ZABBIX 5.0 LTS
Freedom and integrity of monitoring
Quick recap of Zabbix 4.2 and 4.4
Zabbix 4.2

April, 2019

High frequency monitoring with throttling

Data collection: HTTP agent, Prometheus

Preprocessing: validation and JavaScript!

Preprocessing by Proxies

Enhanced tag management
Zabbix 4.4
September, 2019

New Zabbix Agent: plugins, scheduler and more

Web hooks for alerting and notifications

Support of TimescaleDB

Built-in knowledge base for metrics and triggers

Standard for Zabbix Templates
5.0 LTS release
May 12, 2020
## Available templates for monitoring & integrations

- **Integrations**
- **Available templates** for monitoring & integrations

- [https://www.zabbix.com/integrations](https://www.zabbix.com/integrations)
Making a platform for high quality solutions

Zabbix Agent 2

Data

Notifications

Incidents

Events

TEMPLATES & PLUGINS

WEBHOOK BASED INTEGRATIONS

WEBHOOK BASED INTEGRATIONS

WEBHOOK BASED INTEGRATIONS
Easy to contribute!

3 simple steps

Sign Zabbix Contributor Agreement (ZCA)
https://www.zabbix.com/developers

Make Zabbix Pull Request
https://git.zabbix.com

Zabbix Dev Team will review and accept if everything is fine
Available everywhere!

Linux distributions and containers

RHEL and CentOS 6, 7 and 8
Debian 8, 9, 10
SuSE 12, 15
16.04 (Xenial), 18.04 (Bionic) and 20.04 (Focal Fossa)
Raspbian 9 (Stretch), 10 (Buster)
Docker

Linux appliance images

ISO
VMWare, VirtualBox
Microsoft Hyper-V
KVM
XEN
LiveCD

Public clouds

- AWS
  - Zabbix Server 4.4
  - Mysql + Nginx
- Azure
  - Zabbix Server 4.4
  - Mysql + Nginx
- DigitalOcean
  - Zabbix Server 4.4
  - Mysql + Nginx
- Google Cloud
  - Zabbix Server 4.4
  - Mysql + Nginx
Official support of Zabbix Agent2 for Linux and Windows
Most advanced monitoring agent on the market!

Plugin infrastructure
Support of long running scripts
Parallel active checks
Support of flexible intervals for all checks
Support of persistent connections (DB connections)
Accepting incoming traps and events (MQTT subscribe, listening TCP/UDP ports, etc)
Monitoring of systemd services out of the box
Drop-in replacement of the existing agent!
Persistent storage for Agent2

Use cases

Unstable communications
Monitoring of critical data
Bursts of data

Your data is safe!

Zabbix Agent2

No connection

EnablePersistentBuffer=1
PersistentBufferFile=/var/spool/zabbix/agent.db
PersistentBufferPeriod=1d

Zabbix Server
Secure by design
Webhooks over HTTP proxy

Internal network
Webhooks over HTTP proxy

Internal network

HTTP Proxy

No direct connection
Restrict available checks on Agent side

# Whitelist for MySQL related checks
AllowKey=mysql[*]
DenyKey=* 

# Blacklist to deny all shell scripts
DenyKey=system.run[*]

# Blacklist to deny access to /etc/passwd
DenyKey=vfs.file.contents[/etc/passwd,*]
Configurable ciphers

Use cases:

Ability to disable weak ciphers
Ability to have white list of ciphers
Ability to be compliant with internal standards

Override the built-in ciphersuite list for certificate:

TLSCipherCert13 - certificate-based ciphersuite selection criteria for TLS 1.3 (only for OpenSSL 1.1.1 or newer),
TLSCipherCert - certificate-based ciphersuite selection criteria for TLS 1.2/1.3 (for GnuTLS), for TLS 1.2 (OpenSSL).

It override the built-in ciphersuite list for PSK:

TLSCipherPSK13 - PSK-based ciphersuite selection criteria for TLS 1.3 (only for OpenSSL 1.1.1 or newer),
TLSCipherPSK - PSK-based ciphersuite selection criteria for TLS 1.2/1.3 (for GnuTLS), for TLS 1.2 (OpenSSL).

To override the built-in combined ciphersuite list for certificate and PSK:

TLSCipherAll13 - ciphersuite selection criteria for TLS 1.3 (only for OpenSSL 1.1.1 or newer),
TLSCipherAll - ciphersuite selection criteria for TLS 1.2/1.3 (for GnuTLS), for TLS 1.2 (OpenSSL).

Example:

TLSCipherCert13=TLS_AES_256_GCM_SHA384
Encrypted connection to database

**Zabbix Server**

**Database**

**Configurable ciphers**

---

**Configure DB connection**

- **Database port**: 0 (default port)
- **Database name**: zebrbx
- **User**: zebrbx
- **Password**: 
- **TLS encryption**: On
- **TLS key file**: C:/zbxb_data/ssl/postgresql.key
- **TLS certificate file**: C:/zbxb_data/ssl/postgresql.crt
- **TLS certificate authority file**: C:/zbxb_data/ssl/root.crt
- **With host verification**: Off
- **TLS cipher list**: DHE-RSA-AES128-GCM-SHA256
Strong encryption of user password hashes

Blowfish instead of MD5
Hash algorithm will be updated on user login
Secret macros

Use cases

Hide any secrets: passwords, tokens, IDs

Data protection

Secret text cannot be retrieved in UI and alerts, masked with *****

No read access, can only be replaced with a new value
Security and encryption

- Whitelists or blacklists for agent metrics
- Encrypted connections to MySQL and PostgreSQL
- SHA256 for password hashes
- Masking secrets in UI and alerts

Diagram:
- Zabbix Agent to Zabbix Server via PSK or TLS
- Agent to HTTP Proxy via HTTPS
- HTTP Proxy to Jira via HTTPS
- Database to Zabbix Server via HTTPS
SAML authentication for single sign-on
Identity providers
Usability improvements
Optimized for wide screens
Build dashboards faster

Copy

Paste to the same or different dashboard
Export graphs as PNG image
Filter by tags for some widgets

Problem by severity & Problem hosts
UI modules

Create new menu entries
Create new pages
Extend existing functionality
Use and share 3rd party extensions
Permission control
Example:

### Structure

- `modules/`
  - `demo_module/`
    - `manifest.json`
    - `Module.php`
  - `actions/`
    - `DemoReportAction.php`
  - `views/`
    - `demo.report.php`

---

### manifest.json

```json
{
  "manifest_version": 1.0,
  "id": "demo.report",
  "version": "1.0",
  "name": "Custom report",
  "namespace": "Demo",
  "author": "Zabbix",
  "url": "http://www.zabbix.com",
  "description": "Demo report module",
  "actions": {
    "demo.report": {
      "class": "DemoReportAction",
      "view": "demo.report"
    }
  }
}
```

### module.php

```php
<?php declare(strict_types = 1);

namespace Modules\Demo;

use APP;
use Core\CModule as BaseModule;

class Module extends BaseModule {
    public function init(): void {
        (APP::Component()->get('menu.main'))
            ->find(_('Reports'))
            ->add('Custom report', [
                'action' => 'demo.report'
            ]);}
}
```
List of monitored devices

Monitoring->Hosts

<table>
<thead>
<tr>
<th>Name</th>
<th>Interface</th>
<th>Availability</th>
<th>Tags</th>
<th>Problems</th>
<th>Status</th>
<th>Latest data</th>
<th>Problems</th>
<th>Graphs</th>
<th>Screens</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS N30</td>
<td>127.0.0.1:10050</td>
<td>ZBX</td>
<td>OS: Linux, Region: us-east-1, Service: Oracle Cluster</td>
<td>1</td>
<td>Enabled</td>
<td>Latest data</td>
<td>Problems 1</td>
<td>Graphs 14</td>
<td>Screens 2</td>
<td>Web</td>
</tr>
<tr>
<td>AWS N34</td>
<td>127.0.0.1:10050</td>
<td>ZBX</td>
<td>OS: Linux</td>
<td>1</td>
<td>Enabled</td>
<td>Latest data</td>
<td>Problems 2</td>
<td>Graphs 14</td>
<td>Screens 2</td>
<td>Web</td>
</tr>
<tr>
<td>AWS N90</td>
<td>127.0.0.1:10050</td>
<td>ZBX</td>
<td>OS: Linux</td>
<td>1</td>
<td>Enabled</td>
<td>Latest data</td>
<td>Problems 3</td>
<td>Graphs 14</td>
<td>Screens 2</td>
<td>Web</td>
</tr>
<tr>
<td>AZ M08</td>
<td>127.0.0.1:10050</td>
<td>ZBX</td>
<td>OS: Linux</td>
<td>1</td>
<td>Enabled</td>
<td>Latest data</td>
<td>Problems 2</td>
<td>Graphs 14</td>
<td>Screens 2</td>
<td>Web</td>
</tr>
<tr>
<td>AZ M10</td>
<td>127.0.0.1:10050</td>
<td>ZBX</td>
<td>OS: Linux</td>
<td>1</td>
<td>Enabled</td>
<td>Latest data</td>
<td>Problems 2</td>
<td>Graphs 14</td>
<td>Screens 2</td>
<td>Web</td>
</tr>
<tr>
<td>AZ M18</td>
<td>127.0.0.1:10050</td>
<td>ZBX</td>
<td>OS: Linux</td>
<td>1</td>
<td>Enabled</td>
<td>Latest data</td>
<td>Problems 2</td>
<td>Graphs 14</td>
<td>Screens 2</td>
<td>Web</td>
</tr>
<tr>
<td>Linux001</td>
<td>127.0.0.1:10050</td>
<td>ZBX</td>
<td>OS: Linux</td>
<td>1</td>
<td>Enabled</td>
<td>Latest data</td>
<td>Problems 2</td>
<td>Graphs 14</td>
<td>Screens 2</td>
<td>Web</td>
</tr>
</tbody>
</table>

No more Monitoring->WEB and Monitoring->Graphs
Easy navigation to host related resources
# List of monitored devices

**Advanced filtering options**

<table>
<thead>
<tr>
<th>Name</th>
<th>Interface</th>
<th>Availability</th>
<th>Tags</th>
<th>Problems</th>
<th>Status</th>
<th>Latest data</th>
<th>Problems</th>
<th>Graphs</th>
<th>Screens</th>
<th>Web</th>
</tr>
</thead>
<tbody>
<tr>
<td>AWS N30</td>
<td>127.0.0.1: 10050</td>
<td>Zabbix</td>
<td>Service: Oracle Cluster OS: Linux Region: us-east-1</td>
<td>1</td>
<td>Enabled</td>
<td>Latest data</td>
<td>Problems</td>
<td>Graphs</td>
<td>Screens</td>
<td>Web</td>
</tr>
<tr>
<td>Linux02</td>
<td>127.0.0.1: 10050</td>
<td>Zabbix</td>
<td>Service: MySQL OS: Linux</td>
<td>1</td>
<td>Enabled</td>
<td>Latest data</td>
<td>Problems</td>
<td>Graphs</td>
<td>Screens</td>
<td>Web</td>
</tr>
</tbody>
</table>

Displaying 2 of 2 found
New preprocessing operator: **Replace**

### Configuration->Items

<table>
<thead>
<tr>
<th>Item</th>
<th>Preprocessing</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Preprocessing steps**

1. **Replace**
   - Parameters: **Up**, **0**
2. **Replace**
   - Parameters: **Down**, **1**

**Add**

- **Update**
- **Clone**
- **Test**
- **Delete**
- **Cancel**

### Typical use cases

- mappings (text -> numeric, numeric -> text)
- removing characters and strings
- replacing characters and strings
- in many cases easier than dealing with regular expressions!
New operator for JSONPath: ~

It returns property names of matching elements

```json
{
    "consul": [],
    "content": [
        "2020.01.05",
        "golang"
    ],
    "login": [
        "2019.11.02",
        "java"
    ],
    "mail": [
        "2020.01.02",
        "golang"
    ]
}
```

\$.~

```json
[
    "consul",
    "content",
    "login",
    "mail"
]
```
Threading for email notifications

Grouped by event ID + media
Use it with secret macros for extra security!

User macros for IPMI user name and password
Mass update of user macros for hosts & templates

<table>
<thead>
<tr>
<th>Host</th>
<th>Templates</th>
<th>IPMI</th>
<th>Tags</th>
<th>Macros</th>
<th>Inventory</th>
<th>Encryption</th>
</tr>
</thead>
</table>

### Macros Table

<table>
<thead>
<tr>
<th>Macro</th>
<th>Value</th>
<th>Description</th>
<th>Remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>{SSERVICE}</td>
<td>PostgreSQL cluster</td>
<td>Our primary production data store</td>
<td></td>
</tr>
</tbody>
</table>

- **Add**
- **Update**
- **Remove**
- **Remove all**

- **Add existing**
- **Update**
- **Cancel**

---

39
Mass update of user macros for hosts & templates
Message templates for media types
Templates for different message types

Email media type

Message template
Configure message format in on place!

Before

After
CLI tool to test JS scripts
Why

Typical use cases

Test JavaScript code from command line:

- webhooks
- complex preprocessing scripts
How to use it?

```bash
shell> zabbix_js -help
```

Execute script using Zabbix embedded scripting engine.

General options:
- `-s,--script script-file` Specify the filename of script to execute. Specify `-` for standard input.
- `-i,--input input-file` Specify input parameter filename. Specify `-` for standard input.
- `-p,--param input-param` Specify input parameter.
- `-l,--loglevel log-level` Specify log level.
- `-t,--timeout timeout` Specify timeout in seconds.
- `-h,--help` Display this help message.
- `-V,--version` Display version number.
Example 1

```bash
shell> cat test.js

return Math.log(value)

shell> zabbix_js -s test.js -p 10

2.302585092994046
```

Example 2

```bash
shell> zabbix_js -s test.js -i my.json # reading input from file
```

Example 3

```bash
shell> cat test.js

Zabbix.Log(3, value) // use Zabbix.Log(log level, text) fo debug purposes

return Math.log(value)
```
Triggers support text operations
Text data

Typical use cases

Working with software versions

Log file monitoring

Comparing string values of different items

Comparing last and previous values

Supported operators: =<>
Comparing with text constant

{host:zabbix.version.last()}=“5.0.0”
{host:zabbix.version.last()}=“{$ZABBIX.VERSION}”

Comparing last value with previous one

{host:text.last()}{host:text.prev()}

OR

{host:text.last(#1)}{host:text.last(#2)}

Comparing values of different items

{hostA:textA.last()}{hostB:textB.last()}
Automation & Discovery
Discovery for JMX counters

New JMX checks

jmx.get[]
jmx.discovery[]

jmx.get[] is similar to the jmx.discovery[] item, but is does not does not turn Java object properties into low-level discovery macro names and therefore can return values without limitations that are associated with LLD macro name generation such as hyphens or non-ASCII characters.

```
jmx.get[beans,"com.example:type=*,*"]
[
    {        
        "object": "com.example:type=Hello.data-src=data-base,ключ=значение",
        "domain": "com.example",
        "properties": { 
            "data-src": "data-base",
            "ключ": "значение",
            "type": "Hello"
        }
    },
    {        
        "object": "com.example:type=Atomic",
        "domain": "com.example",
        "properties": { 
            "type": "Atomic"
        }
    }
]
```

```
jmx.discovery[...]
[
    
    
]
```

"{#JMXDOMAIN}":"java.lang",
"{#JMXTYPE}":"GarbageCollector",
"{#JMXOBJ}":"java.lang:type=GarbageCollector,name=PS Scavenger",
"{#JMXNAME}":"PS Scavenger"
Discovery for Windows perf counters

Zabbix Agent and Agent2

perf_counter.discovery[object]

perf_counter_en.discovery[object]

[{
  "{#INSTANCE}": "0"
},
{
  "{#INSTANCE}": "1"
},
{
  "{#INSTANCE}": "Total"
}]
Better ODBC monitoring

Current situation

ODBC monitoring work only with data sources (DSN) configured in `/etc/odbc.ini` file.

Now everything can be part of item key

```
db.odbc.select[MySQL db check,,"DRIVER=mysqla;
SERVER=127.0.0.1;
PORT=3306;UID=zabbix;
PWD=zabbix;DATABASE=master;OPTION=3;"]
```

```
db.odbc.select[MySQL db check,,${CONN_STRING}]
```

Also, if connection_string is used, and User name field is not empty, it is appended to the connection_string as UID=<user>. Similar, Password field value is appended to the connection_string as PWD=<password>.
Discovery of IPMI sensors

Typical use cases

Simpler templates

```
[{
  "id": "SubTemp12",
  "name": "(7.1).SubTemp12",
  "sensor": {
    "type": "1",
    "text": "temperature"
  },
  "reading": {
    "type": "1",
    "text": "threshold"
  },
  "state": {
    "state": "3",
    "text": "lower Critical - going high"
  },
  "value": "32",
  "units": "C",
  "threshold": {
    "low": {
      "non_crit": "48",
      "crit": "32",
      "non_recover": "16"
    },
    "up": {
      "non_crit": "112",
      "crit": "144",
      "non_recover": "160"
    }
  }
}]
```

```
[{
  "id": "1.8V Switch",
  "name": "(7.1).1.8V Switch",
  "sensor": {
    "type": "2",
    "text": "voltage"
  },
  "reading": {
    "type": "2",
    "text": "discrete_usage"
  },
  "state": {
    "state": "1",
    "text": "transition to active"
  }
}]
```
Test item from UI
For hosts and **templates**
Do not forget to test media types too!

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Status</th>
<th>Used in actions</th>
<th>Details</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Discord</td>
<td>Webhook</td>
<td>Enabled</td>
<td></td>
<td></td>
<td>Test</td>
</tr>
<tr>
<td>Mattermost</td>
<td>Webhook</td>
<td>Enabled</td>
<td></td>
<td></td>
<td>Test</td>
</tr>
<tr>
<td>Opsgenie</td>
<td>Webhook</td>
<td>Enabled</td>
<td></td>
<td></td>
<td>Test</td>
</tr>
<tr>
<td>Pushover</td>
<td>Webhook</td>
<td>Enabled</td>
<td></td>
<td></td>
<td>Test</td>
</tr>
</tbody>
</table>
Support of **user macros** for host prototypes
User macros for host prototypes

<table>
<thead>
<tr>
<th>Host prototype macros</th>
<th>Inherited and host prototype macros</th>
</tr>
</thead>
<tbody>
<tr>
<td>Macro</td>
<td>Value</td>
</tr>
<tr>
<td>($MACRO)</td>
<td>value</td>
</tr>
<tr>
<td></td>
<td>Description</td>
</tr>
<tr>
<td></td>
<td>T</td>
</tr>
<tr>
<td></td>
<td>description</td>
</tr>
</tbody>
</table>

Use LLD macros in macro value and description!
Support of Float64 data types
Float64

Benefits

Compatible with Float64 returned by Prometheus

Execute to upgrade existing installation:

MySQL: database/mysql/double.sql
PostgreSQL: database/postgresql/double.sql
MySQL: database/oracle/double.sql
Scalability improvements
Zabbix UI

Ready to handle

Millions of devices

Improvements made

No drop-downs for host selections anymore, replaced with host search control

Hardcoded maximum size of the Overview grid

Redesigned Monitoring->Hosts->Graphs (multiselection of graphs, displaying of all graphs, pattern matching)

Introduced paging whenever possible (Monitoring ->Hosts->Web)
Compression for efficiency
TimescaleDB = PostgreSQL + Extension
TimescaleDB

Advantages

- Automatic partitioning
- Zabbix manages removal of old data
- Performance oriented DB
- Compression!
TimescaleDB

Administration->General->Housekeeping

History and trends compression

Enable compression

* Compress records older than 7d

Update  Reset defaults
Lower storage cost
Numbers from a production setup

Zabbix 4.x: 355GB  
Zabbix 5.0: 43GB
SNMP setting on host interface level

Item configuration

- Zabbix agent
- Zabbix agent (active)
- Simple check
- SNMP agent
- SNMP trap
- Zabbix internal
- Zabbix trapper
- Zabbix aggregate
- External check
- Database monitor
- HTTP agent
- IPMI agent
- SSH agent
- TELNET agent
- JMX agent
- Calculated
- Dependent item

Host configuration

- Host name
- Visible name
- Groups
- Interfaces
  - Type
  - IP address
  - DNS name
  - Connect to
  - Port
  - SNMP version
  - Context name
  - Security name
  - Security level
- Use bulk requests
Why?

Simplify templates, one template instead of three

Easy to manage: SNMP related parameters on interface level

Easy to switch from SNMPv2 -> SNMPv3
Availability monitoring will respect proxy availability.
Monitoring host availability

HostA is not available

\{\text{HostA:item.nodata(1m)}\} = 1
Monitoring host availability

**Zabbix 4.x**
- HostA is not available
- HostB is not available
- HostC is not available
- Proxy is not available

**Zabbix 5.0**
- Proxy is not available
Monitoring host availability

{HostA:item.nodata(1m)}=1  # respects proxy availability

{HostA:item.nodata(1m, strict)}=1  # strict check, does not respect proxy availability
Manage LLD rules globally
Filter for discovery rules

Useful for: troubleshooting (find all not supported or disabled), mass operations
Ability to unacknowledge event
Unacknowledge it!

Useful for

- fixing mistakes
- better workflow between various teams
Overrides for LLD rules
Discovery of filesystems

Special treating of Oracle related filesystems
Do not discover temporary filesystems

Override

* Name: Do not monitor temporary filesystems

Stop processing next overrides if matches

Filters

Type of calculation: And/Or

Label Macro

A: (#FSNAME)

Regular expression: tmp

Action: Remove

Operations

Condition

Trigger prototype equals

Item prototype equals

Add

New operation

Object: Item prototype

Condition: equals

Pattern: pattern

Create enabled: No

Original

Discover: Yes

Update interval: No

History storage period: No

Trend storage period: No

Add

Cancel
New macros

**ZBXNEXT-1797** support of macro {HOST.ID} in notifications

Can be used to build URLs to Zabbix UI. For example, Latest data:

```
{$ZABBIX.URL}/zabbix.php?action=latest.view&filter_set=1&filter_hostids%5B0%5D={HOST.ID}
```

**ZBXNEXT-5369** support of macro {EVENT.TAGSJSON} in notifications

Easier to pass all tags to webhooks

**ZBXNEXT-252** support of macro {EVENT.DURATION} in notifications.

Recovery subject “Resolved in 5m: Service Nginx is down.”
Other improvements

ZBXNEXT-5848 increased size of acknowledge messages to 4K (was 256)
ZBXNEXT-5690 added support of LIBSSH to support newer platforms like RHEL 8
ZBXNEXT-5825 support of ElasticSearch 7.x (7.4, 7.6)
ZBXNEXT-5720 Latest data displays data if filter is not set
ZBXNEXT-1561 increased zabbix_sender time resolution to nanoseconds
ZBXNEXT-1234 Monitoring->Latest data: show data if filter is empty
ZBXNEXT-5734 Base64 processing in JavaScript, functions atob() and btoa()
ZBXNEXT-5604 Do not log system.run[] for local use
ZBXNEXT-4584 New API method to get auditlog
ZBXNEXT-5851 Remote monitoring of versions of Zabbix components
And more

ZBXNEXT-1989 Increased size of item key to 2048 characters (was 255)
ZBXNEXT-3940 Ability to flush SNMP cache, SNMPv3 context changes
ZBXNEXT-5829 Faster hash function for internal operations
ZBXNEXT-2081 Documented how to do filtering for vmware.event monitoring
ZBX-15914 Improved consistency of map labels
Removing legacy to make a better product, faster

ZBXNEXT-5697 No support of Internet Explorer 11 anymore

ZBXNEXT-5592 Dropped support of IBM DB2 database

ZBXNEXT-5716 mbedTLS (former polarSSL) is no longer supported for encryption. Only OpenSSL and GnuTLS libraries

Minimum supported version for PHP is now 7.2: safer and more strict code
Upgrade!

Just install new server binaries and front-end files
Upgrade!

Important notes

Upgrading history data to Float64 is optional and may take time
Compressed TimescaleDB data is read-only
PHP 7.2 is required
No DB2 support anymore

Read Upgrade Notes for more details!
https://www.zabbix.com/documentation/5.0/manual/installation/upgrade_notes_500
ETA:
mid-October, 2020
Roadmap

This page contains an incomplete list of planned functionality that can be updated anytime without prior notice.

Zabbix 5.2

ETA: Q4, 2020

Smart problem and anomaly detection

- Zabbix Insights
  A new set of trigger functions for long term analysis of trend data to enable alerts like “average number of transactions increased by 24% in September”.

Make monitoring more secure

- Roles for more granular user permissions
  It will control what parts of UI are available and will also introduce fine-grained management of the availability of Zabbix API methods.

- Support storage of secrets in an external Vault
  Zabbix will be able to securely keep all sensitive information (like passwords, SNMP community names, tokens, etc.) in a Vault. It will make Zabbix compliant to the most strict security standards.

https://www.zabbix.com/roadmap
Smart problem and anomaly detection

- Zabbix Insights
  A new set of trigger functions for long term analysis of trend data to enable alerts like “Average number of transactions increased by 24% in September”.

High availability, performance, and scalability

- Zabbix UI and API to support load balancing
- Clean separation of data collection and trigger processing components

Make monitoring more secure

- Roles for more granular user permissions
  It will control what parts of UI are available and will also introduce fine-grained management of the availability of Zabbix API methods.
- Support storage of secrets in an external Vault
  Zabbix will be able to securely keep all sensitive information (like passwords, SNMP community names, tokens, etc.) in a Vault. It will make Zabbix compliant to the most strict security standards.

Monitoring of IoT devices

- Support data collection using modbus and MQTT protocols

Usability and user experience

- Support of predefined filters for Problems and Hosts for instant switching from one view to another
- Switching to Yaml as the default format for templates and import/export operations
- Selection of timezone for individual users
- Versioning of the templates to simplify the deployment of enhanced templates
  It could be used for monitoring automation when templates are managed externally, for example, stored in an external Git repository.
Thank you!

Some of the used icons made by Freepik from www.flaticon.com