発生回数、説明を含みます。
- **発見されたイシュー**：特定したパフォーマンスの問題を深刻度（**High**、**Medium**、**Low**）とともに表示し、判断の根拠となるスタック情報も提供します。

深刻度	基準
High	デッドロック、完全ブロッキング、同一ブロッキング4回以上繰り返し
Medium	I/O待機、2〜3回のロック競合、遅い外部呼び出し
Low	単一発生の待機、軽微な最適化の機会

- **推奨事項**：優先順位に基づいて具体的な改善方法を案内します。

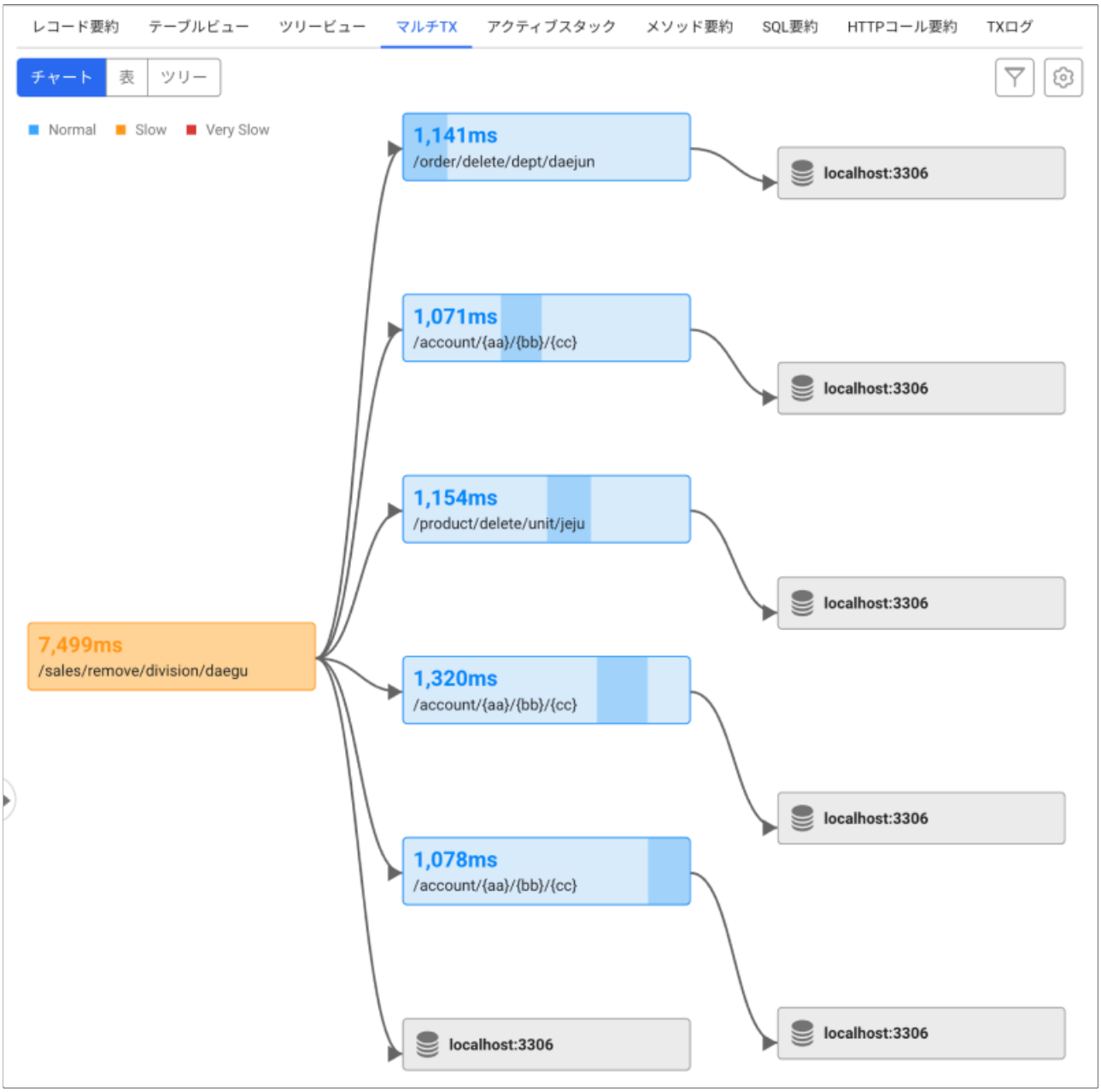
各分析項目には収集したスタック番号（ No.N ）が表示されます。スタック番号をクリックすると、該当スタックの原本を確認できます。

> - AIが生成した分析結果のため、実際の状況と異なる場合があります。参考資料としてご活用ください。
> - プロジェクト**メンバー**権限以上が必要です。

マルチトランザクション

マルチトランザクションとは、他のエージェントやプロジェクトに関連付けられたトランザクションを意味します。**マルチTX**タブでは、WhaTapモニタリングサービスに登録したアプリケーション間の呼び出し関係を確認できます。

> マルチトランザクションを追跡するには、**管理** > **エージェント設定**メニューで `mtrace_enabled` オプションを `true` に設定します。 エージェント設定の詳細については、次の文書を参照してください。



- **チャート**：各トランザクションの呼び出し関係をフローチャート形式で提供します。 トランザクションノードを選択すると、そのトランザクションノードのトレース分析情報を確認できます。 **チャート**モードではマウスを利用して位置を移動したり、スクロールして拡大または縮小できます。
  - ⚙**チャートビュー設定**：チャートに表示するアイテムを表示または非表示します。
- **表**：テーブル形式で、マルチトランザクション内の各トランザクションごとの情報を確認できます。 カラム選択

アイコンを選択して、テーブルヘッダーカラムを編集できます。 各トランザクション項目を選択すると、トレース分析情報を確認できます。

- **ツリー**：ツリー形式でトランザクション間の呼び出し関係を把握できます。 関連した付加機能は、**ツリービュー**タブの機能と同じです。
- **プロジェクト選択**：チャートに表示されるプロジェクトを選択または解除できます。

> ⓘ
> - トランザクション一覧からⓂアイコンがある項目で確認できます。
> - マルチトランザクションの詳細については、次の文書を参照してください。

メソッド要約



No	ク...	メソ...	パラメータ/戻...	数	合計 (ms)	最大 (ms)
1	com.virtual.dao.Remot...	execute	()V	27	39,063	2,776
2	com.virtual.dao.PIITest	exec	(Ljava/sql/PreparedSta...	3	0	0
3	com.virtual.dao.InsertD...	execute2	(Ljava/lang/String;)V	2	4	3
4	com.virtual.dao.Update...	execute2	(Ljava/lang/String;)V	2	18	14
5	com.virtual.dao.Delete...	execute	()V	2	3	2
6	com.virtual.dao.Select...	execute2	(Ljava/lang/String;)V	1	5	5
7	com.virtual.dao.Update...	execute	()V	1	4	4
8	com.virtual.dao.Select...	execute2	()V	1	7	7
9	com.virtual.dao.Select...	execute	()V	1	9	9
10	com.virtual.dao.InsertD...	execute2	()V	1	3	3

メソッド情報のみを確認するには、**メソッド要約**タブを選択しますて。 エージェントに追跡が設定されたメソッド名と所要時間を表示します。 不要なロジックが繰り返し実行される場合、低い件数に比べて実行時間が長くかかる場合を把握し、メソッドロジック改善のための分析情報として活用できます。

- : **TX統計**情報を確認できます。 チャートのグラフを選択すると**分析** > **TX検索**メニューに移動します。 選択した時間帯のトランザクション一覧を確認できます。

- **メソッド**: メソッドに関する詳細情報をTXTファイルにダウンロードできます。

> メソッド(method)と関連するエージェント設定オプションの詳細については、次の文書を参照してください。 ご利用中の製品によって、メソッド追跡サポートの可否は異なる場合があります。

SQL要約



No	データベ...	SQL	数	合計 (ms)	最大 (ms)
1	jdbc:mysql://localhost:...	DELETE FROM Custom...	2	0	0
2	jdbc:mysql://localhost:...	SELECT distance, lat, lo...	1	0	0
3	jdbc:mysql://localhost:...	SELECT distance, lat, lo...	1	0	0
4	jdbc:mysql://localhost:...	UPDATE emp set e.ena...	1	0	0
5	jdbc:mysql://localhost:...	update table set x=# w...	1	0	0
6	jdbc:mysql://localhost:...	SELECT p.Name AS Pr...	1	0	0
7	jdbc:mysql://localhost:...	SELECT customer_info...	1	0	0
8	jdbc:mysql://localhost:...	INSERT INTO EMP VAL...	1	1	1

SQL文についての情報を確認するには、**SQL要約**タブを選択します。 不要に繰り返し実行される場合、低い件数に比べて実行時間が長くかかる場合を把握し、クエリ性能改善のための分析情報として活用できます。

- : **TX統計**情報を確認できます。 チャートのグラフを選択すると**分析** > **TX検索**メニューに移動します。 選択した時間帯のトランザクション一覧を確認できます。
- **SQL**: SQLに関する詳細情報をTXTファイルにダウンロードできます。

HTTPコール要約



No	リモートサー...	URL	数	合計 (ms)	最大 (ms)
1	127.0.0.1:8105	/remote/account/save...	2	4,116	2,776
2	127.0.0.1:8105	/remote/order/save/di...	1	238	238
3	127.0.0.1:8105	/remote/sales/pickup/...	1	922	922
4	127.0.0.1:8105	/remote/sales/remove...	1	1,980	1,980
5	127.0.0.1:8105	/remote/order/save/de...	1	2,639	2,639
6	127.0.0.1:8105	/remote/edu/kill/unit/p...	1	1,519	1,519
7	127.0.0.1:8105	/remote/order/read/de...	1	1,467	1,467
8	127.0.0.1:8105	/remote/order/delete/...	1	485	485
9	127.0.0.1:8105	/remote/sales/kill/emp...	1	2,210	2,210

HTTP呼び出しのリクエスト件数、合計時間、平均時間などを確認するには、**HTTPコール要約**タブを選択します。 不要な外部への呼び出しが繰り返して実行される場合や、低い件数だが比較的に実行時間が長くかかる場合などを把握するなど、トランザクションの遅延要因が外部か内部かを分析する情報として活用できます。

- : **TX統計**情報を確認できます。 チャートのグラフを選択すると**分析** > **TX検索**メニューに移動します。 選択した時間帯のトランザクション一覧を確認できます。
- **HTTP**: HTTP Callに関する詳細情報をTXTファイルにダウンロードできます。

TXログ

トランザクションに関するログ情報を確認するには、**TXログ**タブを選択します。



- : **TX統計**情報を確認できます。 チャートのグラフを選択すると**分析** > **TX検索**メニューに移動します。 選択した時間帯のトランザクション一覧を確認できます。
- **ログ検索機**: ログ検索に移動して、目的のログデータを簡単に照会できます。
- **キーワード検索**：検索バーに検索するテキストを入力し、エンターキーを入力するか、ボタンを選択します。 入力したテキストと一致するキーワードをハイライトで表示します。
- **キーワードハイライト**：キーワードと色を設定すると、自動的にログ一覧でキーワードと一致するテキストをハイライトします。 詳細については、次の文書を参照してください。
- **カラム設定** : ログリストテーブルに表示されるカラムの追加や順序変更ができます。 詳細については、次の文書を参照してください。
- **ログ表示詳細設定**: ログリストに表示される内容を設定できます。 詳細については次の文書を参照してください。

> - **TXログ**タブを有効にするには、ログに関連するエージェント設定オプションを適用する必要があります。 ログ設定の詳細については、次の文書を参照してください。
> - **TXログ**タブは**ログ読取り**権限を持つメンバーのみアクセスできます。 メンバー権限の詳細については、次の文書を参照してください。

キーワードハイライトの設定

1. ハイライトで表示するキーワードを追加するにはボタンを選択します。



2. ご希望のキーワードを入力します。
3. 色アイコンを選択し、ご希望の色を選択します。
4. エンターを入力します。 追加したキーワードが一覧に作成されます。

キーワードハイライト削除

ハイライト一覧から利用しない項目を削除するには、—ボタンを選択します。

テーブルカラムの表示または非表示

1. テーブルの右上の⊞ボタンを選択します。
2. **カラム設定**ウィンドウが表示されたら、左側のリストで目的の項目を選択・解除します。
   右のリストから×ボタンを選択してカラムを非表示にすることもできます。
3. すべての設定を完了した後**確認**ボタンを選択します。 設定した内容がテーブルに適用されます。

テーブルカラムの表示順番の変更

1. テーブルの右上の⊞ボタンを選択します。
2. **カラム設定**ウィンドウが表示されたら、右側のリストで目的の項目をドラッグして順序を変更します。
3. すべての設定を完了した後**確認**ボタンを選択します。 設定した内容がテーブルに適用されます。

テーブルコンテンツの表示設定

1. テーブル右上の⚙ボタンを選択します。
2. **ログ表示詳細設定**ウィンドウが表示されたら、目的のオプションを設定します。
   - **content**または**タグ**オプションを選択して、ログにコンテンツの表示有無を選択します。
   - **タグ管理**: タグは、追加した順にログに表示されます。 タグの順序はドラッグで変更できます。 非活性化されているタグはログに表示されません。
3. すべての設定を完了して✕ボタンを選択します。 設定した内容がテーブルに適用されます。

> ⓘ **ログ表示詳細設定**の詳細については次の文書を参照してください。

追加機能

HTTPパラメータ照会

テーブルビュー タブでそのトランザクションのHTTPパラメータを照会することができます。

1. ページの下にスクロールして、トランザクションの実行の最後のステップに移動してください。
2. 🔒 **HTTPパラメータ**項目を選択します。
3. **HTTP-PARAMETERS**画面が表示されると、**パスワード** ボタンを選択します。
4. 設定したParam Keyを入力します。隠されたパラメータを確認できます。

> ⓘ
> - HTTPパラメータに関するエージェント設定は、次の文書を参考にしてください。
> - **パスワード**：復号化されたパラメータの値を確認できます。 パスワードは `WHATAP_HOME /security.conf (paramkey.txt)`ファイル内の6桁の文字列です。 別の文字列に変更できます。 SQL変数の照会、HTTPクエリの照会、Threadの中止に必要です。
> - エージェントのバージョンによっては、パスワード設定ファイルが異なる場合があります。 security.conf ファイルまたはparamkey.txtファイル内の文字列を変更します。

SQLパラメータの照会

テーブルビュータブでSQLステップを選択するか、**ツリービュー**タブで  ボタンを選択します。 パラメータを照会できる**SQL**画面が表示されます。



- **パスワード**：復号化されたパラメータの値を確認できます。 パスワードは* `WHATAP_HOME` /paramkey.txt*ファイル内の6桁の文字列です。 別の文字列に変更できます。

  > paramkey.txt内のキーは、SQL変数の検索、HTTPクエリの検索、Threadの停止に必要です。

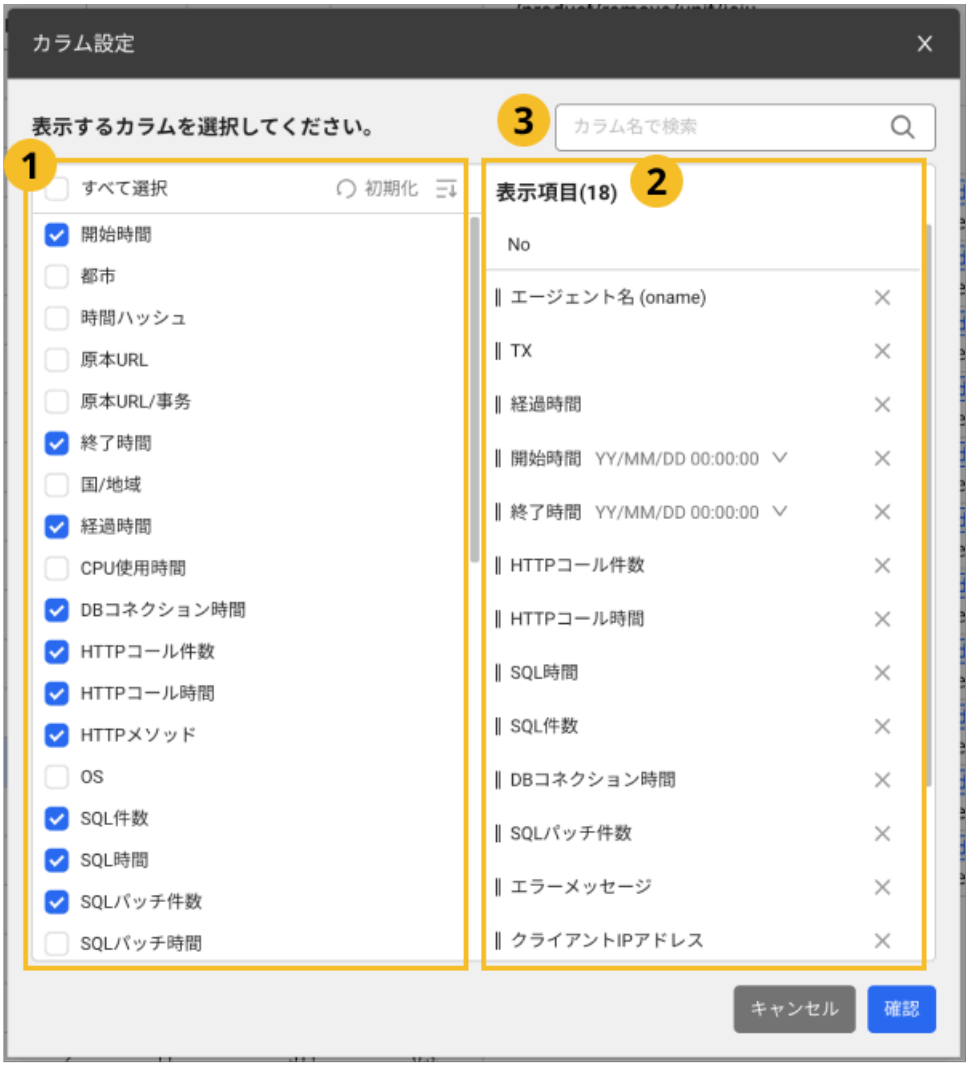
- **SQLフォーマット**：SQL文をインデントまたはフォーマット化して可読性を高めることができます。
- ：SQL文をクリップボードにコピーできます。
- **統計**: **統計** > **SQL**メニューに移動します。

> SQL変数とHTTPクエリを照会するには、次のオプションをエージェント設定に追加します。 DB、SQLに関するエージェント設定の詳細は次の文書を参照してください。
>
> whatap.conf
>
> ```
> # SQLパラメータ照会オプション：オプションが適用されるとSQLパラメータを暗号化して収集します。
> profile_sql_param_enabled=true
> ```

```
# HTTPパラメータ照会オプション：オプションが適用されると、HTTPクエリパラメータを暗号化して収集します。
profile_http_parameter_enabled=true
```

テーブルカラムの設定

トランザクション一覧でテーブルヘッダーカラムを非表示にしたり、希望する項目を追加できます。 カラムの順序を変更することもできます。 **カラム設定**ボタンを選択します。



- 設定を完了した後は、**確認**ボタンを選択すると、設定事項がテーブルに反映されます。
- 3検索バーにテキストを入力し、カラム項目を検索できます。 入力したテキストと一致するカラム項目が表示されます。

カラムの追加

1一覧でテーブルヘッダーカラムに追加する項目を選択します。 すべての項目を追加するには、**すべて選択**を選択します。

カラムを削除する

1一覧から削除するカラム項目のチェックボックスのチェックを解除します。 または2一覧から削除する項目の右側の×ボタンを選択します。

カラム順序の変更

2一覧で順序を変更するアイテムをドラッグして、希望する位置に移動できます。

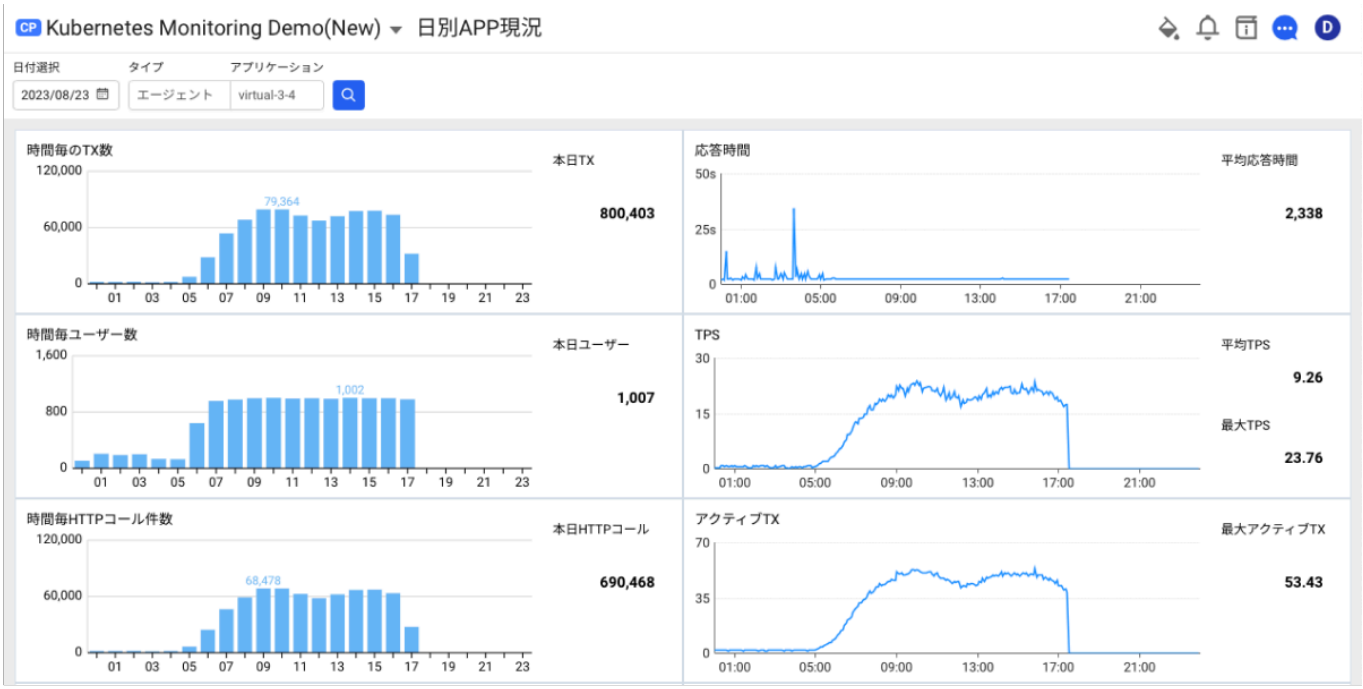
設定事項の初期化

変更事項をすべてキャンセルし、初期化するには、**初期化**ボタンを選択します。

照会データのダウンロード

テーブルに出力されたデータをtxtファイル形式でダウンロードするにはTXTボタンを選択します。

日付別のアプリケーション状況



日付別アプリケーションの状況を確認できます。検索条件は、開始日、分類、アプリケーションで、区分条件は、次のとおりです。

分類	説明
エージェント	コンテナ内部にインストールされているアプリケーションエージェント名
エージェント種類	エージェントがグループ化されたエージェントの種類(`OKIND`)
エージェントサーバー	エージェントが実行されているKubernetesノード名

提供するチャートは次のとおりです。

- 1時間当たりのトランザクション数
- 1時間当たりのユーザー数

- 1時間当たりのHTTP呼び出し件数
- 1時間当たりのSQL件数
- 1時間当たりのSQLフェッチ件数
- 応答時間
- 1秒当たりのトランザクション
- アクティブTX
- 5分間の合計エラー件数
- エラー率

トランザクション検索

トランザクション検索メニューを使用して、特定の期間の各トランザクションの性能属性に関する統計と実行状態を検索できます。すべてのトランザクションを確認し、正常およびエラー状態をすばやく確認できます。提供される情報から、どのようなトランザクションの使用量が多く、リソースを多く消費しているかがわかります。トランザクションの属性に応じて様々なフィルタリングができ、特定の条件のトランザクションを見つけるのに役立ちます。

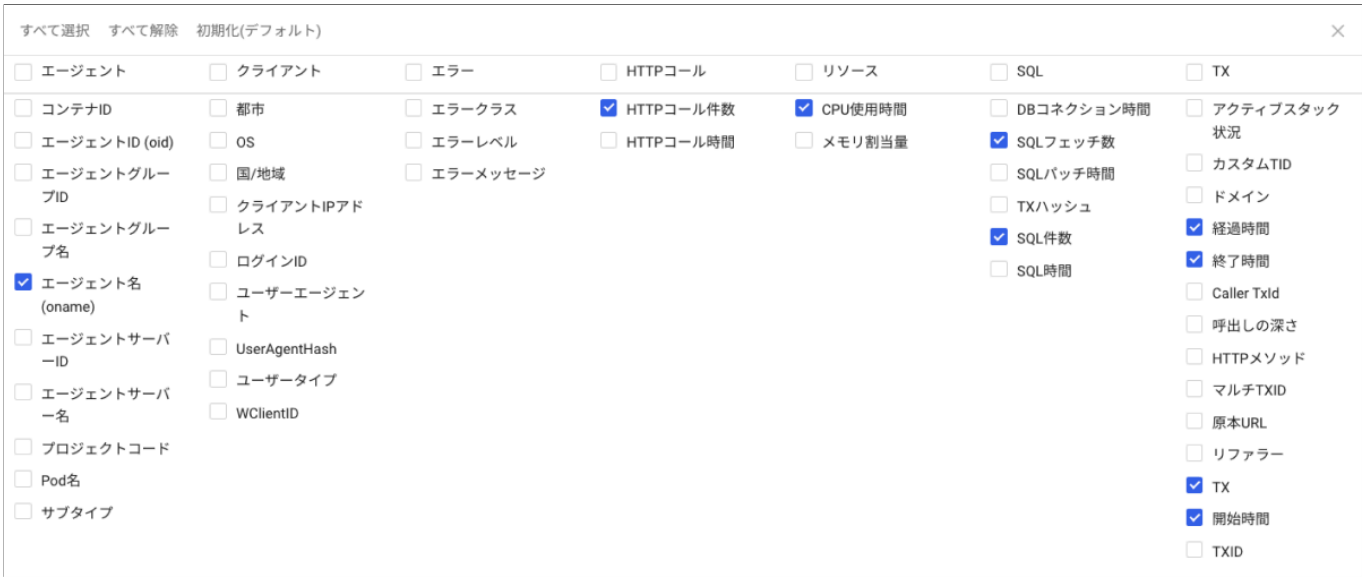


1領域で照会時間を設定した後、コンテナオプションで照会対象を選択します。ボタンを選択すると、検索結果がテーブルに表示されます。2領域の結果リストから希望する項目を選択するとトランザクション情報画面が表示されます。トランザクション情報画面を活用したトランザクショントレースの詳細分析に関する詳細は、次の文書を参考にしてください。

- エラー状態のトランザクションのみを検索するには、**エラーのみ表示**をクリックしボタンを選択します。
- 検索条件によってレスポンス遅延が発生する可能性があるため、検索時間をできるだけ短く設定することをお勧めします。

カラム選択

2領域のカラム選択ボタンをクリックすると、次のようにカラム選択画面が表示されます。希望するカラムを適切に選択します。

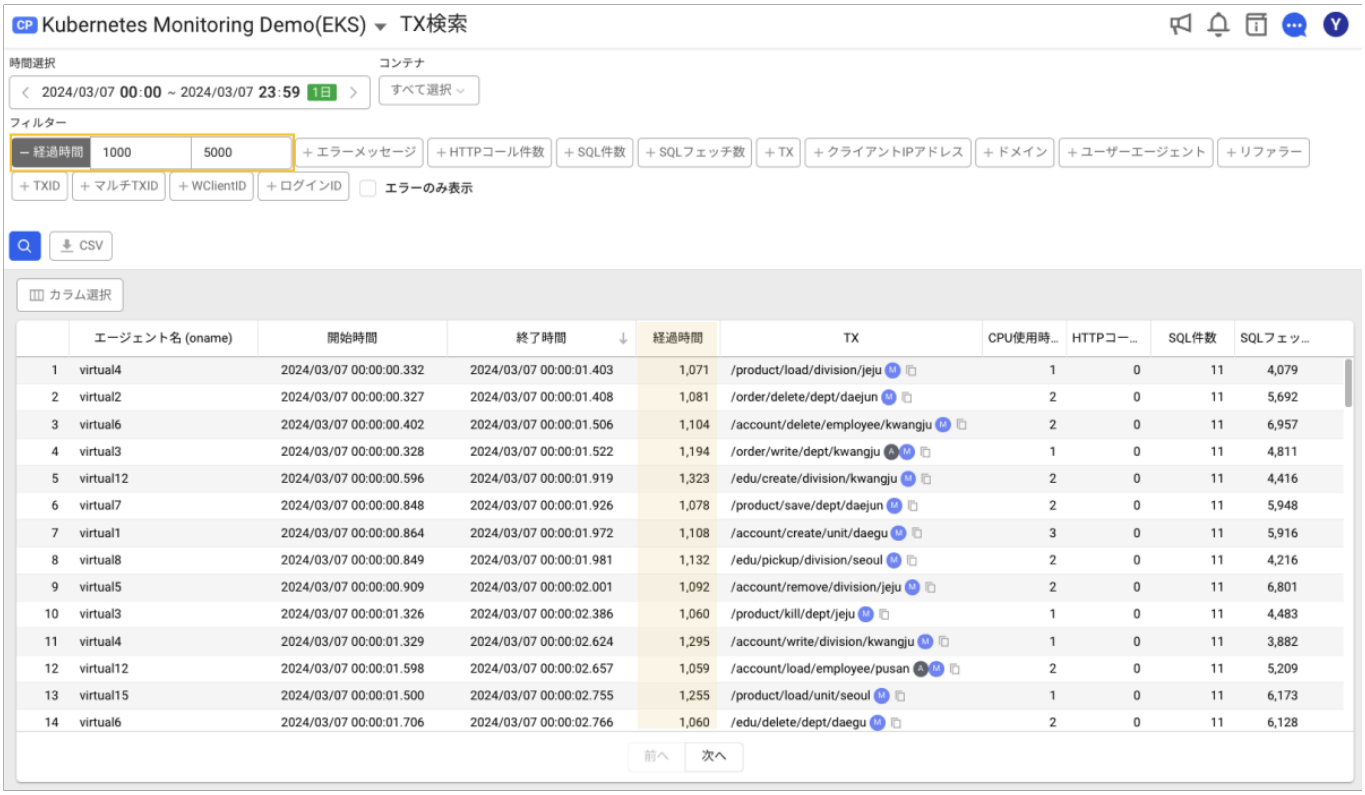


フィルター適用

1領域でフィルターを活用すると詳細検索が可能です。様々な検索条件によるデータの検索方法に対応します。フィルター項目も複数適用できます。希望するフィルターを指定しボタンを選択します。

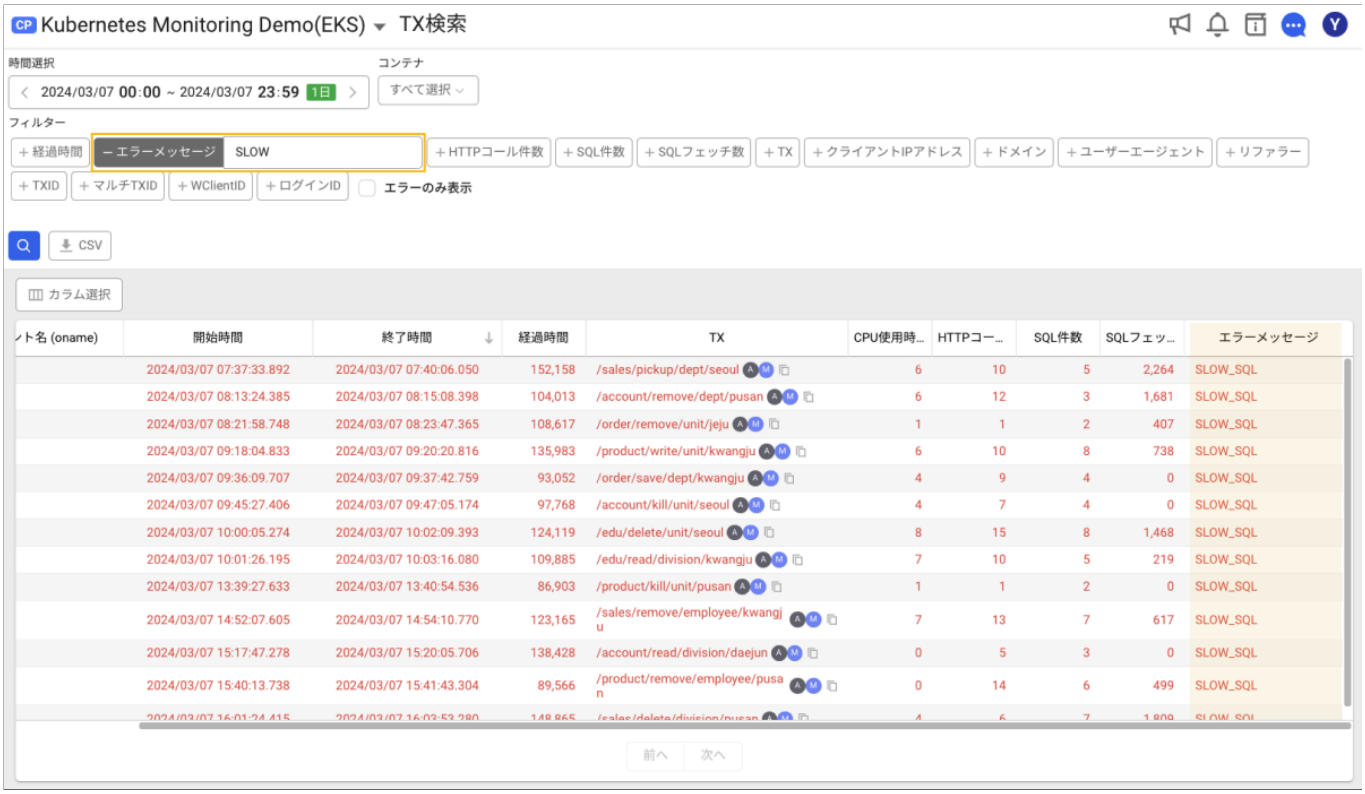
> ⓘ
> - 一部のフィルター項目は、製品によっては対応できない場合があります。
> - 複数条件のフィルターを適用する際に、**マルチTXID**フィルターを適用して検索する場合**TXID**フィルターのみ追加適用できます。

経過時間



トランザクションのレスポンス時間を基準に検索します。基準以上と未満の2つの値を入力して検索します。

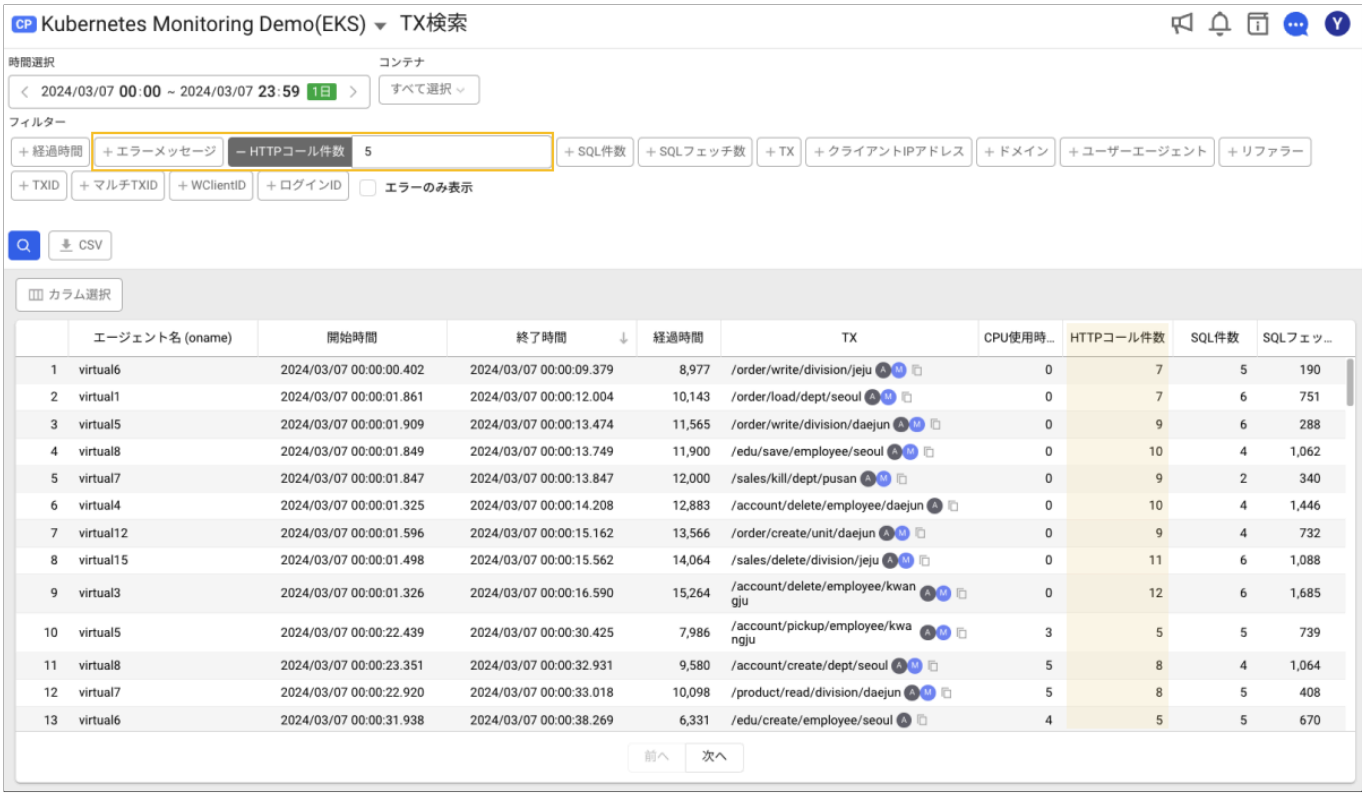
エラーメッセージ



トランザクションのエラーメッセージに基づいて検索します。デフォルト状態ではエラー関連のカラムが表示されない場合があります。カラムを選択して確認してください。

> ⓘ エラー状態のトランザクションは赤で表示されます。

HTTPコール件数



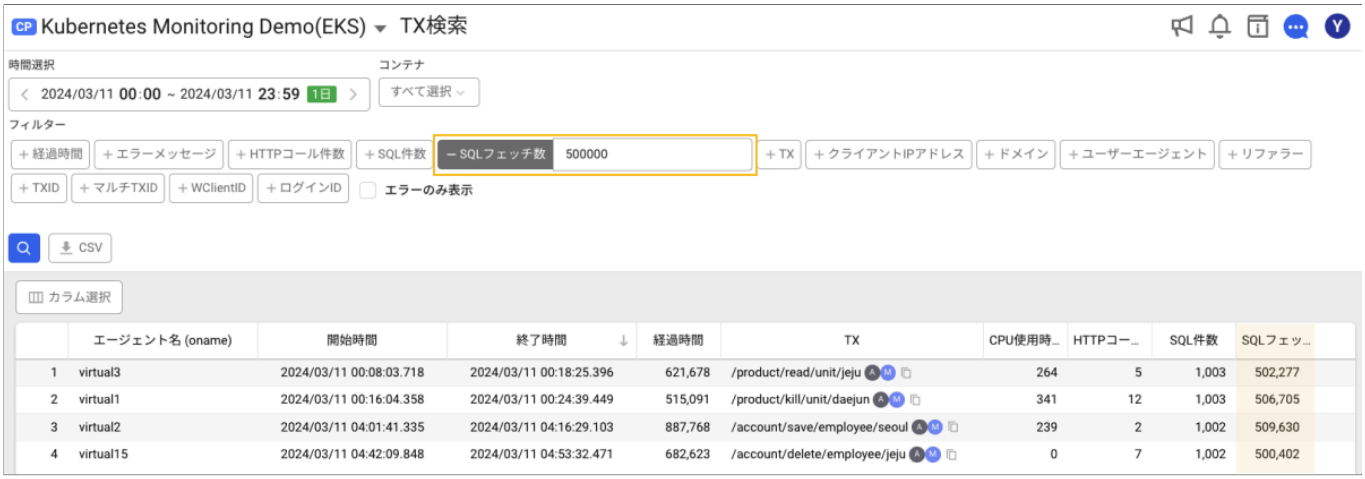
トランザクション中に外部HTTPサービスを呼び出した件数が入力値以上に該当するトランザクションを検索します。
例示画像の場合5件以上を照会します。

SQL件数



トランザクション別のSQL実行件数が設定した入力値以上に該当するトランザクションを検索します。例外的に過度な数のSQLを呼び出すトランザクションを検索するときに利用します。

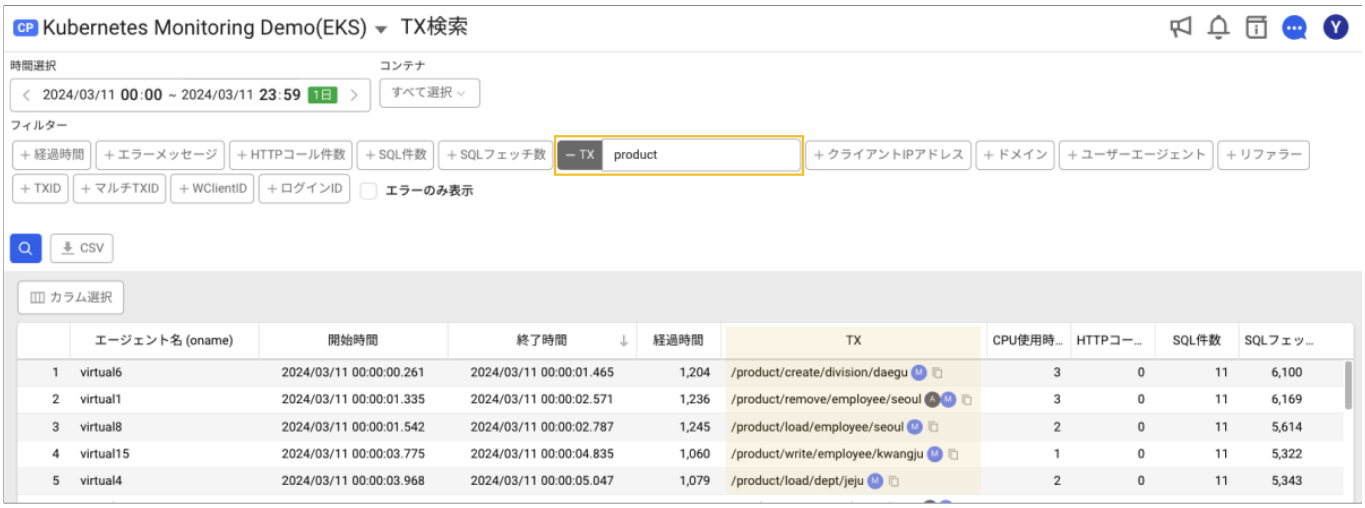
SQLフェッチ件数



	エージェント名 (oname)	開始時間	終了時間	経過時間	TX	CPU使用時...	HTTPコー...	SQL件数	SQLフェッ...
1	virtual3	2024/03/11 00:08:03.718	2024/03/11 00:18:25.396	621,678	/product/read/unit/jeju	264	5	1,003	502,277
2	virtual1	2024/03/11 00:16:04.358	2024/03/11 00:24:39.449	515,091	/product/kill/unit/daejun	341	12	1,003	506,705
3	virtual2	2024/03/11 04:01:41.335	2024/03/11 04:16:29.103	887,768	/account/save/employee/seoul	239	2	1,002	509,630
4	virtual15	2024/03/11 04:42:09.848	2024/03/11 04:53:32.471	682,623	/account/delete/employee/jeju	0	7	1,002	500,402

SQLを実行した後、DBフェッチ件数に基づいて入力値以上に該当するトランザクションを検索します。過度なデータフェッチを実行するトランザクションを検索するときに利用します。

トランザクション



	エージェント名 (oname)	開始時間	終了時間	経過時間	TX	CPU使用時...	HTTPコー...	SQL件数	SQLフェッ...
1	virtual6	2024/03/11 00:00:00.261	2024/03/11 00:00:01.465	1,204	/product/create/division/daegu	3	0	11	6,100
2	virtual1	2024/03/11 00:00:01.335	2024/03/11 00:00:02.571	1,236	/product/remove/employee/seoul	3	0	11	6,169
3	virtual8	2024/03/11 00:00:01.542	2024/03/11 00:00:02.787	1,245	/product/load/employee/seoul	2	0	11	5,614
4	virtual15	2024/03/11 00:00:03.775	2024/03/11 00:00:04.835	1,060	/product/write/employee/kwangju	1	0	11	5,322
5	virtual4	2024/03/11 00:00:03.968	2024/03/11 00:00:05.047	1,079	/product/load/dept/jeju	2	0	11	5,343

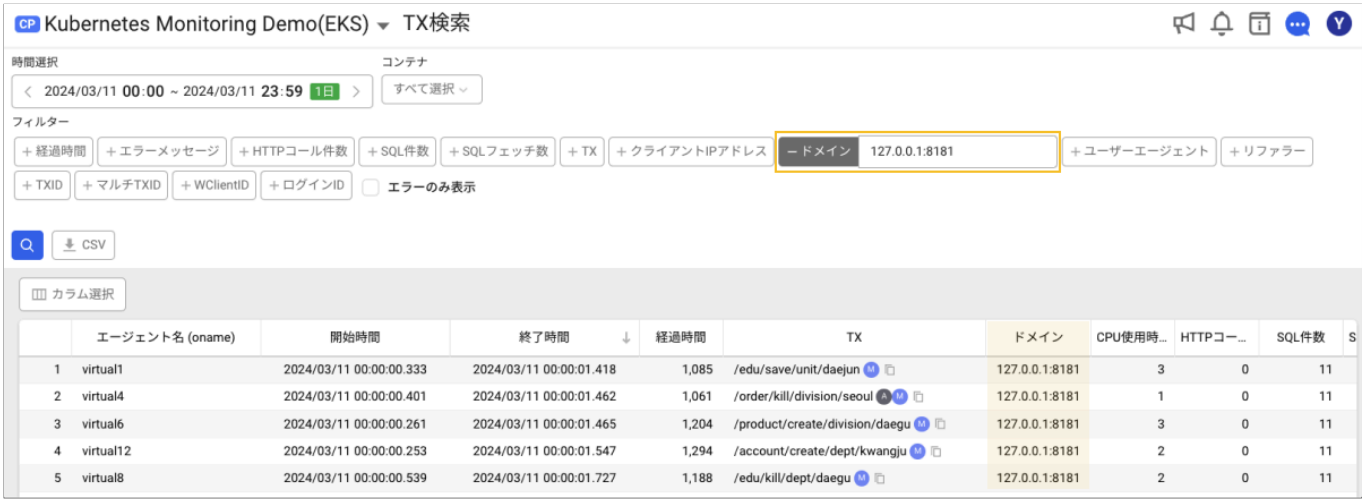
トランザクションベースで検索します。入力した検索文字列は、含めるかどうか(部分一致)で比較検索します。

クライアントIP



クライアントIPベースでトランザクションを検索します。ただ、IP文字列は、PREFIX比較(前方一致)で検索します。

ドメイン



HTTPサービスドメイン(host:port)ベースでトランザクションを検索します。検索条件は、ドメインが完全に一致(完全一致)する必要があります。

ユーザーエージェント



ユーザーエージェントベースでトランザクションを検索(部分一致)します。

リファラー

Referer文字列ベースでトランザクションを検索します。

> ⓘ デフォルト設定状態のエージェントは、Refer情報を収集しません。Referer設定が必要な場合は、次の文書を参照して関連する設定を追加します。
>
> - Java
> - Node.js
> - Python

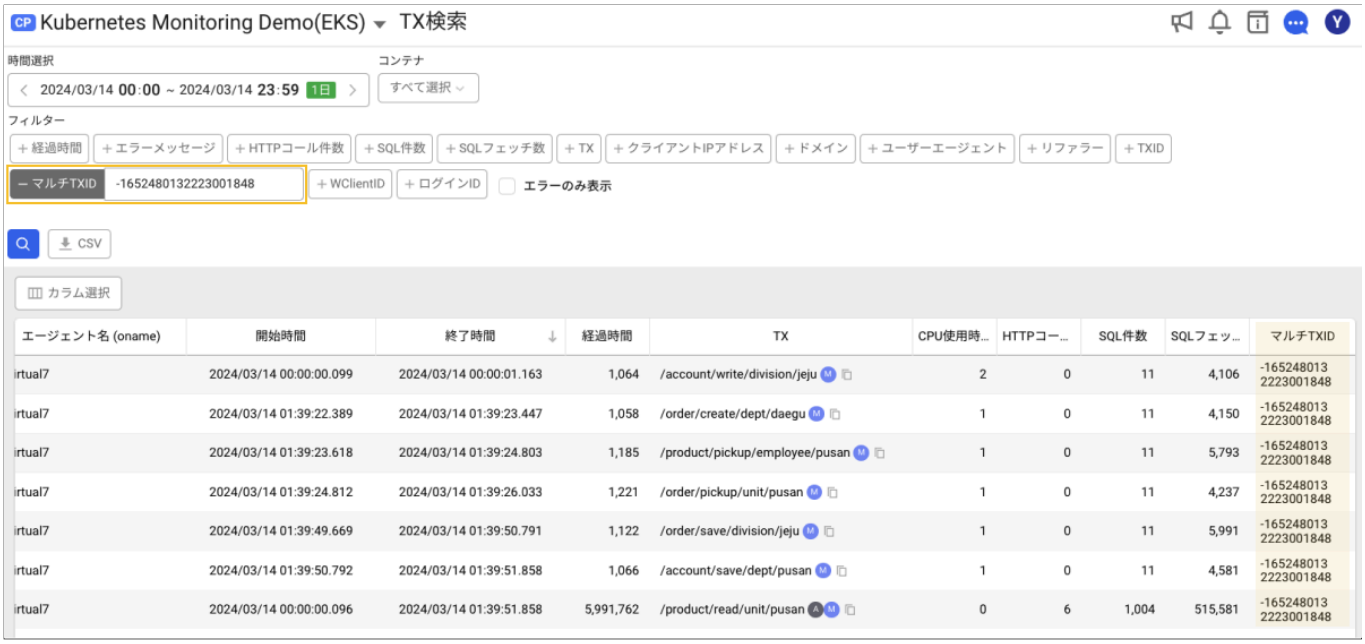
TXID



トランザクションIDベースでトランザクションを検索(完全一致)します。トランザクションIDの場合、検索インデックスが作成されているため、高速で検索できます。

- トランザクションIDは、**トランザクション情報**画面の**レコード要約**タブから確認できます。詳しい内容は、次の文書を参考にしてください。
- 最新バージョンのエージェントを使用していない場合、インデックスを使用しないフルスキャンが発生する可能性があります。

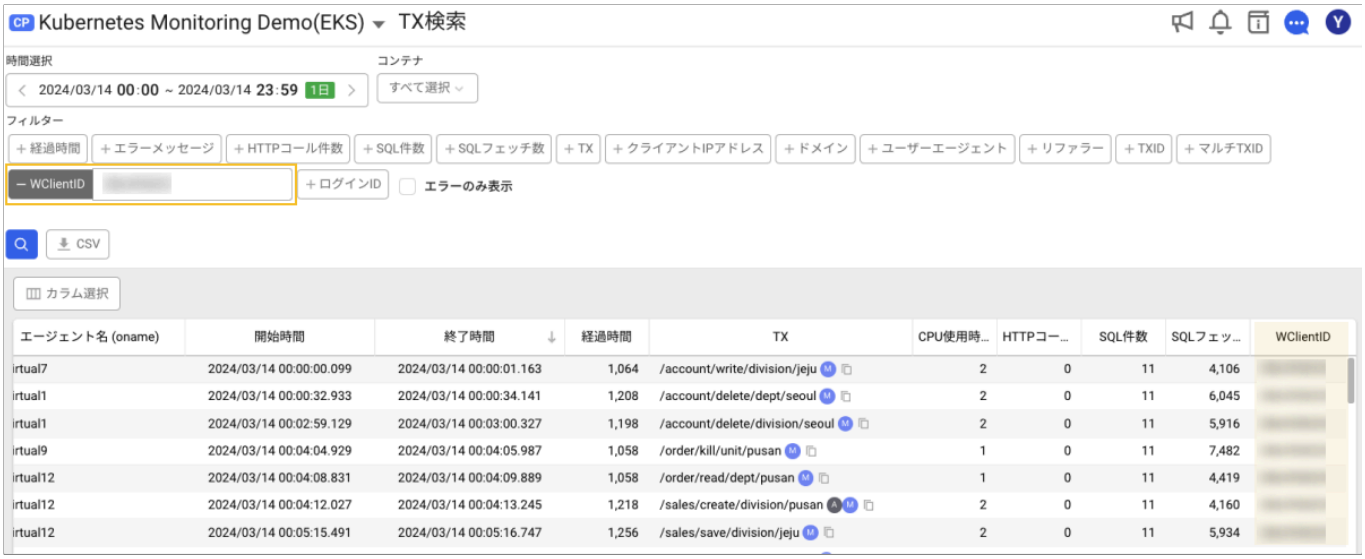
マルチTXID



マルチTXIDベースでトランザクションを検索(完全一致)します。

- マルチトランザクションIDは、**トランザクション情報**画面の**レコード要約**タブで確認できます。詳しい内容は、次の文書を参考にしてください。
- 最新バージョンのエージェントを使用していない場合、インデックスを使用しないフルスキャンが発生する可能性があります。
- **マルチTXID**フィルターを適用して検索する場合、**TXID**フィルターのみ追加できます。

WClientID



クライアントID(wClientId)は、WhaTapエージェントがユーザーを区分するために付与するIDです。このID値に基づいてトランザクションを検索(完全一致)します。検索のための個別のインデックスはありません。

> ⓘ WhaTapは、クライアントに関する情報をデフォルトで保存します。

ログインID

ログインIDに基づいてトランザクションを検索します。

> ⓘ 基本設定状態のエージェントでは、ログインID情報を収集しません。ログインIDの設定が必要な場合は、次の文書を参照して設定を追加します。

検索結果のダウンロード

ユーザーが設定したフィルター条件に従って検索された結果をCSVファイルにダウンロードできます。

1. 検索ボタンをクリックして検索結果を照会します。
2. CSVボタンをクリックします。
3. 最大CSV行数を入力します。入力した行数だけ、照会された結果を保存できます。
4. **ダウンロード**ボタンを選択するとCSVファイルでダウンロードします。

> CSVファイル名は、profile_ {pcode} _ YYMMDD .csv形式です。

警告アラート

監視対象のシステムまたはアプリケーションで問題が発生した際に、あらかじめ設定した条件に基づいてユーザーへ通知を送信する機能です。通知は、メール、SMS、メッセンジャー、アプリのプッシュ通知など、さまざまなチャネルを通じて配信されます。

たとえば、CPU 使用率、メモリ使用量、ディスク I/O、ネットワークトラフィック、トランザクション数、エラー発生、レスポンスタイムの遅延などの主要なパフォーマンス指標にアラート基準を設定しておくと、その基準を超過した場合や異常な兆候が検出された際に通知が送信されます。

WhaTap の AI ベースのアラート機能を使用すると、複雑な条件を手動で設定する必要はありません。AI がアプリケーションの正常な動作パターンを学習し、異常パターンを自動的に検知して通知します。また、タグカウントアラート機能を活用することで、特定のタグを基準に集計された値に条件を設定でき、複雑な状況でも高精度な検知が可能です。

主な機能

- **イベント設定**: 閾値ベースのアラート条件および通知チャネルの設定
- **通知設定**: ユーザーごとの通知チャネルおよび時間帯設定
- **イベント履歴**: アラート発生履歴の確認

権限

イベントを追加または設定するには、**アラート設定権限** が必要です。

- **閲覧**: メトリクスイベント設定を確認するには、**閲覧・分析権限** が必要です。
- **追加／編集／削除**: 設定を変更するには、**アラート設定権限** が必要です。
- **通知受信**: 通知を受信するには、**通知受信権限** が必要です。

> **参考資料**
>
> 詳細については、メンバー権限 を参照してください。

イベント設定

ホーム画面 > プロジェクト選択 > **アラート通知** > **イベント設定**

イベント設定 では、監視対象のシステムやアプリケーションで発生する可能性のある異常兆候を基準に、イベントを設定・管理できます。設定した条件が満たされると、メール、SMS、メッセンジャー、アプリのプッシュ通知など、希望するチャネルで通知を受信できます。監視の目的に応じて、イベントを柔軟に構成できます。

メトリクス

メトリクスイベントは、リアルタイムデータを基にシステムおよびアプリケーションのパフォーマンス指標を監視できます。

複合メトリクス

複合メトリクス機能では、複雑なルールを活用してイベントを作成し、警告アラートを送信できます。

ヒットマップパターン

ヒットマップパターンは、応答時間や呼び出し密度などの分布変化をもとに異常兆候を自動的に検知し、警告アラートを送信できます。

異常値

異常値機能では、想定外のパターンを検出して警告アラートを送信するよう設定できます。

リアルタイムログ

リアルタイムで収集されたログから設定したイベント条件が検出された場合、警告アラートを送信します。

複合ログ

ログの内容を基準に警告通知を発生させる複合ログイベントを設定します。

トランザクション

トランザクションイベントでは、特定のURL、エラー、ステータスコードなどを基準にトランザクションを詳細に監視できます。

ホーム画面 > プロジェクト選択 > **アラート通知 > イベント設定**

イベント設定 では、監視対象のシステムやアプリケーションで発生する可能性のある異常兆候を基準に、イベントを設定・管理する機能を提供します。設定した条件を満たすと、メール、SMS、メッセンジャー、アプリプッシュなど、希望するチャネルで通知を受信できます。監視目的に応じてイベント構成を設計できます。

メトリクス

メトリクスイベントは、リアルタイムデータを基にシステムおよびアプリケーションのパフォーマンス指標を監視できます。

複合メトリクス

複合メトリクス機能では、複雑なルールを活用してイベントを作成し、警告アラートを送信できます。

ヒットマップパターン

ヒットマップパターンは、応答時間や呼び出し密度などの分布変化をもとに異常兆候を自動的に検知し、警告アラートを送信できます。

異常値

異常値機能では、想定外のパターンを検出して警告アラートを送信するよう設定できます。

リアルタイムログ

リアルタイムで収集されたログから設定したイベント条件が検出された場合、警告アラートを送信します。

複合ログ

ログの内容を基準に警告通知を発生させる複合ログイベントを設定します。

トランザクション

トランザクションイベントでは、特定のURL、エラー、ステータスコードなどを基準にトランザクションを詳細に監視できます。

参考

- イベント設定時は、不要なイベント発生を防ぐために適切なしきい値を設定してください。
- 通知設定では、問題解消時に通知される回復通知もあわせて検討すると、運用対応に役立ちます。

共通機能

すべてのイベント設定は **JSON または Excel 形式** でダウンロードできます。同じ種類のプロダクト設定を共有することで、繰り返し発生するイベント設定作業を削減できます。

- **JSON 一括編集**: すべてのイベント条件を JSON 形式で一括編集できます。
- **JSON 一括ダウンロード**: 現在登録されているイベント条件を JSON ファイルとして保存できます。
- **Excel 一括ダウンロード**: イベント一覧を Excel ファイルとしてダウンロードし、外部レポートやバックアップに活用できます。

JSON 一括編集

1. 画面上部の [JSON 一括編集] ボタンをクリックします。
2. **JSON 一括ダウンロード** 機能でダウンロードした JSON ファイルを読み込むか、直接編集します。
3. [上書き] ボタンをクリックして保存します。
4. 確認メッセージが表示されたら、[上書き] ボタンをクリックします。

上書き後は、以前のイベント設定に戻すことはできません。

- `productType` または `platform` プロパティの値を任意に変更すると、他のプロジェクトでイベントが正常に動作しなくなる可能性があります。
- JSON 編集画面には検証機能が組み込まれています。形式や値が正しくない場合、編集画面下部にエラーメッセージが表示され、[上書き] ボタンは無効になります。



JSON データ構造

```
{
 "productType": "{productType}",
 "platform": "{platform}",
 "events": {
  "HITMAP": {
   ...
  },
  "LOG_REALTIME": [
   {...}
  ],
  "METRICS": [
   {...}
  ],
  "COMPOSITE_LOG": [
   {...}
  ],
  "BASIC": [
   {...}
  ],
  "COMPOSITE_METRICS": [
   {...}
  ],
  "ANOMALY": [
   {...}
  ]
 }
}
```

JSON フィールド	説明
productType	プロダクトタイプ（例: APM、Infra など）
platform	プラットフォーム情報（例: java、node など）
events	イベントデータ
ㄴ HITMAP	ヒットマップパターン

JSON フィールド	説明
ㄴ LOG_REALTIME	リアルタイムログイベント
ㄴ METRICS	メトリクス
ㄴ COMPOSITE_LOG	複合ログイベント
ㄴ BASIC	基本イベント
ㄴ COMPOSITE_METRICS	複合メトリクス
ㄴ ANOMALY	異常値

JSON 一括ダウンロード

画面上部の [JSON 一括ダウンロード] ボタンをクリックします。

- JSON ファイル名は integrated-event-`YYYYMMDD`.json 形式です。
- JSON データの構造や `events` 属性配下のサブ属性は、プロダクト（ `platform` ）またはプロダクトタイプ（ `productType` ）によって異なる場合があります。

JSON データ内の `productType` または `platform` プロパティの値を任意に変更すると、他のプロジェクトでイベントが正常に動作しなくなる可能性があります。

Excel 一括ダウンロード

画面上部の [Excel 一括ダウンロード] ボタンをクリックします。

メトリクス

ホーム > プロジェクト選択 > **アラート通知** > **イベント設定** > **メトリクス** タブ

プロジェクトに設定されているすべてのメトリクスベースのイベント一覧を確認できます。設定済みのメトリクスイベントを確認、修正、削除したり、新しいイベントを追加したりできます。

基本オプション

- **イベントレベル**: イベントの深刻度を示し、正常（Info）、警告（Warning）、危険（Critical）の3段階に分類されます。
- **エージェント重複アラート抑止**: 同じ名前のエージェントが同時に2つ以上動作している場合、選択した時間（30秒、1分、3分、5分、10分）の間はイベントが発生しません。
- **検索**: イベント名を基準に検索します。
- **JSON ダウンロード／アップロード**: JSON エディタでイベントルールを直接編集するか、ファイルとしてダウンロードして編集後に再アップロードできます。
- **+ イベント追加**: 新しいメトリクスイベントを追加します。
- **カラムサイズ調整**: 各カラムの境界をドラッグして幅を調整できます。

表 | メトリクスイベント一覧構成

項目	説明
No.	イベントの番号
	イベントの有効化状態 – 有効: 条件を満たすとイベントが発生 – 無効: イベントは動作せず、条件も評価されません
	イベント設定の編集および削除
イベント名	ユーザーが指定したイベント名 – 最大255文字

項目	説明
ルール	イベント発生条件
対象	イベントが適用される監視対象
発生回数	イベント発生の基準となる期間と回数 – 選択した時間内に設定回数分条件を満たすとイベントが発生
一時停止	アラート発生後、同一イベントの発生を一定時間停止 – 選択可能時間: 使用しない、5分、10分、15分、20分、30分、1時間、2時間、3時間、6時間、12時間、1日 – 回復通知が有効な場合、RECOVERED（正常）通知後、選択した時間は同一イベントが発生しても通知されません
回復通知	Critical および Warning レベルのイベントが解消された際に、RECOVERED（正常）通知を受信するかどうか – 有効: 進行中から正常状態に遷移すると RECOVERED 通知を送信し、イベント履歴で進行中／解消状態を区別して確認可能 – 無効: しきい値を超えるたびに単発イベントが発生
イベント受信	該当イベント通知を受信するユーザーまたはグループ（受信タグ）

イベント設定

メトリクスイベントは、監視対象で発生する事象を条件に基づいて定義し、効率的に通知を受信できるよう設定するための機能です。

イベント追加

メトリクスイベントを追加するには、**イベント設定** > **メトリクス** タブに移動し、右上の [+ イベント追加] ボタンをクリックします。

メトリクスイベント追加手順
Step 1. テンプレート選択: イベントの基本構成を選択します。
Step 2. イベント条件定義: どの条件でイベントを発生させるかを設定します。
Step 3. イベント対象選択: イベントが適用される監視対象を選択します。
Step 4. 基本情報および通知設定: イベント名、メッセージ、受信対象を設定します。

テンプレート選択

1. イベントの構成方法に応じて、画面左側の 3 つのテンプレートのいずれかを選択します。
   - **新規作成**: カテゴリやメトリクスを一から自由に構成できます。複雑な条件や詳細な監視ルールが必要な上級ユーザーに適しています。
   - **クイック設定**: あらかじめ定義されたカテゴリ、メトリクス、基本しきい値が提供され、しきい値のみを調整して素早く適用できるため、初心者に適しています。
   - **高度な設定**: あらかじめ定義されたカテゴリやメトリクスを基に、条件を追加したり自由に変更したりできます。

> ⓘ メトリクスイベントでは、さまざまな設定方法をサポートしています。利用する監視プラットフォームによって、提供されるテンプレートは異なる場合があります。

イベント条件定義

イベント条件定義 のステップでは、イベント発生の判断に使用するメトリクス設定、発生回数、一時停止、回復通知、イベント動作時間を設定します。

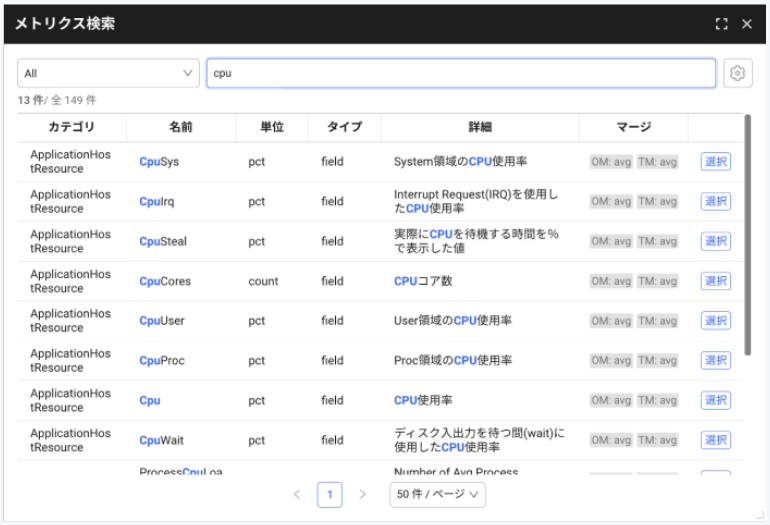
指標設定

1. 検索欄にカテゴリを入力するか、カテゴリー覧からカテゴリ（指標）を選択します。
   指標設定を進めるには、カテゴリの選択が必要です。

- カテゴリ一覧では、カテゴリ名、データ収集間隔、キーを確認できます。直近 3 時間にプロジェクトで収集されたメトリクスデータが表示されます。

2. 一覧にない指標を使用する場合は、**直接入力** を選択してカテゴリを手動で入力します。

> **カテゴリや指標が不明な場合は、メトリクス照会を活用してください。**
>
> 初めて指標を設定する場合、メトリクス照会を利用すると効率的に指標を見つけられます。
>
> - メトリクス照会で指標を検索すると、関連するカテゴリや詳細情報を確認できます。
> - 検索結果から必要なメトリクスの [選択] ボタンをクリックすると、選択した値がイベント設定画面に自動的に反映されます。



3. イベントレベル（Critical、Warning、Info）を選択します。1 つの指標に対して複数のレベルを設定できます。

> **イベントレベルの動作**
>
> 複数の条件を満たしても、最も優先度の高いレベル（Critical → Warning → Info）のみがイベントとして発生します。詳細は レベル動作ガイド を参照してください。
>
> - イベントレベルは、危険（Critical）、警告（Warning）、正常（Info）に分類されます。
> - 同一指標に対して段階的なしきい値でレベルを設定することを推奨します。

4. 選択したイベントレベルのフィールド、演算子、値を入力します。

   a. **+ 追加** をクリックすると条件を追加できます。

   - `&&`: **すべての条件を同時に満たした場合** にイベントが発生
   - `||`: **いずれか 1 つ以上の条件を満たした場合** にイベントが発生

   b. — をクリックすると、その条件が削除されます。

> ⓘ 条件式の書き方や対応する演算子については、条件設定ガイド を参照してください。

発生回数

条件が何回満たされた場合にイベントを発生させるかを設定します。継続的な問題のみを検知したい場合に有効です。

1. 発生頻度を選択します。
   - **連続**: 設定した回数分、イベントが連続して発生した場合にイベントを発生
   - **直近**: 選択した時間内に、設定した回数分イベントが発生した場合にイベントを発生
     - 選択可能時間: `5秒`, `10秒`, `15秒`, `30秒`, `1分`, `2分`, `3分`, `4分`, `5分`, `10分`, `30分`, `1時間`
2. 発生回数を入力します。
   - **Interval** の値は、選択したカテゴリのデータ収集間隔を示します。（Interval: 10s）

一時停止

過剰なイベント発生を防ぐため、イベント動作を一時停止する時間を選択してください。イベント発生後、設定した時間が経過するまで同一イベントは発生しません。

- 選択可能時間: `使用しない`, `5分`, `10分`, `15分`, `20分`, `30分`, `1時間`, `2時間`, `3時間`, `6時間`, `12時間`, `1日`

> ⓘ **解消通知が有効な場合**
>
> 正常状態の通知（**RECOVERED**）を受信した時点から、設定した時間が経過するまで、同一イベントが再発しても通知は送信されません。

解消通知

解消通知機能を有効にすると、イベント発生後の状態変化（進行中 → 解消）を追跡できます。**イベント履歴** メニューでは、進行中のイベントとして表示されます。

- 解消通知は Critical および Warning レベルにのみ適用されます。Info レベルは常に単発イベントとして動作します。
- Critical または Warning レベルのイベントが解消されると、正常（RECOVERED）状態の通知が送信されます。
- トグルボタンをクリックして **有効化** または **無効化** してください。
  - **有効**: イベントは状態ベースで動作します。しきい値を超えると進行中となり、値がしきい値未満に回復すると解消状態に遷移します。
  - **無効**: しきい値を超えるたびに単発イベントが発生します。

イベント動作時間

イベントが特定の時間帯（勤務時間／非勤務時間／点検時間など）にのみ動作するよう、タグを設定してください。タグを設定しない場合、イベントが有効な状態では 24 時間常時動作します。

1. **+ 追加** をクリックします。
2. イベント動作時間で **+ 新しいタグを作成** をクリックします。
3. **イベント動作時間タグ作成** で **タグ名、曜日、時間、色** を選択し、[適用] ボタンをクリックします。
4. 作成したタグはタグ一覧で確認でき、チェックボックスを選択すると適用されます。
5. タグを修正または削除するには、 をクリックし、**イベント動作時間タグ修正** で修正または削除してください。

> タグを削除すると、そのタグが適用されているすべてのユーザーからタグが削除されます。ただし、イベントルールで使用中のタグは削除できません。

シミュレーション

シミュレーション機能では、設定したイベント条件を過去データに基づいて事前にテストできます。想定されるイベント件数が多すぎる、または少なすぎる場合は、条件を調整して最適なしきい値を見つけることができます。

- **ルール検証**: 新しく作成したイベントルールを過去データに適用した場合の結果を確認できます。
- **しきい値チューニング**: イベントが頻発する、またはまったく発生しない場合に、適切なレベルへしきい値を調整します。

シミュレーション開始

1. イベント条件の設定が完了したら、[シミュレーション] ボタンをクリックします。
2. 想定イベント件数と可視化された結果を確認します。

シミュレーション結果の理解

1. **時系列データ**

   指標の時間帯別変化と各レベルのしきい線がグラフで表示され、イベント発生タイミングを直感的に確認できます。

2. **レベル別発生状況**

   画面右側に Critical、Warning、Info レベルごとのイベント発生回数が表示され、全体の分布や頻度を把握できます。

3. **対象別詳細情報**

   グラフ上の特定ポイントにマウスを合わせると、その時点の正確な指標値と監視対象が表示されます。複数の対象がある場合は色で区別され、対象ごとにパターンが異なる場合はフィルタリング機能で個別分析が可能です。

4. **ピーク区間の特定**

   指標が急激に上昇する時間帯を確認し、問題が頻発する時間帯やパターンを把握できます。

> **シミュレーション活用のヒント**
>
> - **複数時間帯でのシミュレーション**
>
>   平日／週末、ピーク／非ピーク時間帯をそれぞれシミュレーションし、時間帯ごとの最適なしきい値を設定してください。

- **実際の障害期間を含める**

  過去に障害が発生した期間を含めてシミュレーションを実行し、現在のルールが障害を正しく検知できるか確認しましょう。

- **対象フィルタリングの追加**

  監視対象が多い場合、結果が複雑になります。フィルタリング機能を使って特定の対象のみを選択し、明確な結果を確認してください。

- **しきい値を段階的に調整**

  一度に大きく変更するのではなく、10 パーセント単位で段階的に調整しながら最適値を探すことをおすすめします。

イベント対象選択

イベント対象を指定しない場合、プロジェクト内のすべての対象に対してイベントが発生し、多数のイベントが生成される可能性があります。そのため、イベントを適用する対象を明確に設定することを推奨します。

イベント対象を変更すると、イベント件数が変わる場合があります。**シミュレーション** ボタンをクリックして再実行し、想定イベント件数を確認してください。

対象選択

イベントを適用する対象は、**選択入力** または **直接入力** の方法で指定できます。

- 選択入力 **直接入力**

1. タグ、演算子、値を選択または入力してください。
   a. タグが見つけにくい場合は、**メトリクス照会** を使ってタグを検索できます。
2. **+ 追加** をクリックして、条件（ `&&` , `||` ）でイベント対象を追加してください。

監視対象のタグ、演算子、値を直接入力してください。

- 直接入力を選択すると、入力欄の下に使用可能なタグ一覧（**タグ選択入力**）が表示されます。

```
ex. endsWith(okindName, 'example_name') && container == 'prod.billing'
ex. ${4xxErrorType} == '401'
```

> 入力方法を変更すると、作成中の内容は初期化されます。

基本情報および通知設定

イベント名、メッセージ、受信対象 を設定します。設定した内容はイベント履歴や実際の通知メッセージにそのまま表示されるため、識別しやすく分かりやすい名称にすることが重要です。

イベント有効化

設定した条件を満たした際にイベントを発生させるかどうかを設定します。

- **有効**: 設定した条件を満たすとイベントが発生します。
- **無効**: 条件を判定しません。

イベント名

イベントのタイトルとして使用するイベント名を入力してください。設定したイベント名は **イベント一覧** および **イベント履歴** メニューに表示され、この名前で検索することができます。

メッセージ

イベントメッセージを入力してください。入力したメッセージは、イベント履歴および通知に使用されます。 をクリックすると、過去に作成したメッセージ履歴を確認できます。メッセージには変数を使用して、イベント発生時点の実際の値を含めることができます。

- **変数の使用ルール**
  - 変数形式: `${指標名}`
  - 設定した **カテゴリと同一カテゴリ内** の指標のみを変数として使用できます。
  - 複数の変数を組み合わせて、より詳細なメッセージを作成できます。
  - 使用可能な変数一覧は **メトリクス照会** メニューで確認できます。

- **変数の使用例**

	イベント設定	イベント発生時
タイトル	Disk 使用量増加	Disk 使用量増加
メッセージ	`Disk = ${disk}%`	`Disk = 89.9%`

イベント受信

イベント通知を受信するメンバーを指定します。

- **全体受信** **タグ選択受信**

プロジェクトで通知設定が有効になっているすべての対象（メンバー、チャンネル）に通知が送信されます。

指定したタグを持つプロジェクトメンバーに通知が送信されます。

1. **タグ選択受信** を選択します。
2. **受信タグ** 項目で [+ タグ追加] または [+] ボタンをクリックします。
3. **イベント受信タグ** 画面の下にある **+ 新しいタグを作成** をクリックします。
4. **タグ作成** 画面でタグ名を入力し、色を選択した後、[タグ作成] ボタンをクリックしてタグを作成します。
5. 作成したタグは、タグ一覧の をクリックして修正または削除できます。
6. **イベント受信タグ** 画面の **タグ一覧** から希望するタグを選択すると適用されます。

> **受信タグ**
>
> 受信タグは、通知を受け取るユーザーやチャンネルを **グループ単位で管理するための機能** です。イベントに受信タグを設定すると、そのタグが付与されたユーザーにのみ通知が送信されます。受信タグを設定しない場合、プロジェクト内のすべてのユーザーに通知が送信されます。

イベント修正／削除

1. **警告アラート** > **イベント設定** の **メトリクス** タブに移動します。

2. イベント一覧で、修正または削除したいイベントの をクリックします。
3. **イベントルール修正** 画面でオプションを修正し、[保存] ボタンをクリックします。

   a. 選択したイベントを削除する場合は、**イベントルール修正** 画面右上の [削除] ボタンをクリックします。

イベント共有

メトリクスイベント設定を JSON ファイルとして保存し、他のユーザーと設定を共有したり、他のユーザーの設定を取り込んだりすることができます。

- JSON ファイル名: event-rules-`YYYYMMDD`.json

エクスポート

1. 画面右上の [**JSON**] ボタンをクリックします。
2. JSON 編集画面が表示されたら、[エクスポート] ボタンをクリックします。
   - イベントを検索した状態でエクスポート機能を使用すると、検索結果のみが JSON ファイルとしてダウンロードされます。
3. JSON ファイルがダウンロードされたら、共有したい他のユーザーに渡してください。

インポート

1. 画面右上の [ ] ボタンをクリックします。
2. **エクスポート** 機能でダウンロードした JSON ファイルを選択します。
3. JSON 編集画面が表示されたら、[一覧に追加] または [上書き] を選択します。

> 同じ種類の製品間での利用を推奨します。異なる製品のプロジェクトからイベント設定を取り込むことは可能ですが、正常に動作しない場合があります。

JSON 形式で編集する

1. 画面右上の [**JSON**] ボタンをクリックします。

2. 編集画面が表示されたら、JSON 形式に従って内容を修正してください。
3. 修正が完了したら、画面下部の [保存] ボタンをクリックします。

> ⓘ 修正した内容が JSON 形式に合っていない場合、画面下部にエラーメッセージが表示され、保存できません。表示されるエラーメッセージは、形式エラーの内容によって異なる場合があります。
>
> JSON error

JSON データ構造


```
{
  "version": 2,
  "eventId": "zcef7s7f27rum1",
  "enabled": true,
  "stateful": false,
  "title": "Active Transaction",
  "message": "Active Transaction = ${active_tx_count}",
  "category": "app_counter",
  "alertLabel": [
   "oid"
  ],
  "repeatCount": 1,
  "repeatDuration": 0,
  "silent": 300000,
  "receiver": [],
  "timeTag": [],
  "selectString": "",
  "conditions": [
   {
    "level": 20,
    "enabled": true,
    "rule": "active_tx_count > 100"
   }
  ],
  "createTime": 1763632674345,
  "lastModifiedTime": 1763632674347,
  "lastModifiedUser": "support@whatap.io",
  "basic": true,
  "metaId": "java004",
```

```
 "selectCondition":{
      "oid":[
       "-1010758404",
       "-250906941"
      ]
 }
}
```

表 | JSON データ構造

JSON フィールド	型	説明	備考
version	Integer	イベントルールのバージョン	変更制限
eventId	String	イベントルールの一意識別子	
enabled	boolean	イベントの有効／無効	
stateful	boolean	状態ベースイベントかどうか	
title	String	ユーザーが設定したイベントルール名	
message	String	ユーザーが設定したイベントルールメッセージ	
category	String	データカテゴリ	
alertLabel	List<String>	イベント状態管理のため、カテゴリごとに指定される基本識別子（プライマリキー）値	変更制限
repeatCount	Integer	繰り返し回数	
repeatDuration	Long	繰り返し期間	
silent	Integer	一時停止時間	
receiver	List<String>	受信タグのキー値一覧	

JSON フィールド	型	説明	備考
timeTag	List<String>	動作時間タグのキー値一覧	
selectString	String	対象選択	
conditions	List<Object>	レベル別の発生条件一覧	
ㄴ level	byte	レベル値	Critical: 30、Warning: 20、Info: 10
ㄴ enabled	boolean	レベルの有効／無効	
ㄴ rule	String	レベル発生条件	
createTime	Long	イベントルールの初回作成時刻	
lastModifiedTime	Long	イベントルールの最終修正時刻	
lastModifiedUser	String	イベントルールを最後に修正したアカウント	
basic	boolean	クイック設定で作成されたイベントかどうか	変更制限
metaId	String	クイック設定テンプレートの一意識別子	変更制限: この値はクイック設定で作成されたイベントにのみ存在します（basic=true）。
selectCondition	Object	クイック設定の対象選択値	この値はクイック設定で作成されたイベントにのみ存在します（basic=true）。

「変更制限」と表示されたフィールドはイベント動作に影響を与える可能性があるため、値を変更しないことを推奨します。

付録

発生条件・対象選択ガイド

メトリクスアラートにおけるイベントの発生条件とイベント対象の選択は、同一の構文を使用します。ただし、イベント発生条件ではフィールド（Field）のキーを変数として使用し、イベント対象の選択ではタグ（Tag）のキーを変数として使用します。

レベル動作ガイド

イベントレベルは、危険（Critical）、警告（Warning）、正常（Info）の3段階に分類されます。

1. **優先順位に基づくイベント発生**

   複数のレベル条件を同時に満たした場合、最も優先度の高いイベントのみが発生します。

   例

   ```
   設定:
   - Warning: CPU > 70%
   - Critical: CPU > 90%

   現在の状態: CPU 95%
   → 結果: Critical イベントのみ発生（Warning は抑制される）
   ```

2. **レベル上昇**

   進行中状態を持つイベントにおいて、低いレベルのイベントが進行中の状態で高いレベルの条件を満たすと、両方のレベルが進行中状態になります。

   例

   ```
   設定:
   - Warning: CPU > 70%
   - Critical: CPU > 90%

   シナリオ:
   ```

```
1) CPU 80% → Warning 発生（進行中）
2) CPU 95% → Warning（進行中）、Critical（進行中）
```

3. **レベル下降**

   進行中状態を持つイベントで、高いレベルから低いレベルへ遷移した場合、高いレベルは解消され、低いレベルが維持されます。

   例

```
設定:
- Warning: CPU > 70%
- Critical: CPU > 90%

シナリオ:
1) CPU 97% → Critical 発生
2) CPU 85% → Critical 解消、Warning 維持
3) CPU 65% → Warning 解消
```

4. **動作フロー**

```
- Warning: CPU > 70%
- Critical: CPU > 90%

メトリクス値: 60%  →  75%  →  92%  →  85%  →  60%
状態:     正常     Warning    Critical    Warning    正常
                （発生）      （発生）      （維持）      （解消）
                          Warning
                          （維持）
```

> ⓘ 進行中状態を持つイベントは、条件が満たされている間は継続して有効となり、条件が解除されるまで維持されます。

条件設定ガイド

メトリクス名が数字で始まる場合、または特殊文字を含む場合は、 `${メトリクス名}` の形式で記述する必要があります。

- 括弧 `()`, `[]`
- 演算子 `+`, `-`, `*`, `/`, `%`
- 区切り文字 `:`, `@`, `#`, `,`, (空白)
- その他の特殊文字 `!`, `^`, `&`, `|`, `~`, `` ` ``, `=`

`${}` **使用例:**

```
${cpu(xos)} > 50
${mem[0]} >= 100
${cpu-usage} > 80
${namespace:cpu} > 70
${metric name} > 60
${4xx_error} > 10
```

`${}` **なしで使用可能:** 英字、アンダースコア（ `_` ）、ドット（ `.` ）のみを含む場合

```
cpu > 80
cpu_usage > 50
CPUUtilization.Average > 0.8
```

1. **比較演算子**

   数値比較

```
より大きい:     cpu > 80
以上:          cpu >= 80
より小さい:     memory < 1000
以下:          memory <= 1000
等しい:        status == 200
等しくない:     error != 0
```

文字列比較

```
status == 'OK'
region == "us-east-1"
```

2. 算術演算子

- 基本的な算術演算

```
加算:    cpu + 10 >= 90
減算:    memory - 100 >= 500
乗算:    cpu * 2 >= 100
除算:    disk / 1024 >= 100
剰余:    value % 10 == 0
```

- 括弧による優先順位の制御

```
(cpu + memory) * 2 >= 200
(disk - used) / total >= 0.2
((cpu + memory) / 2) >= 50
```

- 負の値の表現

```
cpu > -100
```

3. 論理演算子

- AND 演算（&&）

```
cpu > 80 && memory > 1000
${cpu(xos)} > 50 && ${mem(xos)} > 60
```

- OR 演算（||）

```
cpu > 90 || memory > 90
disk < 10 || network > 1000
```

- 複合論理演算

```
(cpu > 50 && memory > 50) || disk < 20
cpu > 80 && (memory > 1000 || disk > 500)
```

4. パターンマッチング演算子

- LIKE 演算子

```
oname like 'prod-*'
url like '*error*'
message like 'WARN%'
```

- NOT LIKE 演算子

```
oname not like 'test-*'
url not like '*debug*'
```

5. 組み込み関数

- Null チェック関数

```
isNull(value)          # null かどうかを確認
isNotNull(value)       # null でないかを確認
nvl(value, 0)          # null の場合に既定値を返す
isEmpty(str)           # 空文字列かどうかを確認
isNotEmpty(str)        # 空文字列でないかを確認
```

- 集計関数

```
sum(cpu, memory, disk)          # 合計
avg(cpu, memory)                # 平均
max(cpu, memory, disk)          # 最大値
min(cpu, memory, disk)          # 最小値
count(value1, value2, value3)   # 件数
```

- 数学関数

```
round(cpu, 2)         # 四捨五入（小数点以下2桁）
```

- 文字列関数

```
length(str)           # 文字列の長さ
startsWith(str, 'prefix')  # 接頭辞の確認
endsWith(str, 'suffix')    # 接尾辞の確認
indexOf(str, 'search')     # 文字列の位置
substring(str, 0, 10)      # 部分文字列
trim(str)             # 空白の削除
replace(str, 'old', 'new') # 文字列の置換
hasStr(str, 'search')      # 文字列の包含判定
```

- 条件関数
  - **if**: 条件式が `true` の場合は `true値` を返し、 `false` の場合は `false値` を返す条件関数

    形式

```
if(条件, true値, false値)
```

    例

```
if(value >= 90, 'High', 'Medium') == 'High'
```

  - **decode**: 値と条件を比較し、一致した条件の結果を返します。一致しない場合は既定値を返します

    形式

```
decode(値, 条件1, 結果1, 条件2, 結果2, ..., 条件N, 結果N, 既定値)
```

    例

```
decode(value, 1, 'Low', 2, 'Medium', 3, 'High', 'Unknown') == 'Unknown'
```

  - **in**: 値が指定した候補値のいずれかと一致した場合は `true` 、それ以外は `false` を返します

形式

```
in(値, 候補値1, 候補値2, ..., 候補値N)
```

例

```
in(status, 200, 201, 204) == false
```

記述時の注意事項

- 推奨事項
  - 簡単な変数名は `${}` を使わずに記述（例: cpu, memory）
  - 特殊文字を含む場合は必ず `${}` を使用
  - 複雑な演算は括弧で明確に区切る
  - 関数名は正確に入力（大文字と小文字を区別）
- 避けるべき事項
  - 不完全な式の記述（例: `cpu >`, `memory &&` ）
  - 括弧の不一致（例: `(cpu > 80 && cpu > 70)` ）
  - 演算子の連続使用(例: `cpu >> 80`, `cpu >< 80`)

複合メトリクス

ホーム > プロジェクト選択 > **アラート通知** > **イベント設定** > **複合指標** タブ

複雑なルールを利用してイベントを作成し、警告通知を送信できます。

基本オプション

- **検索**: イベント名またはイベント受信タグを選択して、目的のイベントを検索します
- **JSON ダウンロード／アップロード**: JSON エディタでイベントルールを直接編集するか、ファイルとしてダウンロードして編集後に再アップロードできます
- **通知メッセージ設定**: 受信するメッセージの詳細項目を選択し、項目ごとに内容を編集します
- **+ イベント追加**: 新しい複合メトリクスイベントを追加します

> **権限**
>
> **複合指標** イベントを設定するには **アラート設定** 権限が必要です。この権限は **ホーム > プロジェクト選択 > 管理 > 統合メンバー管理** で、対象メンバーの をクリックして付与できます。

表 | 複合メトリクスイベント一覧の構成

項目	説明
イベントタイトル	ユーザーが指定したイベント名 - 最大 255 バイトまで入力可能
条件	イベント発生条件
イベント状態が解消された場合の追加通知	Critical および Warning レベルのイベントが解消された際に、RECOVERED（正常）状態の通知を受信するかどうか – 有効: 進行中 → 正常状態に切り替わると RECOVERED 通知を送信し、イベント履歴で進行中／解消状態を区別して確認できます – 無効: しきい値を超えるたびに単発のイベントが発生します

項目	説明
イベント受信タグ	該当イベントの通知を受信するユーザーまたはグループ
	イベントの修正および削除
	イベントの有効／無効状態 – 有効: 条件を満たすとイベントが発生 – 無効: イベントは動作せず、条件も評価されません

イベント設定

複合指標 イベントは、メトリクスデータのクエリ言語である **MXQL** を基にイベント条件を作成します。**チャートから作成** 機能では、**MXQL** の自動補完を支援するコンボボックス機能を提供します。

イベント追加

1. **アラート通知** > **イベント設定** から **複合指標** タブをクリックします。
2. 画面右側の [+ イベント追加] ボタンをクリックします。
3. **複合指標** 画面で **チャートから作成** をクリックします。

   a. 提供されているテンプレートを利用できます。
4. **イベント設定** 画面で、先にイベントデータを取得し、その後イベント条件を設定します。

イベントデータ照会

イベントデータを照会してチャートを構成します。**イベント設定** 画面右側の **イベントデータ照会** で、**ウィジェット** または **テキスト** オプションを選択してイベントを設定します。

- **ウィジェット** **テキスト**

時系列チャートを構成するオプションを使用して、イベント設定に利用する **MXQL** を自動補完できます。

1. データを照会する日付を選択します。
2. データを照会する **カテゴリ**（対象）を選択します。

3. **フィルター** 項目の **+ フィルター追加** をクリックしてイベント条件を作成します。

   a. フィルター画面で、演算式、タグ、フィルター値を選択します。
   b. **+ 追加** をクリックして条件を追加できます。
   c. 条件を削除する場合は、条件の横にある – をクリックします。

4. **グループ化** 項目で、グループ化されたメトリクスデータを選択します。複数選択が可能です。
5. **タイムユニット** 項目で、`秒`、`分`、`時間` 単位のいずれかを選択し、データを分割する時間基準を設定します。
6. **フィールド** 項目で、イベント発生条件に使用するフィールドを選択します。複数選択が可能です。

> テキストモードで入力した MXQL は、ウィジェットモードと互換性がありません。

MXQL を直接編集できるエディタが表示されます。

1. **MXQL** を直接修正し、**照会** ボタンをクリックします。
2. 日付を選択すると、該当内容をチャートまたはグラフで確認できます。

通知

警告通知設定の基本情報を入力します。

- **イベント有効化**: トグルボタンを有効にすると、イベントが動作します。
- **レベル**: イベントレベル（Critical、Warning、Info）を選択します。
  - **イベント状態が解消された場合の追加通知**: トグルを有効にすると、イベントが解消され、正常（RECOVERED）状態の通知が送信されます。
- **タイトル**: イベントルールのタイトルを入力します。
- **メッセージ**: イベント発生時に受信する通知メッセージの内容を入力します。メッセージ内で変数を使用すると、イベント発生時点の実際の値を含めることができます。

  Message example

> **変数使用ルール**

- 変数形式: `${メトリクス名}`
- 設定した **カテゴリと同一カテゴリ** 内のメトリクスのみ変数として使用できます。
- 複数の変数を組み合わせて、詳細なメッセージを構成できます。
- 使用可能な変数一覧は **メトリクス照会** で確認できます。

イベント発行条件

- **データ照会範囲**: リアルタイムデータの照会範囲を選択します。
  - 選択可能な範囲: `30秒` , `1分` , `2分` , `5分` , `10分` , `30分` , `1時間` , `2時間` , `3時間`
- **条件**: イベント発行条件を入力します。イベントデータ照会に含まれるフィールドのみ使用できます。

付加情報

- **インターバル**: 選択した時間間隔（ `1分` , `5分` , `10分` , `30分` , `1時間` , `12時間` ）ごとにアラート条件を確認します。
- **サイレンス**: イベント発生後、選択した時間（ `unset` , `1分` , `5分` , `10分` , `30分` , `1時間` , `12時間` ）の間、同一イベントは発生しません。
- **イベント動作時間**: 勤務時間、非勤務時間、点検時間など、特定の時間帯にイベントが動作するようタグを設定します。タグを設定しない場合、イベントは有効な状態で 24 時間常時動作します。

  1. **+ 追加** をクリックします。
  2. **イベント動作時間** で **+ 新しいタグ作成** をクリックします。
  3. **イベント動作時間タグ作成** で **タグ名、曜日、時間、色** を選択し、[適用] ボタンをクリックします。
  4. 作成したタグはタグ一覧で確認でき、チェックボックスを選択すると適用されます。
  5. 修正または削除するタグの をクリックし、**イベント動作時間タグ修正** で編集または削除できます。

  タグを削除すると、そのタグが適用されているすべてのユーザーから削除されます。ただし、イベントルールで使用中のタグは削除できません。

- **イベント受信タグ**: 受信タグを設定すると、該当タグを持つプロジェクトメンバーに通知が送信されます。受信タグを設定しない場合、プロジェクト内のすべてのユーザーに通知が送信されます。

1. **+ タグ追加** または + をクリックします。
2. **イベント受信タグ** 画面下部の **+ 新しいタグ作成** をクリックします。
3. **タグ作成** 画面でタグ名を入力し、色を選択した後、[タグ作成] ボタンをクリックします。
4. 作成したタグは、タグ一覧の をクリックして修正または削除できます。
5. **イベント受信タグ** 画面の **タグ一覧** から適用したいタグを選択します。

> イベント設定時にイベント受信タグを選択すると、該当タグを持つプロジェクトメンバーおよび 3rd-party プラグインに通知を送信できます。**アラート通知** > **通知設定** で、それぞれにタグを割り当てることができます。

イベントルールテスト

設定した条件を過去データに基づいてテストし、どの程度のイベントが発生するかを予測できます。

1. イベントルールをテストする日付を選択し、[確認] ボタンをクリックします。
2. [実行] ボタンをクリックして、イベント発生件数を確認します。

イベント修正／削除

1. **アラート通知** > **通知設定** の **複合指標** タブに移動します。
2. 複合メトリクス一覧で、修正または削除したいイベントの をクリックします。
3. **通知設定** 画面でオプションを修正し、[保存] ボタンをクリックします。

   a. 選択したイベントを削除する場合は、**通知設定** 画面右上の [削除] ボタンをクリックします。

イベント共有

メトリクスイベント設定を JSON ファイルとして保存し、他のユーザーと設定を共有したり、他のユーザーの設定を取り込んだりすることができます。

- JSON ファイル名: event-rules- `YYYYMMDD` .json

エクスポート

1. 画面右上の [**JSON**] ボタンをクリックします。
2. JSON 編集画面が表示されたら、[エクスポート] ボタンをクリックします。
   - イベントを検索した状態でエクスポート機能を使用すると、検索結果のみが JSON ファイルとしてダウンロードされます。
3. JSON ファイルがダウンロードされたら、共有したい他のユーザーに渡してください。

インポート

1. 画面右上の [ ] ボタンをクリックします。
2. **エクスポート** 機能でダウンロードした JSON ファイルを選択します。
3. JSON 編集画面が表示されたら、[一覧に追加] または [上書き] を選択します。

同じ種類の製品間での利用を推奨します。異なる製品のプロジェクトからイベント設定を取り込むことは可能ですが、正常に動作しない場合があります。

JSON 形式で編集する

1. 画面右上の [**JSON**] ボタンをクリックします。
2. 編集画面が表示されたら、JSON 形式に従って内容を修正してください。
3. 修正が完了したら、画面下部の [保存] ボタンをクリックします。

修正した内容が JSON 形式に合っていない場合、画面下部にエラーメッセージが表示され、保存できません。表示されるエラーメッセージは、形式エラーの内容によって異なる場合があります。

JSON error

通知メッセージ設定

受信するメッセージの詳細項目を選択し、項目ごとに内容を編集できます。

付録

MXQL 文法ガイド

複合指標 は MXQL クエリを基に動作します。詳細な文法については、MXQL 文法ガイド を参照してください。

ヒットマップパターン

ホーム > プロジェクト選択 > **アラート通知** > **イベント設定** > **ヒットマップパターン** タブ

ヒットマップパターン は、レスポンスタイム分布や呼び出し密度など、アプリケーション実行区間の時間帯別パターン変化を基に異常を自動検知する機能です。

複雑な条件やしきい値を手動で設定しなくても、システムがリアルタイムで収集したヒットマップを分析し、通常とは異なるパターンを検出すると自動的に警告通知を発生させます。ヒットマップパターンアラートは、次のような状況で特に有効です。

- 特定のトランザクションが、通常よりも特定の時間帯に過度に集中する場合
- トラフィック分布やレスポンスパターンが通常の状態から変化した場合
- 条件設定だけでは捉えにくい、パターンベースの異常値が必要な場合

イベント設定

検知するイベントパターンのオプションを設定します。

有効化

トグルをオンにすると、該当パターンが検知された際に警告通知が送信されます。

- **有効:** 条件を満たすとイベントが発生します。
- **無効:** イベントは発生せず、条件も評価されません。

サイレンス

イベント発生後、選択した時間の間はイベントが発生せず、**イベント履歴**にも記録されません。

イベント受信タグ

受信タグを設定すると、そのタグを持つプロジェクトメンバーに通知が送信されます。受信タグを設定しない場合、プロジェクト内のすべてのユーザーに通知が送信されます。

1. **+ タグ追加** または + をクリックします。
2. **イベント受信タグ** 画面下部の **+ 新しいタグ作成** をクリックします。
3. **タグ作成** 画面でタグ名を入力し、色を選択した後、[タグ作成] ボタンをクリックします。
4. 作成したタグは、タグ一覧の をクリックして修正または削除します。
5. **イベント受信タグ** 画面の **タグ一覧** から適用したいタグを選択します。

イベント動作時間

勤務時間、非勤務時間、点検時間など、特定の時間帯にイベントが動作するようタグを設定します。タグを設定しない場合、イベントは有効な状態で 24 時間常時動作します。

1. **+ 追加** をクリックします。
2. **イベント動作時間** で **+ 新しいタグ作成** をクリックします。
3. **イベント動作時間タグ作成** で **タグ名、曜日、時間、色** を選択し、[適用] ボタンをクリックします。
4. 作成したタグはタグ一覧で確認でき、チェックボックスを選択すると適用されます。
5. 修正または削除するタグの をクリックし、**イベント動作時間タグ修正** で編集または削除できます。

> タグを削除すると、そのタグが適用されているすべてのユーザーから削除されます。ただし、イベントルールで使用中のタグは削除できません。

ヒットマップパターン種別

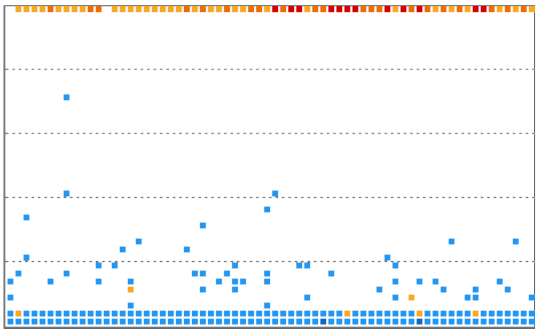
各パターンのトグルをオンにすると、該当パターンが検知された際にイベントが発生します。

ヒットマップ横ラインパターン

複数のトランザクションが、ほぼ同じ応答時間帯で一斉に終了するパターンです。特定の応答時間帯に集中が発生しており、サービス遅延の増加や外部リソースの応答不安定を示します。

- 外部 API、DB、HTTPS 呼び出しでのタイムアウトまたは遅延発生
- 特定リソースを使用しようとしたリクエストが、一定時間待機した後に一括処理される場合

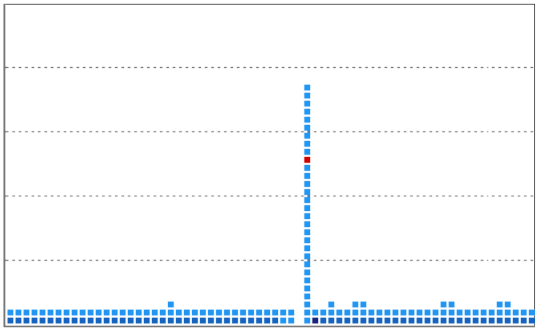
- 共通の I/O またはネットワーク区間で繰り返し遅延が発生している場合



ヒットマップ縦ラインパターン

トランザクションの開始時点は異なるものの、同じタイミングで同時に終了するパターンです。複数のリクエストが特定の瞬間にまとめて処理され、共有リソースのボトルネックを示します。

- DB ロックやファイルロックなど、共有リソースでの待機時間増加
- キューやスレッドプールにおける同時処理遅延
- GC ポーズなどのシステム停止により、一時的にリクエストが止まり、その後同時に再開される状況

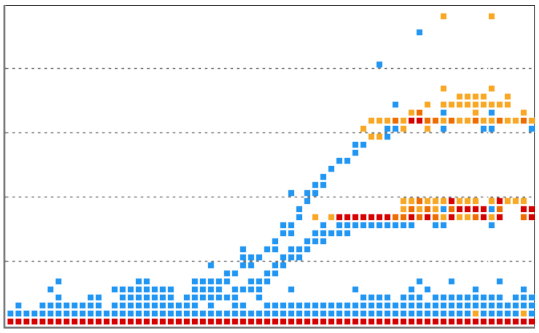


ヒットマップ縦ラインパターンは、ロック競合やボトルネックリソースの状態確認が必要であることを示すサインです。

ヒットマップフライングパターン

特定のリソースやログなどの共通リソース不足により、応答時間が一定間隔で波のように上下するパターンです。

- GC、コネクションプール、ファイルハンドルなどの共通リソース不足
- ログフラッシュやストレージ I/O など、周期的な遅延要因の存在
- バッチ処理やバックグラウンドジョブによる周期的なリソース占有

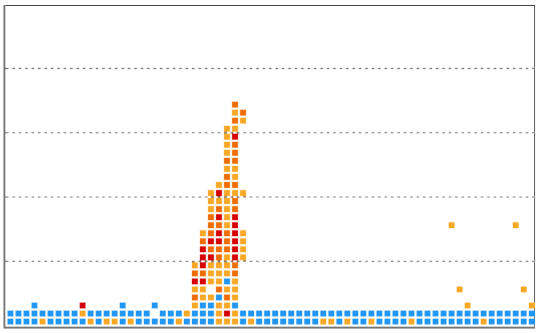


ヒットマップフライングパターンは、周期的なボトルネックやリソース不足の可能性が高いことを示します。

ヒットマップ過負荷パターン

特定の応答時間帯にトランザクションが異常に集中するパターンです。一時的な負荷増大や障害の前兆により、リクエストが正常に分散されず、特定の区間に偏る場合に発生します。

- 一時的な障害や遅延によりトランザクションがキューに溜まり、一括処理されるケース
- バックエンドサービスの応答遅延
- 特定の時間帯におけるトラフィック集中による処理余力不足

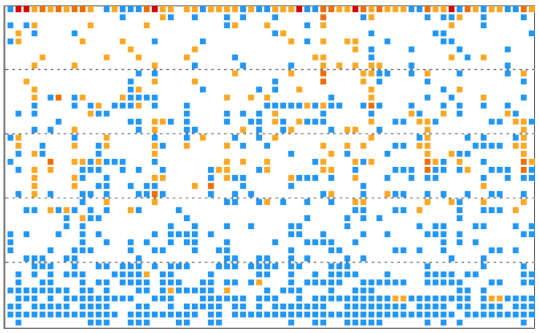


過負荷パターンは、一時的なサービス品質低下や負荷分散失敗を示します。

ヒットマップ暴走パターン

過剰なリクエストや負荷によって、全体的な応答時間が大きく増加するパターンです。サービス全体でパフォーマンス低下が発生している状態を示します。

- ユーザートラフィックや自動化トラフィックの急増
- CPU、メモリ、GC などのシステムリソース不足
- バックエンドのボトルネックによる全体的な応答品質の悪化



暴走パターンは、即時の点検が必要な状態です。

異常値

ホーム > プロジェクト選択 > **アラート通知** > **イベント設定** > **異常検知** タブ

異常検知 は、メトリクスデータから異常な上昇または下降パターンを検出し、通常とは異なる動きが検知された場合にイベントを発生させる機能です。

基本オプション

- **検索**: イベント名またはイベント受信タグを選択して、目的のイベントを検索します。
- **+ イベントルール追加**: 新しい異常値イベントを追加します。

表 | 異常値イベント一覧の構成

項目	説明
イベントメッセージ	イベント発生時に送信されるメッセージ
カテゴリ	メトリクスデータを分類する単位で、メトリクスイベント設定の基準となるカテゴリ
フィールド	異常を検知する指標（メトリクスフィールド）
フィルター	異常値の対象となるデータ範囲を制限する条件
上昇／下降パターン	上昇または下降パターンの感度
イベント受信タグ	該当イベントの通知を受信するユーザーまたはグループ
	イベント設定の修正
	イベントの削除
	イベントの有効／無効状態 - 有効: 条件を満たすとイベントが発生 - 無効: イベントは動作せず、条件も評価されません

イベント設定

メトリクスデータを基に、値の動きが通常とは異なり上昇または下降した場合に、警告通知を送信するよう設定できます。

イベントルール追加

1. 画面右側の [+ イベントルール追加] ボタンをクリックします。
2. **イベント基本情報**, **メトリクス**, **異常値**, **付加情報** を順に設定します。

イベント基本情報

1. イベントレベル（Critical, Warning, Info）を選択します。
   - イベントの重要度および通知レベルが決定されます。
2. イベント発生時に受信する通知メッセージを入力します。
   - メッセージ内で変数を使用すると、イベント発生時点の実際の値を含めることができます。

> **使用可能な変数**
>
> - `$pcode` : プロジェクトコード
> - `$category` : カテゴリ
> - `$field` : フィールド
> - `$value` : 発生値
> - `$oname` : イベント対象

メトリクス

1. **カテゴリ** を選択します。
   - 異常を検知するメトリクスデータの基準となるカテゴリです。

2. イベント発生条件に使用する **フィールド** を選択します。複数選択が可能です。
3. イベント対象を絞り込むためのフィルターを選択します。複数選択が可能です。

異常値

1. **上昇パターン** と **下降パターン** の感度を選択します。
2. 各パターンのトグルボタンをクリックして、有効または無効を切り替えます。
   - 上昇パターンのみを検知する場合: 上昇パターン **ON**, 下降パターン **OFF**
   - 下降パターンのみを検知する場合: 下降パターン **ON**, 上昇パターン **OFF**

付加情報

イベント動作時間

勤務時間、非勤務時間、点検時間など、特定の時間帯にイベントが動作するようタグを設定します。タグを設定しない場合、イベントは有効な状態で 24 時間常時動作します。

1. **+ 追加** をクリックします。
2. **イベント動作時間** で **+ 新しいタグ作成** をクリックします。
3. **イベント動作時間タグ作成** で **タグ名、曜日、時間、色** を選択し、[適用] ボタンをクリックします。
4. 作成したタグはタグ一覧で確認でき、チェックボックスを選択すると適用されます。
5. 修正または削除するタグの をクリックし、**イベント動作時間タグ修正** で編集または削除できます。

> タグを削除すると、そのタグが適用されているすべてのユーザーから削除されます。ただし、イベントルールで使用中のタグは削除できません。

イベント受信タグ

イベント受信タグを設定すると、該当タグを持つプロジェクトメンバーに通知が送信されます。受信タグを設定しない場合、プロジェクト内のすべてのユーザーに通知が送信されます。

1. **+ タグ追加** または **+** をクリックします。
2. **イベント受信タグ** 画面下部の **+ 新しいタグ作成** をクリックします。

3. **タグ作成** 画面でタグ名を入力し、色を選択した後、[タグ作成] ボタンをクリックします。
4. 作成したタグは、タグ一覧の をクリックして修正または削除できます。
5. **イベント受信タグ** 画面の **タグ一覧** から適用したいタグを選択します。

> イベント設定時にイベント受信タグを選択すると、該当タグを持つプロジェクトメンバーおよび 3rd-party プラグインに通知を送信できます。**アラート通知** > **通知設定** で、それぞれにタグを割り当てることができます。

イベント修正

1. **アラート通知** > **イベント設定** の **異常検知** タブに移動します。
2. **異常値** 一覧で、修正するイベントの右側にある をクリックします。
3. **異常値** 画面でオプションを修正し、[保存] ボタンをクリックします。

イベント削除

1. **アラート通知** > **イベント設定** の **異常検知** タブに移動します。
2. **異常値** 一覧で、削除するイベントの右側にある をクリックします。
3. 削除確認画面で [削除] ボタンをクリックします。

リアルタイムログイベント

ホーム > プロジェクト選択 > **アラート通知** > **イベント設定** > **リアルタイムログ** タブ

収集したログデータを条件に合わせてフィルタリングして警告通知を設定できます。ログイベントを使用するには、ログモニタリングを有効化する必要があります。**ログモニタリングの有効化**については、各製品のログドキュメントを参照してください。

- **リアルタイムログイベント**: リアルタイムで収集したログで、設定したイベント条件が満たされるとイベントが発生します。

基本オプション

- **検索**: イベント名またはイベント受信タグを選択して目的のイベントを検索
- **JSONダウンロード/アップロード**: イベントルールはJSONエディタで直接編集するか、ファイルにダウンロードして修正後に再アップロード可能
- **通知メッセージ設定**: 受信するメッセージの詳細項目を選択し、項目ごとに内容を編集
- **+ イベント追加**: 新しいログイベントを追加

表 | リアルタイムログイベント一覧構成

項目	説明
イベント名	通知タイトルとして使用されるイベント名 - 最大255バイトまで入力可能
カテゴリ	ログ分類名（ログフォルダ名）
検索キー	ログデータで検索したい項目 - 例: HTTPレスポンスステータスコードの値にアクセスする場合、検索キー `status`
検索値	検索キーに対応する実際のデータ。ログに入力した文字列が含まれる場合に警告通知を送信 - 例: 検索キー `status`、検索値 `200` を設定した場合、HTTPステータスコード200を含むログデータ収集時に警告通知が発生

項目	説明
イベント発生一時停止	イベント通知発生後、同じイベントの発生を一定時間停止 - 選択可能時間: 使用しない、5分、10分、15分、20分、30分、1時間、2時間、3時間、6時間、12時間、1日 - 解消通知が有効な場合、正常（RECOVERED）状態の通知受信後、選択した時間中は同じイベントが発生しても通知されない
イベント対象フィルタリング	イベントが適用される対象の条件 - 対象条件を設定しない場合、収集されるすべてのデータに対して通知を判断 - 選択入力: タグ、演算子、値を選択して条件を構成 - 直接入力: 条件を直接入力して対象を指定
イベント受信タグ	イベント通知を受け取るユーザーまたはグループ（受信タグ）
	イベントの編集と削除
	イベントの有効化状態 - 有効: 条件を満たすとイベント発生 - 無効: イベントが動作せず、条件をチェックしない

イベント設定

ログイベントは、リアルタイムで収集されるログ、または特定条件を満たすログが一定数蓄積された際に警告通知を設定できる機能です。

リアルタイムログイベントの追加

1. **アラート通知** > **イベント設定**の**リアルタイムログ**タブをクリックしてください。
2. 画面右側の [+ イベント追加] ボタンをクリックしてください。
3. **イベントルール追加**ウィンドウでログイベント条件を設定してください。
    - **イベント名**: イベントタイトルとして使用する名前を入力してください。

- **イベント有効化**: トグルボタンを有効にすると、条件を満たしたときにイベントが発生します。
- **レベル**: イベントレベル（Critical、Warning、Info）を選択してください。
- **メッセージ**: イベント発生時に受け取る通知メッセージを入力してください。メッセージに変数を使用して、イベント発生時点の実際の値を含めることができます。

  > **変数使用ルール**
  >
  > - 変数形式: `${指標名}`
  > - 設定した**カテゴリと同じカテゴリ**内の指標のみ変数として使用できます。
  > - 複数の変数を組み合わせて詳細なメッセージを構成できます。
  > - 使用可能な変数一覧は**メトリクス照会**で確認できます。

- **カテゴリ**: ログ分類名（ログフォルダ名）をリストから選択するか、直接入力してください。カテゴリを選択してから検索キーと検索値を入力できます。
- **検索キー**: ログデータで検索したい項目を入力してください。
  - 例: HTTPレスポンスステータスコードの値にアクセスする場合、**検索キー** `status`
- **検索値**: **検索キー**に対応する実際のデータを入力してください。ログに入力した文字列が含まれる場合に警告通知を送信します。
  - 例: **検索キー** `status` 、**検索値** `200` を設定した場合、HTTPステータスコード200を含むログデータ収集時に警告通知が発生
- **イベント対象フィルタリング**: イベントが適用される対象の条件を入力してください。入力値がない場合、収集されるすべてのデータに対して通知を判断します。
  - **選択入力**: タグ、演算子、値を選択して対象条件を設定してください。

    > 条件式の記述方法とサポートされる演算子については、条件設定ガイドを参照してください。

  - **直接入力**: 条件を直接入力してください。
- **イベント発生一時停止**: 通知を一時停止する時間を選択してください。イベント発生後、設定した時間中は同じイベントが発生しません。

    - 選択可能時間: `使用しない`、`1分`、`2分`、`3分`、`5分`、`10分`、`20分`、`30分`、`1時間`、`3時間`、`6時間`、`12時間`

- **イベント動作時間**: 特定の時間帯（業務/非業務/メンテナンス時間など）にイベントが動作するようにタグを設定してください。タグを設定しない場合、イベントが有効な状態で24時間常時動作します。

    1. **+ 追加**をクリックしてください。
    2. **イベント動作時間**で**+新しいタグを作成**をクリックしてください。
    3. **イベント動作時間タグの作成**で**タグ名、曜日、時間、色**を選択し、[適用] ボタンをクリックしてください。
    4. 作成したタグはタグリストで確認でき、チェックボックスを選択すると適用されます。
    5. 変更または削除するタグの アイコンをクリックして、**イベント動作時間タグの編集**でタグを編集または削除できます。

    > タグを削除すると、そのタグが適用されているすべてのユーザーのタグが削除されます。ただし、イベントルールで使用中のタグは削除できません。

- **イベント受信タグ**: イベントに受信タグを設定すると、そのタグを持つプロジェクトメンバーに通知が送信されます。受信タグを設定しない場合、プロジェクト内のすべてのユーザーに通知が送信されます。

    1. **+ タグ追加**または+をクリックしてください。
    2. **イベント受信タグ**ウィンドウ下部の**+ 新しいタグを作成**をクリックしてください。
    3. **タグ作成**ウィンドウでタグ名を入力し、色を選択してから、[タグ作成] ボタンをクリックしてタグを作成してください。
    4. 作成されたタグはタグリストの アイコンをクリックして編集または削除してください。
    5. **イベント受信タグ**ウィンドウの**タグリスト**から目的のタグを選択すると適用されます。

    > イベント設定時にイベント受信タグを選択して、そのタグを持つプロジェクトメンバーとサードパーティプラグインに通知を送信できます。**アラート通知 > 通知設定**でプロジェクトメンバーとサードパーティプラグインにそれぞれタグを指定できます。

4. **イベントルール追加**ウィンドウで設定が完了したら、[保存] ボタンをクリックしてください。

イベントの編集/削除

1. **アラート通知** > **イベント設定**の**リアルタイムログ**タブに移動してください。
2. ログイベントリストで編集または削除するイベントの アイコンをクリックしてください。
3. **イベント設定**画面でオプションを変更し、[保存] ボタンをクリックしてください。

   a. 選択したイベントを削除するには、**イベント設定**画面の右上の [削除] ボタンをクリックしてください。

イベントの共有

メトリクスイベント設定を JSON ファイルとして保存し、他のユーザーと設定を共有したり、他のユーザーの設定を取り込んだりすることができます。

- JSON ファイル名: event-rules-`YYYYMMDD`.json

エクスポート

1. 画面右上の [**JSON**] ボタンをクリックします。
2. JSON 編集画面が表示されたら、[エクスポート] ボタンをクリックします。
   - イベントを検索した状態でエクスポート機能を使用すると、検索結果のみが JSON ファイルとしてダウンロードされます。
3. JSON ファイルがダウンロードされたら、共有したい他のユーザーに渡してください。

インポート

1. 画面右上の [ ] ボタンをクリックします。
2. **エクスポート** 機能でダウンロードした JSON ファイルを選択します。
3. JSON 編集画面が表示されたら、[一覧に追加] または [上書き] を選択します。

> 同じ種類の製品間での利用を推奨します。異なる製品のプロジェクトからイベント設定を取り込むことは可能ですが、正常に動作しない場合があります。

JSON 形式で編集する

1. 画面右上の [**JSON**] ボタンをクリックします。
2. 編集画面が表示されたら、JSON 形式に従って内容を修正してください。
3. 修正が完了したら、画面下部の [保存] ボタンをクリックします。

> 修正した内容が JSON 形式に合っていない場合、画面下部にエラーメッセージが表示され、保存できません。表示されるエラーメッセージは、形式エラーの内容によって異なる場合があります。
>
> JSON error

付録

発生条件・対象選択ガイド

メトリクスアラートにおけるイベントの発生条件とイベント対象の選択は、同一の構文を使用します。ただし、イベント発生条件ではフィールド（Field）のキーを変数として使用し、イベント対象の選択ではタグ（Tag）のキーを変数として使用します。

レベル動作ガイド

イベントレベルは、危険（Critical）、警告（Warning）、正常（Info）の3段階に分類されます。

1. **優先順位に基づくイベント発生**

   複数のレベル条件を同時に満たした場合、最も優先度の高いイベントのみが発生します。

   例

```
設定:
- Warning: CPU > 70%
- Critical: CPU > 90%
```

```
現在の状態: CPU 95%
→ 結果: Critical イベントのみ発生（Warning は抑制される）
```

2. **レベル上昇**

   進行中状態を持つイベントにおいて、低いレベルのイベントが進行中の状態で高いレベルの条件を満たすと、両方のレベルが進行中状態になります。

   例

```
設定:
- Warning: CPU > 70%
- Critical: CPU > 90%

シナリオ:
1) CPU 80% → Warning 発生（進行中）
2) CPU 95% → Warning（進行中）、Critical（進行中）
```

3. **レベル下降**

   進行中状態を持つイベントで、高いレベルから低いレベルへ遷移した場合、高いレベルは解消され、低いレベルが維持されます。

   例

```
設定:
- Warning: CPU > 70%
- Critical: CPU > 90%

シナリオ:
1) CPU 97% → Critical 発生
2) CPU 85% → Critical 解消、Warning 維持
3) CPU 65% → Warning 解消
```

4. **動作フロー**

```
- Warning: CPU > 70%
- Critical: CPU > 90%
```

```
メトリクス値: 60%  →  75%  →  92%  →  85%  →  60%
状態:     正常    Warning   Critical   Warning   正常
                （発生）    （発生）    （維持）    （解消）
                        Warning
                          （維持）
```

進行中状態を持つイベントは、条件が満たされている間は継続して有効となり、条件が解除されるまで維持されます。

条件設定ガイド

メトリクス名が数字で始まる場合、または特殊文字を含む場合は、`${メトリクス名}` の形式で記述する必要があります。

- 括弧 `()`, `[]`
- 演算子 `+`, `-`, `*`, `/`, `%`
- 区切り文字 `:`, `@`, `#`, `,`, (空白)
- その他の特殊文字 `!`, `^`, `&`, `|`, `~`, `` ` ``, `=`

`${}` **使用例:**

```
${cpu(xos)} > 50
${mem[0]} >= 100
${cpu-usage} > 80
${namespace:cpu} > 70
${metric name} > 60
${4xx_error} > 10
```

`${}` **なしで使用可能:** 英字、アンダースコア（`_`）、ドット（`.`）のみを含む場合

```
cpu > 80
cpu_usage > 50
CPUUtilization.Average > 0.8
```

1. 比較演算子

数値比較

```
より大きい:      cpu > 80
以上:           cpu >= 80
より小さい:      memory < 1000
以下:           memory <= 1000
等しい:          status == 200
等しくない:      error != 0
```

文字列比較

```
status == 'OK'
region == "us-east-1"
```

2. 算術演算子

- 基本的な算術演算

```
加算:      cpu + 10 >= 90
減算:      memory - 100 >= 500
乗算:      cpu * 2 >= 100
除算:      disk / 1024 >= 100
剰余:      value % 10 == 0
```

- 括弧による優先順位の制御

```
(cpu + memory) * 2 >= 200
(disk - used) / total >= 0.2
((cpu + memory) / 2) >= 50
```

- 負の値の表現

```
cpu > -100
```

3. 論理演算子

- AND 演算（&&）

```
cpu > 80 && memory > 1000
${cpu(xos)} > 50 && ${mem(xos)} > 60
```

- OR 演算（||）

```
cpu > 90 || memory > 90
disk < 10 || network > 1000
```

- 複合論理演算

```
(cpu > 50 && memory > 50) || disk < 20
cpu > 80 && (memory > 1000 || disk > 500)
```

4. パターンマッチング演算子

- LIKE 演算子

```
oname like 'prod-*'
url like '*error*'
message like 'WARN%'
```

- NOT LIKE 演算子

```
oname not like 'test-*'
url not like '*debug*'
```

5. 組み込み関数

- Null チェック関数

```
isNull(value)          # null かどうかを確認
isNotNull(value)       # null でないかを確認
nvl(value, 0)          # null の場合に既定値を返す
isEmpty(str)           # 空文字列かどうかを確認
isNotEmpty(str)        # 空文字列でないかを確認
```

- 集計関数

```
sum(cpu, memory, disk)          # 合計
avg(cpu, memory)                # 平均
max(cpu, memory, disk)          # 最大値
min(cpu, memory, disk)          # 最小値
count(value1, value2, value3)     # 件数
```

- 数学関数

```
round(cpu, 2)         # 四捨五入（小数点以下2桁）
```

- 文字列関数

```
length(str)            # 文字列の長さ
startsWith(str, 'prefix')  # 接頭辞の確認
endsWith(str, 'suffix')    # 接尾辞の確認
indexOf(str, 'search')    # 文字列の位置
substring(str, 0, 10)     # 部分文字列
trim(str)              # 空白の削除
replace(str, 'old', 'new') # 文字列の置換
hasStr(str, 'search')     # 文字列の包含判定
```

- 条件関数
  - **if**: 条件式が `true` の場合は `true値` を返し、 `false` の場合は `false値` を返す条件関数

    形式

    ```
    if(条件, true値, false値)
    ```

    例

    ```
    if(value >= 90, 'High', 'Medium') == 'High'
    ```

  - **decode**: 値と条件を比較し、一致した条件の結果を返します。一致しない場合は既定値を返します

    形式

```
decode(値, 条件1, 結果1, 条件2, 結果2, ..., 条件N, 結果N, 既定値)
```

例

```
decode(value, 1, 'Low', 2, 'Medium', 3, 'High', 'Unknown') == 'Unknown'
```

- **in**: 値が指定した候補値のいずれかと一致した場合は `true`、それ以外は `false` を返します

形式

```
in(値, 候補値1, 候補値2, ..., 候補値N)
```

例

```
in(status, 200, 201, 204) == false
```

記述時の注意事項

- 推奨事項
  - 簡単な変数名は `${}` を使わずに記述（例: cpu, memory）
  - 特殊文字を含む場合は必ず `${}` を使用
  - 複雑な演算は括弧で明確に区切る
  - 関数名は正確に入力（大文字と小文字を区別）
- 避けるべき事項
  - 不完全な式の記述（例: `cpu >`, `memory &&` ）
  - 括弧の不一致（例: `(cpu > 80 && cpu > 70)` ）
  - 演算子の連続使用(例: `cpu >> 80`, `cpu >< 80`)

複合ログイベント設定

ホーム > プロジェクト選択 > **アラート通知** > **イベント設定** > **複合ログ** タブ

収集したログデータを条件に合わせてフィルタリングして警告通知を設定できます。ログイベントを使用するには、ログモニタリングを有効化する必要があります。**ログモニタリングの有効化**については、各製品のログドキュメントを参照してください。

- **複合ログイベント**: 設定した**データ照会範囲**内のログを基準に、ログ検索条件を満たすログ数が閾値以上の場合にイベントが発生します。

> **権限**
>
> 複合メトリクスイベントを設定するには**通知設定**権限が必要です。 通知設定権限は、**ホーム画面 > プロジェクト選択 > 管理 > 統合メンバー管理**で、権限を付与するメンバーの アイコンをクリックして設定できます。

基本オプション

- **検索**: イベント名またはイベント受信タグを選択して目的のイベントを検索
- **JSONダウンロード/アップロード**: イベントルールはJSONエディタで直接編集するか、ファイルにダウンロードして修正後に再アップロード可能
- **通知メッセージ設定**: 受信するメッセージの詳細項目を選択し、項目ごとに内容を編集
- **+ イベント追加**: 新しいログイベントを追加

表 | 複合ログイベント一覧構成

項目	説明
イベント名	通知タイトルとして使用されるイベント名
テンプレート	複合ログテンプレート

項目	説明
カテゴリ	ログ分類名（ログフォルダ名）
ルール	イベントが発生する条件
イベント状態が解消されたら追加通知	CriticalとWarningレベルのイベントが解消されたとき、正常（RECOVERED）状態の通知を受信するかどうか - 有効: 進行中 → 正常状態に移行したときRECOVERED通知を送信し、イベント履歴で進行中/解消状態を区別して確認可能 - 無効: 閾値を超えるたびに単発のイベントが発生
イベント発生一時停止	イベント通知発生後、同じイベントの発生を一定時間停止 - 選択可能時間: 使用しない、5分、10分、15分、20分、30分、1時間、2時間、3時間、6時間、12時間、1日 - 解消通知が有効な場合、正常（RECOVERED）状態の通知受信後、選択した時間中は同じイベントが発生しても通知されない
インターバル	ログデータを照会する時間範囲
イベント受信タグ	イベント通知を受け取るユーザーまたはグループ（受信タグ）
	イベントの編集と削除
	イベントの有効化状態 - 有効: 条件を満たすとイベント発生 - 無効: イベントが動作せず、条件をチェックしない

イベント設定

設定した**データ照会範囲**内のログを基準に、ログ検索条件を満たすログ数が閾値以上の場合に警告通知を設定できる機能です。

複合ログイベントの追加

1. **アラート通知** > **イベント設定**の**複合ログ** タブをクリックしてください。
2. 画面右側の [+ イベント追加] ボタンをクリックしてください。
3. **イベントルール追加**ウィンドウでログイベント条件を設定してください。詳細については、基本設定、詳細設定、追加設定を参照してください。

   複合ログイベントの追加
4. **イベントルール追加**ウィンドウで設定が完了したら、[保存] ボタンをクリックしてください。

基本設定

- **イベント名**: イベントタイトルとして使用する名前を入力してください。
- **イベント有効化**: トグルボタンを有効にすると、条件を満たしたときにイベントが発生します。
- **レベル**: イベントレベル（Critical、Warning、Info）を選択してください。

詳細設定

テンプレート（プリセット）、メッセージ、カテゴリ、**フィルター**、イベント発行条件を設定します。

テンプレート（プリセット）

定義済みのプリセットを選択すると、**フィルター、クエリタイプ、イベント発行条件**が自動的に設定されます。プリセットを選択後、必要に応じて変更できます。

複合ログテンプレートの種類

プリセット	クエリタイプ	デフォルトルール
使用しない	none	`rows > 10`
2xxステータスコード件数	count	`count > 10`

プリセット	クエリタイプ	デフォルトルール
3xxステータスコード件数	count	`count > 10`
4xxステータスコード件数	count	`count > 10`
5xxステータスコード件数	count	`count > 10`
正常ステータスコード(2xx, 3xx)件数	count_v2	`include_minus_exclude_count > 10`
エラーステータスコード(4xx, 5xx)件数	count_v2	`include_minus_exclude_count > 10`
エラー受信件数	count	`count > 10`
例外受信件数	count	`count > 10`
毎秒ログ発生件数	rps	`rps > 100`
フィールド集計（Aggregation）	aggr	`avg > 1000`
比率（Rate）	rate	`rate > 10`
ユニーク値数（Cardinality）	cardinality	`cardinality > 100`
パーセンタイル（Percentile）	percentile	`p99 > 3000`

メッセージ

イベント発生時に受け取る通知メッセージを入力してください。メッセージに変数を使用して動的なメッセージを構成できます。

- **例示:** メッセージ `${host}` でエラー `${count}` 件発生 → 通知メッセージ web-01でエラー15件発生

表 | メッセージ変数

変数	説明	使用条件
${count}	条件に一致するログ件数	常に使用可能
${groupField}	グループ化フィールド名	グループ化通知のみ
${groupValue}	グループ化フィールド値	グループ化通知のみ
${フィールド名}	グループ化フィールド値（フィールド名で直接参照）	グループ化通知のみ

> ${count} 変数はグループ化の有無に関わらず、すべての通知で使用できます。 ${groupField} 、 ${groupValue} 、 ${フィールド名} 変数はグループ化フィールドが設定されている場合のみ置換されます。

カテゴリ

ログ分類名（ログフォルダ名）をリストから選択するか、直接入力してください。カテゴリを選択してから検索キーと検索値を入力できます。

フィルター

Luceneクエリ構文でログ検索条件を入力します。2種類の入力モードをサポートします。

- **フィルターモード**: フィールドと値を選択してクエリを構成します。
- **コードモード**: Luceneクエリを直接入力します。

表 | Luceneクエリ例

クエリ	説明
level:ERROR	levelフィールドがERRORのログ
status_nxx:status_4xx OR status_nxx:status_5xx	4xxまたは5xxステータスコード
appender:accessLog	accessLog appenderのログ
response_time.n:*	response_time.nフィールドが存在するログ

クエリ	説明
host:web-01 AND level:ERROR	web-01ホストのエラーログ

イベント発行条件

クエリタイプ別に設定します。選択したテンプレート（プリセット）またはクエリタイプに応じて、追加設定項目が表示されます。

表 | イベント発行条件キーの意味

キー	意味
rows / count	フィルター条件に一致するログ件数
rps	毎秒ログ発生件数（件数 / インターバル秒）
avg	対象フィールド値の平均
sum	対象フィールド値の合計
min	対象フィールド値の最小値
max	対象フィールド値の最大値
rate	分子件数 / 分母件数 × 100（%）
cardinality	対象フィールドのユニーク値数
p99 / p95 / p90 / p75 / p50	対象フィールドのパーセンタイル値
include_minus_exclude_count	含むフィルター件数 - 除外フィルター件数

- **件数（count / none）**: 設定したフィルター条件に一致するログの件数をカウントします。

  表 | 件数設定項目

設定項目	説明
フィルター	ログ検索条件（Luceneクエリ）
イベント発行条件	`件数` または `条件に一致するログ` `{演算子}` `{閾値}`（例: `count > 10`）

- **毎秒ログ発生件数（rps）**: 設定したフィルター条件に一致するログの毎秒発生件数（Requests Per Second）を計算します。インターバル時間中の総件数をインターバル秒数で割って算出します。

表 | 毎秒ログ発生件数設定項目

設定項目	説明
フィルター	ログ検索条件（Luceneクエリ）
イベント発行条件	`RPS` `{演算子}` `{閾値}`（例: `rps > 0.5`）

- **フィールド集計（aggr）**: 指定した数値フィールドの集計値（平均、合計、最小値、最大値）を計算します。イベント発行条件で集計方式をドロップダウンから選択できます。

表 | フィールド集計設定項目

設定項目	説明
フィルター	ログ検索条件（Luceneクエリ）
対象フィールド	集計対象の数値フィールド（例: `response_time.n`）
イベント発行条件	`平均` / `合計` / `最小値` / `最大値` `{演算子}` `{閾値}`（例: `avg > 1000`）

- **比率（rate）**: 分子フィルターと分母フィルターの件数比率（%）を計算します。全ログに対する特定条件の比率を監視する際に有用です。

表 | 比率設定項目

設定項目	説明
フィルター（分子）	比率計算の分子となるログ条件（例: `level:ERROR`）
分母フィルター	比率計算の分母となるログ条件（例: `level:*`）
イベント発行条件	`比率（%）` `{演算子}` `{閾値}`（例: `rate > 10`）

  - **例示**: フィルター `level:ERROR`、分母フィルター `level:*`、ルール `rate > 10` → 全ログ中のエラー比率が10%を超えると通知が発生

- **ユニーク値数（cardinality）**: 指定したフィールドのユニーク値数を計算します。異常に多くの種類の値が発生する場合を検出する際に有用です。

表 | ユニーク値数設定項目

設定項目	説明
フィルター	ログ検索条件（Luceneクエリ）
対象フィールド	ユニーク値を計算するフィールド（例: `host`）
イベント発行条件	`ユニーク値数` `{演算子}` `{閾値}`（例: `cardinality > 100`）

- **パーセンタイル（percentile）**: 指定した数値フィールドのパーセンタイル値を計算します。レスポンスタイムのテールレイテンシを監視する際に有用です。パーセンタイルの選択を変更すると、イベント発行条件のキーが自動的に更新されます。

表 | パーセンタイル設定項目

設定項目	説明
フィルター	ログ検索条件（Luceneクエリ）
対象フィールド	パーセンタイルを計算する数値フィールド（例: `response_time.n`）
パーセンタイル	P99、P95、P90、P75、P50から選択

設定項目	説明
イベント発行条件	`P99` / `P95` /... `{演算子}` `{閾値}`（例: `p99 > 3000`）

表 | パーセンタイルの意味

パーセンタイル	意味
P50	中央値。全データの50%がこの値以下
P75	全データの75%がこの値以下
P90	全データの90%がこの値以下
P95	全データの95%がこの値以下
P99	全データの99%がこの値以下

- **含む/除外件数（count_v2）**: 含むフィルターの件数から除外フィルターの件数を引いた純件数を計算します。

表 | 含む/除外件数設定項目

設定項目	説明
フィルター（含む）	含むログ条件（例: `status_nxx:status_4xx OR status_nxx:status_5xx`）
フィルター除外	除外するログ条件（例: `status_nxx:status_4xx`）
分母フィルター	比率計算用の分母（任意）
イベント発行条件	`含む-除外件数` `{演算子}` `{閾値}`（例: `include_minus_exclude_count > 10`）

  - **例示**: フィルター `status_nxx:status_4xx OR status_nxx:status_5xx`、フィルター除外 `status_nxx:status_4xx`
    → 4xx+5xxから4xxを除いた純5xx件数が10を超えると通知が発生

グループ化フィールド（Group By）

グループ化フィールドを設定すると、そのフィールドの値ごとに独立して通知条件が判定されます。ドロップダウンからグループ化するフィールドを選択します。ドロップダウンをクリックすると、現在のカテゴリで使用可能なフィールド一覧がオートコンプリートで表示されます。

- **例示**: フィルター `level:ERROR`、グループ化フィールド `host`、ルール `rows > 3`

表 | グループ化設定の比較

項目	グループ化なし	グループ化あり
判定単位	全ログを1つに集計	グループ値ごとに独立して集計
通知発生	1件	条件を満たすグループ数分
通知タイトル	`イベントタイトル`	`[グループ値] イベントタイトル`
Stateful	全体基準での状態遷移	グループごとに独立した状態遷移

  - `[web-01] エラー通知` — web-01でエラー8件 → 通知発生
  - `[web-02] エラー通知` — web-02でエラー5件 → 通知発生
  - `[api-01] エラー通知` — api-01でエラー2件 → 条件未達、通知なし

> ⓘ **イベント状態が解消されたら追加通知**を有効にした場合、グループごとに独立してABNORMAL → RECOVERED状態が遷移します。例えば `web-01` のエラーが解消されると `[web-01] RECOVERED` 通知が発生し、他のホストの状態には影響しません。

追加設定

- **インターバル**: 選択した時間（`unset`、`1分`、`5分`、`10分`、`30分`、`1時間`、`12時間`）間隔で通知条件を確認します。
- **無音**: イベント発生後、選択した時間（`unset`、`1分`、`5分`、`10分`、`30分`、`1時間`、`12時間`）中は同じイベントが発生しません。

- **イベント動作時間**: 特定の時間帯（業務/非業務/メンテナンス時間など）にイベントが動作するようにタグを設定してください。タグを設定しない場合、イベントが有効な状態で24時間常時動作します。

    1. **+ 追加**をクリックしてください。
    2. **イベント動作時間**で**+新しいタグを作成**をクリックしてください。
    3. **イベント動作時間タグの作成**で**タグ名、曜日、時間、色**を選択し、[適用] ボタンをクリックしてください。
    4. 作成したタグはタグリストで確認でき、チェックボックスを選択すると適用されます。
    5. 変更または削除するタグの✎アイコンをクリックして、**イベント動作時間タグの編集**でタグを編集または削除できます。

    > タグを削除すると、そのタグが適用されているすべてのユーザーのタグが削除されます。ただし、イベントルールで使用中のタグは削除できません。

- **イベント受信タグ**: イベントに受信タグを設定すると、そのタグを持つプロジェクトメンバーに通知が送信されます。受信タグを設定しない場合、プロジェクト内のすべてのユーザーに通知が送信されます。

    1. **+ タグ追加**または+をクリックしてください。
    2. **イベント受信タグ**ウィンドウ下部の**+ 新しいタグを作成**をクリックしてください。
    3. **タグ作成**ウィンドウでタグ名を入力し、色を選択してから、[タグ作成] ボタンをクリックしてタグを作成してください。
    4. 作成されたタグはタグリストの✎アイコンをクリックして編集または削除してください。
    5. **イベント受信タグ**ウィンドウの**タグリスト**から目的のタグを選択すると適用されます。

    > イベント設定時にイベント受信タグを選択して、そのタグを持つプロジェクトメンバーとサードパーティプラグインに通知を送信できます。**アラート通知 > 通知設定**でプロジェクトメンバーとサードパーティプラグインにそれぞれタグを指定できます。

シミュレーション

通知を保存する前に、過去のデータでシミュレーションして通知発生予測件数を確認できます。

1. 通知設定を完了後、下部の時間範囲を選択してください。
2. [シミュレーション] ボタンをクリックしてください。
3. チャートに時間帯別データと通知発生回数が表示されます。

   > ⓘ シミュレーション時間範囲を調整して、さまざまな時間帯の通知発生パターンを確認できます。直近1〜2分のデータはインデックスビルド遅延により照会されない場合があります。

4. **イベントルール追加**ウィンドウで設定が完了したら、[保存] ボタンをクリックしてください。

ログエクスプローラーから通知を作成

ログエクスプローラーから直接通知を作成できます。

1. **ログ** > **ログ検索**メニューに移動してください。
2. カテゴリと検索クエリを設定してください。
3. 画面右上の [イベント追加] ボタンをクリックしてください。
4. Drawerで通知設定フォームが開き、現在のカテゴリと検索クエリが自動的に入力されます。
5. 追加設定を完了し、[保存] ボタンをクリックしてください。
6. 保存完了後、[移動] ボタンをクリックすると通知一覧に移動します。

イベントの編集/削除

1. **アラート通知** > **イベント設定**の**複合ログ**タブに移動してください。
2. ログイベントリストで編集または削除するイベントの アイコンをクリックしてください。
3. **イベント設定**画面でオプションを変更し、[保存] ボタンをクリックしてください。

   a. 選択したイベントを削除するには、**イベント設定**画面の右上の [削除] ボタンをクリックしてください。

イベントの共有

メトリクスイベント設定を JSON ファイルとして保存し、他のユーザーと設定を共有したり、他のユーザーの設定を取り込んだりすることができます。

- JSON ファイル名: event-rules-`YYYYMMDD`.json

エクスポート

1. 画面右上の [**JSON**] ボタンをクリックします。
2. JSON 編集画面が表示されたら、[エクスポート] ボタンをクリックします。
    - イベントを検索した状態でエクスポート機能を使用すると、検索結果のみが JSON ファイルとしてダウンロードされます。
3. JSON ファイルがダウンロードされたら、共有したい他のユーザーに渡してください。

インポート

1. 画面右上の [ ] ボタンをクリックします。
2. **エクスポート** 機能でダウンロードした JSON ファイルを選択します。
3. JSON 編集画面が表示されたら、[一覧に追加] または [上書き] を選択します。

> 同じ種類の製品間での利用を推奨します。異なる製品のプロジェクトからイベント設定を取り込むことは可能ですが、正常に動作しない場合があります。

JSON 形式で編集する

1. 画面右上の [**JSON**] ボタンをクリックします。
2. 編集画面が表示されたら、JSON 形式に従って内容を修正してください。
3. 修正が完了したら、画面下部の [保存] ボタンをクリックします。

修正した内容が JSON 形式に合っていない場合、画面下部にエラーメッセージが表示され、保存できません。表示されるエラーメッセージは、形式エラーの内容によって異なる場合があります。

JSON error

トランザクション

ホーム > プロジェクト選択 > **アラート通知** > **イベント設定** > **TX** タブ

トランザクションイベントは、特定の URL、エラー、ステータスコードなどを基準にトランザクションを詳細に監視する機能です。設定した条件を満たすトランザクションが検出されるとイベントが発生します。トランザクションデータは 1 分単位で分析され、条件を満たすトランザクション数がしきい値を超えた場合に通知が送信されます。

最新エージェントバージョン

トランザクションイベントを使用するには、最低対応バージョン以上のエージェントが必要です。

言語	最低対応バージョン
Java	2.2.50 以上
PHP	未定（最新バージョンの使用を推奨、別途お問い合わせください）
Node.js	0.5.21 以上
Python	1.7.8 以上
.NET	2.3.5 以上
Go	未定（最新バージョンの使用を推奨、別途お問い合わせください）

基本オプション

- **イベントレベル**: イベントの重大度を示し、Info、Warning、Critical の 3 段階に分類されます。
- **検索**: イベント名またはイベント受信タグでイベントを検索します。
- **+ イベント追加**: 新しいトランザクションイベントを追加します。
- **カラムサイズ調整**: 各カラムの境界をドラッグして幅を調整できます。

表 | トランザクションイベント一覧の構成

項目	説明
No.	イベントの番号
	イベントの有効／無効 - 有効: 条件を満たすとイベントが発生 - 無効: イベントは動作せず、条件を判定しません
	イベント設定の修正および削除
イベント名	ユーザーが指定したイベント名 - 最大 255 バイト
条件タイプ	イベント発生条件の種類（経過時間、エラー発生、パターンのみ）
集計基準	トランザクション数を集計する基準
対象パターン	イベントが適用される包含／除外パターン（URL、ステータスコードなど）
発生回数	イベント発生の基準となる回数 - 集計周期: 1分固定（毎正時00秒〜59秒基準、期間選択不可） - この1分間に条件を満たしたトランザクションが設定回数以上発生するとイベントが発生
一時停止	アラート発生後、同一イベントの発生を一定時間抑制 - 選択可能時間: 使用しない、5分、10分、15分、20分、30分、1時間、2時間、3時間、6時間、12時間、1日 - 解消通知が有効な場合、RECOVERED 通知を受信後、選択した時間内は同一イベントが発生しても通知されません
イベント受信	当該イベント通知を受信するユーザーまたはグループ（受信タグ）

イベント設定

トランザクションイベントは、**イベント条件、対象パターン、基本情報および受信設定** の各ステップで構成されます。
イベント設定により、特定のトランザクションパターンを検知し、通知を受け取ることができます。

イベント追加

トランザクションイベントを追加するには、**イベント設定** > **TX** タブに移動し、画面右上の [+ イベント追加] ボタンをクリックしてください。

トランザクションイベント追加手順
Step 1. イベント条件定義: イベント発生の条件を定義します。
Step 2. イベント対象選択: イベントが適用される監視対象を選択します。
Step 3. 基本情報および受信設定: イベント名、メッセージ、受信対象を設定します。

イベント条件定義

トランザクションイベントの発生条件を設定します。

条件タイプ

イベントを発生させる基準を選択してください。

- **経過時間**: 指定した時間（ms）以上かかったトランザクション
  - **経過時間しきい値**: 条件タイプが経過時間の場合、ミリ秒（ms）単位で入力します。設定した時間（ms）以上かかったトランザクションが発生するとイベントが発生します。
- **エラー発生**: エラーが発生したトランザクション
- **パターンのみ**: 特定のパターン（ヒットマップパターンなど）に該当するトランザクション

集計基準

アラート条件をどの単位で集計するかを選択してください。

- 集計基準: エージェント別、各エージェントの URL 別、各エージェントのエラークラス別、各エージェントの業務別、URL 別、エラークラス別、業務別、全体。

発生回数（必須）

条件を満たしたイベントが何回発生したときに通知を送信するかを設定します。

例: 発生回数 **1回** は1分以内に条件を満たしたトランザクションが1件でもあれば即時通知、発生回数 **3回** は同じ1分間に条件を満たしたトランザクションが3件以上累積した場合に通知されます。

一時停止

過剰なイベント通知を防ぐため、通知を一時停止する時間を選択してください。

- イベント発生後、設定した時間が経過するまで同一イベントは再度発生しません。
- 選択可能時間: `1分` , `5分` , `10分` , `15分` , `30分` , `1時間` , `2時間` , `3時間` , `6時間` , `12時間` , `1日`

イベント動作時間

イベントが動作する時間帯を設定します。**非勤務時間や点検時間帯の通知発生を制限**できます。タグを設定しない場合、イベントが有効な状態では 24 時間常時動作します。

1. **+ 追加** をクリックします。
2. イベント動作時間で **+ 新しいタグを作成** をクリックします。
3. **イベント動作時間タグ作成** で **タグ名、曜日、時間、色** を選択し、[適用] ボタンをクリックします。
4. 作成したタグはタグ一覧で確認でき、チェックボックスを選択すると適用されます。
5. タグを修正または削除するには、 をクリックし、**イベント動作時間タグ修正** で修正または削除してください。

> タグを削除すると、そのタグが適用されているすべてのアラートから削除されます。ただし、イベント設定で使用中のタグは削除できません。

イベント対象選択

トランザクションをフィルタリングするためのパターンを設定します。

パターン設定方法

- `Enter` キーで絞り込んで複数のパターンを入力できます。
- パターンを設定しない場合、プロジェクト内のすべてのトランザクションが対象となります。
- 同一タイプのパターンは **OR** 条件で適用されます。
- 異なるタイプのパターンは **AND** 条件で適用されます。
- ワイルドカード（`*`）を使用できます。ワイルドカードの位置は最大 2 箇所まで指定できます。
- ワイルドカード（`*`）やスラッシュ（`/`）以外の文字を必ず含める必要があります。

> **スラッシュ（`/`）とワイルドカード（`*`）の組み合わせに注意してください**
>
> - `/api/*` ： `/api` 自体はマッチしません。 `/api/` の後にパスが必要です（例: ✅ /api/users、❌ /api）
> - `/api*` ： `/api` 自体もマッチします（例: ✅ /api、✅ /api/users、✅ /apitest）

一致パターン

指定した項目と一致するデータをフィルタリングします。一致パターンは適用後、青色タグ として表示されます。
各項目につき、最大 10 個までパターンを入力できます。

項目	説明
業務名	業務名パターン
URL	トランザクション URL パターン - 例: `/api/*`, `*/users/*`
エラークラス	エラークラス名パターン - 例: `*Exception`, `*Error`
エラーメッセージ	エラーメッセージ内容パターン

項目	説明
ステータスコード	HTTP ステータスコードパターン - 例: `4**`, `5**`

除外パターン

指定した項目と一致するデータをフィルタリング対象から除外します。除外パターンは適用後、赤色タグ として表示され、一致パターンと同じ形式を使用します。各項目につき、最大 10 個までパターンを入力できます。

パターン例

- **完全一致**

```
パターン: "hello"
"hello"   → 一致
"Hello"   → 不一致（大文字・小文字を区別）
```

- **前方一致（プレフィックス一致）**

```
パターン: "hello*"
"hello"     → 一致
"helloworld" → 一致
"hi"        → 不一致
```

- **後方一致（サフィックス一致）**

```
パターン: "*world"
"world"     → 一致
"helloworld" → 一致
"hello"     → 不一致
```

- **部分文字列一致**

```
パターン: "*ell*"
"hello"     → 一致（h'ell'o）
"excellent" → 一致（exc'ell'ent）
"welcome"   → 不一致（"ell" を含まない）
```

- **複合パターン**

```
パターン: "hello*world"
"hellobeautifulworld" → 一致

パターン: "hello*test*"
"hellomytestcase"    → 一致

パターン: "*test*world"
"mytestcaseworld"    → 一致
```

表 | 実際の使用例

パターン種別	例
URL パターン	- `/api/*`：`/api/` で始まるすべての URL - `/users/*`：`users` を含むすべての URL - `/api/v1/users`：完全一致する URL のみ
エラークラスパターン	- `Exception`：`Exception` で終わるすべてのエラー - `java.lang.*`：`java.lang` で始まるすべてのクラス - `Timeout*`：`Timeout` を含むすべてのエラー
ステータスコードパターン	- `4**`：400 番台のすべてのステータスコード - `404`：404 ステータスコードのみ - `5**`：500 番台のすべてのステータスコード

パターンの制限事項

- **必須文字列**：パターンにはワイルドカード（`*`）以外の文字を最低 1 文字以上含める必要があります。ワイルドカードのみではパターンを設定できません。
  - ❌ 不可：`**`, `*`
  - ✅ 可：`a*`, `a`, `a*`, `/api*`
- **空文字列**：空文字列や null 値はマッチしません。
- **連続ワイルドカード**：連続した `*` は 1 つのワイルドカードとして扱われます。
- **複雑なパターン**：ワイルドカードを 3 個以上含むパターンはサポートされていません。

- **正規表現非対応**：文字クラス（`[a-z]`）、量指定子（`+`, `?`）、グループ化（`()`）は使用できません。
- **大文字・小文字の区別**：すべてのパターンマッチングは大文字と小文字を区別します。
- **エスケープ不可**：文字をそのままリテラルとしてマッチさせる方法はありません。

基本情報および受信設定

イベントの基本情報と通知の受信方法を設定します。

イベント有効化

イベント発生時に通知を送信するかどうかを設定します。

- **有効**: 設定した条件を満たすとイベント通知が送信されます。
- **無効**: 設定した条件を満たしても通知は送信されません。

イベント名（必須）

通知タイトルとして使用されるイベント名を入力してください。

イベントレベル（必須）

イベントレベル（Info、Warning、Critical）を選択してください。

イベント受信

イベント通知を受信するメンバーを指定します。

- **全体受信** **タグ選択受信**

プロジェクトで通知設定が有効になっているすべての対象（メンバー、チャンネル）に通知が送信されます。

指定したタグを持つプロジェクトメンバーに通知が送信されます。

1. **タグ選択受信**を選択します。
2. **受信タグ**項目で [+ タグ追加] または [+] ボタンをクリックします。
3. **イベント受信タグ**ウィンドウ下部の **+ 新しいタグ作成** をクリックします。

4. **タグ作成**ウィンドウでタグ名を入力し、色を選択して [タグ作成] をクリックします。
5. 作成したタグは、タグ一覧の をクリックして編集または削除できます。
6. **イベント受信タグ**ウィンドウの **タグ一覧** から適用したいタグを選択します。

> **受信タグ**
>
> 受信タグは、通知を受け取るユーザーやチャンネルを **グループ単位で管理するための機能** です。
> イベントに受信タグを設定すると、そのタグを持つユーザーにのみ通知が送信されます。
> 受信タグを設定しない場合、プロジェクト内のすべてのユーザーに通知が送信されます。

参考. イベント動作例

トランザクションイベントが実際に発生した例です。

> **設定条件**
>
> - 集計基準: エージェント別
> - 発生回数: 2回
> - 一時停止: 5分
> - URL パターン: `/api/posts/**` , `/delete/*` , `/*test*/`
> - エラークラスポターン: `*TimeoutException*` , `*NullPointerException`
> - ステータスコードパターン: `4**`
> - 除外エラークラス: `*IOException`
> - 除外ステータスコード: `5**`

発生した通知メッセージ例

イベントメッセージ内のトランザクション情報は、条件に合致した最後のトランザクションの値が表示されます。イベントメッセージには、設定した条件と実際に発生したトランザクションの情報の両方が含まれるため、どの条件によって通知が発生したのかを明確に確認できます。

```
(ignored count: 3)
```

```
[Per Agent] limit: 2, count: 3
[Condition]
url patterns: /api/posts/**, /delete/*, */test/*
error class patterns: *TimeoutException*, *NullPointerException
status patterns: 4**
exclude error class patterns: *IOException
exclude status patterns: 5**
[Transaction]
url: /api/posts/test/timeout
txid: 408093121470636540
status: 408
error class: java.util.concurrent.TimeoutException
```

イベントメッセージ構成の説明

- **(ignored count: n)**: 一時停止時間中に条件を満たしたが、通知が送信されなかったトランザクションの件数です。一時停止時間中に無視された件数がない場合、この項目は表示されません。
- **集計情報**: 設定した集計方式に応じて表示内容が異なります。
  - **エージェント別**: `[Per Agent] limit: n, count: n`
  - **URL別**: `[Per Url] limit: n` の後に、各URLと発生件数を表示
  - **エラークラス別**: `[Per Error Class] limit: n` の後に、各エラークラスと発生件数を表示
  - **コンテキスト別**: `[Per Context] limit: n` の後に、各コンテキストと発生件数を表示
  - **エージェント別 URL**: `[Per Agent & Url] limit: n` の後に、各URLと発生件数を表示
  - **エージェント別 エラークラス**: `[Per Agent & Error Class] limit: n` の後に、各エラークラスと発生件数を表示
  - **エージェント別 コンテキスト**: `[Per Agent & Context] limit: n` の後に、各コンテキストと発生件数を表示
  - **全体**: `[Total] limit: n, count: n`
- **[Condition]**: イベント発生時点で設定されていた条件を表示します。設定されたパターンのみが表示されます。
  - **context patterns**: コンテキスト一致パターン
  - **url patterns**: URL 一致パターン
  - **error class patterns**: エラークラス一致パターン
  - **error message patterns**: エラーメッセージ一致パターン
  - **status patterns**: ステータスコード一致パターン

- **exclude context patterns**: コンテキスト除外パターン
- **exclude url patterns**: URL 除外パターン
- **exclude error class patterns**: エラークラス除外パターン
- **exclude error message patterns**: エラーメッセージ除外パターン
- **exclude status patterns**: ステータスコード除外パターン

- **[Transaction]**: 条件に合致した最後のトランザクションの詳細情報です。
  - **url**: トランザクションURL
  - **txid**: トランザクション固有識別子
  - **status**: HTTP 応答ステータスコード（HTTP トランザクションでない場合は非表示）
  - **error class**: エラークラス（エラークラスが存在しない場合は非表示）
  - **error message**: エラーメッセージ（エラーメッセージが存在しない場合は非表示）

> **参考**
>
> - トランザクションイベントは1分単位で集計・処理されます。
> - 条件タイプが「パターンのみ」の場合、少なくとも1つ以上の一致または除外パターンを設定する必要があります。
> - パターンマッチングは大文字・小文字を区別します。
> - 通知メッセージには、条件に合致した最後のトランザクションの詳細情報が含まれます。

コンテナ

ホーム > プロジェクト選択 > **アラート通知** > **アプリケーション イベント設定** *classic*

コンテナのリソース使用量、進行中トランザクション、エラー状況などのイベント条件に基づいてアラート通知を送信します。

> イベント設定 *classic* は、従来方式のイベント設定を継続して利用できるよう分離されたメニューです。
> より多くの機能は イベント設定 で利用できます。

基本オプション

エージェント非アクティブアラート

設定した時間内にエージェントからデータを受信できなかった場合、イベントが発生します。トグルボタンで有効または無効を切り替えられます。

- **有効**: 一定期間データ未受信時にエージェント非アクティブイベントが発生
- **無効**: データ受信状況に関係なくイベントは発生しません

> 従来のエージェント非アクティブ確認間隔は **管理 > エージェント一覧** で設定できます。

状態解消時の追加通知

条件を満たしてイベントが発生した後、状態が正常に復旧した際に **解消（RECOVERED）通知** を送信するかを選択できます。トグルボタンで切り替えます。

- **有効**: イベントは状態ベースで動作し、しきい値超過で進行中、しきい値未満に復旧すると解消状態に切り替わります
- **無効**: しきい値を超えるたびに単発イベントが発生します

エージェント重複アラート

同一名のエージェントが同時に 2 つ以上起動すると、正常なモニタリングが困難になるためイベントが発生します。ただし、設定した時間内は重複エージェントイベントは再発しません。

- エージェント名の重複有無は **イベント設定** メニューで確認してください。
- 選択可能時間: 30秒 , 1分 , 3分 , 5分 , 10分

条件

項目ごとにイベント発生条件を設定できます。項目によって入力形式が異なります。

- **サーバー CPU / サーバー メモリ / サーバー ディスク**
- **アクティブTX / トランザクションエラー**
- **トランザクション応答時間**

条件（%）

Warning と Critical の 2 段階のしきい値を設定します。
しきい値超過時にアラートが発生し、Critical が優先されます。

- 左の値: 警告（Warning）しきい値
- 右の値: 危険（Critical）しきい値

条件（件数）

イベント発生の基準となる件数を入力します。指定した継続時間中に設定値を超えた状態が続くとイベントが発生します。

条件（件数）

条件を設定します。設定した応答時間を超えたトランザクションが指定件数以上発生するとイベント条件を満たします。

条件（応答時間）

基準となる応答時間を選択します。

- 選択可能時間: 1秒 , 2秒 , 3秒 , 5秒 , 10秒 , 15秒 , 30秒 , 1分 , 2分 , 3分 , 5分 , 10分 , 30分 , 1時間
- 例: 件数基準が **10件**、応答時間基準が **1秒**、継続時間が **5秒** の場合

→ 1秒を超えるトランザクションが 10 件以上発生し、その状態が 5 秒継続するとイベントが発生します。

継続

選択した時間中にイベント条件が持続する場合、イベントが発生します。

- 選択可能時間: `5秒` , `10秒` , `15秒` , `30秒` , `1分` , `2分` , `3分` , `5分` , `10分` , `30分` , `1時間`

サイレンス

イベント発生後、選択した時間中に同一条件のイベントは発生しません。適切なサイレンス時間の設定により、過剰なイベント発生を防止できます。

- 選択可能時間: `1分` , `2分` , `3分` , `5分` , `10分` , `30分` , `1時間` , `2時間` , `3時間` , `6時間` , `12時間` , `1日`

イベント有効化

トグルボタンでイベントの動作を制御できます。

- **有効:** 設定した条件を満たすとイベントが発生します
- **無効:** 条件を評価しません

イベント設定

次の項目にイベントを設定することで、システム状態をリアルタイムで監視できます。各項目を設定後、画面右上の **保存** ボタンをクリックしてください。

サーバー CPU / サーバー メモリ / サーバー ディスク

サーバーリソース使用量が設定したしきい値（%）を超えるとイベントが発生します。Warning と Critical で異なるしきい値を設定できます。

- 例: CPU 使用率を Warning: 80%、Critical: 90% に設定した場合、90% を超えると Critical イベントが発生します。

アクティブTX

現在処理中のトランザクション数を監視します。設定した基準件数を超え、その状態が **継続時間** 以上維持されるとイベントが発生します。大量の同時トランザクションを検知するのに有効です。

- 発生件数のデフォルト: 100 件

トランザクションエラー

正常に終了しなかったトランザクションをエラーとして判断します。エラー件数が設定した基準を超え、その状態が継続するとイベントが発生します。

- 収集間隔: 5 秒

トランザクション応答時間

各トランザクションの完了までにかかった時間（応答時間）を基準にイベントを設定します。期待される応答時間を基準に、サービス遅延の検知が可能です。

- 例: 応答時間基準を 10 秒、件数基準を 10 件に設定した場合
  「応答時間 10 秒を超えるトランザクションが 10 件以上発生し、その状態が 5 秒間継続するとイベントが発生」

通知設定

ホーム > プロジェクト選択 > **アラート通知** > **通知設定**

ユーザー別アラート受信設定

ユーザー別アラート受信設定 における他メンバーの情報は、**メンバー管理** 権限を持つメンバーのみが確認できます。それ以外のメンバーには、必須情報がマスキングされて表示されます。

- イベント通知の一括受信設定およびアクセス設定のためのモバイル端末管理は、**アカウント情報** で行うことができます。
- メンバー権限の詳細については、こちらのドキュメント を参照してください。

通知手段の設定

メール、**SMS**、**WhatsApp**、**モバイル** による通知を提供します。

ユーザー別アラート受信設定 セクションで、受信したい通知手段のチェックボックスを選択すると警告アラートを受信できます。通知手段のチェックボックスを解除すると、その手段ではアラートは送信されません。

プロジェクトの最高管理者を除き、すべてのユーザーは自分自身の受信設定のみ変更できます。

メール通知

メール通知は、会員登録時に入力したメールアドレス宛に送信されます。

SMS通知

SMSで通知を受信できます。

SMS通知機能は、有料プロジェクトでのみ利用できます。

1. 画面右上の **プロフィールアイコン** をクリックし、**アカウント管理** に移動します。
2. **アカウント情報** の **ユーザー電話番号** セクションで、[一般携帯電話] ボタンをクリックします。
3. 認証番号を受信する **電話番号** を入力します。
    - SMS通知を受信できる電話番号は、**韓国の携帯電話番号のみ**対応しています。
4. [保存] ボタンをクリックします。

> **登録した電話番号の変更および削除**
>
> - 電話番号を変更する場合は、[変更] ボタンをクリックしてください。SMS認証が必要です。
> - 電話番号を削除する場合は、[削除] ボタンをクリックしてください。削除後は、SMSによる通知を受信できなくなります。

WhatsApp通知

WhatsAppを通じて通知を受信できます。

> WhatsApp通知機能は、有料プロジェクトでのみ利用できます。

1. 画面右上の **プロフィールアイコン** をクリックし、**アカウント管理** に移動します。
2. **アカウント情報** の **ユーザー電話番号** セクションで、[WhatsApp] ボタンをクリックします。
3. 認証番号を受信する **電話番号** を入力します。
4. [認証番号送信] ボタンをクリックします。
5. WhatsAppアプリケーションに送信された6桁の認証番号を入力します。
    - 10分以内に認証番号を受信できなかった場合は、[認証番号再送信] ボタンをクリックしてください。
6. [認証] ボタンをクリックします。

モバイル通知

モバイル通知を受信するには、モバイル端末にWhaTapアプリケーションをインストールし、ログインする必要があります。

- WhaTapアプリケーションをダウンロード – **Android**
- WhaTapアプリケーションをダウンロード – **iOS**

繰り返し通知（エスカレーション）設定

アラート発生後も状況が解消されない場合、初回アラート発生時刻を基準に通知を繰り返す間隔を設定できます。この設定は **Critical** レベルのアラートにのみ適用されます。

例えば、初回アラート発生から即時（0分）、1時間後、1日後に通知を繰り返す場合は、**繰り返し通知（エスカレーション）** カラムに `0,1H,1D` と入力します。

1. **アラート通知** > **通知設定** に移動し、**ユーザー別アラート受信設定** セクションを開きます。
2. **繰り返し通知（エスカレーション）** 項目で設定します。
   - **M**: 分、**H**: 時間、**D**: 日。単位を省略した場合は分単位として扱われます。
   - 数値、または 数値＋単位（**M**, **H**, **D**）の形式で入力してください。入力形式が正しくない場合はメッセージが表示されます。
3. [保存] ボタンをクリックしてください。保存しない場合、設定は反映されません。

イベント受信タグ

イベント設定時に **イベント受信タグ** を選択すると、そのタグを持つプロジェクトメンバーや 3rd-party プラグインに通知を送信できます。イベント受信タグを設定しない場合、**すべてのプロジェクトメンバー** に警告アラートが送信されます。

イベント受信タグの追加・編集・削除

> イベントに **適用中のイベント受信タグ** は削除できません。

1. **アラート通知** > **通知設定** に移動します。
2. **ユーザー別アラート受信設定** セクションのユーザー一覧で、**+ タグ追加** または **+** をクリックします。
3. **イベント受信タグ** 画面で **+ 新しいタグ作成** をクリックします。
4. **タグ作成** 画面でタグ名を入力し、色を選択して [タグ作成] ボタンをクリックします。

5. 作成されたタグは、タグ一覧の をクリックして編集または削除できます。
6. **イベント受信タグ** 画面の **タグ一覧** から使用したいタグを選択すると適用されます。

イベント受信タグを解除する

1. **アラート通知** > **通知設定** に移動し、**ユーザー別アラート受信設定** セクションを開きます。
2. イベント受信タグを解除したいユーザーの **イベント受信タグ** 項目にある + をクリックします。
3. **イベント受信タグ** 画面で、解除したいタグの **X** をクリックすると、そのタグが解除されます。

受信タグ未設定のアラートを受信する

イベント設定時にイベント受信タグを選択すると、そのタグを持つプロジェクトメンバーや 3rd-party プラグインに通知を送信できます。

- イベント受信タグが設定されていないアラートを受信するには、**受信タグ未設定のアラートを受信** オプションを有効にしてください。
- イベント受信タグが設定されているアラートのみ受信したい場合は、このオプションを無効にしてください。

> すべての警告アラートを受信しない場合は、このオプションを無効にし、**イベント受信タグ** が一つも選択されていない状態にしてください。

イベント受信モードの選択

ユーザー別アラート受信設定セクションで、メンバーまたはサードパーティチャネルごとにイベント通知の受信範囲を次の3つのモードから選択できます。受信モードは、各メンバーまたはチャネル行の**イベント受信タグ**列で選択します。

Mode	Description
すべての通知を受信	イベント受信タグに関係なく、すべての警告通知を受信
選択したタグのみ通知を受信	選択したイベント受信タグを持つ警告通知のみ受信。タグを最低1つ以上選択
すべての通知を受信しない	いかなるイベント通知も受信しない。選択したタグは削除されずそのまま保存

- **選択したタグのみ通知を受信**モードでは、タグ選択領域で通知を受信するタグを1つ以上選択してください。
- イベント受信タグが指定されていない警告通知まで受信するには、タグ選択領域で**未分類**仮想タグを選択してください。**未分類**タグは通常のタグと同様に選択または解除でき、タグ自体を編集・削除することはできません。

> ⓘ **すべての通知を受信しない**モードでは、該当メンバーまたはチャネルがイベント通知をまったく受信しません。このとき選択したタグは削除されず保存されるため、再び**選択したタグのみ通知を受信**モードに切り替えると、以前に選択したタグ設定をそのまま使用できます。

通知受信言語の選択

ユーザーが受信するアラート通知の言語を選択できます。

1. **アラート通知** > **通知設定** に移動します。
2. **ユーザー別アラート受信設定** セクションのユーザー一覧で、言語を設定したいユーザーの一番左にある ▶ ボタンをクリックします。
3. **通知受信言語** を選択します。

曜日・時間別通知設定

ユーザーごとに、曜日および時間帯別の通知受信可否を設定できます。

1. **アラート通知** > **通知設定** に移動します。
2. **ユーザー別アラート受信設定** セクションのユーザー一覧で、曜日・時間別通知を設定したいユーザーの一番左にある ▶ ボタンをクリックします。
3. 警告アラートを受信したい曜日を選択するか、時間帯を入力します。
    - 通知手段ごとに設定できます。

サードパーティプラグイン

外部アプリケーションを通じて警告アラートを受信できます。

- **対応アプリケーション**: Slack、Telegram、Teams、Teams Workflows、Jandi、AlertNow、Webhook、Webhook

JSON、Opsgenie、PagerDuty、LINE、ilert

> WhaTap のサポート範囲に含まれない社内メッセンジャーは、標準の **Webhook** または **Webhook JSON** を使用して連携できます。

表 | サードパーティプラグイン一覧

項目	説明
プラグイン名	プラグインの名称
認証キー	認証キー
認証値	認証値
受信レベル	警告、全体、危険のいずれか
繰り返し通知（エスカレーション）	問題が解消されない場合、初回通知時刻から再通知する間隔 - Critical レベルのアラートにのみ適用
Smart メッセージ（AI）	Telegram プラグイン追加時に AI 通知機能を提供 - 対応言語：韓国語、英語、日本語
イベント受信タグ	プロジェクト内の特定メンバーまたはチームに対する通知受信設定
削除	サードパーティプラグインの削除

サードパーティプラグインの追加

1. **アラート通知** > **通知設定** に移動します。
2. **サードパーティプラグイン** セクションで **追加** ボタンをクリックします。
3. **サードパーティプラグイン追加** 画面で、使用するサービスのタブを選択します。
4. 選択したサービスの案内（サードパーティプラグイン通知の追加方法）に従って設定します。
5. **受信テスト** ボタンをクリックして設定を確認します。

6. すべて完了したら **追加** または **登録** ボタンをクリックします。

サードパーティプラグイン通知の追加方法

Slack

サードパーティプラグイン追加 画面の **Slack** タブで、[**Slackに登録**] ボタンをクリックすると、Slack のチームおよびチャンネルに **WhaTapNotiBot** を登録できます。

Telegram

1. **Telegram**で **@WhaTapApmNotiBot** を検索してください。

   Telegram

2. ボットを選択し、[開始] ボタンをクリックします。
3. チャット画面に `/license` コマンドと、通知を受信したいプロジェクトのアクセスキーを入力してください。

   Telegram

使用可能なコマンド一覧

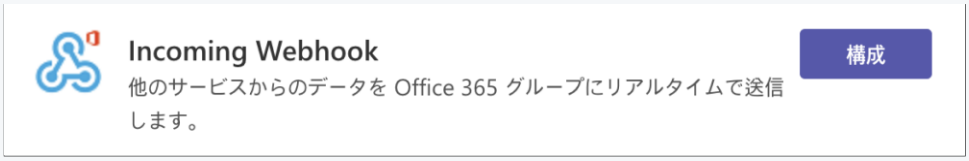
- `/help` : コマンド一覧を表示
- `/license` : 通知を受信するプロジェクトのアクセスキーを入力
- `/remove` : 通知を受信しないプロジェクトコードを入力
- `/list` : 現在通知を受信しているプロジェクトコードを表示

Teams

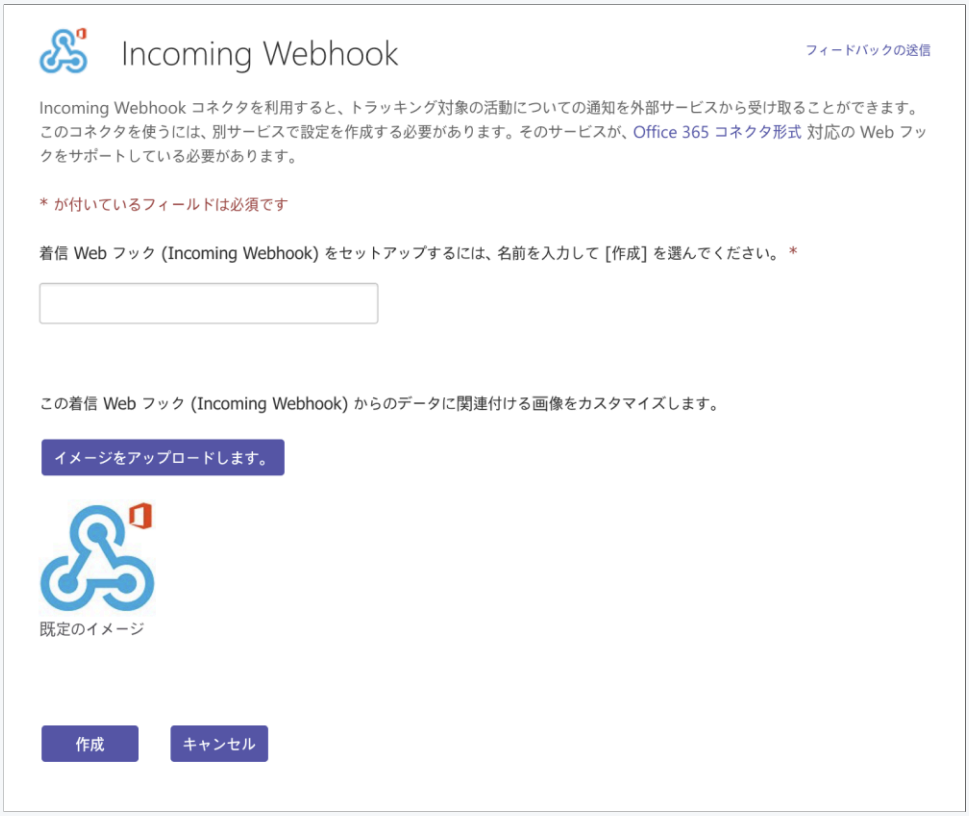
1. 通知を受信したいチャンネルの **コネクタ** に移動します。



2. **Webhook** を追加し、[構成] ボタンをクリックします。



3. Webhook 名を入力し、[作成] ボタンをクリックします。



4. 生成された **Webhook URL** をコピーします。

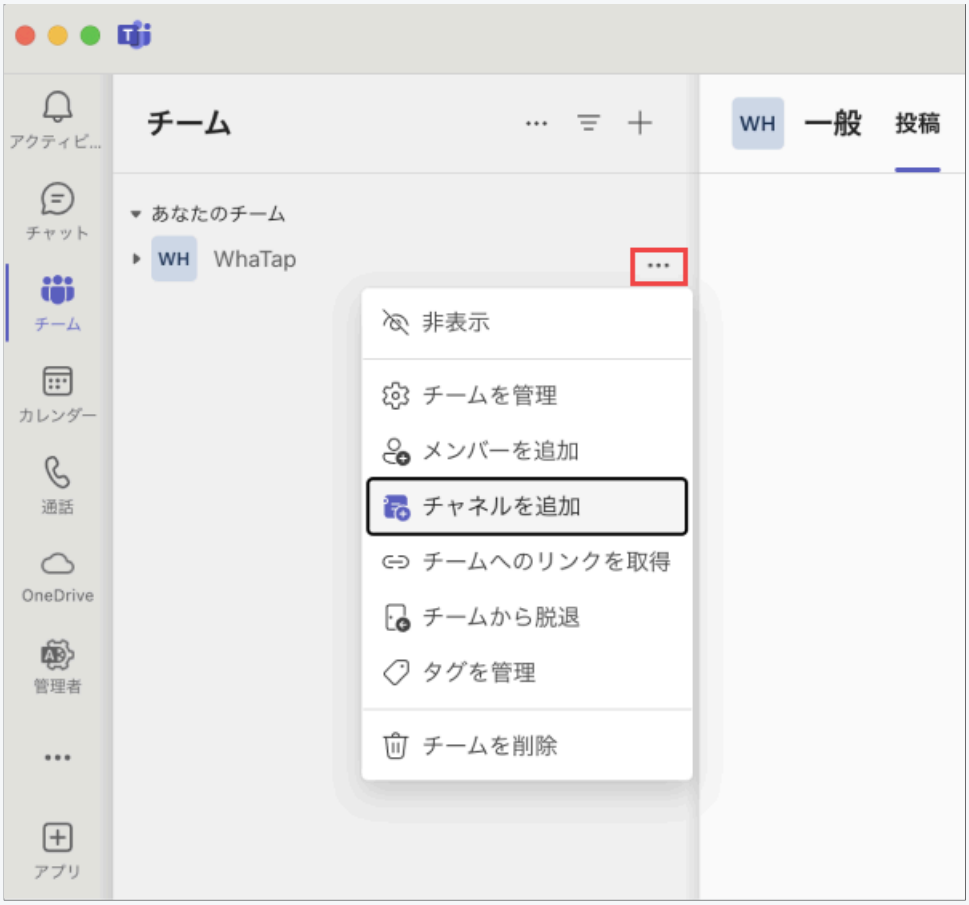


5. WhaTap モニタリングサービスの **サードパーティプラグインの追加** 画面に戻り、**Teams** タブ下部の **Channel Name** と **Webhook URL** に値を入力します。
6. [**登録**] ボタンをクリックします。

Teams Workflows

ワークフローを使用するには、**Microsoft Teams** にアクセス可能なアカウントと **ビジネスプランのサブスクリプション**が必要です。

1. **チーム** メニューの **その他の操作 (...)** > **チャネルを追加** をクリックします。

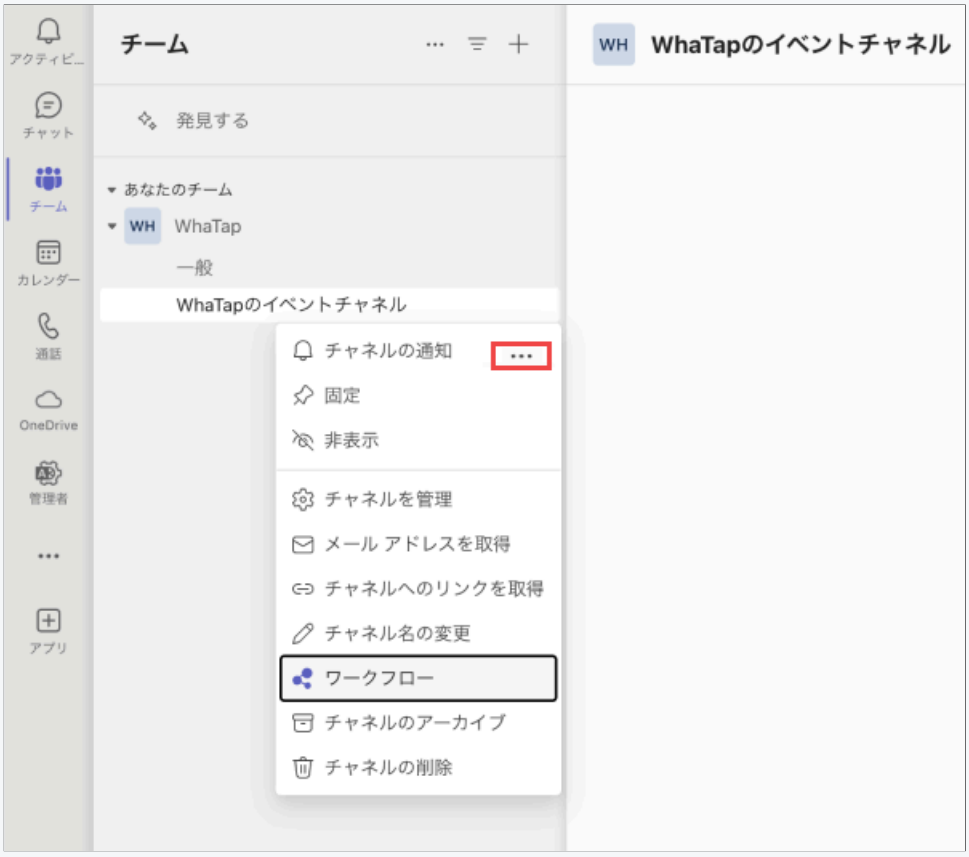


2. **チャネルの作成** 画面で **チャネル名** と **説明** を入力します。**チャネルの種類** は **標準** を選択し、[作成] ボタンをクリックします。

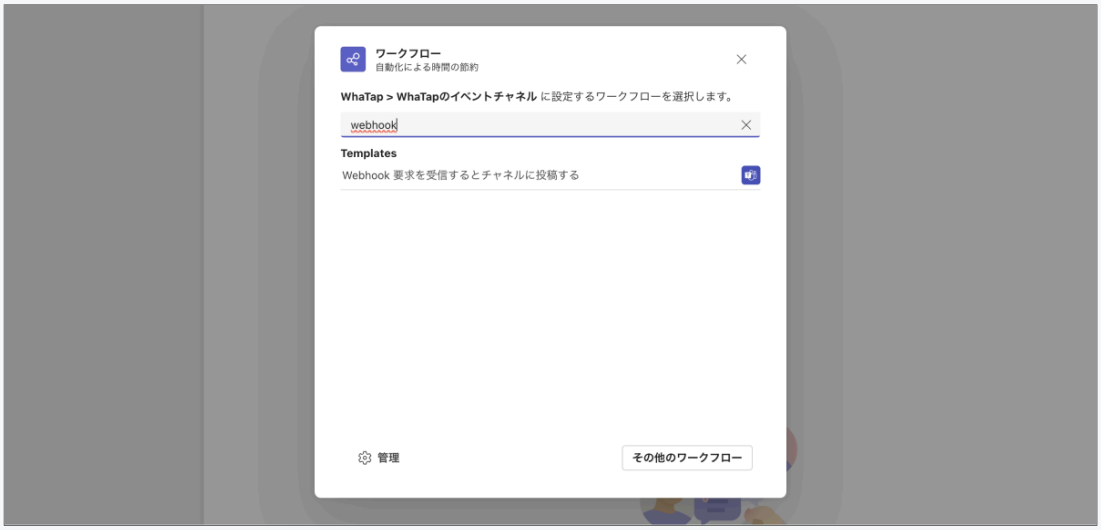


> チャネルの種類を **共有** にするとワークフローは使用できません。**プライベート** を選択した場合、ワークフローは追加できますが通知は受信できません。

3. 作成したチャネルの **その他の操作 (...)** > **ワークフロー** をクリックします。



4. **ワークフロー** ダイアログの一覧から **Webhook 要求を受信したらチャネルに投稿** を選択し、[ワークフローを追加] ボタンをクリックします。



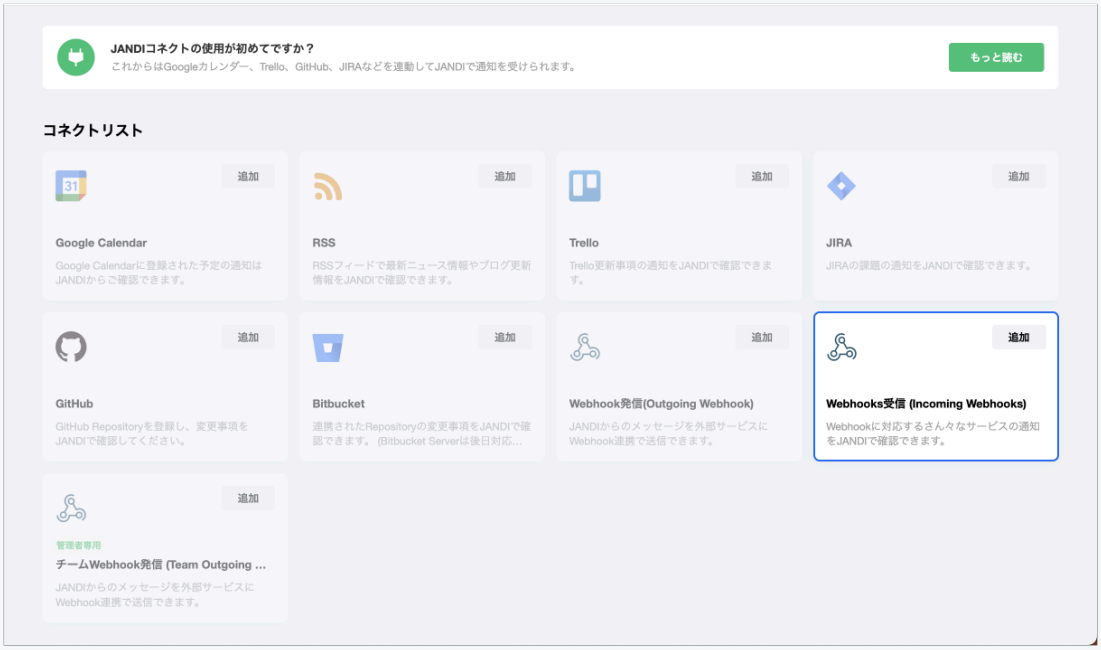
5. **名前** と **接続** の状態を確認し、[次へ] ボタンをクリックします。
6. **詳細** を確認後、[ワークフローを追加] ボタンをクリックします。
7. ワークフローが追加されました。生成された URL をコピーします。



8. WhaTap モニタリングサービスの **サードパーティプラグインの追加** 画面に戻り、**Teams Workflows** タブ下部の **Channel Name** と **Webhook URL** に値を入力します。
9. [**登録**] ボタンをクリックします。

Jandi

1. 通知を受信する **トピック** または **チャット** で、**Jandi Connect 設定** に移動します。
2. **コネクト一覧** から **Webhook 受信（Incoming Webhook）** を選択し、[連携する] ボタンをクリックします。



3. [Webhook 受信を追加] ボタンをクリックします。
4. **Webhook URL 登録** 項目に表示される Webhook URL をコピーします。



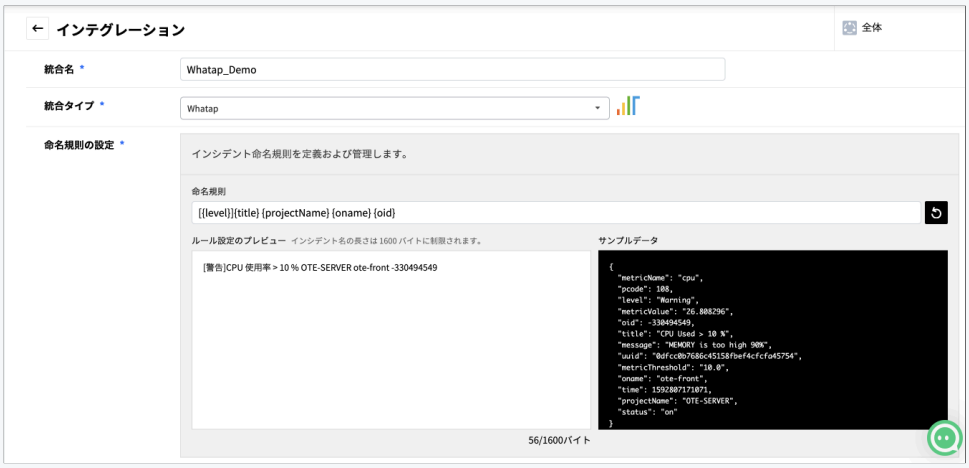
5. WhaTap モニタリングサービスの **サードパーティプラグインの追加** 画面に戻り、**Jandi** タブ下部の **Channel Name** と **Webhook URL** に値を入力します。
6. [**登録**] ボタンをクリックします。

AlertNow

1. **AlertNow** サービスにアクセスし、**設定** > **Integration** メニューに移動します。



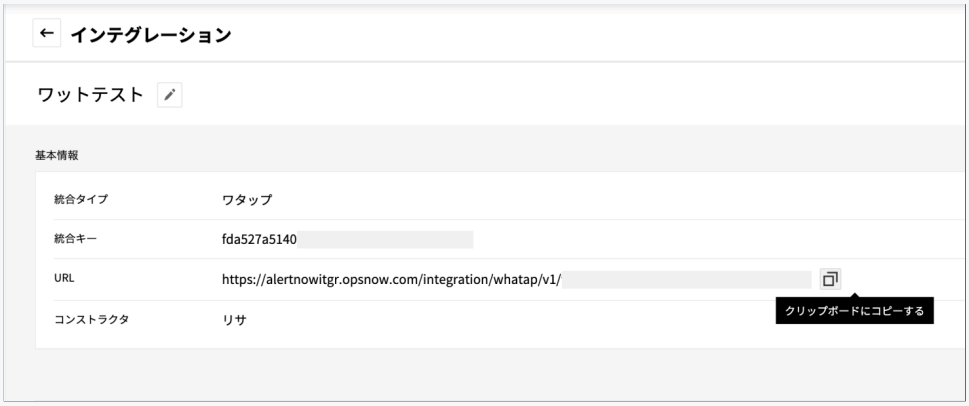
2. **Integration** 画面で [Integration 作成] ボタンをクリックします。
3. Integration タイプの中から **WhaTap** を選択します。
4. **Integration 作成** 画面で、**名前** および **命名規則** を設定します。



5. [確認] ボタンをクリックすると、Integration が作成されます。
6. 作成された Integration の一覧を確認し、該当する Integration を選択します。



7. 詳細画面で、URL 右側のコピーアイコンをクリックして URL をコピーします。



8. WhaTap モニタリングサービスの **サードパーティプラグインの追加** 画面に戻り、**AlertNow** タブ下部の **Channel Name** と **Webhook URL** に値を入力します。
9. [**登録**] ボタンをクリックします。

Webhook

1. 通知を受信する Webhook を登録します。登録した URL に対して、通知の `title` と `text` が **POST** メソッドで送信されます。

```
{
  "title":"Alert Title",
  "text": "Sample warning message"
}
```

2. **Webhook Name** と **Webhook URL** の項目を入力します。
3. [**登録**] ボタンをクリックします。

Webhook JSON

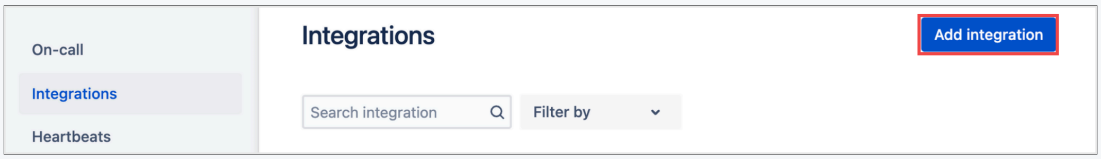
1. 通知を受信する Webhook を登録します。登録した URL に対して、通知データが **JSON 形式**で **POST** メソッドにより送信されます。JSON に含めることができる置換変数については、**使用可能な変数**を参照してください。
   - `${something}` は WhaTap が提供する置換値を表します。対応する全一覧と意味については、**使用可能な変数**を参照してください。
   - イベント状態（ `status` ）は、**イベント発生**（ `on` ）と **イベント解消**（ `off` ）に区別されます。
   - `level` は `None` 、 `Info` 、 `Warning` 、 `Critical` に区別されます。

```
{
  "pcode": "${pcode}",
  "projectName": "${projectName}",
  "time": "${time}",
  "oid": "${oid}",
  "oname": "${oname}",
  "title": "${title}",
  "message": "${message}",
  "level": "${level}",
  "status": "${status}",
  "metricName": "${metricName}",
  "metricValue": "${metricValue}",
  "metricThreshold": "${metricThreshold}",
  "uuid": "${uuid}",
  "okind": "${okind}",
  "onode": "${onode}"
}
```

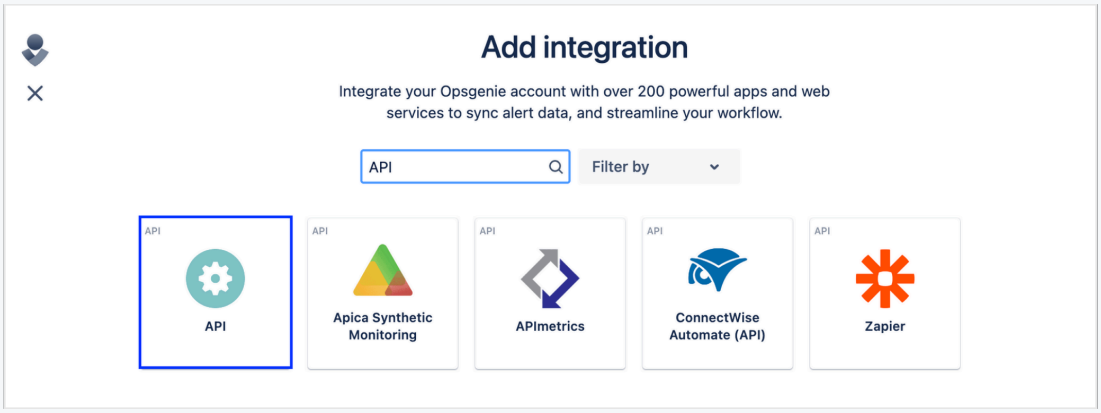
2. **Webhook Name** と **Webhook URL** の項目を入力してください。
3. ヘッダー値を追加するには、**ユーザーヘッダー値の追加** の **+ 追加** をクリックし、key と value を入力してください。
4. [**登録**] ボタンをクリックしてください。

Opsgenie

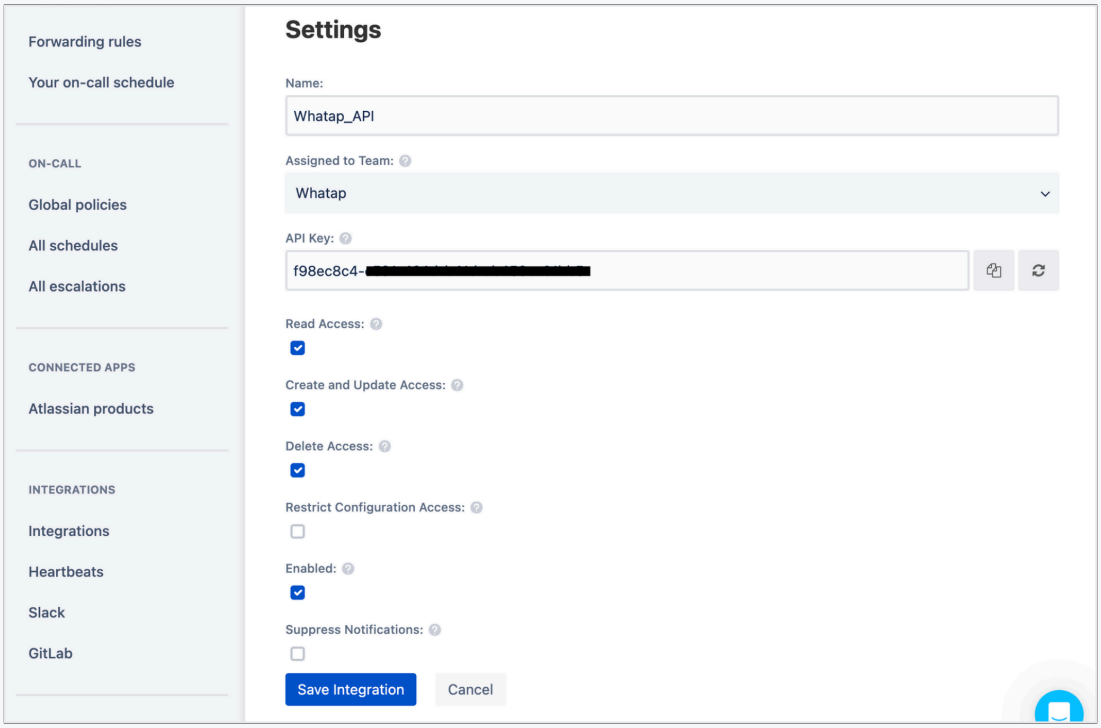
1. **Teams** メニューで、アラートを受信するチームを選択してください。
2. **Integration** メニューに移動し、[Add Integration] ボタンをクリックしてください。



3. インテグレーション一覧で **API** を検索し、選択してください。



4. API 名を設定し、チームを割り当て、各権限を設定した後、[Save Integration] ボタンをクリックしてください。



5. WhaTap モニタリングサービスの **サードパーティプラグインの追加** 画面に戻り、**Opsgenie** タブ下部の **Name** と **API Key** を入力してください。
6. [**登録**] ボタンをクリックしてください。

PagerDuty

1. PagerDuty アカウントを登録し、ログインしてください。
2. PagerDuty で **Services** > **Service Directory** メニューに移動し、[+ New Service] ボタンをクリックしてください。

   PagerDuty
3. サービス情報（**Name**）を入力し、[Next] ボタンをクリックしてください。

   PagerDuty
4. **Escalation Policy** を設定し、[Next] ボタンをクリックしてください。

PagerDuty

5. **Alert Grouping** と **Transient Alerts** を設定し、[Next] ボタンをクリックしてください。

   PagerDuty

6. **Integrations** で **Event API V2** を選択し、[Create Service] ボタンをクリックしてください。

   PagerDuty

7. 作成されたサービス画面で **Integration Name** と **Integration Key** の値をコピーしてください。

   PagerDuty

8. WhaTap モニタリングサービスの **サードパーティプラグインの追加** 画面に戻り、**Integration Name** と **Integration Key** を入力してください。
9. [登録] ボタンをクリックしてください。

> メッセージを受信できない、またはプラグインが登録されない場合は、**Integration Name** または **Integration Key** の値が正しいか確認してください。

LINE

1. LINE アプリをインストールしてください。
2. 以下の QR コードをスキャンして、**WhaTapLabs** 公式チャンネルを友だち追加してください。

   LINE 3rd パーティプラグイン追加

3. 次のコマンドを使用して、プロジェクトを LINE と連携し、通知を設定してください。

   コマンド

```
/help # 使用可能なコマンド一覧を表示
/list # 連携されているプロジェクト一覧を表示
/license [アクセスキー] # 新しいプロジェクト連携を追加
```

```
/remove [プロジェクトコード] # プロジェクト連携を解除
/language [ko|ja|en] # 通知言語を設定（韓国語／日本語／英語）
```

- プロジェクトのアクセスキーは **管理 > プロジェクト管理** で確認できます。
- LINE 内で `/language` コマンドを使用した場合、その設定は **該当する LINE チャットルーム内のみ** に適用されます。他の 3rd パーティプラグインやプロジェクトの通知言語は変更されません。

使用例

```
/license XXXXXXXXXX # プロジェクトを連携
/language ko # 言語を韓国語に設定
/list # 連携済みプロジェクトを確認
```

- **プロジェクトアクセスキー**: WhaTap プロジェクト内の **管理 > プロジェクト管理** で確認できます。
- **言語設定**: `/language` コマンドは LINE チャットルーム内でのみ有効で、他の通知チャネルの言語は変更されません。

ilert

1. ilert コンソールの **Alert sources** 画面右上にある [+ Create alert source] をクリックします。
2. **Integration type** ステップ下部の **Filter integrations** 検索欄に `WhaTap` を入力して検索します。
3. 検索結果から **WhaTap** タイルを選択し、[Next] ボタンをクリックします。
4. アラートソース名を入力し、チームを指定した後、[Next] ボタンをクリックします。
   - エスカレーションポリシーは、新規作成または既存のものを選択できます。
5. アラートのグループ化方式を選択し、[Continue setup] ボタンをクリックします。
   - 最初に選択した **Do not group alerts** は後から変更可能です。
6. 次の画面で **アラートテンプレートや優先度などの追加設定** を行うことができます。ここでは [

Finish setup] ボタンをクリックして設定を完了します。

7. 設定が完了すると、完了画面で **API キー** または **Webhook URL** が生成されます。
8. **Webhook JSON** タブで生成された **Webhook URL** をコピーし、下部の **Webhook URL** 入力欄に貼り付けます。
9. [登録] ボタンをクリックします。

大量アラート発生の防止

短時間に大量のアラートが発生した場合、指定した時間だけアラート通知を一時的に停止します。たとえば、5 分間に 20 件のイベントが発生すると、その後 5 分間は警告アラートが送信されません。設定した **停止時間** が経過すると、大量アラート防止機能は自動的に解除されます。

1. 次の画面に移動してください。**アラート通知** > **通知設定** > **大量アラート通知発生防止** セクション
2. トグルボタンをクリックして **有効化** または **無効化** します。
   - **有効**: 大量アラートを検知した場合、一時的に通知を停止
   - **無効**: アラート通知を再開
3. **検知時間, 検知回数, 停止時間** を設定してください。
   - **検知時間** 内に **検知回数** 以上のイベントが発生すると、**停止時間** の間、警告アラートが停止されます。

大量アラートによってブロックされた場合、ブロック案内ボックスが表示されます。大量アラート防止機能をすぐに解除するには、ボックス内の **大量アラート発生中止** ボタンをクリックしてください。

イベント通知フォーマット

共通アラートテンプレート

共通テンプレートは、ほとんどのケースで適用できます。さまざまな環境において、同一のフォーマットでアラートを受信できます。

- プロダクトタイプ: アプリケーション、データベース、Kubernetes
- イベントタイプ: アプリケーションアラート、データベースアラート、メトリクスアラート
- イベントチャネル: SMS、Mobile、3rd-party プラグイン、プラグイン

イベントタイトル

Event title format

```
[Level][Platform][ProjectName][ApplicationName][EventTitle]
```

Event title example

```
[Info][JAVA][アプリケーションプロジェクト][TC-0-1-8081][CRITICAL_HIGH_MEMORY]
```

ApplicationName が設定されていない場合、イベントタイトルから省略されます。

- Platform は以下のいずれかで表示されます。
  - JAVA
  - NODEJS
  - PYTHON
  - PHP
  - DOTNET
  - GO
  - POSTGRESQL

- ORACLE
- MYSQL
- MSSQL
- BSM_JAVA
- CLOUDWATCH
- TIBERO
- KUBERNETES
- KUBE_NS
- URLCHECK
- URLCHECK_ADMIN
- CUBRID
- ALTIBASE
- CLUSTER
- REDIS
- MONGODB
- VR
- RUM

イベントメッセージ

イベントメッセージに含めることができる情報です。`optional` が `false` の場合、常にメッセージに含まれます。`optional` が `true` の場合、該当データが確認できるときのみ表示されます。

イベントメッセージ例

```
Project Name : アプリケーションプロジェクト
Project Code : 3
Agent Name : TC-0-1-8081
Message : RECOVERED: Memory is too high. less than 10%
Event Time : 2022-04-12 18:53:24 +0900
```

```
Event OFF Time : 2022-04-12 18:53:24 +0900
Alert Type : APPLICATION_MEMORY
Metric Name : memory
Metric Value : 20
Metric Threshold : 10
Stateful : true
```

表 | イベントメッセージ構成要素

En	Ko	対応するアラートタイプ	説明
Project Name	プロジェクト名	全て	-
Project Code	プロジェクトコード	全て	-
Agent Name	エージェント名	全て（optional）	oname
Message	イベントメッセージ	全て	-
Alert Type	イベント種別	全て	下記の AlertType 表を参照
Event Time	イベント発生時間	全て	`2022-04-13 10:40:49 +0900` の `+0900` は GMT オフセット
Event Off Time	イベント解除時間	全て（optional）	`2022-04-13 10:40:49 +0900` の `+0900` は GMT オフセット
Metric Name	メトリクス名	全て（optional）	イベント条件判定に使用されるメトリクス名
Metric Value	メトリクス値	全て（optional）	メトリクス値が閾値を超えた場合に含まれる
Metric Threshold	メトリクス閾値	全て（optional）	メトリクス条件の閾値

En	Ko	対応するアラートタイプ	説明
Stateful	復旧イベント通知	全て（optional）	復旧通知が有効な場合 `true`、それ以外は `false`
Event Rule	イベント発生条件	メトリクスアラート	-
Target Filter	イベント対象選択	メトリクスアラート	特定対象から収集されたメトリクスのみ評価
Repeat Count	イベント繰り返し回数	メトリクスアラート	繰り返し時間内に指定回数条件を満たすと発生
Repeat Duration	イベント繰り返し時間	メトリクスアラート	繰り返し回数を評価する時間
Receiver	受信者	メトリクスアラート	-
Query	MXQL クエリ	複合メトリクスアラート	-
Rule	イベント発生条件	複合メトリクスアラート	-
Query Period	クエリ期間	複合メトリクスアラート	-
Query Interval	クエリ間隔	複合メトリクスアラート	-
Silent Time	サイレント時間	複合メトリクスアラート	-
Query	URL	Exception アラート	例外を発生させたリクエストの URL
TXID	トランザクション ID	Exception アラート	-

En	Ko	対応するアラートタイプ	説明
Class	エラークラス名	Exception アラート	-
Log Message	ログメッセージ	サーバー－ファイルログアラート	-
Log File	ログファイルパス	サーバー－ファイルログアラート	-
IP	IP	サーバーアラート全体	-
CPU	CPU	サーバーアラート全体	イベント発生時のスナップショット
CPU_load1	CPU_load1	サーバーアラート全体	イベント発生時のスナップショット
CPU_loadPerCore	CPU_loadPerCore	サーバーアラート全体	イベント発生時のスナップショット
Memory	メモリ	サーバーアラート全体	イベント発生時のスナップショット
Swap	スワップ	サーバーアラート全体	イベント発生時のスナップショット
Disk Name | Used Percent | Free Size | IO Percent	ディスクパフォーマンス	サーバーアラート全体	イベント発生時のスナップショット
Name | Bps | Pps	トラフィックパフォーマンス	サーバーアラート全体	イベント発生時のスナップショット
Message | Time | Name	処理履歴メッセージ	サーバーアラート全体	-

このセクションでは、該当するイベントで提供可能な情報を最大限に表示します。

- **AlertType** は以下のいずれかの値で表示されます。

AlertType	説明
APPLICATION_CPU	アプリケーション CPU アラート
APPLICATION_MEMORY	アプリケーション メモリアラート
APPLICATION_DISK	アプリケーション ディスクアラート
APPLICATION_ACTIVE_TRANSACTION	アプリケーション アクティブTXアラート
APPLICATION_ERROR_TRANSACTION	アプリケーション エラートランザクションアラート
APPLICATION_SLOW_TRANSACTION	アプリケーション トランザクション応答時間アラート
METRICS	メトリクスアラート
COMPOSITE_METRICS	複合メトリクスアラート
ANOMALY	異常値アラート
LOG_REALTIME	リアルタイムログアラート
COMPOSITE_LOG	複合ログアラート
SERVER_REBOOT	サーバー 再起動アラート
SERVER_NO_DATA	サーバー データ未受信アラート
SERVER_PORT	サーバー ポートアラート
SERVER_NETWORK_IOPS	サーバー ネットワーク IOPS アラート
SERVER_NETWORK_BPS	サーバー ネットワーク BPS アラート

AlertType	説明
SERVER_DISK_IO	サーバー ディスク I/O アラート
SERVER_DISK_QUOTA	サーバー ディスク使用量アラート
SERVER_DISK_INODE	サーバー inode アラート
SERVER_CPU	サーバー CPU アラート
SERVER_MEMORY	サーバー メモリアラート
SERVER_CPU_STEAL	サーバー CPU steal アラート
SERVER_MEMORY_SWAP	サーバー スワップメモリアラート
SERVER_LOG_FILE	サーバー ログファイルアラート
SERVER_WINDOW_EVENT	サーバー Windows イベントアラート
SERVER_OFF	サーバー アラート OFF 通知
SERVER_ACKNOWLEDGE	サーバー 処理確認通知
SERVER_PROCESS_COUNT	サーバー プロセス数アラート
SERVER_PROCESS_CPU	サーバー プロセス CPU アラート
SERVER_PROCESS_MEMORY	サーバー プロセス メモリアラート
SERVER_PROCESS_OFF	サーバー プロセスアラート OFF 通知
AGENT_ACTIVE	エージェント 有効化アラート
AGENT_INACTIVE	エージェント 無効化アラート

AlertType	説明
AGENT_REACTIVATED	エージェント 再有効化アラート
URL	URL アラート
TOO_MANY_EVENT	イベント多発アラート
CLOUD_WATCH	CloudWatch アラート
EXCEPTION	例外アラート

イベント履歴

ホーム > プロジェクト選択 > **アラート通知** > **イベント履歴**

イベント履歴 では、プロジェクトで発生したさまざまなイベントを確認し、詳細情報を分析できます。Elasticsearch DSL クエリをサポートしており、指定した条件に一致するイベントを迅速かつ正確に検索できます。

イベント履歴の確認

検索結果で **イベントタイトルをクリック**すると、そのイベントが発生した時間帯の詳細データを分析できる画面に移動します。これにより、イベント発生前後の状況を総合的に把握できます。

1. **アラート通知** > **イベント履歴** に移動してください。
2. **イベント履歴** 画面の **時間選択** で、イベント履歴を確認する期間を設定してください。
3. フィルター入力欄にクエリを入力し、[虫眼鏡] ボタンをクリックしてください。
    - クエリを入力しない場合、すべてのイベントが表示されます。

カラム設定

表示するカラムや非表示にするカラムを設定できます。

1. **イベント履歴** 画面の左上にある [カラム設定] ボタンをクリックしてください。
2. **カラム設定** 画面で、表示または非表示にするカラムを選択してください。
    - **表示項目** でカラムの順序を変更できます。
    - **カラム名検索** 欄で目的のカラムを検索できます。

カラム項目	説明
イベント名	イベントのタイトル
メッセージ	イベントのメッセージまたはスナップショット情報

カラム項目	説明
イベント状態	イベントの現在の状態
エージェント名	イベントが発生した対象エージェントの名前（Oname） - 特定のエージェントに紐づかないイベントの場合は空欄で表示されます
エージェントグループ	イベントが発生した対象エージェントの種類（OkindName） - 特定のエージェントに紐づかないイベントの場合は空欄で表示されます
エージェントノード	イベントが発生した対象エージェントのサーバー（OnodeName） - 特定のエージェントに紐づかないイベントの場合は空欄で表示されます
イベント発生時刻	イベントが発生した時刻
イベント解消時刻	イベントが解消された時刻 - 解消状態を持たないイベントの場合は `-` と表示されます

3. 設定が完了したら、[確認] ボタンをクリックしてカラム設定を反映してください。

CSV ダウンロード

検索結果を CSV ファイル形式でダウンロードできます。

1. **アラート通知** > **イベント履歴** で [CSV] ボタンをクリックしてください。
2. 現在のフィルター条件に一致するイベント一覧が CSV ファイルとして保存されます。

進行中のイベントのみ表示

現在進行中の状態のイベントのみを表示できます。

1. **アラート通知** > **イベント履歴** で [進行中のイベントのみ表示] ボタンをクリックしてください。
2. フィルタークエリに次の条件が自動的に追加され、検索が実行されます。

```
Stateful: true and Status : "ON"
```

イベント検索

- **正確な検索**: 必要なイベントだけを取得できるよう、条件を正確に指定
- **複合条件**: 複数のフィールドを同時に検索し、複雑な条件も一度に処理
- **柔軟なパターン**: ワイルドカードや部分一致など、さまざまな検索方式をサポート
- **高速パフォーマンス**: インデックス化されたデータを活用した高速検索

検索可能なフィールド

フィールド名および値は大文字と小文字を区別しません。

- **フィールド名タグ**: フィールドがサポートされるイベントタイプ
- **フィールドタイプ**: フィールドのデータ型
  - `#`: Number
  - `T`: String
  - : Boolean
  - : Date

イベント情報フィールド

フィールド名	タイプ	説明	例
Title	String	イベントタイトル	"Database Connection Error"
OffTitle	String	イベント解消タイトル	"RECOVERED: Database Connection Error"
Message	String	イベントメッセージ	"Connection timeout occurred"
OffMessage	String	イベント解消メッセージ	"RECOVERED: Connection timeout occurred"

フィールド名	タイプ	説明	例
Level	String	現在のイベントレベル	`Critical`, `Warning`, `Info`
OriginLevel	String	元のイベントレベル	`Critical`, `Warning`, `Info`
Status	String	イベント状態	`ON`, `OFF`, `CANCEL`, `ACKNOWLEDGE`, `MAINTENANCE`, `DISABLED`
ActCount	Number	発生回数	2
MetricName	String	指標名	"memory"
MetricValue	String	指標値	"85.5"
OffValue	String	解消時の値	"72.8"
MetricThreshold	String	閾値	"80"
alertType	String	イベント種別	"BASIC", "METRICS", "TRANSACTION" など
alertId	String	イベントルールの一意識別子	"zf3uojer0fv4v7"

イベントタイプ別対応フィールド

> ⓘ 特定のイベントタイプでのみサポートされるフィールドです。該当しないイベントタイプの場合、値が存在しないことがあります。

フィールド名	タイプ	イベント種別	説明	例
eventRule	String	基本、メト	イベント発生	"memory ≥ 80"

フィールド名	タイプ	イベント種別	説明	例
		リクス	ルール	
field	String	リアルタイムログ	ログ検索キー	"content"
keyword	String	リアルタイムログ	ログ検索値	"Error"
logCategory	String	リアルタイムログ	ログカテゴリ	"AppLog"
logContent	String	リアルタイムログ	ログ内容	"00:00000:00009:2025/04/06 23:03:52.55 server Error: 1601, Severity: 17, State: 3\n00:00000:00009:2025/04/06 23:03:52.55 server There are not enough 'user connections' available to start a new process."

対象フィールド

フィールド名	タイプ	説明	例
Oid	Number	エージェントの一意識別子	-98765432
Oname	String	エージェント名	"web-server-01"
Okind	Number	エージェント種別の一意識別子	867318026

フィールド名	タイプ	説明	例
OkindName	String	エージェント種別名	"web-server"
Onode	Number	エージェントノードの一意識別子	334634079
OnodeName	String	エージェントノード名	"production-node-1"

イベント固有値フィールド

フィールド名	タイプ	説明	例
Id	Number	イベントの一意識別子	5768121
UUID	String	イベント固有識別子	"550e8400-e29b-41d4-a716"

状態／フラグフィールド

フィールド名	タイプ	説明	例
Stateful	Boolean	状態ベースイベントかどうか	true, false
Disabled	Boolean	無効化されたイベントかどうか	true, false
Escalation	Boolean	エスカレーション有無	true, false
SystemEvent	Boolean	システムイベントかどうか	true, false

検索タイプ別クエリ構文

> ⓘ 検索の基本構文については、WhaTap ログ検索クエリ構文 ドキュメントを参照してください。

1. キーワード検索

「Database Connection」という文字列を含むイベント

```
title: "Database Connection"
```

2. 複数値検索

指定した複数の値のうち、いずれかを含むイベントを検索します。

```
title: Database Connection
```

現在のイベントレベルが Info または Warning のイベント

```
level: info warning
```

3. パターン検索

ワイルドカードを使用してパターン検索ができます。

名前パターンで特定のエージェントで発生したイベントを検索

```
oname: web-*-prod
```

タイトルが Connection で終わるイベントを検索

```
title: *Connection
```

4. 複合条件検索

複数の条件を組み合わせて、より正確に検索できます。

レベルが Critical で、タイトルに Database を含むイベント

```
level: critical and title: Database
```

特定のエージェントにおける Warning または Critical のイベント

```
oname: web-server-01 and level: Warning Critical
```

5. OR 条件検索

複数の条件のうち、**いずれか一つでも満たす**イベントを検索します。

タイトルまたはメッセージに特定のキーワードを含むイベント

```
title: Connection or message: Connection
```

6. 除外条件

特定の条件を除外して検索できます。

Info レベルを除いたすべてのイベント

```
not level: info
```

リアルタイムアラート

リアルタイムアラート 機能は、**イベント設定** で設定したイベントが発生した際に、ポップアップ形式で表示されます。イベント発生と同時に通知を受け取ることで、問題を迅速に把握し、対応することができます。また、すでに発生した通知や見逃した通知の履歴を確認することで、リアルタイム対応が難しかった状況を把握し、解決に役立てることができます。

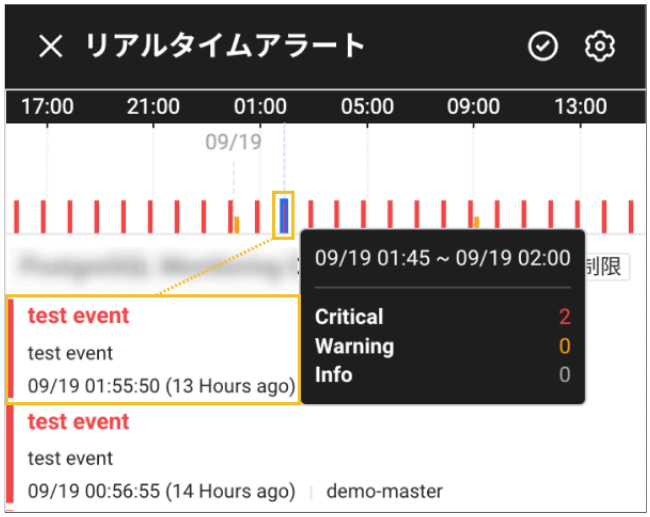
リアルタイムアラートを確認する

リアルタイムアラートは、画面右上にポップアップとして表示されます。通知が発生した時刻、タイトル、メッセージを確認できます。

- 個別の通知を閉じるには、メッセージ右上の × ボタンをクリックしてください。
- 画面に表示されているすべてのポップアップ通知を閉じるには、**すべての通知を閉じる** をクリックしてください。

通知履歴を確認する

1. プロジェクトを選択して表示される画面の最上部にある 🔔 をクリックしてください。
2. **リアルタイムアラート** ウィンドウが表示されます。
    - 現在のプロジェクトについて、直近24時間以内の通知を最大100件まで確認できます。
    - 個別のイベント項目を選択すると、関連するメニューに移動します。
    - **リアルタイムアラート** ウィンドウ内の時間別チャートで任意の時間帯を選択すると、該当するイベント項目に移動します。



- すべての通知履歴を確認するには、**全体の通知履歴** ボタンをクリックしてください。**アラート通知** > **イベント履歴** メニューに移動します。

> ⓘ **リアルタイムアラート** の一覧が空の場合、**全体の通知履歴** ボタンが表示されます。**受信タグ未設定の通知を受け取る** オプションを無効にしてください。

イベントを一括で解消する

リアルタイムアラート の一覧から、進行中のすべてのイベントを一括で解消できます。
イベント解消が完了するまでに、約30秒〜1分程度かかる場合があります。

- **イベント一括処理** 機能を使用すると、現在進行中のイベントに対する解消通知が大量に発生する可能性があります。
- **イベント一括処理** 機能を使用した後でも、イベントルールが有効であり、データが引き続きしきい値を超えている場合、イベントは再度発生することがあります。
- イベント解消通知は、**アラート通知** > **イベント設定** メニューでイベントを編集し、**イベントが解消されたときに追加通知** または **解消通知** オプションを無効にすることで停止できます。

使用方法:

1. プロジェクトを選択して表示される画面上部の をクリックします。
2. **リアルタイムアラート** ウィンドウが表示されたら、右上の をクリックします。
3. リアルタイムアラートリストの進行中イベントを一括で解消します。 というメッセージが表示されたら、**解消** ボタンをクリックします。

リアルタイムアラートを設定する

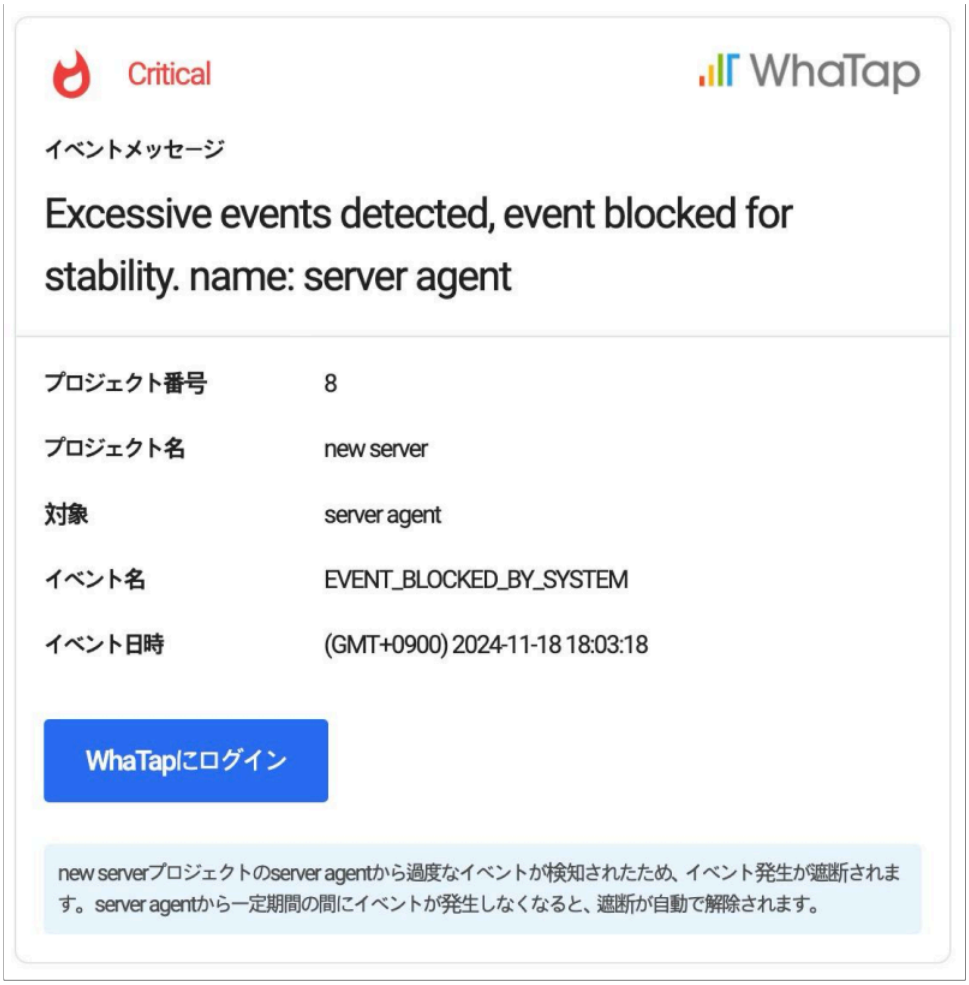
1. プロジェクトを選択して表示される画面上部の をクリックします。
2. **リアルタイムアラート** ウィンドウが表示されたら、右上の をクリックします。
3. リアルタイムアラート設定画面で各項目を設定します。
   - **ポップアップ通知**: **リアルタイムアラート** ウィンドウが閉じている状態でも、ポップアップ形式で通知を表示します。
   - **解消済みイベントを表示**: **リアルタイムアラート** 一覧に解消済みイベントも含めて表示します。
   - **通知音の試聴**: ブラウザで再生される通知音を事前に確認できます。Critical または Warning の横にある をクリックしてください。音が再生されない場合は、ブラウザの設定をご確認ください。

システムイベント

WhaTap モニタリングは、プラットフォーム全体で自動的に検知・生成される共通イベントを提供します。本ドキュメントでは、システムによって発生するイベントの種類、発生条件、および対応方法について説明します。

EVENT_BLOCKED_BY_SYSTEM

プロジェクト内で短時間に過剰な数のイベントが発生した場合、WhaTap システムがこれを検知し、イベントの発生を一時的にブロックします。イベントがブロックされると、システムは次のメッセージをユーザーに通知します。このイベントが発生した場合は、イベント設定内容を確認し、必要に応じて調整してください。



ブロック基準

イベントは次の基準に基づいてブロックされます。

1. **エージェント単位のブロック**

   1つのエージェントで **1分以内に10件** のイベントが発生した場合、そのエージェントから発生する以降のイベントはブロックされます。

2. **複数エージェントによる全体ブロック**

   条件 **1** によりイベントがブロックされたエージェントが **1分以内に3件以上** 発生した場合、プロジェクト全体のイベントがブロックされます。

3. **プロジェクト単位のブロック**

   1つのプロジェクトで **1分以内に100件** のイベントが発生した場合、プロジェクト全体のイベントがブロックされます。

ブロック解除方法

システムによってイベントがブロックされた場合、対象のエージェントまたはプロジェクトで **1分間** イベントが発生しない状態を維持する必要があります。

- **アラート通知 > イベント設定** メニューから設定されているすべてのイベントを無効化してください。**イベント設定を見直し、過剰なイベントが発生しないように閾値を調整する必要があります。**
- イベントを設定する前に、**シミュレーション** 機能を活用してイベント発生回数を事前に予測できます。

実験室

ユーザーに新機能または試験的な機能を提供します。 サイドメニューの **実験室** または **サイトマップ** からアクセスできます。 商品によって提供する機能が異なる場合があります。

サイトマップから実験室メニューにアクセスする

サイドメニューに **実験室** が表示されていない場合は、**サイトマップ** からアクセスできます。

1. 画面左下の **サイトマップ** アイコンをクリックしてください。
2. **サイトマップ** ウィンドウの右下にある **実験室** セクションで、目的のメニューを選択してください。

注意

実験室の機能は現在開発中のベータ版であり、予期せぬ挙動やエラーが発生する可能性があります。 十分な検証の上でご利用ください。 重要なデータがある環境や運用環境では使用を推奨しません。 フィードバックや問題点に対するお問い合わせは、サポートチーム(support@whatap.io)までご連絡ください。

MXQL Data

MXQLは、WhaTapの性能データ(メトリクス)を柔軟に検索するためのクエリ言語です。 プロジェクトに含まれる複数のエージェントから収集したメトリクスを統合的に照会・活用するために使います。 **MXQL**の詳細については、次のドキュメントをご参照ください。

Pending Pod状況

Podのステータスとエラーメッセージを確認してクラスターのリソースステータス、ネットワークポリシー、スケジューラログなどを検討するのに役立つ機能を提供します。

この機能は、Kubernetesマスターエージェント1.7.0バージョン以降で対応しています。

通知メッセージ

プロジェクトメンバーにメールまたはサードパーティープラグインを通じて伝える、アラート通知メッセージを修正できるユーザー定義機能を提供します。 詳しい内容は、次の文書を参照してください。