



ZABBIX 5.0

Certified Specialist Training

Day 3

Rules

It is prohibited to make any video and/or audio recordings during the whole period of this course.

This course is intended only for the officially enrolled student. Subject to the Copyright Notice below, the student is not allowed to share his credentials for attending this course, to allow others to join and take part, or otherwise make use of these Materials.

Copyright notice

© Zabbix, 2020. All rights reserved.

Unless otherwise indicated, Zabbix owns the copyright and other intellectual property rights in the text, graphics, information, designs, data, verbal/audio/video presentations and files, comments, drawings, exam questions and exam answers, and other training content, lab manuals and practical tasks, and training courses themselves (further – Materials).

The Materials are protected by watermarks, copyright statements, and other means. It is prohibited to remove any of watermarks and copyright statements, or in any other way to amend or change the content or appearance of the Materials.

Any unauthorized reprint, publication, reproduction, sharing, or use of the Materials is prohibited. No part of the Materials may be reproduced, transmitted, or published in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage and retrieval system without the express signed written permission from Zabbix.

All course Materials made available to the student during the course of the training may be used solely by the student enrolled in the relevant course for personal and educational purposes only. Materials provided to the student should be treated as confidential information shared with the student only for the purpose of the student performing Zabbix Certified training.

The student acknowledges that damages alone would not be an adequate remedy for the breach of this copyright and the student shall be entitled to the granting of equitable relief concerning any threatened or actual breach of any of the provisions of this Copyright notice.

AGENDA

Zabbix sender



SSH/Telnet checks



HTTP Checks



Dependent Items

Calculated
Checks



Aggregate
Checks



SNMP monitoring



Log file
Monitoring



Web monitoring





Zabbix sender

Trapper items accept incoming data instead of querying for it

⚠ To use a trapper item:

- Set up a trapper item in Zabbix
- Use "zabbix_sender" command-line utility to send in the data

⚠ Allowed hosts: incoming connections will be accepted only from the hosts listed here.

The screenshot shows the configuration of a 'Trapper item' in the Zabbix interface. The 'Name' field is set to 'Example trapper item'. The 'Type' is 'Zabbix trapper' and the 'Unique key' is 'trapper.key.1'. The 'Type of information' is 'Numeric (unsigned)' and the 'Units' are '!units'. The 'History storage period' is 'Do not keep history' and the 'Trend storage period' is 'Do not keep trends'. The 'Show value' option is 'As is'. In the 'Allowed hosts' section, the input field contains '{HOST.CONN},1.2.3.4, any.dns.name, 192.168.0.0/24'. The 'New application' dropdown is set to 'Trapper items'.

```
# zabbix_sender -z <Zabbix server IP/DNS> -s <HOST NAME> -k trapper.key.1 -o 007
```

- ~ Is useful to integrate other data sources
- ~ Can send multiple values from a whitespace delimited file:
 - <hostname> <key> <value>
- ~ Can send multiple timestamped values from a file:
 - <hostname> <key> <timestamp> <value>
- ~ Timestamp supports nanoseconds
- ~ 250 values in a single connection
- ~ Encryption support

Examples:

```
# zabbix_sender -z monitoring.zabbix.com -s OracleDB3 -k db.connections -o 43
# zabbix_sender -c /etc/zabbix/zabbix_agentd.conf -i /var/log/perf.txt
```

Output:

```
Response from "host:10051": "processed: 1; failed: 0; total: 1; seconds spent: 0.012226"
sent: 1; skipped: 0; total: 1
```

心脏病图标 Use “-” to read from the standard input.

```
# echo DB01 db.tps 10 | zabbix_sender -z 127.0.0.1 -i -
```

心脏病图标 Use “-r” to send values one by one as soon as they are received.

```
# echo DB01 db.tps 10 | zabbix_sender -z 127.0.0.1 -r -i -
```

心脏病图标 Add “-c” to use all addresses defined in the agent's configuration parameter ServerActive.

```
# echo - db.tps 15 | zabbix_sender -c /etc/zabbix/zabbix_agentd.conf -i -
```

Example of encrypted communication:

```
# zabbix_sender -z 192.168.1.113 -s "DB01" -k mysql.queries -o 342.45 \
--tls-connect psk \
--tls-psk-identity "PSK ID Zabbix agentd" \
--tls-psk-file /home/zabbix/zabbix_agentd.psk
```



Documentation [5.0/manual:sender](#) and [manpages:Zabbix_sender](#)

PRACTICAL SETUP

1. Create an item on Template Basic":

- ~ Name: Number of persons in the room
- ~ Key: persons
- ~ Units: !persons
- ~ Accept incoming connections only from the training hosts
- ~ Add a preprocessing rule to validate data
 - Accept values from 1 to 20 (use user macros {\$FROM}, {\$TO})
 - If the received value is out of range, set error to "Value not in range {\$FROM}-{\$TO}"

2. Send values via Zabbix sender (e.g. 5, 10000, etc.)

3. Make sure that the item receives data

4. Create a trigger on Template Basic":

- ~ Name: Only 2 persons are attending the training! (use a macro)

5. Send some values to check , whether the trigger works.



Advanced task: Send metrics from file with custom timestamps



SSH / Telnet CHECKS

SSH checks are performed as agent-less monitoring:

- ✓ performed by Zabbix server or proxy
- ✓ Zabbix agent is not required
- ✓ can execute any command on the remote host and return result back to Zabbix
- ✓ a password or a public key can be used for authentication

Key: ssh.run[<unique short description>,<ip>,<port>,<encoding>]

The screenshot shows the Zabbix configuration interface for creating a new check. The form fields are as follows:

- * Name: Status of MySQL server
- Type: SSH agent
- * Key: ssh.run[mysql.status]
- Select button (next to Key)
- Authentication method: Password
- * User name: {\$SSH.USERNAME}
- Password: {\$SSH.PASSWORD}
- * Executed script: /usr/bin/mysqladmin ping | grep -c alive
- Type of information: Character



Make sure that login credentials are valid and no prompts are displayed.

To use a key for authentication, additional server configuration is required:

~ /etc/zabbix/zabbix_server.conf

```
### Option: SSHKeyLocation
# SSHKeyLocation=
```

Example:

```
SSHKeyLocation=/home/zabbix/.ssh
```

Execute script:

* Executed script `/usr/bin/mysqladmin ping | grep -c alive`

~ Any shell command - "one liner"

~ Use "&&" to run multiple commands

* Name	Status of MySQL server
Type	SSH agent
* Key	ssh.run[mysql.status]
Authentication method	Public key
* User name	<code>{\$SSH.USERNAME}</code>
* Public key file	<code>id_rsa.pub</code>
* Private key file	<code>id_rsa.key</code>
Key passphrase	<code>{\$SECURE.PASSWORD}</code>



https://www.zabbix.com/documentation/5.0/manual/config/items/itemtypes/ssh_checks

Telnet checks are performed as agent-less monitoring.

- ~ Work similarly to SSH checks
- ~ Username and password are sent over the network in plain text

Key: telnet.run[<unique short description>,<ip>,<port>,<encoding>]

The screenshot shows the 'Create item' dialog in Zabbix. The 'Type' is set to 'TELNET agent'. The 'Key' field contains 'telnet.run[unique.description]'. Other fields include 'Host interface' (superior.dns.name : 10050), 'User name' ({\$TELNET.USER}), 'Password' ({\$TELNET.PASS}), and 'Executed script' (commands). A 'Select' button is visible next to the Key field.

* Name	Telnet item name
Type	TELNET agent
* Key	telnet.run[unique.description]
* Host interface	superior.dns.name : 10050
* User name	{\$TELNET.USER}
Password	>{\$TELNET.PASS}
* Executed script	commands

Supported characters that the shell prompt can end with:

- ~ \$ # > %



https://www.zabbix.com/documentation/5.0/manual/config/items/itemtypes/telnet_checks

PRACTICAL SETUP

- 1) On your Training-VM-XX:
 - ~ Link template: Template App SSH Service
 - ~ Check whether SSH service is available
- 2) On your host create a new user (use SSH console):
 - ~ Name: monitor
 - ~ Password: sshremoteXX
- 3) In the "Template Basic":
 - ~ Create a new item (Name: Memory available", Type: SSH check)
 - ~ Create a new macro for SSH password authentication
 - ~ Use "cat /proc/meminfo" command
 - ~ Preprocess received values to get only "Memory Available"
- 4) Make sure that the item receives data from all Training-VM-XX hosts.
- 5) In the "Template Basic", create a trigger:
 - ~ Available memory is <100M (use a macro)

 Advanced task: Create an SSH item to get 10 top processes by CPU usage.



HTTP checks

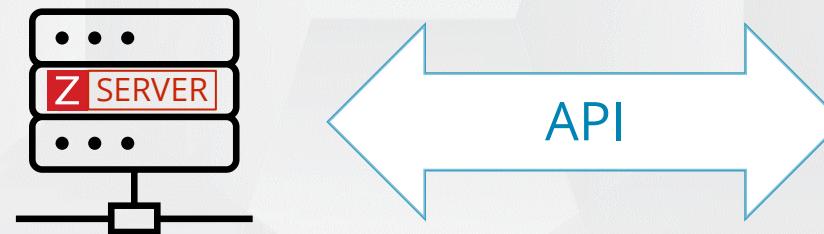
Many applications are exposing their data via RESTful APIs

✓ A RESTful API uses HTTP requests to access or modify data:

- POST (create)
- GET (read)
- PUT (update)
- DELETE (delete)

HTTP checks collect data directly from any web service's endpoint.

- ✓ are executed by Zabbix server or proxy
- ✓ allow data polling using HTTP/HTTPS protocol
- ✓ make it easy to monitor applications and services
- ✓ Zabbix agent is not required



HTTP agent uses curl libraries (libcurl) to query data from servers

* Name

Type

* Key

* URL

Query fields

Name	Value
pretty	true

[Add](#)

Request type

Timeout

Request body type Raw data JSON data XML data

Request body

The same can be achieved from the CLI with curl command and parameters:

```
curl -s -X POST -H "Content-Type:application/json" -H "AuthToken:djA546Z@E"  
https://example.com/api/backups?pretty=true -d '{"status": "Failed"}'
```

1. Enter a unique item key
2. A URL to connect to and retrieve data (supports macros)
3. Variables for the URL

macro variable

! `https://{{HOST.CONN}}/api/backups?pretty=true`

The screenshot shows the configuration for an 'HTTP agent' type check. The 'Name' field is set to 'List of failed backups'. The 'Key' field is set to 'list.backups[failed]'. The 'URL' field is set to 'https://{{HOST.CONN}}/api/backups'. In the 'Query fields' section, there is one entry with 'Name' as 'pretty' and 'Value' as 'true'. Red numbers 1, 2, and 3 are overlaid on the configuration fields to correspond with the numbered steps in the list above.

* Name	List of failed backups
Type	HTTP agent
1 * Key	list.backups[failed]
2 * URL	https://{{HOST.CONN}}/api/backups
Parse	
Query fields	
3	Name: pretty Value: true Remove Add

4. Select a request method type: GET, POST, PUT or HEAD

5. Maximum time for making connection and performing HTTP request:

~ From 1 to 60 seconds

~ Not affected by global Timeout defined in the zabbix_server.conf

6. Select the request body type: Raw, JSON or XML data

7. Request body

The screenshot shows a configuration form for an HTTP check. The steps are numbered as follows:

- 4** Request type: A dropdown menu set to "POST".
- 5** Timeout: An input field containing "3s".
- 6** Request body type: A tabbed panel with three tabs: "Raw data" (selected), "JSON data", and "XML data".
- 7** Request body: A text area containing the JSON string: `{"status": "Failed"}`.

8. Custom HTTP(s) Headers to send

~ Specified as attribute and value pairs.

9. List of expected HTTP status codes

~ For example: 200,201,210-299

10. Select which response part to retrieve : Body, Headers, or Body and headers

The screenshot shows the configuration interface for an HTTP check in Zabbix. The interface includes the following sections:

- Headers:** A table with columns "Name" and "Value". It contains one entry: "AuthToken" with value "djA6ZmE0MDhiNDAtNWFIMy00ZjNjLT". There is a "Remove" link next to the value. An "Add" button is also present.
- Required status codes:** A field containing the value "200".
- Follow redirects:** A checkbox that is unchecked.
- Retrieve mode:** A radio button group with three options: "Body" (selected), "Headers", and "Body and headers".
- Convert to JSON:** A checkbox that is unchecked.

Red circles with numbers 8, 9, and 10 are overlaid on the interface to indicate specific configuration steps:

- ⑧ Points to the "Headers" section.
- ⑨ Points to the "Required status codes" field.
- ⑩ Points to the "Retrieve mode" radio buttons.

11. HTTP proxy to use:

- ~ An optional protocol:// may be used to specify proxy protocols (e.g. https, socks4)
- ~ By default, 1080 port is used

12. Authentication type: None, Basic or NTLM authentication

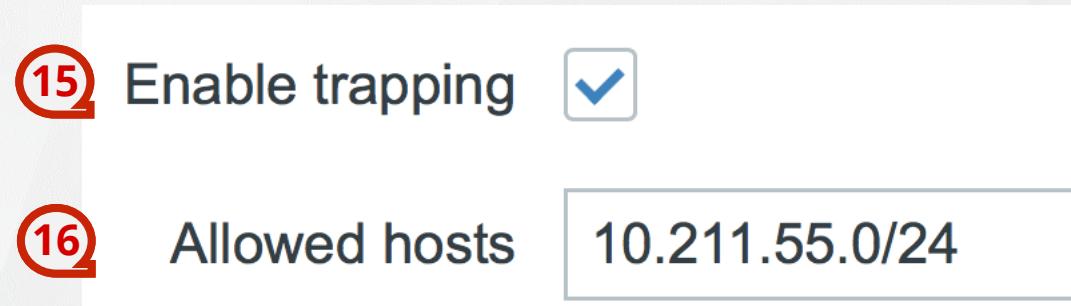
13. Checkboxes to verify SSL certificate of the web server

14. SSL parameters used for client authentication

The screenshot shows a configuration form for SSL parameters. The fields are numbered 11 through 14:

- 11** HTTP proxy: A text input field containing the placeholder `[protocol://][user[:password]@]proxy.example.com[:port]`.
- 12** HTTP authentication: A dropdown menu set to "None".
- 13** SSL verify peer: An unchecked checkbox.
- 13** SSL verify host: An unchecked checkbox.
- SSL certificate file**: An empty text input field.
- SSL key file**: An empty text input field.
- 14** SSL key password: An empty text input field.

15. Enable to accept data sent by Zabbix sender (push)
16. Incoming connections will be accepted only from these hosts



An item will become unsupported, if:

- ~ a status code is not in the expected HTTP status codes list
- ~ the timeout is exceeded
- ~ a wrong proxy protocol/port has been specified
- ~ information type is not selected correctly

Apache monitoring example

→ Create a VirtualHost for server monitoring and secure access:

```
## create resource for apache Enable the server-status page.
```

```
vi /etc/httpd/conf.d/serverstatus.conf
```

```
Listen 127.0.0.1:8080
<VirtualHost localhost:8080>
<Location /server-status>
RewriteEngine Off
SetHandler server-status
Allow from 127.0.0.1
Order deny,allow
Deny from all
</Location>
</VirtualHost>
```

⚠ In the item configuration, use link: <http://{HOST.CONN}:8080/server-status?auto>

* URL Parse

Query fields

Name	Value
auto	value

[Add](#) [Remove](#)

⚠ Example output:

```
Total Accesses: 7
Total kBytes: 22
Uptime: 7
ReqPerSec: 12
BytesPerSec: 3218.29
BytesPerReq: 3218.29
BusyWorkers: 2
IdleWorkers: 5
Scoreboard: K__W_.....
```

PRACTICAL SETUP

1. Check Apache health page on the training.lan host
 - ~- URL: http://TRAINING.LAN_IP_ADDRESS:8080/server-status?auto
2. Create a new template
 - ~- Name: Template Basic App Apache status
 - ~- Group: Training/Templates
3. Create a new item on “Template Basic App Apache status”
 - ~- Name: Apache server status
 - ~- Type: HTTP agent
 - ~- Key: apache.server.status
 - ~- URL: <http://training.lan/server-status?auto>
4. Link the new template to the Training Resources host
5. Check whether the information is collected properly



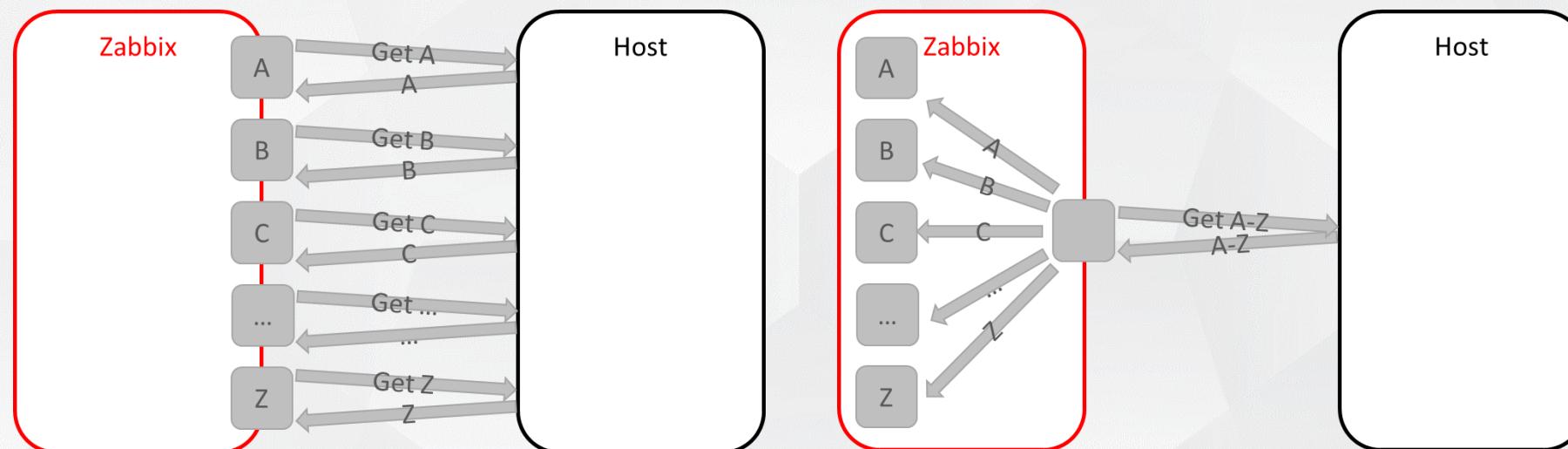
Advanced task: Configure Apache on your host, link the built-in Apache by HTTP template



Dependent items

Zabbix supports dependent items

- ~ They allow for bulk metric collection and simultaneous use in several related items
- ~ Master item automatically populates the values of the dependent items
- ~ Item preprocessing must be used to extract the part that is needed for the dependent item from the master item data
- ~ Zabbix server and proxies process dependent items



There are many situations, when Zabbix may get several values at a time:

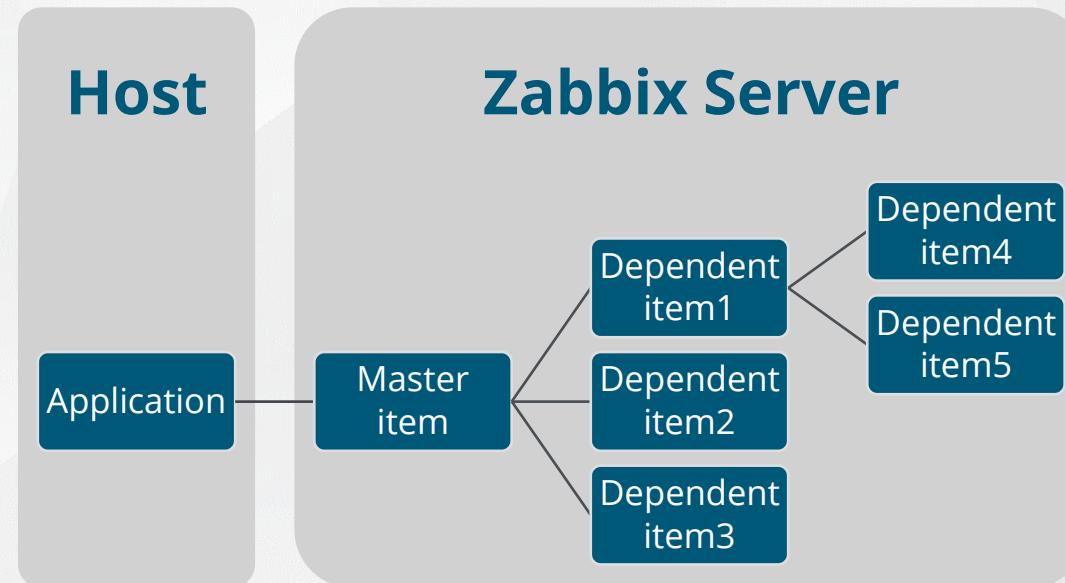
- ~ an external check with command line utilities
- ~ a loadable module that gets multiple values via API
- ~ a user parameter with an SQL query
- ~ SSH agent checks with bulk requests
- ~ Etc.



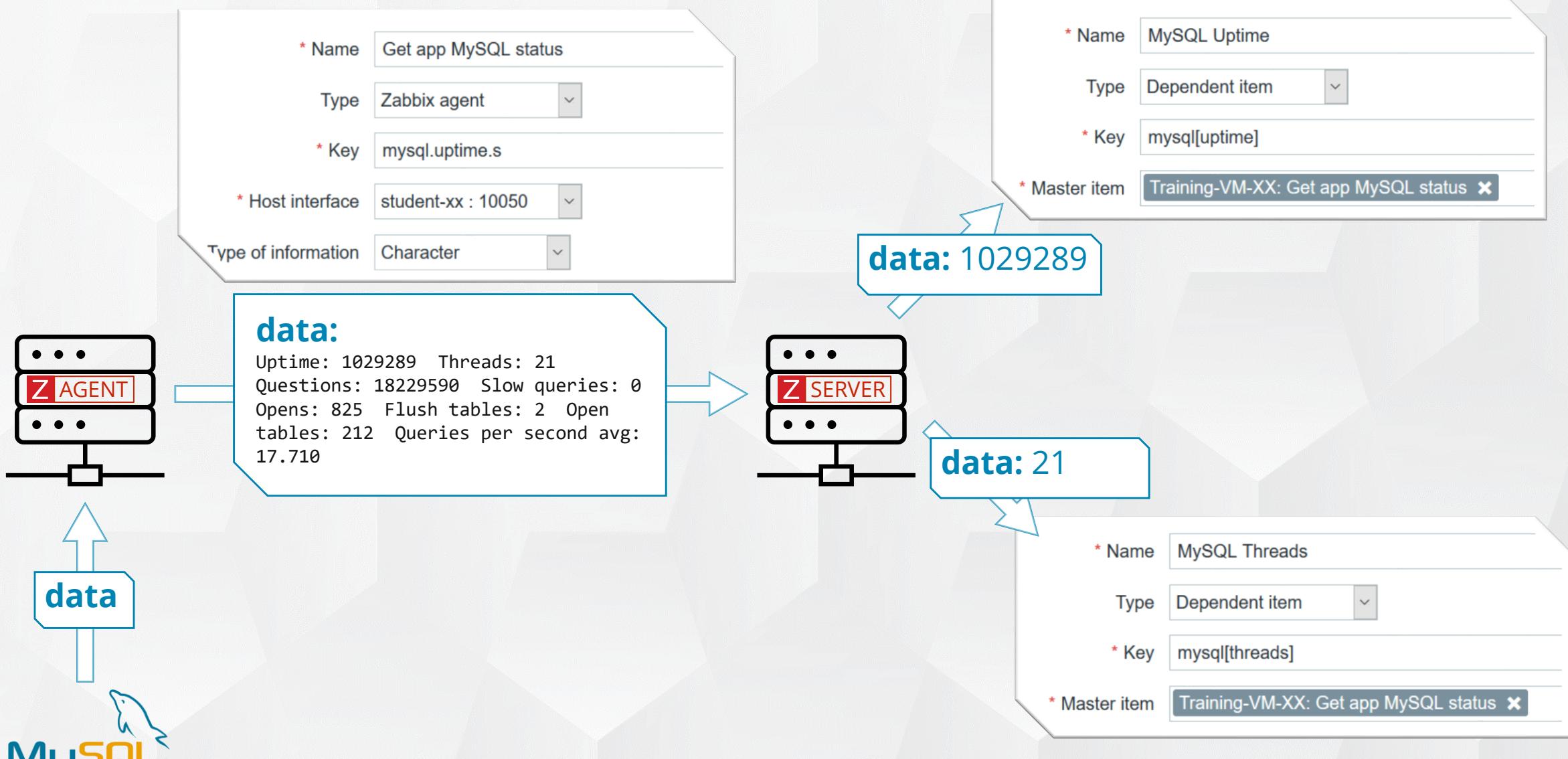
Handler_update	0
Handler_write	414
Innodb_buffer_pool_dump_status	Dumping of buffer pool not started
Innodb_buffer_pool_load_status	Buffer pool(s) load completed at 170531 10:45:37
Innodb_buffer_pool_resize_status	
Innodb_buffer_pool_pages_data	513
Innodb_buffer_pool_bytes_data	8404992
Innodb_buffer_pool_pages_dirty	0
Innodb_buffer_pool_bytes_dirty	0
Innodb_buffer_pool_pages_flushed	37
Innodb_buffer_pool_pages_free	7676
Innodb_buffer_pool_pages_misc	2
Innodb_buffer_pool_pages_total	8191
Innodb_buffer_pool_read_ahead_rnd	0
Innodb_buffer_pool_read_ahead	0
Innodb_buffer_pool_read_ahead_evicted	0
Innodb_buffer_pool_read_requests	2535
Innodb_buffer_pool_reads	479
Innodb_buffer_pool_wait_free	0
Innodb_buffer_pool_write_requests	515
Innodb_data_fsyncs	7
Innodb_data_pending_fsyncs	0
Innodb_data_pending_reads	0
Innodb_data_pending_writes	0
Innodb_data_read	7918080
Innodb_data_reads	505
Innodb_data_writes	54
Innodb_data_written	641024



- ~ Item of any type can be set as the master item
- ~ A large block of data collection provided by a single call
- ~ Significant improvement in performance and efficiency



- ~ The result can be parsed without external scripts/utilities
- ~ Dependent items can be used to extract smaller parts from the value
- ~ If a master item is deleted, so are all its dependent items



⚠ Only same host (template) dependencies are allowed

- Maximum 3 dependency levels allowed

⚠ One master item is limited to 29999 dependent items

- Regardless of the number of dependency levels

⚠ It is recommended to not store history for master item if possible

- If all data are extracted by dependent items, master item contains just extra copy of data
- Master item data may consume large amount of database space

⚠ Dependent items are only updated when master item retrieves new values

- It is not possible to forcibly check just one dependent item

⚠ Dependent item on a host with master item from template will not be exported to XML

PRACTICAL SETUP

1. Create three dependent items on “Template Basic App Apache status”:
 - Master item: Application Apache status page
 - Dependent items:
 - Apache server uptime
 - Apache server total accesses
 - Apache server total kBytes
 - Use preprocessing to extract and transform the values
2. Check the Training Resources host for new dependent items
3. Make sure the data are received



Advanced task: Transform the mysql.uptime.s item into the master item, extract some data.



Calculated checks

Calculated items are used to transform, combine or use in calculations the data received from hosts

- ~ Zabbix agent is not required
- ~ Calculation is done by Zabbix server
- ~ Keys of the items used in the formula must match exactly
- ~ If item keys used in the formula has been changed – the calculated item must be updated

Syntax:

- ~ `func(<key>|<hostname:key>,<parameter1>,<parameter2>,...)`
 - Functions supported in trigger expressions: last, min, max, avg, count, etc

Examples:

- ~ `(last("cpu.temperature[C]")*9/5)+32`
- ~ `100*last("net.if.in[eth0,bytes])/last("net.if.in[eth0,bytes])+last("net.if.out[eth0,bytes])`



<https://www.zabbix.com/documentation/5.0/manual/config/items/itemtypes/calculated>

PRACTICAL SETUP

1. Create a new item on Template Basic":

- ~ Name: Interface eth0: Total traffic" (bytes per second)
- ~ Type: calculated
- ~ Formula: (sum of "Incoming traffic on eth0" and "Outgoing traffic on eth0")

2. Make sure that the item receives data.

3. Are calculation results displayed correctly in the Latest data?

- ~ Use scheduling to collect incoming/outgoing data every 10 seconds.
- ~ Schedule collection to 10 seconds but shifted for 2 seconds for the "Total throughput on eth0" item as well .

4. Is it a good practice to use scheduling on a large-scale? What can be the possible impact of such setup ?



Aggregate checks

Aggregate checks are used to get summarized data from a group of hosts

- ~ Zabbix agent is not required
- ~ Calculation is done by Zabbix server
- ~ Host group names and keys must match exactly
- ~ If the item keys or host group names used in the formula have been changed - aggregated item must be updated

Syntax:

- ~ `groupfunc["Host group","Item key",itemfunc,timeperiod]`
 - Functions: grpavg, grpmax, grpmin, grpsum
 - Item functions: avg, count, last, max, min, sum

Examples:

- ~ `grpsum["MySQL Servers","vfs.fs.size[/,total]",last]`
- ~ `grpavg["MySQL Servers",mysql.qps,avg,5m]`
- ~ `grpavg[["Server group A","Server group B"],system.cpu.load,last]`

PRACTICAL SETUP

1. Create new:

- ~ Host group: Training/HA clusters
- ~ Host: Training HA cluster
- ~ Template: Template Basic Aggregate Check
 - Add it to the "Training/Templates" group

2. On "Template Basic Aggregate Check":

- ~ Add an aggregate item: "Average CPU load in cluster"
- ~ Calculate an average CPU load on all systems from the "Training/Servers" host group

3. Link "Template Basic Aggregate Check" to the Training HA cluster host.

4. Make sure that the item receives data.



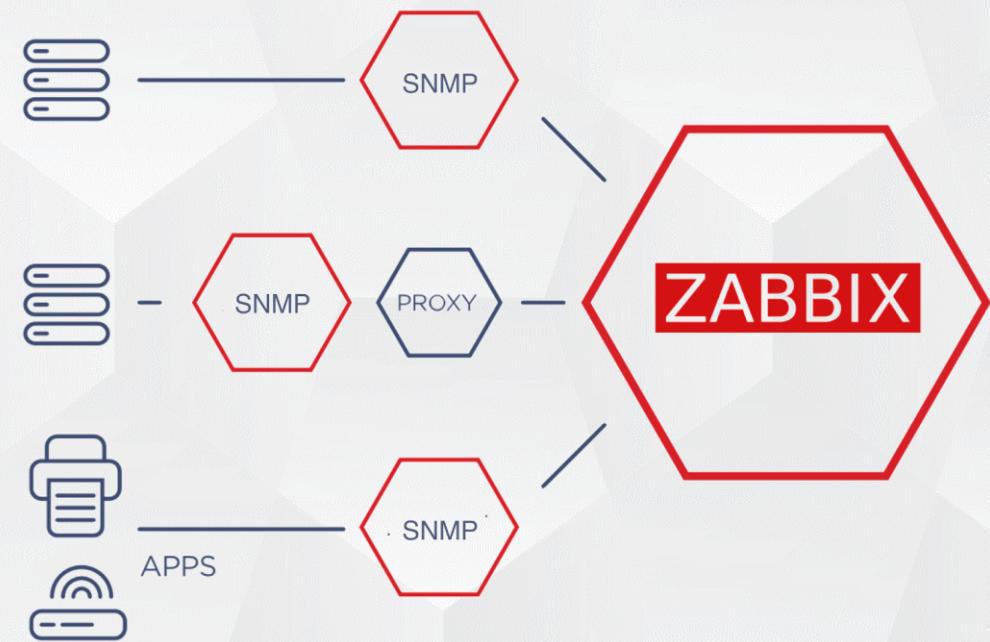
SNMP monitoring

SNMP is a powerful protocol that Zabbix can use to monitor:

- ♥ Network devices
- ♥ Regular computers and servers
- ♥ Applications
- ♥ Anything that supports the SNMP protocol

Zabbix supports SNMP v1, 2c and 3

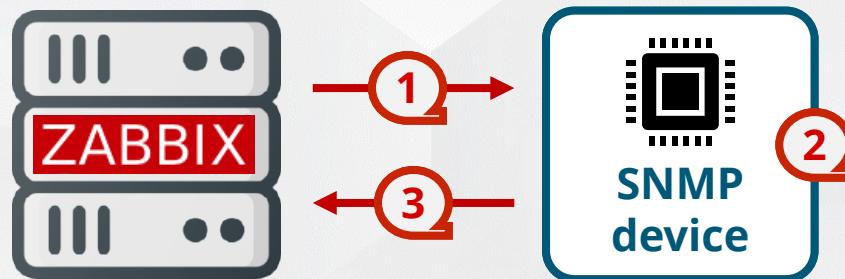
- ♥ SNMPv1 and v2:
 - uses community names for read/write
- ♥ SNMPv3:
 - uses username and password
 - provides authentication and encryption
 - SNMP engine ID must be unique per device



<https://www.zabbix.com/documentation/current/manual/config/items/itemtypes/snmp>

Zabbix server requests data and the devices send back metrics using the SNMP protocol (UDP/161)

- ~ Zabbix server/proxy sends request to a device
 - Possible to request a single value
 - Possible to request multiple values simultaneously if "Use bulk request" is checked
- ~ SNMP agent on the device accesses a table with requested information in the OID format
- ~ Sends back requested values or, if a wrong/not found OID was requested, an error



! Some SNMP devices do not support bulk requests.

- ~ Typical objects to monitor are network traffic, port status, cartridge states, etc.
- ~ MIB is a formatted text file that lists the data objects used by equipment
- ~ OIDs uniquely identify managed objects in the MIB
- ~ OID is a long sequence of numbers, coding the nodes, separated by the dots

MIB:

sysUpTime OBJECT-TYPE
SYNTAX TimeTicks
ACCESS read-only
STATUS mandatory

seconds/100

DESCRIPTION

"The time (in hundredths of a second)
since the network management portion
of the system was last re-initialized."

OID:

iso(1).org(3).dod(6).internet(1).
mgmt(2).mib-2(1)
.system(1).sysUpTime(3)

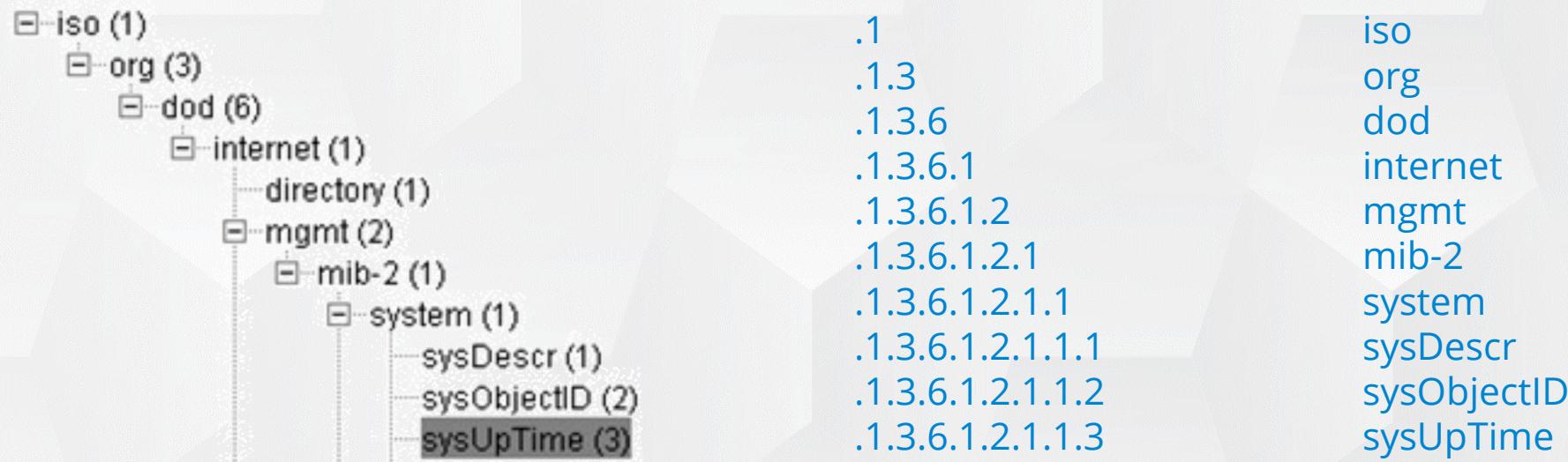
or

1.3.6.1.2.1.1.3



When reading, pay close attention to the kind of information the objects provide

- ~ MIB is organized hierarchically and can be represented as a tree
- ~ Each OID has an address that follows the levels of an OID tree
- ~ Hardware manufacturers often provide suitable MIB files



.1
.1.3
.1.3.6
.1.3.6.1
.1.3.6.1.2
.1.3.6.1.2.1
.1.3.6.1.2.1.1
.1.3.6.1.2.1.1.1
.1.3.6.1.2.1.1.2
.1.3.6.1.2.1.1.3

iso
org
dod
internet
mgmt
mib-2
system
sysDescr
sysObjectID
sysUpTime

! **.1.3.6.1.2.1.1.3 = iso.org.dod.internet.mgmt.mib-2.system.sysUpTime**

When creating a host to monitor, add an SNMP interface:

* Interfaces	Type	IP address	DNS name	Connect to	Port	Default
SNMP	1.2.3.4	training.lan		IP	DNS	161
* SNMP version	SNMPv2					
* SNMP community	{\$SNMP_COMMUNITY}					
<input checked="" type="checkbox"/> Use bulk requests						
Add						

⚠️ SNMP version 3 will show additional fields for authentication and encryption

If necessary, override the community name macro on a template or host level:

Macros	Inventory	Encryption	
Host macros	Inherited and host macros		
Macro	Effective value	Template value	Global value (configure)
{\$SNMP_COMMUNITY}	NewCommunityName	T Remove	= "public"



SNMP version and settings are defined on the host interface level

Name:

✓ Short description of the SNMP metric.

Key:

✓ free form

✓ must be unique on the host/template

The screenshot shows a 'Create New Item' dialog box with the following fields filled in:

- * Name: New SNMP item
- Type: SNMP agent
- * Key: create.your.own.key
- * Host interface: training.lan : 161
- * SNMP OID: .1.3.6.1.2.1.1.1.0

To get the metric, provide a correct OID in numerical or textual format.

✓ Make sure that the "Type of information" and other parameters match your metric.

✓ Testing and "Execute now" work for all kinds of SNMP items (passive checks).

To get the CLI SNMP utilities, install the "net-snmp-utils" package:

的心 snmpget

- Retrieves a single value from SNMP agent

```
$ snmpget -c public -v2c 10.0.0.127 1.3.6.1.2.1.1.3.0  
SNMPv2-MIB::sysUpTime.0 = Timeticks: (1536925142) 14 days, 20:11:35.95
```

的心 snmpwalk

- Retrieves multiple OIDs and values
- Output format can be specified by -On flag

```
$ snmpwalk -c public -v2c 10.0.0.127 .1  
SNMPv2-MIB::sysDescr.0 = HP-UX net-snmp B.10.20 A 9000/715  
SNMPv2-MIB::sysObjectID.0 = OID: enterprises.ucdavis.ucdSnmpAgent.hpux10  
SNMPv2-MIB::sysUpTime.0 = Timeticks: 1536925142) 14 days, 20:11:35.95
```

```
$ snmpwalk -c public -v2c -On 10.0.0.127 .1  
.1.3.6.1.2.1.1.1.0 = HP-UX net-snmp B.10.20 A 9000/715  
.1.3.6.1.2.1.1.2.0 = OID: enterprises.ucdavis.ucdSnmpAgent.hpux10  
.1.3.6.1.2.1.1.3.0 = Timeticks: 1536925142) 14 days, 20:11:35.95
```

Common reasons, why SNMP requests may not work:

- 的心 Wrong credentials (community or username/password)
- 的心 UDP port 161 is closed by a local or remote firewall
- 的心 Zabbix server is not in the ACL whitelist on the remote SNMP device
- 的心 Timeout is too short for Zabbix server or proxy
- 的心 Requested OID is not known by the monitored device

SNMP timeout message does not always mean a communication timeout

If textual MIB syntax is used in SNMP items, the MIB files must be installed on Zabbix server and all proxies used for SNMP monitoring



SNMP Traps

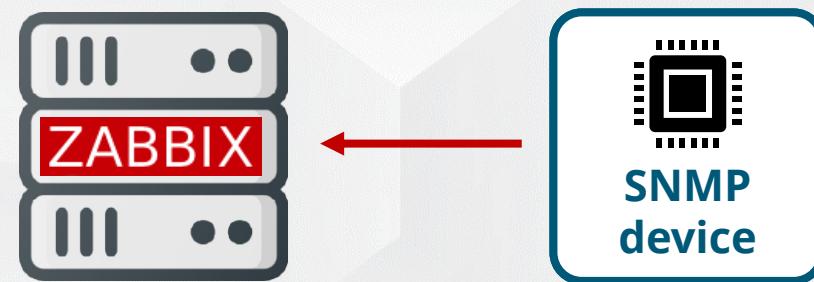
The information is sent from an SNMP-enabled device and collected by Zabbix

~ Receiving SNMP traps in Zabbix is designed to work with snmptrapd

- snmptrapd listens on UDP/162 port

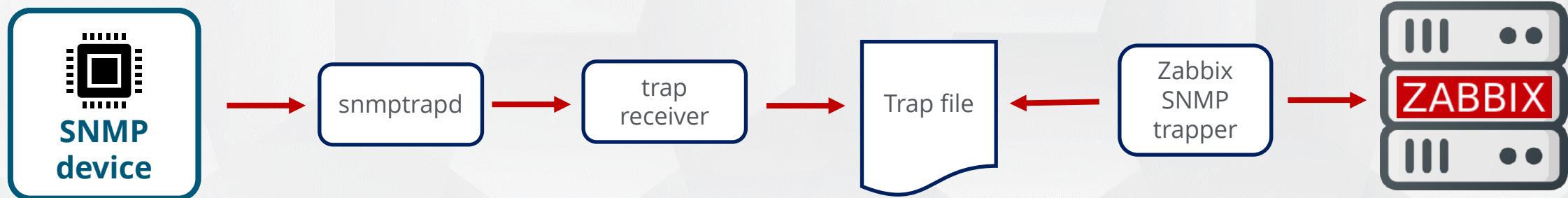
~ Usually, traps are sent upon certain condition change:

- the temperature is high/low
- the interface is down/up
- an administrative login to the device



The workflow for receiving an SNMP trap:

- ~ snmptrapd receives a trap and passes the trap to the receiver
- ~ Trap receiver parses, formats and writes the trap to a file
- ~ Zabbix SNMP trapper process reads and parses the trap file
- ~ Zabbix checks all SNMP trap items with SNMP interface matching the trap address
 - A trap is compared to an expression in snmptrap[regexp] items . If matches, the value gets stored in the item
 - If a matching item is not found and there is "snmptrap.fallback" item, the trap is stored there
- ~ If a trap has not been set as the value of any item, the unmatched trap will be logged in the Zabbix log file



Common trap receivers:

✓ zabbix_trap_receiver.pl (Perl script)

✓ SNMPPTT

✓ other (e.g. snmptrapfmt)

	Perl script	SNMPPTT
MIBs	Not required	Required
Trap formatting	Script	Configuration file
Trap matching	snmptrap["PCRE"]	snmptrap["PCRE"]
Unknown traps	snmptrap.fallback	Configuration file
Accept or reject trap	No	Yes
Search and replace	No	Yes

vs

To read traps:

- ~ Zabbix server must be configured to start the SNMP trapper process
- ~ Point to the trap file (must be the same as in zabbix_trap_receiver.pl)

```
# vi ./zabbix_server.conf
StartSNMPTrapper=1
SNMPTrapperFile=/tmp/zabbix_traps.tmp
```

Use MIB files to provide trap OIDs in a human-readable format:

- ~ place your MIBs into usr/local/share/snmp/mibs
- ~ configure "snmptrapd" to import required MIBs (or all)

```
# vi /etc/snmp/snmp.conf
mibs +JUNIPER-MIB:JUNIPER-FABRIC-CHASSIS:BGP4-MIB
```

Restart Zabbix server and snmptrapd processes to apply the changes

Zabbix does not provide any log rotation system. Use logrotate daemon to rotate the trap file. Example:

1. Send a test SNMP trap:

```
# snmptrap -v 1 -c Public12 127.0.0.1 '.1.3.6.1.6.3.1.1.5.4' '0.0.0.0' 6 33 '55' \
    .1.3.6.1.6.3.1.1.5.4 s "eth0"
```

2. Check that the trap is received in the "/tmp/zabbix_traps.tmp".

3. Configure the items and triggers.



https://zabbix.org/wiki/Start_with_SNMP_traps_in_Zabbix

Example: a trap is created by the Perl script by default

```
18:58:38 2018/02/26 ZBXTRAP 127.0.0.1
PDU INFO:
notificationtype      TRAP
version               0
receivedfrom          UDP: [127.0.0.1]:40780->[127.0.0.1]
errorstatus            0
messageid              0
community             public
transactionid          7
errorindex              0
requestid              0
VARBINDS:
DISMAN-EVENT-MIB::sysUpTimeInstance type=67 value=Timeticks: (55) 0:00:00.55
SNMPv2-MIB::snmpTrapOID.0    type=6 value=OID: IF-MIB::linkUp.0.33
IF-MIB::linkUp type=4 value=STRING: "eth0"  SNMP-COMMUNITY
MIB::snmpTrapCommunity.0 type=4 value=STRING: "public"
SNMPv2-MIB::snmpTrapEnterprise.0 type=6 value=OID: IF-MIB::linkUp
```

Syntax:

~snmptrap[regex]

- Catches all SNMP traps that match the regular expression specified in the regex
- Any part of the trap can be used as a regex

~snmptrap.fallback

- Catches all SNMP traps that were not caught by any of the "snmptrap[]" items

Item examples:

~snmptrap[LineVoltageProblem]

~snmptrap["IF-MIB::(linkDown | linkUp)"]

Trigger examples:

~{Template:snmptrap.fallback.nodata(10m)}=0

- Zabbix will give a "signal" that some SNMP trap items are missing on the host

~{Template:snmptrap["ShutdownNotification"].strlen()}>0

- Problem state if a trap is received + Manual closing of problems

PRACTICAL SETUP

- 1) Create a new template:
 - ~ Name: Template Basic SNMP
 - ~ Host group: Training/Templates
- 2) Create a new SNMP item on "Template Basic SNMP":
 - ~ Name: System description
 - ~ OID: 1.3.6.1.2.1.1.1.0
- 3) Add an SNMP interface to the Training Resources host:
 - ~ Community: training
 - ~ Version: 2c
 - ~ DNS: training.lan Port: 161
- 4) Link "Template Basic SNMP" to the Training Resources host
- 5) Make sure that the item receives data
- 6) Add a preprocessing step Discard unchanged to the item



Advanced task: Find SNMP OID for incoming ICMP packets and create an item to monitor



Log file monitoring

Zabbix can be used for centralized monitoring and analysis of log files

- Zabbix agent (active) must be running on a host
- Filter content using REGEX by certain strings or string patterns

⚠ Settings in zabbix_agentd.conf related to log items:

- MaxLinesPerSecond – configurable per agent and item (default: 20 lines).
 - Maximum number of new lines the agent will send per second to Zabbix server or proxy
 - Provided value will be overridden by the parameter 'maxlines' in the item key

⚠ Settings in the item configuration:

- Type of information: Log
- Log time format: optionally specify a pattern for parsing the log line timestamp
 - y, M, d, h, m, s – everything else works as a placeholder
 - Numeric values only accepted

⚠ Global regular expressions can be used in the 'regexp' parameter prefixed with @

- Example: "@Apache errors_log monitoring"



https://www.zabbix.com/documentation/current/manual/config/items/itemtypes/log_items

Get lines from a regular log file:

Item parameter	Definition
Type	Zabbix agent (active)
Key	<code>log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>]</code>
Value type	Log
Update interval	usually 1 sec

key parameter	Description
<code>file</code>	Full path and name of a log file
<code><regexp></code>	Regular expression describing required pattern
<code><encoding></code>	Code page identifier
<code><maxlines></code>	Overrides 'MaxLinesPerSecond' in zabbix_agentd.conf
<code><mode></code>	<code>all</code> (default), <code>skip</code> - skip processing of older data (affects only newly created items)
<code><output></code>	If not specified: <code>\0</code> (all) or capture groups from the <code><regexp></code> - <code>\1 \2</code> , etc.
<code><maxdelay></code>	Maximum delay in seconds.
<code><options></code>	Deprecated since 5.0.2, modification time change is ignored.

Example:

`~log["/var/log/httpd/error_log","error"]`

Log rotation support:

- ⚠ "file" becomes a regular expression (not a path)
 - Directory regular expression matching is not supported
- ⚠ More resource intensive: agent must re-read the directory content with each check

Item parameter	Definition
Type	Zabbix agent (active)
Key	<code>logrt[file_regex,<regexp>,<encoding>,<maxlines>,<mode>,<output>,<maxdelay>,<options>]</code>
Value type	Log
Update interval	usually 1 sec

Examples:

Item key	Description
<code>logrt["/home/zabbix/logs/^logfile[0-9]{1,3}\$","",100]</code>	will match file like "logfile1" (will not match ".logfile1")
<code>logrt["/home/user/^logfile_.*_[0-9]{1,3}\$","","UTF-8",100]</code>	will collect data from files such "logfile_abc_1" or "logfile_001"

Items: log.count[...] and logrt.count[...]

- ✓ Save server resources
- ✓ Count matched lines in a log file

Parameter	Definition
Type	Zabbix agent (active)
Key	<code>log.count[file,<regexp>,<encoding>,<maxproclines>,<mode>,<maxdelay>,<options>]</code>
Value type	Log
Update interval	usually 1 sec

Benefits:

- ✓ Processing is done on the agent side using resources of the monitored host
- ✓ Saves network traffic
- ✓ Saves server resources (CPU, DB space, etc.)

Event log monitoring can be used with Zabbix Windows agent only

Parameter	Definition
Type	Zabbix agent (active)
Key	eventlog[name,<regexp>,<severity>,<source>,<eventid>,<maxlines>,<mode>]
Value type	Log
Update interval	usually 1 sec

Examples:

- ~ eventlog[System,"Warning | Error",,,,skip]
- ~ eventlog[Security,"Failure Audit",,,^(529 | 680)\$]

Special history view:

- ✓ Mark selected/other
- ✓ Hide/Show selected
- ✓ Add multiple log items, sorted by entry data

received by server written in log

Timestamp	Local time	Value
2020-05-26 17:39:58	2020-05-26 17:39:57	19197:20200526:173957.960 failed to accept an incoming connection: connection from "165.22.23.79" rejected, allowed hosts: "127.0.0.1,trainer,student-01,student-02,student-03,student-04,student-05,student-06,student-07,student-08,student-09"
2020-05-26 17:35:09	2020-05-26 17:30:34	19197:20200526:173034.129 agent #2 started [listener #1]
2020-05-26 17:35:09	2020-05-26 17:30:34	19200:20200526:173034.126 agent #5 started [active checks #1]
2020-05-26 17:35:09	2020-05-26 17:30:34	19196:20200526:173034.125 agent #1 started [collector]
2020-05-26 17:35:09	2020-05-26 17:30:34	19199:20200526:173034.119 agent #4 started [listener #3]
2020-05-26 17:35:09	2020-05-26 17:30:34	19198:20200526:173034.117 agent #3 started [listener #2]

Items list **Production server: Zabbix Agent log**

Value

Selected

There is an error in the log:

~{host:log["/var/log/httpd/error_log"].str(ERROR)}=1

There are several errors in the log for last 3 minutes:

~{host:log["/var/log/httpd/error_log",ERROR].count(3m,ERROR,like)}>2

Problem if error is received and automatically returns to OK state in 5 minutes:

~{host:log["/var/log/httpd/error_log",ERROR].nodata(5m)}=0



Don't use nodata() function with "Multiple problem generation" mode for triggers

⚠ The agent starts reading a log file from the point where it previously stopped

- The server keeps size and time counters in a database
- For logrt[...] two additional counters are used
- If log file becomes smaller than the log size counter – the counter is reset to zero and the agent starts reading the log file from the beginning

⚠ Agent processes new records of a log file once per "Update interval" seconds

- Recommended update interval is 1s

⚠ Restoring or replacing files with older versions may lead to log being analyzed from the beginning and duplicated alerts



Advanced log file Monitoring

To save Zabbix server resources and react only on the core problems:

~ Filtering of log lines with a regular expressions is possible:

```
log[file,<regexp>,<encoding>,<maxlines>,<mode>,<output>]
```

~ Capturing groups from the regex are specified in the output using \1 \2 etc

~ Saves database space by storing only the necessary information

~ Log lines are processed by an active agent, which saves network traffic and Zabbix server's CPU

~ Recommended:

- Use output parameter in log and logrt items to extract the desired number
- Use Numeric type of information to see graphs and create triggers easily

Example:

```
log[/var/log/syslog,"Total processors activated: ([0-9]+)",,,,\1]
```



[.../log_items#extracting_matching_part_of_regular_expression](#)

Examples:

Logging 55 message 33

Item key	Output
log[path,([0-9]+) message ([0-9]+),,\1]	55
log[path,([0-9]+) message ([0-9]+),,\1 and \2]	55 and 33
log[path,([0-9]+) message ([0-9]+),,we got \1 and \2]	we got 55 and 33

```
Fr Feb 07 2014 11:07:36.6690 */ Thread Id 1400 (GLEWF) large result buffer allocation - /Length:  
437136/Entries: 5948/User: AUser/Form:CFG:ServiceLevelAgreement
```

Item key	Output
log[/path/to/the/file,large result buffer allocation.*Entries: ([0-9]+),,\1]	5948

```
30289:20200511:145609.891 failed to accept an incoming connection: from 11.22.33.44: TLS connection  
has been closed during handshake:
```

Item key	Output
log["/var/log/zabbix/zabbix_agentd.log",":(\d{8}):\d{6}).*connection: from \"(\d+\.\d+\.\d+\.\d+)\\"",,\1 \2]	20200511:145609 11.22.33.44

Macro functions are used to extract the information from item values

~ Syntax: {<macro>.<func>(<params>)}

- Case sensitive: regsub (<pattern>,<output>)
- Case insensitive: iregsub (<pattern>,<output>)

~ Used in triggers, tags, web scenarios (check the documentation)

~ If a wrong regular expression is used - the macro evaluates to 'UNKNOWN'

Example

MySQL crashed errno 4056

Item key	Output
<code>{ITEM.VALUE}.regsub("^([a-zA-Z]+)*errno\s+([0-9]+)", "Problem ID: \1_\2 ")</code>	Problem ID: MySQL_4056

group 1 group 2



https://www.zabbix.com/documentation/5.0/manual/config/macros/macro_functions

PRACTICAL SETUP

1. Create a new template:

✓ Name: Template Basic active

✓ Group: Training/Templates

✓ Template may be already created by completing one of the previous advanced tasks !

2. Create a new item on “Template Basic active”:

✓ Name: Zabbix agent log - rejected server connections

✓ Use file: /var/log/zabbix/zabbix_agentd.log

✓ Filter: "...failed to accept..." lines

3. Create a new trigger on “Template Basic active”:

✓ Name: Rejected server connection on {MACRO} from {MACRO}

✓ Mode: Multiple problem generation

✓ Tag: extract the IP address from a log using regsub() function

4. Make sure that the item receives data and the trigger works

✓ Disable the trigger to avoid generating a lot of problems



Advanced task: Create new item to extract only timestamp and IP address from log file



Q&A
Don't use Zabbix as SYSLOG server!!!
Why?



WEB Monitoring

Zabbix can check several availability aspects of websites.

~ Checks are performed by Zabbix server/proxy

~ Zabbix agent is not required

~ Complex scenarios are supported:

- Multiple steps
- Data posting
- Logging in / out

~ Performance monitoring of Web applications:

- Response time
- Download speed per second

~ Availability monitoring of Web applications:

- Response code
- Availability

~ Templates can be used to monitor WEB scenarios on multiple hosts

- Use {HOST.CONN} built-in macro in the URL field



https://www.zabbix.com/documentation/5.0/manual/web_monitoring

A simple example:

Scenario "Our Intranet"

Step 1	First page returns code 200 and contains a copyright string
Step 2	Log in returns code 200 and contains a string that is visible only when logged in
Step 3	A post to forum returns code 200 and contains a string, informing about successful post
Step 4	Log out returns code 200 and checks for a unique string

! If a check fails at any step, server will not proceed to the next

Creating/configuring Web scenario:

- ~ Unique scenario name
- ~ Application
- ~ Update interval
- ~ Attempts
- ~ HTTP proxy
- ~ Variables that may be used in steps
- ~ Agent - Browser emulation
 - Can be Zabbix, Chrome, Firefox, Safari, etc.
 - Useful when a website returns different content for different browsers.
- ~ Custom HTTP headers that will be sent when performing a request
 - (example: Content-Type=application/xml; charset=utf-8)

The screenshot shows the Zabbix Web Monitoring interface for creating a new scenario. The 'Scenario' tab is active. The 'Name' field is filled with 'Online Banking availability'. In the 'Application' section, a new application named 'Online Banking' is being added. The 'Update interval' is set to '1m', and 'Attempts' are set to '2'. The 'Agent' is chosen as 'Zabbix'. The 'HTTP proxy' field contains the URL 'http://[user[:password]@]proxy.example.com[:port]'. The 'Variables' section lists two variables: '{user}' with value 'zabbix' and '{password}' with value 'Z1nk#dna'. The 'Headers' section shows a single header entry: 'name' with value 'value'. The 'Enabled' checkbox is checked. At the bottom are 'Add' and 'Cancel' buttons.

Creating/configuring steps:

- ~ Unique step name
- ~ URL to retrieve data
- ~ HTTP GET variables
- ~ Post type and data
- ~ Variables
- ~ Headers
- ~ Follow redirects
- ~ Timeout
- ~ Required string
- ~ Status codes
- ~ Cookies preserved inside one scenario

Step of web scenario

* Name	Open main page	
* URL	https://www.bank.com/eBankingWeb/login?Lang=en	Parse
Query fields		
Name	Value	
name	= value	
Add	Remove	
Post type		
Form data	Raw data	
Post fields		
Name	Value	
Username	= {user}	
Password	= {password}	
Enter	= Continue	
Add	Remove	
Variables		
Name	Value	
{sid}	= regex:name="sid" value="([0-9a-z]{16})"	
Add	Remove	
Headers		
Name	Value	
name	= value	
Add	Remove	
Follow redirects		
<input checked="" type="checkbox"/>		
Retrieve only headers		
<input type="checkbox"/>		
* Timeout	15s	
Required string		
Banking		
Required status codes		
200		

Configuring Authentication:

HTTP Authentication

- None
- Basic
- NTLM
- Kerberos

SSL verify peer

- Certificate is valid – trusted by a known certificate authority, not expired, etc.
- Specified in zabbix_server.conf SSLCAlocation=

SSL verify host

- The server name matches the name in the certificate

SSL certificate file

- Specified in zabbix_server.conf SSLCertLocation=

SSL key file

- Specified in zabbix_server.conf SSLKeyLocation=

SSL key password

Scenario	Steps	Authentication
		HTTP authentication <input type="button" value="NTLM"/>
		User <input type="text" value="{\$NTLM.USER}"/>
		Password <input type="text" value="{\$NTLM.PASSWORD}"/>
		SSL verify peer <input checked="" type="checkbox"/>
		SSL verify host <input checked="" type="checkbox"/>
		SSL certificate file <input type="text" value="{\$SSL.CERTIFICATE.FILE}"/>
		SSL key file <input type="text" value="{\$SSL.KEY.FILE}"/>
		SSL key password <input type="text" value="{\$SSL.KEY.PASSWORD}"/>
		<input type="button" value="Add"/> <input type="button" value="Cancel"/>

The section Monitoring > Hosts > Web contains:

~ Per scenario statistics

~ Per step statistics

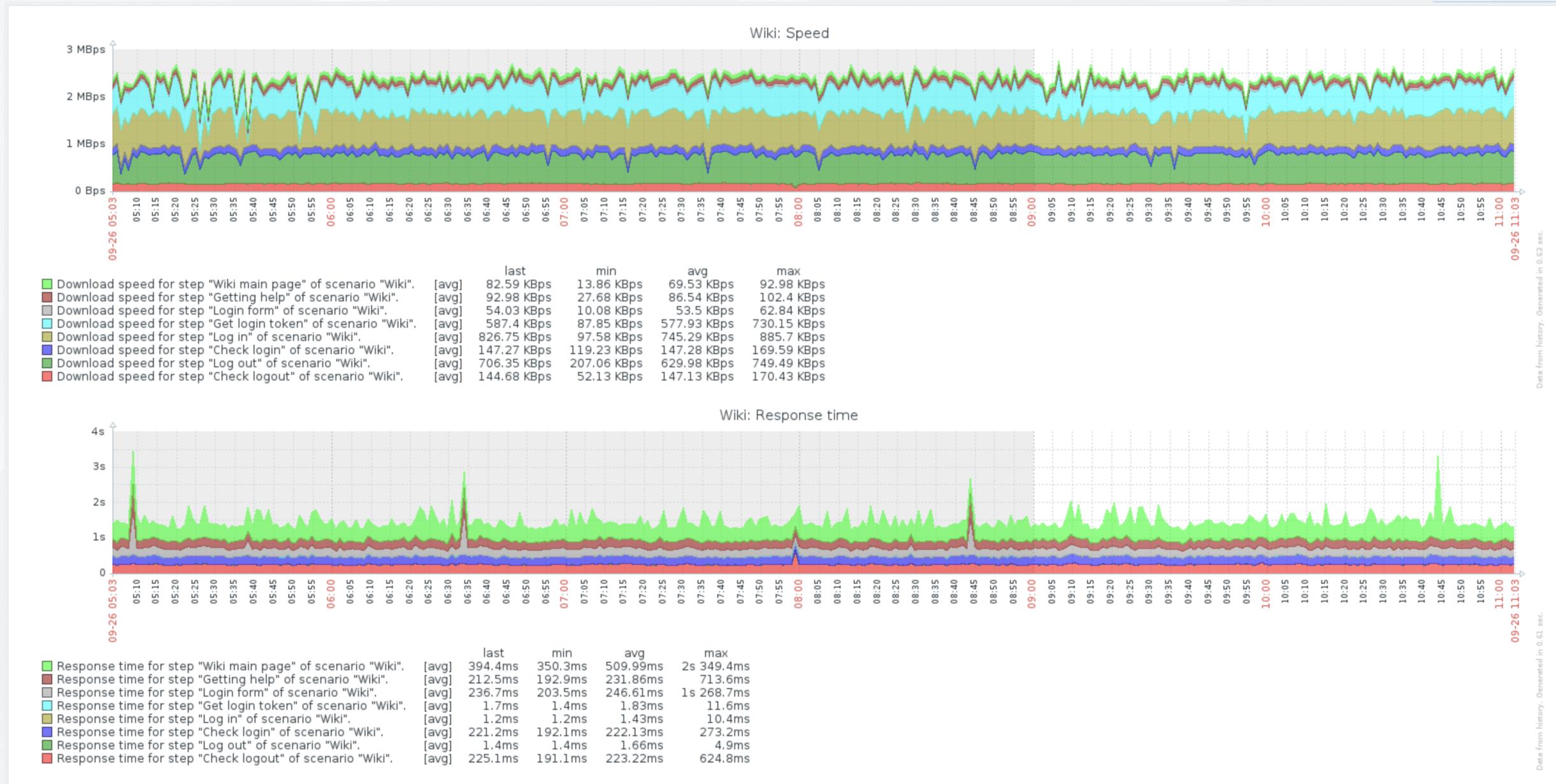
~ Pre-built graphs on:

- Speed
- Response times

Details of web scenario: Wiki



Step	Speed	Response time	Response code	Status
Wiki main page	84.85 KBps	383.9ms	200	OK
Getting help	95.27 KBps	207.4ms	200	OK
Login form	58.25 KBps	219.5ms	200	OK
Get login token	639.79 KBps	1.6ms	404	OK
Log in	800.12 KBps	1.3ms	404	OK
Check login	163 KBps	199.8ms	200	OK
Log out	719.83 KBps	1.4ms	404	OK
Check logout	154.74 KBps	210.5ms	200	OK
TOTAL		1s 225.4ms		OK



Web scenario items:

- ~ Show in the Latest data
- ~ Invisible in configuration
- ~ Work as normal items

Scenario level:

- ~ Download speed
 - web.test.in[Scenario,,bps]
- ~ Failed step (0 if none)
 - web.test.fail[Scenario]
- ~ Error message
 - web.test.error[Scenario]

Results can be used for:

- ~ Triggers, notifications, custom graphing

Step level:

- ~ Download speed
 - web.test.in[Scenario,Step,bps]
- ~ Response time
 - web.test.time[Scenario,Step]
- ~ Response code
 - web.test.rspcode[Scenario,Step]

Host	Name	Last check ▾	Last value	Change	
Zabbix.org	MediaWiki (27 Items)				
	Download speed for scenario "Wiki".	2018-09-26 17:03:25	321.87 KBps	-17.61 KBps	Graph
	Download speed for step "Check logout" of scenario "Wiki".	2018-09-26 17:03:25	150.97 KBps	-3.77 KBps	Graph
	Failed step of scenario "Wiki".	2018-09-26 17:03:25	0		Graph
	Response code for step "Check logout" of scenario "Wiki".	2018-09-26 17:03:25	200		Graph

No processing of JavaScript.

⚠ Session IDs are generated by JavaScript in some applications.

No IF-ELSE scenarios.

Hardcoded: 30 days history, 90 days trends

⚠ Web items are not visible in the host configuration page, defaults are used

Trigger examples:

⚠ {host:web.test.fail[Scenario].last()}><0

⚠ {host:web.test.time[Scenario>Login,resp].percentile(5m,,95)}>3

PRACTICAL SETUP

1. Create a new Zabbix Super Admin user for frontend monitoring:

- ~ Name: webcheck
- ~ Password: superAdm1n!

2. On "Template Basic"

- ~ Create a new web scenario to monitor your Zabbix frontend.
- ~ Add five steps :
 - first page
 - log in
 - check login
 - logout
 - check logout
- ~ Use a macro in URLs to get the IP of the frontend.
- ~ Use macros in variables for the username and password.



QUESTIONS?



Time for a break :)