

# Zabbix 4.4 解説

Zabbix Conference Japan 2019  
2019/11/15

SRA OSS, Inc. 日本支社  
赤松 俊弘



## 自己紹介

- ☑ 赤松 俊弘 (Toshihiro Akamatsu)
- ☑ SRA OSS, Inc. 日本支社  
OSS 事業本部 基盤技術グループ

## 職務

- ☑ PostgreSQL 以外の OSS 全般の技術サポート、構築
- ☑ 主に Zabbix を担当



# Zabbix 4.4 概要

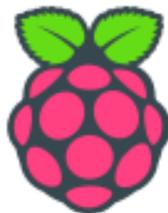
- ☑ 2019/10/07 リリース
- ☑ ポイントリリース
  - ☑ フルサポート6カ月
  - ☑ リミテッドサポート1ヶ月
  - ☑ 延長サポートなし



## OS



Debian 10



Raspbian 10



Mac OS/X



SUSE Linux  
Enterprise  
Server 15



Red Hat  
Enterprise  
Linux 8



MSI for  
Windows  
agent

パッケージ提供バージョン

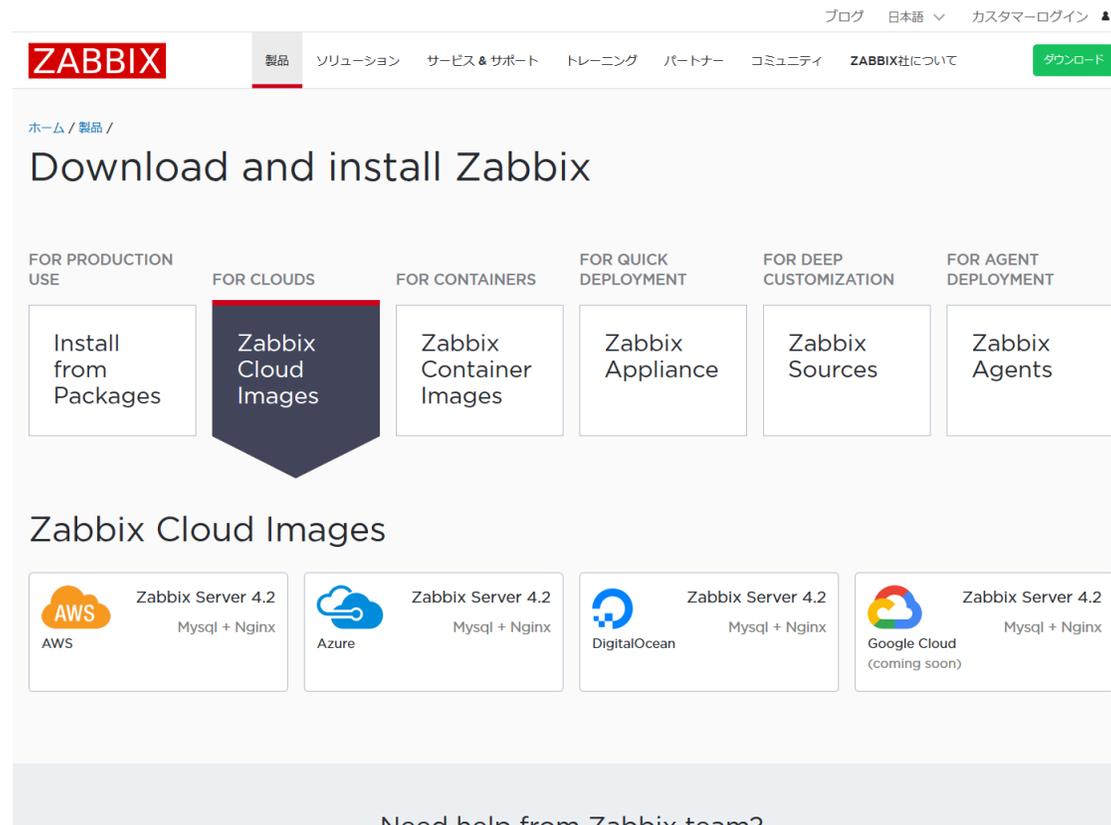
3.0 LTS

4.0 LTS

4.2

4.4

## クラウドイメージ



The screenshot shows the Zabbix website's 'Download and install Zabbix' page. The navigation bar includes 'ZABBIX', '製品', 'ソリューション', 'サービス & サポート', 'トレーニング', 'パートナー', 'コミュニティ', and 'ZABBIX社について'. A 'ダウンロード' button is visible. The main content area is titled 'Download and install Zabbix' and features six categories: 'FOR PRODUCTION USE', 'FOR CLOUDS', 'FOR CONTAINERS', 'FOR QUICK DEPLOYMENT', 'FOR DEEP CUSTOMIZATION', and 'FOR AGENT DEPLOYMENT'. The 'FOR CLOUDS' category is highlighted with a dark blue arrow pointing to a 'Zabbix Cloud Images' box. Below this, there are four boxes for 'Zabbix Server 4.2' with 'Mysql + Nginx' support, each associated with a different cloud provider: AWS, Azure, DigitalOcean, and Google Cloud (coming soon). At the bottom, there is a link that says 'Need help from Zabbix team?'.

# ZABBIX

## Zabbix Code Guidelines

3.0 4.0 4.4 (current) | In development: 5.0 (devel) | Unsupported: 1.8 2.0 2.2 2.4 3.2 3.4 4.2 **Guidelines**

 **Zabbix guidelines**

- Coding
- Naming
- Development process
- Templates

### Zabbix guidelines

These pages contain official Zabbix guidelines with regard to:

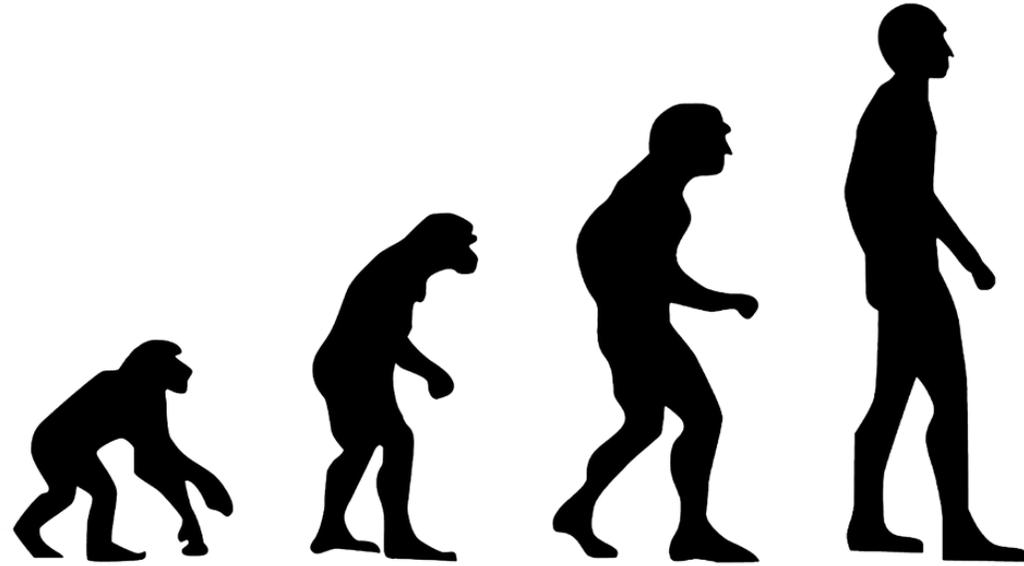
- Coding
  - C coding guidelines
  - HTML/CSS coding guidelines
  - JavaScript coding guidelines
  - Perl coding guidelines
  - PHP coding guidelines
  - Shell coding guidelines
- Naming
  - Error message style
- Development process
- Templates

Except where otherwise noted, content on this wiki is licensed under the following license:  CC Attribution-NonCommercial

© 2001-2019 by Zabbix SIA. All rights reserved.



- ☑ 次世代の Zabbix Agent
- ☑ Webhook によるアラート
- ☑ TimescaleDB の公式サポート
- ☑ グラフでの集約関数
- ☑ LLD アイテムの追加
- ☑ 保存前処理の追加
- ☑ Frontend の改善
- ☑ etc...



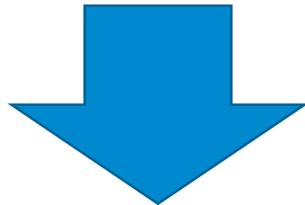
# 次世代の Zabbix Agent

その名も

# Zabbix Agent 2



詳細は…



**Magic of the new Zabbix Agent**

**Alexey Pustovalov**

**15:30 - 16:10**



- ✓ Go 言語で開発された新しい Zabbix Agent
- ✓ 現状は実験的なサポート段階

## サポートプラットフォーム



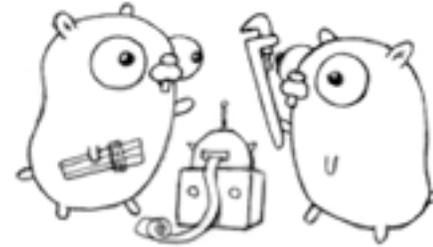
以下はパッケージあり

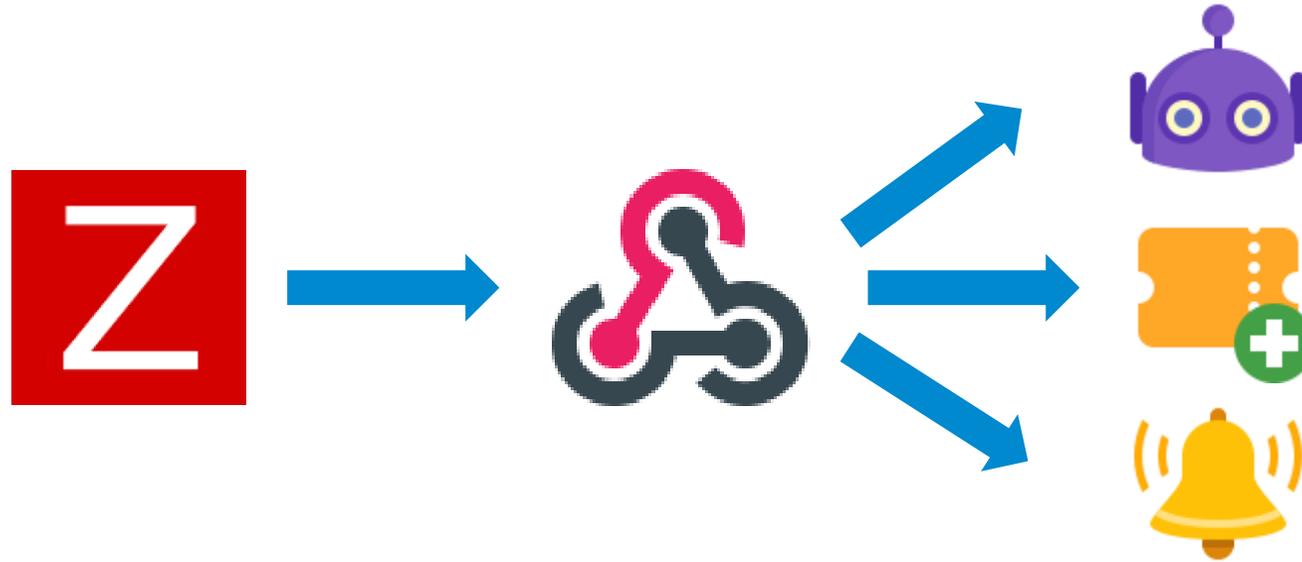
- ✓ RHEL/CentOS 8
- ✓ SLES 15 SP1+
- ✓ Debian 9, 10
- ✓ Ubuntu 18.04

それ以外はソースからコンパイルが必要

## 機能・特徴

- ✓ 監視の高い並列実行性
- ✓ プラグインによる拡張容易性
- ✓ TCP 接続数の削減
- ✓ 現行の Zabbix Agent の機能は全て踏襲
- ✓ アクティブチェックの監視間隔カスタマイズが可能
- ✓ 現行の設定ファイルとほぼ互換性あり





# Webhook によるアラート

メディアタイプ

メディアタイプ オプション

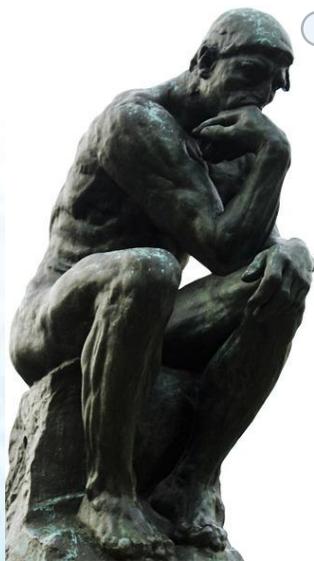
\* 名前

タイプ **Webhook** ▼

パラメータ	名前	値	アクション
	URL	<input type="text"/>	削除
	To	{ALERT.SENDTO}	削除
	Subject	{ALERT.SUBJECT}	削除
	Message	{ALERT.MESSAGE}	削除

- ☑ メディアタイプに **Webhook** が追加
- ☑ 外部システムとの連携がより容易に

それってカスタムスクリプトと  
どう違うの？



## Frontend から JavaScript

### JavaScript

```
1 try {
2   Zabbix.Log(4, 'Redmine webhook script value='+value);
3
4   var result = {
5     'tags': {
6       'endpoint': 'Redmine'
7     }
8   },
9   params = JSON.parse(value),
10  req = new CurlHttpRequest(),
11  issue = {},
12  resp;
13
14  req.AddHeader('Content-Type: application/json');
15  req.AddHeader('X-Redmine-API-Key: '+params.api_key);
```

## レスポンスの値を利用可能

Process tags

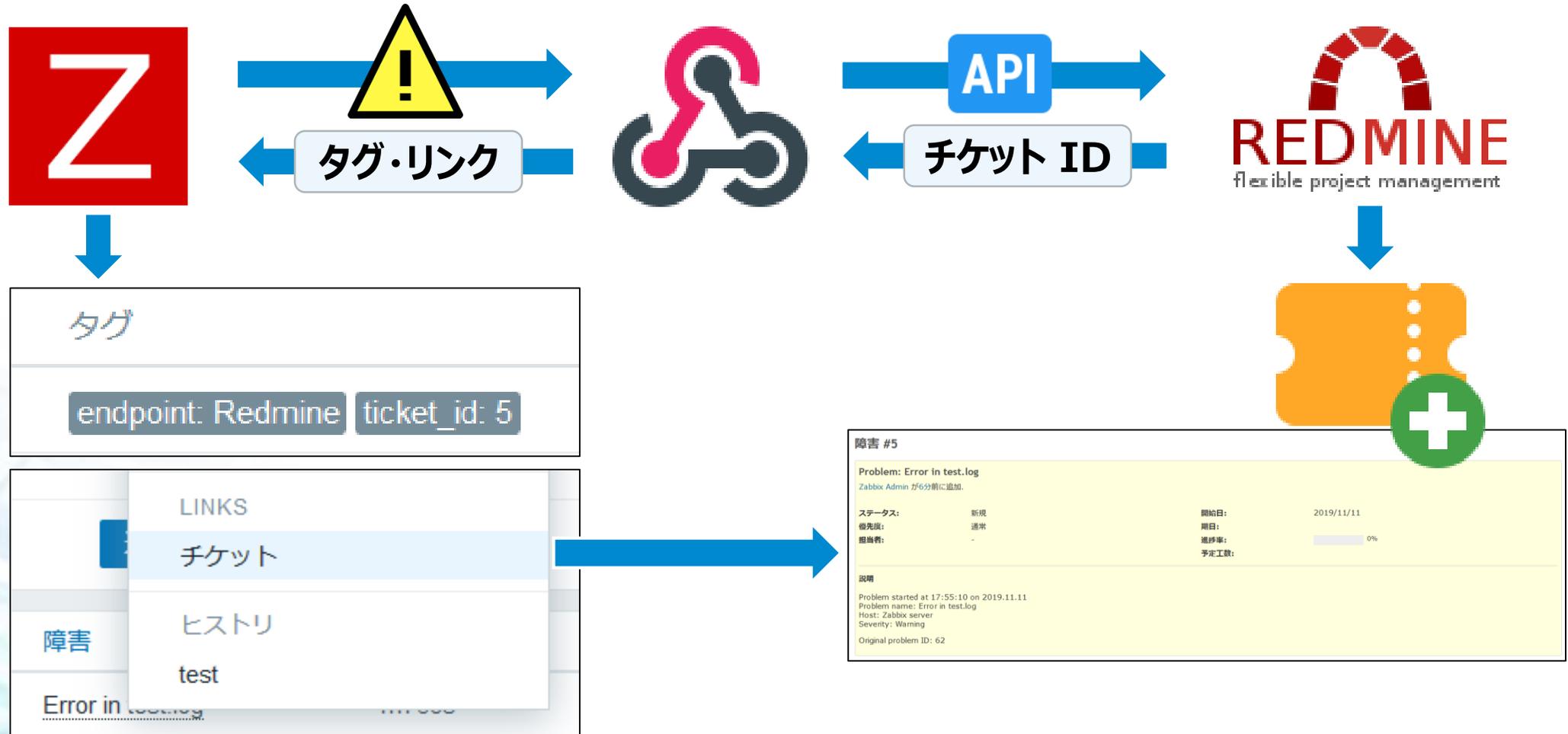
Include event menu entry

\* Menu entry name

\* Menu entry URL

- ☑ Process tags  
⇒ 特定の値を動的にイベントのタグに追加
- ☑ Include event menu entry  
⇒ イベントのメニューに動的に任意のリンクを追加

## 例：Redmine にチケットを登録



\* 名前 Redmine create ticket

タイプ Webhook

パラメータ	名前	値	アクション
	tracker_id	4	削除
	project_id	1	削除
	api_key	376d1ad161d2710605ceb78e3e9324	削除
	description	{ALERT.MESSAGE}	削除
	subject	{ALERT.SUBJECT}	削除

[追加](#)

\* スクリプト `try {`

タイムアウト 30s

Process tags

Include event menu entry

\* Menu entry name チケット

\* Menu entry URL `http://dhcp-175-108/redmine/issues/{EVENT.TAGS.ticket_id}`

説明 Creating a Redmine ticket

有効

## JavaScript

```

1 try {
2   Zabbix.Log(4, 'Redmine webhook script value='+value);
3   var result = {
4     'tags': {
5       'endpoint': 'Redmine'
6     }
7   },
8   params = JSON.parse(value),
9   req = new XMLHttpRequest(),
10  issue = {},
11  resp;
12
13  req.AddHeader('Content-Type: application/json');
14  req.AddHeader('X-Redmine-API-Key: '+params.api_key);
15  issue.project_id = params.project_id;
16  issue.subject = params.subject;
17  issue.description = params.description;
18  issue.tracker_id = params.tracker_id;
19  resp = req.Post('http://dhcp-175-108/redmine/issues.json',
20    JSON.stringify({"issue": issue})
21  );
22
23  if (req.Status() != 201) {
24    throw 'Response code: '+req.Status();
25  }
26
27  resp = JSON.parse(resp);
28  result.tags.ticket_id = resp.issue.id;
29 } catch (error) {
30   Zabbix.Log(4, 'Redmine ticket creation failed json : '+JSON.stringify({"issue": issue}));
31   Zabbix.Log(4, 'Redmine ticket creation failed : '+error);
32   result = {};
33 }
34
35 return JSON.stringify(result);

```

64588文字入力可



# TimescaleDB の公式サポート



## 時系列データに強い PostgreSQL



### PostgreSQL の EXTENSION

OSS

PostgreSQL との互換性

### パフォーマンス向上

データ削除が  
速い

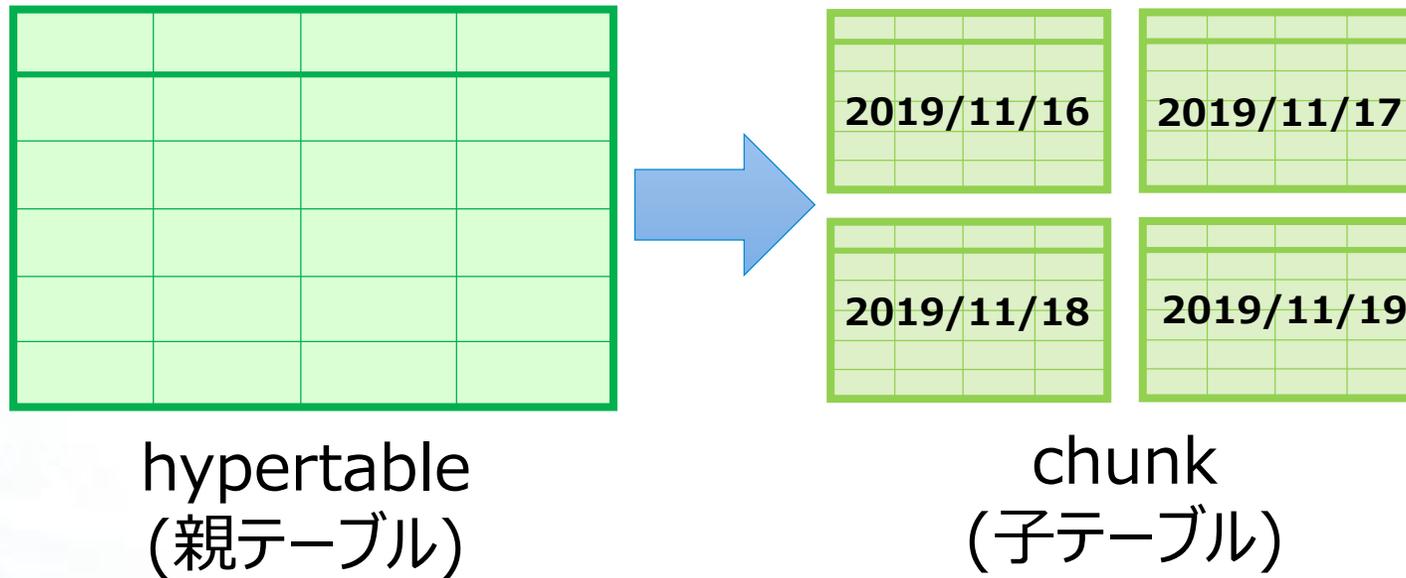
クエリも速い

### 時系列データ特化

時系列分析  
関数

時系列データ  
管理機能

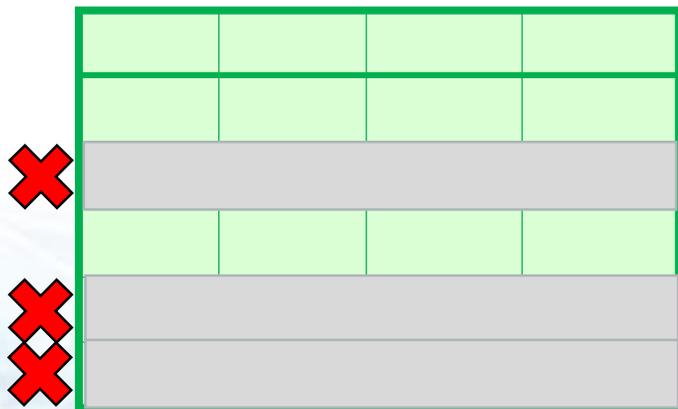
## テーブルを時間単位でパーティショニング



SELECT や INSERT などの親テーブルで実行した操作は、自動的に子テーブルに伝播

## PostgreSQL

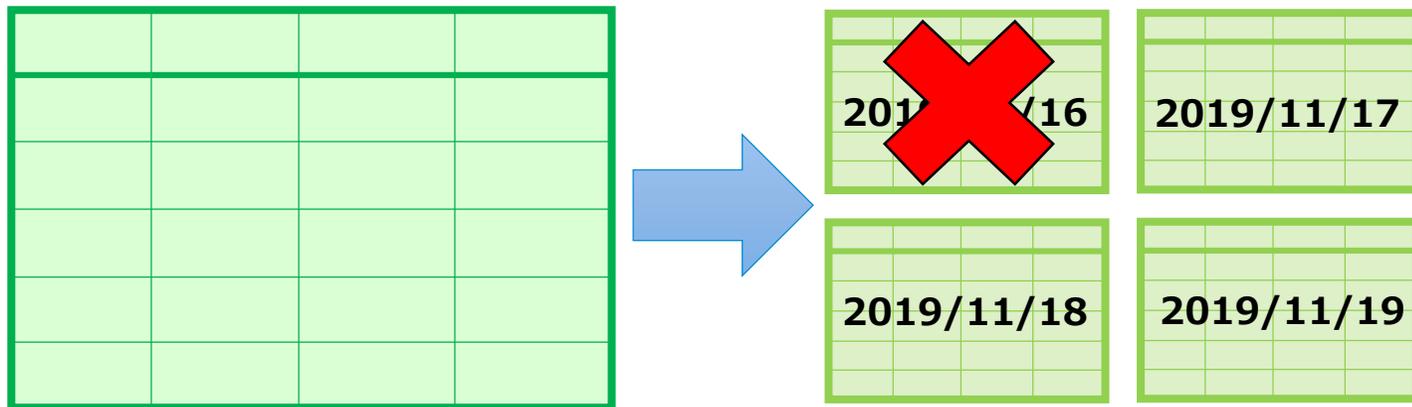
```
DELETE FROM history
WHERE
itemid = xxxx AND clock < yyyy
```



該当行を検索して削除  
不要領域回収に要 VACUUM

## TimescaleDB

```
SELECT
drop_chunks('history')
```



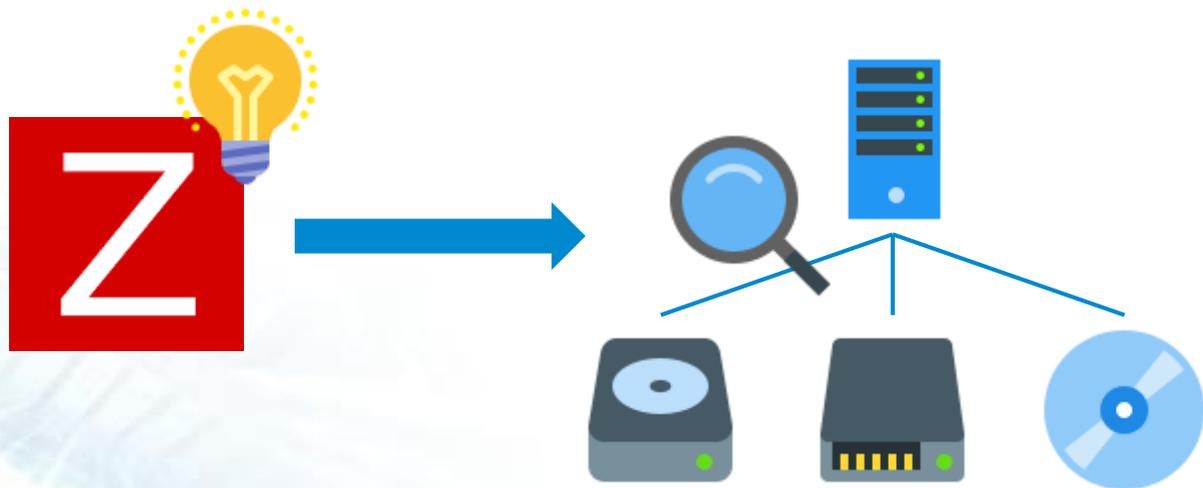
子テーブルごと削除  
VACUUM 不要



# LLD アイテムの追加

## ブロックデバイスのディスカバリ

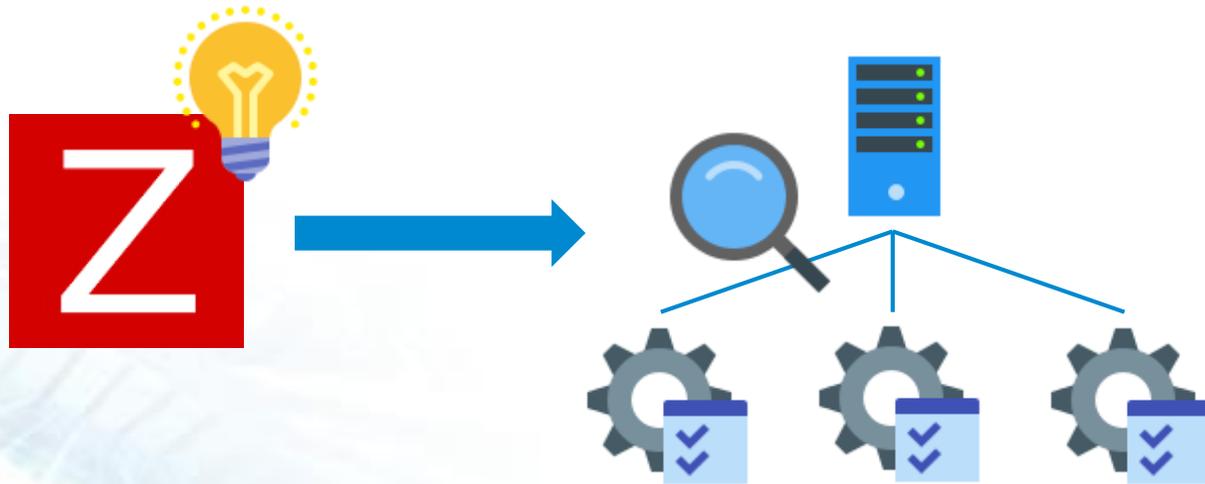
vfs.dev.discovery

 Linux のみでサポート

```
[  
  {  
    "{#DEVNAME}":"cdrom",  
    "{#DEVTYPE}":"disk"  
  },  
  {  
    "{#DEVNAME}":"fd0",  
    "{#DEVTYPE}":"disk"  
  },  
  {  
    "{#DEVNAME}":"sda",  
    "{#DEVTYPE}":"disk"  
  },  
  {  
    "{#DEVNAME}":"sda1",  
    "{#DEVTYPE}":"partition"  
  },  
  ...  
]
```

systemd サービスのディスカバリ

**systemd.unit.discovery**

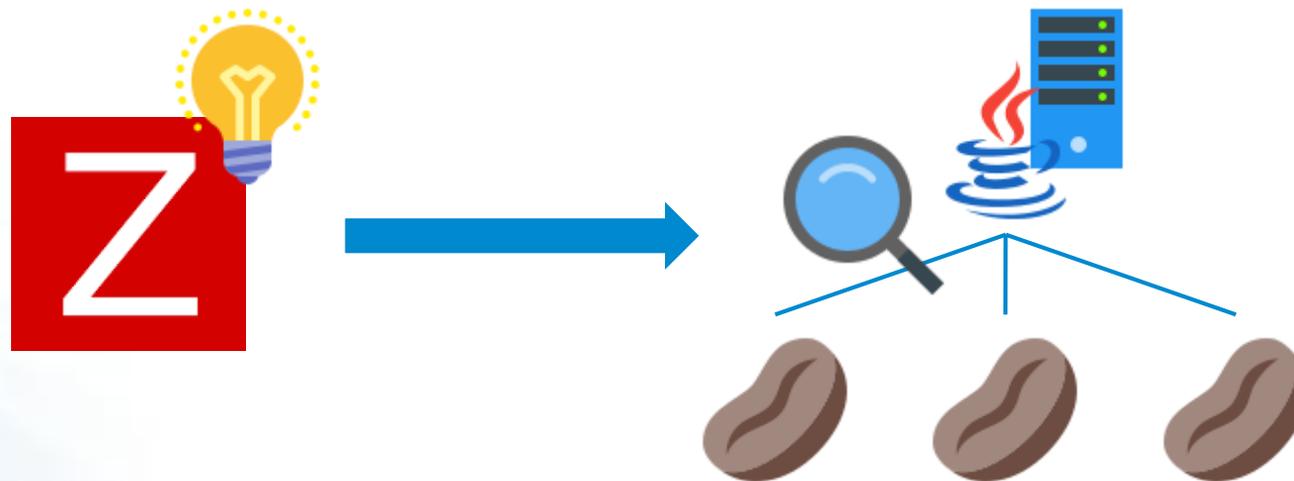


```
[
  {
    "{#UNIT.NAME}": "zabbix-server.service",
    "{#UNIT.DESCRPTION}": "Zabbix Server",
    "{#UNIT.LOADSTATE}": "loaded",
    "{#UNIT.ACTIVESTATE}": "active",
    "{#UNIT.SUBSTATE}": "running",
    "{#UNIT.FOLLOWED}": "",
    "{#UNIT.PATH}": "/org/freedesktop/...",
    "{#UNIT.JOBID}": 0,
    "{#UNIT.JOBTYPE}": "",
    "{#UNIT.JOBPATH}": "/"},
  },
  ...
]
```

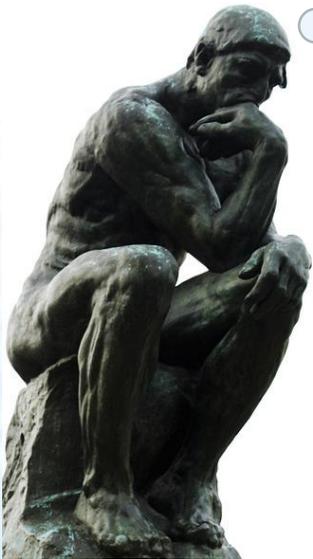
☑ Zabbix Agent 2 かつ Linux のみでサポート

## JMX MBean のディスカバリ

```
jmx.get[<discovery mode>, <object name>]
```



前からある  
jmx.discovery と  
どう違うの？



## jmx.discovery

ObjectName="java.lang:type=MemoryPool,name=Metaspace,attr-1=hoge"



```
[  
  {  
    "{#JMXOBJ}": "java.lang:type=MemoryPool,name=Metaspace,attr-1=hoge",  
    "{#JMXDOMAIN}": "java.lang",  
    "{#JMXTYPE}": "MemoryPool",  
    "{#JMXNAME}": "Metaspace"  
  }  
]
```

- ☑ MBean やその属性を **LLD マクロ付きの JSON** で返す
- ☑ マクロでサポートされない文字列が含まれる属性は無視
  - ☑ ハイフンや括弧、非アスキー文字など

## jmx.get

```
ObjectName="java.lang:type=MemoryPool,name=Metaspace,attr-1=hoge"
```

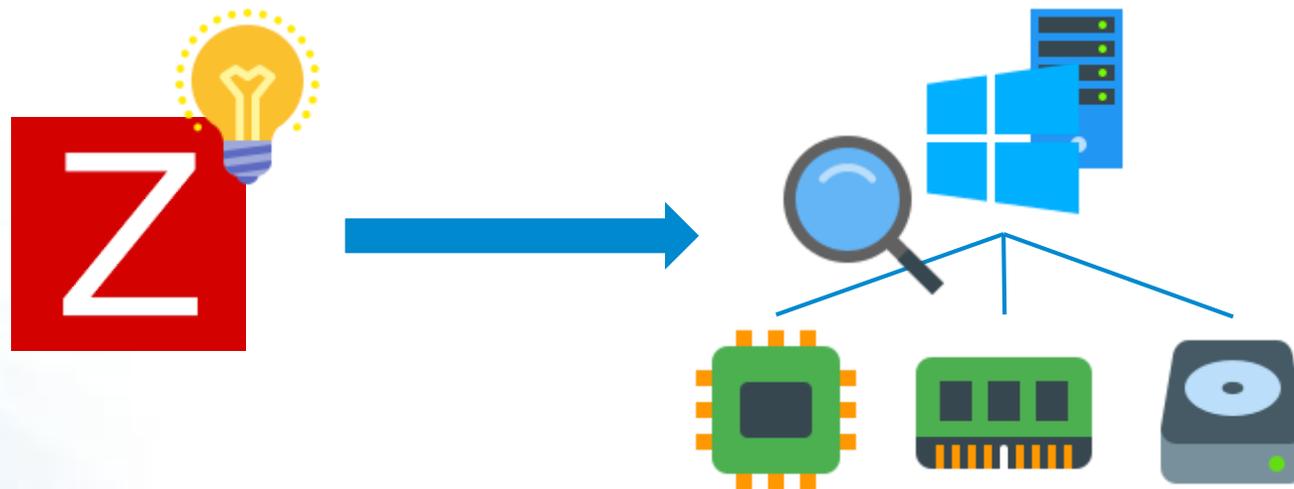


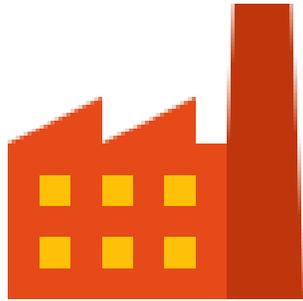
```
[  
  {  
    "object": "java.lang:type=MemoryPool,name=Metaspace,attr-1=hoge"  
    "domain": "java.lang",  
    "properties": {  
      "name": "Metaspace",  
      "type": "MemoryPool",  
      "attr-1": "hoge"}  
    }  
  ]
```

- ☑ MBean やその属性を **JSON** で返す
- ☑ キーに対するカスタム LLD マクロの定義が必要

## WMI のディスカバリ

```
wmi.getall[<namespace>, <query>]
```





# 保存前処理の新機能

- ☑ CSV to JSON
- ☑ JSON 内 / XML 内 / 正規表現使用時のエラーチェックでの失敗時のカスタマイズ
- ☑ LLD ルールで使用できる保存前処理の追加

保存前処理の設定

名前	パラメータ		
1: CSV to JSON	,	"	<input checked="" type="checkbox"/> With header row

[追加](#)



## CSV to JSON のパラメータ

保存前処理の設定

名前	パラメータ
1: CSV to JSON	, " 1

追加

With header row

- 1 最初の行をヘッダ行とする
- 2 デリミタの指定
- 3 指定した文字を文字ではなくクォート記号として扱う

# 1 最初の行をヘッダ行とする

```
ID,Host,IP  
1,host1,10.1.1.1  
2,host2,10.1.1.2  
...
```

 With header row

```
[  
  {  
    "ID": "1",  
    "Host": "host1",  
    "IP": "10.1.1.1"  
  },  
  {  
    "ID": "2",  
    "Host": "host2",  
    "IP": "10.1.1.2"  
  },  
  ...  
]
```

 With header row

```
[  
  {  
    "1": "ID",  
    "2": "Host",  
    "3": "IP"  
  },  
  {  
    "1": "1",  
    "2": "host1",  
    "3": "10.1.1.1"  
  },  
  ...  
]
```

## 2 デリミタの指定

```
ID,Host,IP
1,host1,10.1.1.1
2,host2,10.1.1.2
...
```

パラメータ

With header row

```
[
  {
    "ID":"1",
    "Host":"host1",
    "IP":"10.1.1.1"
  },
  {
    "ID":"2",
    "Host":"host2",
    "IP":"10.1.1.2"
  },
  ...
]
```

```
ID@Host@IP
1@host1@10.1.1.1
2@host2@10.1.1.2
...
```

パラメータ

With header row

☑ ちなみに CSV の頭に Sep=<デリミタ> でも OK

Sep=@

ID@Host@IP

1@host1@10.1.1.1

2@host2@10.1.1.2

...

パラメータ

,

"

With header row

```
[
  {
    "ID":"1",
    "Host":"host1",
    "IP":"10.1.1.1"
  },
  {
    "ID":"2",
    "Host":"host2",
    "IP":"10.1.1.2"
  },
  ...
]
```

## ☑ さらにパラメータと Sep= ではパラメータ優先

```
Sep=|  
ID@Host@IP  
1@host1@10.1.1.1  
2@host2@10.1.1.2  
...
```

パラメータ

@ "  With header row

```
[  
  {  
    "ID":"1",  
    "Host":"host1",  
    "IP":"10.1.1.1"  
  },  
  {  
    "ID":"2",  
    "Host":"host2",  
    "IP":"10.1.1.2"  
  },  
  ...  
]
```

3

## 指定した文字を文字ではなくクオート記号として扱う

```
"ID",Host,IP  
1,host1,10.1.1.1  
2,host2,10.1.1.2  
...
```

パラメータ

, "  With header row

パラメータ

, "  With header row

```
[  
  {  
    "¥"ID¥"" : "1",  
    "Host": "host1",  
    "IP": "10.1.1.1"  
  },  
  ...  
]
```

```
[  
  {  
    "ID": "1",  
    "Host": "host1",  
    "IP": "10.1.1.1"  
  },  
  ...  
]
```

- ☑ JSON 内 / XML 内 / 正規表現使用時のエラーチェックで失敗時のカスタマイズが使用可能

アイテム 保存前処理

保存前処理の設定

名前	パラメータ	失敗時のカスタマイズ	アクション
1: JSON内のエラーチェック	\$.error	<input checked="" type="checkbox"/>	<a href="#">テスト</a> <a href="#">削除</a>

失敗時のカスタマイズ

値を破棄	値を設定	エラーを設定	0
------	------	--------	---

[追加](#)

[追加](#) [キャンセル](#)

[Test all steps](#)

- ☑ LLD ルールで使用できる保存前処理に以下が追加
  - ☑ XML Xpath
  - ☑ CSV to JSON
  - ☑ XML 内でのエラーチェック





# セキュリティの新機能

- ☑ Web 監視および HTTP アイテムに Kerberos 認証サポートが追加
- ☑ 自動登録の通信に PSK ベースの暗号化サポートが追加

- ☑ Web 監視と HTTP エージェントの認証方法に Kerberos 認証が追加

シナリオ ステップ 認証

HTTP認証 Kerberos ▼

ユーザー

パスワード

SSLピア検証

SSLホスト検証

SSL証明書ファイル

SSL秘密鍵ファイル

SSL秘密鍵パスワード

追加 キャンセル

- エージェントの自動登録で PSK ベースの通信が可能に

一般設定 プロキシ 認証 ユーザーグループ ユーザー メディアタイプ スクリプト キュー

### 自動登録

Encryption level  暗号なし  
 PSK

\* PSKアイデンティティ

\* PSK

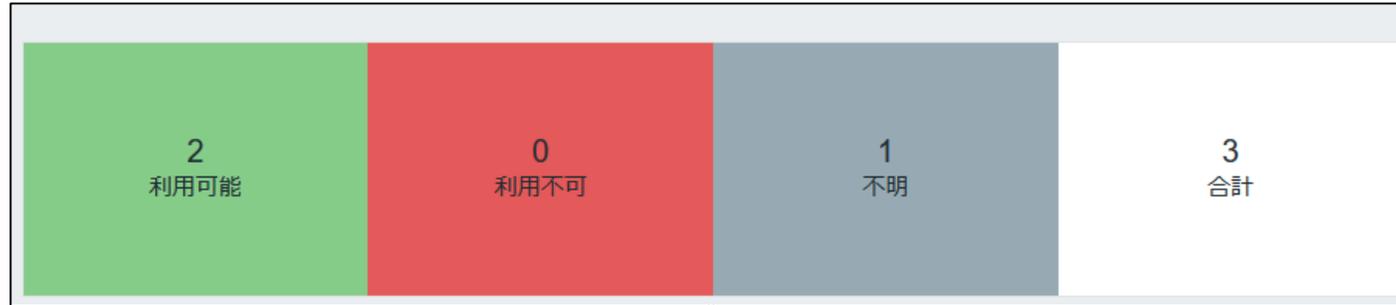


# Frontend の新機能

- ☑ Host availability の追加
- ☑ ウィジェットのヘッダ非表示機能
- ☑ bar グラフの追加
- ☑ グラフでの集約関数
- ☑ Operational data 表示
- ☑ アイテム/トリガーの説明表示

- ☑ Host Availability ウィジェットが追加
- ☑ 深刻度ごとの障害数の見た目も変更





ウィジェットの変更

タイプ Host availability

Show header



データセット 表示オプション 期間 軸 凡例 障害 オ

データセット ■ Zabbix server  選択

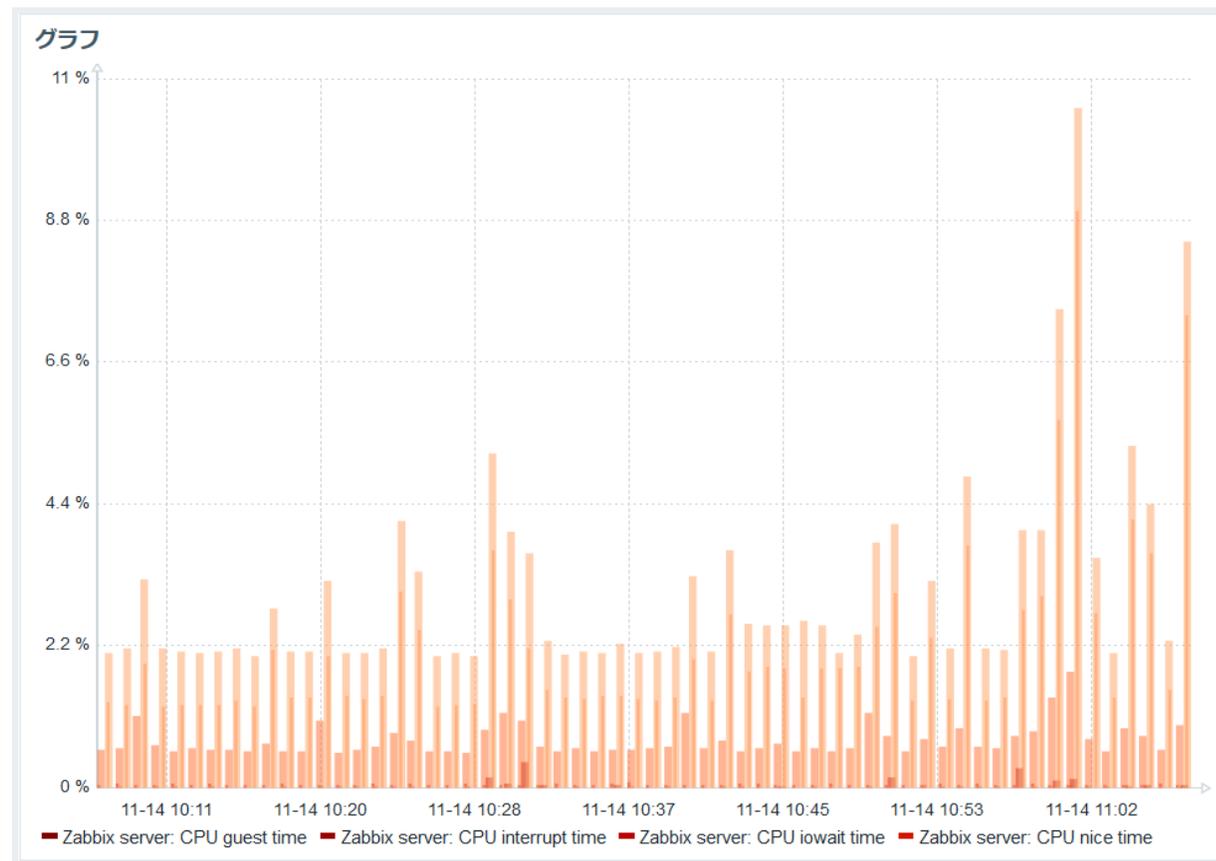
基本色 ■ FF4000

グラフの形式  線  ポイント  棒グラフ  Bar

幅  1

ポイントサイズ  3

透明度  5



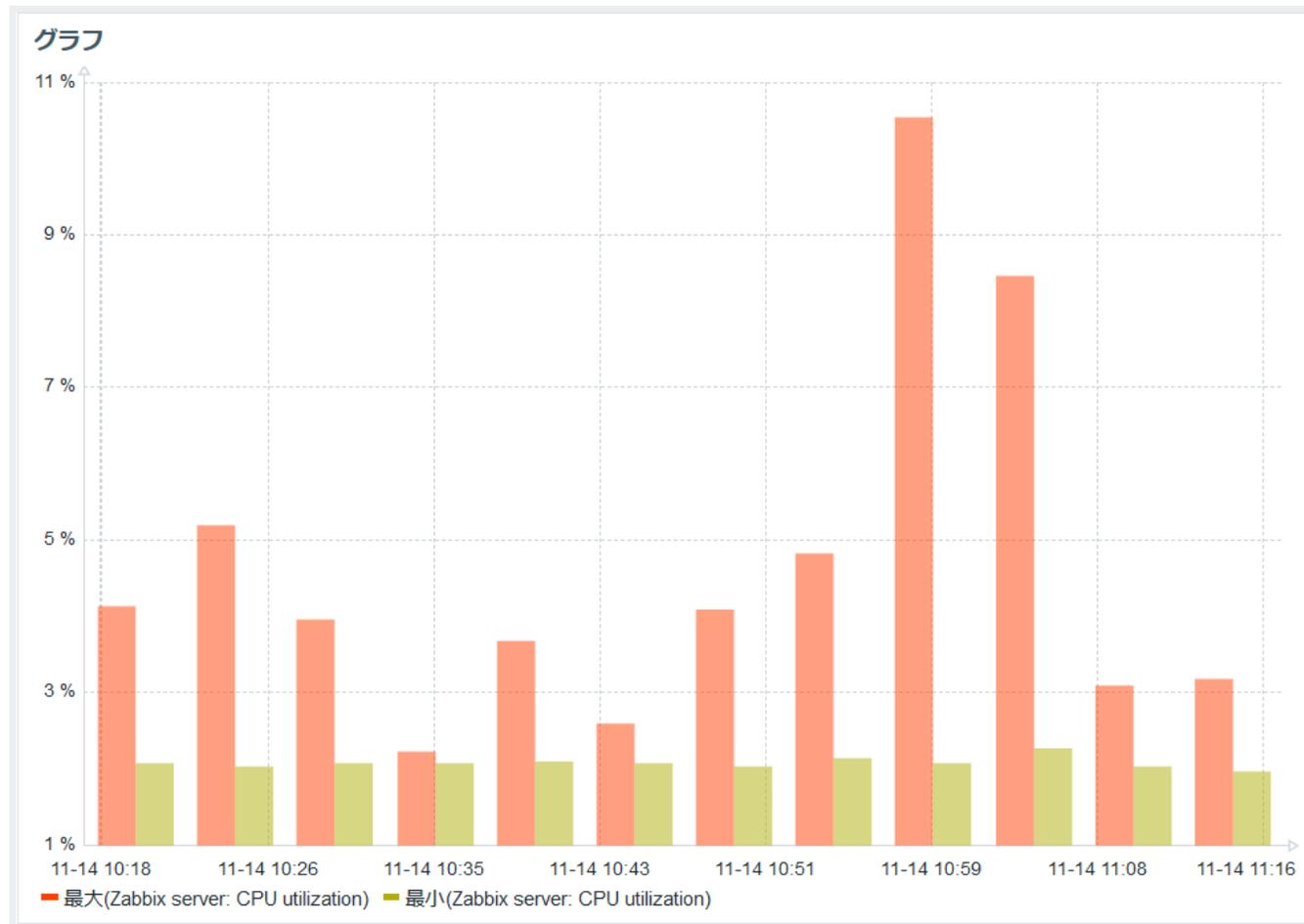
Aggregation function

Aggregation interval

Aggregate  Each item  データセット

Aggregation function

- なし
- 最小
- 最大
- 平均
- count
- sum
- 第 1
- 最新値



## トリガー設定

トリガー タグ 依存関係

\* 名前

Operational data

## 障害画面

Show operational data  なし  Separately  With problem name

時間	<input type="checkbox"/> 深刻度	復旧時刻	ステータス	情報	ホスト	障害	Operational data	継続期間
11:35:13	<input type="checkbox"/> 警告		障害		Zabbix server	↓ High CPU utilization (over 90% for 5m)	Current utilization: 100 %	5m 8s

Show operational data  なし  Separately  With problem name

時間	<input type="checkbox"/> 深刻度	復旧時刻	ステータス	情報	ホスト	障害	Operational data	継続期間
11:35:13	<input type="checkbox"/> 警告		障害		Zabbix server	↓ High CPU utilization (over 90% for 5m) (Current utilization: 100 %)		5m 56s

# ☑ アイテム/トリガーの説明を tooltip で表示

## アイテム

動設定

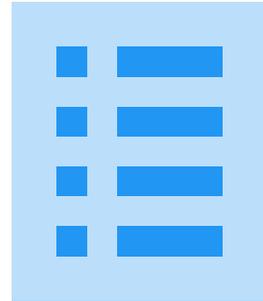
説明

<input type="checkbox"/>	CPU utilization <sup>?</sup>	2019/11/14 11:46:13	2.9337 %
<input type="checkbox"/>	Number of CPUs <sup>?</sup>	2019/11/14 10:45:01	1

## トリガー

説明

時間	<input type="checkbox"/> 深刻度	復旧時刻	ステータス	情報	ホスト	障害	継続期間	確認済	アク
11:59:21	<input type="checkbox"/> 軽度の障害		障害		Zabbix server	Zabbix agent is not available (for 3m) <sup>?</sup>	2s	いいえ	
11:35:13	<input type="checkbox"/> 警告	11:41:13	解決済		Zabbix server	High CPU utilization (over 90% for 5m) <sup>?</sup>			



# その他の新機能

**NEW**

- ☑ db.odbc.get アイテムの追加
- ☑ ホスト一覧画面などでプロキシの列が分離
- ☑ トリガー条件式のフォントが等幅に変更
- ☑ DNS 名での自動登録
- ☑ メディアタイプのエクスポート/インポート
- ☑ エクスポートファイルの構造の改善
- ☑ items テーブルから一部のフィールドを item\_rtdata テーブルに分離

