1 Server

Overview

Zabbix server is the central process of Zabbix software.

The server performs the polling and trapping of data, it calculates triggers, sends notifications to users. It is the central component to which Zabbix agents and proxies report data on availability and integrity of systems. The server can itself remotely check networked services (such as web servers and mail servers) using simple service checks.

The server is the central repository in which all configuration, statistical and operational data is stored, and it is the entity in Zabbix that will actively alert administrators when problems arise in any of the monitored systems.

The functioning of a basic Zabbix server is broken into three distinct components; they are: Zabbix server, web frontend and database storage.

All of the configuration information for Zabbix is stored in the database, which both the server and the web frontend interact with. For example, when you create a new item using the web frontend (or API) it is added to the items table in the database. Then, about once a minute Zabbix server will query the items table for a list of the items which are active that is then stored in a cache within the Zabbix server. This is why it can take up to two minutes for any changes made in Zabbix frontend to show up in the latest data section.

Running server

If installed as package

Zabbix server runs as a daemon process. The server can be started by executing:

```
shell> service zabbix-server start
```

This will work on most of GNU/Linux systems. On other systems you may need to run:

```
shell> /etc/init.d/zabbix-server start
```

Similarly, for stopping/restarting/viewing status, use the following commands:

```
shell> service zabbix-server stop
shell> service zabbix-server restart
shell> service zabbix-server status
```

Start up manually

If the above does not work you have to start it manually. Find the path to the zabbix_server binary

and execute:

```
shell> zabbix_server
```

You can use the following command line parameters with Zabbix server:

```
-c --config <file>              path to the configuration file (default is
/usr/local/etc/zabbix_server.conf)
-f --foreground                 run Zabbix server in foreground
-R --runtime-control <option>   perform administrative functions
-h --help                       give this help
-V --version                    display version number
```

Runtime control is not supported on OpenBSD and NetBSD.

Examples of running Zabbix server with command line parameters:

```
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf
shell> zabbix_server --help
shell> zabbix_server -V
```

Runtime control

Runtime control options:

Option	Description	Target
config_cache_reload	Reload configuration cache. Ignored if cache is being currently loaded.	
housekeeper_execute	Start the housekeeping procedure. Ignored if the housekeeping procedure is currently in progress.	
log_level_increase[=<**target**>]	Increase log level, affects all processes if target is not specified.	**process type** - All processes of specified type (e.g., poller) See all server process types. **process type,N** - Process type and number (e.g., poller,3) **pid** - Process identifier (1 to 65535). For larger values specify target as 'process type,N'.
log_level_decrease[=<**target**>]	Decrease log level, affects all processes if target is not specified.	

Example of using runtime control to reload the server configuration cache:

```
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R
config_cache_reload
```

Example of using runtime control to trigger execution of housekeeper:

```
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R
housekeeper_execute
```

Examples of using runtime control to change log level:

```
Increase log level of all processes:
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R
log_level_increase

Increase log level of second poller process:
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R
log_level_increase=poller,2

Increase log level of process with PID 1234:
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R
log_level_increase=1234

Decrease log level of all http poller processes:
shell> zabbix_server -c /usr/local/etc/zabbix_server.conf -R
log_level_decrease="http poller"
```

Process user

Zabbix server is designed to run as a non-root user. It will run as whatever non-root user it is started as. So you can run server as any non-root user without any issues.

If you will try to run it as 'root', it will switch to a hardcoded 'zabbix' user, which must be present on your system. You can only run server as 'root' if you modify the 'AllowRoot' parameter in the server configuration file accordingly.

If Zabbix server and agent are run on the same machine it is recommended to use a different user for running the server than for running the agent. Otherwise, if both are run as the same user, the agent can access the server configuration file and any Admin level user in Zabbix can quite easily retrieve, for example, the database password.

Configuration file

See the configuration file options for details on configuring zabbix_server.

Start-up scripts

The scripts are used to automatically start/stop Zabbix processes during system's start-up/shutdown.

The scripts are located under directory misc/init.d.

Server process types

- `alert manager` - manager of alerter tasks
- `alerter` - process for sending notifications
- `configuration syncer` - process for managing in-memory cache of configuration data
- `discoverer` - process for discovery of devices
- `escalator` - process for escalation of actions
- `history syncer` - history DB writer
- `housekeeper` - process for removal of old historical data
- `http poller` - web monitoring poller
- `icmp pinger` - poller for icmpping checks
- `ipmi manager` - IPMI poller manager
- `ipmi poller` - poller for IPMI checks
- `java poller` - poller for Java checks
- `poller` - normal poller for passive checks
- `preprocessing manager` - manager of preprocessing tasks
- `preprocessing worker` - process for data preprocessing
- `proxy poller` - poller for passive proxies
- `self-monitoring` - process for collecting internal server statistics
- `snmp trapper` - trapper for SNMP traps
- `task manager` - process for remote execution of tasks requested by other components (e.g. close problem, acknowledge problem, check item value now, remote command functionality)
- `timer` - timer for processing maintenances
- `trapper` - trapper for active checks, traps, proxy communication
- `unreachable poller` - poller for unreachable devices
- `vmware collector` - VMware data collector responsible for data gathering from VMware services

The server log file can be used to observe these process types.

Various types of Zabbix server processes can be monitored using the **zabbix[process,<type>,<mode>,<state>]** internal item.

Supported platforms

Due to the security requirements and mission-critical nature of server operation, UNIX is the only operating system that can consistently deliver the necessary performance, fault tolerance and resilience. Zabbix operates on market leading versions.

Zabbix server is tested on the following platforms:

- Linux
- Solaris
- AIX
- HP-UX
- Mac OS X
- FreeBSD

- OpenBSD
- NetBSD
- SCO Open Server
- Tru64/OSF1

Zabbix may work on other Unix-like operating systems as well.

Locale

Note that the server requires a UTF-8 locale so that some textual items can be interpreted correctly. Most modern Unix-like systems have a UTF-8 locale as default, however, there are some systems where that may need to be set specifically.