

AI Agent Observability

release date. 2026. 09. 11.

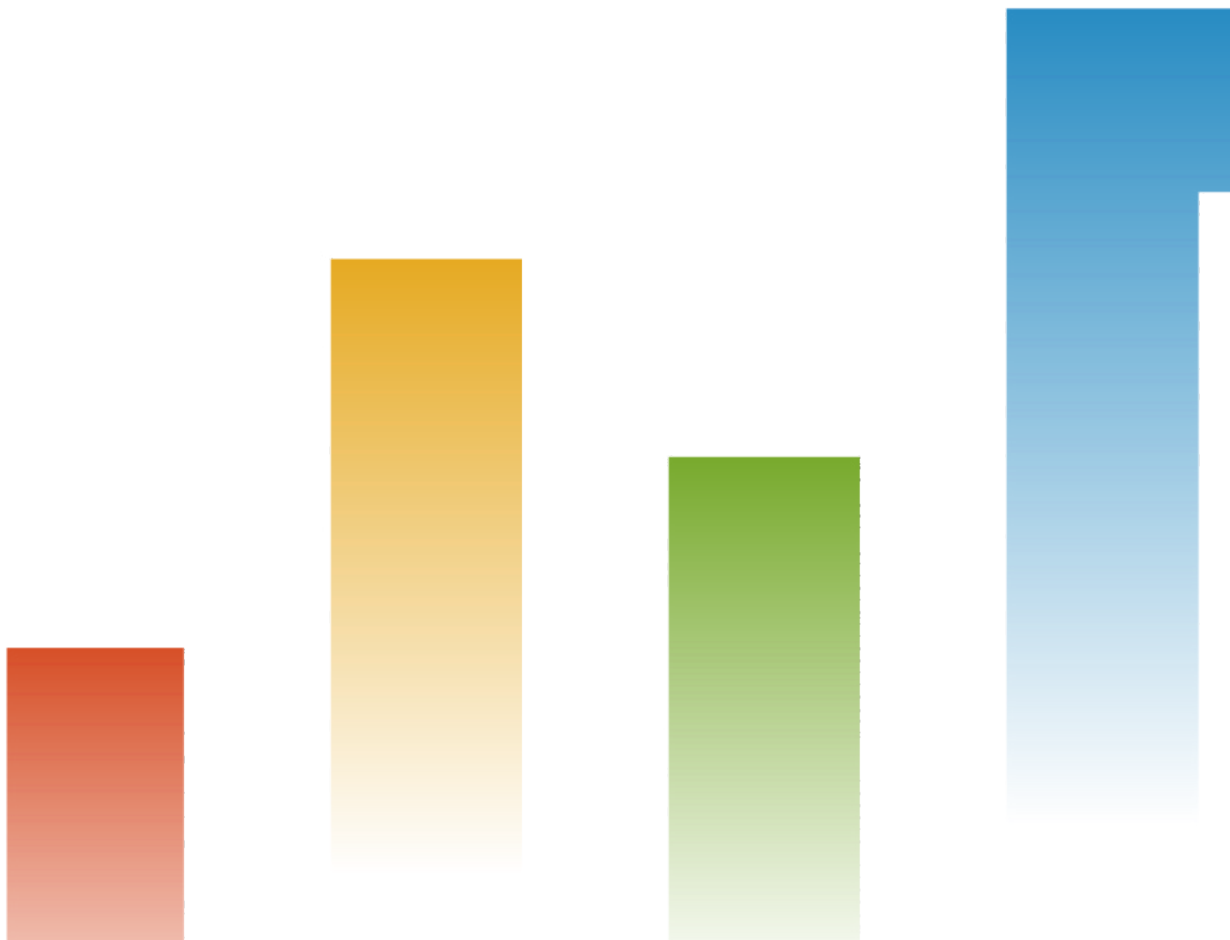


Table of Contents

 This page is currently being translated. We will complete the translation as soon as possible.

• AI Agent Observability.....	4
◦ はじめに.....	7
◦ 対応環境.....	11
◦ インストール	
▪ Python.....	18
◦ 設定	
▪ Python.....	26
▪ LLMトランザクション設定	28
▪ Evaluation.....	32
▪ Operation TypeおよびPrompt Version	39
▪ Session & User IDの設定.....	41
◦ ダッシュボード	
▪ LLMダッシュボード	43
▪ 共通機能.....	47
▪ ウィジェット.....	52
◦ 分析	
▪ トークン推移	76
▪ コスト分析.....	85
▪ LLM API分析	94
▪ プロンプト	
▪ プロンプトログ	102

▪ プロンプト分析	109
▪ プロンプト比較	116
▪ レスポンス品質分析	125
◦ メトリクス	
▪ LLMメトリクス	132
◦ OpenMetrics	
▪ テンプレートリスト	135
▪ エクスプローラー	138
◦ ログ	
▪ ライブテール	141
▪ ログエクスプローラー	153
▪ ログトレンド	160
▪ ログの検索	168
▪ ログの設定	183
◦ アラート通知	
▪ イベント設定概要	0
▪ 通知設定	190
▪ イベント履歴	212

AI Agent Observability

AI Agent Observabilityは、LLM（Large Language Model）ベースのアプリケーションの性能、コスト、安定性を統合モニタリングするプラットフォームです。LLM APIのリクエスト量、レスポンス性能、トークン使用量、コスト、エラー状況をリアルタイムで収集し、モデル・エージェント・プロバイダー単位の詳細分析を提供します。WhaTap APM、サーバー、Kubernetesインフラと組み合わせて、LLM呼び出しをアプリケーショントランザクションからGPUインフラまでエンドツーエンドで追跡できます。

LLMモニタリング、なぜ必要なのか？

推論エンジン200レスポンスの背後に隠れたLLM異常検知

LLM推論エンジンは、推論モデルがハルシネーションを生成したり異常な応答を返したりしても、HTTP 200を返します。従来のサーバーモニタリングではこの問題を検知できないため、障害認知が遅れ、対応タイミングを逃します。AI Agent Observabilityは応答時間、トークンパターン、エラー率の異常変化をリアルタイムで追跡し、HTTPステータスコードだけでは見えないモデルの異常を素早く検知します。

モデルコストが見えなければ、コントロールできません

LLM APIは呼び出すたびにトークン単位で課金されます。モデル、プロンプトの長さ、応答のサイズによって1件あたりのコストが大きく異なり、トラフィックが増加すると予測できないコストが発生する可能性があります。どのモデルが、どのリクエストで、どれだけのコストを発生させているかをリアルタイムで把握してこそ、コストをコントロールできます。エラーで失敗したリクエストにもトークンコストが発生するため、エラーコストを別途追跡して無駄な金額を定量化する必要があります。

遅いレスポンスはユーザーが最初に体感します

LLMのレスポンスは従来のAPIより数秒単位で遅くなることがあります。特にストリーミング環境で最初のトークンの到着が遅れたり、トークン生成速度が低下すると、ユーザーは「応答が止まった」と感じます。大多数のユーザーは正常なのに一部だけが遅い状況は、平均値では検知できません。どのモデルが、どの時間帯に、どのパターンで遅くなるかを時系列で追跡し、モデル間の比較まで可能にすることで、実質的な改善が可能になります。

プロンプト再現のための呼び出しコンテキスト保存

LLM呼び出しは同じプロンプトでも毎回異なる応答を生成します。問題が発生した際に「どのプロンプトで、どのモデルに、どのパラメータで呼び出したか」が保存されていなければ、再現自体が不可能です。AI Agent ObservabilityはすべてのLLM呼び出しのシステムメッセージ、入力プロンプト、モデル応答、ツール呼び出しを原本のまま収集・保存します。問題発生時に該當時点の正確な呼び出しコンテキストを復元し、「その時何が起きたか」を即座に確認・再現できます。

マルチモデル環境では比較分析が必須です

1つのアプリケーションで複数のLLMモデルとプロバイダーを同時に使用するのが一般的です。モデル別の性能、コスト、エラー率を比較してワークロードに最適なモデルを選択し、コスト対比性能が低いモデルを交替する意思決定に、データに基づく根拠が必要です。

モニタリングデータが分散していると、原因を特定できません

AIアプリケーションを運用すると、ログはログプラットフォームに、メトリクスはインフラモニタリングに、コストはプロバイダーコンソールに、トレースはAPMに断片化されます。問題が発生した際に複数のツールを行き来しながらデータを手動で関連付ける必要があるため、原因特定に時間がかかります。AI Agent Observabilityは性能、コスト、エラー、プロンプトログ、トランザクショントレース、GPUインフラを1つのプラットフォームで統合し、コンテキスト切り替えなしに問題の原因までドリルダウンできます。

AI Agent Observability主要機能

リアルタイムLLMダッシュボード

LLM APIのリアルタイムステータスを1つの画面でモニタリングします。リクエスト量、レスポンス性能、TTFT、TPOT、トークン、コスト、エラーなどの主要指標をウィジェットで提供します。

トークン推移分析およびモデル別性能比較

トークン使用パターンとLLMレスポンス性能を時系列で分析します。モデル・エージェント・プロバイダー間の性能分布を比較して、最も安定したモデルを特定できます。

コスト分析および最適化

トークン単位のコスト追跡、前期間比変動率、モデル別コスト比較を提供します。性能対比コストのバブルチャートでコスト最適化の意思決定を支援します。

キャッシュ効率モニタリング

プロンプトキャッシングのキャッシュヒット率と削減コストを時間帯別に追跡します。キャッシング戦略が実際のコスト削減に寄与しているかを定量的に確認できます。

プロンプトログ分析

LLM呼び出しのシステムメッセージ、入力プロンプト、モデル応答、ツール呼び出しを個別件単位で収集・保存します。ダッシュボードで異常を発見した後、ドリルダウンして原因を特定できます。

LLMトランザクション追跡およびGPU連携分析

LLM呼び出しをアプリケーショントランザクションの一部として追跡します。トランザクションプロファイルでLLMステップのプロンプト、トークン、コストはもちろん、GPU関連分析まで確認できます。

次のステップ

- [はじめに](#) — プロジェクト作成とエージェントタイプの選択
- [対応環境](#) — Python対応環境の確認
- [LLMダッシュボード](#) — リアルタイムモニタリングの開始

AI Agent Observabilityをはじめる

WhaTap AI Agent Observabilityを開始するには、プロジェクトを作成し、LLMアプリケーションの言語に合うエージェントタイプを選択した後、インストールガイドに従ってエージェントを適用します。エージェントを適用すると、LLMダッシュボードで収集されたデータを確認できます。

WhaTap AI Agent Observability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

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。

ステップ2. プロジェクト作成

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. [WhaTapモニタリングサービス](#)にログインします。
2. 画面左上の  **プロジェクト全体** > **プロジェクトの生成** ボタンをクリックします。
3. ❶ **商品選択画面**で**AI Observability**カテゴリを選択します。
4. **AI Agent Observability製品**の**作成**ボタンをクリックします。
5. ❷ **プロジェクト作成画面**でプロジェクト情報を入力します。

項目	説明
プロジェクト名	プロジェクトを識別できる名前を入力します。
データサーバーリージョン	データが保存されるサーバーリージョンを選択します。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	アラート、レポートに使用される基準時間を設定します。

項目	説明
通知言語の設定	アラートメッセージの言語を選択します。韓国語、英語に対応しています。
プロジェクトグループ(任意)	プロジェクトをグループに分類できます。
プロジェクト説明(任意)	プロジェクトに関する追加説明を入力できます。

6. すべての項目を入力した後、**プロジェクトを作成する**ボタンをクリックします。

❗ 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。

ステップ3. エージェントタイプの選択

プロジェクトを作成すると、自動的に❸ **エージェントインストールページ**に移動します。LLMアプリケーションの言語に合うエージェントタイプを選択します。エージェントタイプを選択すると、該当言語のインストール案内ページに移動します。

❶ 商品選択
❷ プロジェクト作成
❸ エージェントインストール

インストールするエージェントの種類を選択してください。


Python
OpenAIやAnthropicなどの主要なLLMの呼び出しを追跡し、対応フレームワークの範囲を拡大していく予定です。

[エージェント設定 >](#)


Java
Spring AIベースのLLM呼び出しを追跡し、対応フレームワークの範囲を拡大していく予定です。

[エージェント設定 >](#)


Node.js
OpenAI SDKなどの主要なライブラリをサポートしており、対応するフレームワークの範囲を拡大していく予定です。

[エージェント設定 >](#)


Others
多様な言語環境を順次サポートしていく予定です。

[エージェント設定 >](#)

エージェントタイプ	説明	サポート状況
Python	Pythonベースのアプリケーション用エージェント	サポート
Java	Javaベースのアプリケーション用エージェント	今後サポート予定
Node.js	Node.jsベースのアプリケーション用エージェント	今後サポート予定
Others	その他の言語ベースのLLMアプリケーション	今後サポート予定

❗ 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。

ステップ4. 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。エージェントインストール案内ページの最初のステップで**プロジェクトアクセスキーの発行**ボタンをクリックしてアクセスキーを発行します。

発行したアクセスキーは、エージェント設定ファイル(whatap.conf)に入力する必要があります。

ステップ5. エージェントのインストール

選択したエージェントタイプに応じてインストールを進めます。エージェントタイプ別の詳細なインストール方法は、次のドキュメントを参照してください。

- [Pythonエージェントインストール](#)

ステップ6. ダッシュボードの確認

エージェントをインストールしてアプリケーションを再起動すると、数分以内にWhaTap収集サーバーへのデータ送信が開始されます。

1. 左側のサイドバーでダッシュボード>LLMダッシュボードをクリックします。
2. LLM API呼び出しが発生すると、リクエスト量、トークン使用量、コスト、パフォーマンス指標がウィジェットに表示されます。

AI Agent Observability対応環境

このドキュメントはAI Agent Observabilityエージェントの言語別対応環境とバージョン要件を案内します。エージェントをインストールする前に、アプリケーション環境が対応範囲に該当するか確認してください。

Pythonエージェントの動作環境

WhaTap Pythonモニタリングエージェントをインストールする前に、サポート環境を確認してください。

📅 最終更新日: 2026-09-07

サポートバージョン

アプリケーションタイプ	Pythonバージョン
WSGIアプリケーション	3.7 ~ 3.13
ASGIアプリケーション	3.7 ~ 3.13

オペレーティングシステム

OS	最小バージョン	アーキテクチャ
Debian / Ubuntu	12.04	64bit
Red Hat / CentOS	6.x	64bit
macOS	10.15 Catalina	-
Windows	Windows Server 2008 R2, Windows 7	64bit

⚠ AI Agent Observabilityは現在、Windows OSに対応していません。

アプリケーションサーバー

フレームワーク	サポートバージョン
Bottle	-
Celery	-
CherryPy	-
Django	-
FastAPI	0.84.0 ~ 0.139.2
Flask	-
Frappe	-
Nameko	-
Odoo	v14 ~ v19
Tornado	-

ライブラリ

カテゴリ	ライブラリ
外部呼び出し	HttpLib, HttpLib2, Requests, Urllib, HTTPX
データ処理	SQL Alchemy, GraphQL

カテゴリ	ライブラリ
ロギング	smtplib, logging, loguru
メッセージング	Kombu, Pika
LLM SDK	OpenAI, Anthropic
Agent Orchestration	Hermes, LangGraph

データベース

データベース	ドライバー
MongoDB	pymongo
MySQL	MySQLdb, pymysql
Neo4j	neo4j
Oracle	cxOracle
PostgreSQL	psycopg2, psycopg, psycopg_pool
Redis	redis
SQLite	sqlite3

❗ `whatap-python[llm]` パッケージは、外部ライブラリ `datasketches>=5.2.0`、`genai-prices` を必要とします。

今後サポート予定

エージェントタイプ	状態
Node.js	今後サポート予定
Others (Go, PHP, .NETなど)	今後サポート予定

共通サポート環境

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

① 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。

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。

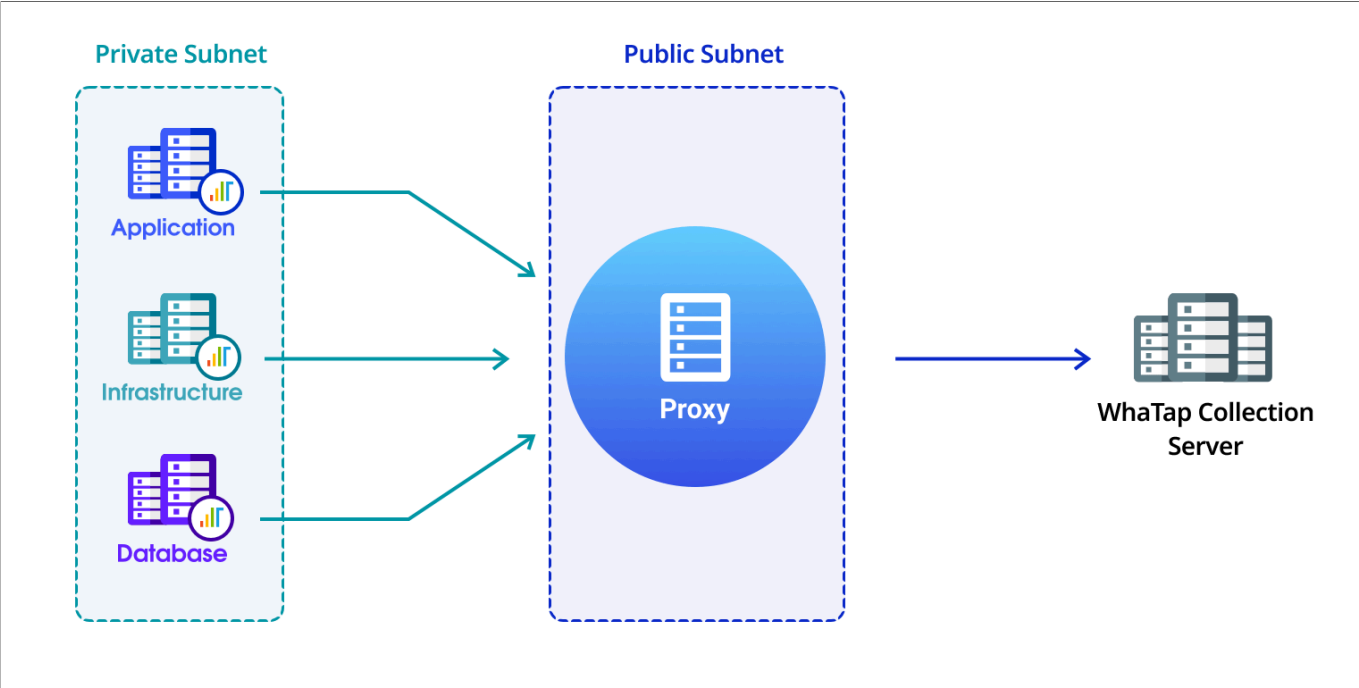
ファイアウォール

WhaTap エージェントは、収集サーバーの **TCP 6600** ポートにアクセスする必要があります。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
	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。



LLM Pythonエージェントのインストール

AI Agent Observability Pythonエージェントは `pip install whatap-python[llm]` コマンドでインストールし、サーバー環境に合わせて実行コマンドの前に `whatap-start-agent` を追加して適用します。このドキュメントは、仮想環境のアクティベートからサービス実行確認までのインストール全過程を案内します。

インストール前の確認事項

エージェントをインストールする前に、次の点を確認してください。

- [はじめに](#)ドキュメントで[プロジェクト作成](#)と[アクセスキーの発行](#)を完了しているか確認してください。
- [対応環境](#)でアプリケーション環境が対応範囲に該当するか確認してください。

1 仮想環境の有効化

アプリケーションが[virtualenv](#)を使用している場合は、`bin/activate`ファイルを実行して仮想環境を有効化してください。

2 エージェントダウンロード

アクセスキーを発行した後、エージェントダウンロードセクションに移動してください。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。

```
pip install whatap-python[llm]
```

- ✔ pipコマンドでインストールできない場合は、[pypi WhaTapページ](#)からインストールファイルをダウンロードしてください。ダウンロードしたファイルを解凍してからインストールを進めてください。

```
tar xzvf whatap_python-2.x.x.tar.gz
cd whatap_python-2.X.Y.Z
python setup.py install
```

❗ `whatap-python[llm]` パッケージは、外部ライブラリ `datasketches>=5.2.0`、`genai-prices` を必要とします。依存関係が不足すると、一部のデータ収集が欠落する可能性があります。手動インストール時にも該当の依存関係を含めてください。

エージェント構成ファイル

Pythonエージェントファイルは、アプリケーションモニタリングに必要な情報を抽出してWhaTap収集サーバーに転送するトレーサーと、それを支援する要素で構成されています。エージェントファイルの構成は以下を参照してください。

📁 `whatap`

3

エージェント設定

WHATAP_HOMEパスの設定

ログと設定ファイルのパスのために `$WHATAP_HOME` パスを指定してください。`whatap`ディレクトリを新規作成することを推奨します。

```
$ export WHATAP_HOME=[PATH]
```

アクセスキーおよび収集サーバーIPの設定

アクセスキーおよび収集サーバーIPを設定するために、`whatap-llm-setting-config` コマンドを実行してください。

```
$ whatap-llm-setting-config \  
--host [ COLLECTION_SERVER_IP ] \  
--license [ ACCESS_KEY ] \  
--app_name [ USER_DEFINED_AGENT_NAME ] \  
--app_process_name [ APPLICATION_PROCESS_NAME(uwsgi, gunicorn etc..) ]
```

設定の確認

`$WHATAP_HOME` で指定したパスに`whatap.conf`ファイルが生成・設定されます。次のコマンドを実行して`whatap.conf`ファイルが生成されたか確認してください。

```
$ cat $WHATAP_HOME/whatap.conf
```

whatap.conf

```
llm_license=[ACCESS_KEY]
llm.whatap.server.host=[COLLECTION_SERVER_IP]

llm_enabled=true

# application name
app_name=[ USER_DEFINED_AGENT_NAME ]

# middleware process name ex)uwsgi, gunicorn ..
app_process_name=[ APPLICATION_PROCESS_NAME(uwsgi, gunicorn etc..) ]
```

LLM専用モジュールのインストール

Python AI Agent Observabilityを使用するには、WhaTapがLLM専用開発したGoモジュールを追加でインストールする必要があります。

```
cd $WHATAP_HOME

curl -fsSL -O https://repo.whatap.io/python/llm/whatap-python-llm.tar.gz

tar -xzf whatap-python-llm.tar.gz
```

圧縮を解除すると生成される`whatap-python-llm`フォルダーは、先に設定した `$WHATAP_HOME` パス配下に配置する必要があります。

⚠ 権限の問題が発生した場合

- WhaTap設定のための*`$WHATAP_HOME/whatap.conf`*ファイルの読み取りおよび書き込み権限



- WhaTapログのための*\$WHATAP_HOME/logs*パスとサブファイルの読み取りおよび書き込み権限

\$WHATAP_HOME パスに対する権限の問題が発生した場合は、次のコマンドを実行して権限を付与してください。

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

4

サーバー環境別の適用

サーバー環境に合った方法を選択してください。

- [Command](#) [uWSGI](#) [Gunicorn](#) [Supervisor](#) [WSGI](#) [Docker](#) [Code](#)

Command環境では、次のように `whatap-start-agent` コマンドをアプリケーション起動コマンドの前に追加してください。

BASH

```
# $ whatap-start-agent [Application start command]
$ whatap-start-agent python manage.py runserver
```

uWSGI環境では、次のように `whatap-start-agent` コマンドを `uwsgi` コマンドの前に追加してください。

BASH

```
$ whatap-start-agent uwsgi --ini myapp.ini
```

サービスにuWSGIを登録してアプリケーションを実行する場合は、次のサンプルコードを参照してください。

/etc/init.d/uwsgi

```
description "uWSGI application server handling myapp"
start on runlevel [2345]
stop on runlevel [!2345]
exec whatap-start-agent [YOUR_APPLICATION_START_COMMAND]
# or if you are using a virtual environment
exec env WHATAP_HOME=[PATH] [ABSOLUTE_PATH]/whatap-start-agent [YOUR_APPLICATION_START_COMMAND]
```

/etc/init/uwsgi.conf

```
...
NAME="uwsgi"
PATH=/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
DAEMON=/usr/local/bin/uwsgi
PID=$RUN/$NAME.pid
INI_PATH=/etc/$NAME/$NAME.ini

##### WHATAP_AGENT_CONF #####
WHATAP_HOME=[PATH]
WHATAP_AGENT=[ABSOLUTE_PATH]/whatap-start-agent
...
do_start(){
  env WHATAP_HOME=$WHATAP_HOME $WHATAP_AGENT [YOUR_APPLICATION_START_COMMAND]
}
```

- ❗ • ユーザーを変更する場合は、WHATAP_HOME 環境変数を追加してください。
- 仮想環境を使用する場合は、エージェント起動コマンドを絶対パスで追加してください。
- サービス実行ファイル（/etc/init/uwsgi.conf）を修正して、エージェント起動コマンドとともにアプリケーションを起動してください。
- ユーザー環境によって、サービス実行ファイルのパスが異なる場合があります。

Gunicorn環境では、次のように whatap-start-agent コマンドを Gunicorn コマンドの前に追加してください。

BASH

```
$ whatap-start-agent gunicorn myapp.wsgi
```

サービスにGunicornを登録してアプリケーションを実行する場合は、次のサンプルコードを参照してください。

/etc/init.d/gunicorn

```
description "gunicorn application server handling myapp"
start on runlevel [2345]
stop on runlevel [!2345]
exec whatap-start-agent [YOUR_APPLICATION_START_COMMAND]
```

or if you are using a virtual environment

```
exec env WHATAP_HOME=[PATH] [ABSOLUTE_PATH]/whatap-start-agent [YOUR_APPLICATION_START_COMMAND]
```

/etc/init/gunicorn.conf

```
...
NAME="gunicorn"
PATH=/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
DAEMON=/usr/local/bin/gunicorn
PID=$RUN/$NAME.pid
INI_PATH=/etc/$NAME/$NAME.ini

##### WHATAP_AGENT_CONF #####
WHATAP_HOME=[PATH]
WHATAP_AGENT=[ABSOLUTE_PATH]/whatap-start-agent

...
do_start(){
  env WHATAP_HOME=$WHATAP_HOME $WHATAP_AGENT [YOUR_APPLICATION_START_COMMAND]
}
```

- ❗ ユーザーを変更する場合は、`WHATAP_HOME` 環境変数を追加してください。
- 仮想環境を使用する場合は、エージェント起動コマンドを絶対パスで追加してください。
- サービス実行ファイル（`/etc/init/gunicorn.conf`）を修正して、エージェント起動コマンドとともにアプリケーションを起動してください。
- ユーザー環境によって、サービス実行ファイルのパスが異なる場合があります。

Supervisor環境では、`supervisor.conf`ファイルに次の設定を追加してください。

supervisor.conf

```
[program:app-uwsgi]
environment = WHATAP_HOME=[PATH]
command = [ABSOLUTE_PATH]/whatap-start-agent /usr/local/bin/uwsgi --ini /home/blog/backend/config/uwsgi.ini

[program:nginx-app]
command = /usr/sbin/nginx
```

- ❗ ユーザーを変更する場合は、`WHATAP_HOME` 環境変数を追加してください。
- 仮想環境を使用する場合は、エージェント起動コマンドを絶対パスで追加してください。
- サービス実行ファイル (`/etc/supervisor/conf.d/supervisor.conf`) を修正して、エージェント起動コマンドとともにアプリケーションを起動してください。
- ユーザー環境によって、サービス実行ファイルのパスが異なる場合があります。

WSGIアプリケーションを直接実装する方法を案内します。以下を参照してください。

PYTHON

```
import whatap

@whatap.register_app
def simple_app(env, start_response):
    """Simplest possible application object"""
    status = '200 OK'
    response_headers = [('Content-type', 'text/plain')]
    start_response(status, response_headers)
    return ['Hello world!\n']
```

Dockerコンテナとしてサービスされるアプリケーションの場合、[Dockerfile](#)に以下の内容を追加してください。

1. Pythonエージェントをインストールしてください。

Dockerfile

```
ENV WHATAP_HOME /whatap
RUN mkdir -p /whatap
RUN pip install whatap-python[llm]
```

- ❗ [requirements.txt](#)を使用する場合は、該当ファイルに`whatap-python[llm]`を追加してください。

2. アクセスキーおよび収集サーバーIPの設定を登録してください。

Dockerfile

```

RUN touch /whatap/whatap.conf
RUN echo "llm_license=[ ACCESS_KEY ]" > /whatap/whatap.conf
RUN echo "llm.whatap.server.host=[ COLLECTION_SERVER_IP ]" >> /whatap/whatap.conf
RUN echo "app_name=[ AGENT_NAME ]" >> /whatap/whatap.conf
RUN echo "app_process_name=[ APPLICATION_PROCESS_NAME(uwsgi, gunicorn etc..) ]" >> /whatap/whatap.conf

```

3. 既存の実行コマンド `python manage.py runserver` の前に `whatap-start-agent` を追加してください。

Dockerfile

```

CMD ["whatap-start-agent", "python", "manage.py", "runserver", "0.0.0.0:8000"]

```

コードの先頭にAPIを直接呼び出すコードを次のように追加してエージェントを適用できます。

コードを通じてエージェントを実行する場合は、[whatap.conf](#)に次のオプションを追加してください。

whatap.conf

```

whatap_code_start=true

```

コードの先頭に次のように追加してください。

PYTHON

```

import whatap
whatap.agent()

```

アプリケーションサーバーを起動すると、エージェントがモニタリングデータの収集を開始します。

5

サービス実行の確認

次のコマンドを実行して、WhaTap Pythonサービスが正常に実行されているか確認してください。

```

ps -ef | grep whatap_python

```

Pythonエージェント設定

基本の必須有効化オプション

```
llm_license= llm.whatap.server.host=
```

PythonエージェントはAPMとLLMOBSのプロジェクトをあわせて使用できます。LLMプロジェクトに関連する値は、これらのオプションにタグ付けする必要があります。

```
llm_enabled=true
```

このオプションでLLMのログシンクパックとメトリクス収集を有効化できます。LLMOBSを使用する場合、このオプションは必ず `true` に設定されている必要があります。

LLM APIホストの認識

```
llm_api_hosts=localhost:8000,api.openai.com:443
```

自動計装が認識しないLLM APIエンドポイント（vLLM / LiteLLM / プライベートゲートウェイなど）をLLM呼び出しと判断するための設定オプションです。httpc driverが、該当ホストへの呼び出しを `LLM API` ドライバーとしてマーキングします。

モデル価格の手動入力

```
llm_model_pricing=gpt-4o|2.50|10.00|1.25,gpt-4o-mini|0.15|0.60|0.075,my-custom-model|1.00|5.00
```

モデルのトークン別の価格を手動で指定できます。設定した値は、実際のコストが認識される場合でも最優先で反映されます。

形式: `model|input_per_1m|output_per_1m|cached_per_1m`

- 4フィールド
- モデル間はカンマ区切り

- 4番目 (cached) はオプションで、空にするか省略できます

LLMトランザクション設定

AI Agent Observabilityでワークフローとエージェントのパターンを計測するための公開API利用ガイドです。

ワークフローおよびAIエージェントの領域を宣言するだけで、そのスコープ内で発生するすべてのLLM API呼び出しが別々のトランザクションではなく、1つのLLMトランザクション配下の「ステップ」としてまとめて収集されます。ユーザーが行うことは領域を示すことです。

AI Agent Observabilityのトランザクションの種類

LLMトランザクションの種類は全3種です。

- Workflow (既定) — [Workflow] <name>
- Agent — [Agent] <name>
- LLMAPI — ワークフローで包んでいない単独の呼び出しは [LLMAPI] Pure LLM API として収集されます。

ワークフローの中で別のワークフローが実行されると、親子トランザクションがトラジェクトリ (Trajectory) として連結され、呼び出し関係を追跡できます。

ワークフロートランザクションの設定

デコレーターとコンテキストマネージャーの2つの形態をどちらもサポートします。

デコレーター

ワークフロー名が関数単位で固定されている場合に使用します。

```
from whatap.llm import workflow

@workflow("support_chat")
async def support_chat(conversation_id, message, history):
    intent = await classify_intent(message) # この中のLLM呼び出しは
    docs = await retrieve(message) # すべてこのワークフローの「ステップ」になります
```

```
answer = await generate(message, docs)
return {"intent": intent, "answer": answer}
```

コンテキストマネージャー

ワークフロー名をランタイムに決定する場合や、関数の一部の区間のみを包みたい場合に使用します。

```
from whatap.llm import workflow

with workflow("ticket_triage"):
    classification = await _classify(ticket_text)
    entities = await _extract(ticket_text)
```

❗ 非同期コードでも `async with` ではなく、上記のように `with` を使用してください。 `workflow` と `agent` は同期コンテキストマネージャーであるため、スコープ内で `await` することは問題ありませんが、`async with` で開くとエラーが発生します。

エージェントトランザクションの設定

使用方法是ワークフローの設定と同一で、トランザクション名の接頭辞のみが `[Agent]` に変わります。デコレーターとコンテキストマネージャーの2つの形態をどちらもサポートします。

デコレーター

エージェントの実行関数が固定されている場合に使用します。

```
from whatap.llm import agent

@agent("react_agent")
async def run_agent(task):
    ...
    return result
```

コンテキストマネージャー

エージェント名をランタイムに決定する場合や、実行区間の一部のみを包みたい場合に使用します。

```
from whatap.llm import agent

with agent("react_agent"):
    result = await react_agent.run(task)
```

コードを修正せずに領域を宣言する

アプリケーションコードを修正できない場合は、設定でワークフローの境界を指定できます。whatap.confに対象メソッドを列挙すると、そのメソッドの実行がそのままワークフローの境界になります。ワークフロー名にはメソッド名が使用されます。

whatap.conf

```
workflow_method_patterns=myapp:RagService.run, myapp.agent:run_agent
```

モジュール:クラス.メソッド または モジュール:関数 の形式で、カンマ区切りで指定します。

自動計測対応フレームワーク

次のフレームワークは、別途の領域宣言なしでLLMトランザクションが自動的に収集されます。

フレームワーク	収集方式
hermes-agent	エージェント実行レイヤー（ <code>AIAgent.run_conversation</code> ）を計測し、 <code>[Agent] <name></code> トランザクションとして登録します。ゲートウェイインバウンド（ <code>GatewayRunner._handle_message</code> ）は <code>[Workflow] gateway/<platform></code> トランザクションとして登録され、その中で実行されたエージェントが子トランザクションとして連結されます。
LangGraph	コンパイルされたグラフの実行境界（ <code>Pregel.invoke</code> / <code>ainvoke</code> / <code>stream</code> / <code>astream</code> ）を計測し、 グラフ

フレームワーク	収集方式
	1回の実行 = LLMトランザクション1件として登録します。 tools ノードを持つグラフ（create_react_agent）は [Agent]、それ以外は [Workflow] として登録されます。サブグラフは子トランザクションとして入れ子に連結されます。

Evaluation

WhaTap AI Agent Observabilityは、運用中のLLMアプリケーションのレスポンス品質と安定性を自動で測定する **Evaluation**機能を提供します。

Evaluationは、LLMのレスポンスに対してハルシネーション、回答の関連性、有害性（toxicity）、プロンプトインジェクション、事実性（factuality）、PII漏洩、URLの含有などの評価を自動で適用する機能です。評価結果はスコア（0.0～1.0）として算出され、メトリクスとログシンクにあわせて収集されるため、レスポンス品質を時系列で追跡し、しきい値ベースのアラートを構成できます。

ⓘ Evaluation機能はPythonエージェントのみ対応しています。

⚠ 評価パイプラインを有効にすると、ユーザーへのレスポンスのためのLLM呼び出しとは別に、**judge**呼び出しの分だけLLMリソース（トークン・コスト）が追加で発生します。

設定オプション

キー	既定値	説明
llm_eval_enabled	false	評価パイプラインのマスタートグル。trueの場合、パイプラインが起動し、既定の評価器が自動登録されます。その他のオプションはすべて任意です。
llm_eval_evaluators	combined_judge, pii_leak, url_scan	エージェントで実行する評価パイプラインを選択します。

combined_judge は、judgeを1回だけ呼び出して5つの観点（hallucination、answer_relevance、toxicity、prompt_injection、factuality）を一度に推論する統合評価器です。

観点を個別に制御するには、combined_judge を外して必要なラベルのみを列挙してください。指定できる値は次のとおりです。

- LLM judgeベース: hallucination ・ answer_relevance ・ toxicity ・ prompt_injection ・ factuality
- ルールベース（LLM呼び出しなし）: pii_leak ・ url_scan

whatap.conf

```
llm_eval_evaluators=hallucination,toxicity,pii_leak
```

ただし、個別のラベルはそれぞれjudgeを呼び出すため、5つすべてを指定するとレスポンス1件当たりjudge呼び出しが5回発生します。コストの面では combined_judge の使用を推奨します。

judge呼び出しの方式

judge LLMの呼び出しには、ユーザー呼び出しで計測がキャプチャした資格情報（エンドポイント・APIキー）のスナップショットを再利用し、エージェントが自身のclientを別途生成します。ユーザーリクエストのclientインスタンスをそのまま借用しないため、リクエストごとにclientを生成・クローズするアプリケーションでも安全に動作します。

judgeモデルは、アプリケーションが使用したモデルと同一に設定されます。スナップショットされたエンドポイントがセルフホスティングやプロキシである可能性があり、クラウドの既定モデルを前提にできないためです。つまり、アプリケーションが高品質または低品質のモデルを使用すると、judge呼び出しも同じモデルで実行されます。評価モデルを調整するには、 llm_eval_judge_model にモデルを明示的に指定してください。

Evaluationの資格情報

次の場合にのみ、whatap.confにEvaluationの資格情報を直接指定します。

- スナップショットがOAuth認証のため、平文のAPIキーがない場合
- スナップショットされたエンドポイントが chat.completions パスをサービングしない場合
- 評価を意図的に別のモデル・エンドポイントに送る場合

指定方法は次の2つのモードのいずれかです。どちらのモードも、指定した時点で資格情報スナップショットの経路を完全に置き換えます。

OpenAI互換のカスタムエンドポイント

whatap.conf

```
llm_eval_judge_base_url=http://10.0.0.5:8000/v1
llm_eval_judge_api_key=EMPTY
llm_eval_judge_model=qwen2.5-7b-instruct
```

- `llm_eval_judge_provider` は省略できます。 `base_url` があればOpenAI互換とみなします。
- 呼び出しは `POST <base_url>/chat/completions` として送信されます。
- キーが不要なローカルエンジンの場合は `llm_eval_judge_api_key=EMPTY` を指定してください。

クラウドプロバイダーAPIの直接指定

OpenAIまたはAnthropicの既定エンドポイントにjudgeを送信します。

whatap.conf

```
llm_eval_judge_provider=openai
llm_eval_judge_api_key=OPENAI_API_KEY
llm_eval_judge_model=gpt-4o-mini
```

- `llm_eval_judge_provider` には `openai` または `anthropic` を指定します。
- `llm_eval_judge_base_url` は指定しません。SDKの既定エンドポイントを使用します。

コスト・パフォーマンスオプション

キー	既定値	説明
<code>llm_eval_sample_rate</code>	1.0	LLM judge評価の比率。0.1 は10%のみ評価し、コストが1/10になります。ルールベースの評価器は影響を受けず、常に100%実行されます。
<code>llm_eval_judge_timeout_sec</code>	30	judgeの1回の呼び出しの上限（秒）。0 または負の値の場合は無制限です。
<code>llm_eval_workers</code>	4	評価ワーカースレッド数。
<code>llm_eval_buffer_limit</code>	1000	評価キューの最大サイズ。超過すると評価がドロップされ、LLM030 警告が発生します。アプリケーションはブロックされません。

キー	既定値	説明
<code>llm_eval_track_judge_calls</code>	<code>false</code>	judge自体のLLM呼び出しもメトリクス・ログシンクに収集するかどうか。 <code>true</code> の場合、judgeのコスト・遅延も可視化されますが、LLM呼び出しのカウンタが増加します（ユーザー呼び出し + judge呼び出し）。このとき、judge呼び出しの <code>operation_type</code> は <code>whatap_evaluation</code> として記録されます。

特定のパイプラインにのみ適用

デコレーターとコンテキストマネージャーを使用して、特定の関数や区間のLLM呼び出しのみを評価できます。

```
from whatap.llm.evaluators import evaluate_with, evaluation_scope
from whatap.llm.evaluators.builtins import CombinedJudgeEvaluator

@evaluate_with(CombinedJudgeEvaluator())
def chat(q):
    ...

def chat2(q):
    with evaluation_scope(CombinedJudgeEvaluator()):
        ...
```

コードで登録できる評価器クラスは、whatap.confで登録する選択肢と同一です。

```
from whatap.llm.evaluators.builtins import (
    # LLM judgeベース — judge呼び出しのコストが発生
    CombinedJudgeEvaluator,
    HallucinationEvaluator,
    AnswerRelevanceEvaluator,
    ToxicityEvaluator,
    PromptInjectionEvaluator,
    FactualityEvaluator,

    # ルールベース — LLM呼び出しなし、コスト0
    PIILeakEvaluator,
```

```
URLScanEvaluator,
```

```
)
```

Evaluation指標の説明

Evaluator	LABEL	種類	スコアの方向	コスト
CombinedJudgeEvaluator	combined_judge	LLM judge	総合リスク — 低いほど良い	judge1回で5つの aspect
HallucinationEvaluator	hallucination	LLM judge	低いほど良い	judge1回
AnswerRelevanceEvaluator	answer_relevance	LLM judge	高いほど良い	judge1回
ToxicityEvaluator	toxicity	LLM judge	低いほど良い	judge1回
PromptInjectionEvaluator	prompt_injection	LLM judge	低いほど良い	judge1回
FactualityEvaluator	factuality	LLM judge	高いほど良い	judge1回

各Evaluationが測定する項目は次のとおりです。

- **answer_relevance** : 回答の関連性（高いほど良い）

レスポンスがユーザーの質問にどれだけ忠実に答えているかを測定します。質問に完全に答えている場合は 1.0、部分的・周辺的にしか答えていない場合は 0.5、質問と無関係、または回避・逸脱している場合は 0.0 と採点されます。

- **hallucination** : ハルシネーション（低いほど良い）

レスポンスに根拠のない主張が含まれているかを測定します。コンテキスト（ground truth）が与えられている場合はコンテキストに対する忠実性（faithfulness）を、コンテキストがない場合はレスポンス自体の自己一貫性

(self-consistency) を基準に評価します。0.0 はコンテキストに完全に忠実な状態、1.0 は全面的に捏造された状態です。

- **toxicity : 有害性 (低いほど良い)**

レスポンスに有害なコンテンツが含まれているかを、hate / harassment / violence / sexual / self_harm / profanity の6カテゴリを基準に判定し、検出されたカテゴリの一覧もあわせて返します。0.0 は完全に安全、1.0 は深刻に有害な状態です。

- **prompt_injection : プロンプトインジェクション (低いほど良い)**

ユーザー入力に含まれる「ignore previous instructions」のようなオーバーライドの試みが成功したか、あるいはレスポンスがシステムプロンプト・隠された指示・機密情報を漏洩したかを判定します。0.0 は本来のタスクをそのまま実行し、保護対象の情報を一切公開していない状態、1.0 はインジェクションに完全に乗っ取られた状態です。

- **factuality : 事実性 (高いほど良い)**

レスポンスの事実に正確さを測定します。歴史・科学・地理・数学的な事実や、日付・名前・数値など検証可能な客観的主張を基準とし、意見や婉曲的な表現は評価対象から除外します。1.0 はすべての事実主張が正確な状態、0.0 は明らかに虚偽の主張が多数含まれる状態です。コンテキストへの忠実性を見る hallucination とは異なり、factualityはコンテキストとは無関係にレスポンス自体の事実に正確さを見ます。

総合リスクスコア (combined_judge)

combined_judge は5つのaspectのスコアを1つの総合リスク (risk) に合算します。その際、スコアの方が異なる2つのグループをリスク基準に統一します。

- **リスク方向のaspect** (hallucination 、 toxicity 、 prompt_injection) — スコアが高いほど危険なため、スコアをそのままリスクとして使用します。
- **品質方向のaspect** (answer_relevance 、 factuality) — スコアが高いほど良いため、1 - score をリスクとして使用します。

各aspectを独立したリスクと仮定し、「すべてのaspectが安全である確率」の補数 (complement) として総合リスクを計算します。統計学のProbabilistic ORの公式です。

Probabilistic OR公式

計算例

リスクが [0.7, 0.3, 0.1, 0.1, 0.2] の場合: $1 - (0.3 \times 0.7 \times 0.9 \times 0.9 \times 0.8) = 0.864$

リスク合算の特性（シミュレーション）

個別リスク	最大値 (max)	総合 (compound)
[0.5, 0, 0, 0, 0]	0.50	0.50
[0.5, 0.5, 0, 0, 0]	0.50	0.75
[0.3, 0.3, 0.3, 0.3, 0.3]	0.30	0.83
[1.0, 0, 0, 0, 0]	1.00	1.00

総合リスクは、単純な最大値より常に大きいか等しくなります。低いリスクであっても複数のaspectにわたって蓄積されると総合リスクはその分高くなり、いずれか1つのaspectが 1.0 であれば、残りに関係なく総合リスクも 1.0 になります。これにより、単一の指標では見逃しやすい「複数の領域で同時に少しずつ悪いレスポンス」を効果的に捉えることができます。

Operation TypeおよびPrompt Version

Operation Typeは、LLM呼び出しをビジネスの文脈単位でグループ化・フィルタリングするためのユーザー定義ラベルです。指定したラベルはメトリクスタグの一部になるため、ユーザー指定の単位でさまざまな指標を分析できます。

Prompt Versionは、`operation_type` の中でプロンプトの文面が何番目の改訂版であるかを示すユーザー定義ラベルです。`operation_type` が「何をする呼び出しか」（ビジネス単位）であれば、`prompt_version` は「その呼び出しをどのプロンプトで行ったか」（改訂単位）です。

⚠ Operation TypeとPrompt Versionのラベリングは、Pythonエージェントのみ対応しています。

使用方法

(a) デコレーター（推奨）

```
from whatap.llm import prompt_meta

@prompt_meta(operation_type='checkout_chain', prompt_version='v3')
def checkout(question):
    return client.chat.completions.create(...)
```

(b) コンテキストマネージャー

```
from whatap.llm import prompt_meta_scope

def handler(req):
    with prompt_meta_scope(operation_type='greeting', prompt_version='v2'):
        return client.chat.completions.create(...)
```

現在アクティブなmetaの照会（カスタムロジックで）

```
from whatap.llm import get_prompt_meta

op_type, version = get_prompt_meta() # スcopeがない場合は('default', 'v1')を返します
```

値を指定しない場合

`operation_type` は、ラベルがない場合、囲んでいるワークフロー・エージェントの名前に自動的に置き換えられます。一方、プロンプトの文面が改訂されたかどうかはWhaTapエージェントが知り得ない情報のため、`prompt_version` は `llm_prompt_default_version` の値として記録されます。このオプションを設定しない場合は `v1` として記録されます。

プロンプトバージョンの一括修正

プロンプトを一括改訂したものの、デコレーターを一つずつ修正するのが難しい場合は、whatap.confで既定バージョンを移します。すでに明示的なラベルを付けた呼び出しは影響を受けません。

`whatap.conf`

```
llm_prompt_default_version=v2
```

設定はランタイムに反映されるため、再起動なしで次の呼び出しから新しい既定値が適用されます。

Session & User IDの設定

Session IDとUser IDを設定すると、LLM呼び出しをセッション・ユーザー単位でまとめて追跡できます。設定方法は2つあり、複数の方法が同時に適用されている場合は、優先順位に従って1つだけが反映されます。

コード（API）方式

リクエストハンドラーで `set_session_id()`、`set_user_id()` を呼び出すと、そのリクエストコンテキスト内で発生するすべてのLLM呼び出しにSession IDとUser IDが適用されます。

```
from whatap.llm import set_user_id, set_session_id

@app.post("/chat")
async def chat(req):
    set_user_id(req.user.id)      # 誰が
    set_session_id(req.conversation_id) # どの会話
    ...
```

この2つの関数はアクティブなトランザクションコンテキストに値を記録します。したがって、リクエストハンドラー内または計測されたワークフロー内で呼び出す必要があり、トランザクション外（モジュールのロード時点、ワーカー起動前など）で呼び出しても何も動作しません。戻り値（`bool`）で適用されたかどうかを確認できます。

スコープの指定

リクエストハンドラー以外の場所（バックグラウンドジョブ、バッチ）で区間を明示するには、コンテキストマネージャーを使用します。ブロックを抜けると以前の値に復元されます。

```
from whatap.llm import session, user

with user(user_id):
    with session(conversation_id):
        client.chat.completions.create(...)
```

HTTPヘッダー方式

Session IDとUser IDがHTTPリクエストヘッダーで渡される場合は、[whatap.conf](#)の設定だけでコードを修正せずに適用できます。

whatap.conf

```
llm_session_header=X-Conversation-Id  
llm_user_header=X-User-Id
```

上記の例では、`X-Conversation-Id` ヘッダーの値がSession IDとして、`X-User-Id` ヘッダーの値がUser IDとして使用されます。

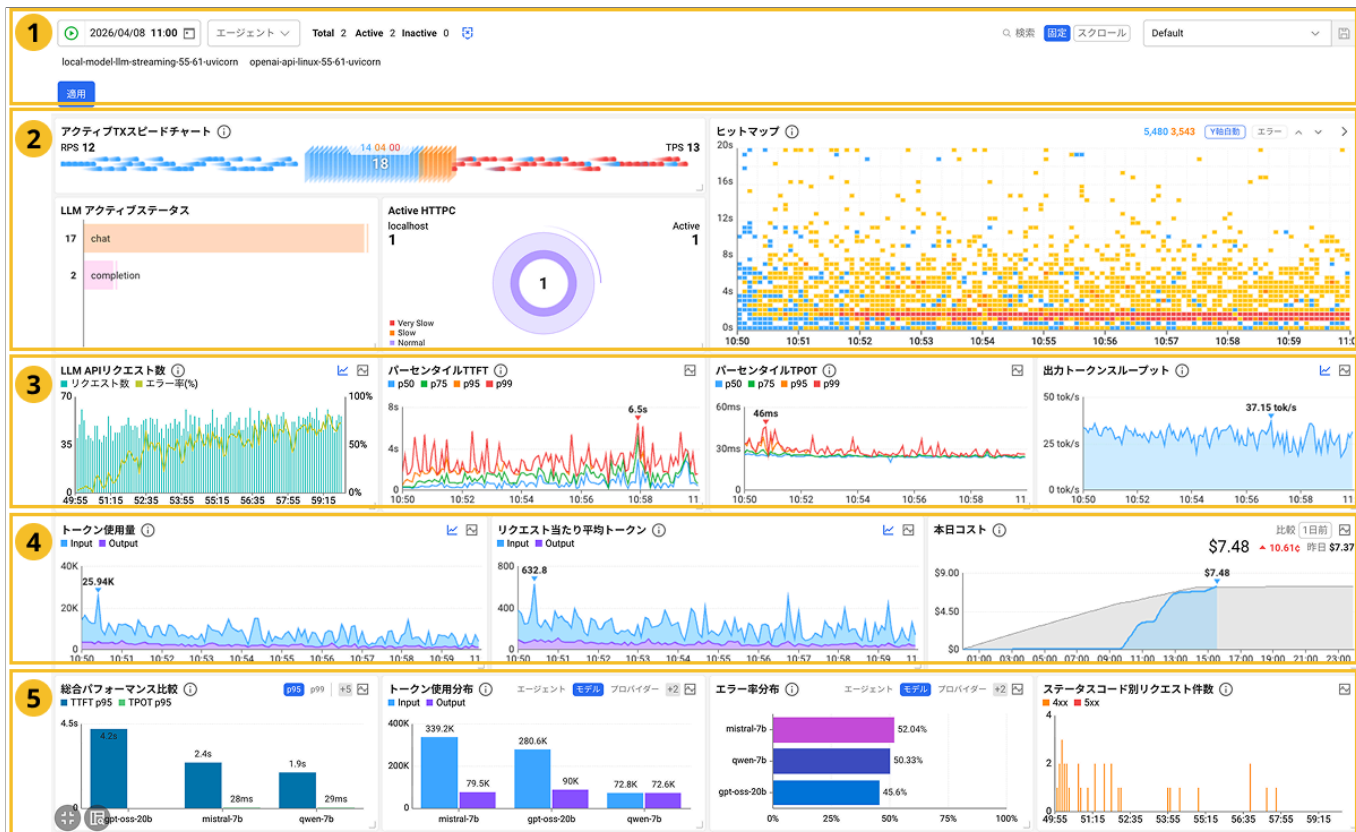
自動計測対応フレームワーク

次のフレームワークは、別途の設定なしでSession IDとUser IDが自動的に収集されます。

フレームワーク	収集方式
hermes-agent	ゲートウェイインバウンド（ <code>GatewayRunner._handle_message</code> ）でイベント情報から自動収集します。 User ID はメッセージプラットフォームのユーザー識別子（telegram、slack、whatsappなどのアダプターが設定する値）で、 Session ID はhermesが会話履歴をまとめるセッションキーをそのまま使用します。

LLM ダッシュボード

LLMダッシュボードは、LLM（Large Language Model）APIのリクエスト量、レスポンス性能、トークン使用量、コスト、エラー状況を1つの画面でリアルタイムにモニタリングするメニューです。画面の上から下へリアルタイム状態 → リクエストと性能 → トークンとコスト → 比較とエラーの順にウィジェットが配置されており、現状把握から原因分析まで自然に追跡できます。このレイアウトはデフォルトプリセットとして保存されており、追加設定なしですぐにモニタリングを開始できます。



LLMダッシュボードは、デフォルトプリセット基準で5つのエリアで構成されます。

エリア	名前	説明
①	オプションバー	照会時間、エージェントフィルタ、画面モード、プリセットを設定します。

エリア	名前	説明
②	リアルタイム状態	現在のトランザクション速度、LLMコールタイプ、アクティブHTTTPC、ヒットマップをリアルタイムで確認します。
③	LLM性能メトリクス	APIリクエスト数、TTFT・TPOTパーセンタイル、出力トークン処理量でLLMレスポンス品質を把握します。
④	トークンおよびコスト	入出力トークン使用量、リクエストあたりの平均トークン、本日のコストを追跡します。
⑤	モデル別比較・分析	モデル間の性能・トークン・エラーを比較し、ステータスコード別リクエスト件数を確認します。

オプションバー

ダッシュボード上部のオプションバーで照会時間、エージェントフィルタ、画面モード（固定/スクロール）、プリセットを設定します。各機能の詳細な使い方は[ダッシュボード共通機能](#)を参照してください。

リアルタイム状態

アプリケーションのリアルタイム状態を表示します。

- **アクティブTXスピード** — 1秒あたりのリクエスト数（RPS）とレスポンス数（TPS）、進行中のトランザクション件数をリアルタイムで表示します。
- **LLMアクティブステータス** — 現在処理中のLLMコールをタイプ別（`chat`、`completion` など）に分類して表示します。
- **アクティブHTTTPC** — 外部LLM APIへのHTTPリクエストのリアルタイム状態を速度区間（Very Slow、Slow、Normal）で分類します。
- **ヒットマップ** — 完了したトランザクションのレスポンス時間を時間軸（X）と経過時間軸（Y）にドットで表示します。ドットが上にあるほどレスポンスが遅いトランザクションです。

LLM性能メトリクス

LLM APIの主要な性能メトリクスをまとめて確認します。

- **LLM APIリクエスト数** — 時間帯別のリクエスト件数（バー）とエラー率（ライン）を合わせて表示します。リクエストの急増とエラー率の上昇が同時に発生すると、Rate Limit超過やプロバイダーの障害を疑うことができます。
- **パーセンタイルTTFT** — 最初のトークンが到着するまでの時間（Time To First Token）をp50・p75・p95・p99パーセンタイルで表示します。TTFTが高いと、ユーザーが体感する初期待機時間が長くなります。
- **パーセンタイルTPOT** — トークン1つを生成するのにかかる時間（Time Per Output Token）をパーセンタイルで表示します。TPOTが高いとストリーミングレスポンスが途切れたり遅くなったりする可能性があります。
- **出力トークン処理量** — 1秒あたりの出力トークン数（tok/s）を表示します。処理量が急激に低下するとプロバイダー側の性能低下を疑うことができます。

トークンおよびコスト

トークン使用量とコストを追跡します。

- **トークン使用量** — 時間帯別の入力（Input）・出力（Output）トークン使用量をトレンドチャートで表示します。
- **リクエストあたりの平均トークン** — リクエスト1件あたりの平均入力・出力トークン数を表示します。プロンプト最適化の効果を確認する際に役立ちます。
- **本日のコスト** — 本日の累計コストと前日比の増減を表示します。コストの異常兆候を迅速に検知できます。

モデル別比較・分析

モデル別の性能・トークン・エラーを比較します。

- **総合性能比較** — モデル別TTFT p95、TPOT p95を1つのチャートで比較します。モデルの切り替えやバージョンアップグレードの効果を判断する際に役立ちます。
- **トークン使用分布** — モデル別の入力・出力トークン使用量をエージェント、プロバイダー、オペレーション、エンドポイント基準で比較します。
- **エラー率分布** — モデル別のエラー率をエージェント、プロバイダー、オペレーション、エンドポイント基準で比較

します。特定モデルにエラーが集中しているかを素早く確認できます。

- **ステータスコード別リクエスト件数** — HTTPステータスコード（4xx、5xx）別のリクエスト件数を時間帯別に表示します。

① ウィジェット編集、ウィジェットオプション、プリセットなどダッシュボードの共通機能は[ダッシュボード共通機能](#)を参照してください。

各ウィジェットのチャートタイプ、フィルタ、活用方法は[LLMダッシュボードウィジェット](#)を参照してください。

LLMダッシュボード共通機能

ダッシュボードで共通で利用できる機能を案内します。照会時間とエージェントフィルタでデータ範囲を絞り込み、ウィジェットオプションでチャート进行操作し、プリセットでウィジェット構成を保存・管理できます。

照会時間設定

ダッシュボードではリアルタイムモニタリング機能を基本的に提供しますが、画面上部のタイムセクターを利用して過去時間のデータを照会できます。タイムセクターでIIボタンを選択して時間を設定します。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。

エージェント確認

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。この機能で、モニタリング対象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

- ✔ 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**をクリックします。

エージェントを検索する

プロジェクトに接続されたエージェントの一覧が多く、見つけるのが難しい場合は、エージェント検索機能を利用します。画面の左上にある🔍検索ボタンをクリックすると、文字列を入力するボックスが表示されます。

入力した文字列に一致するエージェントのみが、エージェント一覧に表示されます。検索したエージェントに基づいてダッシュボードのデータをフィルタリングするには、文字入力ボックスの右側にある**選択する**ボタンをクリックします。

選択したエージェント項目を初期化するには、文字入力ボックスの右側の**閉じる**ボタンをクリックします。

ウィジェットオプション

ウィジェット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。

❗ 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。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。

プリセット設定

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したい項目の右側に **×** ボタンをクリックします。

ⓘ **Default**プリセットは削除できません。

基本プリセットを変更する

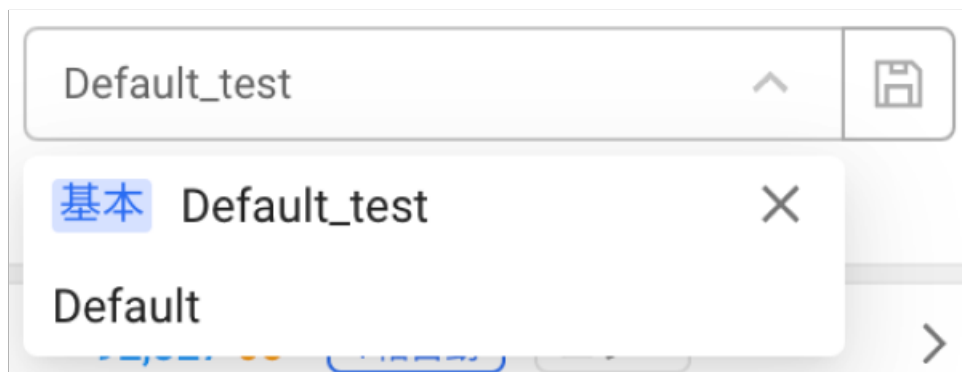
アプリケーションメニューに進入した時に表示されるプリセットを変更できます。

1. ダッシュボードの右上にあるプリセット選択ボックスを選択します。
2. 基本に設定するプリセットに、マウスをオーバーします。



3. デフォルトで表示ボタンが表示されたら選択します。

選択したプリセットが基本プリセットに変更され、プリセット一覧に**基本**タグが表示されます。



リアルタイムアラート

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。

① 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。

LLMダッシュボードウィジェットリファレンス

このドキュメントはLLMダッシュボードで提供するウィジェットをダッシュボードエリア別に整理したリファレンスです。各ウィジェットのチャートタイプ、フィルター、活用方法を確認できます。ダッシュボード全体の画面構成とデフォルトプリセットは[LLMダッシュボード](#)ドキュメントを参照してください。

提供ウィジェット

LLM性能メトリクス

ウィジェット	説明
LLM APIリクエスト数	時間帯別リクエスト件数 + エラー率
ステータスコード別リクエスト件数	HTTPステータスコード（2xx/4xx/5xx）分布
Latency平均	時間帯別平均レスポンス時間推移
Latencyパーセンタイル	Latency p50・p75・p95・p99推移
Latencyパーセンタイル個別ウィジェット	パーセンタイル別個別ウィジェット
Latency分布比較	モデル別Latencyボックスプロット
TTFT平均	時間帯別TTFT平均推移
TTFTパーセンタイル	TTFT p50・p75・p95・p99推移
TTFTパーセンタイル個別ウィジェット	パーセンタイル別個別ウィジェット
TTFT分布比較	モデル別TTFTボックスプロット
TPOT平均	時間帯別TPOT平均推移

ウィジェット	説明
TPOTパーセンタイル	TPOT p50・p75・p95・p99推移
TPOTパーセンタイル個別ウィジェット	パーセンタイル別個別ウィジェット
TPOT分布比較	モデル別TPOTボックスプロット
総合性能比較	モデル別TTFT・TPOT・Output TPS比較
出力トークン処理量	1秒あたりの出力トークン数 (tok/s)

トークンおよびコスト

ウィジェット	説明
トークン使用量	時間帯別入力・出力トークン推移
リクエストあたりの平均トークン	リクエスト1件あたりの平均入力・出力トークン
トークン使用分布	モデル別入力・出力トークン総量比較
本日のトークン使用量	本日の累計トークン + 前日比
キャッシュヒット率	キャッシュトークン比率 (%) 推移
キャッシュヒット率分布	モデル別キャッシュヒット率比較
キャッシュ削減コスト	キャッシングによる節約金額 (\$) 推移
キャッシュ削減コスト分布	モデル別キャッシュ削減コスト比較
コスト使用量	時間帯別入力・出力コスト (\$) 推移
リクエストあたりの平均コスト	リクエスト1件あたりの平均コスト

ウィジェット	説明
コスト使用分布	モデル別コスト総量比較
本日のコスト	本日の累計コスト + 前日比

モデル別比較・分析

ウィジェット	説明
総合性能比較	モデル別TTFT・TPOT・Output TPS比較
トークン使用分布	モデル別入力・出力トークン総量比較
エラー率分布	モデル別エラー率（%）比較
エラー件数	時間帯別APIエラー・プログラムエラー件数
ステータスコード別リクエスト件数	HTTPステータスコード（4xx/5xx）分布

ライブ観測

ウィジェット	説明
アクティブLLM API	実行中のLLM API呼び出しと長時間実行リスト
アクティブトラジェクトリ	進行中のTrajectory数と等級別分布

提供ウィジェット詳細説明

LLM APIリクエスト

LLM APIのリクエスト量とステータスコードをモニタリングするウィジェットグループです。

LLM APIリクエスト数

Property	Value
チャートタイプ	バーチャート（リクエスト数） + ラインチャート（エラー率）複合
フィルタ	タグフィルタ
アクション	個別に見る、マージして見る、詳細を見る

LLM APIに送信された時間帯別リクエスト件数とエラー率（%）を合わせて表示します。バーチャート（左側Y軸）はリクエスト量、ライン（右側Y軸）は全体リクエストに対するエラー比率です。

- リクエスト量が急増すると同時にエラー率も上昇する場合、Rate Limit超過またはプロバイダー障害の可能性あります。
- リクエスト量に変化がないのにエラー率のみ上昇する場合、特定モデルやエージェントで問題が発生している可能性があるため、**個別に見る**で原因を絞り込みます。

ステータスコード別リクエスト件数

Property	Value
チャートタイプ	スタックバーチャート
フィルタ	なし
アクション	詳細を見る

時間帯別HTTPステータスコード分布を2xx（成功）、4xx（クライアントエラー）、5xx（サーバーエラー）に分けて表示します。

- **4xx増加** — 不正なリクエストパラメータ、認証失敗、トークン上限超過など、リクエスト側の問題を点検する必要があります。
- **5xx増加** — LLMプロバイダー側のサーバー障害や一時的な過負荷状態を意味します。
- **LLM APIリクエスト数ウィジェットのエラー率と合わせて見ると、エラー原因を素早く分類できます。**

LLM APIレスポンス性能

LLM APIの全体レスポンス時間（Latency）をさまざまな観点で分析するウィジェットグループです。Latencyはリクエストを送信した時点からレスポンスが完全に完了するまでの全体所要時間です。

Latency平均

Property	Value
チャートタイプ	ラインチャート
フィルタ	タグフィルタ
アクション	個別に見る、マージして見る、詳細を見る

時間帯別Latency平均推移を表示します。

- Latency平均が全体的に上昇する場合、モデルのレスポンス速度低下やネットワーク遅延を点検する必要があります。
- **個別に見る**でモデル別Latencyを比較し、どのモデルで遅延が発生しているかを確認できます。

Latencyパーセンタイル

Property	Value
チャートタイプ	ラインチャート（p50、p75、p95、p99の4シリーズ）
フィルタ	なし

Property	Value
アクション	詳細を見る

時間帯別Latencyのパーセンタイル推移をp50、p75、p95、p99の4つのシリーズで表示します。例えば、p95の値が3秒であれば、全体リクエストの95%が3秒以内に完了したことを意味します。

- p50とp99の差が大きい場合、大部分のリクエストは速いものの、一部のリクエストで深刻な遅延が発生していることを意味します。
- p50とp99が一緒に上昇する場合、全体的な性能低下を意味します。

Latencyパーセンタイル個別ウィジェット

Property	Value
チャートタイプ	ラインチャート
フィルタ	タグフィルタ
アクション	個別に見る、マージして見る、詳細を見る

各パーセンタイルを個別ウィジェットで提供します。各パーセンタイルの意味は次のとおりです。

Percentile	Description
p50	全体リクエストの半分がこの時間以内に完了。一般的なユーザーが体感する平均的なレスポンス速度。
p75	全体リクエストの75%がこの時間以内に完了。p50との差で上位25%リクエストの追加遅延を把握。
p95	全体リクエストの95%がこの時間以内に完了。SLA（サービスレベル合意）基準メトリクスとして頻繁に使用。
p99	全体リクエストの99%がこの時間以内に完了。最も遅い上位1%のリクエスト、システム安定性の判断。

Latency分布比較

Property	Value
チャートタイプ	ボックスプロット（x軸：モデル）
フィルタ	タグフィルタ（常時表示）
アクション	詳細を見る

タグ基準（モデルなど）別にLatencyの分布をボックスプロットで比較します。ボックス領域はp25～p75（中間50%範囲）、中央線は中央値、ウィスカーは最小値～最大値を表します。

- ボックスが広いモデルはレスポンス時間のばらつきが大きく、安定性の改善が必要な場合があります。
- ウィスカーが長く伸びたモデルは、間欠的に極端な遅延が発生しています。
- 複数モデルの分布を並べて比較し、最も安定したモデルを特定できます。

TTFT（Time To First Token）

TTFTはリクエストを送信した後、最初のレスポンストークンが到着するまでの時間です。TTFTが長くなると、ユーザーは「レスポンスが始まらない」という体感を受けます。

TTFT平均

Property	Value
チャートタイプ	ラインチャート
フィルタ	タグフィルタ
アクション	個別に見る、マージして見る、詳細を見る

時間帯別TTFT平均推移を表示します。

- 特定の時間帯にTTFTが急騰する場合、モデルキュー待機の増加やプロバイダー側の遅延を点検する必要があります。

- 個別に見るでどのモデルの初期レスポンスが遅いかを比較できます。

TTFTパーセンタイル

Property	Value
チャートタイプ	ラインチャート（p50、p75、p95、p99の4シリーズ）
フィルタ	なし
アクション	詳細を見る

時間帯別TTFTのパーセンタイル推移をp50、p75、p95、p99の4つのシリーズで表示します。例えば、p95の値が2秒であれば、全体リクエストの95%が2秒以内に最初のトークンを受信したことを意味します。

- p50は安定しているのにp99だけ急騰する場合、間欠的にキュー待機が発生しているシグナルです。
- すべてのパーセンタイルが一緒に上昇する場合、全体的な初期レスポンス遅延が発生しています。

TTFTパーセンタイル個別ウィジェット

Property	Value
チャートタイプ	ラインチャート
フィルタ	タグフィルタ
アクション	個別に見る、マージして見る、詳細を見る

各パーセンタイルを個別ウィジェットで提供します。

Percentile	Description
p50	大部分のユーザーが体感する初期レスポンス待機時間。
p75	p50との差で上位25%リクエストの追加初期遅延を把握。

Percentile	Description
p95	「大部分のユーザーにこの程度の初期レスポンス待機は保証されます」というSLA基準メトリクス。
p99	最も遅い上位1%の初期レスポンス時間。間欠的なタイムアウトや特定条件での遅延を点検。

TTFT分布比較

Property	Value
チャートタイプ	ボックスプロット（x軸：モデル）
フィルタ	タグフィルタ（常時表示）
アクション	詳細を見る

タグ基準（モデルなど）別にTTFTの分布をボックスプロットで比較します。ボックス領域はp25～p75、中央線は中央値、ウィスカーは最小値～最大値です。

- 中央値が低くボックスが狭いモデルが、最も安定して速い初期レスポンスを提供します。
- ウィスカーが長く伸びたモデルは、間欠的に極端な初期遅延が発生します。

TPOT（Time Per Output Token）

TPOTは出力トークン1つが生成されるのにかかる平均時間です。TPOTが低いほどストリーミングレスポンスが滑らかに感じられ、高くなると「レスポンスが途切れる」感じを与える可能性があります。

TPOT平均

Property	Value
チャートタイプ	ラインチャート
フィルタ	タグフィルタ
アクション	個別に見る、マージして見る、詳細を見る

時間帯別TPOT平均推移を表示します。

- 特定の時間帯にTPOTが急騰する場合、モデル負荷の増加やRate Limitの影響を点検する必要があります。
- 個別に見るでモデル別のトークン生成速度を比較できます。

TPOTパーセンタイル

Property	Value
チャートタイプ	ラインチャート（p50、p75、p95、p99の4シリーズ）
フィルタ	なし
アクション	詳細を見る

時間帯別TPOTのパーセンタイル推移をp50、p75、p95、p99の4つのシリーズで表示します。例えば、p95の値が40msであれば、全体リクエストの95%でトークン1つが40ms以内に生成されたことを意味します。

- p50は安定しているのにp99だけ高くなる場合、間欠的にトークン生成が停止する現象が発生しているシグナルです。
- すべてのパーセンタイルが一緒に上昇する場合、全体的なストリーミング速度の低下を意味します。

TPOTパーセンタイル個別ウィジェット

Property	Value
チャートタイプ	ラインチャート
フィルタ	タグフィルタ
アクション	個別に見る、マージして見る、詳細を見る

各パーセンタイルを個別ウィジェットで提供します。

Percentile	Description
p50	一般的なストリーミング速度。最も滑らかなストリーミングを提供するモデルを確認。
p75	p50との差で上位25%リクエストのストリーミング速度低下を把握。
p95	SLA基準ストリーミング速度メトリクス。
p99	最も遅い上位1%のトークン生成速度。間欠的な停止現象を点検。

TPOT分布比較

Property	Value
チャートタイプ	ボックスプロット（x軸：モデル）
フィルタ	タグフィルタ（常時表示）
アクション	詳細を見る

タグ基準（モデルなど）別にTPOTの分布をボックスプロットで比較します。ボックス領域はp25～p75、中央線は中央値、ウィスカーは最小値～最大値です。

- 中央値が低くボックスが狭いモデルが、最も安定したストリーミング性能を提供します。
- ウィスカーが長く伸びたモデルは、間欠的にトークン生成が大幅に遅くなる現象が発生します。

総合性能比較

Property	Value
チャートタイプ	水平バーチャート（モデル別、3つのメトリクス）
フィルタ	タグフィルタ（常時表示）
アクション	詳細を見る、p95/p99タブ切り替え

モデル別の主要性能メトリクス3つを水平バーチャートで比較します。p95/p99タブを切り替えて、一般的な性能と最悪条件での性能をそれぞれ確認できます。

Metric	Description	Direction
TTFT	ユーザーが最初のレスポンスを受け取るまでの待機時間	低いほど良い
TPOT	トークン1つが生成されるのにかかる時間（ストリーミング速度）	低いほど良い
Output TPS	1秒あたりに生成される出力トークン数（全体処理量）	高いほど良い

3つのメトリクスを総合すると、「速く開始し、滑らかにストリーミングし、高い処理量を出すモデル」を一目で特定できます。

トークン

トークン使用パターンとキャッシュ効率をモニタリングするウィジェットグループです。

トークン使用量

Property	Value
チャートタイプ	ラインチャート（input、outputの2シリーズ）
フィルタ	タグフィルタ
アクション	個別に見る、マージして見る、詳細を見る

時間帯別の入力トークン数と出力トークン数の推移を表示します。入力トークンはプロンプトに含まれるトークン、出力トークンはモデルが生成したレスポンストークンです。

- 特定の時間帯にトークン使用量が急増する場合、トラフィック増加、プロンプト変更、異常リクエストの流入を点検する必要があります。
- **個別に見る**でどのモデルやエージェントにトークン消費が集中しているかを確認できます。

リクエストあたりの平均トークン

Property	Value
チャートタイプ	ラインチャート（input、outputの2シリーズ）
フィルタ	タグフィルタ
アクション	個別に見る、マージして見る、詳細を見る

リクエスト1件あたりに使用された平均入力/出力トークン数の時間帯別推移を表示します。

- リクエストあたりの入力トークンが急増する場合、プロンプトの長さが長くなったか、長いコンテキストが含まれ始めた可能性があります。
- リクエストあたりの出力トークンが急増する場合、レスポンスの長さが増加したことを意味し、コスト増加に直接影響します。

出力トークン処理量

Property	Value
チャートタイプ	ラインチャート
フィルタ	タグフィルタ
アクション	個別に見る、詳細を見る

1秒あたりに生成される出力トークン数（Tokens Per Second）を示す処理量メトリクスです。

- 処理量が低下する区間は、モデルレスポンスのボトルネックやRate Limitの影響を点検する必要があります。
- トークン使用量と合わせて見ると、「トークンは多く使用しているが処理速度が遅い区間」を特定できます。

トークン使用分布

Property	Value
チャートタイプ	ヒストグラム（タグ別グループ）
フィルタ	タグフィルタ（常時表示）
アクション	詳細を見る

タグ基準（モデルなど）別に入力/出力トークン総量を比較する分布チャートです。

- どのモデルが最も多くのトークンを消費しているかを一目で確認できます。
- 入力の比重が高いモデルは、プロンプト最適化によるコスト削減の余地があります。
- 出力の比重が高いモデルは、レスポンス長さの制限（`max_tokens`）の調整を検討できます。

本日のトークン使用量

Property	Value
チャートタイプ	日次比較チャート（本日/前日）
フィルタ	なし
アクション	日付比較

本日の累計トークン使用量（input + output）と前日同時間帯比の増減を表示します。

- 前日比でトークン消費速度が速い場合、トラフィック増加やプロンプトサイズの変化を点検する必要があります。
- Rate Limit上限や日次予算超過を事前に検知するのに活用できます。

キャッシュヒット率

Property	Value
チャートタイプ	ラインチャート (%)
フィルタ	タグフィルタ
アクション	個別に見る、詳細を見る

全体入力トークン中、キャッシュから取得したトークンの比率 (%) を時間帯別に表示します。

- ヒット率が高いほど、プロンプトキャッシングが効果的に動作しており、コスト削減効果が大きいです。
- ヒット率が急激に低下する場合、プロンプト内容の変更、キャッシュTTLの満了、新しいタイプのリクエスト流入を点検する必要があります。
- **個別に見る**でモデル別のキャッシュ効率を比較できます。

❗ キャッシュヒット率 = (区間内の `cached_tokens` 合計 / 区間内の `input_tokens` 合計) × 100

キャッシュヒット率分布

Property	Value
チャートタイプ	ヒストグラム (タグ別)
フィルタ	タグフィルタ (常時表示)
アクション	詳細を見る

タグ基準 (モデルなど) 別にキャッシュヒット率 (%) を比較する分布チャートです。

- ヒット率が高いモデルはキャッシングが十分に活用されており、低いモデルはキャッシング対象のプロンプト構造を点検する必要があります。
- モデル間のヒット率格差が大きい場合、キャッシング戦略をモデル別に差別化して追加コストを削減できます。

キャッシュ削減コスト

Property	Value
チャートタイプ	ラインチャート（\$）
フィルタ	タグフィルタ
アクション	個別に見る、詳細を見る

プロンプトキャッシングにより実際に節約された金額（\$）を時間帯別に表示します。キャッシュされたトークンは通常の入力トークンより低い単価が適用され、この差が削減コストとして算出されます。

- 削減コストが安定して発生していれば、キャッシング戦略が効果的に動作しているという意味です。
- 削減コストが減少する場合、キャッシュヒット率と合わせてキャッシュミスの増加有無を点検する必要があります。
- **個別に見る**でどのモデルで最も大きな削減効果が発生しているかを確認できます。

キャッシュ削減コスト分布

Property	Value
チャートタイプ	ヒストグラム（タグ別）
フィルタ	タグフィルタ（常時表示）
アクション	詳細を見る

タグ基準（モデルなど）別にキャッシュ削減コスト（\$）を比較する分布チャートです。

- 削減コストが大きいモデルはキャッシング効果が高いため、そのモデルのキャッシング設定を維持・強化することが有利です。
- 削減コストが低いモデルは、キャッシュヒット率と合わせてキャッシング対象のプロンプト構造の改善やキャッシュTTLの調整を検討できます。

コスト

LLM API使用コストをモニタリングするウィジェットグループです。

コスト使用量

Property	Value
チャートタイプ	ラインチャート（input、outputの2シリーズ）
フィルタ	タグフィルタ
アクション	個別に見る、マージして見る、詳細を見る

時間帯別の入力/出力コスト（\$）推移を表示します。入力コストはプロンプト処理コスト、出力コストはレスポンス生成コストです。

- コストが急増する時間帯がある場合、該当区間のリクエスト量とリクエストあたりのトークン数を合わせて確認する必要があります。
- 入力コストと出力コストの比重を比較して、プロンプト最適化とレスポンス長さ制限のどちらに集中すべきか判断できます。
- **個別に見る**でモデル別のコスト寄与度を比較できます。

リクエストあたりの平均コスト

Property	Value
チャートタイプ	ラインチャート（input、outputの2シリーズ）
フィルタ	タグフィルタ
アクション	個別に見る、マージして見る、詳細を見る

リクエスト1件あたりに発生する平均入力/出力コスト（\$）の時間帯別推移を表示します。

- リクエストあたりのコストが急変する場合、モデル切り替え、プロバイダーの価格変更、プロンプトサイズの変化

を点検する必要があります。

- 総コストは増加しているがリクエストあたりのコストに変動がない場合、単純にリクエスト量が増加したもので、リクエスト量の管理に集中すれば良いです。

コスト使用分布

Property	Value
チャートタイプ	ヒストグラム（タグ別グループ）
フィルタ	タグフィルタ（常時表示）
アクション	詳細を見る

タグ基準（モデルなど）別に入力/出力コスト総量を比較する分布チャートです。

- どのモデルが最も高いコストを占めているかを一目で確認できます。
- コストが高いモデルの使用頻度とリクエストあたりのトークンを点検して、コスト最適化対象を選定できます。

本日のコスト

Property	Value
チャートタイプ	日次比較チャート（本日/前日）
フィルタ	なし
アクション	日付比較

本日の累計コスト（\$）と前日同時間帯比の増減を表示します。

- 前日比でコスト消費速度が速い場合、リクエスト量の増加、モデル切り替え、プロンプトサイズの変化を点検する必要があります。
- 日次予算超過を事前に検知して対応措置を取ることができます。

エラー

LLM APIのエラー状況をモニタリングするウィジェットグループです。

エラー率分布

Property	Value
チャートタイプ	水平バーチャート
フィルタ	タグフィルタ
アクション	個別に見る、マージして見る、詳細を見る

タグ基準（モデルなど）別にエラー率（%）を水平バーチャートで比較します。エラー率は、該当グループの全体リクエスト中エラー（`status_code` 400以上）が占める比率です。

- 特定モデルやエージェントにエラーが集中しているかを素早く把握できます。
- エラー率が高いモデルは、プロバイダー障害、不正なパラメータ、Rate Limit超過などを点検する必要があります。
- 個別に見るで `Agent`、`Provider` などさまざまな基準でエラー原因を絞り込めます。

エラー件数

Property	Value
チャートタイプ	スタックバーチャート
フィルタ	なし
アクション	詳細を見る

時間帯別のエラー件数をAPIエラーとプログラムエラーに区分して表示します。

- **APIエラー** — LLMプロバイダー側から返されたエラーで、Rate Limit超過、サーバーエラー、認証失敗などが該当します。

- **プログラムエラー** — クライアント側で発生したエラーで、レスポンスパース失敗、タイムアウト、接続エラーなどが該当します。
- エラータイプ別の推移を通じて、問題の原因がプロバイダー側かクライアント側かを素早く区別できます。

ライブ観測

現在実行中のLLM呼び出しとTrajectoryを確認するウィジェットグループです。Trajectoryは、1つのLLM作業フローに参加したトランザクションを1つにまとめた単位です。2つのウィジェットのリスト画面は、エージェントから直接照会した実行中データを使用するため、過去区間はありません。ダッシュボードを停止している場合や過去の時刻を表示している場合は、リストにつながる導線を提供しません。

アクティブLLM API

Property	Value
チャートタイプ	スウォームチャート（モデル別トラック）
フィルタ	ダッシュボードエージェントフィルタ
アクション	詳細を見る、新しいウィンドウで開く

現在実行中のLLM API呼び出しを、モデル別のトラック上にドットで表示します。1つのドットが1つの呼び出しであり、横位置は **Normal**、**Slow**、**Very Slow** の区間を表します。3つの区間の基準は、エージェントオプションに設定した値に応じてデータが集計されます。

- **Very Slow** 区間にドットが積み上がる場合は、エージェントに設定した基準より長く実行される呼び出しが特定のモデルやプロバイダーに偏っていないかを確認してください。
- ウィジェット本文をクリックすると、長時間実行リストがモーダルで開きます。プロンプトの展開により実行中のシステムプロンプトと入力プロンプトを確認でき、リストで呼び出しを選択すると、その呼び出しが属するTrajectoryビューに移動して実行フロー全体を確認できます。
- ウィジェットヘッダーの**詳細を見る**ボタンをクリックすると、同じリストを広い画面で開けます。モーダルヘッダーの**新しいウィンドウで開く**ボタンをクリックすると、別のウィンドウで表示します。

アクティブトラジェクトリ

Property	Value
チャートタイプ	軌道チャート+等級別件数リスト
フィルタ	ダッシュボードエージェントフィルタ
アクション	詳細を見る、新しいウィンドウで開く

現在アクティブなTrajectory数と、各TrajectoryのStep数を **Light**、**Middle**、**Heavy** の等級に分けて表示します。等級はStepをどれだけ多く、繰り返し呼び出したかを表すものであり、速度や実行時間を意味しません。

- 所属するマルチランザクションのいずれか1つでも実行中であれば、そのTrajectoryをライブ状態として数えます。
- 等級の境界値は、エージェントオプションに設定した基準に従います。
- 等級をクリックすると、その等級で絞り込んだTrajectoryリストが開きます。リストで行を選択すると、Trajectoryビューにつながります。
- 軌道チャートをクリックすると、等級で絞り込まないリスト全体が開きます。

Trajectoryリスト

リストは、Trajectory単位のグループヘッダーと、その下のランザクション行で構成されます。グループヘッダーには **mtid**、まとめられたランザクション数、ユーザー、セッションを表示し、**mtid** のない単独フローは単独実行として表示します。リストはCSVでエクスポートできます。

Column	Description
Transaction	ランザクション名
ステータス	ランザクションの進行状態（実行中・完了・失敗）
mtid	Trajectoryをまとめるキー
ステップ	ランザクションが実行したステップ数

Column	Description
エージェント	トランザクションを実行したエージェント
LLM呼び出し数	トランザクション内で発生したLLM呼び出し数
経過	トランザクション開始後の経過時間
ユーザー	ログに付与された <code>user_id</code>
セッション	ログに付与された <code>session_id</code>

リファレンス

収集元フィールド

Field	Description
<code>request_count</code>	LLM APIコール件数
<code>status_code</code>	HTTPレスポンスステータスコード（2xx、4xx、5xx）
<code>latency</code>	リクエスト開始～レスポンス完了までの全体時間（ms）
<code>ttft</code>	Time To First Token。リクエスト開始～最初のトークン受信（ms）
<code>tpot</code>	Time Per Output Token。出力トークン間の平均間隔（ms）
<code>input_tokens</code>	全体入力トークン数（cached含む）
<code>output_tokens</code>	出力トークン数
<code>cached_tokens</code>	キャッシュから取得した入力トークン数（ <code>input_tokens</code> のサブセット）
<code>input_tokens_cost</code>	全体入力トークンコスト（\$）。 <code>cached_tokens_cost</code> 含む

Field	Description
output_tokens_cost	出力トークンコスト (\$)
cached_tokens_cost	キャッシュトークンコスト (\$)
error_type	エラータイプ (api_error、program_error)

ウィジェットグループサマリー

Group	Widgets	Description
LLM APIリクエスト	2	リクエスト量とステータスコードのモニタリング
LLM APIレスポンス性能	8	Latencyの推移と分布分析
TTFT	7	最初のトークン受信時間の分析
TPOT	7	トークン生成速度の分析
トークン	9	トークン使用パターンとキャッシュ効率の分析
コスト	4	LLM APIコストのモニタリング
エラー	2	エラータイプ別状況分析
ライブ観測	2	実行中のLLM呼び出しとTrajectoryの状況把握
合計	41	

用語

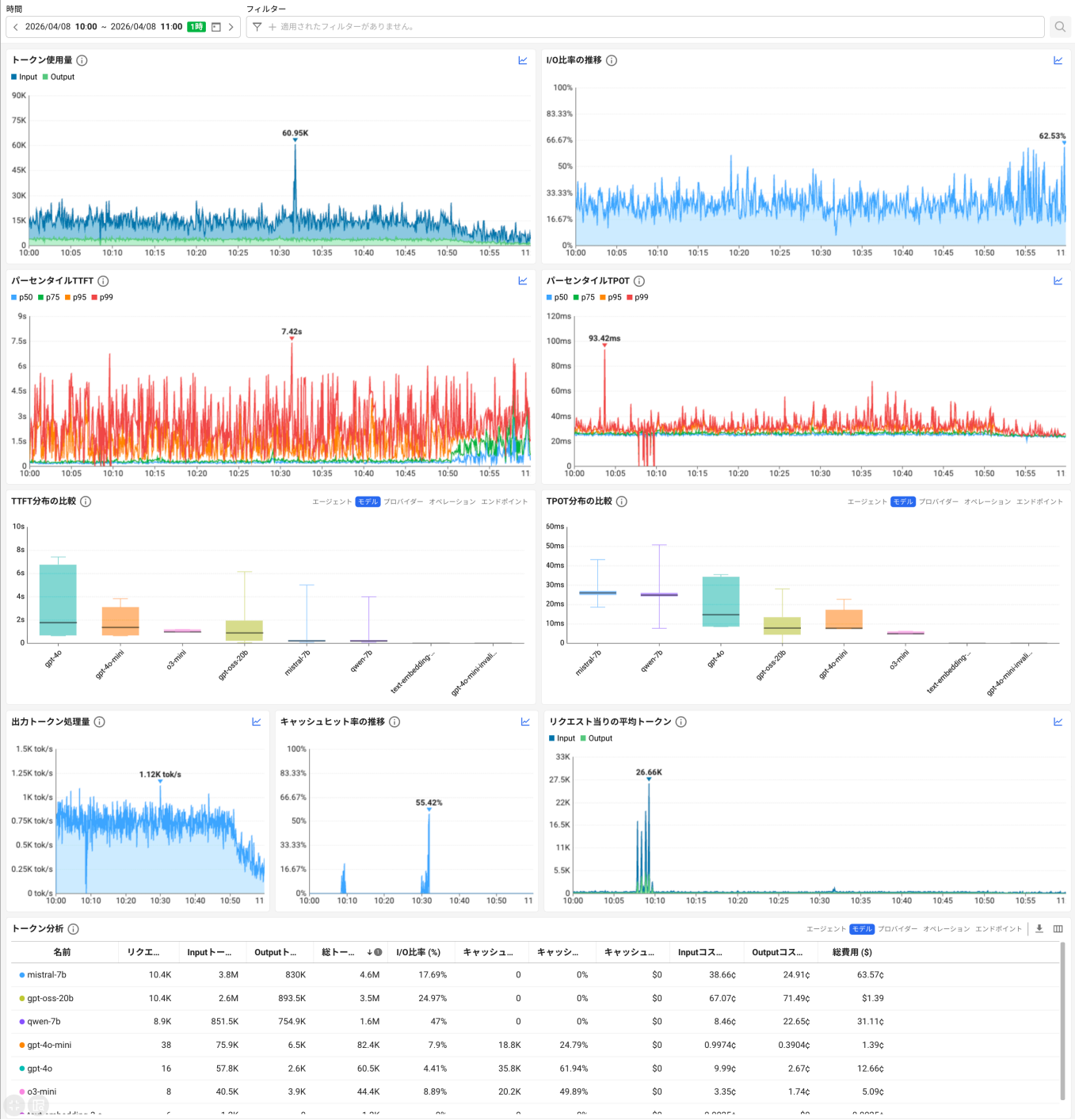
Term	Description
Latency	LLM APIリクエストを送信した時点からレスポンスが完全に完了するまでの全体所要時間 (ms)

Term	Description
TTFT	Time To First Token。リクエストを送信した後、最初のレスポンストークンが到着するまでの時間 (ms)
TPOT	Time Per Output Token。出力トークン1つが生成されるのにかかる平均時間 (ms)
Output TPS	Tokens Per Second。1秒あたりに生成される出力トークン数 (tokens/s)
パーセンタイル	データを大きさ順に並べたとき、該当パーセント以下にある値。例：p95 = 全体データの95%がこの値以下
キャッシュヒット率	全体入力トークン中、キャッシュから取得したトークンの比率 (%)。値が大きいほどコスト削減効果が大きい
タグフィルタ	Agent、Model、Provider、Operation 基準でデータをフィルタリングする機能
プリセット	ウィジェット構成（種類、位置、サイズ）を保存した設定。ユーザー定義プリセットの作成可能
Trajectory	1つのLLM作業フローに参加したトランザクションを mtid 基準でまとめた単位

LLMトークン推移分析

トークン推移はLLM APIのトークン使用パターンと処理効率を時系列ベースで分析するメニューです。ダッシュボードがリアルタイム状態のモニタリングに焦点を当てているのに対し、このページは**区間別推移比較**と**モデル/エージェント間の詳細分析**に活用されます。

上部オプションバーで時間範囲、フィルター、検索条件を設定してウィジェットデータを照会します。チャートドラッグで区間を絞り込んだり、チャート/ボックスプロットのクリックでプロンプトログページに移動できます。



トークン使用量

Property	Value
チャートタイプ	ラインチャート（Input、Output 2つのシリーズ）
フィルター	タグフィルター
アクション	個別表示、マージ表示、詳細表示

時間帯別のInput（入力）トークン数とOutput（出力）トークン数の推移を表示します。Inputトークンはプロンプトに含まれるトークン、Outputトークンはモデルが生成した応答トークンです。

- 特定の時間帯にトークン使用量が急増した場合、トラフィック変化、プロンプト変更、または異常リクエストの流入を確認してください。
- InputとOutputの比率変化を通じて、ワークロード特性（入力中心 vs 生成中心）の変化を把握できます。
- 個別表示で、どのモデルやエージェントにトークン消費が集中しているかを確認できます。

I/O比率推移

Property	Value
チャートタイプ	ラインチャート（%、エリア塗りつぶし）
フィルター	タグフィルター
アクション	個別表示、マージ表示、詳細表示

I/O比率は全体トークン（Input + Output）のうちOutputトークンが占める割合（%）です。50%基準線（点線）が合わせて表示されます。

- **50%以上:** 応答生成の比率が高いワークロードです。要約、コード生成など出力の長いタスクが多い状態です。
- **50%以下:** 入力比率が高いワークロードです。分類、エンベディング、RAG検索などプロンプトの長いタスクが多い

状態です。

- 比率が急変した場合、ワークロードパターンやプロンプト構造が変更された可能性があるため確認してください。

❗ I/O比率 = $\text{output_tokens合計} / (\text{input_tokens合計} + \text{output_tokens合計}) \times 100$

TTFTパーセンタイル

Property	Value
チャートタイプ	ラインチャート (p50、p75、p95、p99 4つのシリーズ)
フィルター	タグフィルター
アクション	個別表示、マージ表示、詳細表示、パーセンタイル選択

TTFT (Time To First Token) のパーセンタイル推移を表示します。トークン使用量が急増する区間でTTFTも合わせて上昇する場合、トークン規模の増加が初期応答遅延に影響を与えていることを意味します。

- p50は安定しているのにp99だけ急騰する場合、断続的にキュー待ちやプロバイダ遅延が発生しているシグナルです。
- 異常区間を発見した場合、TTFT分布比較でどのモデルが原因かを確認できます。

TPOTパーセンタイル

Property	Value
チャートタイプ	ラインチャート (p50、p75、p95、p99 4つのシリーズ)
フィルター	タグフィルター
アクション	個別表示、マージ表示、詳細表示、パーセンタイル選択

TPOT（Time Per Output Token）のパーセンタイル推移を表示します。TTFTパーセンタイルと合わせて確認すると、「初回トークンも遅く生成も遅い区間」と「初回トークンは速いが生成が遅い区間」を区別できます。

- p99が急騰する場合、断続的にトークン生成が停止したり大幅に遅くなるリクエストが存在することを意味します。
- 異常区間を発見した場合、TPOT分布比較でどのモデルが原因かを確認できます。

TTFT分布比較

Property	Value
チャートタイプ	ボックスプロット（x軸: モデル）
フィルター	タグフィルター（常時表示）
アクション	詳細表示

タグ基準（モデルなど）別にTTFTの分布をボックスプロットで比較します。ボックス領域はp25～p75（中間50%範囲）、中央線は中央値、ウィスカーは最小値～最大値を表します。

- 中央値が低くボックスが狭いモデルが、最も安定して速い初期応答を提供します。
- ウィスカーが長く伸びているモデルは、断続的に極端な初期遅延が発生します。
- モデルをクリックすると、該当モデルのプロンプトログに移動します。

TPOT分布比較

Property	Value
チャートタイプ	ボックスプロット（x軸: モデル）
フィルター	タグフィルター（常時表示）
アクション	詳細表示

タグ基準（モデルなど）別にTPOTの分布をボックスプロットで比較します。

- 中央値が低くボックスが狭いモデルが、最も安定したストリーミング性能を提供します。
- ウィスカーが長く伸びているモデルは、断続的にトークン生成が大幅に遅くなる現象が発生します。

出力トークンスループット

Property	Value
チャートタイプ	ラインチャート
フィルター	タグフィルター
アクション	個別表示、マージ表示、詳細表示

1秒あたりに生成される出力トークン数（Tokens Per Second）を表すスループット指標です。

- スループットが低下する区間は、モデル応答のボトルネックまたはRate Limitの影響を確認してください。
- トークン使用量と合わせて確認すると、「トークンは多く使用しているが処理速度が遅い区間」を識別できます。
- 個別表示でモデル別のスループットを比較できます。

キャッシュヒット率推移

Property	Value
チャートタイプ	ラインチャート（%、エリア塗りつぶし）
フィルター	タグフィルター
アクション	個別表示、マージ表示、詳細表示

全入力トークンのうちキャッシュから取得したトークンの割合（%）を時間帯別に表示します。

- ヒット率が高いほどプロンプトキャッシングが効果的に機能しており、コスト削減効果が大きくなります。

- ヒット率が急に低下した場合、プロンプト内容の変更、キャッシュTTLの期限切れ、または新しいタイプのリクエスト流入を確認してください。
- 個別表示でモデル別のキャッシュ効率を比較し、どのモデルでキャッシングが最も効果的かを確認できます。

❗ キャッシュヒット率 = (cached_tokens合計 / input_tokens合計) x 100

リクエスト当たり平均トークン

Property	Value
チャートタイプ	ラインチャート (Input、Output 2つのシリーズ、エリア塗りつぶし)
フィルター	タグフィルター
アクション	個別表示、マージ表示、詳細表示

リクエスト1件あたりの平均Input/Outputトークン数の時間帯別推移を表示します。

- リクエスト当たりInputトークンが急増した場合、プロンプト長が長くなったか、長いコンテキストが含まれ始めた可能性があります。
- リクエスト当たりOutputトークンが急増した場合、応答長が伸びたことを意味し、コスト増加に直接影響します。
- トークン使用量が増加する際にこのウィジェットの数値も合わせて増加する場合はリクエスト当たりのサイズが大きくなったことを意味し、トークン使用量だけ増加する場合は単純にリクエスト数が増えたことを意味します。

トークン分析

Property	Value
チャートタイプ	テーブル
フィルター	Agent / Model / Provider タブ切り替え

Property	Value
アクション	詳細表示、CSVダウンロード、カラム設定

タグ基準（Agent、Model、Provider）別にトークン使用量、キャッシュ、コストの主要指標をテーブルで整理します。

Column	Description
Name	タグ値（エージェント名、モデル名、プロバイダ名）
Requests	リクエスト件数
Total Input	入力トークン総量
Total Output	出力トークン総量
Total Tokens	全体トークン総量
I/O Ratio (%)	出力トークン比率
Cached Tokens	キャッシュトークン総量
Cache Hit (%)	キャッシュヒット率
Cached Cost (\$)	キャッシュトークンコスト
Input Cost (\$)	入力コスト
Output Cost (\$)	出力コスト
Total Cost (\$)	全体コスト

- Total Tokens が高い項目は全体コストで最も大きな割合を占めるため、最適化の優先対象です。
- I/O Ratio が高い項目は出力中心のワークロードで、Outputトークン単価の影響を大きく受けます。
- Cache Hit が低い項目はキャッシング戦略を確認し、コスト削減の余地を検討してください。

- **Total Cost** を基準にソートすると、コスト最適化の優先順位を素早く把握できます。
- 右上のダウンロードボタンで現在のデータをCSVにエクスポートできます。
- カラム設定で表示するカラムを選択できます。

参考

ダッシュボードとの違い

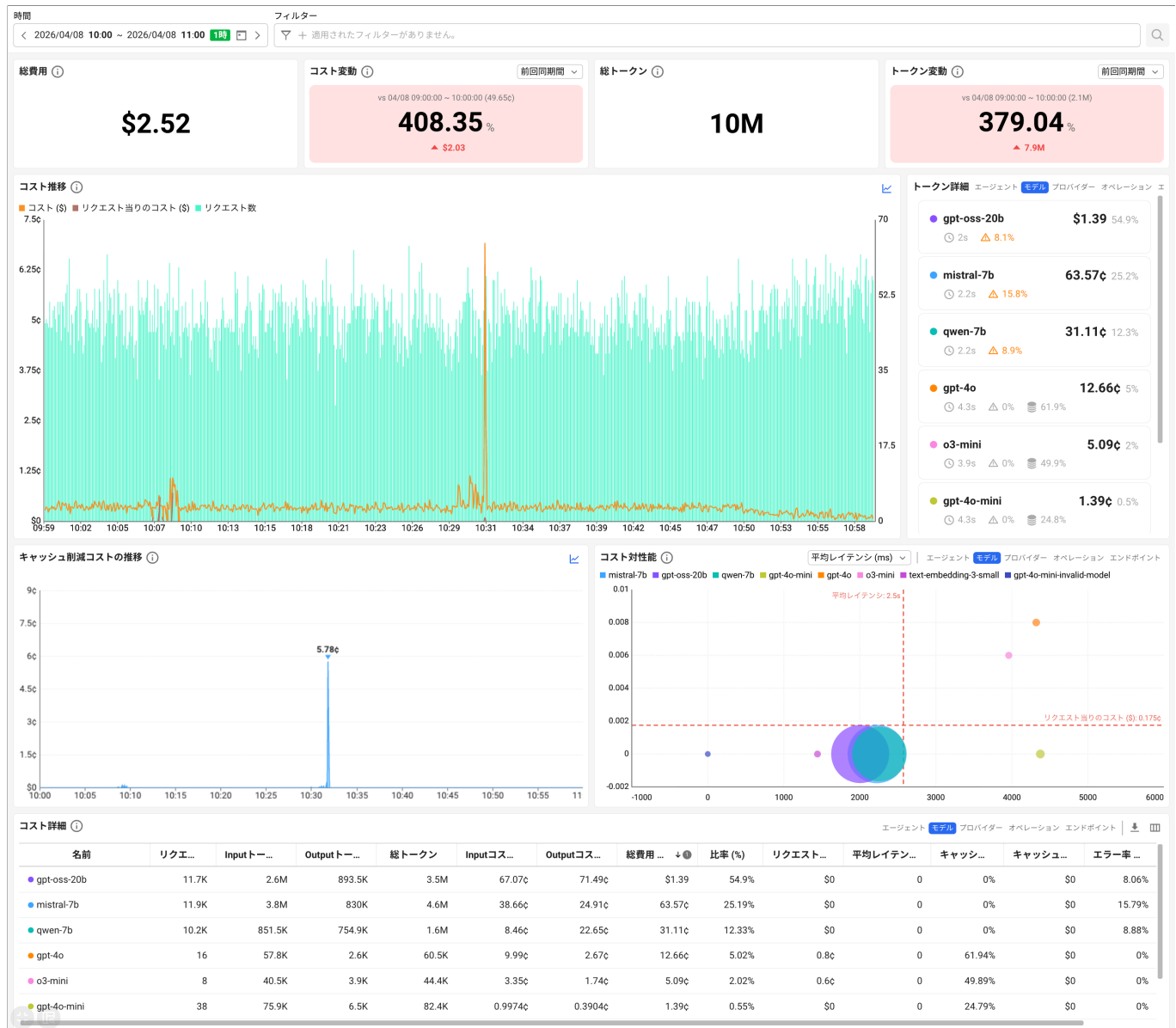
項目	LLMダッシュボード	トークン推移
時間モード	リアルタイム（Live）+ 過去時点	過去区間（Range）専用
主な目的	リアルタイム状態モニタリング	区間別推移分析
フィルター	エージェント選択	複合条件フィルター（Agent、Model、Provider、Operation、URL）
プロンプトログ連動	なし	チャートクリック時にプロンプトログへ移動
データエクスポート	なし	CSVダウンロード対応

LLMコスト分析

コスト分析はLLM API使用コストを多角的に分析するメニューです。コスト状況の把握、モデル別コスト比較、性能対比コスト効率分析、コスト最適化対象の選定に活用できます。

画面は上から下へ**サマリー** → **推移・詳細** → **キャッシュ・性能効率** → **テーブル**の順で構成されており、現状把握から最適化対象の選定まで自然に進められます。

上部オプションバーで時間範囲、フィルター、検索条件を設定してウィジェットデータを照会します。サマリーカードで比較期間（**直前同一期間**または**前週同一期間**）を選択でき、チャートドラッグで区間を絞り込めます。



サマリーカード

選択した時間範囲の主要指標を一目で確認できるカードです。

合計コスト

選択した時間範囲内で発生した合計コスト（\$）を表示します。InputコストとOutputコストを合算した金額です。

- コスト分析の出発点として、全体のコスト規模を一目で把握できます。
- コスト変動と合わせて確認すると、前期間と比べてコストがどのように変化したかを確認できます。

コスト変動

前期間比のコスト変動率（%）と変動金額（\$）を表示します。

- **プラス（▲ 赤）**：コストが増加しました。リクエスト量の増加、高額モデルの使用比率の増加、プロンプトサイズの変化などを確認してください。
- **マイナス（▼ 青）**：コストが減少しました。最適化効果やトラフィック減少を意味する場合があります。

合計トークン

選択した時間範囲内で使用された合計トークン数（Input + Output）を表示します。

- 合計コストと合わせて確認すると、コスト増加がトークン使用量によるものか、高額モデルの使用によるものかを区別できます。
- トークンが増加してもコストが同程度であれば、低価格モデルへの移行がうまく進んでいることを意味します。
- クリックするとトークン推移ページに移動します。

トークン変動

前期間比のトークン使用量変動率（%）と変動数を表示します。

- コスト変動と比較すると、コスト変動の原因を把握できます。
- **トークン変動率 > コスト変動率**: 低価格モデルの比率が増加したことを意味します。
- **トークン変動率 < コスト変動率**: 高額モデルの比率が増加したことを意味します。
- クリックすると比較期間範囲でトークン推移ページに移動します。

コスト推移

Property	Value
チャートタイプ	バーチャート（リクエスト数）+ ラインチャート（コスト、リクエスト当たりコスト）複合
フィルター	タグフィルター
アクション	個別表示、マージ表示、詳細表示

時間帯別のリクエスト数、コスト、リクエスト当たりコストの推移を表示します。バーチャートはリクエスト数、ラインはコスト（\$）とリクエスト当たりコスト（\$）です。

- リクエスト数とコストが連動して動く場合、使用量に比例した正常なコスト増加です。
- リクエスト数に変化がないのにコストだけ上昇する場合、リクエスト当たりコストが上がったことを意味します。高額モデルの比率増加やプロンプトサイズの変化を確認してください。
- リクエスト当たりコストのラインが急騰した場合、モデル切り替えやプロバイダの価格変更が考えられます。

トークン詳細

Property	Value
チャートタイプ	カードリスト（モデル別カード）
フィルター	Model / Provider / Operation / Agent タブ切り替え
アクション	詳細表示

タグ基準（Model、Provider、Operation、Agent）別にコスト、比率（%）、平均Latency、エラー率をカード形式で整理します。

各カードには以下の情報が表示されます。

Property	Description
モデル名/エージェント名	タグ値
コスト	該当グループの合計コスト（\$）
比率	該当グループコスト / 全体コスト x 100（%）
平均Latency	該当グループのLatency平均
エラー率	該当グループのエラー件数 / 全体リクエスト件数 x 100（%）
キャッシュヒット率	該当グループのcached_tokens / input_tokens x 100（%）

- コストが高い項目が上部にソートされるため、コスト寄与度の大きいモデル/エージェントをすぐに確認できます。
- エラー率が高い項目は、失敗したリクエストにもコストが発生しているため、エラーを削減するとコスト削減に直接的な効果があります。

キャッシュ節約コスト推移

Property	Value
チャートタイプ	ラインチャート（\$）
フィルター	タグフィルター
アクション	個別表示、マージ表示、詳細表示

プロンプトキャッシングにより節約された金額（\$）の時間帯別推移を表示します。キャッシュされたトークンは通常の入力トークンより低い単価が適用され、この差額が節約コストとして算出されます。

- 節約コストが安定して発生していれば、キャッシング戦略が効果的に機能していることを意味します。
- 節約コストが減少した場合、キャッシュヒット率の低下を確認してください。
- 合計コスト対比の節約コスト比率で、キャッシングの全体コスト削減寄与度を把握できます。

性能対比コスト

Property	Value
チャートタイプ	バブルチャート（x軸: 性能指標、y軸: リクエスト当たりコスト、バブルサイズ: リクエスト数）
フィルター	Model / Provider / Operation / Agent タブ切り替え
アクション	詳細表示、x軸指標選択

モデル/エージェント別の性能とコスト（リクエスト当たりコスト）の関係をバブルチャートで表示します。バブルサイズはリクエスト数を表し、点線で全体平均基準線が表示されます。

x軸選択可能な指標

指標	説明
Avg Latency (ms)	平均応答完了時間
Avg TTFT (ms)	平均初回トークン受信時間
Avg TPOT (ms)	平均トークン生成間隔
p95 Latency (ms)	95パーセンタイル応答時間
p95 TTFT (ms)	95パーセンタイル初回トークン時間
p95 TPOT (ms)	95パーセンタイルトークン生成間隔

四分面の解釈

位置	意味	推奨アクション
左下（速くて安い）	最もコスト効率が高い領域	このモデルの使用比率を維持または拡大

位置	意味	推奨アクション
右上（遅くて高い）	コスト最適化が最も急がれる領域	代替モデルの検討またはプロンプト最適化
左上（速いが高い）	性能は良いがコストが高い	該当の性能が本当に必要なワークロードか検討
右下（遅いが安い）	コストは低いが性能が不足	性能要件が低いワークロードに適合

- バブルが大きい項目ほど全体コストに占める割合が大きいため、最適化の効果が大きくなります。

コスト詳細

Property	Value
チャートタイプ	テーブル
フィルター	Model / Provider / Operation / Agent タブ切り替え
アクション	詳細表示、CSVダウンロード、カラム設定

タグ基準（Model、Provider、Operation、Agent）別にコスト関連の主要指標をテーブルで整理します。

Column	Description
Name	タグ値（モデル名、エージェント名など）
Requests	リクエスト件数
Input Tokens	入力トークン数
Output Tokens	出力トークン数
Total Tokens	全体トークン数
Input Cost (\$)	入力コスト

Column	Description
Output Cost (\$)	出力コスト
Total Cost (\$)	全体コスト
Share (%)	コスト占有率（該当グループコスト / 全体コスト）
Cost/Req (\$)	リクエスト当たりコスト
Avg Latency	平均応答時間
Cache Hit (%)	キャッシュヒット率
Cached Cost (\$)	キャッシュトークンコスト
Error Rate (%)	エラー率
Error Cost (\$)	エラーにより発生したコスト

- Total Cost (\$) が高い項目を中心にコスト最適化対象を選定してください。
- Cost/Req (\$) が高い項目はリクエスト当たりのコストが高いモデルのため、代替モデルの検討やプロンプト最適化を検討してください。
- Error Rate (%) が高い項目は失敗したリクエストにもコストが発生しているため、エラー原因を解決するとコスト削減に直接的な効果があります。
- Error Cost (\$) でエラーにより無駄になった金額を正確に確認できます。
- 右上のダウンロードボタンで現在のデータをCSVにエクスポートできます。
- カラム設定で表示するカラムを選択できます。

参考

比較期間の計算方式

比較モード	計算
直前同一期間	現在の照会期間と同じ長さだけ前の時点のデータと比較します。例: 本日09:00～12:00照会時、06:00～09:00と比較
前週同一期間	現在の照会期間からちょうど7日前の同じ時間帯のデータと比較します。例: 4/6 09:00～12:00照会時、3/30 09:00～12:00と比較

モデル別トークン価格のユーザー設定

モデル別のトークン料率を直接指定します。指定した値はエージェントの自動算出（`genai-prices` パッケージ）より優先して適用されるため、セルフホスティング・ファインチューニング・プロキシ経由のモデルのように公開料率表にないモデルのコストを算出したり、契約単価が定価と異なる場合に実際の単価へ補正したりできます。

whatap.conf

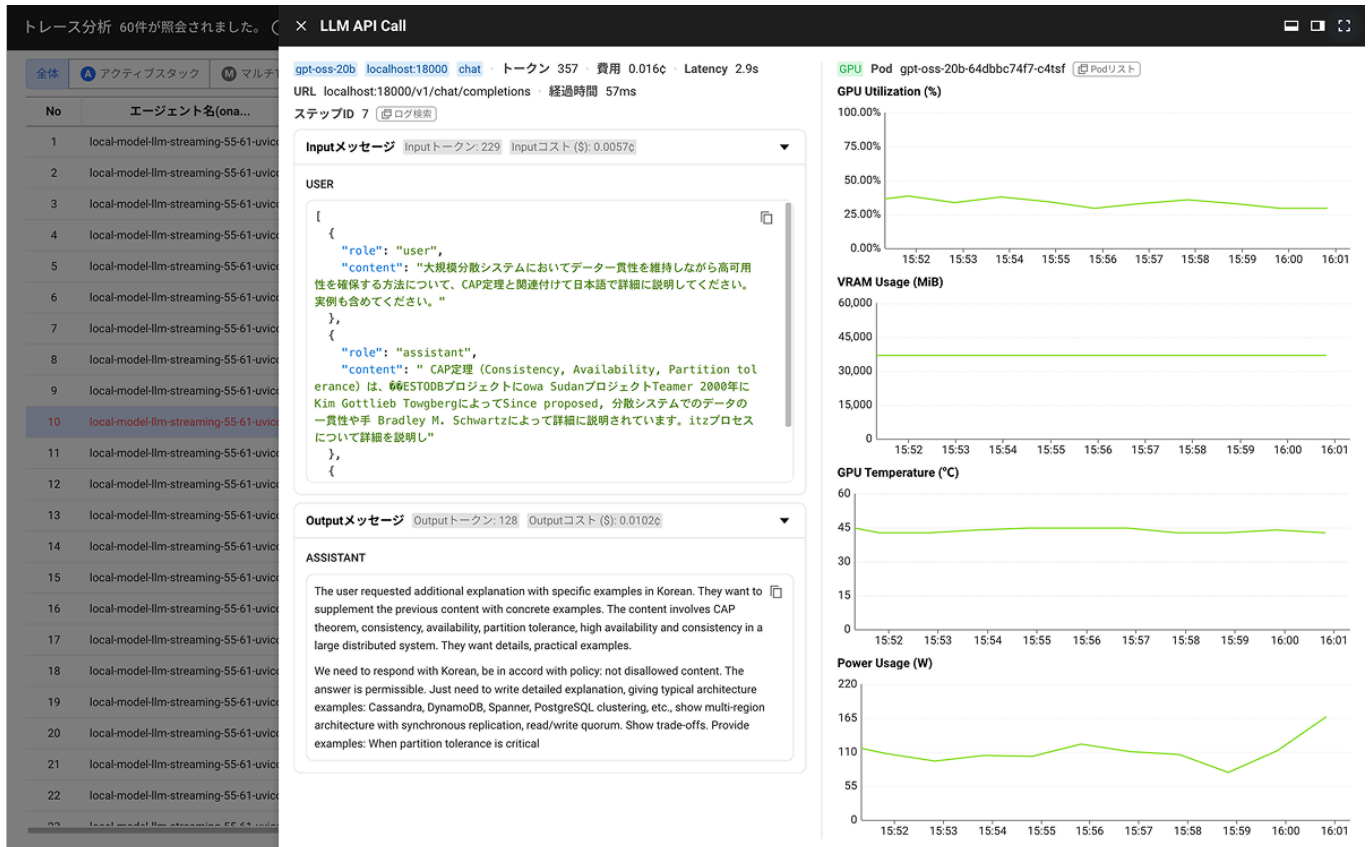
```
llm_model_pricing=model|input_per_1m|output_per_1m|cached_per_1m
# llm_model_pricing=gpt-4o|2.50|10.00|1.25,gpt-4o-mini|0.15|0.60|0.075,my-custom-model|1.00|5.00
```

- モデル間の区切りはカンマ（,）、フィールド間の区切りはパイプ（|）を使用します。
- 4番目のフィールドは省略（`model|1.00|5.00`）または空欄（`model|1.00|5.00|`）のどちらも可能です。
- 設定の変更は即時反映され、再起動は不要です。

LLM API分析（トランザクション連携）

AI Agent ObservabilityはWhaTap APMと統合されており、LLM API呼び出しをアプリケーショントランザクションの一部として追跡します。トランザクションプロファイルでLLM HTTPステップをクリックするとLLM API詳細Drawerが開き、プロンプト入力/出力、トークン使用量、コスト、エラー情報はもちろん、GPUインフラストラクチャの相関関係まで1つの画面で確認できます。

! プロンプト入力/出力の照会にはログ読み取り権限が必要です。



アクセス方法

1. LLMダッシュボードのヒットマップまたはトランザクション検索などからトランザクションを選択します。
2. トランザクションプロファイル画面でLLM HTTPCステップ（LLMプロバイダURLへのHTTP外部呼び出し）をクリックします。
3. 右側にLLM API詳細Drawerが開きます。

サマリーおよびプロンプト

LLM呼び出しの主要情報を一目で確認でき、プロンプト入力/出力の原本を確認できる領域です。

Identityタグ

呼び出し対象のモデルとプロバイダ情報をタグ形式で表示します。

Tag	Description	Example
Model	使用されたLLMモデル名	gpt-4o , claude-sonnet-4-20250514
Provider	LLM APIプロバイダ	api.openai.com , api.anthropic.com
Operation	呼び出しのOperation Type	chat , completion

呼び出しが失敗した場合（success=false）、赤色のErrorタグが追加で表示されます。

主要数値指標

Identityタグの横にインラインで表示されます。

Metric	Description
Token	全体トークン数（Input + Output）

Metric	Description
Cost	該当呼び出しのコスト（\$）
Latency	リクエスト開始～応答完了時間（ms）

HTTP情報

Property	Description
URL	呼び出しエンドポイント（host:port + path）
Elapsed	HTTP呼び出し所要時間（ms）
Step ID	このLLM呼び出しの固有識別子。ログ検索への遷移基準キーです。

Step ID横のログ検索ボタンをクリックすると、該当Step IDでフィルタリングされたログエクスプローラーが新しいウィンドウで開きます。

エラー情報

呼び出しにエラーが発生した場合のみ表示されます。

Property	Description
Error Class	エラークラス名（例: <code>RateLimitError</code> 、 <code>TimeoutError</code> ）
Error Message	エラー詳細メッセージ

プロンプト入力（Input）

折りたたみ/展開が可能なセクションで、LLMに送信された入力メッセージを表示します。

ヘッダーバッジ

Badge	Description
Input Tokens	入力トークン数
Cached Tokens	キャッシュから取得したトークン数（該当する場合のみ表示）
Input Cost (\$)	入力コスト

メッセージタイプ

Label	Description
SYSTEM	システムメッセージ。モデルの役割と動作を定義する指示文です。
USER	ユーザー入力メッセージ。実際のプロンプト内容です。

メッセージがチャンク（chunk）に分割されて収集された場合、自動的に正しい順序で組み立てて完全なテキストとして表示します。

モデル応答（Output）

折りたたみ/展開が可能なセクションで、モデルが生成した応答を表示します。

ヘッダーバッジ

Badge	Description
Output Tokens	出力トークン数
Reasoning Tokens	推論（Reasoning）トークン数（該当する場合のみ表示）
Output Cost (\$)	出力コスト

メッセージタイプ

Label	Description
ASSISTANT	モデルのテキスト応答です。
TOOL CALL	モデルがリクエストしたツール呼び出し（Function Calling）の内容です。
TOOL RESULT	ツール呼び出しの実行結果です。

GPU関連関係

マルチトランザクション（mtidが存在する場合）の場合、Drawer右側に表示されます。LLM呼び出しを処理したPodのGPUインフラストラクチャ状態をリアルタイムチャートで表示し、モデル応答遅延がGPUリソース不足によるものかを判断できます。

GPU情報ヘッダー

Property	Description
Pod	LLM推論を実行したKubernetes Pod名
Pod詳細ボタン	クリックするとKubernetesモニタリングのPod詳細ページが新しいウィンドウで開きます。

GPUチャート

LLM呼び出し時点の前後5分範囲のGPUメトリクスをラインチャートで表示します。

Chart	Metric	Unit	Description
GPU Utilization	DCGM_FI_DEV_WEIGHTED_GPU_UTIL	%	GPU演算リソースの使用率。100%に近い場合、GPUが飽和状態であり、LLM応答遅延の原因となる可能性があります。
VRAM Usage	DCGM_FI_DEV_FB_USED	MiB	GPUメモリ（Video RAM）使用量。モデルのロード

Chart	Metric	Unit	Description
			と推論に使用され、不足するとOOMやスワップが発生します。
GPU Temperature	DCGM_FI_DEV_GPU_TEMP	°C	GPU温度。過熱時に自動スロットリングが発生し、性能が低下します。
Power Usage	DCGM_FI_DEV_POWER_USAGE	W	GPU電力消費。Utilizationと合わせて確認すると、実際の演算負荷を把握できます。

ⓘ GPU相関関係は以下の条件がすべて満たされた場合に表示されます。

- トランザクションがマルチトランザクション（mtid存在）の場合
- LLM推論を実行したPod名が識別された場合
- 連携されたKubernetesプロジェクトでDCGM GPUメトリクスが収集されている場合

分析シナリオ

LLM応答遅延の原因特定

1. LLMダッシュボードのヒットマップで応答時間が長いトランザクションをクリックします。
2. トランザクションプロファイルでLLM HTTPCステップの**Elapsed**時間を確認します。
3. LLM API詳細Drawerを開いて**Latency**値と**GPU Utilization**チャートを比較します。
4. GPU Utilizationが高い状態でLatencyが急増している場合、GPUリソース不足が原因です。
5. GPU Utilizationが低いのにLatencyが高い場合、プロバイダ側の遅延（キュー待ち、Rate Limitなど）を疑います。

エラー発生LLM呼び出しの再現

1. トランザクションプロファイルでエラーが表示されたLLM HTTPCステップをクリックします。

2. Drawerの**Error Class**と**Error Message**でエラータイプを確認します。
3. **プロンプト入力（Input）**セクションでSYSTEMメッセージとUSERメッセージの原本を確認します。
4. 収集されたプロンプト原本とモデル/パラメータ情報で同一の呼び出しを再現し、問題を分析します。

コストが高いLLM呼び出しの分析

1. コスト分析ページでコストが高い時間帯を特定します。
2. 該当時間帯のトランザクションを検索してプロファイルを開き、LLM HTTPCステップをクリックします。
3. Drawerで**Token**数と**Cost**を確認します。
4. プロンプト入力セクションでInput Tokensバッジを確認し、プロンプトが不必要に長くないか確認します。
5. **Cached Tokens**バッジが0の場合、キャッシングが適用されていない呼び出しのため、キャッシング対象のプロンプトかどうかを検討します。

参考

データ収集構造

LLM API詳細Drawerは2つのデータソースを組み合わせています。

Data Source	Description	Usage
APMトレース	HTTP呼び出し情報（URL、host、port、elapsed、エラークラス/メッセージ）、Step ID	トランザクションプロファイルでの呼び出し識別およびHTTPレベル情報表示
#LlmCallLog ログ	モデル、プロバイダ、トークン、コスト、成功可否、メッセージ内容	プロンプト原本復元およびLLMレベルメタデータ表示

2つのデータは**Step ID**を基準に連結されます。

メッセージチャンク組み立て

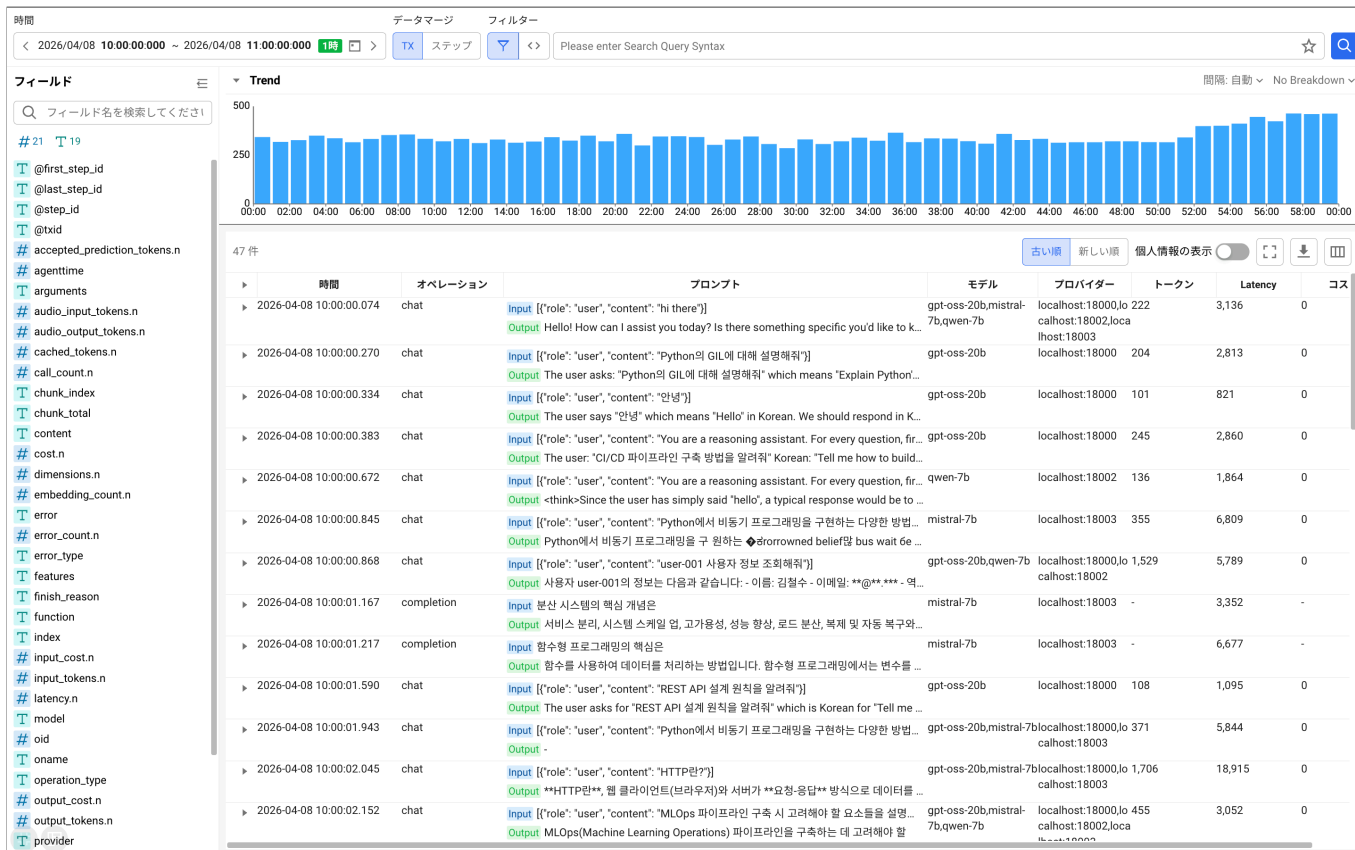
LLMプロンプトと応答は長くなる場合があるため、サーバーで複数のチャンク（chunk）に分割して収集されます。

Drawerは `chunk_index` フィールドを基準にチャンクを正しい順序で組み立て、完全なメッセージを復元します。タイプ当たり最大100個のチャンクまでサポートします。

LLMプロンプトログ

プロンプトログはLLM API呼び出しの詳細ログを照会・分析するメニューです。各呼び出しの入力（プロンプト）、出力（応答）、トークン使用量、コスト、パフォーマンス指標を個別件単位で確認できます。トークン推移やコスト分析で異常区間を発見した後、原因を特定するためにドリルダウンする用途で活用します。

ⓘ このメニューにはログ読み取り権限が必要です。権限がない場合、メニューにアクセスできません。



画面構成

プロンプトログは上段オプションバー、左側フィルターパネル、右側メイン領域（トレンドチャート + ログリスト）の3段レイアウトで構成されます。

上段オプションバー

- **時間範囲選択:** 開始日時と終了日時を指定します。クイック選択オプション（直近5分、10分、30分、1時間、3時間、6時間、12時間）を提供します。秒単位およびミリ秒単位まで指定できます。
- **マージ基準選択:** ログをマージする基準を選択します。
 - **トランザクション:** 1つのトランザクション内で発生したすべてのLLM呼び出しを1行にマージします。呼び出し数（Call Count）とエラー数（Error Count）が併せて表示されます。
 - **ステップ:** 個別LLM呼び出し（Step）単位で1行ずつ表示します。各呼び出しのモデル、temperature、TTFTなどの詳細情報を確認できます。
- **フィルター:** 検索クエリを入力して特定条件のログのみ照会できます。

左側フィルターパネル

ログフィールドベースのタグフィルターを提供します。各フィールドの値リストが表示され、クリックしてフィルターを適用できます。

トレンドチャート

選択した時間範囲のログ件数を時間帯別バーチャートで表示します。チャートを折りたたんだり展開してスペースを調整でき、チャートとログリスト間の区切り線をドラッグして比率を調整できます。

ログリスト

LLM呼び出しログをテーブル形式で表示します。マージ基準（トランザクション/ステップ）によって表示されるカラムが異なります。

基本カラム

Column	Description
ステータス	エラー有無を色で表示します。エラーが発生した行は赤色で表示されます。

Column	Description
Time	ログ発生時刻
Operation	LLM呼び出しのOperation Type
Prompt	プロンプト入力と応答の要約。クリックするとトランザクションプロファイルポップアウトが開きます。
Model	使用されたLLMモデル名
Provider	LLMプロバイダー（例：api.openai.com）
Tokens	全トークン数（Input + Output）
Latency	リクエスト開始～レスポンス完了時間（ms）
Cost	該当呼び出しのコスト（\$）

トランザクションモード専用カラム

Column	Description
Call Count	トランザクション内のLLM呼び出し数
Error Count	トランザクション内でエラーが発生した呼び出し数

ステップモード専用カラム

Column	Description
Step	トランザクション内のLLM呼び出し順序インデックス
Step ID	個別LLM呼び出しの固有識別子
temperature	LLM呼び出し時に設定されたtemperatureパラメータ値

Column	Description
TTFT	Time To First Token (ms)。ストリーミングモードでのみ表示されます。

追加表示可能カラム

カラム設定で追加表示できます。

Column	Description
Input Tokens	入力トークン数
Output Tokens	出力トークン数
Cached Tokens	キャッシュから取得したトークン数
Reasoning Tokens	推論 (Reasoning) トークン数
Input Cost	入力コスト (\$)
Output Cost	出力コスト (\$)
Stream	ストリーミングモード有無 (True/False)
Features	有効化された機能フラグ
Agent	エージェント名
Error	エラーメッセージ (エラー発生時のみ表示)
Error Type	エラー種別 (api_error、program_error など)
TXID	トランザクションID

ログ詳細表示

ログリストで**Prompt**カラムをクリックすると、トランザクションプロファイルポップアウトが開きます。該当LLM呼び出しを含む全トランザクションの詳細情報を確認できます。

- **トランザクションモード**: 全トランザクションビューに移動します。
- **ステップモード**: 該当Stepがハイライトされた状態でトランザクションビューが開きます。

トランザクションプロファイルでLLM HTTPCステップをクリックすると、プロンプト入力/出力の原本、トークン/コストの詳細、GPU関連分析まで確認できます。詳細は[LLM API分析](#)ドキュメントを参照してください。

マージ基準の詳細

トランザクションマージ

1つのトランザクション（@txid基準）内で発生したすべてのLLM呼び出しを1行にマージします。

- **Prompt表示**: 最初のStepの入力（Input）と最後のStepの出力（Output）が表示されます。
- **トークン/コスト**: トランザクション内の全呼び出しの合計が表示されます。
- **モデル**: トランザクションの代表モデルが表示されます。
- **Call Count / Error Count**: 総呼び出し数とエラー発生呼び出し数が表示されます。

ステップマージ

個別LLM呼び出し（@step_id基準）単位で1行ずつ表示します。

- **Prompt表示**: 該当Stepの入力（Input Message）と出力（Output Message）が表示されます。
- **トークン/コスト**: 該当呼び出しの個別値が表示されます。
- **追加情報**: temperature、TTFT、Stream有無など呼び出し単位の詳細情報を確認できます。

他のメニューからの遷移

プロンプトログは他の分析メニューからドリルダウンする際に頻繁に使用されます。

遷移パス	説明
トークン推移 > 時系列チャートをクリック	該時点の時間範囲が自動設定され、プロンプトログが開きます。
トークン推移 > ボックスプロットのモデルをクリック	該当モデルのフィルターが自動適用され、プロンプトログが開きます。

逆に、プロンプトログで**Prompt**カラムをクリックすると、トランザクションプロファイルに移動し、トランザクション内のすべてのステップ（DB、HTTP、LLMなど）を時系列で確認できます。

参考

ログタイプ

プロンプトログは内部的に `#LlmCallLog` カテゴリのログを照会します。

Log Type	Description
<code>llm_step_status</code>	個別LLM呼び出しの最終結果（モデル、トークン、コスト、パフォーマンスなど）
<code>llm_tx_status</code>	トランザクション単位の集計結果（呼び出し数、エラー数、合算トークン/コスト）
<code>system_message</code>	システムメッセージ（プロンプトのシステムロール部分）
<code>input_message</code>	入力メッセージ（ユーザープロンプト）
<code>output_message</code>	出力メッセージ（モデル応答）
<code>tool</code>	ツール呼び出し（Function calling）

Log Type	Description
tool_result	ツール呼び出し結果

プロンプト分析

アプリケーションで使用中のプロンプトを識別し、プロンプト単位で呼び出し数・トークン・レイテンシ（TTFT）・コスト・エラーを分析するメニューです。どのプロンプトが最も多く使われているか、どのプロンプトが遅く、コストが高く、不安定かを把握し、最適化の対象を選定するために活用できます。

呼び出しは**オペレーション（operation）とプロンプトバージョン（prompt version）**の2つの単位で識別されます。オペレーションは機能単位の呼び出しのまとまりで、プロンプトバージョンは該当オペレーションで使用されたシステムプロンプトのバージョンです。1つのオペレーションは複数のプロンプトバージョンを持つことができ、チャートの大部分はプロンプトバージョン単位で集計されます。

画面は上から下へサマリーカード → 呼び出し・推移 → 分布・エラー → 機能使用 → テーブルの順に構成されており、状況の把握から個別オペレーションの点検まで自然にたどることができます。上段オプションバーで時間範囲とフィルターを設定してから、ウィジェットのデータを照会します。

プロンプト分析

サマリーカード

選択した時間範囲の主要な指標を一目で確認できるカードです。

固有プロンプト数

選択した時間範囲内で1回以上呼び出された、異なるプロンプトの数を表示します。カード下部には、それらのプロンプトの総呼び出し数もあわせて表示されます。

- ・ プロンプト運用の規模を一目で把握するための出発点です。
- ・ 固有プロンプト数が普段より急増した場合は、意図しないプロンプトの変種がデプロイされていないかを確認してください。

総プロンプト呼び出し数

選択した時間範囲内に発生したプロンプト呼び出しの総回数を表示します。カード下部にはプロンプト当たりの平均呼び出し数もあわせて表示されます。プロンプト当たりの平均とあわせて見ることで、少数のプロンプトに呼び出しが集

中しているのか、均等に分散しているのかを推し量ることができます。

- $\text{プロンプト当たり平均} = \text{総プロンプト呼び出し数} / \text{固有プロンプト数}$

プロンプト関連の総コスト

選択した時間範囲にLLM API呼び出しで発生した総コスト（\$）を表示します。カード下部には**最もコストが高いオペレーション**のバージョン名と金額、全体に対する比重（%）もあわせて表示されます。

- 比重が高い単一のプロンプトは、コスト最適化の優先対象です。
- 特定のプロンプトが全体コストの大部分を占める場合は、そのプロンプトのトークンの長さやモデルの選択を優先的に検討してください。

プロンプト別の呼び出し量TOP 10

Property	Value
チャートタイプ	縦棒バーチャート
集計単位	プロンプトバージョン
アクション	表示指標の選択（右上のドロップダウン）

呼び出し量が多い上位10件のプロンプトバージョンを、バーの長さで並べて表示します。右上のドロップダウンで表示指標（呼び出し数、総トークン、総コスト）を切り替えられます。

- 特定のバージョンに呼び出しが集中している場合、そのバージョンが全体の性能・コストに与える影響が大きいため、優先的に管理してください。
- 新規にデプロイしたバージョンが上位に素早く入ってくる場合は、トラフィックの切り替え（ロールアウト）が正常に進行しているという意味です。

プロンプト使用推移

Property	Value
チャートタイプ	積み上げエリアチャート（stacked area）
集計単位	プロンプトバージョン
アクション	凡例項目のトグル

時間帯別に各プロンプトバージョンの呼び出し量の推移を積み上げエリアで表示します。凡例の色が各プロンプトバージョンに対応し、エリアの厚さがそのバージョンの呼び出し量を示します。

- エリア全体の厚さの変化から、時間帯別の総呼び出し量の流れを把握できます。
- 特定の時間帯に特定のバージョンの比重が急変した場合は、トラフィックパターンの変化や新バージョンのデプロイの影響を確認してください。
- 新しいバージョンのデプロイ直後にそのエリアが急速に厚くなる場合は、トラフィックの切り替えが進行中であるという意味です。

プロンプト別のTTFT分布

Property	Value
チャートタイプ	箱ひげ図（X軸: プロンプトバージョン）
集計単位	プロンプトバージョン
アクション	詳細表示

プロンプトバージョン別の**TTFT（Time To First Token、最初のトークンまでの時間）**の分布を箱ひげ図で比較します。ボックスの領域はp25～p75（中間50%の範囲）、中央の線は中央値、ひげは最小値～最大値を示します。

- 中央値が低くボックスが狭いバージョンが、最も安定して速い初期レスポンスを提供します。

- ひげが長く伸びているバージョンは断続的に極端な初期遅延が発生するため、入力の長さやモデルの負荷を確認してください。

Finish Reason分布

Property	Value
チャートタイプ	ドーナツチャート
集計単位	終了理由 (length / stop / none)
アクション	凡例項目のトグル

LLM APIがレスポンスで返す終了理由 (finish reason) の値をそのまま集計し、分布として表示します。表示される値と名称は**プロバイダー・モデルによって異なる場合があります**、WhaTapが独自に定義するのではなく、APIが返した値をそのまま使用します。

- 正常終了を示す値 (例: `stop`) の比重が大部分であれば、レスポンスが意図どおりに完了しているという意味です。
- 長さ制限による中断を示す値 (例: `length`) の比重が高い場合は、最大トークン (`max_tokens`) の設定が小さすぎてレスポンスが途中で切れていないかを確認してください。
- 終了理由が空、または記録されていない呼び出しは、別の値 (例: `none`) としてまとめて表示される場合があります。

❗ 表示されるfinish reasonの値は、使用するLLMプロバイダー・モデルのレスポンス仕様に従います。各値の正確な意味は、該当プロバイダーのAPIドキュメントを参照してください。

プロンプト別のエラー率

Property	Value
チャートタイプ	縦棒バーチャート（X軸: プロンプトバージョン）
集計単位	プロンプトバージョン
アクション	詳細表示

プロンプトバージョン別の、呼び出しに対するエラーの発生率（%）を表示します。

- エラー率が高いバージョンは、誤ったツール定義、スキーマの不一致、トークン超過などが原因である可能性があります。そのため、オペレーション一覧でそのバージョンの指標もあわせて確認してください。
- 失敗したリクエストにもコストが発生するため、エラー率が高いバージョンを改善すると、コスト削減にも直接的な効果があります。

エラー率 = エラー呼び出し数 / 全呼び出し数 x 100

オペレーション別の機能使用比率

Property	Value
チャートタイプ	カードグリッド（オペレーション別カード）
集計単位	オペレーション
アクション	-

オペレーションごとにカードが1枚生成され、カード上部にオペレーション名と総呼び出し数が表示されます。カード内には、そのオペレーションの呼び出しで各機能（例: reasoning）が使用された比率がバーとパーセントで表示されます。

- 1回の呼び出しで複数の機能を同時に使用できるため、比率の合計は100%にならない場合があります、使用されてい

い機能は表示されません。

- 機能使用のパターンから、オペレーションの動作方式を素早く理解し、意図と異なる機能の使用を発見できます。

オペレーション一覧

Property	Value
チャートタイプ	テーブル
集計単位	オペレーション+プロンプトバージョン
アクション	ソート

期間内に識別されたオペレーションをプロンプトバージョン単位で一覧表示し、主要な指標をテーブルにまとめます。同じオペレーションでも、プロンプトバージョンが異なる場合は別の行として表示されます。

Column	Description
オペレーション	機能単位の呼び出しのまとまりの名前
プロンプトバージョン	該当オペレーションで使用されたシステムプロンプトのバージョン
プロンプトプレビュー	システムプロンプト原文の冒頭部分（ない場合は―）
呼び出し数	期間内の総呼び出し回数
平均input	呼び出し当たりの平均入力トークン数
平均output	呼び出し当たりの平均出力トークン数
平均TTFT	最初のトークンまでの平均時間（ms）
平均コスト	呼び出し1件当たりの平均コスト（¢）
エラー率（%）	呼び出しに対するエラーの発生率

- 既定のソートは呼び出し数の降順で、カラムヘッダーをクリックしてソート基準を変更できます。
- **平均コスト** が高い行は、リクエスト当たりの単価が高いプロンプトであるため、プロンプトの長さの短縮やモデルの検討を考慮してください。
- **プロンプトプレビュー** で、同じオペレーションのバージョン間のプロンプトの違いを素早く比較できます。

参考

プロンプトの識別方式

単位	説明
オペレーション	機能単位の呼び出しのまとまり（例: <code>cs_chat_support</code> ）
プロンプトバージョン	該当オペレーションで使用されたシステムプロンプトのバージョン（例: <code>v2.1-ko</code> ）

- 1つのオペレーションは複数のプロンプトバージョンを持つことができ、TOP 10・使用推移・TTFT分布・エラー率のウィジェットはプロンプトバージョン単位で集計されます。
- プロンプトバージョン間の性能差を定量的に比較する場合は**プロンプト比較メニュー**を、レスポンス品質・安全性の評価が必要な場合は**レスポンス品質分析メニュー**をあわせて活用してください。

プロンプト比較

2つのプロンプトを並べて、呼び出し数・コスト・性能・レスポンス品質を比較するメニューです。プロンプトを修正した後、その変更が実際に性能とコストを改善したかを定量的に検証したり、異なるオペレーション・バージョン間の特性の違いを分析したりするために活用できます。

比較対象は**オペレーション+プロンプトバージョン+日付**の組み合わせで指定し、基準となる側を**A**、比較対象を**B**とします。通常はAに既存バージョン、Bに変更バージョンを置いて差分を確認します。

画面は上から下へ**サマリー（呼び出し・コスト）→プロンプトDiff→指標比較→推移比較→分布比較→プロンプトログ**の順に構成されており、「何が変わったのか」（Diff）から「その変更がどのような結果をもたらしたのか」（指標・推移・ログ）までたどることができます。

プロンプト比較

比較条件の設定

上段オプションバーで比較する2つのプロンプトを指定します。

- ・ **基準日 / オペレーションA / プロンプトバージョンA**: 基準となるプロンプト（A）と照会する日付を選択します。
- ・ **比較日 / オペレーションB / プロンプトバージョンB**: 比較対象のプロンプト（B）と照会する日付を選択します。

✔ AとBには異なる日付を指定できるため、同じプロンプトのデプロイ前後を異なる期間で比較することもできます。すべてのウィジェットで、Aは青色、Bは紫色で一貫して表示されます。

総プロンプト呼び出し数

Property	Value
チャートタイプ	バーチャート（A vs B）

Property	Value
比較対象	A（基準）・B（比較）
アクション	マウスオーバー時に値を表示

選択した2つのプロンプトの総呼び出し数をバーで並べて比較します。各バーには 日付・プロンプトバージョン がラベルとして表示されます。

- 呼び出し数の差が大きい場合は、平均指標や推移を解釈する際にサンプル規模の差もあわせて考慮してください。
- 新しいバージョン（B）の呼び出し数が急速に増えている場合は、トラフィックの切り替え（ロールアウト）が進行中であるという意味です。

プロンプト関連の総コスト

Property	Value
チャートタイプ	バーチャート（A vs B）
比較対象	A（基準）・B（比較）
アクション	マウスオーバー時に値を表示

選択した2つのプロンプトの総コストをバーで並べて比較します。

- 呼び出し数が同程度なのに総コストの差が大きい場合は、プロンプトの長さやモデルの単価の違いが原因である可能性があります。
- 呼び出し当たりのコストは、下の指標比較の 呼び出し当たりコスト カードで、サンプル規模に関係なく比較できます。

システムプロンプトDiff

Property	Value
チャートタイプ	テキストDiff（左: A / 右: B）
アクション	Side-by-side / Inline表示の切り替え

AとBのシステムプロンプトの原文を行単位で比較します。追加された行と削除された行が色で区別され、右上のアイコンで**Side-by-side**（左右分割）と**Inline**（1つの流れ）の表示を切り替えられます。

- このウィジェットは「何が変わったのか」を、下の指標・推移ウィジェットは「その変更がどのような結果をもたらしたのか」を示すため、あわせて解釈してください。
- 一方のバージョンのプロンプトが空の場合は、新規導入または全面的に置き換えられたバージョンであることを意味します。

指標比較カード

Property	Value
チャートタイプ	指標カード（A・Bの値を並列表示）
比較対象	A（基準）・B（比較）
アクション	-

主要な性能・コスト・品質の指標を、カードごとにAとBの値を並べて表示します。

Metric	Description
平均Latency	リクエスト開始～レスポンス完了までの平均時間
平均TTFT	最初のトークンまでの平均時間（体感の応答性）

Metric	Description
平均TPOT	出力トークン当たりの平均生成間隔（ストリーミング速度）
呼び出し当たりコスト（\$）	リクエスト1件当たりの平均コスト
エラー率（%）	呼び出しに対するエラーの発生率
キャッシュヒット率	入力トークンのうちキャッシュから取得したトークンの割合

- AとBの値を直接対照することで、変更が各指標を改善したのか悪化させたのかを一目で確認できます。
- キャッシュヒット率が上がりながら呼び出し当たりコストが下がった場合は、プロンプトの共通prefixが増えてキャッシュ効率が良くなった可能性があります。

TTFT推移（A vs B）

Property	Value
チャートタイプ	ラインチャート（A、Bの2系列）
比較対象	A（基準）・B（比較）
アクション	マウスオーバー時に値を表示

時間帯別の平均TTFTをA・Bで重ねて表示します。

- 特定の時間帯（例: トラフィックのピーク）だけで差が開くのか、全区間で均等に差があるのかを確認してください。
- 一方で断続的なスパイクが現れる場合は、キュー待ちやプロバイダーの遅延が疑われます。

呼び出し当たりコスト推移（A vs B）

Property	Value
チャートタイプ	ラインチャート（A、Bの2系列）
比較対象	A（基準）・B（比較）
アクション	マウスオーバー時に値を表示

呼び出し1件当たりのコストの時間帯別推移をA・Bで比較します。

- キャッシュヒットのパターンや出力の長さの変化が、呼び出し当たりコストに与えた影響を時間の流れで確認できます。
- 平均値は同じでも、特定の区間だけコストが跳ね上がるパターンを発見できます。

呼び出しボリューム推移（A vs B）

Property	Value
チャートタイプ	ラインチャート（A、Bの2系列）
比較対象	A（基準）・B（比較）
アクション	マウスオーバー時に値を表示

2つのプロンプトの時間帯別の呼び出し量の推移を比較します。

- 新しいバージョン（B）へのトラフィック切り替えの進行状況を把握できます。
- ボリュームの差が大きい区間は、他の指標を解釈する際のサンプル規模の差の根拠になります。

エラー推移（A vs B）

Property	Value
チャートタイプ	ラインチャート（A、Bの2系列）
比較対象	A（基準）・B（比較）
アクション	マウスオーバー時に値を表示

時間帯別のエラー率の推移をA・Bで比較します。

- ・変更直後の特定の時点にエラーが集中したのか、安定して低く維持されているのかを確認してください。
- ・Bのエラー率がAより高くなる区間がある場合は、システムプロンプトDiffで原因となった変更を逆にたどってください。

Finish Reason分布

Property	Value
チャートタイプ	ドーナツチャート（A・Bそれぞれ）
集計単位	finish reasonの値
アクション	凡例項目のトグル

AとBそれぞれのレスポンス終了理由（finish reason）の分布をドーナツで並べて表示します。表示される値と名称はプロバイダー・モデルによって異なる場合があります、WhaTapが独自に定義するのではなく、APIが返した値をそのまま使用します。

- ・長さ制限による中断を示す値（例: `length`）の比重がBで減った場合は、出力の制御が改善され、レスポンスが途中で切れるケースが減ったという意味です。
- ・正常終了を示す値（例: `stop`）の比重の変化から、レスポンスの完結性の違いを推し量ることができます。

ⓘ 表示されるfinish reasonの値は、使用するLLMプロバイダー・モデルのレスポンス仕様に従います。各値の正確な意味は、該当プロバイダーのAPIドキュメントを参照してください。

機能使用の比較

Property	Value
チャートタイプ	バーチャート (A vs B)
比較対象	A (基準) ・ B (比較)
アクション	-

機能別の使用比率をA・Bで比較します。

- プロンプトの変更（例: 例示の追加、指示の強化）が、意図した機能使用パターンの変化を導いたかを検証できます。
- 1回の呼び出しで複数の機能を同時に使用できるため、比率の合計は100%にならない場合があります。

モデル分布

Property	Value
チャートタイプ	100%積み上げ横棒 (A・Bの各行)
比較対象	A (基準) ・ B (比較)
アクション	凡例項目のトグル

各プロンプトが呼び出したモデルの構成比率をA・Bの行で比較します。凡例の色が各モデルに対応します。

- AとBのモデル構成が異なる場合は、レイテンシ・コストの差がプロンプトの変更によるものか、モデルの違いによるものを区別して解釈する必要があります。

- 特定のモデルの比重が大きく変わった場合は、ルーティングルールやモデル指定が変更されていないかを確認してください。

プロンプトログ (A・B)

Property	Value
チャートタイプ	テーブル (A・Bそれぞれ)
アクション	行の展開、プロンプトログへの移動

AとBそれぞれの最近の呼び出しログをテーブルで表示し、指標の差が実際のレスポンスにどう現れているかを個別の事例で確認できます。

Column	Description
時間	呼び出しが発生した時刻
プロンプト	入力 (Input) とレスポンス (Output) の要約
モデル	使用されたLLMモデル名
トークン	全トークン数
Latency	リクエスト開始～レスポンス完了までの時間
コスト	その呼び出しのコスト

- レスポンス品質評価が適用された呼び出しは、**Eval** 領域に評価項目別のスコア (例: Hallucination、Factuality、Relevance、Toxicity、Prompt Injection、PII Leak、URL Scan、Judge) もあわせて表示されます。評価項目の詳細は**レスポンス品質分析**のドキュメントを参照してください。
- 各テーブル右上の**プロンプトログへの移動**ボタンで、該当プロンプトの元のログに移動できます。

参考

比較対象の指定方法

Term	Description
オペレーション	機能単位の呼び出しのまとまり（例: <code>cs_chat_support</code> ）
プロンプトバージョン	該当オペレーションで使用されたシステムプロンプトのバージョン（例: <code>v3.0-structured</code> ）
日付	各対象を照会する基準日（A・Bで異なる指定が可能）

- 比較対象の候補をまず探す場合は**プロンプト分析**を、レスポンス品質・安全性の観点での評価が必要な場合は**レスポンス品質分析**メニューを活用してください。
- すべてのウィジェットで、Aは青色、Bは紫色で表示されます。

レスポンス品質分析

LLMが生成したレスポンスの品質と安全性を自動評価し、ハルシネーション・有害性・PII漏洩・プロンプトインジェクション・事実性・回答の適切性などのリスクをカテゴリ別にモニタリングするメニューです。呼び出し数・コスト・レイテンシなどの運用指標だけでは見えない「レスポンスの内容自体が信頼でき安全か」を診断するために活用します。

評価は正規表現・NLI・LLM Judgeなど複数の評価器を組み合わせた**ハイブリッド方式**で実行され、コストを抑えるために全呼び出しのうち一部を**サンプリング**して評価します。各評価項目はスコアとして算出され、ウィジェットの**違反基準（しきい値）**をもとに違反の有無が集計されます。

画面は上から下へ**サマリー→違反推移・ヒットマップ→スコア分布・モデル比較→時間帯ヒットマップ→違反ケース**の順に構成されており、全体状況の把握から個別の違反事例の確認までたどることができます。上段オプションバーで時間範囲とフィルターを設定してから、ウィジェットのデータを照会します。

プロンプト比較

サマリーカード

選択した時間範囲の主要な品質指標を一目で確認できるカードです。

評価率

全LLM呼び出しのうち、品質評価が実行された呼び出しの割合を表示します。カード下部には評価LLM呼び出し数と全呼び出し数もあわせて表示されます。（例: 評価LLM呼び出し 170 / 134.8K）

- 評価にはコストが発生するため、サンプリング比率の範囲内で運用されます。評価率が低い場合はサンプルが小さいため、他のウィジェットの比率を解釈する際は注意してください。

レスポンス品質

評価されたレスポンスの総合品質スコアを表示します。

- 複数の評価項目を総合した代表指標です。推移が下降した場合は品質低下のシグナルとして解釈してください。

ハルシネーション疑い率

ハルシネーションスコアが基準以上の呼び出しの割合を表示します。カード下部には判定基準と該当する呼び出し数もあわせて表示されます。（例: `score ≥ 0.5 11.4K`）

- RAG・検索ベースのレスポンスで、事実と異なる内容が生成される度合いを示します。比率が上昇した場合は、コンテキストの構成やモデルを確認してください。

事実性

レスポンスが事実に合致している度合い（Factuality）の平均水準を表示します。

- 値が高いほどレスポンスが事実に忠実です。ハルシネーション疑い率とあわせて見ることで、事実の歪みの全体的な傾向を把握できます。

有害性

有害（Toxicity）と評価されたレスポンスの割合を表示します。

- ユーザーに表示されるべきではない攻撃的・有害な表現の発生度合いをモニタリングします。値が急増した場合は、違反ケースで実際のレスポンスを確認してください。

カテゴリ別の違反推移

Property	Value
チャートタイプ	積み上げエリアチャート（stacked area）
集計単位	評価カテゴリ
アクション	違反基準（しきい値）の選択、凡例項目のトグル

時間帯別に各リスクカテゴリの違反発生量の推移を積み上げエリアで表示します。凡例の色が各カテゴリ（ハルシネーション・有害性・PII漏洩・プロンプトインジェクション・事実性・回答の適切性）に対応します。

- 右上の**違反基準**ドロップダウンで、違反として集計するスコアのしきい値を調整できます。
- 特定の時点で特定のカテゴリが急増した場合は、デプロイやトラフィックの変化と関連づけて原因を追跡してください。

オペレーション × カテゴリ違反率ヒットマップ

Property	Value
チャートタイプ	ヒットマップテーブル（行: オペレーション、列: カテゴリ）
集計単位	オペレーション × カテゴリ
アクション	違反基準（しきい値）の選択

行はオペレーション、列はリスクカテゴリで、各セルにはその組み合わせの違反率（%）が表示されます。違反率が高いほど赤系、低いほど緑系で表示されます。

- 「どのオペレーションがどのリスクに弱いか」を一目で把握できるウィジェットです。
- 例えば、特定のオペレーションの回答の適切性の列が赤く表示される場合、そのプロンプトが質問の意図と異なるレスポンスを頻繁に生成しているという意味です。
- 右上の**違反基準**ドロップダウンでしきい値を調整すると、セルの色と値もあわせて更新されます。

ハルシネーションスコア分布

Property	Value
チャートタイプ	ヒストグラム（X軸: スコア区間、Y軸: 評価呼び出し数）
集計単位	スコア区間（0～10% … 100%）
アクション	マウスオーバー時に値を表示

ハルシネーション評価スコアの分布を、スコア区間別の呼び出し数で表示します。

- スコアが低い区間（0～10%）に分布が集中している場合、ほとんどのレスポンスでハルシネーションが疑われないという意味です。
- 高いスコア区間（特に100%）に厚い裾が現れる場合、事実の歪みが疑われるレスポンスが一定量存在するため、違反ケースで実際の事例を確認してください。

モデル別の品質比較

Property	Value
チャートタイプ	バーチャート（X軸: モデル）
集計単位	モデル
アクション	マウスオーバー時に値を表示

モデル別のレスポンス品質スコアをバーで比較します。

- 品質スコアが低いモデルは、同じワークロードでより多くの違反を引き起こす可能性があります。
- コスト・レイテンシの指標とあわせて検討し、品質に対する効率が良いモデルへの切り替えを判断してください。

事実性スコア分布

Property	Value
チャートタイプ	ヒストグラム（X軸: スコア区間、Y軸: 評価呼び出し数）
集計単位	スコア区間（0～10% … 100%）
アクション	マウスオーバー時に値を表示

事実性（Factuality）評価スコアの分布を、スコア区間別の呼び出し数で表示します。

- スコアが高い区間（特に100%）に分布が集中している場合、ほとんどのレスポンスが事実に忠実であるという意

味です。

- 低いスコア区間（0～10%）の比重が厚い場合は、事実と異なるレスポンスが一定量存在するため、ハルシネーションスコア分布とあわせて解釈してください。

時間帯別の違反ヒットマップ

Property	Value
チャートタイプ	ヒットマップ（曜日 × 時間帯）
集計単位	曜日（Sun～Sat） × 時間（0～23）
アクション	違反基準（しきい値）の選択

曜日（縦）と時間帯（横）のグリッドで違反の発生密度を表示します。色が濃いセルほど、その時間帯に違反が集中していることを意味します。

- 特定の曜日・時間帯（例: 平日の午後）に違反が集中する場合は、トラフィック負荷や特定のユーザーパターンとの関連性を確認してください。
- 右上の**違反基準**ドロップダウンでしきい値を調整できます。

違反ケース

Property	Value
チャートタイプ	テーブル
アクション	違反基準（しきい値）の選択、行の展開

評価の結果、違反と判定された個別の呼び出しを行単位で一覧表示します。統計ではなく実際の事例を直接確認するためのウィジェットです。

Column	Description
時間	呼び出しが発生した時刻
オペレーション	違反が発生したオペレーション
プロンプト	入力（Input）とレスポンス（Output）の要約
モデル	使用されたLLMモデル名
プロバイダー	LLMプロバイダー
トークン	全トークン数
Latency	リクエスト開始～レスポンス完了までの時間
コスト	その呼び出しのコスト

- 行を展開すると、入力・レスポンスの原文とあわせて Eval 領域に評価項目別のスコアが表示されます。
（Hallucination、Factuality、Relevance、Toxicity、Prompt Injection、PII Leak、URL Scan、Judge）
- 右上の違反基準ドロップダウンで、違反として表示するしきい値を調整できます。

参考

評価カテゴリ

画面上のカテゴリ名と Eval 領域の評価項目は次のように対応します。

Category	Eval Item	Description
ハルシネーション	Hallucination	コンテキストにない内容を生成した度合い
事実性	Factuality	レスポンスが事実に合致している度合い
回答の適切性	Relevance	レスポンスが質問の意図に合致している度合い

Category	Eval Item	Description
有害性	Toxicity	攻撃的・有害な表現が含まれている度合い
プロンプトインジェクション	Prompt Injection	システムプロンプトの奪取・回避の試み
PII漏洩	PII Leak	個人識別情報の露出度合い
-	URL Scan	レスポンスに含まれるURLの危険性
-	Judge	LLM Judgeの総合判定スコア

違反基準（しきい値）

- カテゴリ別の違反推移、違反率ヒットマップ、時間帯別の違反ヒットマップ、違反ケースの各ウィジェットでは、右上で**違反基準**（スコアのしきい値）を選択できます。
- しきい値を調整すると、そのウィジェットが違反として集計する基準が変わり、データもあわせて更新されます。
- 評価は全呼び出しのうち一部をサンプリングして実行されるため、違反率は評価された呼び出しを基準に算出されます。

LLMメトリクス用語辞典

このドキュメントはAI Agent Observabilityで使用する主要メトリクスと概念を整理した用語辞典です。ダッシュボード、トークン推移、コスト分析など複数のメニューで共通して使用される指標の定義と単位を確認できます。

LLMパフォーマンス指標

Metric	Description
Latency	LLM APIリクエストからレスポンス完了までの全体所要時間（ms）。ユーザー体感の全体待ち時間
TTFT	Time To First Token。リクエスト後、最初のレスポンストークン到着までの時間（ms）。ストリーミング環境でのレスポンス開始体感時点
TPOT	Time Per Output Token。出力トークン1つの生成平均時間（ms）。高いほど応答が途切れる感覚
Output TPS	Tokens Per Second。1秒あたりの出力トークン数。モデル全体のスループット指標

トークンおよびコスト

Metric	Description
Input Tokens	プロンプトの全入力トークン数。Cached Tokens含む
Output Tokens	モデル生成の応答トークン数。コストに直接影響
Cached Tokens	キャッシュから取得した入力トークン数。Input Tokensのサブセット。通常入力より低単価適用
Cache Hit Rate	全入力トークン中のキャッシュ適中トークン割合（%）。高いほどコスト削減
I/O Ratio	全トークン中の出力トークン割合（%）。ワークロード特性の把握用（入力中心 vs 生成中心）

パーセンタイル

Percentile	Description
p50	全データの50%以下の値。一般的なユーザー体感パフォーマンス
p75	全データの75%以下の値。p50との差で上位リクエストの追加遅延を把握
p95	全データの95%以下の値。SLA基準として主に使用
p99	全データの99%以下の値。上位1%のパフォーマンス、システム安定性の判断基準

品質評価指標

Metric	Description
Hallucination	幻覚スコア。虚偽情報またはコンテキストの矛盾の程度。低いほど良い (0.0 ~ 1.0)
Answer Relevance	回答関連性スコア。質問への適切な回答かどうか。高いほど良い (0.0 ~ 1.0)
Toxicity	有害性スコア。ヘイト・暴言・暴力・性的コンテンツの水準。低いほど良い (0.0 ~ 1.0)
Combined Judge	総合リスク度。 $\max(\text{hallucination}, \text{toxicity}, 1 - \text{answer_relevance})$ 。3指標中の最大リスク値。低いほど良い (0.0 ~ 1.0)
Prompt Injection	プロンプトインジェクション試みの検出程度。低いほど良い (0.0 ~ 1.0)
Factuality	応答の事実正確性スコア。高いほど良い (0.0 ~ 1.0)
PII Leak	個人識別情報 (PII) の漏洩程度

Metric	Description
URL Scan	応答内URLのリスク度
Eval Success	評価自体の成功可否（1 = 正常、0 = judge呼び出し/パース失敗）

タグベースフィルター

AI Agent Observabilityのすべてのデータは、以下のタグ基準でフィルタリングまたはグループ化できます。

Tag	Description	Example
Agent	LLMエージェント（アプリケーションインスタンス）	llm-app-01, llm-app-02
Model	使用されたLLMモデル	gpt-4o, claude-sonnet-4-20250514
Provider	LLM APIプロバイダー	api.openai.com, api.anthropic.com
Operation	ユーザー指定の呼び出し単位ラベル（chain/prompt/agent名など）。未指定時は default。prompt_meta で設定	checkout_chain, rag_query, default
Prompt Version	プロンプト/チェーンのバージョン	v1, v2, v3

OpenMetricsテンプレートリスト

OpenMetricsを初めて使用するユーザーが複雑な設定プロセスなしに素早くメトリクスを収集して可視化できるよう、デフォルトテンプレートと事前定義済みテンプレートを提供します。これにより、よく使用されるオープンソースメトリクスを簡単に適用でき、初期導入の障壁を下げ、運用者がスケーラブルなモニタリング環境を構築できるよう支援します。

テンプレートリスト

テンプレート一覧はデフォルトの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、テンプレートへの移動、収集案内が含まれています。

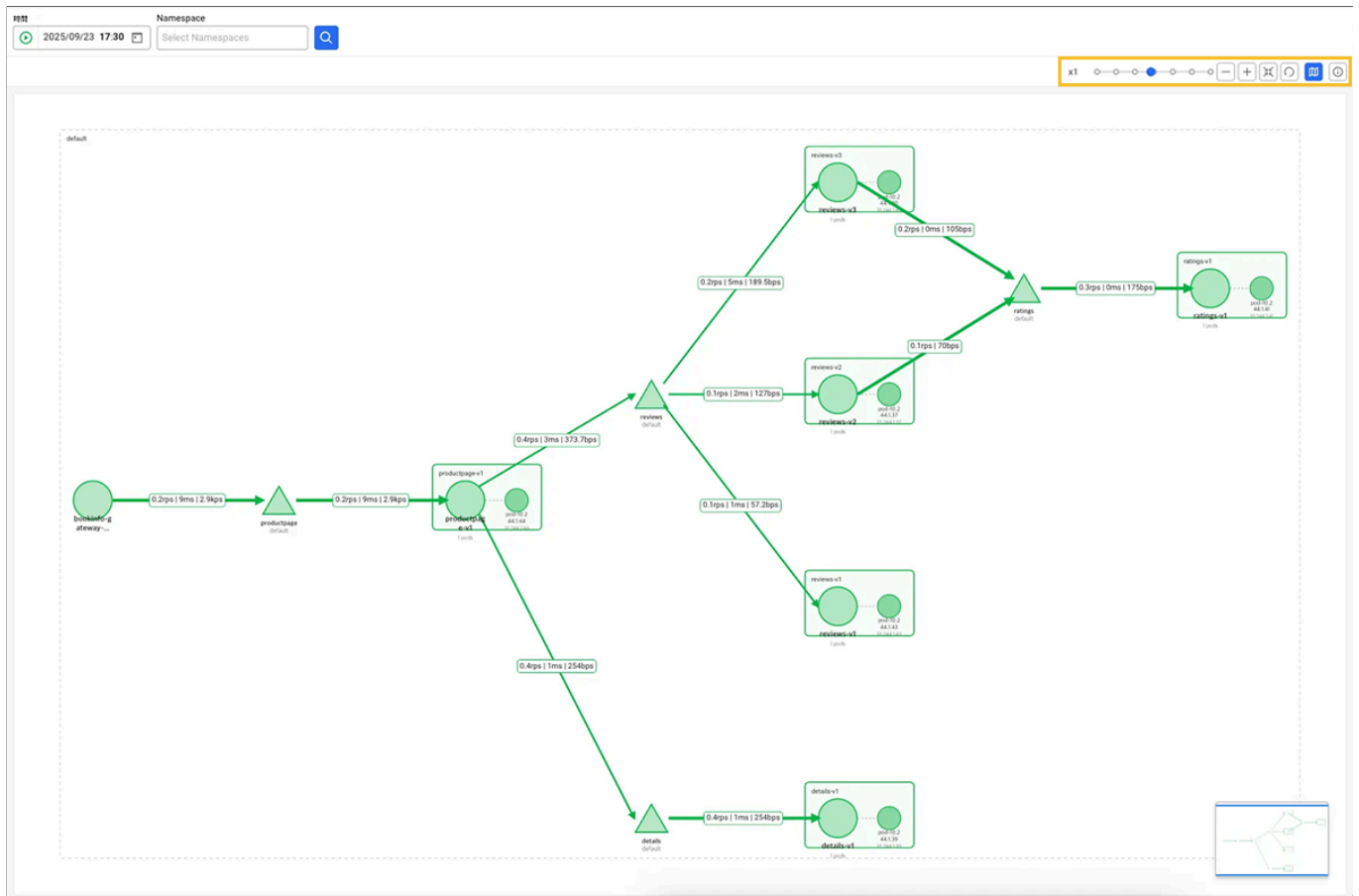
- → **テンプレート**: クリックすると該当テンプレート画面にそのまま移動します。
- ⓘ **収集案内**: クリックするとデータ収集設定方法を提供します。

Nginxテンプレート

Apache Nginxモニタリングにより、Webサーバーのパフォーマンスをリアルタイムで追跡・分析できます。

Istioテンプレート

Istioサービスメッシュ環境のサービス間関係とトラフィックフローを視覚的に表現し、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。画面右上のボタンをクリックして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。

ノードの形状

- 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
- 小さい円: Pod (ポッド)

ノードのステータス

HTTPリクエスト/レスポンス全体におけるエラー（4xx、5xx）の割合

- 緑色: エラー率5%未満
- オレンジ色: エラー率5%以上
- 赤色: エラー率10%以上
- 灰色: リクエストなし

Podノードのステータス

Kubernetes PodのPhase

- 緑色: Running状態のPod
- オレンジ色: Pending状態のPod
- 赤色: 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（4xx、5xx）の割合

- 緑色: エラー率10%未満
- オレンジ色: エラー率10%以上
- 赤色: エラー率20%以上
- 青色: TCP通信
- 灰色: ステータス不明

OpenMetricsエクスプローラー

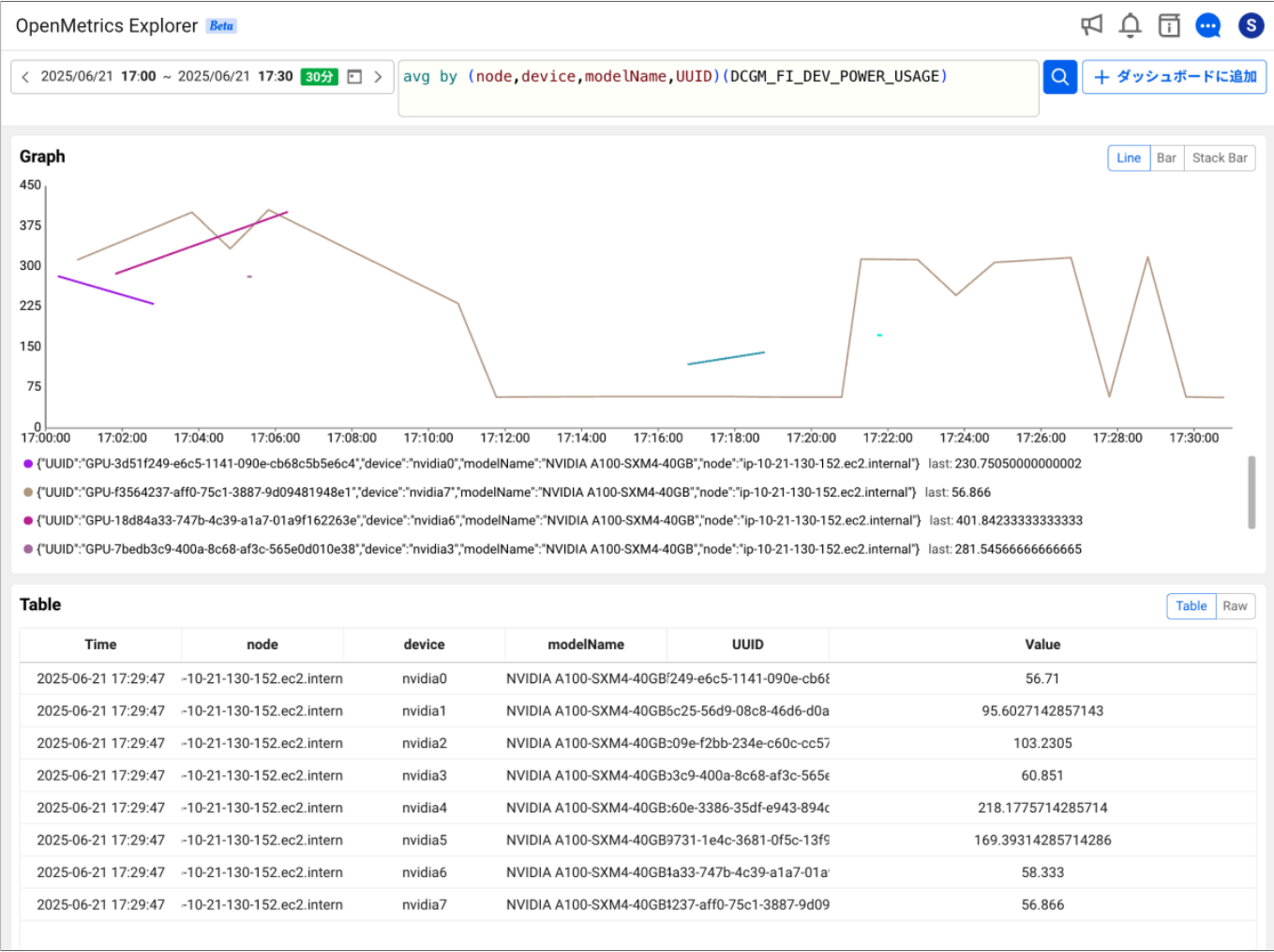
OpenMetrics エクスプローラーは、PrometheusベースのカスタムメトリクスをPromQLで直接探索し、さまざまな形式で可視化してメトリクスの状態を確認したりインサイトを得ることができる機能です。

この機能により、必要なメトリクスを自由に照会し、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できます。

ⓘ OpenMetrics 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。

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。



PromQL構文サポートおよびオートコンプリート

`node_cpu_seconds_total`、`container_memory_usage_bytes`などのメトリクスを入力すると、オートコンプリート機能でメトリクスおよび演算子のリストを提示します。

`{pod="example"}`形式のラベルフィルタリングもサポートしています。

ダッシュボードに追加する

照会したPromQLクエリをダッシュボードウィジェットとして保存し、必要な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. **OpenMetrics エクスプローラー画面の右上にある+ 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開きます。**
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、該当ウィジェットが指定され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名および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位置調整が可能で、PromQLを修正できます。

ライブテール

ⓘ ログ参照権限がない場合は、そのメニューにアクセスできません。

ホーム画面 > プロジェクト選択 > ログ > ログTail

ログTail メニューでは、サーバーコンソールに接続せずに、ログデータストリームをモニタリング画面上で直接確認できます。複雑で大量のログも、フィルター と ハイライト 機能を活用して絞り込み、素早く把握できます。ログデータの取得周期は2秒です。

主な用語は次のとおりです。

- **Category**：ログの収集および照会単位
- **Content**：ログメッセージ
- **Search Key**：ログパーサーの設定によって生成される検索キー
- **Tag**：収集されたログを検索できるタグ

ⓘ ログテーブルカラムの端をドラッグしてカラムの幅を修正できます。

エージェントオプション

エージェントオプションが設定されている場合、ログレベルを収集し、各ログの左側に色付きの縦線が次のように表示されます。

2023-12-18 14:31:02.563	level	INFO	pcode	2277	agenttime	1702877462042	oid	778873916	category	AppLog	loggerName	io.home.test.logback02starter.base.web.LogbackControllerGreeting
												INFO log in our greeting method.
2023-12-18 14:31:02.563	level	WARN	pcode	2277	agenttime	1702877462043	oid	778873916	category	AppLog	loggerName	io.home.test.logback02starter.base.web.LogbackControllerError
											threadName	http-nio-19090-exec-2
												io.home.test.logback02starter.base.errors.exception.ApiException: [2204] Process failure. Please try again.
2023-12-18 14:31:02.586	level	error	pcode	2277	agenttime	1702877462043	oid	778873916	category	AppServer	event_status	error
											event_status_literal_name	level
												Servlet.service() for servlet [dispatcherServlet] in context with path [] threw exception [Request processing failed: io.home.test.logback02starter.base.error
2023-12-18 14:31:02.586	level	error	pcode	2277	agenttime	1702877462057	oid	778873916	category	AppServer	event_status	error
											event_status_literal_name	level
												Servlet.service() for servlet [dispatcherServlet] in context with path [] threw exception [Request processing failed: io.home.test.logback02starter.base.error
2023-12-18 14:31:02.586	level	error	pcode	2277	agenttime	1702877462081	oid	778873916	category	AppServer	event_status	error
											event_status_literal_name	level
												Servlet.service() for servlet [dispatcherServlet] in context with path [] threw exception [Request processing failed: io.home.test.logback02starter.base.error
2023-12-18 14:31:02.586	level	error	pcode	2277	agenttime	1702877462087	oid	778873916	category	AppServer	event_status	error
											event_status_literal_name	level
												Servlet.service() for servlet [dispatcherServlet] in context with path [] threw exception [Request processing failed: io.home.test.logback02starter.base.error

❗ エージェントのオプション設定

```
# whatap.conf  
weaving=log4j-2.17  
weaving=logback-1.2.8
```

- Javaエージェント2.2.22バージョンからウィービング設定にlog4j-2.17またはlogback-1.2.8設定時に使用できます。エージェントの再起動が必要です。
- ログレベルは、パーシングされたキーワードの中で level 、 type 基準で判別します。 level 、 type にパーシングされたキーが存在し、パーシングの値がFATAL、CRITICAL、ERROR、WARN、WARNING、INFOを含む場合は、ログレベルの色を表示します。

1 フィルター領域

フィルター適用

フィルターを適用すると、入力した条件に合ったログをフィルタリングします。複数のフィルターを入力できます。フィルターのタグが同じ場合、OR(||)、そうでない場合はAND(&&)が適用されます。

入力欄に文字を直接入力するか、**フィルター**入力欄をクリックしてフィルターを指定します。フィルタータグは **検索キー**、**オペレーター**、**検索値** の順に入力します。🔍 **検索** ボタンを選択すると、フィルターが適用されたデータをログ一覧で確認できます。

❗ ガイドUI

次のように入力ウィンドウの下に**ガイドUI**を提供します。入力画面またはフィルタータグにマウスカースールがある場合、**ガイドUI**を閉じる時はESCキーで操作できます。



検索キー、オペレーター、検索値の入力

- **検索キー** 入力時、一般インデックス、予約語インデックス、数字のみ入力できるインデックスを区分して推奨値を提供します。
- **オペレーター** 入力時、一般インデックス検索キーの場合は、`==`、`!=` オプションが下に表示されます。数字のみ入力できるインデックスの場合、`>`、`<`、`<=`、`>=`、`==`、`!=` オプションを提供します。
- **検索値** 入力時の一致検索(`>`、`<`、`<=`、`>=`、`==`)の時は青色で、除外検索(`!=`)の時は赤色でハイライトします。**検索値** 入力時に大文字と小文字の区分オプションを活用して検索することができます。

- ① フィルタータグが2行以上長くなる場合へ **折りたたみ**アイコンを選択して折りたたむことができます。
- フィルタータグを入力した後、入力画面でShiftとTabキーを同時に使用して、以前のフィルタータグに移動できます。
- フィルタータグを入力した後、入力画面でTABキーを通じて次のフィルタータグに移動できます。

フィルタータグ追加

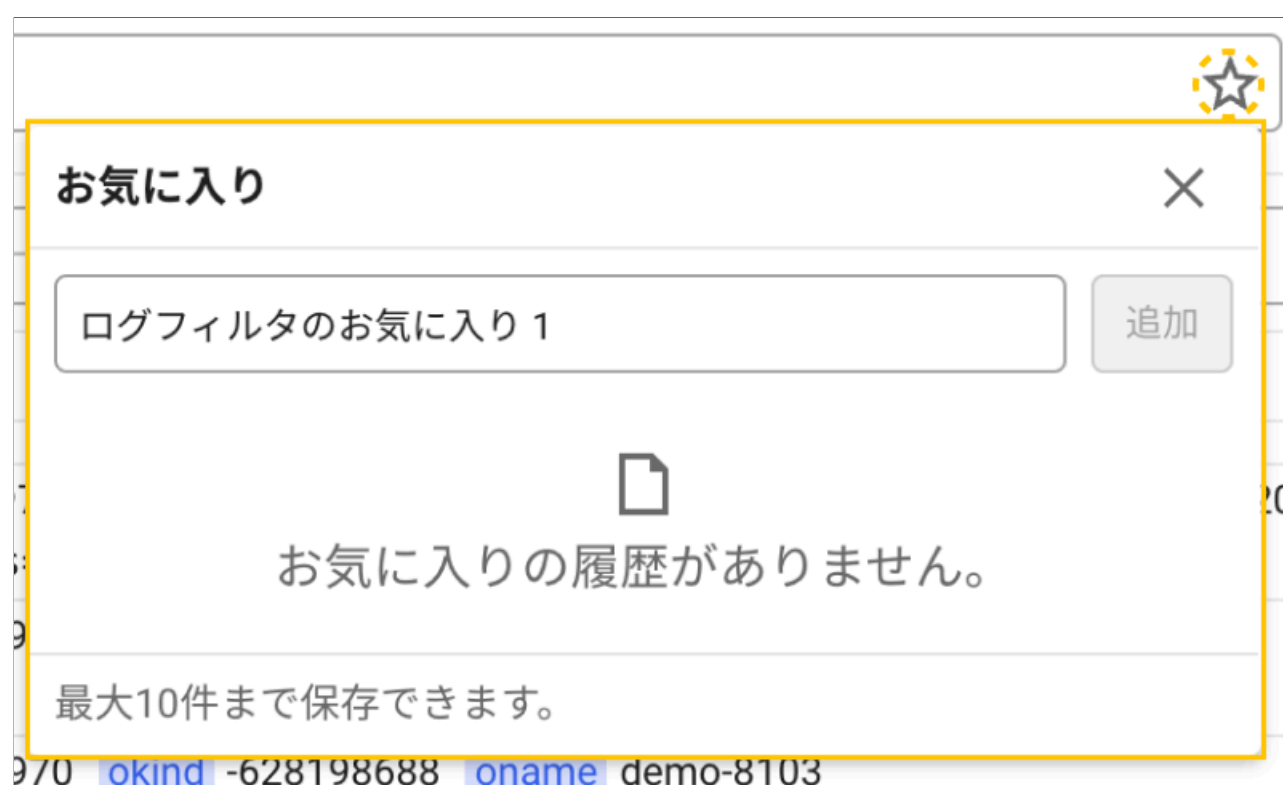
入力フィールドにテキストを入力し、Enter または Tab キーを押すか、入力欄下のガイド UI で推奨値をクリックまたは矢印キーで選択して追加できます。

フィルタータグの削除

単一のタグを削除するには、**Backspace** キーを押すか、タグ上の **X** アイコンをクリックします。すべてのタグを一度に削除するには、入力フィールド右側の **X** アイコンをクリックします。

フィルターのお気に入り

フィルターお気に入り機能を提供します。入力画面の右側にある ☆ お気に入り アイコンを選択して、目的のフィルター検索条件をお気に入りに追加、削除および既存のお気に入りを選択できます。お気に入りは最大50個まで保存されます。



フィルター適用の例外状況

- 数字のみ入力できるインデックス(.n で終わる 検索キー)を入力したタグで、 検索値 は数字のみ入力できます。
- 重複した 検索キー 、 検索値 は入力できません。
- 検索キー 、 検索値 いずれかのタグが存在しない場合は、検索できません。有効でないタグの場合、灰色と表示します。

❗ **ログTail** 検索キー として category は入力できません。

入力フィルター値の下にある数式(expression)は、ログデータを表示するときに適用されるフィルター式のプレビューです。

パースングされていないキーワードのフィルター適用

ログからパースングされていない、つまりインデックスが作成されていないキーワードを含むログを照会できます。この場合、指定範囲内のすべてのログをFull Scanします。そのため、インデックスが生成されたキーに比べて検索速度が多少落ちる可能性があります。定型化されたログデータの場合、**ログパーサの設定**を通じてインデックスキーを活用した検索をお勧めします。

フィルター ⓘ

▼ content == *select* ×

フィルターを入力してから「Enter」キーを押して追加

content == *select*

1. カテゴリを選択します。
2. フィルター入力欄に content を選択後、検索キーワードを入力します。
例) content *select*
3. 🔍 検索ボタンをクリックしてログを照会します。

- ❗ **ログTail**の場合、すべてのログ検索が可能であるため、カテゴリを指定する必要はありません。
- パーサ設定の詳細については、[次の文書](#)を参考にしてください。

フィルター修正

フィルターに値を入力し、入力した値をクリックすると、その値を修正できます。

フィルター ⓘ

🔍 oname == ote-account X key == aws.base:index X appender != fileAppender X フィルターを入力してから「Enter」キーを押して追加 X

☐ 大小文字を区別する

fileAppender

searchValue*	検索値で始まるログを検索します。 ex key!=searchValue*
*searchValue	検索値で終わるログを検索します。 ex key!=*searchValue
searchValue	検索値を含むログを検索します。 ex key!=*searchValue*
*search*Value*	検索値を含むログを検索します。 ex key!=*search*Value*
re:searchValue	正規表現に一致するログを検索します。 ex key!=re:searchValue
**	検索語に該当するすべてのログを検索します。 ex key!=**

key == value 一致 key != value 除外 key == value 無効なタグです。 📖 ガイド表示

- フィルター入力欄にテキストを書き換えて修正できます。
- フィルター入力欄の下に表示されるガイドUIから推奨値を選択して修正できます。

検索キー(Search Key)

次の画像で青いボックス部分はパーシング(parsing)された検索キーです。検索キーはログ設定のログパーサの設定タブでパーシングロジックを登録し、設定できます。

検索キー

フィルター入力の文法

タグは検索キーと検索値で構成されています。次の例で、検索キーは `exception`、検索値は `UnknownHostException` となります。この例は、収集したログデータのうち、IPアドレスとドメインアドレスが一致せず、サーバーをホストに接続できない場合に発生する例外(`UnknownHostException`)が含まれたログデータを照会

します。

検索キーの種類

検索キーの種類	検索キーフォーマット	意味	検索キーと検索値の例	検索例
文字列キーワード	keyword	ファイル名	- キー: fileName - 値: /data/whatap/logs/yard.log	fileName:/data/whatap/logs/yard.log
数字キーワード	keyword.n	レスポンス時間	- キー: response_time.n - 値: 2945	response_time.n>=2945
予約語キーワード (辞書定義キーワード)	@keyword	TXID	- キー: @txid - 値: 85459614215434144	-
ログ本文キーワード	content	ログ本文	- キー: content - 値: ユーザー入力値	content: *ERROR*

ⓘ Content検索キー

- Content検索キーは、インデックスされていないログの本文を対象に検索します。例えば、`content *ERROR*` のように入力する場合は、ログ本文中で ERROR を含めたログを検索します。
- どのキーワードでインデックスをかけるべきかわからない場合は、Content検索キーを活用して問題となるキーワードを含むログを識別します。その後、**ログ設定メニュー**のログパーサ設定を通じて、該当キーワードでパーサを設定し、インデックスを生成する方式で、検索速度を早くできます。

共通文法

文法の種類	説明	例
<code>==searchValue</code>	検索値と一致するログを検索します。	<code>exception==RuntimeExceptionexception</code>
<code>!=searchValue</code>	検索値を除いたログを検索します。	<code>exception!=RuntimeException</code>
<code>*searchValue</code>	検索値で終わるログを検索します。	<code>word==*hello</code>
<code>searchValue*</code>	検索値で始まるログを検索します。	<code>word==hello*</code>
<code>*searchValue*</code>	検索値として中間に含まれるログを検索します。	<code>word==*hello*</code>
<code>*search*Value*</code>	検索値として含まれるログを検索します。	<code>word==*he*llo*</code>
<code>re:{正規表現}</code>	正規表現にマッチングされるログを検索します。	<code>caller==re:^(i\.w\.a\.w\.s\.v\.r\.</code>
<code>**</code>	検索語に該当するすべてのログを検索します。	

検索キーが数字キーワード(keyword.n)の場合の文法

次の文法は検索キーが、`keyword.n` 形式の場合のみに対応します。

- 検索値は数字のみ使えます。
- `.n` キーワードの値にはPrefixを付けません。`.n` 以外のキーワードは、すべてPrefixを付ける必要があります。

例) `+>searchValue` は有効ではありません。

文法の種類	説明	例
<code>>searchValue</code>	検索値よりも大きな値を含むログを照会します。	<code>response_time.n>3000</code>
<code>>=searchValue</code>	検索値よりも大きい、同じ値を含むログを照会します。	<code>response_time.n>=3000</code>
<code>==searchValue</code>	検索値と同じ値を含むログを照会します。	<code>response_time.n==3000</code>

文法の種類	説明	例
<code>!=searchValue</code>	検索値とは異なる値を含むログを照会します。	<code>response_time.n!=3000</code>
<code><searchValue</code>	検索値より小さい値を含むログを照会します。	<code>response_time.n<3000</code>
<code><=searchValue</code>	検索値より小さいか、同じ値を含むログを照会します。	<code>response_time.n<=3000</code>

個人情報の照会

ログの個人情報照会権限がある場合**個人情報の表示**  トグルボタンの活性化を通じて非識別化処理した個人情報を表示することができます。

- ① ログの個人情報照会権限についての詳細は[次の文書](#)を参照してください。
- ログ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 コンテンツハイライト領域

ログのコンテンツ内で目的のキーワードを簡単に識別できるよう、ハイライト機能を提供します。単一または複数のキーワードでフィルタリングできます。

ログTail

カテゴリー

AppLog

フィルター

content

+*select*

×

フィルターを入力してから「Enter」キーを押して追加します。

content == +*select*

act

🔍

⏸

🔄

🔍

⚙

	oname	タイムスタンプ	ログ
▶	demo-8102	2023-11-24 14:06:43.648	@txid 4785826165275137961 pcode 5490 oname demo-8102 onodeName node-0 oid 1387800924 okindName demo-okind-0 okind -1383513482 select distinct ename, deptno, sal, job from emp
▶	demo-8102	2023-11-24 14:06:43.654	@txid 4785826165275137961 pcode 5490 oname demo-8102 onodeName node-0 oid 1387800924 okindName demo-okind-0 okind -1383513482 select ename, 1000 from emp
▶	demo-8102	2023-11-24 14:06:43.659	@txid -240363117644553212 pcode 5490 oname demo-8102 onodeName node-0 oid 1387800924 okindName demo-okind-0 okind -1383513482 select distinct pp.lastname, pp.firstname
▶	demo-8102	2023-11-24 14:06:43.659	@txid 4785826165275137961 pcode 5490 oname demo-8102 onodeName node-0 oid 1387800924 okindName demo-okind-0 okind -1383513482 select 'total income is', ((orderqty * unitprice) * (1.0 - unitpricediscount)), ' for '

1. ハイライトしたいキーワードをキーワード入力欄に入力します。

- 例: `select`

2. 検索したキーワードがログリストのContent内でハイライト表示されます。

-  **拡大表示** アイコンをクリックすると、**ログ** と **タイムスタンプ** を全画面で確認できます。
- ハイライトされたキーワード間の移動はショートカットで操作できます。
 - `Enter` : 次の行へ移動
 - `Shift+Enter` : 前の行へ移動
 - `↑ / ↓` アイコン: 行の移動
 - `Cmd + F` (Mac) / `Ctrl + F` (Windows): キーワード検索

複数キーワード条件

複数キーワードでハイライトする場合は、次のように作成します。

入力文字列	説明	結果
<code>a b c</code>	各キーワードをスペースで区切ります。	a, b, c
<code>"Whatap is good."</code>	キーワードにスペースを含める場合は、 <code>"</code> または <code>"</code> で囲みます。	Whatap is good.
<code>"Whatap\\ is good."</code>	<code>"</code> で囲まれたキーワードに、 <code>\</code> を含める場合は、 <code>\\</code> と入力する必要があります。	Whatap\\ is good.

ハイライトカラー設定

ハイライトするキーワードおよび色を設定できます。

- デフォルトでは、ログレベルに応じたハイライト (**FATAL**、**ERROR**、**WARN**、**SEVERE**、**CRITICAL**) が適用されています。
- 設定した内容は**プロジェクト単位**で保存されます。

1.  アイコンをクリックします。

2. **ハイライト** ウィンドウで、色を設定するキーワードを入力欄に追加してください。
3. 入力欄の左側の色をクリックすると、色を選択できます。

テーブル設定

カラム設定

2 領域の右で  **カラム設定** ボタンを選択します。カラムを追加したり、順序を設定できます。

- **カラム追加**

タグを選択してテーブルにカラムを追加できます。**log**カラム選択を解除する場合**ログ表示詳細設定**が確認できません。必ず1つ以上のカラムを選択します。

- **カラム順序設定**

カラムを追加すると**表示項目**に該当カラムが追加されます。希望のカラムをドラッグしてカラムの順序を変更します。

ログ表示詳細設定

2 領域右で  **ログ表示詳細設定** ボタンを選択します。デフォルトで**content**、**タグ**すべてチェックされており、両方の項目を表示します。**content**と**タグ**どちらかは必ず選択します。

チェックが解除されたアイテムはテーブルに表示されません。次のように**タグ**を解除する場合、テーブルでログの**タグ**は表示されません。

ログテーブルコンテンツタグ非表示

タグ管理一覧にタグを追加すると、追加した順にログのタグがソートされます。タグの順序はドラッグして変更できます。追加したタグを無効にすると、無効にしたタグはログのタグに表示されません。

❗ **カラムおよびログ表示設定の適用範囲**

- **カラム設定**と**ログ表示詳細設定**は、**ログTail**、**ログ検索**、**ログトレンド**で使用できます。
- 同一プロジェクト内の**ログTail**、**ログ検索**、**ログトレンド**は、**カラム設定**と**ログ表示詳細設定**を共有します。

ログファイルのダウンロード

検索したログの結果を **CSV** または **TXT** ファイル形式でダウンロードできます。

1. 画面中央右側の **↓ ダウンロード** ボタンをクリックします。
2. **CSVダウンロード** または **Logダウンロード** をクリックして、希望の形式のファイルをダウンロードします。

ログエクスプローラー

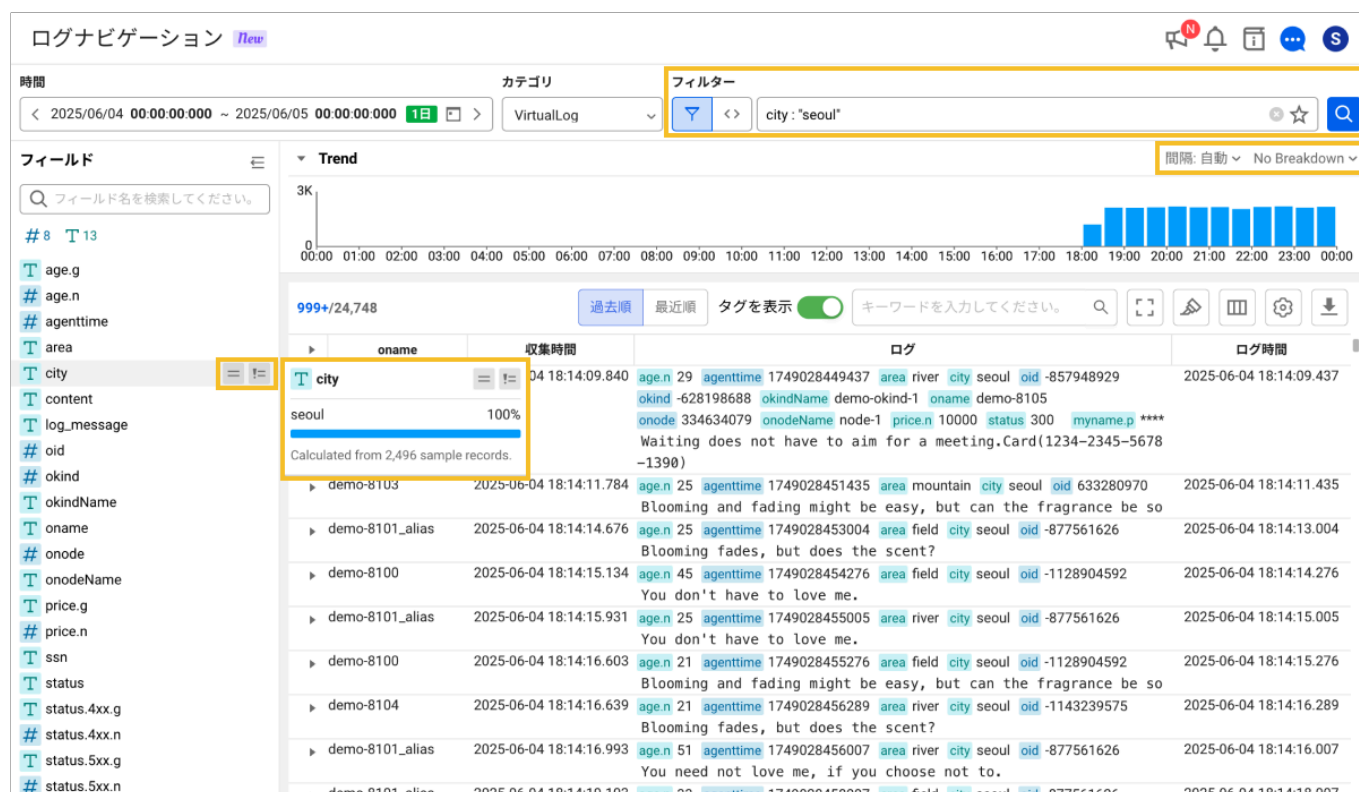
ⓘ ログ参照権限がない場合は、そのメニューにアクセスできません。

ホーム画面 > プロジェクト選択 > ログ > ログエクスプローラー *New*

ログエクスプローラー (Log Explorer) 機能は、ログデータを迅速に検索、フィルタリング、分析します。フィールド構造に関する情報を把握し、クエリベースのフィルタリングによって必要なログを効率的に検索し、結果をデータの可視化を通じて確認できます。本機能はLuceneインデックスをベースに動作し、主な特徴は以下のとおりです。

1. **ログトレンド分析および検索:** ログのトレンドと検索機能を同時にサポートし、時間経過によるログの変化傾向を把握できます。
2. **高速検索およびフィルタリング:** ユーザーは簡単なキーワード入力だけで、問題のあるログを迅速に検索できます。WhaTapログ検索クエリおよびLuceneクエリ言語に対応しており、さまざまな検索条件を設定できます。
3. **オートコンプリート機能:** クエリ入力時にオートコンプリート機能が提供され、ユーザーが簡単にクエリを作成し、習得できるようサポートします。
4. **データ可視化:** スタックグラフを活用して時系列のログ件数データを視覚的に表示し、ドリルダウン機能を通じて特定のログを深く分析できます。
5. **お気に入りと共有:** よく使うフィルターはお気に入りとして保存でき、URLを通じて他のユーザーと簡単に共有できます。

基本画面ガイド



時間設定

上部のオプションバーにある**タイムセレクター**を使用して、ログ検索の時間範囲を設定できます。

カテゴリ

上部のオプションバーにある**カテゴリ選択**で、探索したいログの種類を選択します。（例：AppLogなど）

フィルター

上部オプションバーの**フィルター**でクエリ言語を選択し、検索条件を入力してフィルターを適用します。
検索クエリ構文は、WhaTapログ検索クエリとLuceneクエリの両方をサポートしています。

-  : WhaTapログ検索クエリ
-  : Luceneクエリ

検索クエリ構文

WhaTap log search query

WhaTapのログ検索クエリはオートコンプリート機能を提供しており、簡単なキーワード入力だけで問題のログを迅速に検索できます。and、orなどのさまざまな構文を活用して、さまざまな検索条件を入力できます。

- 例、検索フィルターに `error` と入力するだけで、content（ログ本文）を検索できます。
 1. 上部オプションバーでフィルターアイコンをクリックした後、クエリ検索ウィンドウでフィールド（選択した時間とカテゴリに応じたリストが表示されます）を選択してください。
 2. 演算子を選択してください。
 - a. 基本: `:`、`*`、`and`、`or`
 - b. 選択したフィールドが `Number` 型の場合（基本 + `<=`、`>=`、`<`、`>`）
 - c. 選択したフィールドが `String` の場合（基本のみ）
 3. 値（選択した時間、カテゴリ、フィールドに一致するリストが表示されます）を選択してください。

検索インデックスフィールドタイプ

サポートされているフィールドタイプは、`Number`、`String`、`Boolean` の3種類のみです。

- `Number` : Long、double、float 型の数値
- `String` : メール本文のような全文テキスト
- `Boolean` : True または False の値

 WhaTapログ検索クエリの構文に関する詳細は[次の文書](#)を参照してください。

Lucene

Apache Lucene 8.11.1バージョンのオープンソースの構文がそのまま利用可能です。ただし、自動補完機能は提供していません。

❗ LuceneのQueryParserクラシックパッケージに関する詳細は[公式ドキュメント](#)を参照してください。

フィールド一覧

ドリルダウン分析に活用できます。

フィールド情報のフィルターを反映する

フィールド一覧でフィールド名を確認し、`=` または `!=` ボタンをクリックして条件をフィルターに適用してください。

- フィールドにマウスオーバーすると、`=`、`!=` ボタンが表示され、クリックするとフィルターに反映されます。
- フィールド検索機能を使って、目的のフィールドを素早く見つけることができます。

Trendチャート

Trendチャートでインターバルやブレイクダウンを適用して、ログをグラフで確認できます。

- グラフィインタラクション:** マウสดラッグや  ボタンで時間のズームイン、バークリックで該当領域のログを確認可能
- インターバル:** 自動、1秒、5秒、30秒、1分、5分、10分、30分、1時間から選択可能
- Breakdown:** 特定のフィールドを基準にTop 3 + Othersの値を分布として可視化し、ドリルダウンフィルターの適用が可能

breakdown

Breakdownは、ログ件数の推移を特定のフィールドで分析する可視化機能です。

- 特定のフィールドのTop 3の値とOthers（その他の値）を区別して、スタックグラフで分布を表示します。
- ドリルダウン分析に活用でき、凡例やスタックグラフの特定領域（部分グラフ）をクリックして、特定の値のみを

強調したり、フィルター条件を追加することができます。

ログの確認

検索条件に一致するログの一覧を確認でき、**フィールド一覧**ではフィールド情報とログタグが同じ色で表示されます。

コンテンツハイライト

ログのコンテンツ内で目的のキーワードを簡単に識別できるよう、ハイライト機能を提供します。単一または複数のキーワードでフィルタリングできます。

1. ハイライトしたいキーワードをキーワード入力欄に入力します。
 - 例: `select`
2. 検索したキーワードがログリストのContent内でハイライト表示されます。
 -  **拡大表示** アイコンをクリックすると、**ログ** と **タイムスタンプ** を全画面で確認できます。
 - ハイライトされたキーワード間の移動はショートカットで操作できます。
 - `Enter` : 次の行へ移動
 - `Shift+Enter` : 前の行へ移動
 - `^ / v` アイコン: 行の移動
 - `Cmd + F` (Mac) / `Ctrl + F` (Windows): キーワード検索

複数キーワード条件

複数キーワードでハイライトする場合は、次のように作成します。

入力文字列	説明	結果
<code>a b c</code>	各キーワードをスペースで区切ります。	a, b, c
<code>"Whatap is good."</code>	キーワードにスペースを含める場合は、 <code>"</code> または <code>"</code> で囲みます。	Whatap is good.

入力文字列	説明	結果
"Whatap\\ is good."	""で囲まれたキーワードに、\を含める場合は、\\と入力する必要があります。	Whatap\ is good.

ハイライトカラー設定

ハイライトするキーワードおよび色を設定できます。

- デフォルトでは、ログレベルに応じたハイライト（FATAL、ERROR、WARN、SEVERE、CRITICAL）が適用されています。
- 設定した内容はプロジェクト単位で保存されます。
 - 🔗アイコンをクリックします。
 - ハイライト ウィンドウで、色を設定するキーワードを入力欄に追加してください。
 - 入力欄の左側の色をクリックすると、色を選択できます。

カラム設定

☐ ボタンをクリックして、カラムを追加したり順序を設定することができます。

カラム追加

カラム追加：タグを選択して、テーブルにカラムを追加できます。

❗ ログカラムの選択を解除すると、ログ表示詳細設定を確認できません。必ず1つ以上のカラムを選択します。

カラム順序設定

カラム設定でカラムを追加すると、表示項目に該当カラムが追加されます。希望するカラムをドラッグして、カラムの順序を変更してください。

ログ表示詳細設定

⚙ ボタンをクリックして、ログの表示を詳細に設定します。デフォルトでは **content**、**タグ** の両方がチェックされており、両方の項目が表示されます。

- **content**と**タグ**のいずれかを必ず選択してください。
- **タグ管理**リストにタグを追加すると、追加した順にログのタグが表示されます。タグの順序はドラッグで変更できます。追加したタグを無効にすると、無効にしたタグはログのタグに表示されません。
- チェックが解除された項目は、テーブルに表示されません。次のように**タグ**を解除する場合、テーブルからログの**タグ**は表示されません。

log table contents remove tag sc

ⓘ カラムおよびログ表示設定の適用範囲

- **カラム設定**と**ログ表示詳細設定**は、**ログTail**、**ログ検索**、**ログトレンド**で使用できます。
- 同一プロジェクト内の**ログTail**、**ログ検索**、**ログトレンド**は、**カラム設定**と**ログ表示詳細設定**を共有します。

ログファイルのダウンロード

検索したログの結果を **CSV** または **TXT** ファイル形式でダウンロードできます。

1. 画面中央右側の **↓ ダウンロード** ボタンをクリックします。
2. **CSVダウンロード** または **Logダウンロード** をクリックして、希望の形式のファイルをダウンロードします。

ログトレンド

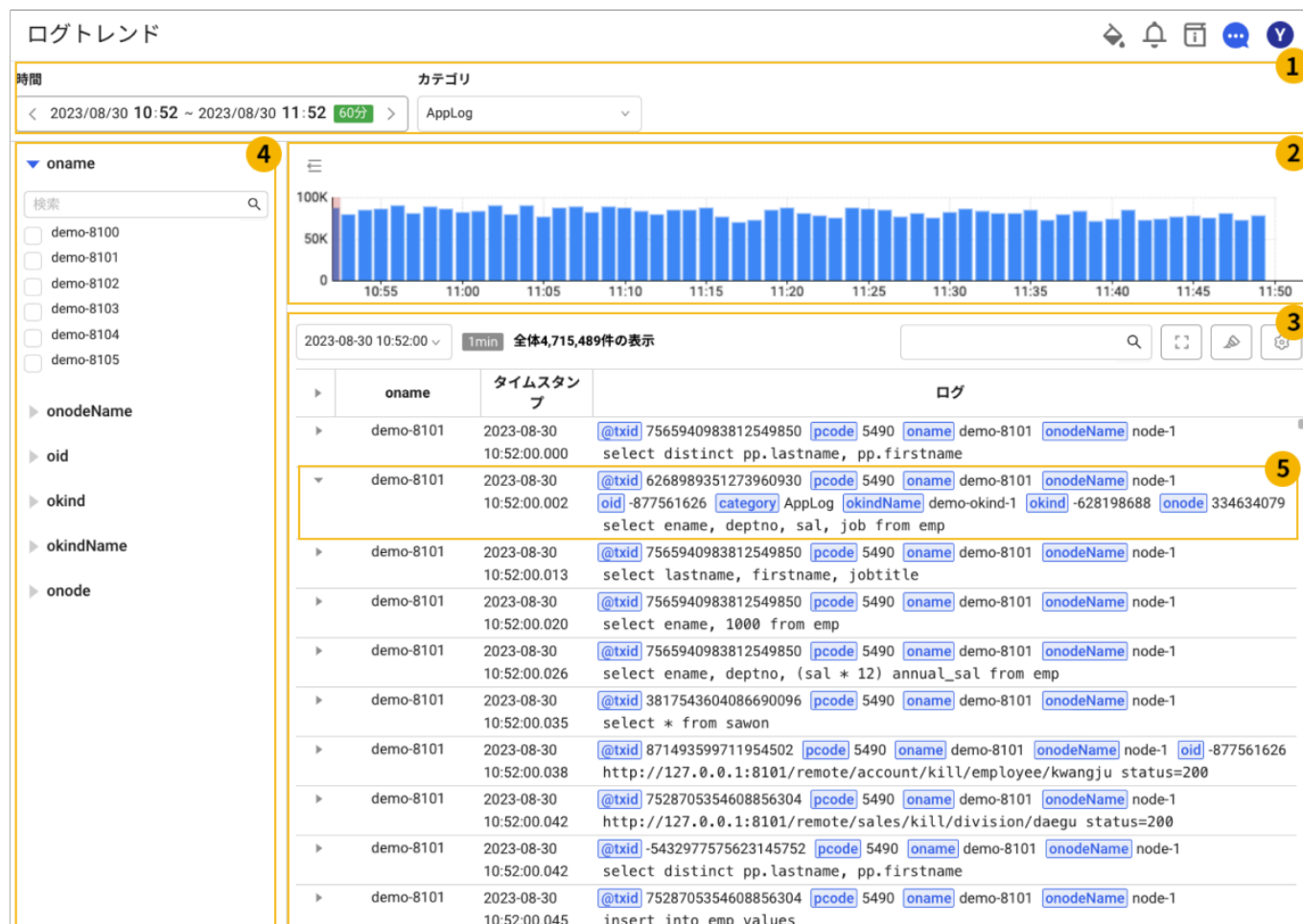
ⓘ ログ参照権限がない場合は、そのメニューにアクセスできません。

ホーム画面 > プロジェクト選択 > ログ > ログトレンド

ログトレンドでは、タイプ別に分類されたログの発生件数の推移を通じて、特定エラータイプの発生パターンを確認し、時間別の詳細ログデータを確認できます。ハイライト機能により、目的のログをすばやく識別できます。カテゴリ別に収集されたログの推移を照会できます。

主な用語は次のとおりです。

- **Category**：ログの収集および照会単位
- **Content**：ログメッセージ
- **Search Key**：ログパーサーの設定によって生成される検索キー
- **Tag**：収集されたログを検索できるタグ



データ参照

- 下部までスクロールすると、次のデータを表示します。
- 1で検索するログデータの時間と収集単位としてカテゴリを指定します。
- カテゴリを変更すると、選択されたカテゴリに対応するログを表示します。2バーチャートと3ログテーブルで確認できます。
- 2棒グラフのバーをクリックすると、バーの時間範囲に対応するログを検索します。
- 3ログテーブルの左上から照会されたログの総数を確認できます。
- 3 ログテーブル上段の右の「全画面」アイコンを選択すると、ログと収集時間を全画面で確認できます。
- 4 サイドメニューからタグでフィルタリングすることでログを確認できます。検索キーは最大2つまで選択でき、

検索値は複数選択できます。

- エージェントオプションが設定されている場合、ログレベルを収集し、各ログの左側に色付きの縦線が次のように表示されます。

2023-12-18 14:31:02.563	level	INFO	pcode	2277	agenttime	1702877462042	oid	778873916	category	AppLog	loggerName	io.home.test.logback02starter.base.web.LogbackControllerGreeting
2023-12-18 14:31:02.563	level	WARN	pcode	2277	agenttime	1702877462043	oid	778873916	category	AppLog	loggerName	io.home.test.logback02starter.base.web.LogbackControllerError
2023-12-18 14:31:02.586	level	error	pcode	2277	agenttime	1702877462043	oid	778873916	category	AppServer	event_status	error
2023-12-18 14:31:02.586	level	error	pcode	2277	agenttime	1702877462057	oid	778873916	category	AppServer	event_status	error
2023-12-18 14:31:02.586	level	error	pcode	2277	agenttime	1702877462081	oid	778873916	category	AppServer	event_status	error
2023-12-18 14:31:02.586	level	error	pcode	2277	agenttime	1702877462087	oid	778873916	category	AppServer	event_status	error

❗ エージェントのオプション設定

```
# whatap.conf
weaving=log4j-2.17
weaving=logback-1.2.8
```

- Javaエージェント2.2.22バージョンからウィービング設定にlog4j-2.17またはlogback-1.2.8設定時に使用できます。エージェントの再起動が必要です。
- ログレベルは、パースされたキーワードの中で level 、 type 基準で判別します。 level 、 type にパースされたキーが存在し、パースの値がFATAL、CRITICAL、ERROR、WARN、WARNING、INFOを含む場合は、ログレベルの色を表示します。

- ❗ ログテーブルカラムの端をドラッグしてカラムの幅を修正できます。

個人情報の照会

ログの個人情報照会権限がある場合個人情報の表示  トグルボタンの活性化を通じて非識別化処理した個人情報を表示することができます。

- ❗ ログの個人情報照会権限についての詳細は次の文書を参照してください。
- ログ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。

ログContentの確認

Content はログメッセージです。ログカラムの1行目には、パースされたキーと値が表示され、2行目にはContent（ログメッセージ）が表示されます。

- **3** ログテーブルの行(ログ)ごとに ▶ 詳細ボタンがあります。 ▶ 詳細ボタンを選択すると **5** のように、ログの全体 contentを確認できます。

ハイライト設定

目的のキーワードを簡単に識別できるように、ハイライト機能を提供しています。🔍 アイコンをクリックし、キーワードを入力してからEnterキーを押して追加してください。

コンテンツハイライト

ログのコンテンツ内で目的のキーワードを簡単に識別できるよう、ハイライト機能を提供します。単一または複数のキーワードでフィルタリングできます。

1. ハイライトしたいキーワードをキーワード入力欄に入力します。
 - 例: `select`
2. 検索したキーワードがログリストのContent内でハイライト表示されます。
 - **[] 拡大表示** アイコンをクリックすると、**ログ** と **タイムスタンプ** を全画面で確認できます。
 - ハイライトされたキーワード間の移動はショートカットで操作できます。
 - `Enter` : 次の行へ移動
 - `Shift+Enter` : 前の行へ移動
 - `^ / v` アイコン: 行の移動
 - `Cmd + F (Mac) / Ctrl + F (Windows)`: キーワード検索

複数キーワード条件

複数キーワードでハイライトする場合は、次のように作成します。

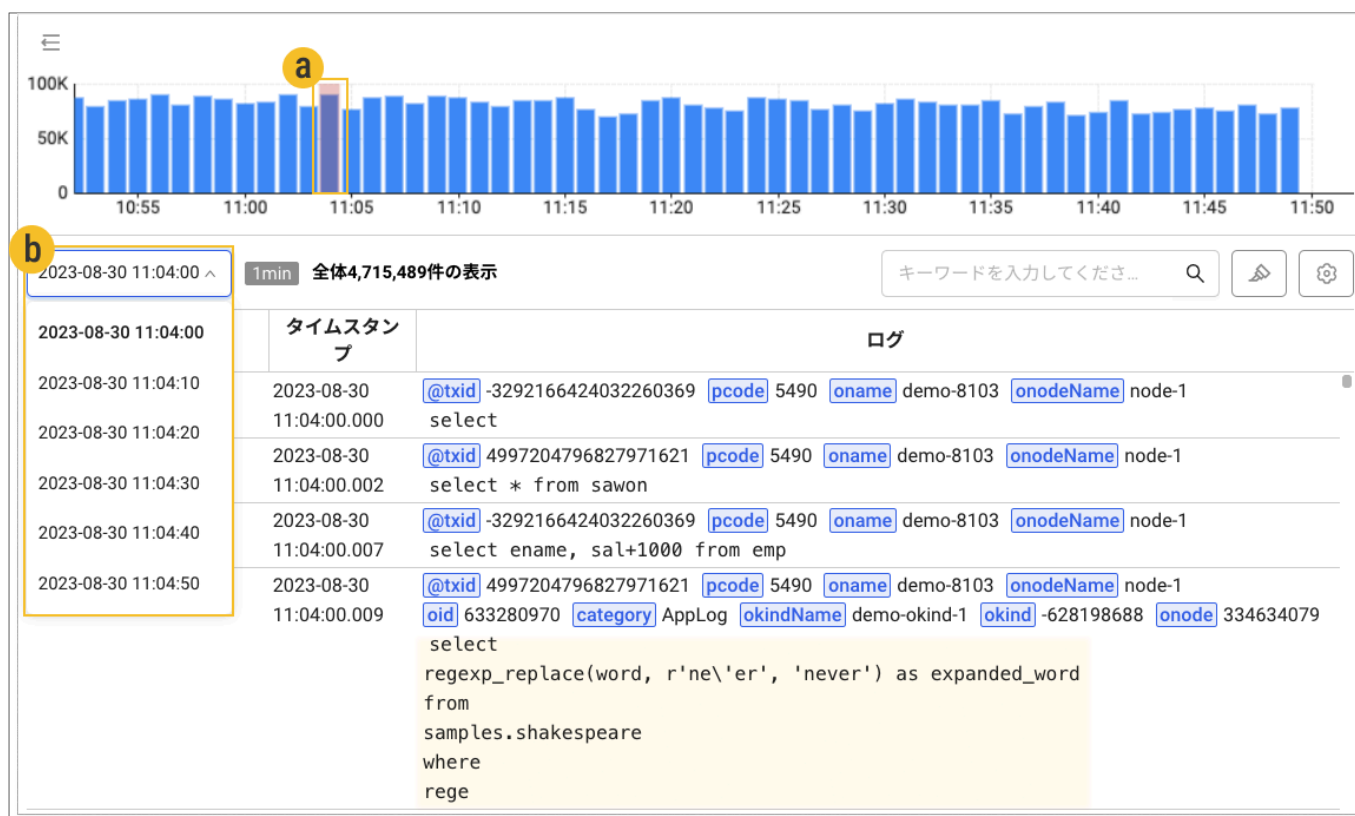
入力文字列	説明	結果
a b c	各キーワードをスペースで区切ります。	a, b, c
"Whatap is good."	キーワードにスペースを含める場合は、" または "" で囲みます。	Whatap is good.
"Whatap\\ is good."	""で囲まれたキーワードに、\ を含める場合は、\\ と入力する必要があります。	Whatap\\ is good.

ハイライトカラー設定

ハイライトするキーワードおよび色を設定できます。

- デフォルトでは、ログレベルに応じたハイライト（**FATAL**、**ERROR**、**WARN**、**SEVERE**、**CRITICAL**）が適用されています。
- 設定した内容は**プロジェクト単位**で保存されます。
 -  アイコンをクリックします。
 - ハイライト** ウィンドウで、色を設定するキーワードを入力欄に追加してください。
 - 入力欄の左側の色をクリックすると、色を選択できます。

ログをチャートで検索



- バーチャートで **a** 目的の時間領域バーをクリックまたはドラッグして、その時間範囲のログを確認できます。
- 棒チャートの下ログテーブル上部左側の **b** 時間選択オプションを使用して、次のように選択した時間帯でより詳細なログ検索が可能です。
 - 1min : interval (チャート内の棒の間隔)
 - 時間選択オプション : 選択された時間範囲を6つの間隔に分割した時間帯

テーブル設定

カラム設定

3 領域の右で  **カラム設定** ボタンを選択します。カラムを追加したり、順序を設定できます。

- **カラム追加**

タグを選択してテーブルにカラムを追加できます。**log**カラム選択を解除する場合**ログ表示詳細設定**が確認できません。必ず1つ以上のカラムを選択します。

- **カラム順序設定**

カラムを追加すると**表示項目**に該当カラムが追加されます。希望のカラムをドラッグしてカラムの順序を変更します。

ログ表示詳細設定

3 領域右で  **ログ表示詳細設定** ボタンを選択します。デフォルトで**content**、**タグ**すべてチェックされており、両方の項目を表示します。**content**と**タグ**どちらかは必ず選択します。

チェックが解除されたアイテムはテーブルに表示されません。次のように**タグ**を解除する場合、テーブルでログの**タグ**は表示されません。

ログテーブルコンテンツタグ非表示

タグ管理一覧にタグを追加すると、追加した順にログのタグがソートされます。タグの順序はドラッグして変更できます。追加したタグを無効にすると、無効にしたタグはログのタグに表示されません。

ⓘ カラムおよびログ表示設定の適用範囲

- **カラム設定**と**ログ表示詳細設定**は、**ログTail**、**ログ検索**、**ログトレンド**で使用できます。
- 同一プロジェクト内の**ログTail**、**ログ検索**、**ログトレンド**は、**カラム設定**と**ログ表示詳細設定**を共有します。

ログファイルのダウンロード

検索したログの結果を **CSV** または **TXT** ファイル形式でダウンロードできます。

1. 画面中央右側の **↓ ダウンロード** ボタンをクリックします。
2. **CSVダウンロード** または **Logダウンロード** をクリックして、希望の形式のファイルをダウンロードします。

ログの検索

ⓘ ログ参照権限がない場合は、そのメニューにアクセスできません。

ホーム画面 > プロジェクト選択 > ログ > ログ検索

ログ検索メニューから収集された大量のログをさまざまな条件で検索し、ユーザーが必要なログを指定できます。複数の検索条件をパーシングされたキーとバリューで指定できるため、目的の条件に一致するログデータのみを抽出します。

動的ページで取得したログデータは、設定された行単位で取得され、スクロールで画面の下まで移動すると自動的に次のデータが取得されて表示されます。

主な用語は次のとおりです。

- **Category**：ログの収集および照会単位
- **Content**：ログメッセージ
- **Search Key**：ログパーサーの設定によって生成される検索キー
- **Tag**：収集されたログを検索できるタグ

1 ログ検索

時間 2023/08/30 11:28 ~ 2023/08/30 12:28 60分

フィルター ① category AppLog フィルターを入力してから「Enter」キーを押して追加します。 category == AppLog

2

3 全体4,424,485件の中、9999+件が検索条件に合致します。 過去 最近 タグを表示 キーワードを入力してください。

4

▶	oname	タイムスタンプ	ログ
▶	demo-8102	2023-08-30 11:28:00.003	@txid -8737177787317841782 pcode 5490 oname demo-8102 onodeName node-0 select productid, specialofferid, avg(unitprice) as [average price],
▶	demo-8102	2023-08-30 11:28:00.012	@txid 8009922019355005360 pcode 5490 oname demo-8102 onodeName node-0 select distinct pp.lastname, pp.firstname
▶	demo-8102	2023-08-30 11:28:00.008	@txid -8737177787317841782 pcode 5490 oname demo-8102 onodeName node-0 select ename, sal+1000 from emp
▼	demo-8102	2023-08-30 11:28:00.017	@txid -8737177787317841782 pcode 5490 oname demo-8102 onodeName node-0 oid 1387800924 category AppLog okindName demo-okind-0 okind -1383513482 onode 1693789385 select e.ename ' departments's name is:' d.dname from emp e, dept d where e.deptno=d.deptno
▶	demo-8102	2023-08-30 11:28:00.017	@txid 5677524395951858637 pcode 5490 oname demo-8102 onodeName node-0 http://127.0.0.1:8102/remote/account/kill/division/daegu status=200
▶	demo-8102	2023-08-30 11:28:00.019	@txid 8009922019355005360 pcode 5490 oname demo-8102 onodeName node-0 select ename, sal=sal+1000 from emp
▶	demo-8102	2023-08-30 11:28:00.030	@txid 8009922019355005360 pcode 5490 oname demo-8102 onodeName node-0 select employeeid, lastname, salary
▶	demo-8102	2023-08-30 11:28:00.025	@txid 8009922019355005360 pcode 5490 oname demo-8102 onodeName node-0 select productid, sum(linetotal) as total
▶	demo-8102	2023-08-30 11:28:00.033	@txid -8302816252130563957 pcode 5490 oname demo-8102 onodeName node-0 http://127.0.0.1:8102/remote/order/delete/dept/jeju status=200
▶	demo-8102	2023-08-30 11:28:00.041	@txid 8009922019355005360 pcode 5490 oname demo-8102 onodeName node-0 select pp.lastname, pp.firstname, e.jobtitle
▶	demo-8102	2023-08-30 11:28:00.036	@txid 8009922019355005360 pcode 5490 oname demo-8102 onodeName node-0 select * from emp

AppLog (4,404,050)

AppStdErr (438)

AppStdOut (19,489)

データ参照

- 画面の下までスクロールすると、次のデータを検索します。一度に、10,000個のログを参照できます。
- 3 ログテーブルの左上に検索されたログの総数を確認することができます。
- ログデータを時間の順と逆順で照会できます。3 ログテーブルの右上の古い順と新しい順の中から照会方式を選択します。
- 時間範囲を指定した後適用ボタンを選択して照会時間を設定し、🔍 検索ボタンを選択してデータを照会します。
- 3 ログテーブルの右上の🔍 拡大表示アイコンを選択すると、ログとタイムスタンプを全画面で確認できます。
- エージェントオプションが設定されている場合、ログレベルを収集し、各ログの左側に色付きの縦線が次のように表示されます。

2023-12-18 14:31:02.563	INFO	2277	1702877462042	778873916	AppLog	io.home.test.logback02starter.base.web.LogbackControllerGreeting	
	INFO log in our greeting method.						
2023-12-18 14:31:02.563	WARN	2277	1702877462043	778873916	AppLog	io.home.test.logback02starter.base.web.LogbackControllerError	http-nio-19090-exec-2
	io.home.test.logback02starter.base.errors.exception.ApiException: [2204] Process failure. Please try again.						
2023-12-18 14:31:02.586	error	2277	1702877462043	778873916	AppServer		
	Servlet.service() for servlet [DispatcherServlet] in context with path [] threw exception [Request processing failed: io.home.test.logback02starter.base.error]						
2023-12-18 14:31:02.586	error	2277	1702877462057	778873916	AppServer		
	Servlet.service() for servlet [DispatcherServlet] in context with path [] threw exception [Request processing failed: io.home.test.logback02starter.base.error]						
2023-12-18 14:31:02.586	error	2277	1702877462081	778873916	AppServer		
	Servlet.service() for servlet [DispatcherServlet] in context with path [] threw exception [Request processing failed: io.home.test.logback02starter.base.error]						
2023-12-18 14:31:02.586	error	2277	1702877462087	778873916	AppServer		
	Servlet.service() for servlet [DispatcherServlet] in context with path [] threw exception [Request processing failed: io.home.test.logback02starter.base.error]						

① エージェントのオプション設定

```
# whatap.conf
weaving=log4j-2.17
weaving=logback-1.2.8
```

- Javaエージェント2.2.22バージョンからウィービング設定にlog4j-2.17またはlogback-1.2.8設定時に使用できます。エージェントの再起動が必要です。
- ログレベルは、パースされたキーワードの中で level 、 type 基準で判別します。 level 、 type にパースされたキーが存在し、パースの値がFATAL、CRITICAL、ERROR、WARN、WARNING、INFOを含む場合は、ログレベルの色を表示します。

- ① ログテーブルカラムの端をドラッグしてカラムの幅を修正できます。

ログContentの確認

Content はログメッセージです。ログカラムの1行目には、パースされたキーと値が表示され、2行目にはContent（ログメッセージ）が表示されます。

- ③ ログテーブルの行(ログ)ごとに ▶ 詳細ボタンがあります。 ▶ 詳細ボタンを選択すると ④ のように、ログの全体Contentを確認できます。
- ログのタグを選択すると、コピー, 検索, 除外検索, 隣接ログを検索できるドロップダウンメニューが表示されます。

フィルター

フィルター適用

① 左側の**時間選択バー**で時間範囲を指定できます。右側から、フィルターを適用すると、入力した条件に合ったログをフィルタリングします。複数のフィルターを入力できます。フィルターのタグが同じ場合、OR(||)、そうでない場合はAND(&&)が適用されます。

入力欄に文字を直接入力するか、**フィルター**入力欄をクリックしてフィルターを指定します。フィルタータグは **検索キー**、**オペレーター**、**検索値** の順に入力します。🔍 **検索** ボタンを選択すると、フィルターが適用されたデータをログ一覧で確認できます。

① ガイドUI

次のように入力ウィンドウの下に**ガイドUI**を提供します。入力画面またはフィルタータグにマウスカーソルがある場合、**ガイドUI**を閉じる時はESCキーで操作できます。



検索キー、オペレーター、検索値の入力

- **検索キー** 入力時、一般インデックス、予約語インデックス、数字のみ入力できるインデックスを区分して推奨値を提供します。
- **オペレーター** 入力時、一般インデックス検索キーの場合は、`==`、`!=` オプションが下に表示されます。数字のみ入力できるインデックスの場合、`>`、`<`、`<=`、`>=`、`==`、`!=` オプションを提供します。
- **検索値** 入力時の一致検索(`>`、`<`、`<=`、`>=`、`==`)の時は青色で、除外検索(`!=`)の時は赤色でハイライトします。 **検索値** 入力時に大文字と小文字の区分オプションを活用して検索することができます。

- ① フィルタータグが2行以上長くなる場合へ **折りたたみ**アイコンを選択して折りたたむことができます。
- フィルタータグを入力した後、入力画面でShiftとTabキーを同時に使用して、以前のフィルタータグに移動できます。
- フィルタータグを入力した後、入力画面でTABキーを通じて次のフィルタータグに移動できます。

フィルタータグ追加

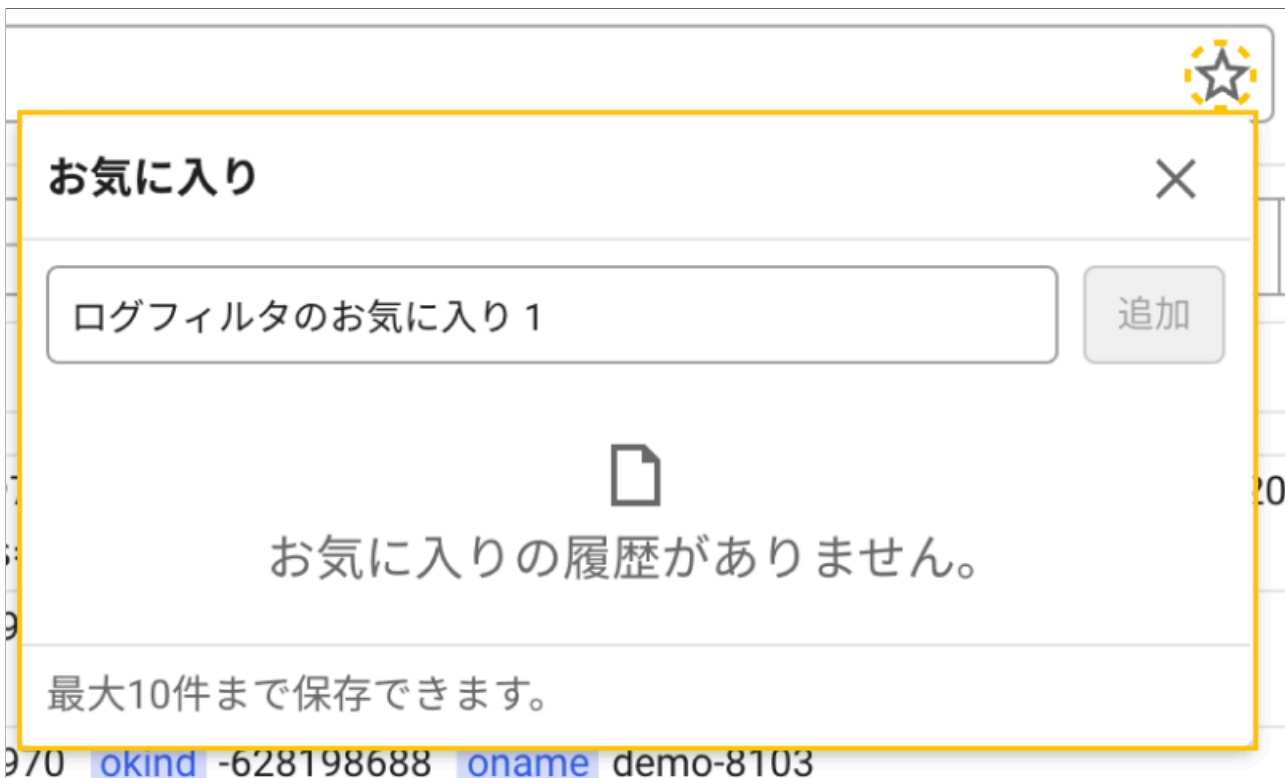
入力フィールドにテキストを入力し、Enter または Tab キーを押すか、入力欄下のガイド UI で推奨値をクリックまたは矢印キーで選択して追加できます。

フィルタータグの削除

単一のタグを削除するには、**Backspace** キーを押すか、タグ上の **X** アイコンをクリックします。すべてのタグを一度に削除するには、入力フィールド右側の **X** アイコンをクリックします。

フィルターのお気に入り

フィルターお気に入り機能を提供します。入力画面の右側にある ☆ お気に入り アイコンを選択して、目的のフィルター検索条件をお気に入りに追加、削除および既存のお気に入りを選択できます。お気に入りは最大50個まで保存されます。



フィルター適用の例外状況

- 数字のみ入力できるインデックス(.n で終わる 検索キー)を入力したタグで、 検索値 は数字のみ入力できます。
- 重複した 検索キー 、 検索値 は入力できません。
- 検索キー 、 検索値 いずれかのタグが存在しない場合は、検索できません。有効でないタグの場合、灰色と表示します。

構文解析されていないキーワードのフィルター適用

ログからパーシングされていない、つまりインデックスが作成されていないキーワードを含むログを照会できます。この場合、指定範囲内のすべてのログをFull Scanします。そのため、インデックスが生成されたキーに比べて検索速度が多少落ちる可能性があります。定型化されたログデータの場合、**ログパーサの設定**を通じてインデックスキーを活用した検索をお勧めします。

フィルター ⓘ

🔍 content == *select* ✕

フィルターを入力してから「Enter」キーを押して追加

✕

content == *select*

1. **カテゴリ**を選択します。カテゴリ指定が必須です。
2. **フィルター**入力欄に **content** を選択後、検索キーワードを入力します。
例) content *select*
3. 🔍 **検索**ボタンをクリックしてログを照会します。ログ全体の一部を先に照会します。1回あたりの検索結果は最大1万件です。
4. スクロールを下げて下段の**追加照会**ボタンを選択すると、つづけて照会できます。

ログ検索

時間: 2024/01/15 14:50 ~ 2024/01/15 15:50 6000

フィルター: category == ApplLog && oid == 397157872 && content != "select"

category == ApplLog && oid == 397157872 && content != "select"

全体の41,513件の内、10,500件を照会し、検索された結果は9999+件です。

AppLog (41,513)

controller debug_point status
exception filename host io
java_class key level log_source
loggerName method namespace
oid okind okindName name
referer region responseTime
send_packet_count service status
status_4xx status_success
status_nxx targetPcode task
taskPoolActiveCount
task_delayedTime threadName
type uri

AppStdErr (2,355)

AppStdOut (56,731)

AppStdOut debug_point status
exception filename host io
java_class key level log_source
loggerName method namespace
oid okind okindName name
referer region responseTime
send_packet_count service status
status_4xx status_success
status_nxx targetPcode task
taskPoolActiveCount
task_delayedTime threadName
type uri

ログをもっと照会しますか？
ログパーサーを追加することで、追加の照会なしで検索ができます。
ログパーサーの設定 追加照会

- ① 全体ログのうち、サーバー照会範囲当たり1万件ずつ照会します。サーバー照会範囲の場合、基本20万件ですが、ログ全体の量によって比率が変わることがあります。
- パーサ設定の詳細については、[次の文書](#)を参考にしてください。

フィルター修正

フィルターに値を入力し、入力した値をクリックすると、その値を修正できます。

フィルター ⓘ

🔍 oname == ote-account × key == aws.base:index × appender != fileAppender × フィルターを入力してから「Enter」キーを押して追加 ×

☐ 大小文字を区別する

fileAppender

searchValue*	検索値で始まるログを検索します。 ex key!=searchValue*
*searchValue	検索値で終わるログを検索します。 ex key!=*searchValue
searchValue	検索値を含むログを検索します。 ex key!=*searchValue*
*search*Value*	検索値を含むログを検索します。 ex key!=*search*Value*
re:searchValue	正規表現に一致するログを検索します。 ex key!=re:searchValue
**	検索語に該当するすべてのログを検索します。 ex key!=**

key == value 一致 key != value 除外 key == value 無効なタグです。 ⓘ ガイド表示

- ・ フィルター入力欄にテキストを書き換えて修正できます。
- ・ フィルター入力欄の下に表示されるガイドUIから推奨値を選択して修正できます。

検索キー(Search Key)

検索キーは、ログデータ内で希望する特定の値にアクセスするための識別子を意味します。検索キーに該当する実際のデータが検索値です。左側の②領域のタグはカテゴリ別にパーシング(parsing)された検索キーです。タグを選択してフィルターを入力します。オレンジ色タグはカテゴリで、青色タグは検索キーです。

例えば②領域のAppLogとAppStdOutはカテゴリであり、その下のoidなどのタグは、パーシング(parsing)された検索キーです。検索キーはログ>ログ設定のログパーサの設定タブでパーシングロジックを登録し、設定できます。

フィルター入力文法

タグは検索キーと検索値で構成されています。次の例で、検索キーは `exception`、検索値は `UnknownHostException` となります。この例は、収集したログデータのうち、IPアドレスとドメインアドレスが一致せず、サーバーをホストに接続できない場合に発生する例外(`UnknownHostException`)が含まれたログデータを照会します。



exception == UnknownHostException X

検索キーの種類

検索キーの種類	検索キーフォーマット	意味	検索キーと検索値の例	検索例
文字列キーワード	keyword	ファイル名	- キー: fileName - 値: /data/whatap/logs/yard.log	fileName:/data/whatap/logs/yard.log
数字キーワード	keyword.n	レスポンス時間	- キー: response_time.n - 値: 2945	response_time.n>=2945
予約語キーワード (辞書定義キーワード)	@keyword	TXID	- キー: @txid - 値: 85459614215434144	-

共通文法

文法の種類	説明	例
==searchValue	検索値と一致するログを検索します。	exception==RuntimeExceptionexception
!=searchValue	検索値を除いたログを検索します。	exception!=RuntimeException
*searchValue	検索値で終わるログを検索します。	word==*hello
searchValue*	検索値で始まるログを検索します。	word==hello*
searchValue	検索値として中間に含まれるログを検索します。	word==*hello*
*search*Value*	検索値として含まれるログを検索します。	word==*he*llo*

文法の種類	説明	例
re:{正規表現}	正規表現にマッチングされるログを検索します。	caller==re:^i\.w\.a\.w\.s\.v\.r\.
**	検索語に該当するすべてのログを検索します。	

検索キーが数字キーワード(keyword.n)の場合の文法

次の文法は検索キーが、 keyword.n 形式の場合のみに対応します。

- 検索値は数字のみ使えます。
- .n キーワードの値にはPrefixを付けません。 .n 以外のキーワードは、すべてPrefixを付ける必要があります。

例) +>searchValue は有効ではありません。

文法の種類	説明	例
>searchValue	検索値よりも大きな値を含むログを照会します。	response_time.n>3000
>=searchValue	検索値よりも大きいか、同じ値を含むログを照会します。	response_time.n>=3000
==searchValue	検索値と同じ値を含むログを照会します。	response_time.n==3000
!=searchValue	検索値とは異なる値を含むログを照会します。	response_time.n!=3000
<searchValue	検索値より小さい値を含むログを照会します。	response_time.n<3000
<=searchValue	検索値より小さいか、同じ値を含むログを照会します。	response_time.n<=3000

個人情報の照会

ログの個人情報照会権限がある場合**個人情報の表示**  トグルボタンの活性化を通じて非識別化処理した個人情報を表示することができます。

- ❗ ・ ログの個人情報照会権限についての詳細は[次の文書](#)を参照してください。

- ・ ログ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。

ログのタグオプション

ログのタグを選択すると、ドロップダウンメニューが表示されます。検索、除外検索、隣接ログ、トランザクション情報オプションが確認できます。

≡

全体90,699件の表示

過去順

最近順

キーワードを入力してください。

🔍

🔗

📄

🏠

⚙️

▶	oname	タイムスタンプ	ログ	
▶	java-demo3	2025-04-17 09:17:13.018	@txid 7759974649447934516 pcode 8 onodeName node-0 oid -278185929 okind 1152715164 oname java-demo3	eException: random exception
▶	java-demo3	2025-04-17 09:17:13.018	demo3 onodeName node-0 agentime 1744849031927 oid -278185929 category AppStdErr okindName dev-okind-0	rtual.web.Simula.exec(Simula.java:79)
▶	java-demo3	2025-04-17 09:17:13.018	demo3 onodeName node-0 agentime 1744849031927 oid -278185929 category AppStdErr okindName dev-okind-0	ion: java.lang.RuntimeException: random exception
▶	java-demo3	2025-04-17 09:17:13.018	demo3 onodeName node-0 agentime 1744849031928 oid -278185929 category AppStdErr okindName dev-okind-0	rtual.web.VExec.doFilter(VExec.java:16)

コピー

検索

除外検索

隣接ログ

トランザクション...

ログタグ オプション メニュー	説明
コピー	該当するタグのログをコピーします。
検索	フィルターには、該当のタグを含む ('==') 条件で入力されます。
除外検索	フィルターには、該当のタグを除外する ('!=') 条件で入力されます。
隣接ログ	隣接ログの詳細画面では、選択したログのサーバーを対象に、選択したログと隣接する時間帯のログを照会できます。時間を選択して、周辺の時間帯に発生したログを確認することも可能です。基準となるログは青い背景で表示されます。
トランザクション 情報	ログに @txid や @mtid タグが含まれている場合は、トランザクション情報をクリックすると、トランザクション情報の詳細画面で内容を確認できます。

コンテンツハイライト

ログのコンテンツ内で目的のキーワードを簡単に識別できるよう、ハイライト機能を提供します。単一または複数のキーワードでフィルタリングできます。

1. ハイライトしたいキーワードをキーワード入力欄に入力します。
 - 例: `select`
2. 検索したキーワードがログリストのContent内でハイライト表示されます。
 -  **拡大表示** アイコンをクリックすると、**ログ** と **タイムスタンプ** を全画面で確認できます。
 - ハイライトされたキーワード間の移動はショートカットで操作できます。
 - `Enter` : 次の行へ移動
 - `Shift+Enter` : 前の行へ移動
 - `↑ / ↓` アイコン: 行の移動
 - `Cmd + F` (Mac) / `Ctrl + F` (Windows): キーワード検索

複数キーワード条件

複数キーワードでハイライトする場合は、次のように作成します。

入力文字列	説明	結果
<code>a b c</code>	各キーワードをスペースで区切ります。	a, b, c
<code>"Whatap is good."</code>	キーワードにスペースを含める場合は、 <code>"</code> または <code>"</code> で囲みます。	Whatap is good.
<code>"Whatap\\ is good."</code>	<code>"</code> で囲まれたキーワードに、 <code>\</code> を含める場合は、 <code>\\</code> と入力する必要があります。	Whatap\ is good.

ハイライトカラー設定

ハイライトするキーワードおよび色を設定できます。

- デフォルトでは、ログレベルに応じたハイライト（**FATAL**、**ERROR**、**WARN**、**SEVERE**、**CRITICAL**）が適用されています。
- 設定した内容はプロジェクト単位で保存されます。
 -  アイコンをクリックします。
 - ハイライト ウィンドウで、色を設定するキーワードを入力欄に追加してください。
 - 入力欄の左側の色をクリックすると、色を選択できます。

テーブル設定

カラム設定

3 領域の右で  **カラム設定** ボタンを選択します。カラムを追加したり、順序を設定できます。

- カラム追加

タグを選択してテーブルにカラムを追加できます。logカラム選択を解除する場合**ログ表示詳細設定**が確認できません。必ず1つ以上のカラムを選択します。

- カラム順序設定

カラムを追加すると**表示項目**に該当カラムが追加されます。希望のカラムをドラッグしてカラムの順序を変更します。

ログ表示詳細設定

3 領域右で  **ログ表示詳細設定** ボタンを選択します。デフォルトで**content**、**タグ**すべてチェックされており、両方の項目を表示します。**content**と**タグ**どちらかは必ず選択します。

チェックが解除されたアイテムはテーブルに表示されません。次のように**タグ**を解除する場合、テーブルでログの**タグ**は表示されません。

ログテーブルコンテンツタグ非表示

タグ管理一覧にタグを追加すると、追加した順にログのタグがソートされます。タグの順序はドラッグして変更できます。追加したタグを無効にすると、無効にしたタグはログのタグに表示されません。

ⓘ カラムおよびログ表示設定の適用範囲

- カラム設定とログ表示詳細設定は、ログTail、ログ検索、ログトレンドで使用できます。
- 同一プロジェクト内のログTail、ログ検索、ログトレンドは、カラム設定とログ表示詳細設定を共有します。

ログファイルのダウンロード

検索したログの結果を CSV または TXT ファイル形式でダウンロードできます。

1. 画面中央右側の  ダウンロード ボタンをクリックします。
2. CSVダウンロード または Logダウンロード をクリックして、希望の形式のファイルをダウンロードします。

ログの設定

ログ設定 の上部タブからログモニタリングを設定できます。エージェント設定の確認、ログモニタリングの有効化設定、ログデータの保持期間および照会パスワードの設定、ログパーサーの登録、クイックインデックスの設定、個人情報のマスキング管理などを行うことができます。

ⓘ 権限

- ログモニタリング機能の活性化 を使用するには、プロジェクト修正 権限が必要です。
- ログエディター 権限があれば、ログモニタリング機能の活性化 のほか、ログ設定 メニューを編集できます。

ログモニタリングを開始する

1. ログ > ログ設定 の ログモニタリングの開始 タブをクリックしてください。
2. ⓘ ガイド表示アイコンと 料金プラン表示ボタンを選択すると、案内画面に移動します。

エージェント設定とログモニタリングの有効化

ログモニタリングのデータ設定

② 領域でログ使用量が確認できます。またログ保存期間とログ照会パスワード設定を変更できます。

ログ参照パスワード

セキュリティを強化するため、ログ照会パスワードを設定してください。ログ検索パスワードの指定はオプションです。ログ検索パスワードを使用中の場合は、ログ画面に入るときは必ずパスワードを入力します。

ⓘ パスワードを忘れた場合

ログ編集権限を持つユーザーは、ログ設定 で新しいパスワードに変更できます。

ログ保存期間

共通に適用する基本(default) データ保持期間です。

- 最小1日から最大40日まで入力できます。未指定の場合、既定値は1日です。ログ保存期間の選択オプションの他に、ユーザーが希望する期間を入力できます。
- **ログ使用量**一覧で別に設定しない場合、このログデータの保存期間がデフォルトで適用されます。**ログ使用量** リストでカテゴリごとのデータ保持期間を設定し、**初期化** ボタンをクリックすると、デフォルトのデータ保持期間に初期化されます。

ログ使用量

ログ使用量一覧でカテゴリ別のログデータの保存期間を指定できます。ログ数は、当該期間中に蓄積されたログの件数を意味します。例えば、**日間ログ数**は、1日にたまったログ件数、**予想ログ数**は、データ保持日に今日のログ数を掛けたログ件数を意味します。

ログデータの保存期間を次のように指定できます。期間指定により、古いデータを削除してスペースを確保できます。

- **トライアルプロジェクト**

データ保存期間として1日、2日、3日を選択できます。

- **有料プロジェクト**

データの保存期間として1日、2日、3日、4日、5日、6日、7日、10日、30日、40日を選択できます。

- **保存量基準課金**

データの保存期間によってコストが異なります。

例、1日平均200万ログ件数が蓄積され、データ保存期間を3日に指定した場合、平均600万ログ件数が収集サーバーに保持され、課金の対象となります。

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、メッセージャー、アプリのプッシュ通知など、希望するチャン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。

イベント設定概要

Javaアラートのためのイベント設定メニューの構成と使い方を案内します。

通知設定

Javaアラートイベントを受信するチャンネルと対象を設定する方法を案内します。

イベント履歴

Javaで発生したイベント履歴を照会し推移をモニタリングする方法を案内します。

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、メッセージャー、アプリプッシュなど、希望するチャン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。

📄 イベント設定概要

Javaアラートのためのイベント設定メニューの構成と使い方を案内します。

📄 通知設定

Javaアラートイベントを受信するチャネルと対象を設定する方法を案内します。

📄 イベント履歴

Javaで発生したイベント履歴を照会し推移をモニタリングする方法を案内します。

📄 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
└ HITMAP	ヒットマップパターン
└ LOG_REALTIME	リアルタイムログイベント
└ METRICS	メトリクス
└ COMPOSITE_LOG	複合ログイベント
└ BASIC	基本イベント
└ COMPOSITE_METRICS	複合メトリクス

!	JSON フィールド	説明
	L 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。

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。

- 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

- Telegramで @WhaTapApmNotiBot を検索してください。

- ボットを選択し、[開始] ボタンをクリックします。
- 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。



① 使用可能なコマンド一覧

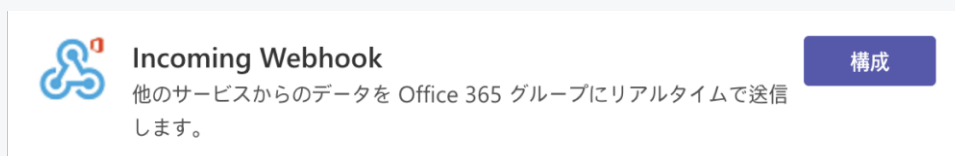
- `/help` : コマンド一覧を表示
- `/license` : 通知を受信するプロジェクトのアクセスキーを入力
- `/remove` : 通知を受信しないプロジェクトコードを入力
- `/list` : 現在通知を受信しているプロジェクトコードを表示

Teams

- 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。



Incoming Webhook

フィードバックの送信

Incoming Webhook コネクタを利用すると、トラッキング対象の活動についての通知を外部サービスから受け取ることができます。このコネクタを使うには、別サービスで設定を作成する必要があります。そのサービスが、Office 365 コネクタ形式 対応の Web フックをサポートしている必要があります。

* が付いているフィールドは必須です

着信 Web フック (Incoming Webhook) をセットアップするには、名前を入力して [作成] を選んでください。*

この着信 Web フック (Incoming Webhook) からのデータに関連付ける画像をカスタマイズします。

イメージをアップロードします。



既定のイメージ

作成

キャンセル

4. 生成された Webhook URL をコピーします。

以下の URL をコピーしてクリップボードに保存してから、[保存] を選んでください。この URL は、グループへのデータ送信元となるサービスの画面で必要になります。

`https://whatap.webhook.office.com/webf`



URL は最新です。

完了

削除

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。

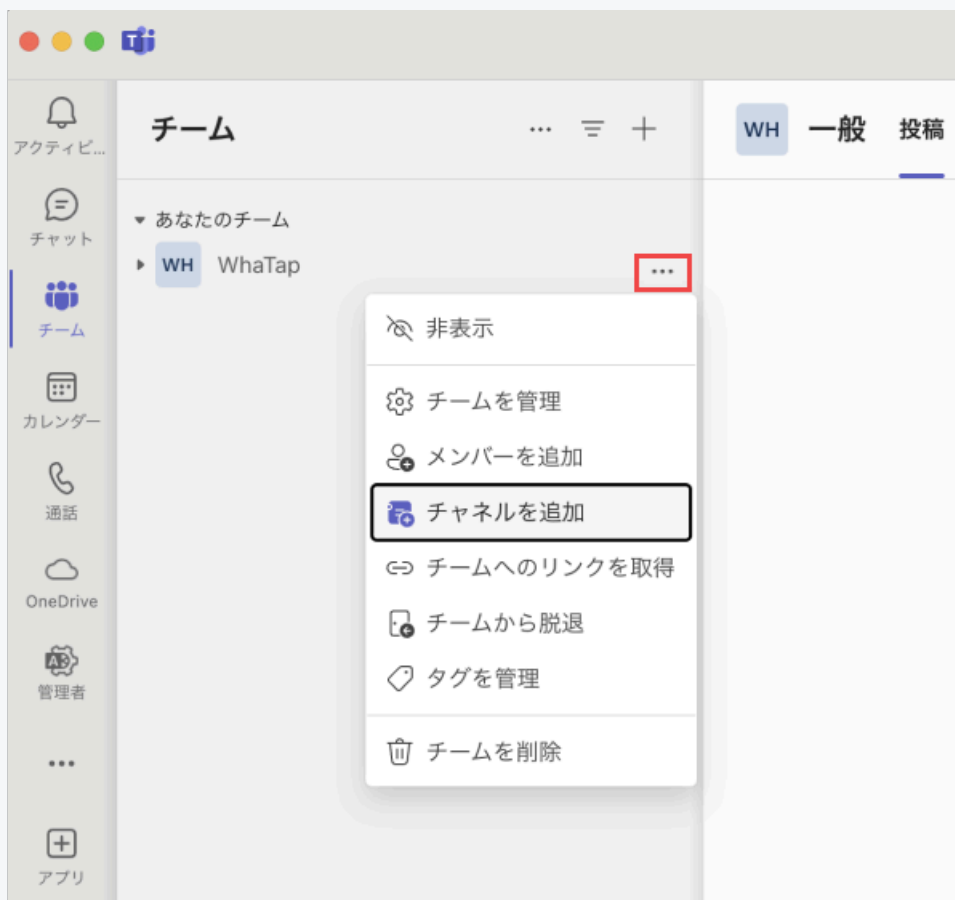
6. [登録] ボタンをクリックします。



Teams Workflows

- ① 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 (...) > チャンネルを追加 をクリックします。



2. チャンネルの作成 画面で チャンネル名 と 説明 を入力します。チャンネルの種類 は 標準 を選択し、[作成] ボタンをクリックします。



チャンネルの作成

チャンネル名 *

WhaTapのイベントチャンネル

説明

他のユーザーが、適切なチャンネルを見つけられるように説明を入力します

チャンネルの種類を選択する * ⓘ

選択

☒ 標準
チームの全員がアクセスできます。

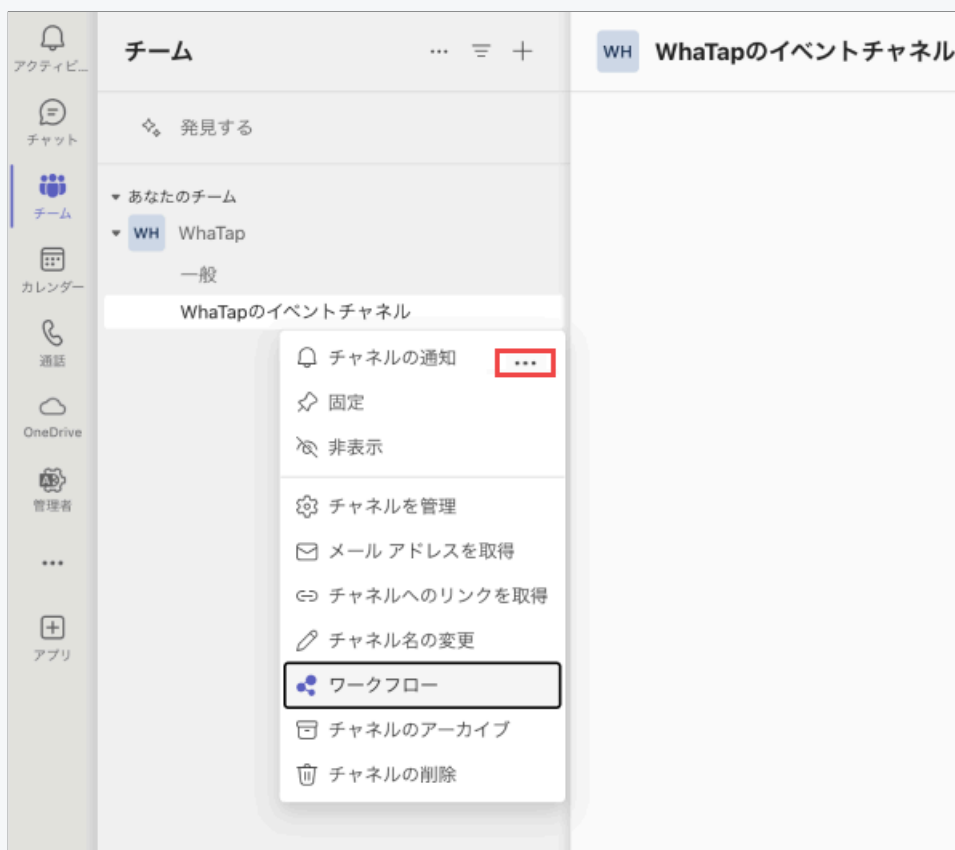
☐ 共有
組織内または組織外の人またはチームがアクセスできます。

☐ プライベート
チームの特定のユーザーがアクセスできます。

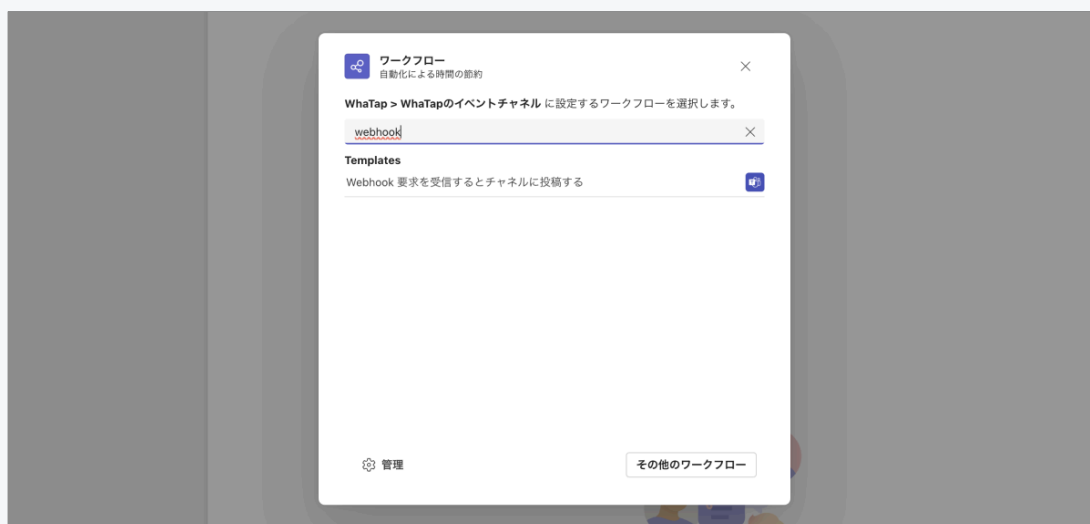
キャンセル 作成

❗ チャンネルの種類を **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。

- 作成したチャンネルの **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ンネルに投稿 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。



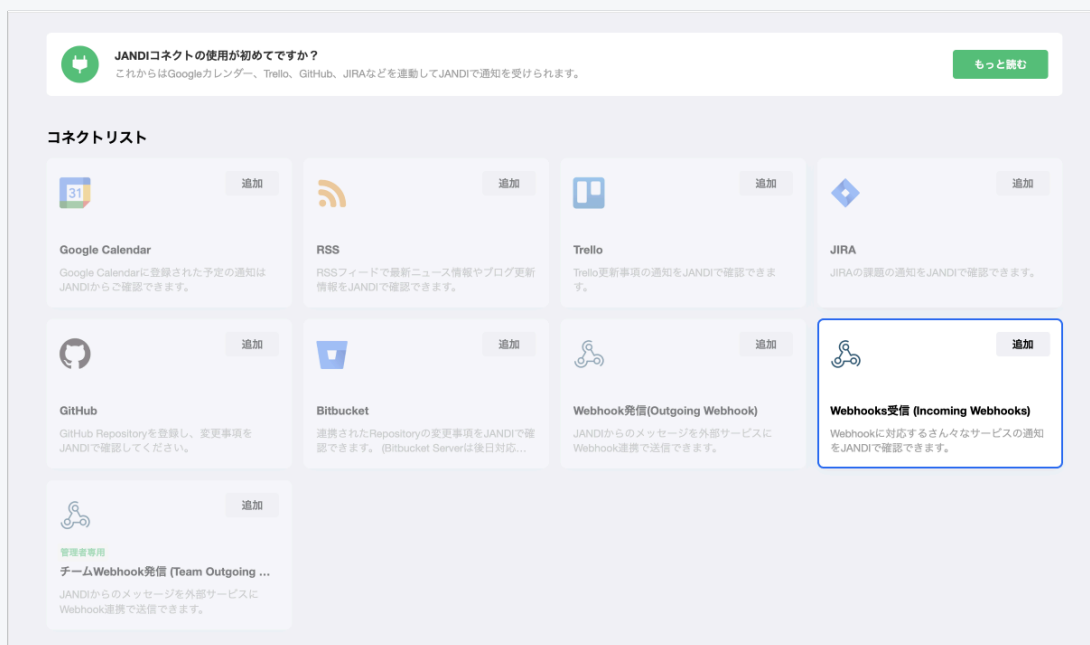
8. WhaTap モニタリングサービスの **サードパーティプラグインの追加** 画面に戻り、**Teams Workflows** タブ下部の **Channel Name** と **Webhook URL** に値を入力します。
9. [登録] ボタンをクリックします。

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。



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。

← インテグレーション 全体

統合名 * Whatap_Demo

統合タイプ * Whatap

命名規則の設定 *

インシデント命名規則を定義および管理します。

命名規則

[[level]][title] (projectName) (oname) (oid)

ルール設定のプレビュー インシデント名の長さは 1600 バイトに制限されます。

サンプルデータ

```
{
  "metricName": "cpu",
  "pcode": 188,
  "level": "Warning",
  "metricValue": "25.808296",
  "oid": "330494549",
  "title": "CPU Used > 10 %",
  "message": "MEMORY is too high 100%",
  "url": "http://localhost:45158/monitor/330494549",
  "metricThreshold": "10.0%",
  "oname": "ote-front",
  "time": "15/07/2017 17:17",
  "projectName": "OTE-SERVER",
  "status": "on"
}
```

56/1600 バイト

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。



インテグレーション

インテグレーションの作成 検索語を入力 最後のアップデート: 2024-02-21 11:25:10

統合名	統合タイプ	統合キー	サービス名
ワットテスト	ワタップ		ワットテスト

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。

← インテグレーション

ワットテスト

基本情報

統合タイプ	ワタップ
統合キー	fda527a5140
URL	https://alertnowitgr.opsnow.com/integration/whatap/v1/
コンストラクタ	リサ

クリップボードにコピーする

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}"
}
```

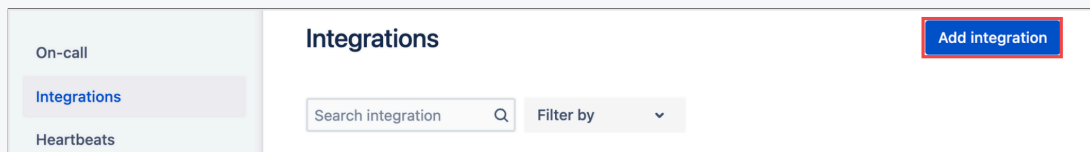
2. **Webhook Name** と **Webhook URL** の項目を入力してください。
3. ヘッダー値を追加するには、**ユーザーヘッダー値の追加** の + 追加 をクリックし、key と value を入力してください。
4. [登録] ボタンをクリックしてください。

Opsgenie

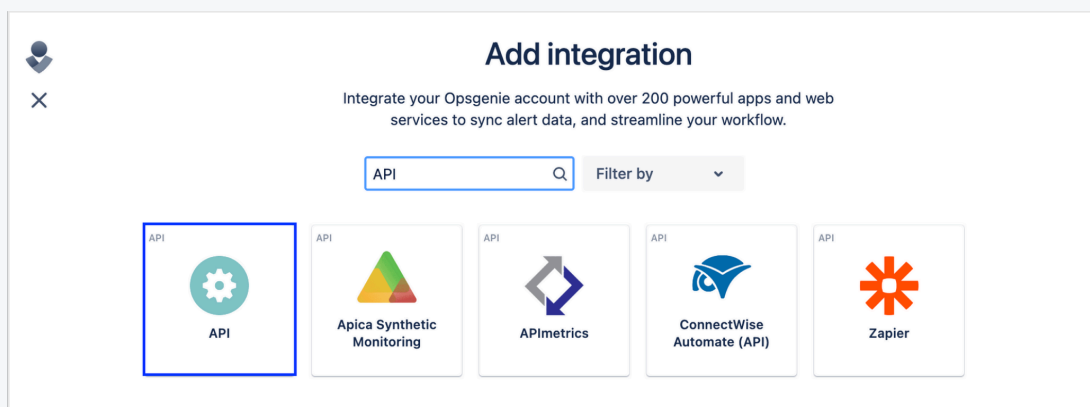




1. **Teams** メニューで、アラートを受信するチームを選択してください。
2. **Integration** メニューに移動し、[Add Integration] ボタンをクリックしてください。



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。



Forwarding rules
Your on-call schedule

ON-CALL
Global policies
All schedules
All escalations

CONNECTED APPS
Atlassian products

INTEGRATIONS
Integrations
Heartbeats
Slack
GitLab

Settings

Name:

Assigned to Team:

API Key:

Read Access: ☒

Create and Update Access: ☒

Delete Access: ☒

Restrict Configuration Access: ☐

Enabled: ☒

Suppress Notifications: ☐

[Save Integration](#) [Cancel](#)

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。
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。
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、他の通知チャンネルの言語は変更されません。

ilert



- 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。
- 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。
- 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。
- 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。
- 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。
- 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
 -  C: Boolean
 -  D: 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

① 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
```