Zabbix 7 Monitoring

Enterprise Monitoring With Zabbix

Patrik Uytterhoeven

Table of contents

1. What is this book about ?

Hi, welcome and thank you for your interest in my Zabbix book. I wrote the Zabbix cookbook and co wrote with Richards Zabbix 4 Network Monitoring a few years ago for PackPub.

The cookbook the first of it's kind probably outdated and will be replaced by the Zabbix 7 IT Infrastructure Monitoring Cookbook, written by Brian and Nathan, 2 people I like a lot to work with and can higly recommend. There are many more books available from Packt about Zabbix a complete overview can be found here Zabbix books at pack. Or if you like to find some non English books Amazon has some books form Packt and other Publishers in Chinese, Spanish and maybe some other languages as well. Other books

As Zabbix is an opensource product and making money out of the books was never my intention, it got me thinking how to do things different. How to make a new book without using a publisher like I had done before. After a while, I came up with the idea to make a book that would be free and that would be updated when new versions came out. Since I am a huge fan of documentation in markdown or asciidoc I came up with the idea to share the book in git and use markdown. The only problem left was how to make those markdown files readable in an easy way like a book ? After some searching trying to look for a good solution I found MkDocs. MkDocs is a Python-Markdown library that can convert everything to HTML and can be templated. So the problem was solved and a new book was born.

2. Who am I ?

My name is Patrik Uytterhoeven and I work for a Belgium company named Open-Future. I started at this company at Januari 2013 and that's when my journey started with Zabbix as well. They gave me the opportunity to build my experience and to get certified as Zabbix trainer. Since this year I am officially 10y Zabbix trainer. If you would like to follow one of my trainings feel free to register for a training at our website www.open-future.be. Why would you follow a training if you can read this book for free are you now thinking? Because trainings just like the book explain you all the details on how to set up and do things but also give you valueable tips and feedback that you never get from a book. Books just can't cover everything.

3. What OS do I need ?

Since I work mostly with RHEL based systems and since I am convinced that RHEL is the better choice in Production environments I have chosen to focus on using one of the forks that is available for free. Zabbix is supported on Ubuntu, Debian, Suse, Raspberry .... and it can be compiled on any OS that is Unix based so it's almost impossible to cover them all. However the book is Opensource and in GIT so feel free to contribute the code for your favorite flavour :). I will use Rocky Linux 9 in this book, but it should work for most of the other installations as well.

4. What version of Zabbix is used in this book ?

Since we are almost at the release of Zabbix 7, I will focus on version 7 since it will be the new LTS. It should also apply to most other versions but of course there will be minor changes. In the future, if there is enough support from the community to update this book together, it would be great if we could build a book for every LTS version available.

5. How to use this book ?

The book will try to cover all the topics, feel free to let me know if something is missing or feel free to make a pull request. There is no need to start from page 1 and read the book till the end. Some people will be looking for basic knowledge others might want to skip to the fun part, so I want the book to be useful for everyone. Therefor I will try to explain as best as possible in every topic the exact steps needed to reproduce.

There will be moments in the book where you need to type some code, I will show the commands you need to type in a box just like here.

```
# some command
```

Notes to some useful documentation will be added at the bottom of the page.

Here is a simple footnote[1]. With some additional text after it.

In case there is some important information to share I will add notes in the documentation like can be seen here :

Note

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Info

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Tip

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Question

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Warning

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Bug

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

Example

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla et euismod nulla. Curabitur feugiat, tortor non consequat finibus, justo purus auctor massa, nec semper lorem quam in massa.

1. My reference. ↩

6. Getting started

6.1 Requirements

Zabbix has a set of requirements that need to be met on the hardware level and software level. These requirements can change over time and also depends on the size of your setup and the software you choose. So before you start buying metal or installing a random database version have a look at the Zabbix documentation and check the latest requirements for the version you want to install. The latest requirements can be found here. Don't forget to select your correct *Zabbix* version from the list.

If you don't plan to run anything big just a small setup or a test setup Zabbix will run happy on a system with 2cpu and 8G ram. But all depends on how big your setup will be and how many items you will monitor, triggers you will create and for how long you want to keep that data. My advice in the days of Virtualization is you can start small and add more later.

For the setup you can choose to install all components on 1 server or every component on a different server. For the ease of use just make a few notes for yourself:.

server	ip
zabbix server	
database server	
web server	

Tip

While zabbix uses dashes "-" in it's names when we need to install packages like zabbix-get or zabbix-sender it's binaries use "_". like zabbix_sender or zabbix_server. This of course can vary depending if you use the packages from the original Zabbix repositories or not. Just be aaware that it's sometimes rather confusing and that if you installed somepackage with a dash that maybe the binary is with an underscore.

6.1.1 Basic OS configuration

firewall

It's important for our Zabbix server to have an OS that is well prepared before we start to install our monitoring tool. First we need to make sure our firewall is installed.

```
# dnf install firewalld --now
```

Our firewall is installed now, and we are ready to configure the needed ports. For our Zabbix server, we need to allow access to port 10051/tcp this is the port where our Zabbix trapper listens on for incoming data. So we need to open this port in our firewall to allow access to our Zabbix trapper.

```
# firewall-cmd --add-service=Zabbix-server --permanent
```

or if the service is not known

```
# firewall-cmd --add-port=10051/tcp --permanent
```

firewalld

"Firewalld is the replacement of iptables in Redhat and allows us to make changes available immediately without the need to restart a service. It's possible that your distribution is not using Firewalld in this case you have to look to the documentation of your OS."

timeserver

Another thing we need to configure is the setup of timeserver and sync our Zabbix server to the timeserver by making use of an ntp client. This needs to be done for the Zabbix server but also for the devices we will monitor as time is very important for Zabbix. Imagine one of our hosts having a time zone that is wrong we could end up looking for a problem in Zabbix that happened 6h ago while it had happened maybe only 2h ago.

```
# dnf install chronyd --now
```

Chrony should be installed now and enabled and running. This can be verified with the command:

```
# systemctl status chronyd
```

dnf

"dnf is a packagemanager from RedHat you need to replace dnf with your correct packagemanager like zyper, apt, yum, ... chrony is a replacement for ntpd and does a better job being faster and more accurate. If your OS does not support chrony then maybe ntpd is still available."

Once Chrony is installed we also need to setup our correct time zone. We can have a look first with 'timedatectl' to see how our time is configured

```
# timedatectl
               Local time: Thu 2023-11-16 15:09:14 UTC
           Universal time: Thu 2023-11-16 15:09:14 UTC
                 RTC time: Thu 2023-11-16 15:09:15
                Time zone: UTC (UTC, +0000)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Make sure that the service cronyd is active, see above on how to do if you missed it. We can choose the correct time zone from a list that we can lookup with the following command:

```
# timedatectl list-time zones
```

This will give us a list with all available time zones. Choose the one closest to you.

```
Africa/Abidjan
Africa/Accra

...

Pacific/Tongatapu
Pacific/Wake
Pacific/Wallis
UTC
```

We can now configure our correct time zone with the following command:

```
timedatectl set-time zone Europe/Brussels
```

When we look again we should see our time zone properly configured.

```
# timedatectl
               Local time: Thu 2023-11-16 16:13:35 CET
           Universal time: Thu 2023-11-16 15:13:35 UTC
                 RTC time: Thu 2023-11-16 15:13:36
                Time zone: Europe/Brussels (CET, +0100)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

Note

"Some people like to install all servers in the UTC time zone so that all server logs are in the same time zone when having servers all over the world. Zabbix supports user based time zone settings so it's possible to keep the time zone in UTC on the server and then add the correct time zone in the user interface if you like."

We can test if Chrony is syncronizing with the correct timeservers as well by running the command chronyc

```
# chronyc
chrony version 4.2
Copyright (C) 1997-2003, 2007, 2009-2021 Richard P. Curnow and others
chrony comes with ABSOLUTELY NO WARRANTY.  This is free software, and
you are welcome to redistribute it under certain conditions.  See the
GNU General Public License version 2 for details.

chronyc>
```

Then we type `sources`

```
chronyc> sources
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^- 51-15-20-83.rev.poneytel>     2   9   377   354   +429us[ +429us] +/-  342ms
^- 5.255.99.180                  2  10   377   620  +7424us[+7424us] +/-   37ms
^- hachi.paina.net               2  10   377   412   +445us[ +445us] +/-   39ms
^* leontp1.office.panq.nl        1  10   377   904  +6806ns[ +171us] +/- 2336us
```

Here we can see that we are using a bunch of ntp servers that are not in our own country so we better swicht to some timeservers in our local country or if we have a timeserver in our company we could use this one. We can find some local timeservers here : https://www.ntppool.org/

To change this we have to edit our config file "/etc/chrony.conf" and replace the existing ntp server with our local one

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
pool 2.centos.pool.ntp.org iburst
```

And change it to a local server:

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
pool be.pool.ntp.org iburst
```

Don't forget to restart the ntpd client of course.

```
# systemctl restart chronyd
```

When we look again we will see that we are now using our local timeservers.

```
chronyc> sources

MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^- ntp1.unix-solutions.be        2   6    17    43   -375us[ -676us] +/-   28ms
^* ntp.devrandom.be              2   6    17    43   -579us[ -880us] +/- 2877us
^+ time.cloudflare.com           3   6    17    43   +328us[  +27us] +/- 2620us
^+ time.cloudflare.com           3   6    17    43   +218us[  -83us] +/- 2815us
```

6.2 Installing Zabbix DB Server

6.2.1 Installing Zabbix with MariaDB

Let us start with the installation of the MariaDB server, you need to create a MariaDB repository configuration file `mariadb.repo` manually in the following path `/etc/yum.repos.d/`. To create a MariaDB repository file, you can use the following command.

Add the MariaDB repo

```
# vi /etc/yum.repos.d/mariadb.repo
```

The above command will create a new repository file, Once it is created, you need to add the following configuration into the file. Make sure your version, in this case 10.11, is supported by Zabbix by looking at the latest requirements for your version.

```
# MariaDB 10.11 RedHatEnterpriseLinux repository list - created 2023-11-01 14:20 UTC
# https://mariadb.org/download/
[mariadb]
name = MariaDB
# rpm.mariadb.org is a dynamic mirror if your preferred mirror goes offline. See https://mariadb.org/mirrorbits/ for details.
# baseurl = https://rpm.mariadb.org/10.11/rhel/$releasever/$basearch
baseurl = https://mirror.23m.com/mariadb/yum/10.11/rhel/$releasever/$basearch
# gpgkey = https://rpm.mariadb.org/RPM-GPG-KEY-MariaDB
gpgkey = https://mirror.23m.com/mariadb/yum/RPM-GPG-KEY-MariaDB
gpgcheck = 1
```

Lets update our OS first with the latest patches

```
# dnf update -y
```

Install the MariaDB database

Now we are ready to install our MariaDB database.

```
# dnf install MariaDB-server MariaDB-client
```

We are now ready to enable and start or MariaDB database.

```
# systemctl enable mariadb --now
```

Once the installation is complete, you can verify the version of the MariaDB server by using the following command:

```
# mysql -V
```

The output should look like this:

```
mysql  Ver 15.1 Distrib 10.11.6-MariaDB, for Linux (x86_64) using  EditLine wrapper
```

And when we ask the status of our MariaDB server we should get an output like this:

```
# systemctl status mariadb

● mariadb.service - MariaDB 10.11.6 database server
     Loaded: loaded (/usr/lib/systemd/system/mariadb.service; enabled; preset: disabled)
    Drop-In: /etc/systemd/system/mariadb.service.d
             └─migrated-from-my.cnf-settings.conf
     Active: active (running) since Sat 2023-11-18 19:19:36 CET; 2min 13s ago
       Docs: man:mariadbd(8)
             https://mariadb.com/kb/en/library/systemd/
    Process: 41986 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited, status=0/SUCCESS)
    Process: 41987 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR= ||   VAR=`cd /usr/bin/..; /usr/bin/galera_recovery`; [ $? -eq 0 ]   &&
systemctl set-environment _WSREP_START>
    Process: 42006 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=exited, status=0/SUCCESS)
   Main PID: 41995 (mariadbd)
     Status: "Taking your SQL requests now..."
      Tasks: 9 (limit: 12344)
     Memory: 206.8M
        CPU: 187ms
```

Securing the MariaDB database

It's time to secure our database by removing the test database and user and set our own root password. Run the command `mariadb-secure-installation`, you should get the following output.

```
# mariadb-secure-installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n] n
 ... skipping.

You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
```

Create the Zabbix database

```
# mysql -uroot -p
password

MariaDB [(none)]> CREATE DATABASE zabbix CHARACTER SET utf8mb4 COLLATE utf8mb4_bin;
MariaDB [(none)]> CREATE USER 'zabbix-web'@'<zabbix server ip>' IDENTIFIED BY '<password>';
MariaDB [(none)]> CREATE USER 'zabbix-srv'@'<zabbix server ip>' IDENTIFIED BY '<password>';
MariaDB [(none)]> GRANT ALL PRIVILEGES ON zabbix.* TO 'zabbix-srv'@'<zabbix server ip>';
```

```
MariaDB [(none)]> GRANT SELECT, UPDATE, DELETE, INSERT ON zabbix.* TO 'zabbix-web'@'<zabbix server ip>';
MariaDB [(none)]> SET GLOBAL log_bin_trust_function_creators = 1;
MariaDB [(none)]> QUIT
```

Warning

"The Zabbix documentation explicitly mentions that deterministic triggers need to be created during the import of schema. On MySQL and MariaDB, this requires GLOBAL log_bin_trust_function_creators = 1 to be set if binary logging is enabled and there is no superuser privileges and log_bin_trust_function_creators = 1 is not set in MySQL configuration file."

Add the Zabbix repository and populate the DB

```
# rpm -Uvh https://repo.zabbix.com/zabbix/6.5/rocky/9/x86_64/zabbix-release-6.5-2.el9.noarch.rpm
# dnf clean all
# dnf install zabbix-sql-scripts
```

Upload the data from zabbix (db structure, images, user, ...)

```
# zcat /usr/share/zabbix-sql-scripts/mysql/server.sql.gz | mysql --default-character-set=utf8mb4 -uroot -p zabbix
```

Warning

"Depending on the speed of your hardware or VM this can take a few seconds upto a few minutes so please don't cancel just sit and wait for the prompt."

Log back into your MariaDB Database as root

```
# mysql -uroot -p
```

Remove the global parameter again as its not needed anymore and also for security reasons.

```
MariaDB [(none)]> SET GLOBAL log_bin_trust_function_creators = 0;
Query OK, 0 rows affected (0.001 sec)
```

Configure the firewall

One last thing we need to do is open the firewall and allow incoming connections for the MariaDB database from our Zabbix server because at the moment we dont accept any connections yet.

```
# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3 enp0s8
  sources:
  services: cockpit dhcpv6-client  ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

First we will create an appropriate zone for our MariaDB and open port 3306/tcp but only for the ip from our Zabbix server.

```
# firewall-cmd --new-zone=mariadb-access --permanent
success

# firewall-cmd --reload
success

# firewall-cmd --get-zones
block dmz drop external home internal mariadb-access nm-shared public trusted work

# firewall-cmd --zone=mariadb-access --add-source=<zabbix-serverip> --permanent

success
```

```
# firewall-cmd --zone=mariadb-access --add-port=3306/tcp  --permanent

success
# firewall-cmd --reload
```

Now lets have a look to our firewall rules to see if they are what we expected:

```
# firewall-cmd --zone=mariadb-access --list-all
```

```
mariadb-access (active)
  target: default
  icmp-block-inversion: no
  interfaces:
  sources: <ip from zabbix-server>
  services:
  ports: 3306/tcp
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Our database server is ready now to accept connections from our Zabbix server :). You can continue with the next task Installing the Zabbix Server

6.2.2 Installing Zabbix with MySQL

Let us start with the installation of the MySQL server, you need to create a MySQL repository first so that we can install the proper files for our MySQL server It's alwqys best to check the Zabbix documentation to see what version is supported so you don't install a version that is not supported or is not supported anymore.

Add the MySQL repo

Run the following command to install the MySQL repo for version 8.0

```
# dnf -y install https://dev.mysql.com/get/mysql80-community-release-el9-1.noarch.rpm
```

> **Note**
>
> "If you install this on RedHat 8 and higher or alternatives like CentOS, Rocky or Alma 8 then you need to disable the mysql module by running 'module disable mysql'."

Let's update our OS first with the latest patches

```
# dnf update -y
```

INSTALLING THE MYSQL DATABASE

```
# dnf -y install mysql-community-server
```

We are now ready to enable and start or MySQL database.

```
# systemctl enable mysqld --now
```

Once the installation is complete, you can verify the version of the MySQL server by using the following command:

```
# mysql -V
```

The output should look like this:

```
mysql  Ver 8.0.35 for Linux on x86_64 (MySQL Community Server - GPL)
```

And when we ask the status of our MariaDB server we should get an output like this:

```
# systemctl status mysqld

● mysqld.service - MySQL Server
     Loaded: loaded (/usr/lib/systemd/system/mysqld.service; enabled; preset: disabled)
     Active: active (running) since Mon 2023-11-20 22:15:51 CET; 1min 15s ago
       Docs: man:mysqld(8)
             http://dev.mysql.com/doc/refman/en/using-systemd.html
```

```
     Process: 44947 ExecStartPre=/usr/bin/mysqld_pre_systemd (code=exited, status=0/SUCCESS)
    Main PID: 45012 (mysqld)
      Status: "Server is operational"
       Tasks: 37 (limit: 12344)
      Memory: 448.3M
         CPU: 4.073s
      CGroup: /system.slice/mysqld.service
              └─45012 /usr/sbin/mysqld

Nov 20 22:15:43 mysql-db systemd[1]: Starting MySQL Server...
Nov 20 22:15:51 mysql-db systemd[1]: Started MySQL Server.
```

Securing the MySQL database

MySQL will secure our database with a random root password that is generated when we install the database. First thing we need to do is replace it with our own password. To find what the password is we need to read the log file with the followin command:

`# grep 'temporary password' /var/log/mysqld.log`

Change the root password as soon as possible by logging in with the generated, temporary password and set a custom password for the superuser account:

```
# mysql -uroot -p
```

```
mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY '<my mysql password>';
mysql> quit
```

Next we can run the command mysql_secure_installation, you should get the following output:

> **Note**
>
> "There is no need to reset the root password for MySQL again as we have reset it already. The next step is optional but recommended."

```
# mysql_secure_installation

Securing the MySQL server deployment.

Enter password for user root:
The 'validate_password' component is installed on the server.
The subsequent steps will run with the existing configuration
of the component.
Using existing password for root.

Estimated strength of the password: 100
Change the password for root ? ((Press y|Y for Yes, any other key for No) : n

 ... skipping.
By default, a MySQL installation has an anonymous user,
allowing anyone to log into MySQL without having to have
a user account created for them. This is intended only for
testing, and to make the installation go a bit smoother.
You should remove them before moving into a production
environment.

Remove anonymous users? (Press y|Y for Yes, any other key for No) : y
Success.


Normally, root should only be allowed to connect from
'localhost'. This ensures that someone cannot guess at
the root password from the network.

Disallow root login remotely? (Press y|Y for Yes, any other key for No) : y
Success.

By default, MySQL comes with a database named 'test' that
anyone can access. This is also intended only for testing,
and should be removed before moving into a production
environment.


Remove test database and access to it? (Press y|Y for Yes, any other key for No) : y
 - Dropping test database...
Success.

 - Removing privileges on test database...
Success.

Reloading the privilege tables will ensure that all changes
made so far will take effect immediately.

Reload privilege tables now? (Press y|Y for Yes, any other key for No) : y
```

```
Success.

All done!
```

Let's create our DB users and the correct permissions in the database:

`mysql -uroot -p`

```
mysql> CREATE DATABASE zabbix CHARACTER SET utf8mb4 COLLATE utf8mb4_bin;
mysql> CREATE USER 'zabbix-web'@'<zabbix server ip>' IDENTIFIED BY '<password>';
mysql> CREATE USER 'zabbix-srv'@'<zabbix server ip>' IDENTIFIED BY '<password>';
mysql> GRANT ALL PRIVILEGES ON zabbix.* TO 'zabbix-srv'@'<zabbix server ip>';
mysql> GRANT SELECT, UPDATE, DELETE, INSERT ON zabbix.* TO 'zabbix-web'@'<zabbix server ip>';
mysql> SET GLOBAL log_bin_trust_function_creators = 1;
mysql> QUIT
```

Warning

"The Zabbix documentation explicitly mentions that deterministic triggers need to be created during the import of schema. On MySQL and MariaDB, this requires GLOBAL log_bin_trust_function_creators = 1 to be set if binary logging is enabled and there is no superuser privileges and log_bin_trust_function_creators = 1 is not set in MySQL configuration file."

Add the Zabbix repository and populate the DB

```
# rpm -Uvh https://repo.zabbix.com/zabbix/6.5/rocky/9/x86_64/zabbix-release-6.5-2.el9.noarch.rpm
# dnf clean all
# dnf install zabbix-sql-scripts
```

Now let;s upload the data from zabbix (db structure, images, user, ...)

```
# zcat /usr/share/zabbix-sql-scripts/mysql/server.sql.gz | mysql --default-character-set=utf8mb4 -uroot -p zabbix
Enter password:
```

Warning

"Depending on the speed of your hardware or VM this can take a few seconds upto a few minutes so please don't cancel just sit and wait for the prompt."

```
Log back into your MySQL Database as root

# mysql -uroot -p
```

Remove the global parameter again as its not needed anymore and also for security reasons.

```
mysql> SET GLOBAL log_bin_trust_function_creators = 0;
Query OK, 0 rows affected, 1 warning (0.00 sec)
```

Configure the firewall

One last thing we need to do is open the firewall and allow incoming connections from our Zabbix server to our MySQL database because at the moment we dont accept any connections yet.

```
# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3 enp0s8
  sources:
  services: cockpit dhcpv6-client  ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

First we will create an appropriate zone for our MySQL Database and open port 3306/tcp but only for the IP from our Zabbix server. This way no one unallowed is able to connect.

```
# firewall-cmd --new-zone=mysql-access --permanent
success

# firewall-cmd --reload
success

# firewall-cmd --get-zones
block dmz drop external home internal mysql-access nm-shared public trusted work

# firewall-cmd --zone=mysql-access --add-source=<zabbix-serverip> --permanent

success
# firewall-cmd --zone=mysql-access --add-port=3306/tcp  --permanent

success
# firewall-cmd --reload
```

Now lets have a look to our firewall rules to see if they are what we expected:

```
# firewall-cmd --list-all --zone=mysql-access
```

```
mysql-access (active)
  target: default
  icmp-block-inversion: no
  interfaces:
  sources: <ip from the zabbix-server>
  services:
  ports: 3306/tcp
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Our database server is ready now to accept connections from our Zabbix server :). You can continue with the next task Installing the Zabbix Server

6.2.3 Installing Zabbix with PostgreSQL

For our DB setup with PostgreSQL we need to add our PostgreSQL repository first to the system. As of writing PostgreSQL 13-16 are supported but best is to have a look before you install it as new versions may be supported and older maybe unsupported both by Zabbix and PostgreSQL. Usually it's a good idea to go with the latest version that is supported by Zabbix. Zabbix also supports the extension TimescaleDB this is someting we will talk later about. As you will see the setup from PostgreSQL is very different from MySQL not only the installation but also securing the DB.

The table of compatibility can be found here.

Add the PostgreSQL repo

So let us start first setting up our PostgreSQL repository with the folowing commands.

```
# Install the repository RPM:
sudo dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-9-x86_64/pgdg-redhat-repo-latest.noarch.rpm

# Disable the built-in PostgreSQL module:
sudo dnf -qy module disable postgresql

# Install PostgreSQL:
sudo dnf install -y postgresql16-server

# Initialize the database and enable automatic start:
sudo /usr/pgsql-16/bin/postgresql-16-setup initdb
sudo systemctl enable postgresql-16 --now
```

Securing the PostgreSQL database

As i told you PostgreSQL works a bit different then MySQL or MariaDB and this applies aswell to how we manage access permissions. Postgres works with a file with the name pg_hba.conf where we have to tell who can access our database from where and what encryption is used for the password. So let's edit this file to allow our frontend and zabbix server to access the database.

> **Note**
>
> "Client authentication is configured by a configuration file with the name `pg_hba.conf`. HBA here stands for host based authentication. For more information feel free to check the PostgreSQL documentation."

Add the following lines, the order here is important.

```
# vi /var/lib/pgsql/16/data/pg_hba.conf
```

```
# "local" is for Unix domain socket connections only
local   zabbix          zabbix-srv                              scram-sha-256
local   all             all                                     peer
# IPv4 local connections:
host    zabbix          zabbix-srv      <ip from zabbix server/24>  scram-sha-256
host    zabbix          zabbix-web      <ip from zabbix server/24>  scram-sha-256
host    all             all             127.0.0.1/32                scram-sha-256
```

After we changed the pg_hba file don't forget to restart postgres else the settings will not be applied. But before we restart let us also edit the file postgresql.conf and allow our database to listen on our network interface for incomming connections from the zabbix server. Postgresql will standard only allow connections from the socket.

```
# vi /var/lib/pgsql/16/data/postgresql.conf
```

and replace the line with listen_addresses so that PostgreSQL will listen on all interfaces and not only on our localhost.

```
#listen_addresses = 'localhost' with  listen_addresses = '*'
```

When done restart the PostgreSQL cluster and see if it comes back online in case of an error check the `pg_hba.conf` file you just edited for typos.

```
# systemctl restart postgresql-16
```

For our Zabbix server we need to create tables in the database for this we need ot install the Zabbix repository like we did for our Zabbix server and install the Zabbix package containing all the database tables images icons, ...

Add the Zabbix repository and populate the DB

```
# dnf install https://repo.zabbix.com/zabbix/6.0/rhel/9/x86_64/zabbix-release-6.0-4.el9.noarch.rpm -y
# dnf install zabbix-sql-scripts -y
```

Now we are ready to create our Zabbix users for the server and the frontend:

```
# su - postgres
# createuser --pwprompt zabbix-srv
Enter password for new role: <server-password>
Enter it again: <server-password>
```

Let's do the same for our frontend let's create a user to connect to the database:

```
# createuser --pwprompt zabbix-web
Enter password for new role: <frontend-password>
Enter it again: <frontend-password>
```

Next we have to unzip the database schema files. Run as user root followin command::

```
# gzip -d /usr/share/zabbix-sql-scripts/postgresql/server.sql.gz
```

We are now ready to create our database zabbix. Become user postgres again and run next command to create the database as our user zabbix-srv:

```
# su - postgres
# createdb -E Unicode -O zabbix-srv  zabbix
```

Let's verify that we are really connected to the database with the correct session. Login from the Postgres shell on the zabbix database

```
# psql -d zabbix -U zabbix-srv
```

Make sure we are logged in with our correct user `zabbix-srv`.

```
zabbix=> SELECT session_user, current_user;
 session_user | current_user
--------------+--------------
 zabbix-srv   | zabbix-srv
(1 row)
```

PostgreSQL works a bit different then MySQL or MariaDB when it comes to almost everything :) One of the things that it has that MySQL not has are for example shemas. If you like to know more about it i can recommend this URI. It explains in detail what it is and why we need it. But in short ... In PostgreSQL schema enables a multi-user environment that allows multiple users to access the same database without interference. Schemas are important when several users use the application and access the database in their way or when various applications utilize the same database. There is a standard schema that you can use but the better way is to create our own schema.

Note

"There is a standard schema `public` that you can use but the better way is to create our own schema this was if later something else is installed next to the Zabbix database it will be easier to create users with only access to the newly created database tables."

```
zabbix=> CREATE SCHEMA zabbix_server AUTHORIZATION "zabbix-srv";
CREATE SCHEMA
zabbix=> set search_path to "zabbix_server";
zabbix=> \dn
          List of schemas
     Name      |       Owner
---------------+-------------------
 public        | pg_database_owner
 zabbix_server | zabbix-srv
(2 rows)
```

Now we have our DB ready with correct permissions for user `zabbix-srv` but not yet for our user `zabbix-web`. Let's fix this first and give the rights to connect to our schema.

```
zabbix=# GRANT USAGE ON SCHEMA zabbix_server TO "zabbix-web";
GRANT
```

The user `zabbix-web` has now the rights to connect to our schema but cannot to anything yet lets fix this but also don't give too many rights.

```
zabbix=# GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA zabbix_server TO "zabbix-web";
GRANT
zabbix=# GRANT SELECT, UPDATE ON ALL SEQUENCES IN SCHEMA zabbix_server TO "zabbix-web";
GRANT
```

There we go both users are created with the correct permissons. We are now ready to populate the database with the Zabbix table structures etc ... log back in as user postgres and run the following commands

Let's upload the Zabbix SQL file we extracted earlier to populate our database with the needed schemas images users etc ...

Warning

"Depending on the speed of your hardware or VM this can take a few seconds upto a few minutes so please don't cancel just sit and wait for the prompt."

```
zabbix=# \i /usr/share/zabbix-sql-scripts/postgresql/server.sql
CREATE TABLE
CREATE INDEX
...
...
INSERT 0 1
INSERT 0 1
INSERT 0 1
INSERT 0 1
COMMIT
zabbix=#
```

> **Note**
>
> "If the import fails with `psql:/usr/share/zabbix-sql-scripts/postgresql/server.sql:7: ERROR:  no schema has been selected to create in` then you probably made an error in the line where you set the search path."

Lets verify that our tables are properly created with the correct permissions

```
zabbix=# \dt
                      List of relations
    Schema     |            Name            | Type  |   Owner
---------------+----------------------------+-------+------------
 zabbix_server | acknowledges               | table | zabbix-srv
 zabbix_server | actions                    | table | zabbix-srv
 zabbix_server | alerts                     | table | zabbix-srv
 zabbix_server | auditlog                   | table | zabbix-srv
 zabbix_server | autoreg_host               | table | zabbix-srv
...
...
 zabbix_server | usrgrp                     | table | zabbix-srv
 zabbix_server | valuemap                   | table | zabbix-srv
 zabbix_server | valuemap_mapping           | table | zabbix-srv
 zabbix_server | widget                     | table | zabbix-srv
 zabbix_server | widget_field               | table | zabbix-srv
(173 rows)
```

> **Note**
>
> "If you are like me and don't like to set the search path every time you logon with the user zabbix-srv to the correct search path you can run the following SQL.
> `zabbix=> alter role "zabbix-srv" set search_path = "$user", public, zabbix_server ;`"

If you are ready you can exit the database and return as user root.

```
zabbix=>  \q
# exit
```

Configure the firewall

One last thing we need to do is open the firewall and allow incoming connections for the PostgreSQL database from our Zabbix server because at the moment we dont accept any connections yet.

```
# firewall-cmd --list-all
public (active)
  target: default
  icmp-block-inversion: no
  interfaces: enp0s3 enp0s8
  sources:
  services: cockpit dhcpv6-client  ssh
  ports:
  protocols:
  forward: yes
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

First we will create an appropriate zone for our PostgreSQL DB and open port 5432/tcp but only for the ip from our Zabbix server.

```
# firewall-cmd --new-zone=postgresql-access --permanent
success

# firewall-cmd --reload
success

# firewall-cmd --get-zones
block dmz drop external home internal nm-shared postgresql-access public trusted work

# firewall-cmd --zone=postgresql-access--add-source=<zabbix-serverip> --permanent

success
# firewall-cmd --zone=postgresql-access --add-port=5432/tcp  --permanent

success
# firewall-cmd --reload
```

Now lets have a look to our firewall rules to see if they are what we expected:

```
# firewall-cmd --zone=postgresql-access --list-all
```

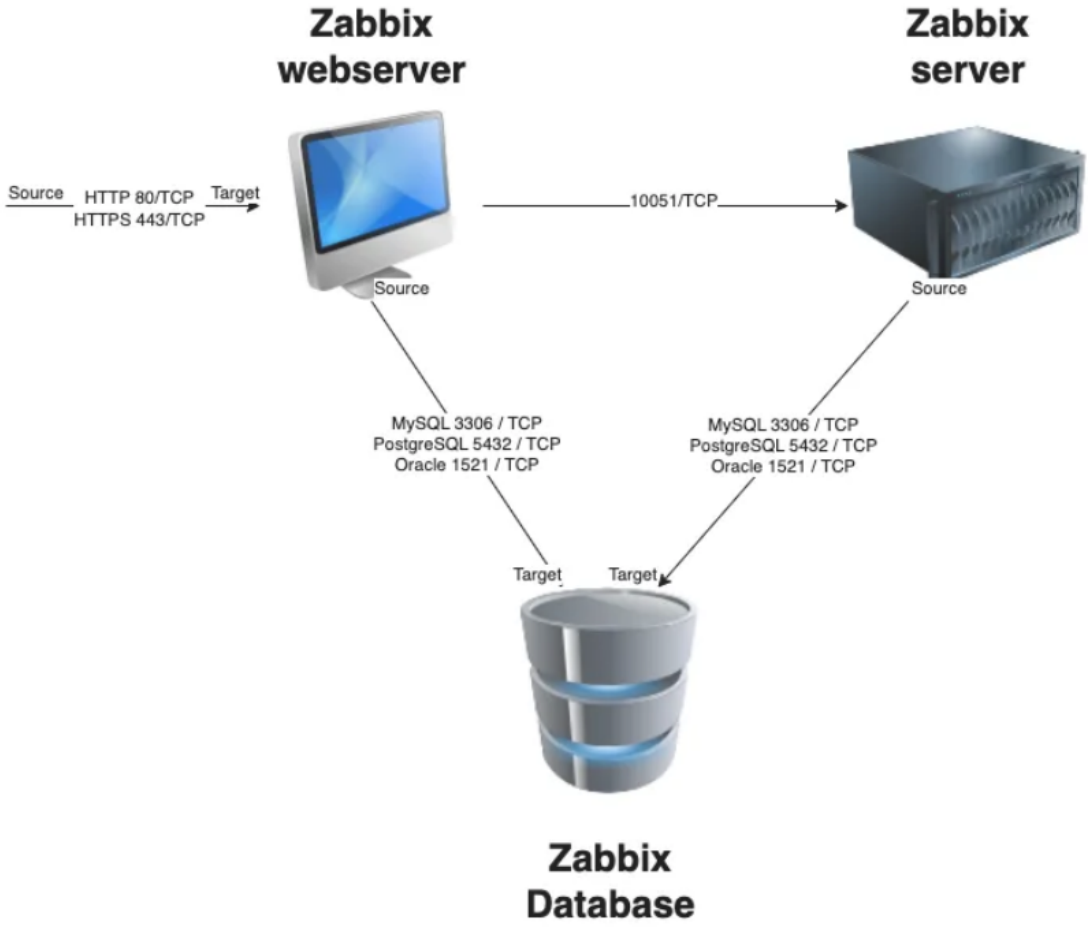
```
postgresql-access (active)
  target: default
  icmp-block-inversion: no
  interfaces:
  sources: 192.168.56.18
  services:
  ports: 5432/tcp
  protocols:
  forward: no
  masquerade: no
  forward-ports:
  source-ports:
  icmp-blocks:
  rich rules:
```

Our database server is ready now to accept connections from our Zabbix server :). You can continue with the next task Installing the Zabbix Server

6.3 Installing Zabbix

Before we can install Zabbix we first have to know how the design is. The Zabbix server has been build op modular based on 3 components.

- The Zabbix server
- The Zabbix web server
- The Zabbix database



All these components can be installed on 1 server or can be split over 3 different servers. The Zabbix server itself is the brain this part is doing all the trigger calculations and sending all the alert. The database is where the Zabbix server stores its config and all the data that we have gathered. The web server provides us with a front-end. Note that Zabbix has a API and that this is also located on the front-end and not on the Zabbix server side.

All these parts have to work together so as you can see in our image above. The Zabbix server needs to read the config and store the data in our database and the Zabbix front-end needs to be able to write the configuration in the database as well. The Zabbix front-end also needs to check the online status of our Zabbix server and needs to read some other information as well.

For our setup, we will use 2 VM's, 1 VM with a Zabbix server and our Zabbix web server and another VM with our Zabbix database.

6.3.1 Installing the Zabbix Server

Before you start to install your Zabbix server make sure the server is properly configure as we explained in our topic Basic OS configuration before we start. Something else that is important in this case is that we need to disable SELinux. We will see later in chapter Securing Zabbix how to do this properly. We can check the status of SELinux with the command `sestatus` :

```
# sestatus
SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
```

```
Loaded policy name:             targeted
Current mode:                   enforcing
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      33
```

As you can see we are now in enforcing mode. To disable SELinux just run `setenforce 0` to disable it.

```
# setenforce 0
# sestatus

SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   permissive
Mode from config file:          enforcing
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      33
```

As you can see our current mode is now permissive. However this is not persistent so we also need to alter our SELinux configuration file. This can be done by altering the file `/etc/config/selinux` and replacing enforcing by permissive. A more easy way is to run the following command :

```
sed -i 's/SELINUX=enforcing/SELINUX=permissive/g' /etc/selinux/config
```

This line will alter the config file for you. So when we run `sestatus` again we will see that we are in `permissive` mode and that our config file is also in `permissive` mode.

We can verify this with our **cat** commando.

```
# cat /etc/selinux/config

# This file controls the state of SELinux on the system.
# SELINUX= can take one of these three values:
#     enforcing - SELinux security policy is enforced.
#     permissive - SELinux prints warnings instead of enforcing.
#     disabled - No SELinux policy is loaded.
# See also:
# https://docs.fedoraproject.org/en-US/quick-docs/getting-started-with-selinux/#getting-started-with-selinux-selinux-states-and-modes
#
# NOTE: In earlier Fedora kernel builds, SELINUX=disabled would also
# fully disable SELinux during boot. If you need a system with SELinux
# fully disabled instead of SELinux running with no policy loaded, you
# need to pass selinux=0 to the kernel command line. You can use grubby
# to persistently set the bootloader to boot with selinux=0:
#
#    grubby --update-kernel ALL --args selinux=0
#
# To revert back to SELinux enabled:
#
#    grubby --update-kernel ALL --remove-args selinux
#
SELINUX=permissive
# SELINUXTYPE= can take one of these three values:
#     targeted - Targeted processes are protected,
#     minimum - Modification of targeted policy. Only selected processes are protected.
#     mls - Multi Level Security protection.
SELINUXTYPE=targeted
```

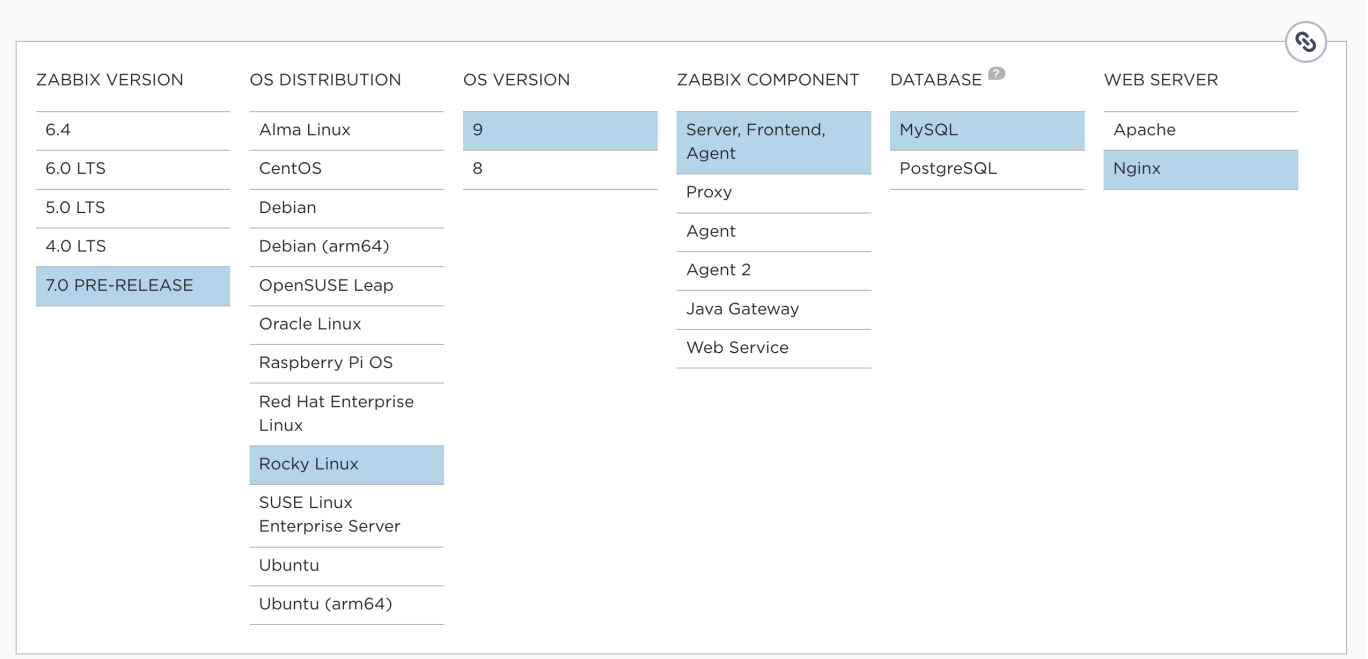
And we can also verify it with our commando `setstatus`

```
# sestatus

SELinux status:                 enabled
SELinuxfs mount:                /sys/fs/selinux
SELinux root directory:         /etc/selinux
Loaded policy name:             targeted
Current mode:                   permissive
Mode from config file:          permissive
Policy MLS status:              enabled
Policy deny_unknown status:     allowed
Memory protection checking:     actual (secure)
Max kernel policy version:      33
```

ADDING THE ZABBIX REPOSITORY

From the Zabbix Download page select the correct Zabbix version you would like to install. In our case it will be 7.0 LTS. Select the correct OS distribution as well. This will be Rocky Linux 9 in our case. We are going to install the Server and will be using NGINX.



Our first step is to disable Zabbix packages provided by EPEL, if you have it installed. Edit file /etc/yum.repos.d/epel.repo and add the following statement.

```
[epel]
...
excludepkgs=zabbix*
```

Tip

Having the EPEL repository enabled is a bad practice and could be dangerous if you use EPEL it's best to disable the repo and use dnf install --enablerepo=epel. This way you will never overwrite or install unwanted packages by accident.

Our next task is to install the Zabbix repository on our OS and do a dnf cleanup so that old cache files from our repository metadata is cleaned up.

```
rpm -Uvh https://repo.zabbix.com/zabbix/7.0/rhel/9/x86_64/zabbix-release-6.5-2.el9.noarch.rpm
dnf clean all
```

Note

A repository is a config in Linux that you can add to make packages available for you OS to install. The best way to look at it is maybe to think of it like an APP store that you add where you can find the software of your vendor. In this case the repository form Zabbix. There are many repositories you can add but you should be sure that they can be trusted. So it's always a good idea to stick to the repositories of your OS and only add extra repositories when you are sure they are to be trusted and needed. In our case the repository is from our vendor Zabbix so it should be safe to add. Epel is another popular repository for RedHat systems that is considered to be safe.

INSTALLING THE ZABBIX SERVER FOR MYSQL/MARIADB

Now that we have our repository with software added to our system we are ready to install our Zabbix server and webserver. Remember the webserver could be installed on another system. There is no need to install both on the same server.

```
dnf install zabbix-server-mysql zabbix-web-mysql
```

Now that we have installed our packages for the Zabbix server and our frontend we still need to change the configuration of our Zabbix server so that we can connect to our database. Open the file `/etc/zabbix/zabbix_server.conf` and replace the following lines:

```
DBHost=<ip or dns of your MariaDB server>
DBName=<the name of your database>
DBUser=<the user that will connect to the database>
DBPassword=<your super secret password>
```

Make sure you don't have a '#' in front of the config parameter else Zabbix will see this as text and not as a parameter. Also make sure that there are not extra duplicate lines Zabbix will always take the last config parameter if there is more then 1 line with the same parameter

In our case the config will look like this:

```
# vi /etc/zabbix/zabbix_server.conf

DBHost=<ip or dns of your MariaDB server>
DBName=zabbix
DBUser=zabbix-srv
DBPassword=<your super secret password>
DBPort=3306
```

Note

The Zabbix server configuration file has the option to include an extra config file with parameters you like to alter or add. In production it's probably better to not touch the configuration file but to add a new file and include the parameters you like to change. This way you never have to edit your original configuration file after an upgrade it will also make your life more easy when working with configuration tools like Ansible, Puppet, SaltStack, ... The only thing that needs to be done is remove the # in front of the line '# Include=/usr/local/etc/zabbix_server.conf.d/*.conf' and make sure the path exists with a customized config file of your won that is readable by the user zabbix.

Ok now that we have changed the configuration of you Zabbix server so that it is able to connect to our DB we are ready to start. Run the following command to enable the Zabbix server and make it active on boot next time.

`systemctl enable zabbix-server --now`

Our Zabbix server service will start and if everything goes well you should see in the Zabbix server log file the following output

`tail /var/log/zabbix/zabbix_server.log`

```
 1123:20231120:110604.440 Starting Zabbix Server. Zabbix 7.0.0alpha7 (revision 60de6a81aca).
 1123:20231120:110604.440 ****** Enabled features ******
 1123:20231120:110604.440 SNMP monitoring:           YES
 1123:20231120:110604.440 IPMI monitoring:           YES
 1123:20231120:110604.440 Web monitoring:            YES
 1123:20231120:110604.440 VMware monitoring:         YES
 1123:20231120:110604.440 SMTP authentication:       YES
 1123:20231120:110604.440 ODBC:                      YES
 1123:20231120:110604.440 SSH support:               YES
 1123:20231120:110604.440 IPv6 support:              YES
 1123:20231120:110604.440 TLS support:               YES
 1123:20231120:110604.440 ******************************
 1123:20231120:110604.440 using configuration file: /etc/zabbix/zabbix_server.conf
 1123:20231120:110604.470 current database version (mandatory/optional): 06050143/06050143
 1123:20231120:110604.470 required mandatory version: 06050143
 1124:20231120:110604.490 starting HA manager
 1124:20231120:110604.507 HA manager started in active mode
 1123:20231120:110604.508 server #0 started [main process]
 1126:20231120:110604.509 server #2 started [configuration syncer #1]
 1125:20231120:110604.510 server #1 started [service manager #1]
 1133:20231120:110604.841 server #9 started [lld worker #1]
 1132:20231120:110604.841 server #8 started [lld manager #1]
 1134:20231120:110604.841 server #10 started [lld worker #2]
```

If there was an error and the server was not able to connect to the database you would see something like this in the server log file :

```
10773:20231118:213248.570 Starting Zabbix Server. Zabbix 7.0.0alpha7 (revision 60de6a81aca).
10773:20231118:213248.570 ****** Enabled features ******
10773:20231118:213248.570 SNMP monitoring:           YES
10773:20231118:213248.570 IPMI monitoring:           YES
10773:20231118:213248.570 Web monitoring:            YES
10773:20231118:213248.570 VMware monitoring:         YES
10773:20231118:213248.570 SMTP authentication:       YES
10773:20231118:213248.570 ODBC:                      YES
10773:20231118:213248.570 SSH support:               YES
10773:20231118:213248.570 IPv6 support:              YES
10773:20231118:213248.570 TLS support:               YES
10773:20231118:213248.570 ******************************
10773:20231118:213248.570 using configuration file: /etc/zabbix/zabbix_server.conf
10773:20231118:213248.574 [Z3001] connection to database 'zabbix' failed: [2002] Can't connect to server on 'xxx.xxx.xxx.xxx' (115)
10773:20231118:213248.574 database is down: reconnecting in 10 seconds
10773:20231118:213258.579 [Z3001] connection to database 'zabbix' failed: [2002] Can't connect to server on 'xxx.xxx.xxx.xxx' (115)
10773:20231118:213258.579 database is down: reconnecting in 10 seconds
```

Let's check the Zabbix server service to see if it's enabled so that it survives a reboot

```
# systemctl status zabbix-server

● zabbix-server.service - Zabbix Server
     Loaded: loaded (/usr/lib/systemd/system/zabbix-server.service; enabled; preset: disabled)
     Active: active (running) since Mon 2023-11-20 11:06:04 CET; 1h 2min ago
   Main PID: 1123 (zabbix_server)
      Tasks: 59 (limit: 12344)
     Memory: 52.6M
        CPU: 20.399s
     CGroup: /system.slice/zabbix-server.service
             ├─1123 /usr/sbin/zabbix_server -c /etc/zabbix/zabbix_server.conf
             ├─1124 "/usr/sbin/zabbix_server: ha manager"
             ├─1125 "/usr/sbin/zabbix_server: service manager #1 [processed 0 events, updated 0 event tags, deleted 0 problems, synced 0 service updates, idle
5.008686 sec during 5.016382 sec]"
             ├─1126 "/usr/sbin/zabbix_server: configuration syncer [synced configuration in 0.092797 sec, idle 10 sec]"
             ├─1127 "/usr/sbin/zabbix_server: alert manager #1 [sent 0, failed 0 alerts, idle 5.027620 sec during 5.027828 sec]"
             ├─1128 "/usr/sbin/zabbix_server: alerter #1 started"
             ├─1129 "/usr/sbin/zabbix_server: alerter #2 started"
             ├─1130 "/usr/sbin/zabbix_server: alerter #3 started"
             ├─1131 "/usr/sbin/zabbix_server: preprocessing manager #1 [queued 1, processed 2 values, idle 5.490312 sec during 5.490555 sec]"
             ├─1132 "/usr/sbin/zabbix_server: lld manager #1 [processed 1 LLD rules, idle 5.028973sec during 5.029123 sec]"
             ├─1133 "/usr/sbin/zabbix_server: lld worker #1 [processed 1 LLD rules, idle 60.060180 sec during 60.085009 sec]"
             ├─1134 "/usr/sbin/zabbix_server: lld worker #2 [processed 1 LLD rules, idle 60.065526 sec during 60.095165 sec]"
             ├─1135 "/usr/sbin/zabbix_server: housekeeper [deleted 0 hist/trends, 0 items/triggers, 0 events, 0 sessions, 0 alarms, 0 audit items, 0
autoreg_host, 0 records in 0.019108 sec, idle for 1 hour(s)]"
             ├─1136 "/usr/sbin/zabbix_server: timer #1 [updated 0 hosts, suppressed 0 events in 0.002856 sec, idle 59 sec]"
             ├─1137 "/usr/sbin/zabbix_server: http poller #1 [got 0 values in 0.000059 sec, idle 5 sec]"
             ├─1138 "/usr/sbin/zabbix_server: discovery manager #1 [processing 0 rules, 0.000000% of queue used, 0 unsaved checks]"
             ├─1139 "/usr/sbin/zabbix_server: history syncer #1 [processed 0 values, 0 triggers in 0.000036 sec, idle 1 sec]"
             ├─1140 "/usr/sbin/zabbix_server: history syncer #2 [processed 1 values, 0 triggers in 0.005016 sec, idle 1 sec]"
             ├─1141 "/usr/sbin/zabbix_server: history syncer #3 [processed 0 values, 0 triggers in 0.000031 sec, idle 1 sec]"
             ├─1142 "/usr/sbin/zabbix_server: history syncer #4 [processed 0 values, 0 triggers in 0.000014 sec, idle 1 sec]"
             ├─1143 "/usr/sbin/zabbix_server: escalator #1 [processed 0 escalations in 0.005587 sec, idle 3 sec]"
             ├─1144 "/usr/sbin/zabbix_server: proxy poller #1 [exchanged data with 0 proxies in 0.000010 sec, idle 5 sec]"
             ├─1145 "/usr/sbin/zabbix_server: self-monitoring [processed data in 0.000016 sec, idle 1 sec]"
             ├─1146 "/usr/sbin/zabbix_server: task manager [processed 0 task(s) in 0.002511 sec, idle 5 sec]"
             ├─1147 "/usr/sbin/zabbix_server: poller #1 [got 0 values in 0.000009 sec, idle 1 sec]"
             ├─1148 "/usr/sbin/zabbix_server: poller #2 [got 1 values in 0.000232 sec, idle 1 sec]"
             ├─1149 "/usr/sbin/zabbix_server: poller #3 [got 0 values in 0.000015 sec, idle 1 sec]"
             ├─1150 "/usr/sbin/zabbix_server: poller #4 [got 0 values in 0.000010 sec, idle 1 sec]"
```

This concludes our chapter on installing and configuring our Zabbix server. Next we have to configure our frontend. You can have a look at Installing Zabbix frontend with Nginx or Installing Zabbix frontend with Apache

INSTALLING THE ZABBIX SERVER FOR POSTGRESQL

Now that we have our repository with software added to our system we are ready to install our Zabbix server and webserver. Remember the webserver could be installed on another system. There is no need to install both on the same server.

```
dnf install zabbix-server-pgsql zabbix-web-pgsql
```

Now that we have installed our packages for the Zabbix server and our frontend we still need to change the configuration of our Zabbix server so that we can connect to our database. Open the file `/etc/zabbix/zabbix_server.conf` and replace the following lines:

```
DBHost=<ip or dns of your PostgreSQL server>
DBName=<the name of your database>
DBSchema=<our PostgreSQL schema name>
DBUser=<the user that will connect to the database>
DBPassword=<your super secret password>
```

Make sure you don't have a '#' in front of the config parameter else Zabbix will see this as text and not as a parameter. Also make sure that there are not extra duplicate lines Zabbix will always take the last config parameter if there is more then 1 line with the same parameter

In our case the config will look like this:

```
# vi /etc/zabbix/zabbix_server.conf

DBHost=<ip or dns of your MariaDB server>
DBName=zabbix
DBSchema=zabbix_server
DBUser=zabbix-srv
DBPassword=<your super secret password>
DBPort=5432
```

Note

The Zabbix server configuration file has the option to include an extra config file with parameters you like to alter or add. In production it's probably better to not touch the configuration file but to add a new file and include the parameters you like to change. This way you never have to edit your original configuration file after an upgrade it will also make your life more easy when working with configuration tools like Ansible, Puppet, SaltStack, ... The only thing that needs to be done is remove the # in front of the line '# Include=/usr/local/etc/zabbix_server.conf.d/*.conf' and make sure the path exists with a customized config file of your won that is readable by the user zabbix.

Ok now that we have changed the configuration of you Zabbix server so that it is able to connect to our DB we are ready to start. Run the following command to enable the Zabbix server and make it active on boot next time.

```
systemctl enable zabbix-server --now
```

Our Zabbix server service will start and if everything goes well you should see in the Zabbix server log file the following output

```
tail /var/log/zabbix/zabbix_server.log
```

```
 1123:20231120:110604.440 Starting Zabbix Server. Zabbix 7.0.0alpha7 (revision 60de6a81aca).
 1123:20231120:110604.440 ****** Enabled features ******
 1123:20231120:110604.440 SNMP monitoring:           YES
 1123:20231120:110604.440 IPMI monitoring:           YES
 1123:20231120:110604.440 Web monitoring:            YES
 1123:20231120:110604.440 VMware monitoring:         YES
 1123:20231120:110604.440 SMTP authentication:       YES
 1123:20231120:110604.440 ODBC:                      YES
 1123:20231120:110604.440 SSH support:               YES
 1123:20231120:110604.440 IPv6 support:              YES
 1123:20231120:110604.440 TLS support:               YES
 1123:20231120:110604.440 ******************************
 1123:20231120:110604.440 using configuration file: /etc/zabbix/zabbix_server.conf
 1123:20231120:110604.470 current database version (mandatory/optional): 06050143/06050143
 1123:20231120:110604.470 required mandatory version: 06050143
 1124:20231120:110604.490 starting HA manager
 1124:20231120:110604.507 HA manager started in active mode
 1123:20231120:110604.508 server #0 started [main process]
 1126:20231120:110604.509 server #2 started [configuration syncer #1]
 1125:20231120:110604.510 server #1 started [service manager #1]
 1133:20231120:110604.841 server #9 started [lld worker #1]
 1132:20231120:110604.841 server #8 started [lld manager #1]
 1134:20231120:110604.841 server #10 started [lld worker #2]
```

If there was an error and the server was not able to connect to the database you would see something like this in the server log file :

```
10773:20231118:213248.570 Starting Zabbix Server. Zabbix 7.0.0alpha7 (revision 60de6a81aca).
10773:20231118:213248.570 ****** Enabled features ******
10773:20231118:213248.570 SNMP monitoring:           YES
10773:20231118:213248.570 IPMI monitoring:           YES
10773:20231118:213248.570 Web monitoring:            YES
10773:20231118:213248.570 VMware monitoring:         YES
10773:20231118:213248.570 SMTP authentication:       YES
10773:20231118:213248.570 ODBC:                      YES
10773:20231118:213248.570 SSH support:               YES
10773:20231118:213248.570 IPv6 support:              YES
10773:20231118:213248.570 TLS support:               YES
10773:20231118:213248.570 ******************************
10773:20231118:213248.570 using configuration file: /etc/zabbix/zabbix_server.conf
10773:20231118:213248.574 [Z3001] connection to database 'zabbix' failed: [2002] Can't connect to server on 'xxx.xxx.xxx.xxx' (115)
10773:20231118:213248.574 database is down: reconnecting in 10 seconds
10773:20231118:213258.579 [Z3001] connection to database 'zabbix' failed: [2002] Can't connect to server on 'xxx.xxx.xxx.xxx' (115)
10773:20231118:213258.579 database is down: reconnecting in 10 seconds
```

Let's check the Zabbix server service to see if it's enabled so that it survives a reboot

```
# systemctl status zabbix-server
```

```
● zabbix-server.service - Zabbix Server
     Loaded: loaded (/usr/lib/systemd/system/zabbix-server.service; enabled; preset: disabled)
     Active: active (running) since Mon 2023-11-20 11:06:04 CET; 1h 2min ago
   Main PID: 1123 (zabbix_server)
      Tasks: 59 (limit: 12344)
     Memory: 52.6M
        CPU: 20.399s
     CGroup: /system.slice/zabbix-server.service
             ├─1123 /usr/sbin/zabbix_server -c /etc/zabbix/zabbix_server.conf
             ├─1124 "/usr/sbin/zabbix_server: ha manager"
             ├─1125 "/usr/sbin/zabbix_server: service manager #1 [processed 0 events, updated 0 event tags, deleted 0 problems, synced 0 service updates, idle
5.008686 sec during 5.016382 sec]"
             ├─1126 "/usr/sbin/zabbix_server: configuration syncer [synced configuration in 0.092797 sec, idle 10 sec]"
             ├─1127 "/usr/sbin/zabbix_server: alert manager #1 [sent 0, failed 0 alerts, idle 5.027620 sec during 5.027828 sec]"
```

```
             ├─1128 "/usr/sbin/zabbix_server: alerter #1 started"
             ├─1129 "/usr/sbin/zabbix_server: alerter #2 started"
             ├─1130 "/usr/sbin/zabbix_server: alerter #3 started"
             ├─1131 "/usr/sbin/zabbix_server: preprocessing manager #1 [queued 1, processed 2 values, idle 5.490312 sec during 5.490555 sec]"
             ├─1132 "/usr/sbin/zabbix_server: lld manager #1 [processed 1 LLD rules, idle 5.028973sec during 5.029123 sec]"
             ├─1133 "/usr/sbin/zabbix_server: lld worker #1 [processed 1 LLD rules, idle 60.060180 sec during 60.085009 sec]"
             ├─1134 "/usr/sbin/zabbix_server: lld worker #2 [processed 1 LLD rules, idle 60.065526 sec during 60.095165 sec]"
             ├─1135 "/usr/sbin/zabbix_server: housekeeper [deleted 0 hist/trends, 0 items/triggers, 0 events, 0 sessions, 0 alarms, 0 audit items, 0
autoreg_host, 0 records in 0.019108 sec, idle for 1 hour(s)]"
             ├─1136 "/usr/sbin/zabbix_server: timer #1 [updated 0 hosts, suppressed 0 events in 0.002856 sec, idle 59 sec]"
             ├─1137 "/usr/sbin/zabbix_server: http poller #1 [got 0 values in 0.000059 sec, idle 5 sec]"
             ├─1138 "/usr/sbin/zabbix_server: discovery manager #1 [processing 0 rules, 0.000000% of queue used, 0 unsaved checks]"
             ├─1139 "/usr/sbin/zabbix_server: history syncer #1 [processed 0 values, 0 triggers in 0.000036 sec, idle 1 sec]"
             ├─1140 "/usr/sbin/zabbix_server: history syncer #2 [processed 1 values, 0 triggers in 0.005016 sec, idle 1 sec]"
             ├─1141 "/usr/sbin/zabbix_server: history syncer #3 [processed 0 values, 0 triggers in 0.000031 sec, idle 1 sec]"
             ├─1142 "/usr/sbin/zabbix_server: history syncer #4 [processed 0 values, 0 triggers in 0.000014 sec, idle 1 sec]"
             ├─1143 "/usr/sbin/zabbix_server: escalator #1 [processed 0 escalations in 0.005587 sec, idle 3 sec]"
             ├─1144 "/usr/sbin/zabbix_server: proxy poller #1 [exchanged data with 0 proxies in 0.000010 sec, idle 5 sec]"
             ├─1145 "/usr/sbin/zabbix_server: self-monitoring [processed data in 0.000016 sec, idle 1 sec]"
             ├─1146 "/usr/sbin/zabbix_server: task manager [processed 0 task(s) in 0.002511 sec, idle 5 sec]"
             ├─1147 "/usr/sbin/zabbix_server: poller #1 [got 0 values in 0.000009 sec, idle 1 sec]"
             ├─1148 "/usr/sbin/zabbix_server: poller #2 [got 1 values in 0.000232 sec, idle 1 sec]"
             ├─1149 "/usr/sbin/zabbix_server: poller #3 [got 0 values in 0.000015 sec, idle 1 sec]"
             ├─1150 "/usr/sbin/zabbix_server: poller #4 [got 0 values in 0.000010 sec, idle 1 sec]"
```

This concludes our chapter on installing and configuring our Zabbix server. Next we have to configure our frontend. You can have a look at Installing Zabbix frontend with Nginx or Installing Zabbix frontend with Apache

Installing Zabbix frontend with Nginx

Before we can configure our frontend we need to install our package first. If you run the frontend on the same server as the Zabbix server then there is nothing else you have to do you can just run the following command on your server to install the packages needed for our frontend to install:

```
dnf install zabbix-nginx-conf and zabbix-web-mysql or if you used Postgres dnf install zabbix-web-pgsql
```

In case the frontend is on another server installed you need to add the Zabbix repository first like we did on our Zabbix server. In case you forgot or just skipped to this topic and don't know how to do this have a look at Adding the Zabbix repository

First thing we have to do is alter the Nginx configuration file so that we don't use the standard config.

```
vi /etc/nginx/nginx.conf
```

In this config look for the followin block that starts with :

```
    server {
        listen       80;
        listen       [::]:80;
        server_name  _;
        root         /usr/share/nginx/html;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;
```

And place the following lines in comment:

```
    server {
#        listen       80;
#        listen       [::]:80;
#        server_name  _;
#        root         /usr/share/nginx/html;
```

We now have to alter the Zabbix configuration file so that it matches our setup. Edit the following file:

```
vi /etc/nginx/conf.d/zabbix.conf
```

```
server {
        listen          8080;
        server_name     example.com;

        root    /usr/share/zabbix;

        index   index.php;
```

Replace the first 2 lines with the correct port and domain for your frontend in case you don't have a domain you can replace server_name with _; like in the exaple below:

```
server {
#        listen          8080;
#        server_name     example.com;
        listen          80;
        server_name     _;

        root    /usr/share/zabbix;

        index   index.php;
```

We are now ready to start our websever and enable it so that it comes online after a reboot.

```
systemctl enable php-fpm --now
systemctl enable nginx --now
```

Let's verify if the service is properly started and enabled so that it survives our reboot next time.

```
# systemctl status nginx

● nginx.service - The nginx HTTP and reverse proxy server
     Loaded: loaded (/usr/lib/systemd/system/nginx.service; enabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/nginx.service.d
             └─php-fpm.conf
     Active: active (running) since Mon 2023-11-20 11:42:18 CET; 30min ago
   Main PID: 1206 (nginx)
      Tasks: 2 (limit: 12344)
     Memory: 4.8M
        CPU: 38ms
     CGroup: /system.slice/nginx.service
             ├─1206 "nginx: master process /usr/sbin/nginx"
             └─1207 "nginx: worker process"

Nov 20 11:42:18 zabbix-srv systemd[1]: Starting The nginx HTTP and reverse proxy server...
Nov 20 11:42:18 zabbix-srv nginx[1204]: nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
Nov 20 11:42:18 zabbix-srv nginx[1204]: nginx: configuration file /etc/nginx/nginx.conf test is successful
Nov 20 11:42:18 zabbix-srv systemd[1]: Started The nginx HTTP and reverse proxy server.
```

The service is running and enabled so there is only 1 thing left to do before we can start the configuration in the GUI and that is to configure our firewall to allow incoming communication to the webserver.

```
firewall-cmd --add-service=http --permanent
firewall-cmd --reload
```

Open your browser and go to the url or ip of your frontend :

```
http://<ip or dns of the zabbix frontend server>/
```

If all goes well you should be greeted with a Zabbix welcome page. In case you have an error check the configuration again or have a look at the nginx log file :

`/var/log/nginx/error.log`

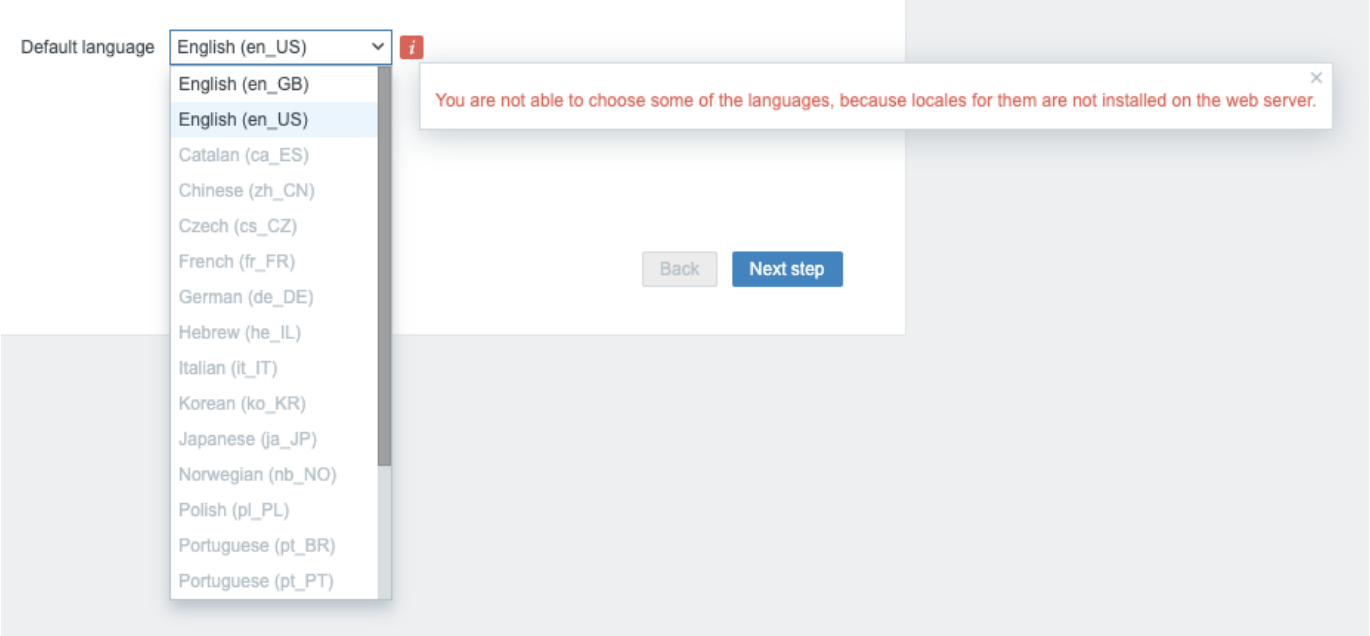
or run

`journalctl -xe`

This should help you in locating the errors you made.

When you point your browser to the correct URL you should be greeted with a page like here :

As you see there is only a limited list of local translations available on our Zabbix frontend to choose from



What if we want to install Chinese as language or another language from the list ? Run the next command to get a list of all locales available for your OS.

```
dnf list glibc-langpack-*
```

This will give you a list like

```
Installed Packages
glibc-langpack-en.x86_64
Available Packages
glibc-langpack-aa.x86_64
...
```

```
glibc-langpack-zu.x86_64
```

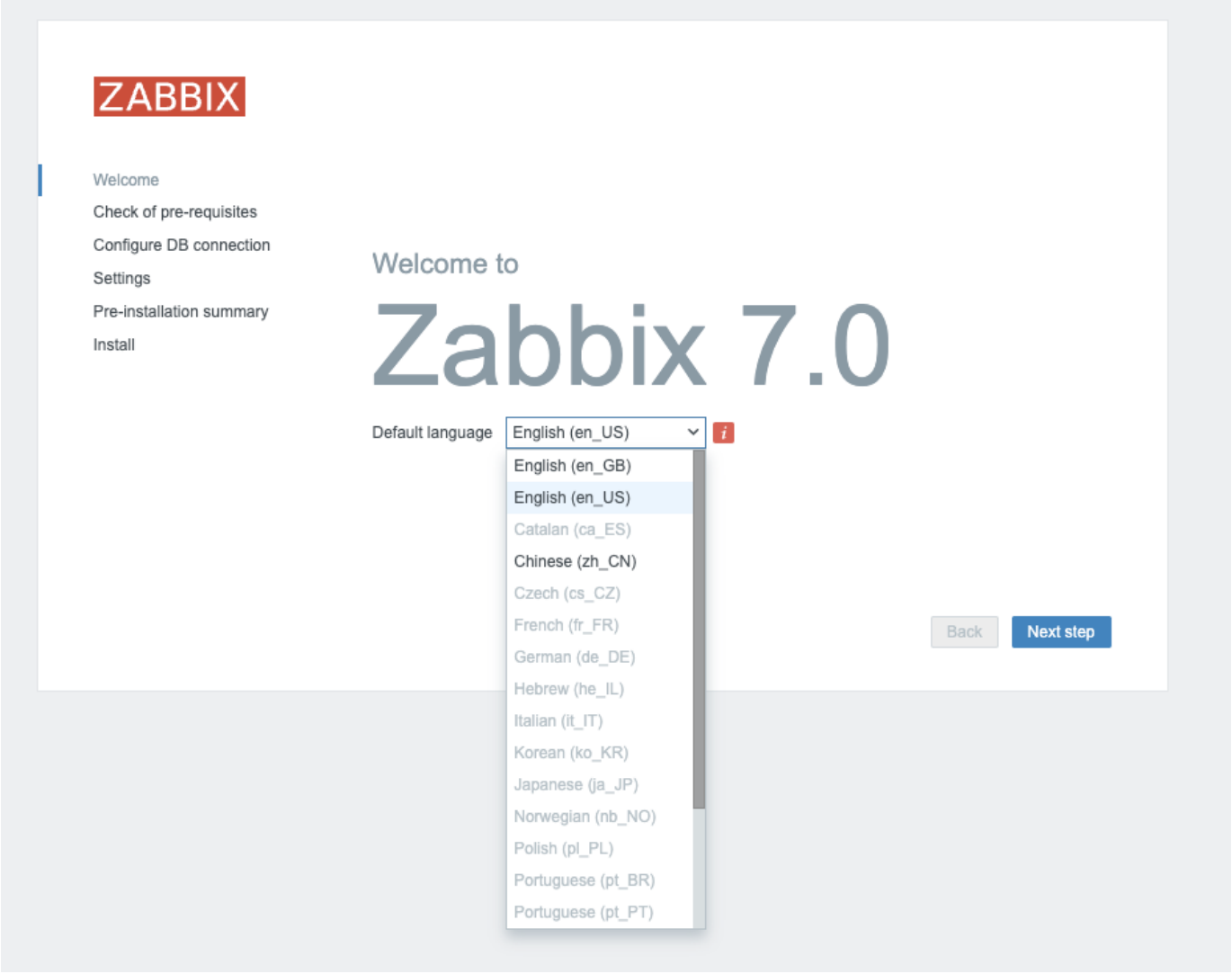
Let's search for our Chinese locale to see if it is available. As you can see the code starts with zh

```
# dnf list glibc-langpack-* | grep zh
glibc-langpack-zh.x86_64
glibc-langpack-lzh.x86_64
```

The command returns us 2 lines but as we have seen that the code was zh_CN we only have to install the first package.

```
# dnf install glibc-langpack-zh.x86_64 -y
```

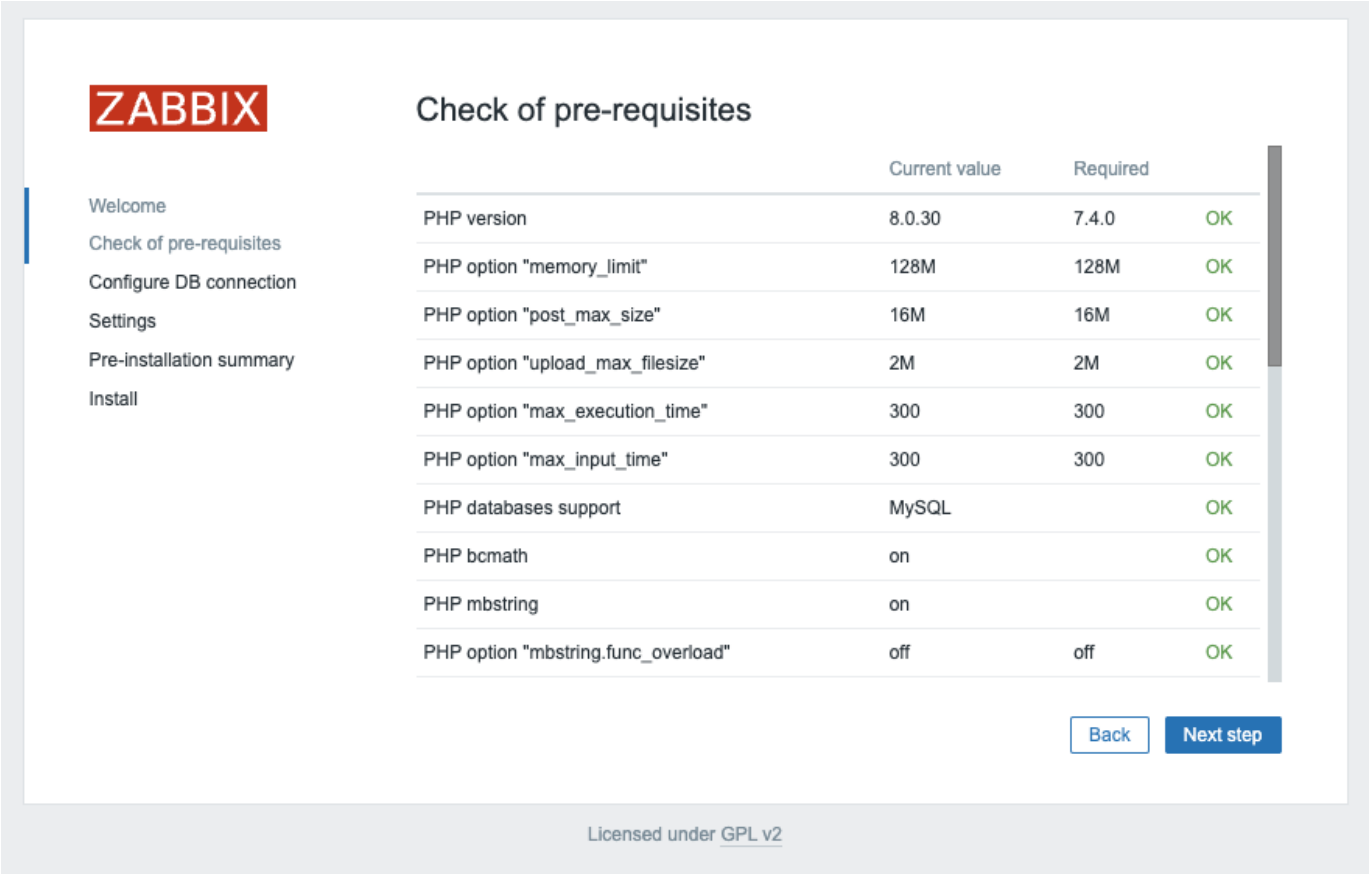
When we return now to our frontend we are able to select the Chinese language.



Note

If your language is not available in the frontend don't panic it just means that there is no translation or that the translation was not 100% complete. Zabbis is free and relies on the community for it's translations so you can help in creating the translation. Go to the page `https://translate.zabbix.com/` and help us to make Zabbix get better. Once the translation is complete the next Zabbix minor patch version should have your language included.

Click next when you are satisfied with the transaltions available. You will arrive at a screen to verifiy if all pre-requisites are met. If not fix them first but normaly it should be fine and you should be just able to click Next



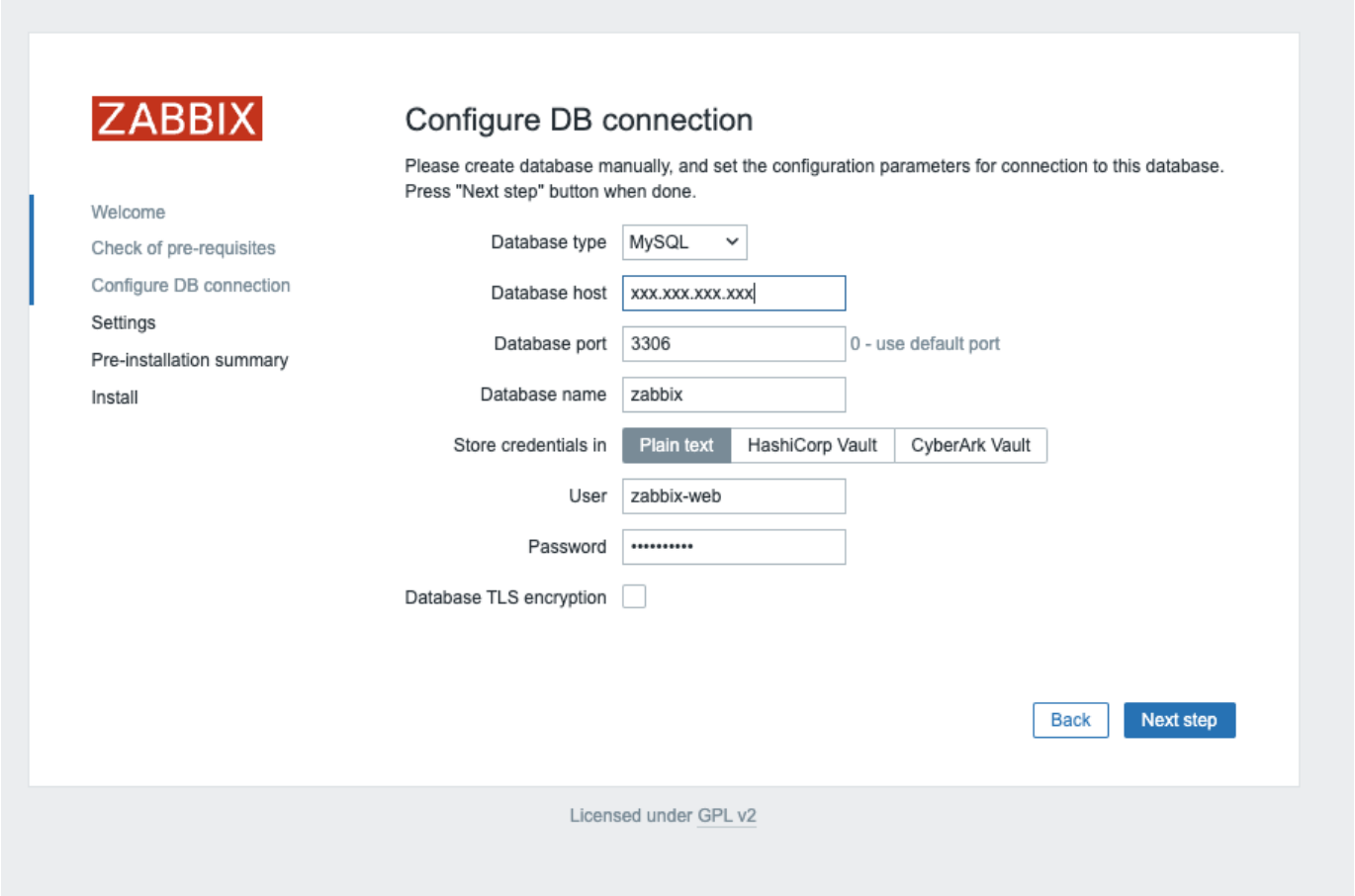
The next page will show you a page with the connection parameters for our database.

First you select your DB type 'MySQL' or 'PostgreSQL' and fill in the IP or DNS name of the location of your database server. Use port 3306 for MariaDB/MySQL or 5432 if you used PostgreSQL.

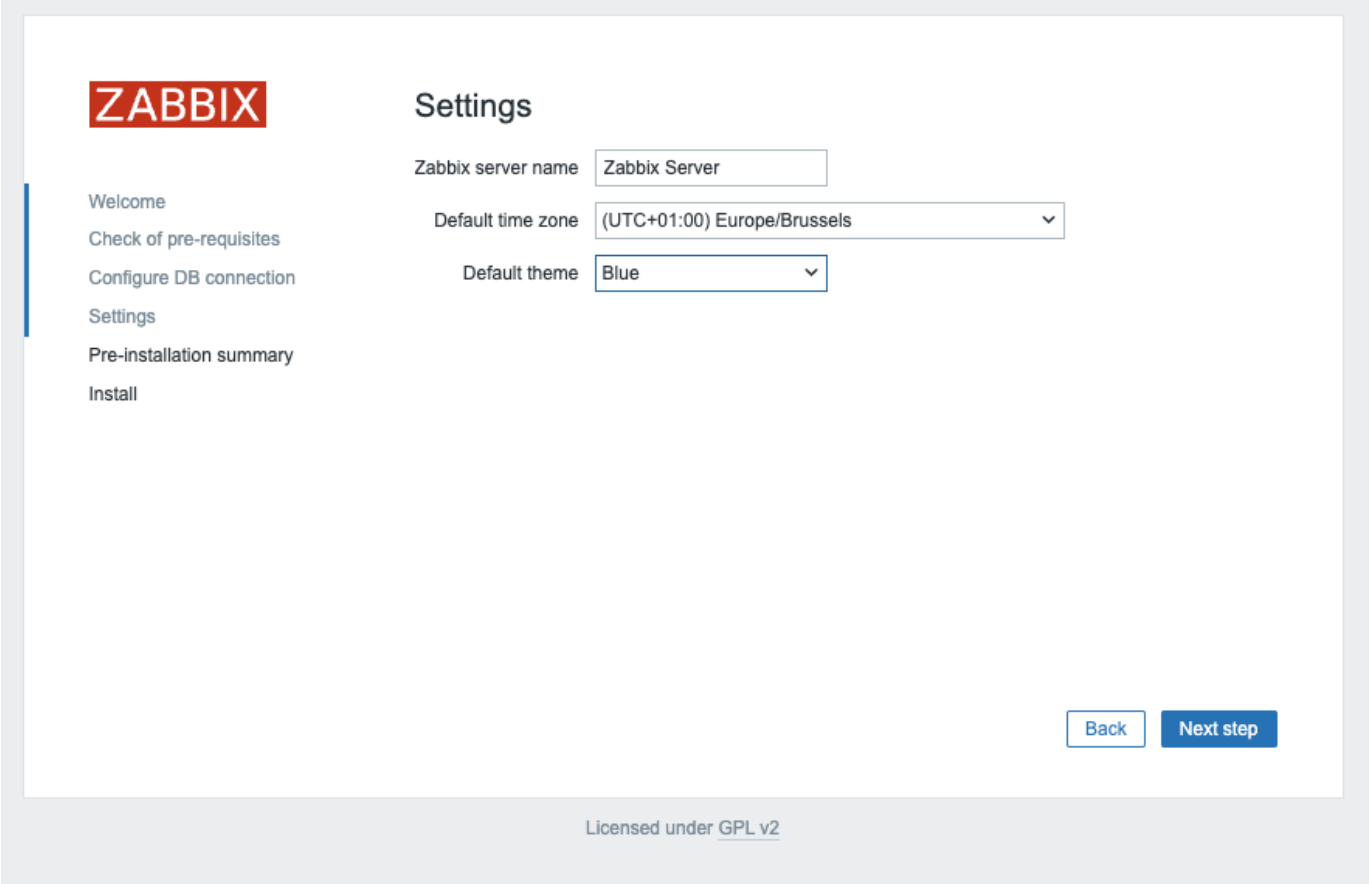
Fill in the correct database name, in our case it was `zabbix`. If you used PostgreSQL then you also need to fill in the correct schema name in our case it was `zabbix_server`

Next line will ask you for the DB users here we created a user `zabbix-web`. Enter it in the correct field and fill in the password that you used for this user.

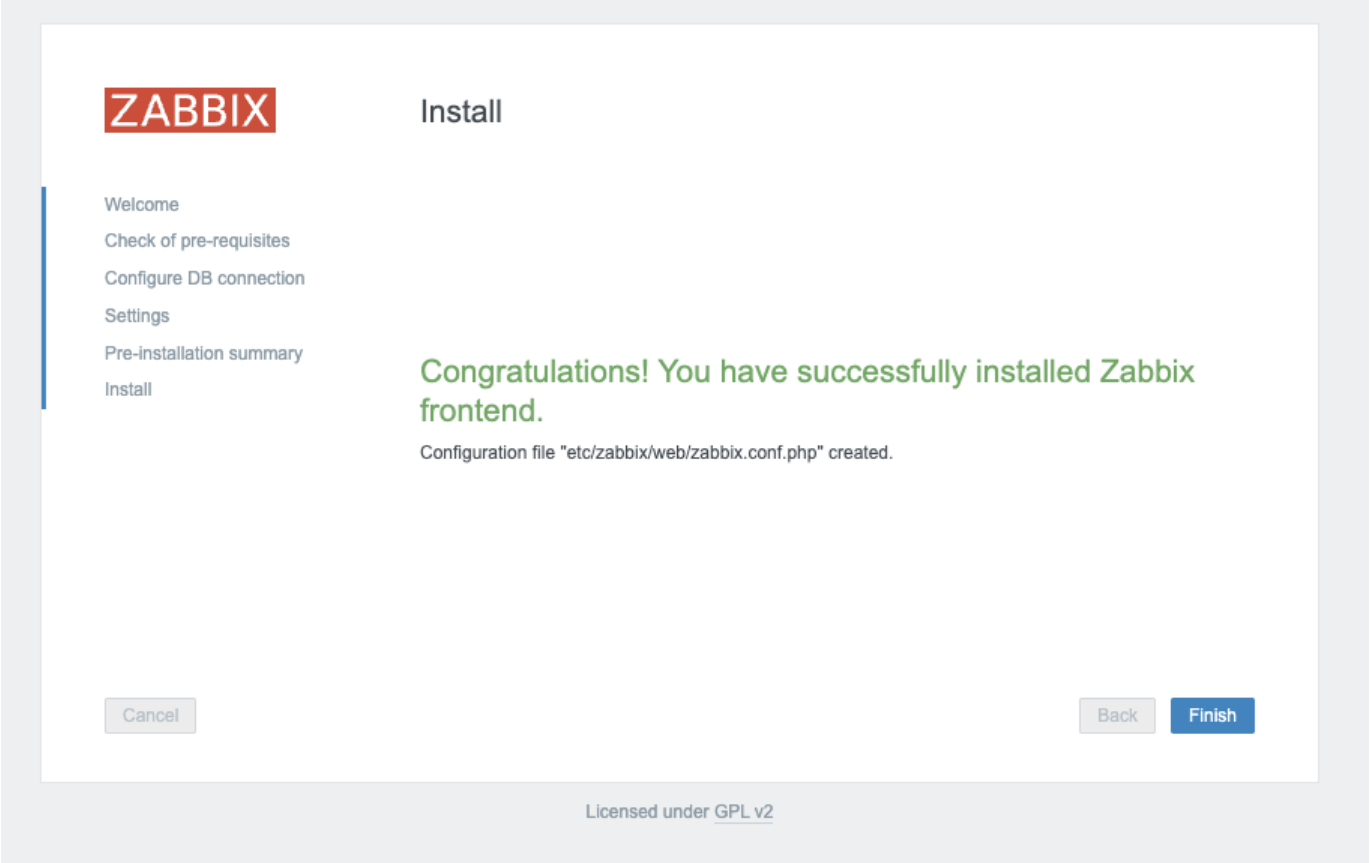
Make sure the option `Database TLS encryption` is not selected and press `Next step`.



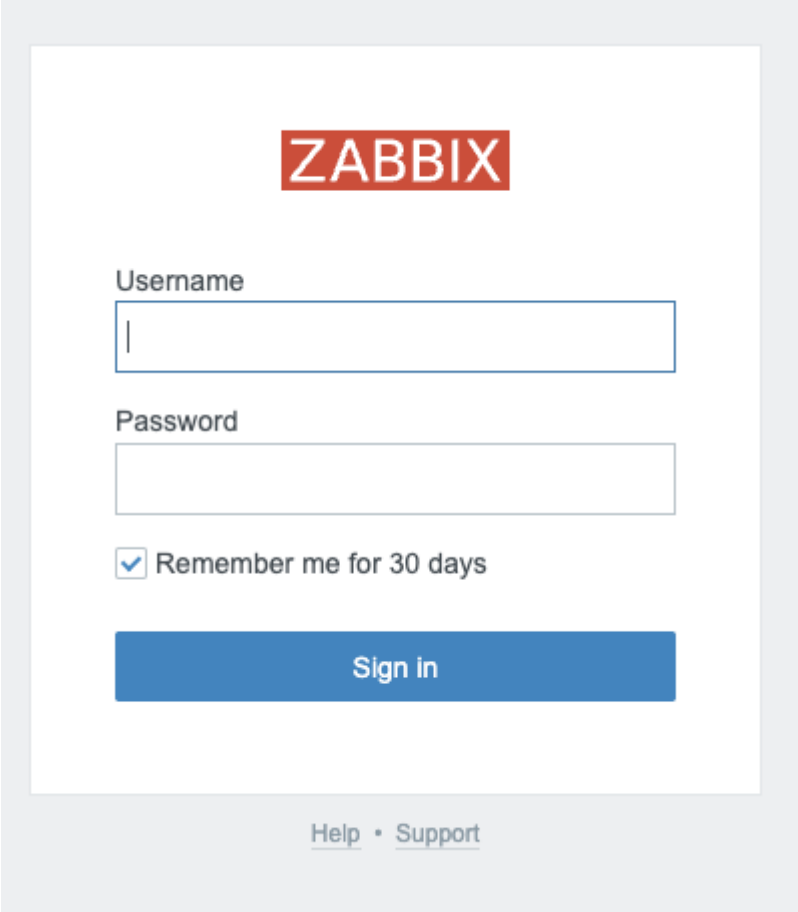
We are almost there. The only thing that rests us to do is give our instance a name, select our timezone and select a default time we like to use.



Press `Next step` again you will see a page that tells you that the configuration is successful. Press Finish to end the configuration.



We are now ready to login :



Login : Admin Password : zabbix

If you like to secure the frontend with SSL then checkout the following topic

Securing Zabbix

Installing Zabbix frontend with Apache

Before we can configure our frontend we need to install our package first. If you run the frontend on the same server as the Zabbix server then there is nothing else you have to do you can just run the following command on your server to install the packages needed for our frontend to install:

```
dnf install zabbix-apache-conf and zabbix-web-mysql or if you used Postgres dnf install zabbix-web-pgsql
```

In case the frontend is on another server installed you need to add the Zabbix repository first like we did on our Zabbix server. In case you forgot or just skipped to this topic and don't know how to do this have a look at Adding the Zabbix repository

We are now ready to start our websever and enable it so that it comes online after a reboot.

```
systemctl enable php-fpm --now
systemctl enable httpd --now
```

Let's verify if the service is properly started and enabled so that it survives our reboot next time.

```
# systemctl status httpd
```

```
● httpd.service - The Apache HTTP Server
     Loaded: loaded (/usr/lib/systemd/system/httpd.service; enabled; preset: disabled)
    Drop-In: /usr/lib/systemd/system/httpd.service.d
             └─php-fpm.conf
     Active: active (running) since Mon 2024-03-04 08:50:17 CET; 7min ago
```

```
       Docs: man:httpd.service(8)
   Main PID: 690 (httpd)
     Status: "Total requests: 96; Idle/Busy workers 100/0;Requests/sec: 0.213; Bytes served/sec: 560 B/sec"
      Tasks: 278 (limit: 22719)
     Memory: 39.6M
        CPU: 1.132s
     CGroup: /system.slice/httpd.service
             ├─ 690 /usr/sbin/httpd -DFOREGROUND
             ├─ 736 /usr/sbin/httpd -DFOREGROUND
             ├─ 737 /usr/sbin/httpd -DFOREGROUND
             ├─ 738 /usr/sbin/httpd -DFOREGROUND
             ├─ 739 /usr/sbin/httpd -DFOREGROUND
             └─4534 /usr/sbin/httpd -DFOREGROUND

Mar 04 08:50:17 localhost.localdomain systemd[1]: Starting The Apache HTTP Server...
Mar 04 08:50:17 localhost.localdomain httpd[690]: AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using
localhost.localdomain. Set th>
Mar 04 08:50:17 localhost.localdomain httpd[690]: Server configured, listening on: port 80
Mar 04 08:50:17 localhost.localdomain systemd[1]: Started The Apache HTTP Server.x
```

The service is running and enabled so there is only 1 thing left to do before we can start the configuration in the GUI and that is to configure our firewall to allow incoming communication to the webserver.

```
firewall-cmd --add-service=http --permanent
firewall-cmd --reload
```

Open your browser and go to the url or ip of your frontend :

```
http://<ip or dns of the zabbix frontend server>/zabbix/
```

If all goes well you should be greeted with a Zabbix welcome page. In case you have an error check the configuration again or have a look at the Apache log file :

`/var/log/httpd/error_log`

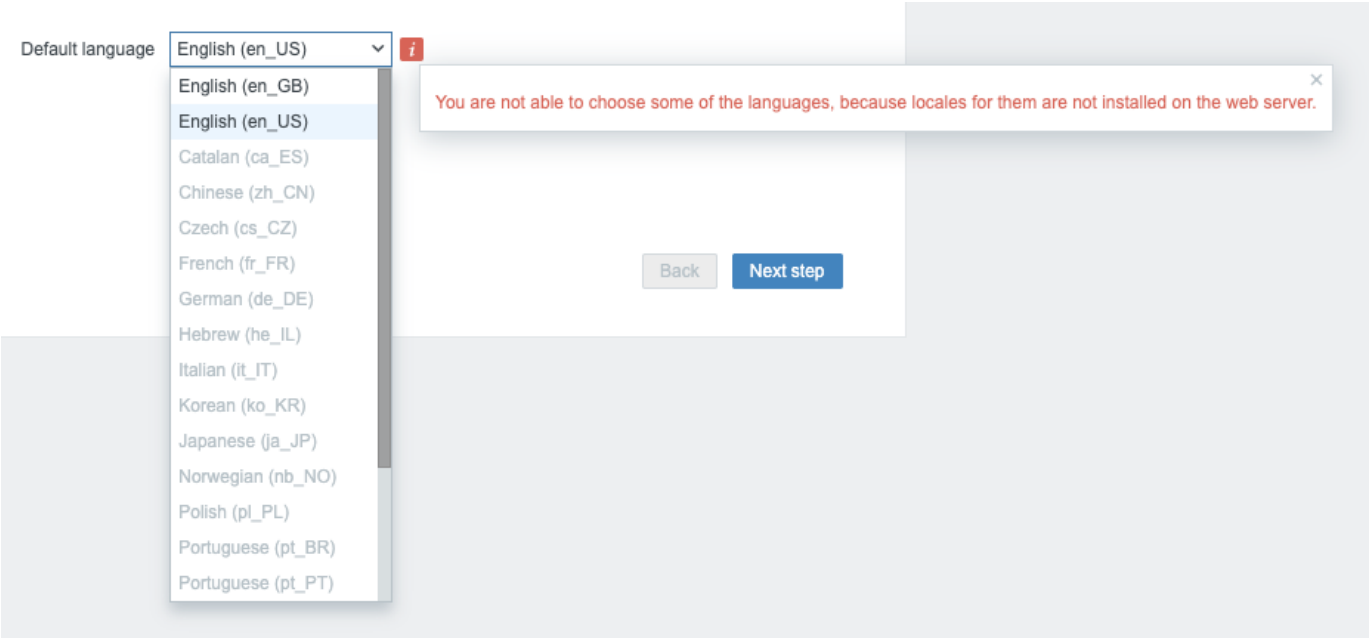
or run

`journalctl -xe`

This should help you in locating the errors you made.

When you point your browser to the correct URL you should be greeted with a page like here :

As you see there is only a limited list of local translations available on our Zabbix frontend to choose from



What if we want to install Chinese as language or another language from the list ? Run the next command to get a list of all locales available for your OS.

```
dnf list glibc-langpack-*
```

This will give you a list like

```
Installed Packages
glibc-langpack-en.x86_64
Available Packages
glibc-langpack-aa.x86_64
...

glibc-langpack-zu.x86_64
```

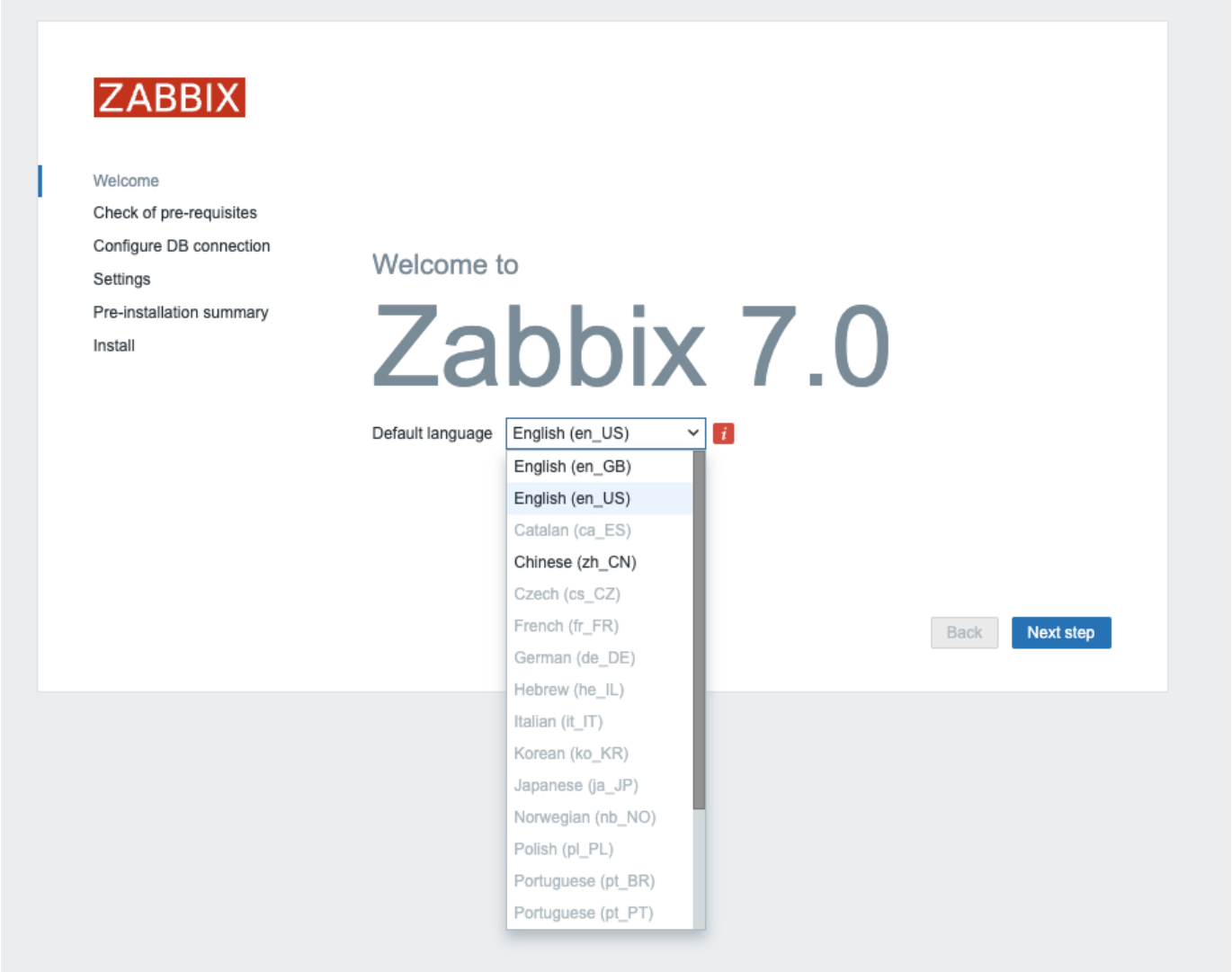
Let's search for our Chinese locale to see if it is available. As you can see the code starts with zh

```
# dnf list glibc-langpack-* | grep zh
glibc-langpack-zh.x86_64
glibc-langpack-lzh.x86_64
```

The command returns us 2 lines but as we have seen that the code was zh_CN we only have to install the first package.

```
# dnf install glibc-langpack-zh.x86_64 -y
```

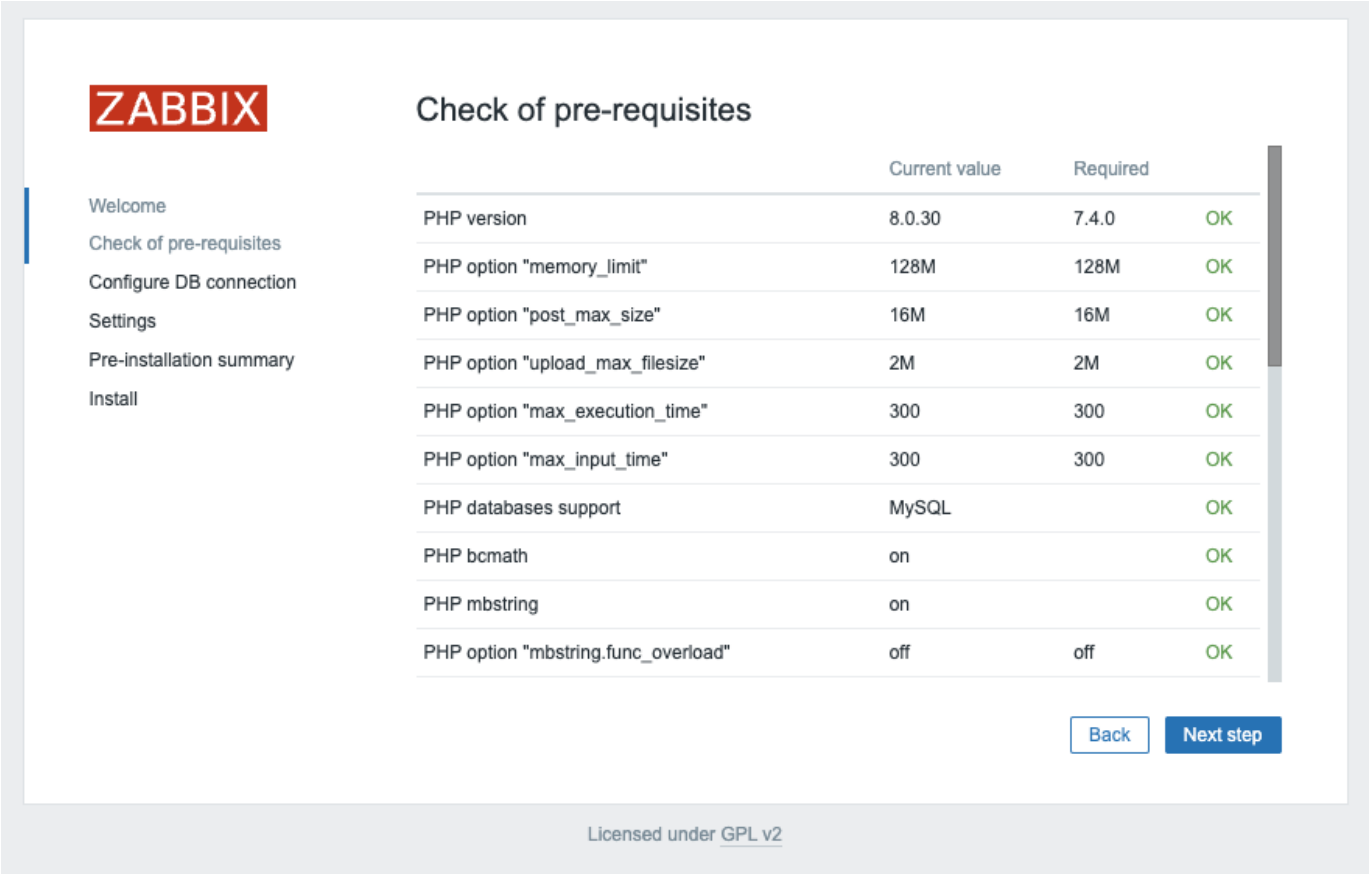
When we return now to our frontend we are able to select the Chinese language.



Note

If your language is not available in the frontend don't panic it just means that there is no translation or that the translation was not 100% complete. Zabbis is free and relies on the community for it's translations so you can help in creating the translation. Go to the page `https://translate.zabbix.com/` and help us to make Zabbix get better. Once the translation is complete the next Zabbix minor patch version should have your language included.

Click next when you are satisfied with the transaltions available. You will arrive at a screen to verifiy if all pre-requisites are met. If not fix them first but normaly it should be fine and you should be just able to click Next



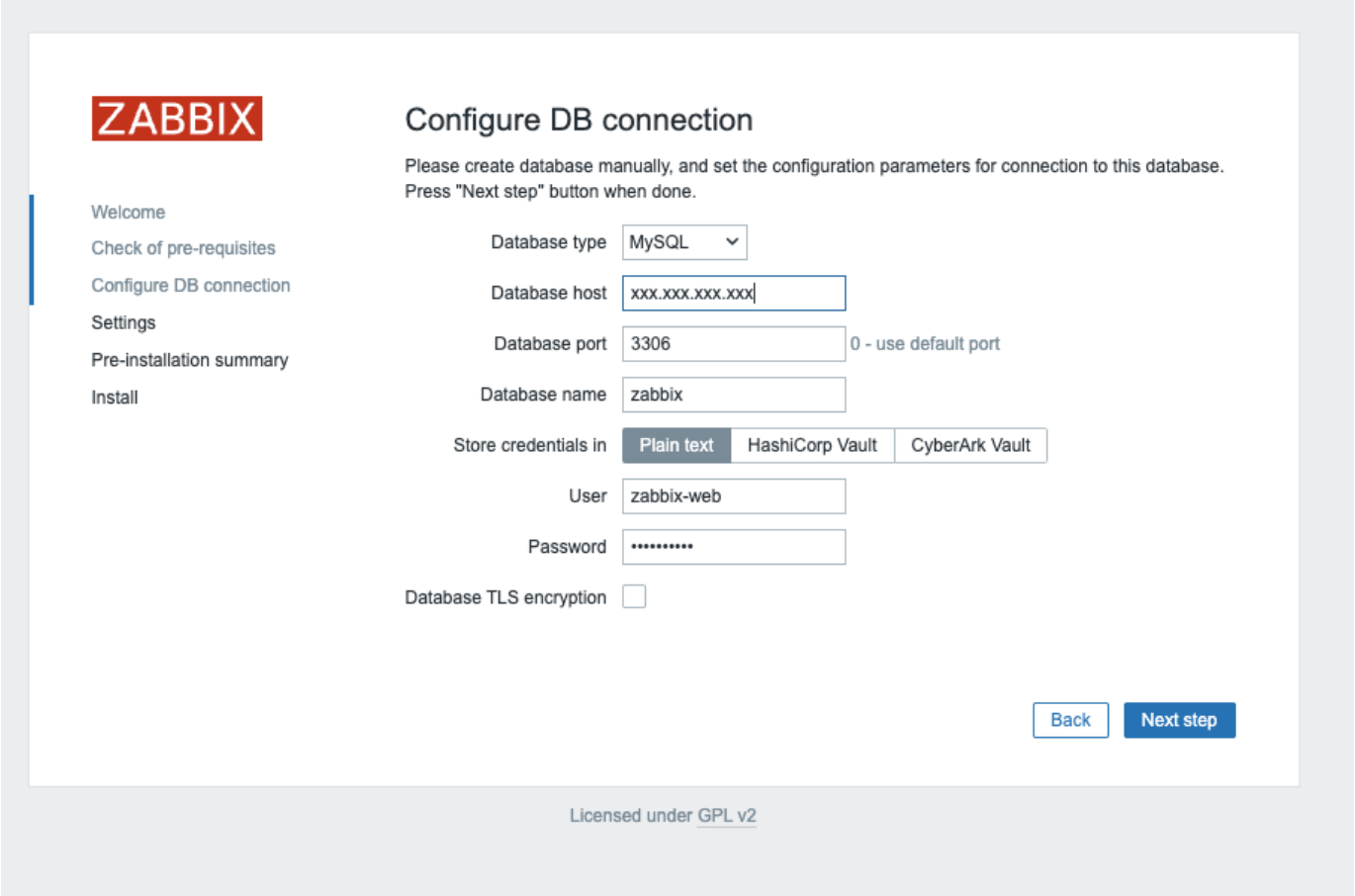
The next page will show you a page with the connection parameters for our database.

First you select your DB type 'MySQL' or 'PostgreSQL' and fill in the IP or DNS name of the location of your database server. Use port 3306 for MariaDB/MySQL or 5432 if you used PostgreSQL.

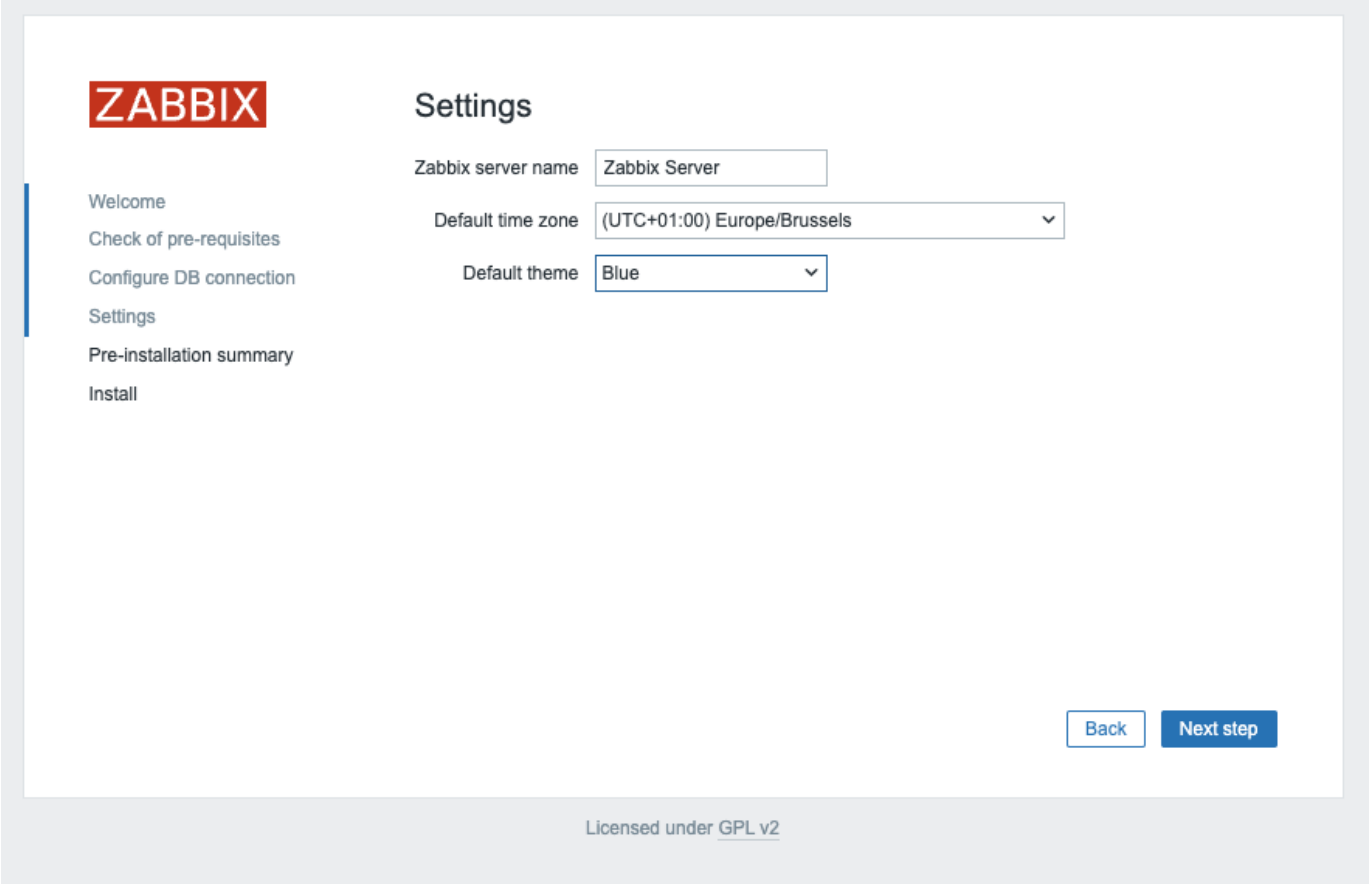
Fill in the correct database name, in our case it was `zabbix`. If you used PostgreSQL then you also need to fill in the correct schema name in our case it was `zabbix_server`

Next line will ask you for the DB users here we created a user `zabbix-web`. Enter it in the correct field and fill in the password that you used for this user.

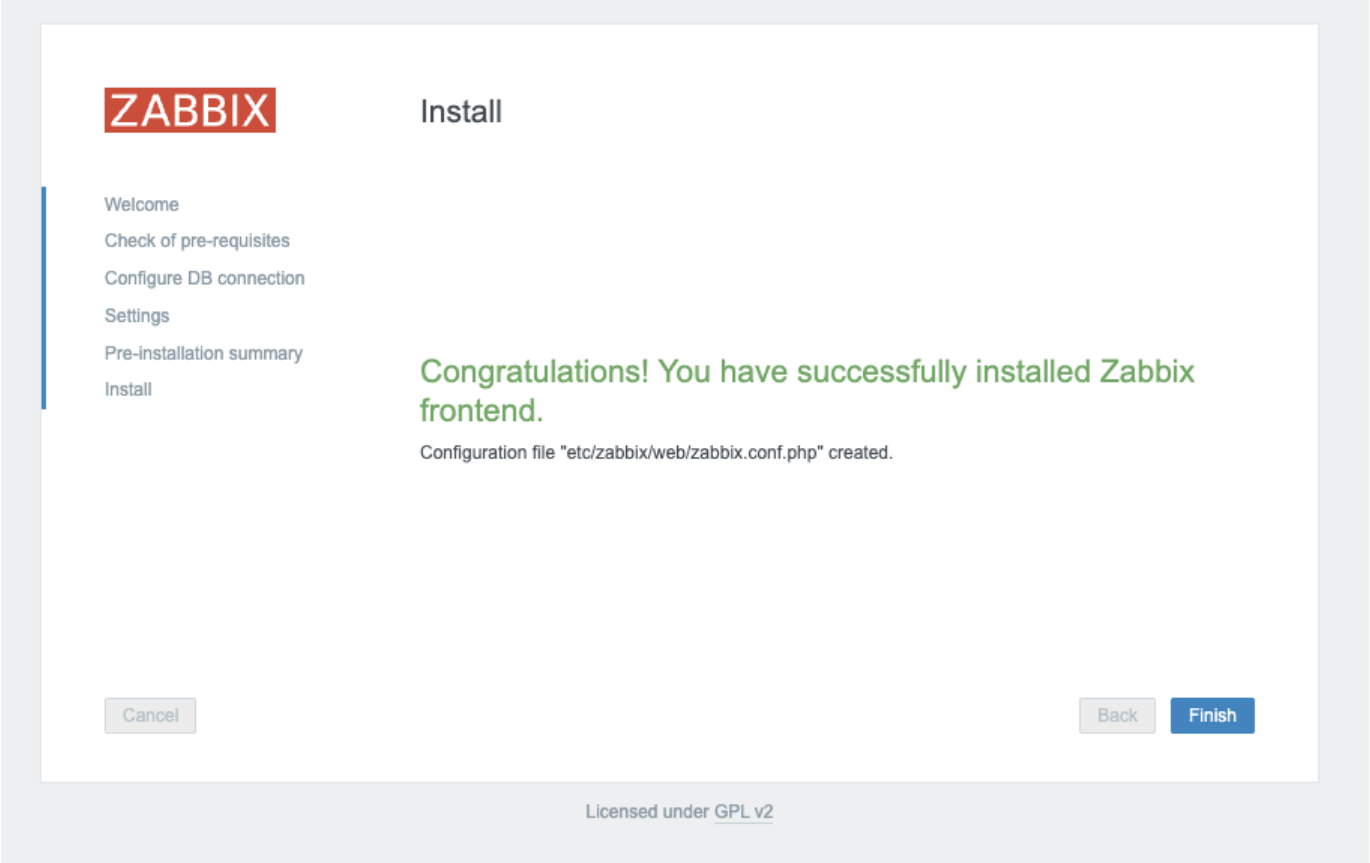
Make sure the option `Database TLS encryption` is not selected and press `Next step`.



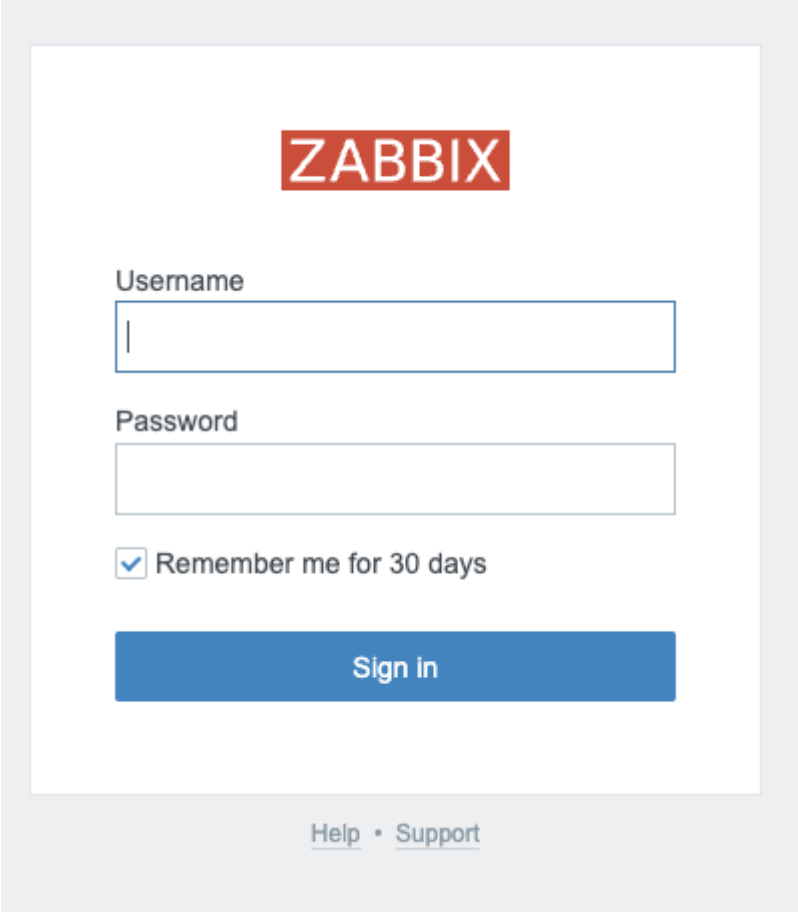
We are almost there. The only thing that rests us to do is give our instance a name, select our timezone and select a default time we like to use.



Press `Next step` again you will see a page that tells you that the configuration is successful. Press Finish to end the configuration.



We are now ready to login :



Login : Admin Password : zabbix

In case you are like me and don't like the /zabbix path at the end of you url then there is an easy way to remove this. Edit you `httpd` config file and add the lines below and replace it with your own domain:

```
vi /etc/httpd/conf/httpd.conf
```

```
NameVirtualHost 172.1.11.21:80

<VirtualHost "your ip or dns":80>
  ServerName zabbixserver.mydomain.org
  ServerAlias zabbixserver
  DocumentRoot /usr/share/zabbix
</VirtualHost>
```

Don't forget to restart the `httpd` service

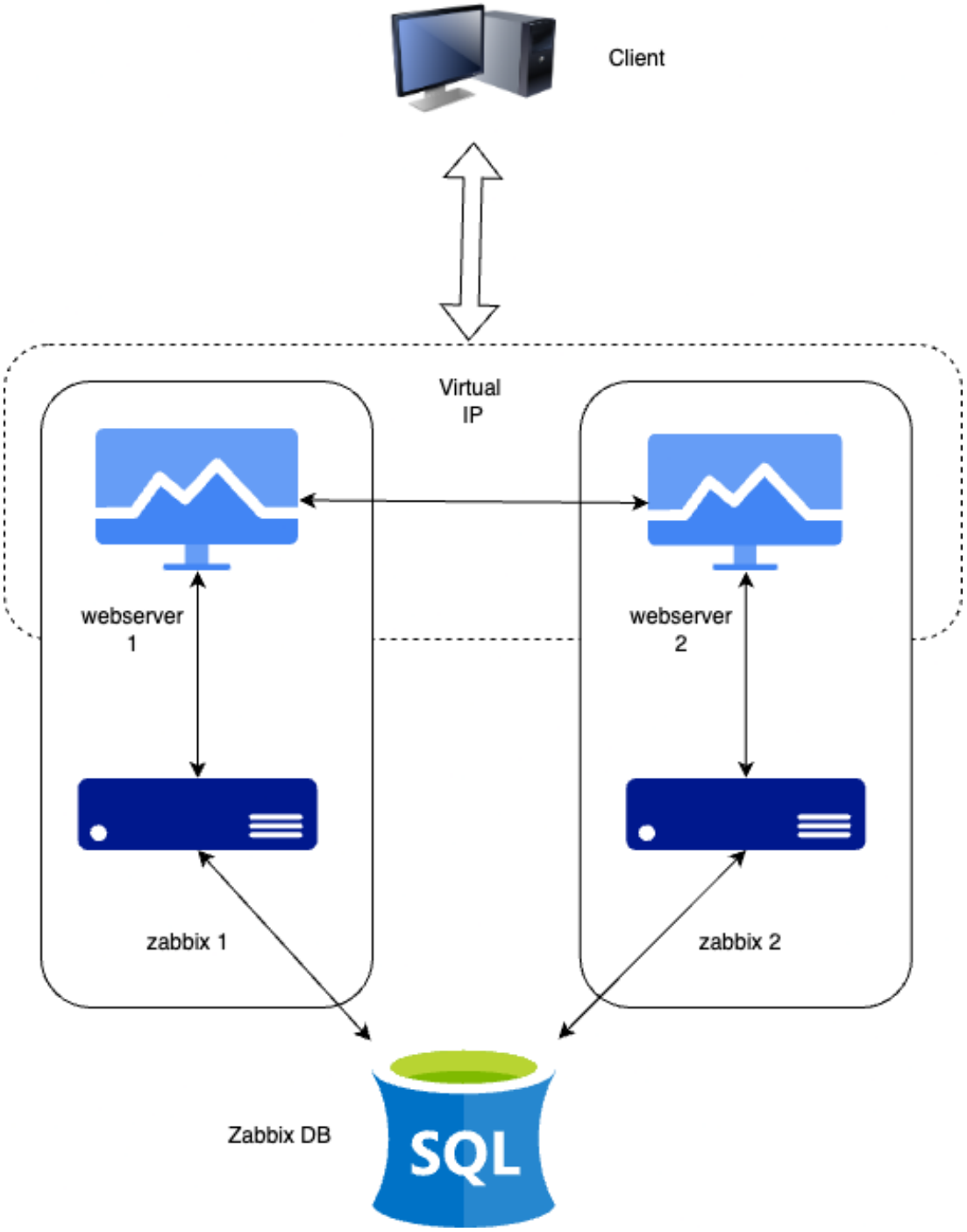
```
systemctl restart httpd
```

6.4 Configure Zabbix HA

In this topic we will setup Zabbix in a High Available setup. This feature was added in Zabbix 6 and was one of the most important features added that time. The idea about this functionallity is that if your Zabbix server fails that another Zabbix server can take over. In this setup we will use 2 Zabbix servers but you are not limited to this you can add as many as you like.

The HA setup in Zabbix is rather basic but works like a charm so don't expect fancy things like load balancing.

Just like we did in our basic setup we will make a few notes again about the setup of the servers we have. I added the IP's that we will use here don't forgot to make notes of your own ip adresses.



Server	IP
Zabbix Server 1	192.168.0.130
Zabbix Server 2	192.168.0.131
Postgres DB	192.168.0.132
Virtual IP	192.168.0.135

> **Note**
>
> As you notice our DB is not HA this is not a Zabbix component you have to implement your own solution this can be a HA SAN or you DB in a HA cluster setup. The cluster setup of our DB is out of the scope and not related to Zabbix so we will not cover this here.

6.4.1 Let's install our Postgres DB

> **Note**
>
> If you are not running on x86 or like to try on another OS, then have a look at https://www.postgresql.org/download/ for the commands you need.

> **Warning**
>
> In this exercise we will take some shortcuts for the installation of the PostgreSQLDB and the OS. Look at our previous topics to get a better understanding where to tweak.

```
# Install the repository RPM:
sudo dnf install -y https://download.postgresql.org/pub/repos/yum/reporpms/EL-9-x86_64/pgdg-redhat-repo-latest.noarch.rpm

# Disable the built-in PostgreSQL module:
sudo dnf -qy module disable postgresql

# Install PostgreSQL:
sudo dnf install -y postgresql16-server

# Initialize the database and enable automatic start:
sudo /usr/pgsql-16/bin/postgresql-16-setup initdb
sudo systemctl enable postgresql-16 --now
```

Securing the PostgreSQL database¶

PostgreSQL works a bit different then MySQL or MariaDB and this applies aswell to how we manage access permissions. Postgres works with a file with the name `pg_hba.conf` where we have to tell who can access our database from where and what encryption is used for the password. So let's edit this file to allow our frontend and zabbix server to access the database.

```
# vi /var/lib/pgsql/16/data/pg_hba.conf
```

```
# "local" is for Unix domain socket connections only
local   all             all                                     trust
# IPv4 local connections:
host    zabbix          zabbix     192.168.0.130/32         scram-sha-256
host    zabbix          zabbix     192.168.0.131/32         scram-sha-256
host    all             all            127.0.0.1/32             scram-sha-256
```

After we changed the pg_hba file don't forget to restart postgres else the settings will not be applied. But before we restart let us also edit the file postgresql.conf and allow our database to listen on our network interface for incomming connections from the zabbix server. Postgresql will standard only allow connections from the socket.

```
# vi /var/lib/pgsql/16/data/postgresql.conf
```

Replace the line with listen_addresses so that PostgreSQL will listen on all interfaces and not only on our localhost.

```
listen_addresses = '*'
```

When done restart the PostgreSQL cluster and see if it comes back online in case of an error check the `pg_hba.conf` file you just edited for typos.

```
# systemctl restart postgresql-16
```

For our Zabbix server we need to create tables in the database for this we need ot install the Zabbix repository like we did for our Zabbix server and install the Zabbix package containing all the database tables images icons, ...

ADD THE ZABBIX REPOSITORY AND POPULATE THE DB

Add the Zabbix repo to your server (Don't forget to select the correct repo for your OS and Zabbix version) for this go to www.zabbix.com/download

```
# rpm -Uvh https://repo.zabbix.com/zabbix/7.0/rhel/9/x86_64/zabbix-release-6.5-2.el9.noarch.rpm
```

Install the database scripts.

```
# dnf install zabbix-sql-scripts -y
```

Next we have to unzip the database schema files. Run as user root followin command::

```
# gzip -d /usr/share/zabbix-sql-scripts/postgresql/server.sql.gz
```

Create the DB users

Now we are ready to create our Zabbix users for the server and the frontend. If you like to separate users for frontend and server have a look at the basic installation guide.

```
# su - postgres
# createuser --pwprompt zabbix
Enter password for new role: <server-password>
Enter it again: <server-password>
```

POPULATE THE DATABASE.

We are now ready to create our database zabbix. Become user postgres again and run next command to create the database as our user zabbix:

```
# su - postgres
# createdb -E Unicode -O zabbix  zabbix
```

Let's upload the Zabbix SQL file we extracted earlier to populate our database with the needed schemas images users etc ... For this we need to connect to the DB as user zabbix.

```
# psql -U zabbix -W zabbix
Password:
psql (16.2)
Type "help" for help.

zabbix=> SELECT session_user, current_user;
 session_user | current_user
--------------+--------------
 zabbix       | zabbix
(1 row)

zabbix=> \i /usr/share/zabbix-sql-scripts/postgresql/server.sql
CREATE TABLE
CREATE INDEX
CREATE TABLE
....
....
INSERT 0 1
DELETE 80424
COMMIT
```

Make sure the owner of your tables is the user zabbix;

```
zabbix=> \dt
                  List of relations
 Schema |            Name            | Type  | Owner
--------+----------------------------+-------+--------
 public | acknowledges               | table | zabbix
 public | actions                    | table | zabbix
...
...
...

zabbix=> \q
```

Configure the firewall¶

One last thing we need to do is open the firewall and allow incoming connections for the PostgreSQL database from our Zabbix server because at the moment we dont accept any connections yet.

```
# firewall-cmd --new-zone=postgresql-access --permanent
success

# firewall-cmd --reload
success

# firewall-cmd --get-zones
block dmz drop external home internal nm-shared postgresql-access public trusted work

# firewall-cmd --zone=postgresql-access --add-source=<zabbix-serverip 1> --permanent
# firewall-cmd --zone=postgresql-access --add-source=<zabbix-serverip 1> --permanent

success
# firewall-cmd --zone=postgresql-access --add-port=5432/tcp  --permanent

success
# firewall-cmd --reload
```

Now lets have a look to our firewall rules to see if they are what we expected:

```
# firewall-cmd --zone=postgresql-access --list-all
```

Our database server is ready now to accept connections from our Zabbix server :). You can continue with the next task

6.4.2 Install our Zabbix Cluster

Setting up a Zabbix cluster is not really different from setting up a regular Zabbix server obviously we need more then one. And there are also a few parameters that we need to configure.

Let's start by adding our Zabbix 7.0 repositories to our 2 Zabbix servers.

```
rpm -Uvh https://repo.zabbix.com/zabbix/7.0/rhel/9/x86_64/zabbix-release-6.5-2.el9.noarch.rpm
```

Once this is done we can install our Zabbix servers on both systems.

```
dnf install zabbix-server-pgsql -y
```

We will now edit the config file on our first zabbix server. Run the next command:

```
vi /etc/zabbix/zabbix_server.conf
```

Once in the file edit the following lines to make our Zabbix server 1 connnect to the database

```
DBHost=<zabbix db ip>
DBName=zabbix
DBUser=zabbix
DBPassword=<your secret password>
```

In the same file we also have to edit another parameter to activate HA on this host.

```
HANodeName=zabbix1 (or whatever you like)
```

We are not done yet. We also have to tell Zabbix in case of a node fail to what server the frontend needs to connect.

```
NodeAddress=<Zabbix server 1 ip>>:10051
```

We are now done with the configuration of our 1st Zabbix server. Now let's do the same for our second server. I case you have more then 2 servers you can update them as well.

When you are done patching the config of your servers you can start the `zabbix-server` service on both servers

```
systemctl enable zabbix-server --now
```

Let's have a look at the log files from both servers to see if it came online as we had hoped. on our first server we can run:

```
#grep HA /var/log/zabbix/zabbix_server.log

22597:20240309:155230.353 starting HA manager
22597:20240309:155230.362 HA manager started in active mode
```

Now do the same on our other node(s)

```
# grep HA /var/log/zabbix/zabbix_server.log
22304:20240309:155331.163 starting HA manager
22304:20240309:155331.174 HA manager started in standby mode
```

6.4.3 Installing the frontends

First things first before we can install and configure our webserver we need to install keepalived. Keepalived allows us to use a VIP for our frontends. Keepalived provides frameworks for both load balancing and high availability.

Info

Some useful documentation on the subject you might like. https://www.redhat.com/sysadmin/advanced-keepalived and https://keepalived.readthedocs.io/en/latest/introduction.html

Setup keepalived

So let's get started. On both our servers we have to install keepalived.

```
dnf install keepalived
```

We also need to adapt the configuration of keepalived on both servers. The configuration for both servers needs to be a bit changed so let's start with our server 1. Edit the config file with the following command:

```
# vi /etc/keepalived/keepalived.conf
```

Delete everything and replace it with the following lines:

```
vrrp_track_process track_nginx {
    process nginx
    weight 10
}

vrrp_instance VI_1 {
    state MASTER
    interface enp0s1
    virtual_router_id 51
    priority 244
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 12345
    }
    virtual_ipaddress {
        192.168.0.135
    }
    track_process {
         track_nginx
      }
}
```

Replace `enp0s1` with the interface name of your machine and replace the password with something secure. For the virual_ipaddress use aa free IP from your network. Now do the same thing for our second Zabbix server.

```
# vi /etc/keepalived/keepalived.conf
```

Delete everything and replace it with the following lines:

```
vrrp_track_process track_nginx {
      process nginx
      weight 10
}
```

```
vrrp_instance VI_1 {
    state BACKUP
    interface enp0s1
    virtual_router_id 51
    priority 243
    advert_int 1
    authentication {
        auth_type PASS
        auth_pass 12345
    }
    virtual_ipaddress {
        192.168.0.135
    }
    track_process {
         track_nginx
      }
}
```

Just as with our 1st Zabbix server, replace `enp0s1` with the interface name of your machine and replace the password with something secure. For the virual_ipaddress use aa free IP from your network.

Install and configure the frontend

On both servers we can run the following commands to install our webserver and the zabbix frontend packages:

```
dnf install nginx zabbix-web-pgsql zabbix-nginx-conf
```

Also let's not forget to configure our firewall

```
firewall-cmd --add-service=http --permanent
firewall-cmd --add-service=zabbix-server --permanent
firewall-cmd --reload
```

And now we can start our keepalived and enable it so that it comes up next reboot

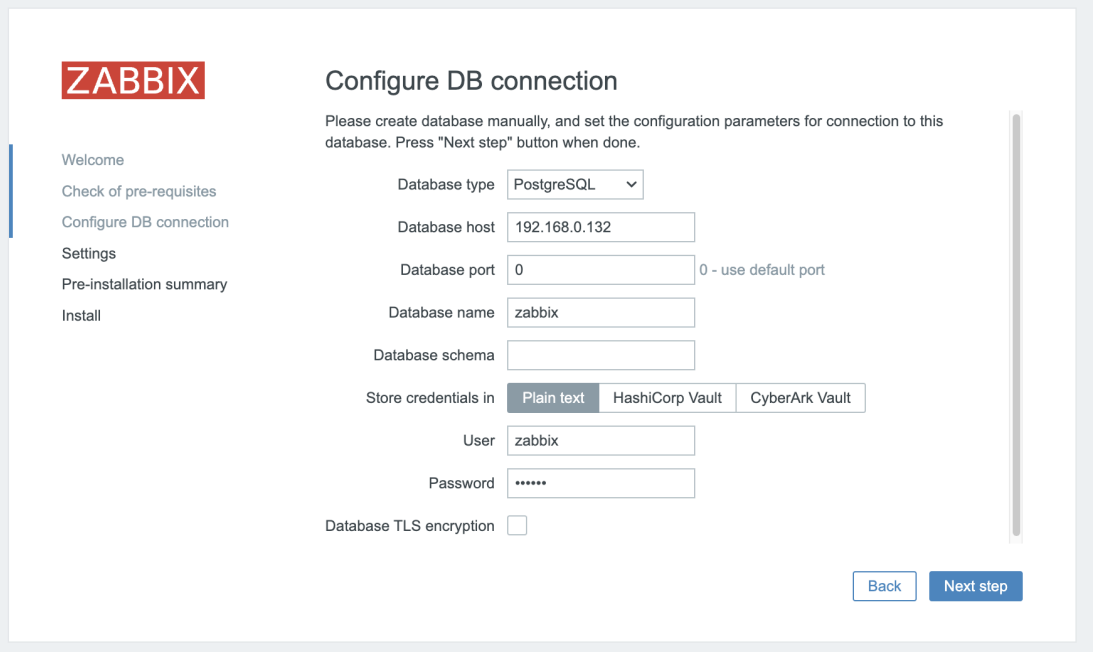
```
systemctl enable keepalived nginx --now
```

Configure Zabbix Frontend

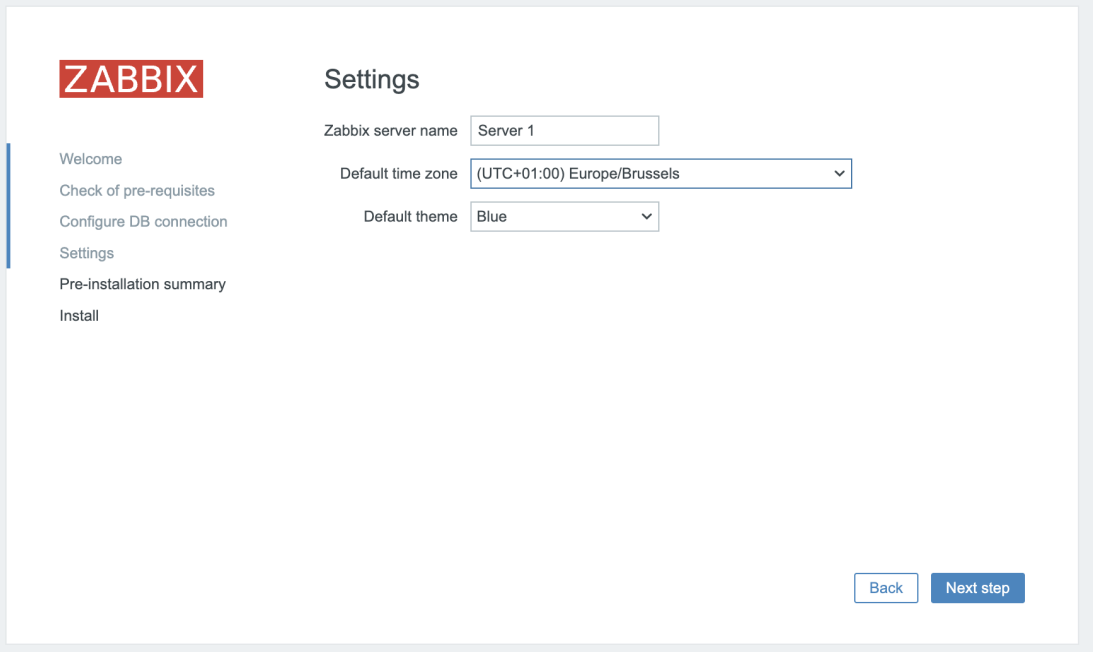
Click next till you see the following page and fill in the ip of your DB server. The port can be 0 this means we will use the default port. fill in the database name, user and password you used for the database. Make sure you deselect TLS encryption and select store passwords as plaintext. When you click next it won't work because we did not disable SELinux. Run the following command first on both Zabbix servers.

```
setsebool -P httpd_can_network_connect_db on
setsebool -P httpd_can_connect_zabbix on
```

This will allow your webservers to communicate with our database over the network. Now when we click next it should work.



We are almost ready the only thing left here is now to add the name of our server and configure the default timezone.



Since you're using a host-based firewall, you need to add the necessary rules to permit IP protocol 112 traffic. Otherwise, Keepalived's advertisement method won't work.

```
firewall-cmd --add-rich-rule='rule protocol value="112" accept' --permanent
```

Now that this is all taken care of stop keepalived on our server and repeat the same steps on the second server. After this is finished start keepalived again.

Congratulations you have a HA Zabbix server now .

CHECKING THE DATABASE FOR HA INFO.

Now that everything is up and running there is probably something you like to know. Where can we find the info in our database ?

It's actually very straighforward we can go to our zabbix database and run the following query to see our servers: `SELECT *FROM ha_node;`

```
zabbix=# SELECT *FROM ha_node;
         ha_nodeid         |  name   |    address    | port  | lastaccess | status |       ha_sessionid
---------------------------+---------+---------------+-------+------------+--------+---------------------------
 cltk7h2n600017kkd1jtx6f1f | zabbix2 | 192.168.0.131 | 10051 | 1710085786 |      0 | cltlov4ly0000jkkdteikeo77
 cltk7ci340001inkc2befwg9f | zabbix1 | 192.168.0.130 | 10051 | 1710085787 |      3 | cltlov1r00000jtkcpeh9oqhp
```

This is also how our frontend is able to know what server it needs to connect to. Remember our picture in the first page ? Actually the frontend has a connection to our database and reads out the status from our `zabbix server`. This way it knows what server is active.

It's probably also good to know that we can have 4 statusses:

status	number	info
Active	3	Only one node can be active
Standby	0	Multiple nodes can be in standby
Stopped	1	A previous detected node is nog stopped
Unavailable	2	A previous dtected node was lost without being shutdown

Note

Zabbix agents need to have their Server and ServerActive addresses pointed to both active and passive Zabbix server. This option is supported in all agents since Zabbix 6.0

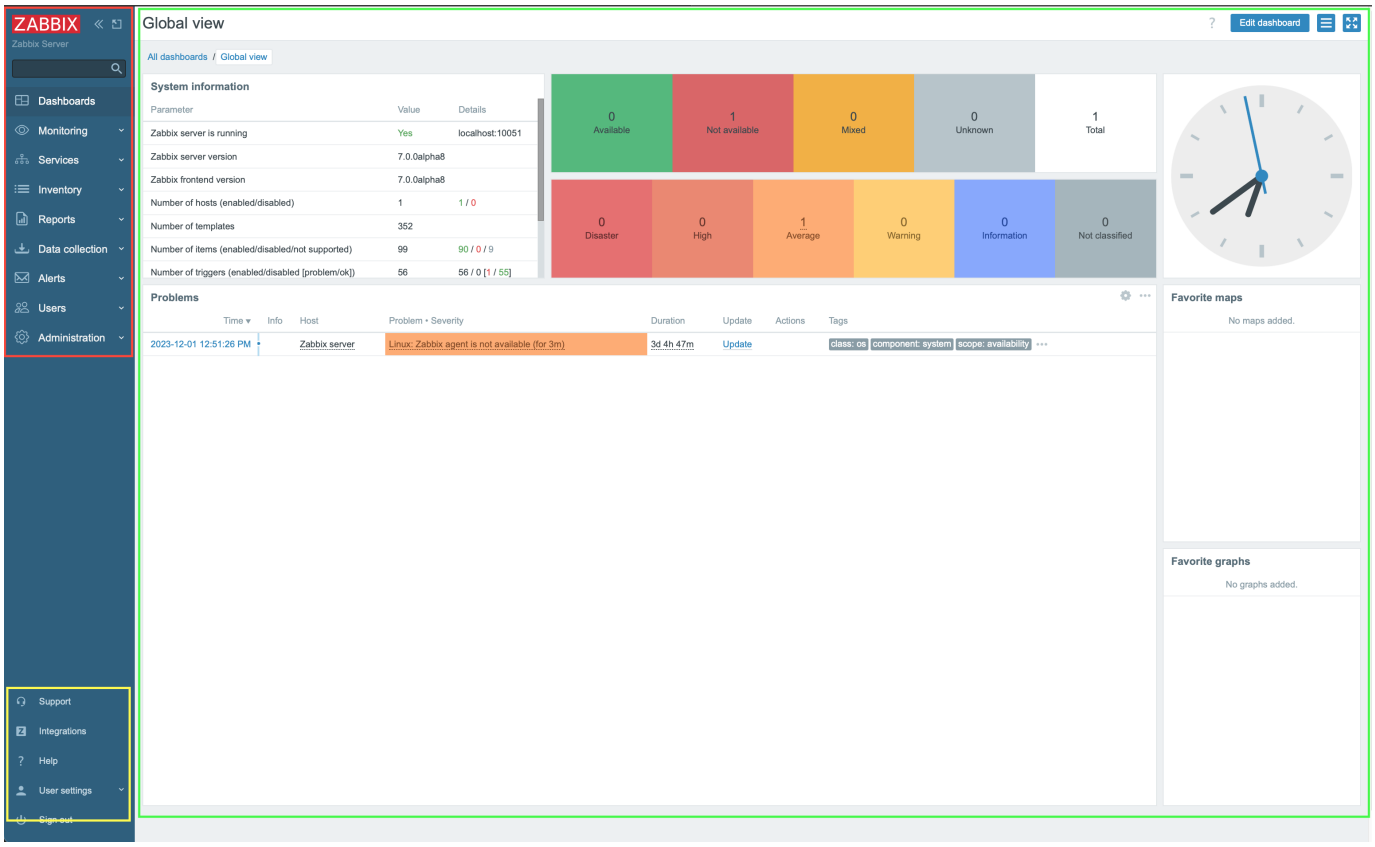
7. The basics

7.1 Zabbix Interface

This chapter is going to cover the basics we need to know when it comes to the Zabbix userinterface and the thing we need to know before we can start to fully dive into our monitoring tool. We will see how the userinteface works how to add a host, groups users, items ... so that we have a good understanding of the basics. This is something that is sometimes missed and can lead to frustrations not knowing why things don't work like we had expected them to work. So even if you are an advanced user it may be usefull to have a look into this chapter.

Overview of the interface

With Zabbix 7 the user interface after logging in is a bit changed. Our menu on the left side of the screen has has a small overhaul. Let's dive into it. When we login into our Zabbix setup the first time with our Admin user we see a page like this where we have our `main window` in green our `main menu` marked in red and our `links` marked in yellow.



The main menu can be hidden by collapsing it completely or to reduce it to a set of small icons.

When we click on the button with the 2 arrows to the left:

You will see that the menu collapses to a set of small icons. Pressing ">>" will bring the `main menu` back to it's original state. Pressing the box with the arrow sticking out next to the "<<" button will hide the `main menu` completely.

To get the main menu back it's not too difficult we just look for the button on the left with three horizontal lines and click it. This will bring the menu back and clicking on the box with the arrow agian will bring the `main menu` back.

Yet another way to make the screen bigger that is quit useful for monitors in NOK teams for example is the kiosk mode button. This one is however located on the left side of your screen and looks like 4 arrows pointing to every corner of the screen. Pressing this button will remove all the menus and leave only main window to focus on.

When wanting to leave the kios mode the button will be changed to 2 arrows poiting to the inside of the screen. Pressing this button will revert us back to the original state.

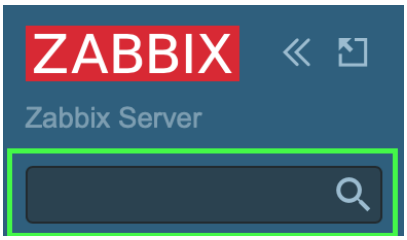
Tip

We can also enter and exit kiosk mode by making use of parameters in our Zabbix url: `/zabbix.php?action=dashboard.view&kiosk=1` - activate kiosk mode or `/zabbix.php?action=dashboard.view&kiosk=0` - activate normal mode

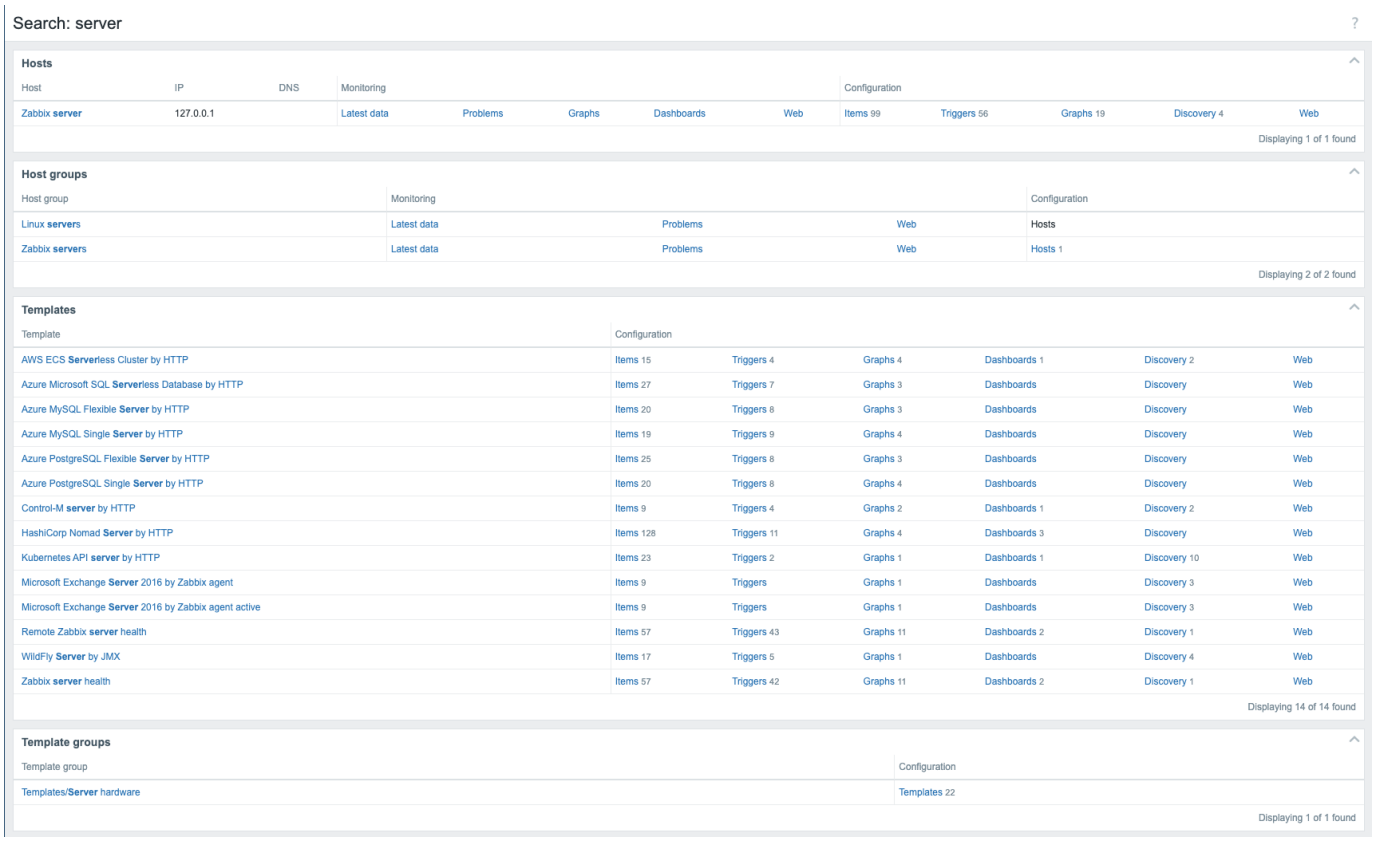
Note

There are many other page parameters we can use. A full list can be found here

Zabbix also has a global search menu that we can use to find hosts, host groups and templates.



If we look in the search box for `server` you will see that we get an overview of all templates, host groups and hosts with the name `server` in it.



7.1.1 Main menu

Our main menu on the left consists of a few sections, 9 to be exact:

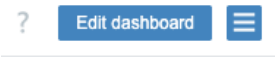
Menu Name	Details
Dashboards	Contains an overview of all the dashboards we have access to.
Monitoring	Shows us the hosts, problems, latest data, maps, ...
Services	An overview of all the Services and SLA settings.
Inventory	An overview of our collected inventory data.
Reports	Shows us the system information, scheduled reports, audit logs, action logs, etc .
Data collection	Contains all things related to collecting data like hosts, templates, maintenance, discovery, ...
Alert	The configuration of our media types, scripts and actions
Users	User configuration like user roles, user groups, authentication, API tokes, ...
Administration	The administration part containing all global settings, housekeeper, proxies, queue, ...

7.1.2 Links menu

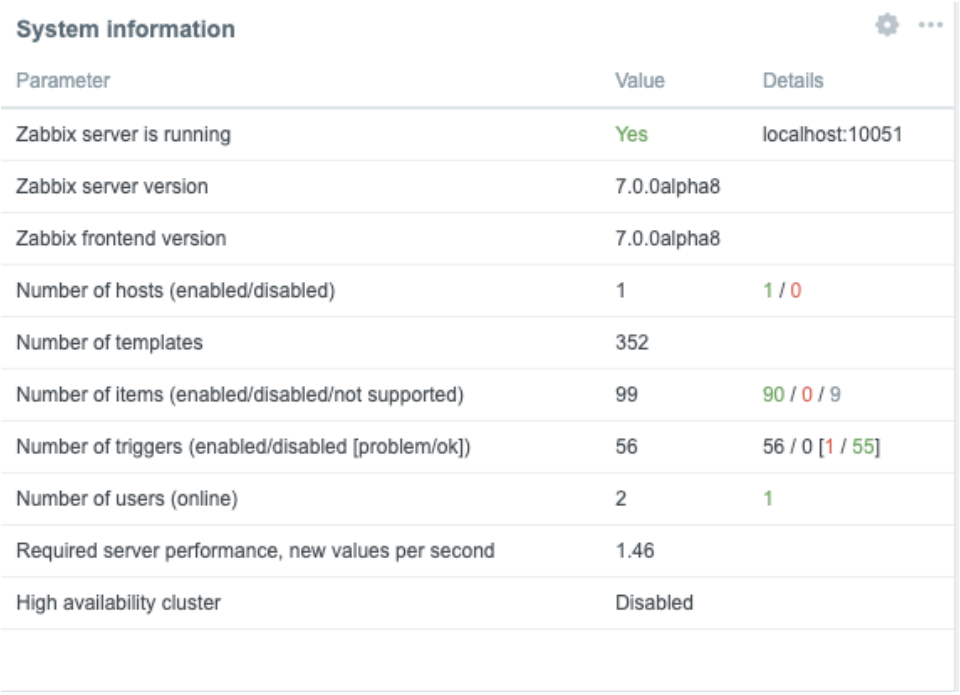
Our last part the `links` part contain a set of useful links that we can use:

Menu name	Details
Support	This brings us to the technical support page that you can buy from Zabbix. Remember that your local partner is also able to sell these contracts and can help you in your own language. Your local distributors
Integrations	The official zabbix integeration page
Help	The link to the documentation of your Zabbix version
User settings	The user profile settings.
Sign out	Log out of the current session.

There are still a few buttons that we need to cover on the right side of our screen



The edit button allows us to change our dashboard. This is something we will cover later. On the far left side there is a "?" this will bring you to the Zabbix documentation page that explains everything about the dashboard. The button on the right side with the 3 horizontal lines is the one to share, rename, delete, ... our dashboards.



System information

Parameter	Value	Details
Zabbix server is running	Yes	localhost:10051
Zabbix server version	7.0.0alpha8	
Zabbix frontend version	7.0.0alpha8	
Number of hosts (enabled/disabled)	1	1 / 0
Number of templates	352	
Number of items (enabled/disabled/not supported)	99	90 / 0 / 9
Number of triggers (enabled/disabled [problem/ok])	56	56 / 0 [1 / 55]
Number of users (online)	2	1
Required server performance, new values per second	1.46	
High availability cluster	Disabled	

7.1.3 System Information

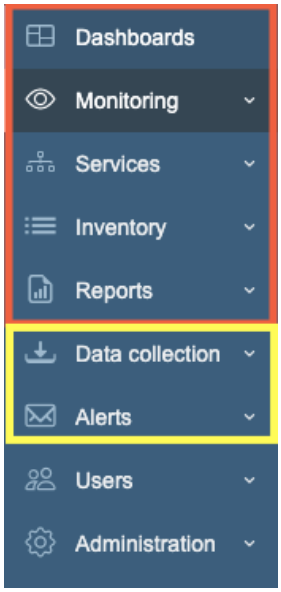
There is also a box on the dashboard called `System Information`. This widget will show you the current System status of your Zabbix setup. Let's go over the different lines of information as they are important to understand.

Parameter	Value	Details
Zabbix server is running	This gives us the status of our zabbix server if it is running yes or no and if it is running on our localhost or another IP and on what port the zabbix server is listening. If no trapper is listening the rest of the information can not be displayed	IP and port of the Zabbix server
Zabbix server version	This shows us the version of the `Zabbix server` so the version you see at the bottom of your screen is the one from the Zabbix frontend and can be different but should be in the same major version.	Version Number
Zabbix frontend version	This is the version of the frontend and should match with what you see at the bottom of your screen.	Version Number
Number of hosts (enabled/disabled)	The total number of hosts configured on our system	How many of those are enabled and disabled
Number of templates	The number of templates installed on our Zabbix server.	
Number of items (enabled/disabled/not supported)	This line shows us the number of items we have configured in total in this case 99	90 are enabled and 0 are disabled but 9 of them are unsupported. This last number is important as those are items not working. We will look into this later why it happens and how to fix it. For now remember that a high number of unsupported items is not a good idea.
Number of triggers (Enabled/ disabled[problem/ok])	The number of triggers configured	Number of enabled and disabled triggers. Just as with items we also see if there are triggers that are in a problem state or ok state. A trigger in a problem state is a non working trigger something we need to monitor and fix. We will cover this also later.
Number of users (online)	Here we see the number of users that are configured on our system	The nunber of users currently online.
Required server performance, nvps	The number of new values per second that Zabbix will process per second.	This is just an estimated number as some values we get are unknown so the real value is probably higher. So we can have some indication about how many IOPS we need and how busy our database is. A better indication is probably the internal item `zabbix[wcache,values,all]`
High availability cluser	It will show us if we are running on a Zabbix HA cluster or not	Failover delay once HA is activated

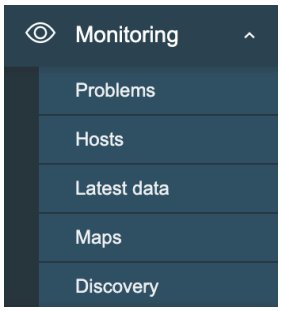
System information may display some additonal warnings like when your database doesnt have the correct character set or collation UTF-8. Also when the database you used is lower or higher then the recommended version or when there are misconfigurations on housekeeper or TimescaleDB. Another warning you can see is about database history tables that aren't upgraded or primary keys that have not been set. This is possible if you are coming from an older version before Zabbix 6 and never did the upgrade.

7.1.4 The main menu explained

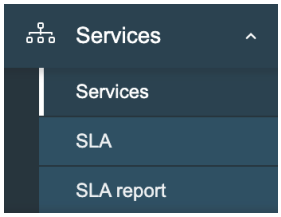
It's important to know that we have so far seen our dashboard with the Admin user and that this user is a `Zabbix Super Admin` user. This means that the user has no restrictions. Zabbix works with 3 different levels of users we have the regular users, Zabbix Admin and Zabbix Super Admin users. Let's have a look



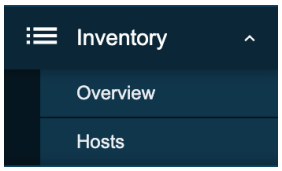

```
* A ```Zabbix User``` will only see the <font color='red'>red</font> part of our ```main menu``` and will only be able to see our collected data.
* A ```Zabbix Admin``` will see the red part and the <font color='gold'>yellow</font>  part of the ```main menu``` and is able to change our configuration.
* A ```Zabbix Super Admin``` will see the complete ```main menu``` and so is able to change the configuration and all the global settings.
```



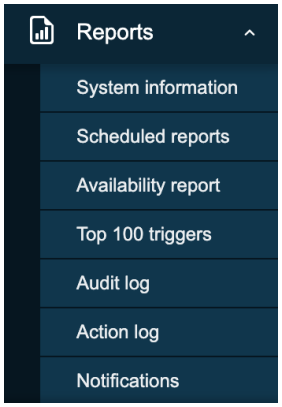

```
- Problems: This page will give us an overview of all the problems. With filter we can look at recent problems past problems and problems that are active now.
There are many more filters tor drill down more.
- Hosts: This will give us a quick overview page with whats happening on our hosts and allows us to quickly go to the latest data, graphs and dashboards.
- Latest data: This page I probably use the most, it shows us all the information collected from all our hosts.
- Maps: The location where we can create map that are an oveview of our IT infrastructure very useful to get a high level overview of the network.
- Discovery: When we run a network discovery this is the place where we can find the results.
```



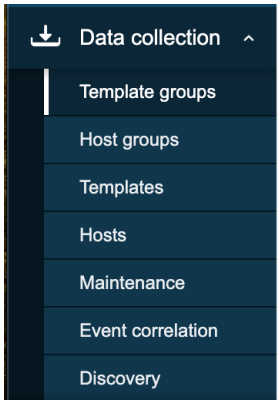

```
- Services This page will give us a high level overview of all services configured in Zabbix.
- SLA: An overview of all the SLAs configured in Zabbix.
- SLA Report: Here we can watch all SLA reports based on our filters.
```



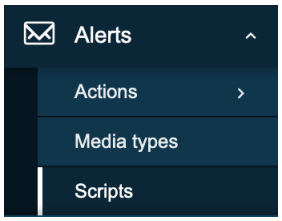

```
- Overview: A place where we can watch all our iventory data that we have retrieved from our hosts.
- Hosts: Here we can filter by host and watch all inventory data for the hosts we have selected.
```



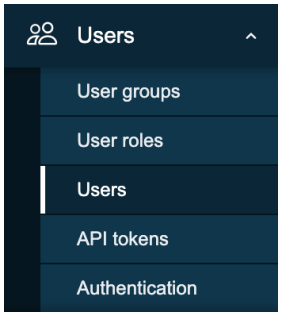

```
- System information: System information is a summary of key Zabbix server and system data.
- Scheduled reports: The place where we can schedule our reports, a pdf of the dashboard that will be sent at a specified time and date.
- Availability report: A nice overview where we can see what trigger has been in ok/nok state for how much % of the time
- Top 100 triggers: Another page I visit a lot here we have our top list with triggers that have been in a nok state.
- Audit log: An overview of the user activity that happend on our system. Useful if we want to know who did what and when.
- Action log: A detailed overview of our actions can be found here. What mail was sent to who and when ...?
- Notifications: A quick overview of the number of notifications sent to each user.
```



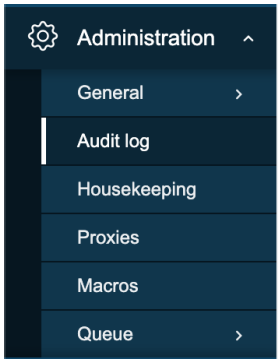

```
- Template groups: A place to logical group all templates together in different groups. Before it was mixed together with hosts in host groups.
- Host groups: A logical collection of different hosts put together. Host groups are used for our permissions.
- Templates: A set off entities like items and triggers can be grouped together on a template, A template can be applied to one or more hosts.
- Hosts: What we need in Zabbix to monitor A  host, application, service ...
- Maintenance: The place to configure our maintenance windows. A maintenance can be planned in this location.
- Event correlation: When we have multiple events that fires triggers related we can configure correlations in this place.
- Discovery: Sometimes we like to use Zabbix to discover devices, services,... on our network. This can be done here.
```




```
- Actions:
- Media types:
- Scripts:
```




```
- User groups:
- User roles:
- Users:
- API tokens:
- Authentication:
```




```
- General:
- Audit log:
- Housekeeping:
- Proxies:
- Macros:
- Queue:
```

Info

More information can be found in the online Zabbix documentation here

Info

You will see that Zabbix is using the modal forms in the frontend on many places. The problem is that they are not movable. This module created by one of the Zabbix devs `UI Twix` will solve this problem for you.

Note

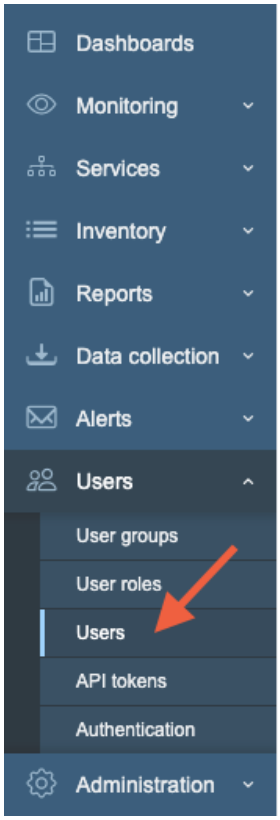
At time of writing there is no Dashboard import/export functionality in zabbix. So when upgrading dashboards need to be created for admin by hand. This should be fixed in 7 onces it comes out. If not feel free to track https://support.zabbix.com/browse/ZBXNEXT-5419

7.2 Zabbix Users & User groups

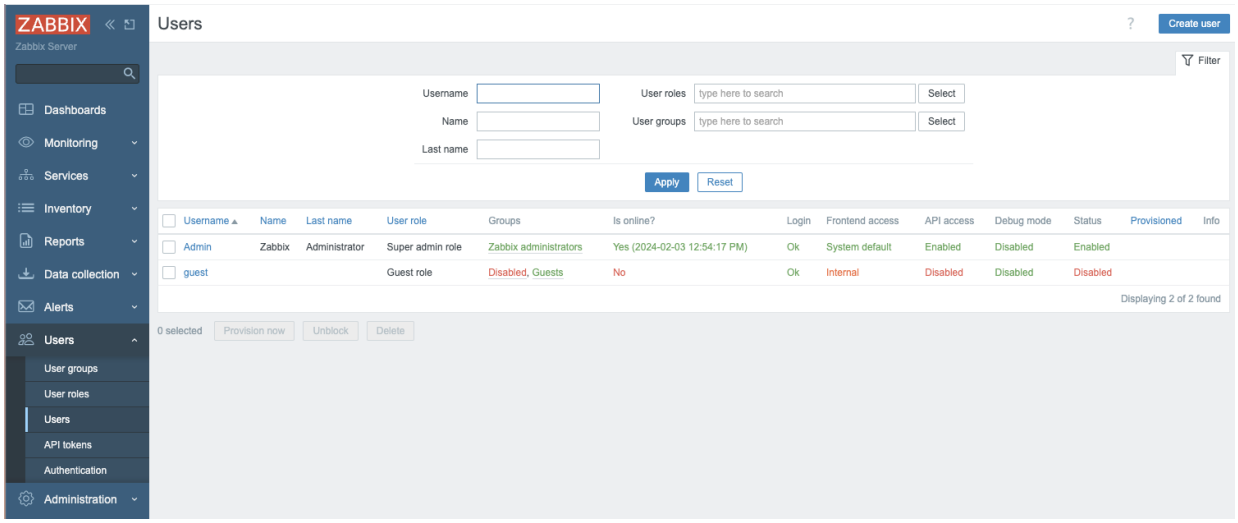
Now that we know how the Zabbix dashboard is build up our first task will be to create a user. In case you missed it the standard Zabbix (yes the capital Z here is eeded to login.) user is Admin and has the password zabbix so we need to change this ASAP. The most confusing part is probably that the user Admin in zabbix is actually a `super admin` but more about that later.

7.2.1 Changing the Zabbix super admin password

In our menu on the right side of the screen, click the `Users` section, and then choose `users`. As you can see here in the screenshot.



You will now see a list of all the users that are created on the system when installing a new Zabbix instance. Here you will always see a list of all users that are configured on the system.



To change the password, do the following steps: - Click user `Admin` - Click on the button `Change password`. - Fill in the current password, `zabbix` - Fill in the new password twice and press `Update` at the bottom of the page.

7.2.2 Zabbix User types

Before we create new users, it's important to know that Zabbix has three user types that are built-in.

User type	Description
Zabbix User	This is a normal user that only has read-only permissions if given. So there are no permissions assigned by default.
Zabbix Admin	A user with read/write permissions. Just like the Zabbix user, there are no permissions by default. However access can be denied to some groups.
Zabbix Super Admin	A user with group read/write permissions. The user will have read/write access to all host and template groups. Access can't be revoked by denying access to groups, like with a normal admin.

Besides these differences, these users also have different access rights to our menu. Let's have a closer look.

- A normal user will only see a part of our menu on the left. Also, some sub-sections of the menu will not be visible. An `Admin` user will have more rights than a regular user and will be able to make some configuration changes in Zabbix. A `Super Admin` user will have unlimted right and see every part of the menu. The only way to limit a `Super Admin` will be by making use of roles. Something we cover later.
- An `Admin` user will have more rights than a regular user and will be able to make some configuration changes in Zabbix.
- A `Super Admin` can access all parts of the menu.

This table gives an overview of all the permissions a Zabbix user, admin, and super admin have in the Zabbix menu:

	Zabbix User	Zabbix Admin	Zabbix Super Admin
Dashboards	✅	✅	✅
Monitoring	✅	✅	✅
- Problems	✅	✅	✅
- Hosts	✅	✅	✅
- Latest data	✅	✅	✅
- Maps	✅	✅	✅
- Discovery	❌	✅	✅
Services	✅	✅	✅
- Services	✅	✅	✅
- SLA	❌	✅	✅
- SLA Report	✅;	✅	✅
Inventory	✅	✅	✅
- Overview	✅	✅	✅
- Hosts	✅	✅	✅
Reports	✅	✅	✅
- System information	❌	❌	✅
- Scheduled reports	❌	✅	✅
- Availability report	✅	✅	✅
- Triggers top 100	✅	✅	✅
- Audit log	❌	❌	✅
- Action log	❌	❌	✅
- Notifications	❌	✅	✅
Data Collection	❌	✅	✅
- Template groups	❌	✅	✅
- Host groups	❌	✅	✅
- Templates	❌	✅	✅
- Hosts	❌	✅	✅
- Maintenance	❌	✅	✅
- Event correlation	❌	❌	✅
- Discovery	❌	✅	✅
Alerts	❌	✅	✅
- Trigger actions	❌	✅	✅
- Service actions	❌	✅	✅
- Autoregistration actions	❌	✅	✅
- Internal actions	❌	✅	✅
- Media types	❌	❌	✅

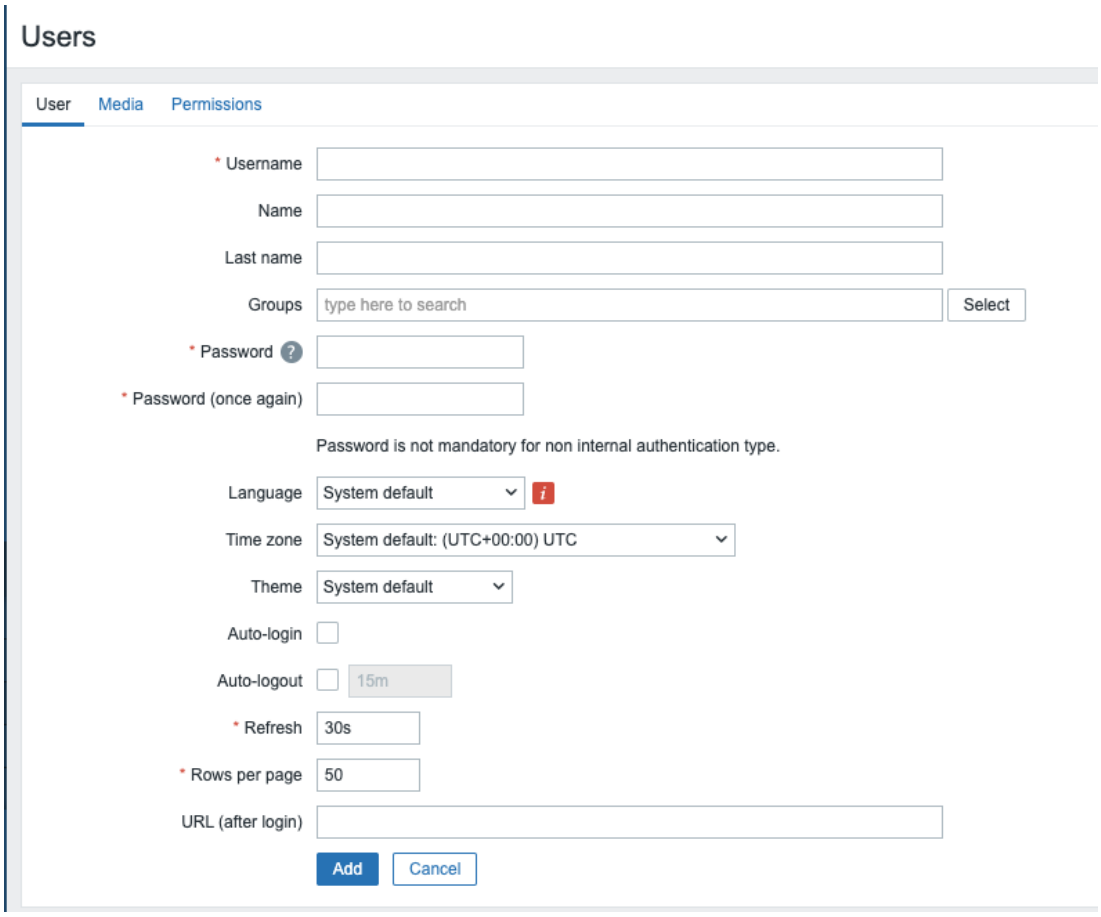
	Zabbix User	Zabbix Admin	Zabbix Super Admin
- Scripts	❌	❌	✅
Users	❌	❌	✅
- User groups	❌	❌	✅
- User roles	❌	❌	✅
- Users	❌	❌	✅
- Api tokens	❌	❌	✅
- Authentication	❌	❌	✅
Administration	❌	❌	✅
- General	❌	❌	✅
- Audit log	❌	❌	✅
- Housekeeping	❌	❌	✅
- Proxies	❌	❌	✅
- Macros	❌	❌	✅
- Queue	❌	❌	✅

- An `Admin` user will have more rights than a regular user and will be able to make some configuration changes in Zabbix.
- A `Super Admin` can access all parts of the menu.

7.2.3 Creating a new User in Zabbix

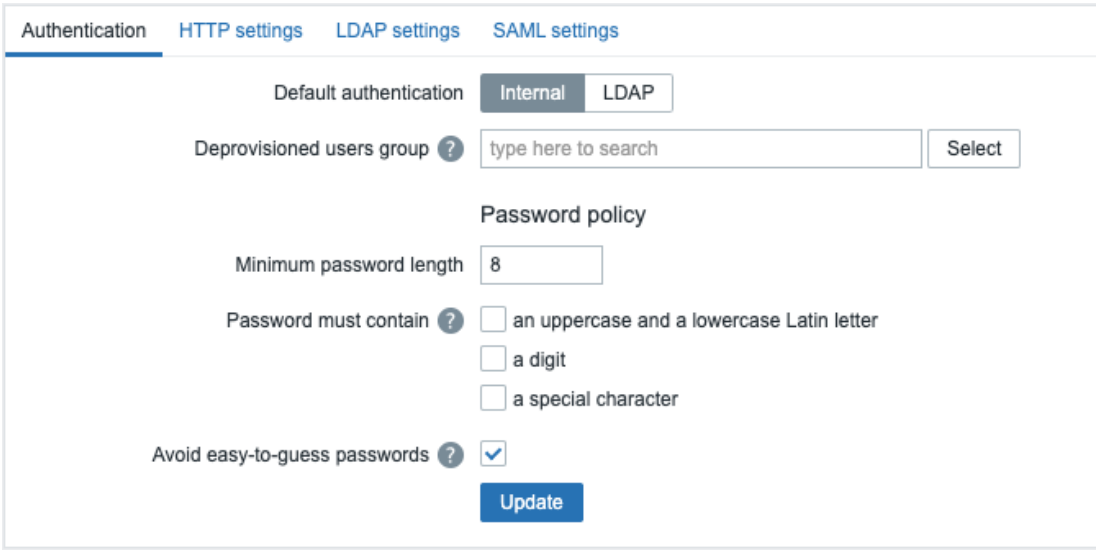
So now that we are in the users section of Zabbix, it's probably a good time to create a new user for our system. If you skipped the previous step, go to the menu `Users` -> `Users`.

Click on the top right on `Create user` and fill in the details of your new users. You will see that some fields have red asterisks in front of them, like Username and Password, ... this means that those fields are mandatory to fill in.



STRENGTHEN THE ZABBIX PASSWORD POLICY.

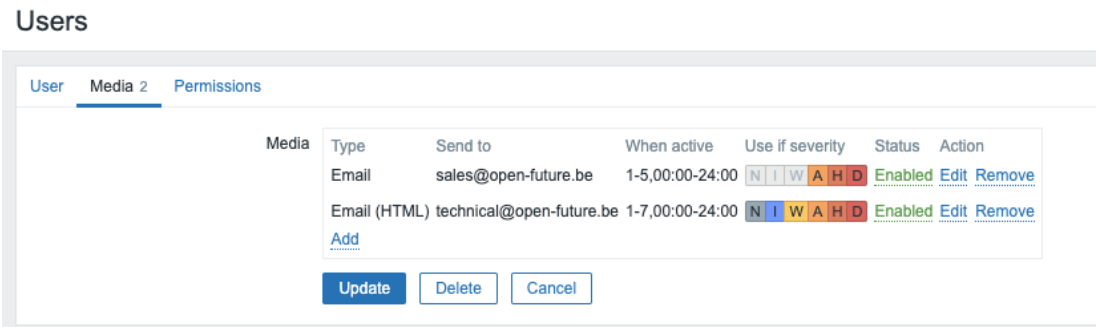
Zabbix passwords rely on a minimum length of 8 characters and also block a list of easy-to-guess passwords. We can make our passwords more secure by telling Zabbix that our passwords must contain uppercase and lowercase characters, a digit, and a special character. This policy is a global policy that will be enforced, and we have to set this policy as Super Admin. Go to the menu Users -> Authentication. In older versions, you can find it under Administration Authentication.



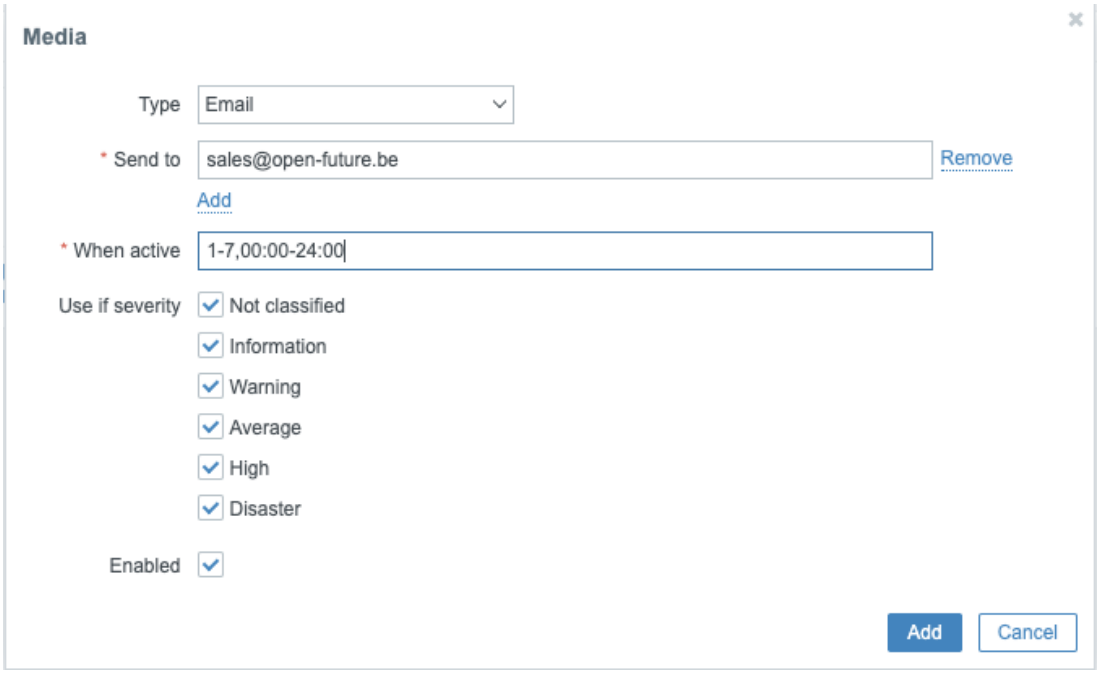
Parameter	Description
Username	A unique name that will be used as username when we login.
Name	The users firstname this field is optional visible in acknowledgment information and notification recipient information if set.
Last Name	Users last name. Optional, this field is optional visible in acknowledgment information and notification recipient information if set.
Groups	Select what group the user will belong to. Atleast 1 group needs to be selected. This feeld will auto complete or you can press the '"Select"' button at the end of the field.
Password	There are 2 password fields they can only be used for internal authentication but more about this later. If the user has the Super admin role then clicking on the Change password button opens an additional field to entering the current (old) password. On a successful password change, the user for which the password was changed will be logged out of all active sessions.
Language	Language of the frontend. The php gettext extension is required for the translations to work. And the language needs to be configured on the system. See the chapter "Installing Zabbix" in case you forgot.
Timezone	Select the time zone per user or use the default timezone that is configured on the Zabbix server.
Theme	Here users can select their own look and feel by choosing one of the 4 themes provided by Zabbix or another custom made theme. Default will switch to the default theme chosen by the admin.
Auto-Login	Check this box so that the user will be remembered for 30 days. The browser must accept cookies for this to work.
Auto-Logout	Checking this box makes sure the user gets logged out automatically, after the set amount of seconds (minimum 90 seconds, maximum 1 day). Time suffixes are supported, e.g. 90s, 5m, 2h, 1d. Note that this option will not work if : • If the "Show warning if Zabbix server is down" global configuration option is enabled and Zabbix frontend is kept open. • When Monitoring menu pages perform background information refreshes. • If logging in with the Remember me for 30 days option checked.
Refresh	Set the refresh rate used for graphs, plain text data, etc. Can be set to 0 to disable. Time suffixes are supported. Ex: 90s, 5m, 1h.
Rows per page	Define how many rows per page will be displayed in lists.
URL(after login)	You can make Zabbix transfer the user to a specific URL after successful login. This can be useful for monitors in NOC team for example so you arrive on a specific dashboard that is maximised. You can make Zabbix transfer the user to a specific URL after successful login. This can be useful for monitors in NOC team for example so you arrive on a specific dashboard that is maximised.

User Media

The tab '" Media "' contains a list of all media that are defined for our user. Media is used for sending notifications to the user. We can click the `Add` button.



Adding the media here is not enough to receive notification; we also need to configure our media properly, and we still need to configure actions as well. When pressing the '" Add "' button, we get a popup where we can select some information.



Parameter	Description
Type	A drop down list with the names of all media types. When a media type is disabled it will be in red.
Send to	Here we can provide contact information. For an email media type it is possible to add several addresses by clicking on "'Add"' below the address field. In this case, the notification will be sent to all email addresses provided. It's also possible to specify recipient name in the Send to field of the email recipient in a format 'Recipient name <address1@company.com>'. Note that if a recipient name is provided, an email address should be wrapped in angle brackets (<>). UTF-8 characters in the name are supported, quoted pairs and comments are not. For example: John Doe <manager@open-future.com> and manager@nycdatacenter.com are both valid formats. Incorrect examples: John Doe manager@open-future.com, %%"Zabbix\@\<H(comment)Q\>" zabbix@company.com %%.
when active	The time when media will be active from monday till sundat, 1-7 and the time from 00:00 till 24:00 for example only in weekends from 6 in the morning till 5 in the evening: 6-7,06-17:00i. This is based on the user his timezone
Use if severity	A list of checkboxes from the severities you would like to recieve notifications from. Selected severities will be displayed in color. !! Read the warning below!!
Status	Status of the media we have selected either enabled or disabled (in use or not)

Warning

When selecting the different severity levels, be aware that you have to select `Not classified` if you want to receive notifications about non-trigger events, like internal events. For more information, check out Event Sources. This is something that is not obvious, and Zabbix documentation could be better at explaining this.

User permissions

When we go to the `Permissions` tab in our `Users`, we will get an overview of all permissions our users had in the menu structure. Or when creating a new user, we have the option to select a `User Role`. Zabbix has four different `User Roles` built-in. There is a User role, Admin role, Super admin role, and a Guest role.

The `Guest role` is a role with very strict access limitations. Its role is intended for users to access Zabbix without any user account. I never advise using this role unless you know what you are doing. When you open your GUI to users without any authorization, this could leak potential sensitive data like hostnames, IPs, etc.

Choosing a `User type` is one thing; based on the `User type` we choose, our users will have more or less rights in our main menu. But there is another important part when choosing the `User Type`. This also has an impact on the rights each user has over host groups. For example, a regular user can only have read rights or no rights. A Zabbix admin user can have full, read-only, or no rights, and a Zabbix Super Admin always has full rights on host groups, and his rights on the host groups cannot be revoked.

Here is an overview of every user and his rights:

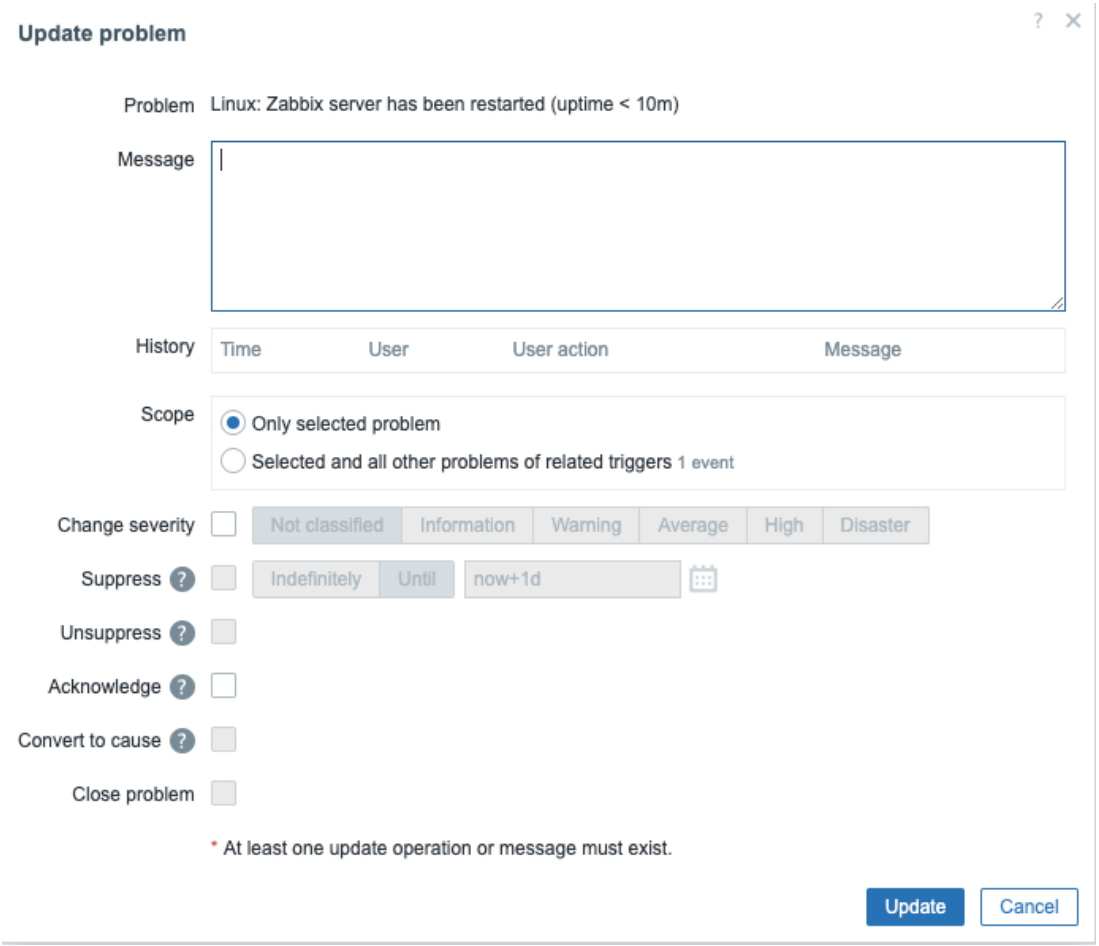
Group rights	Zabbix User	Zabbix Admin	Zabbix Super Admin
Read/Write	Read Only	Full	Full
Read only	Read Only	Read Only	Full
Deny	None	None	Full

Note

With all this knowledge, we now know that if we want to create a regular user who also has access to certain parts of the Administration menu, that it's not possible. We can never create a user that has only RO access to certain host groups and RW access to the `Administration` part. What we could do, however, is create a `Super Administrator` account and remove access from the menu for certain parts of the `Administration` menu by creating a special role. There is no limit on the number of roles you can create.

Note

Also, be aware that when you click on an item on the dashboard on `Update`, you will see a modal window popup with some options to change the severity, close a problem, etc., so some will be greyed out. This is because the user needs write permissions. For example, a user needs write permissions to close a problem and change the severity level.



Note

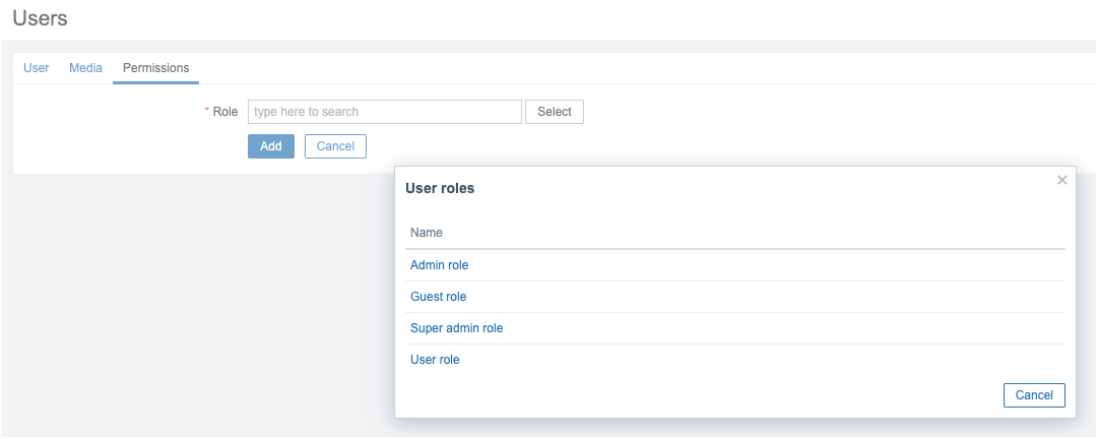
With Zabbix 7 Permission checks have been made much faster. This was made possible by making some improvements on how permissions are stored. This should make the frontend faster when when we have permission havy pages to load like the ones with hosts or problems widgets.
- New tables have been introduced for the check of non-privileged users.
- The new tables will keep hashes (SHA-256) of user group sets and host group sets for each user/host.
- Also a new permission table was introduced for storing only the accessible combinations of users and hosts, specified by the hash IDs.
- Hashes and permissons are not calculated for Super Admin users.

7.2.4 User Roles

User roles have been in Zabbix since version 5.2 and make our lives easier by allowing us to make some custom adjustments to the standard defind user types in Zabbix.

When we go to our `Permissions` tab, we can see a box `Role`. Press the `Select` box to see a popup with a list of roles to choose from. There are four standard roles to choose from. You can create your own list of rules by going to the menu `Users -> User Roles` and create your own limited user.

The box is marked with an asterisk in front, so you need to select a user role for every user you create.



Warning

Be aware that no permissions can be added to user roles only permissions can be revoked.

7.2.5 User Groups

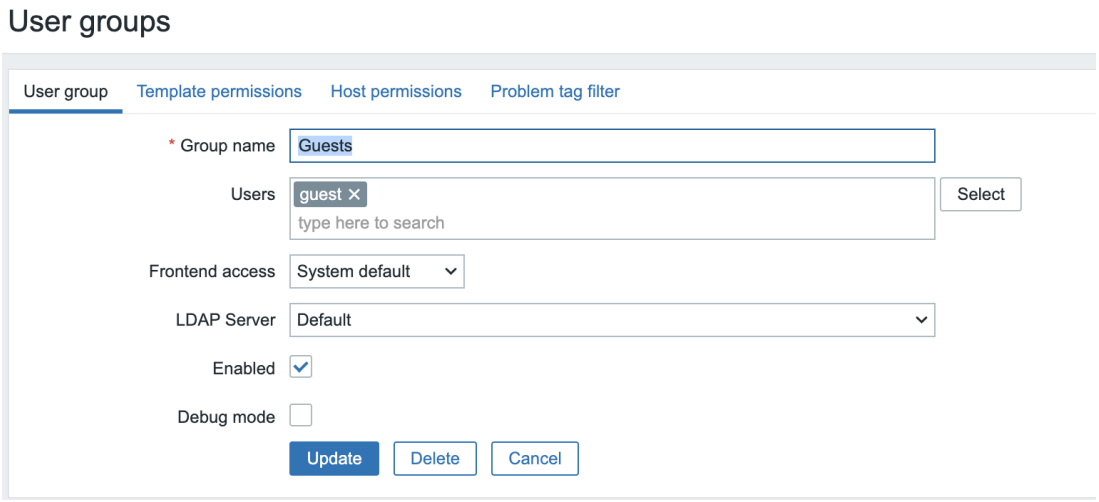
A user always needs to be member of one or more `User groups`. We will not set any user rights directly on Users in Zabbix but we do this on User groups. So if a User needs the permission to view or edit a host or a template then this is set on the `User group` wich has the permisson to view or edit a host or template group and never on a host or a template directly.

Zabbix has a few different rights we can use on group level, as we have seen above. To make it easier for you I add them again:

Group rights	Zabbix User	Zabbix Admin	Zabbix Super Admin
Read/Write	Read Only	Full	Full
Read only	Read Only	Read Only	Full
Deny	None	None	Full

When it comes to permissions in Zabbix groups, the highest level will win. A user that has read and read-write rights on the same host will get read-write permissions. Except for Deny, Deny will always overrule. So if we have a Zabbix `Admin` user then this user can have Read/Write rights, if we add a host in a hostgroup where our usergroup has read rigths, and the same server is in another hostgroup with Read/Write rights, then our user will have Read/Write permissions on the hosts. However if the same host is only in the `Read` hostgroup then our user will only have read rights. If we also add host in a Hostgroup where our usergroup has `Deny` rights then the server will not be visible.

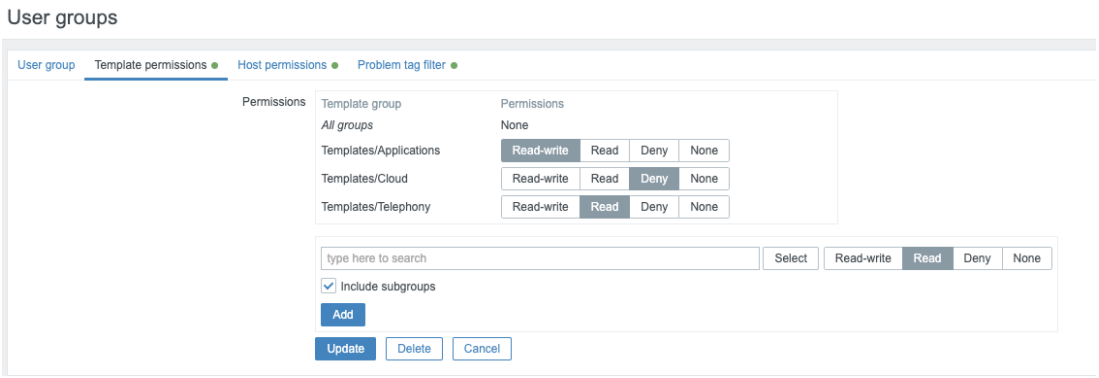
Let's have a look at our `User groups`, for this go to the menu `Users -> User groups` and click on one of the existing users. I used `Guest` in this case.



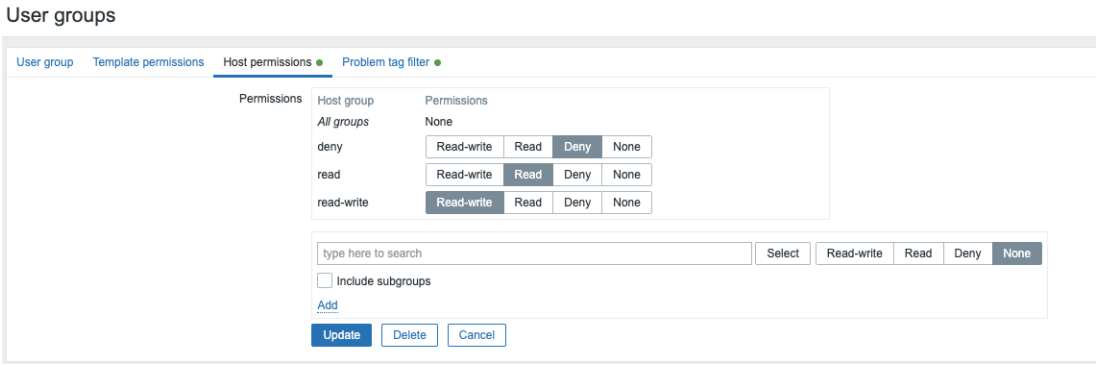
User Groups Overview

Under the tab `User group` we see the following options:

- User group : A field where we have to specify a unique name. This field in mandatory
- Users : Here we add users to our group. Users need to exist before we can add them. Just press select or type the name.
- Frontend Access : How users of the group will authenticate with Zabbix.
- System default : The global configure access method
- Internal : The most easiest way User and Password are configured in Zabbix (Ignored if HTTP authentication is the global default.)
- LDAP: LDAP/AD authentication (Ignored if HTTP authentication is the global default)
- Enabled : If checked the group is Enabled else it will be Disabled
- Debug : Activate debug mode for the users in this group More info about debug
- The next tab next to `User group` is the tab `Template permissions`. Here we can define what `User group` will have access to what `template group`. We can define if a `User group` has read, read-write permissions or if all access must be denied. When selecting a template group don't forget to press the `Add` button first so that you see the `Template group` appear in the Permissions box. Then when you are ready confirm again at the bottom of the page with `Updqte`.



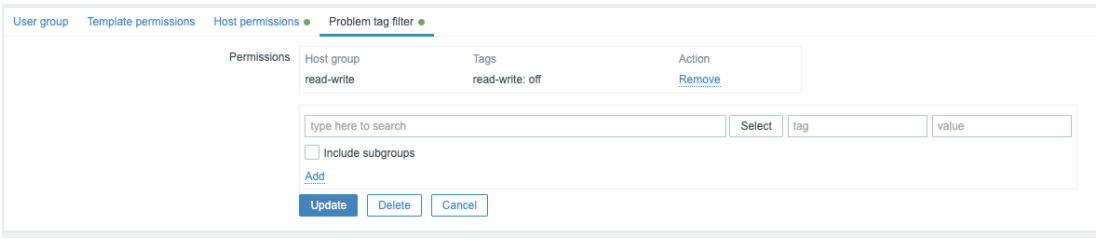
- The `Hosts permissions` tab allows us to specify what `User group```` will have what kind of access on the selected` Host groups `this can again be read, read-write or explicit deny. Just as with the` Templates permissions `tab don't forget to click` Add `first and when you are ready defining all the permissions click` Update``` at the bottom. The name is a bit confusing as we don't select permissions for a host but a host group.



Note

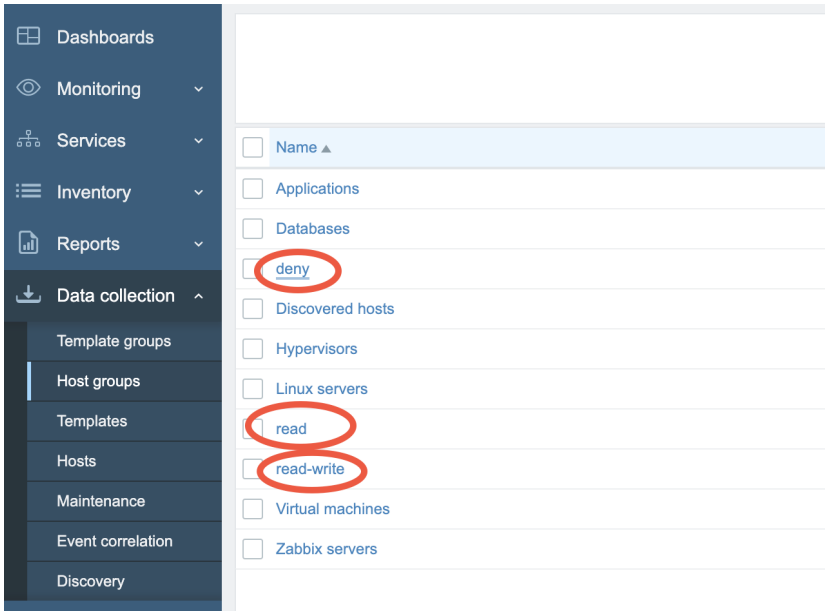
If we add multiple lines with the same host group or template group with different permissions Zabbix will apply the strongest permission.
Alow be aware that a Super admin user can enforce nested groups to have the same level of permissions as the parent group. It can be done in the host group or template group configuration.

- The `Problem tag filter` allows us to filter problems based on tags and their value. It also allows us to separate the access to host groups from our possibility to see only the problems we want.

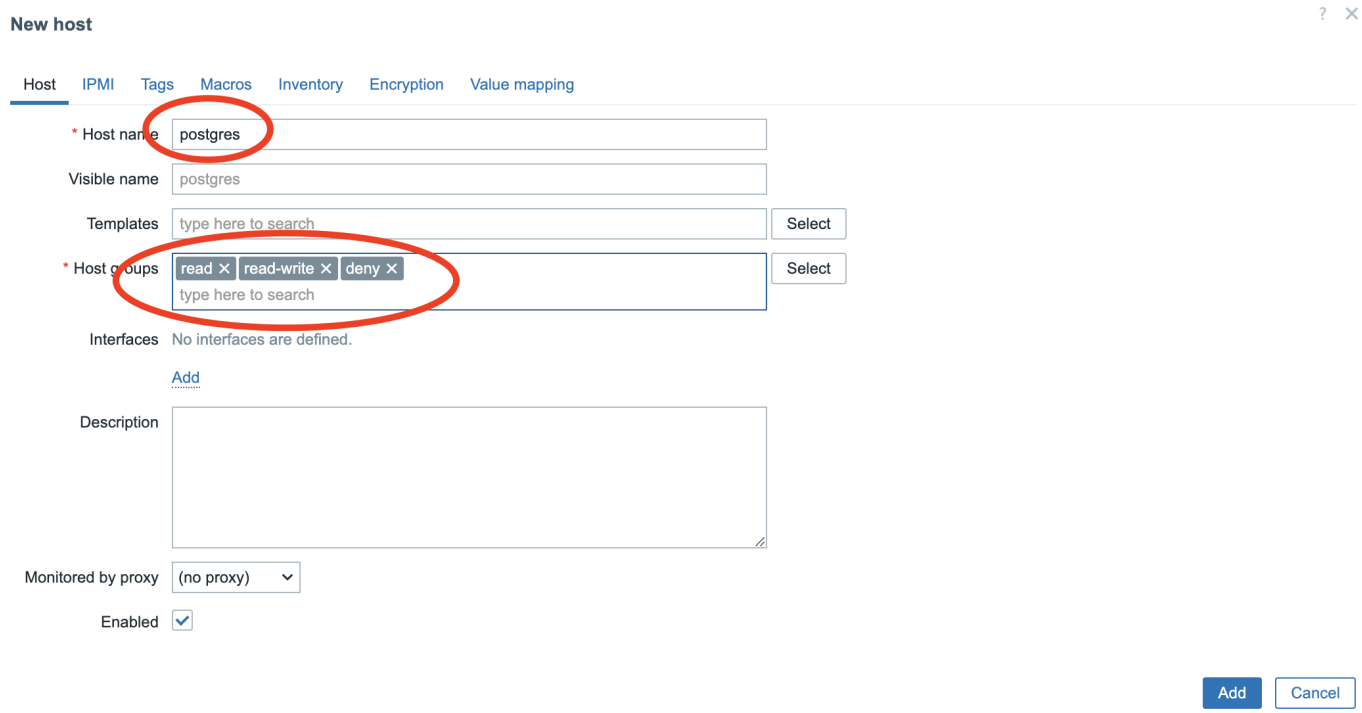


7.2.6 Let's do this together:

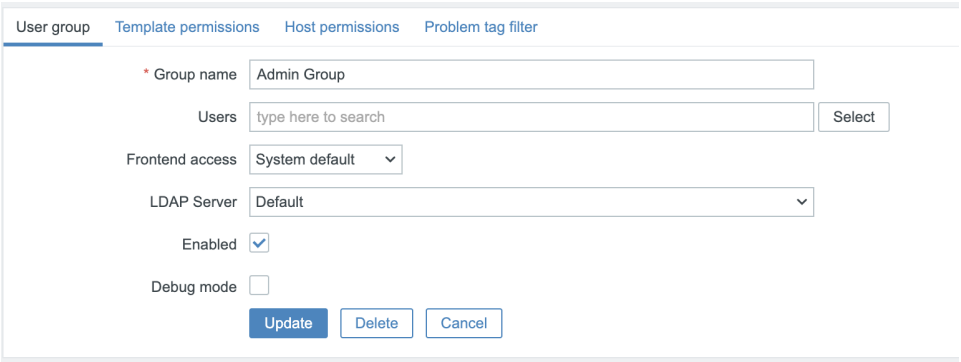
Let us make three `Host groups`, go to the `Data collection` menu -> `Host groups` and create a Host group for `read`, `read-write`, and `deny`.



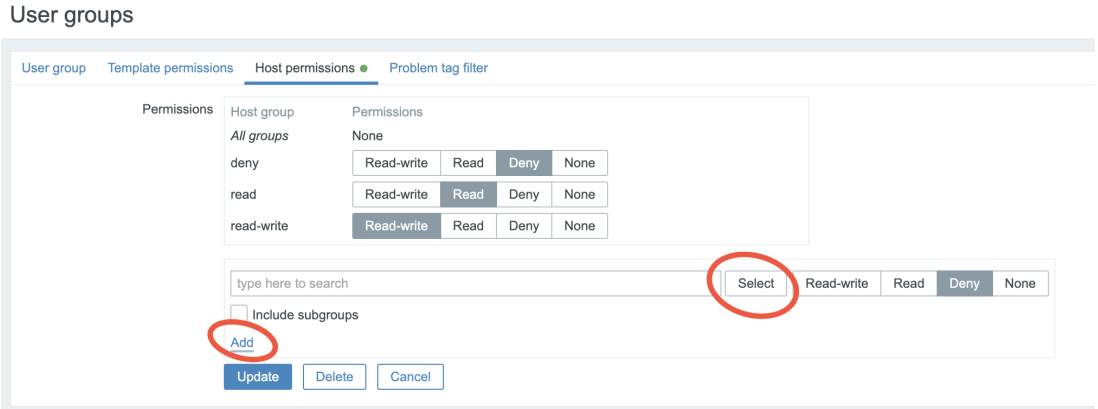
Next step is to create a host and add the host in our three groups. Go to the `Data collection` menu -> `Hosts` and press `Create host` on the right. Add a `Host name`, the name is not that important and add the three `Host groups` we just made.



The only thing we need to do now is create our `User` and `User group` and give the correct rights. Go to our menu `Users` -> `Users group` and click on the top right to `Create user group`. Let's call this group our `Admin Group` as we need a Zabbix `Admin` that we can give read, read-write and later deny to show this.

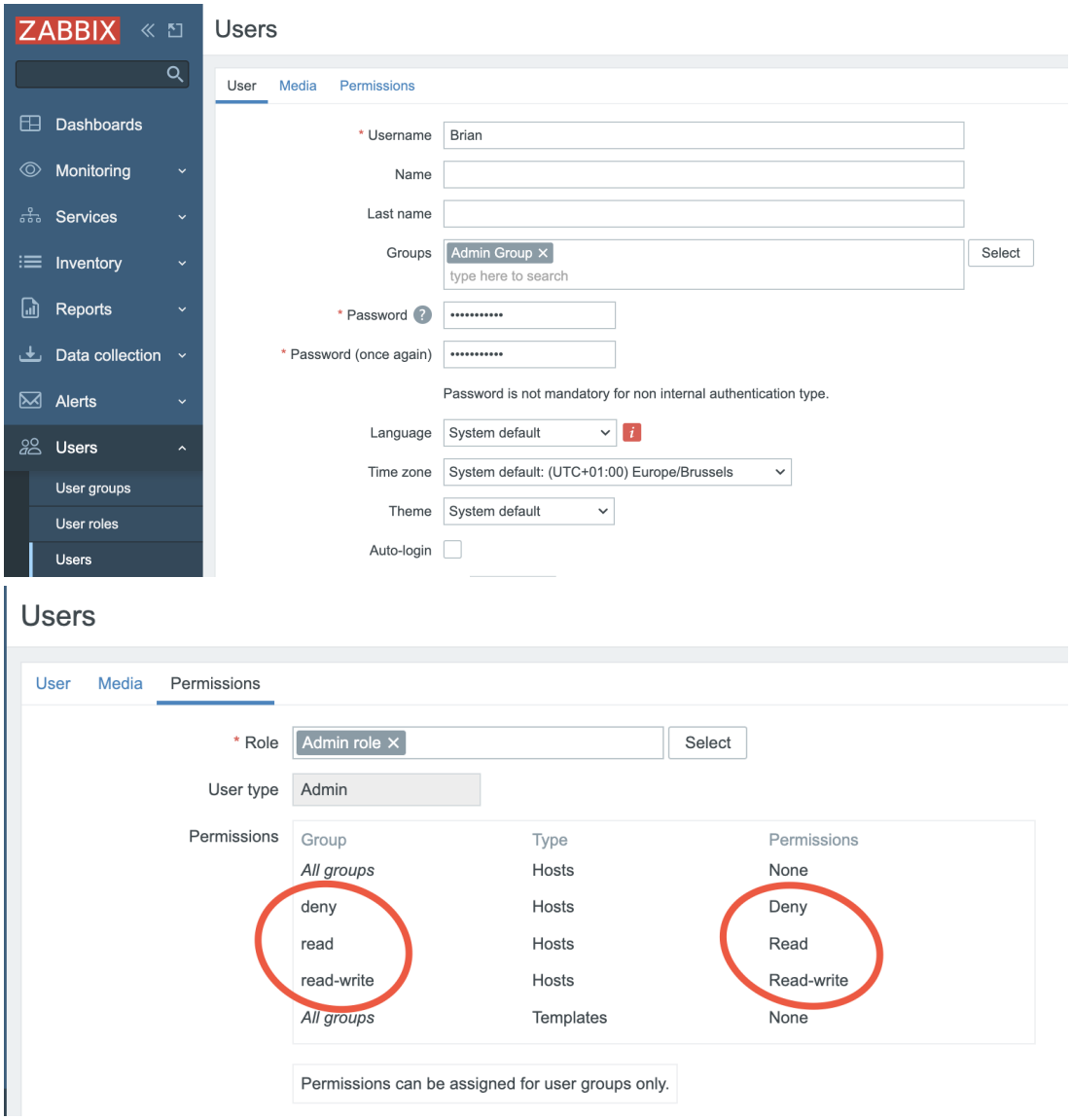


Next go to the tab `Host permissions` and start typing the name of our group `read` in the search box or press the `Select` button and select the correct group. Next before we do anything select also the correct permissions `Deny` and press the add just below NOT the button. Do this also for the group `read-write` and `deny`. If everything looks like in our screenshot then press the `Add` button

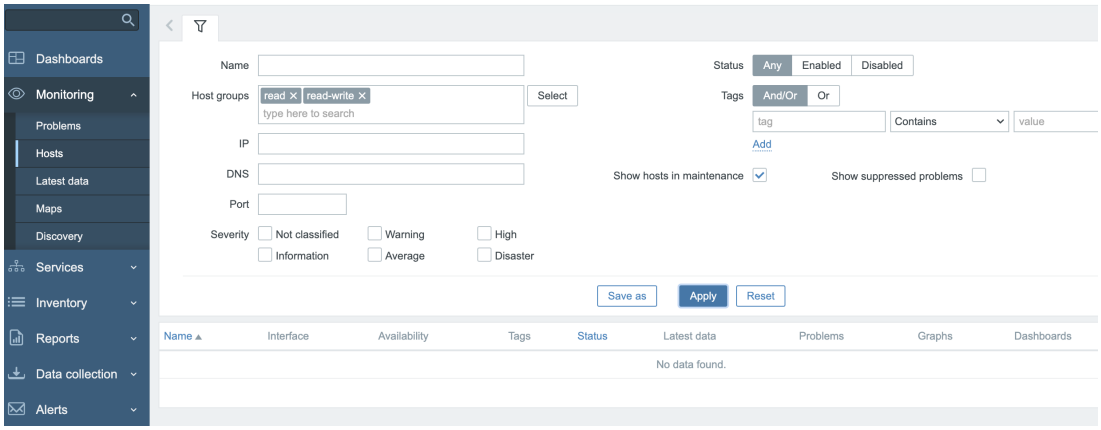


Now for the final step let's create a user. Go to the menu `Users` -> `Users` and create a new user, in the field `Username` we can add our fictive user with the name Brian. In the `Groups` box we select our `Users group` this was `Admin Group`. Don't forget also to add a Password we need to do this twice. Next go to the tab

`Permissions` and select the role `Admin role`. You will see directly once selected that our users bridan has read, write and deny on the correct groups. Press `Add` at the bottom.

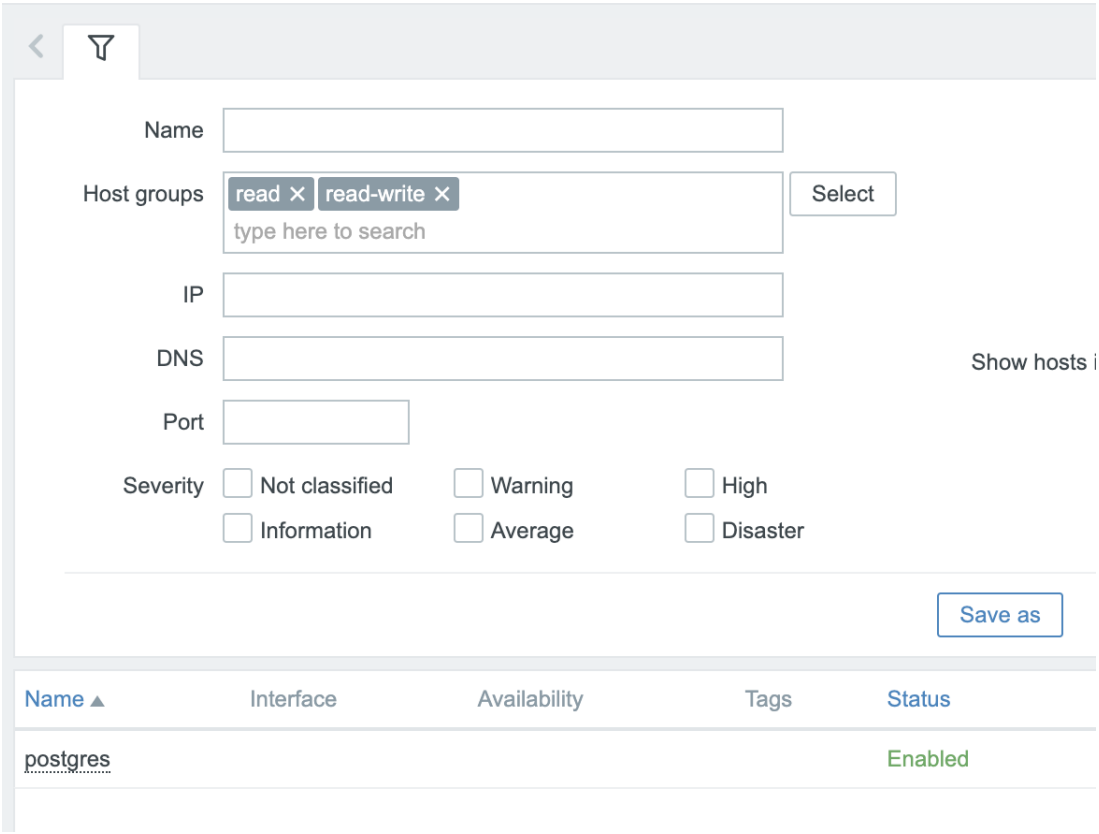


Now it's time to check if everything is as expected. Our user `Brian` if all goes well shouldn't have any rights as we explicitly denied accesss. Press `Sign out` at the bottom left and then login as user `Brian`. Go to the menu `Monitoring` -> `Hosts`. Select all the hosts groups, you should normally only see read, and read-write. Our host group `Deny` is not visible and our host `postgres` is not visible either.



Now log back in as user `Admin`, our Zabbix Super Admin and remove the deny group from our `Admin group`. This can be done by selecting the `None` permissions for the group `Deny` in the `Host permissions` tab from our `User group`.

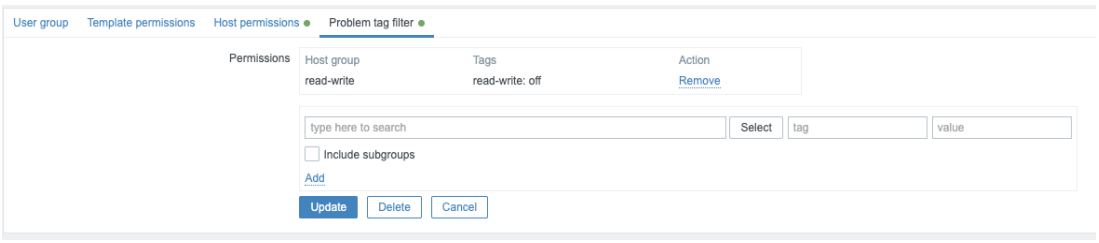
Log back in as our user Brian go back to the `Monitoring` menu to `Hosts`. If all goes well our groups `read` and `read-write` are still selected if nog you just select them again. You will see that our host `postgres` is visisble and that you can click on it to edit the host propreties.



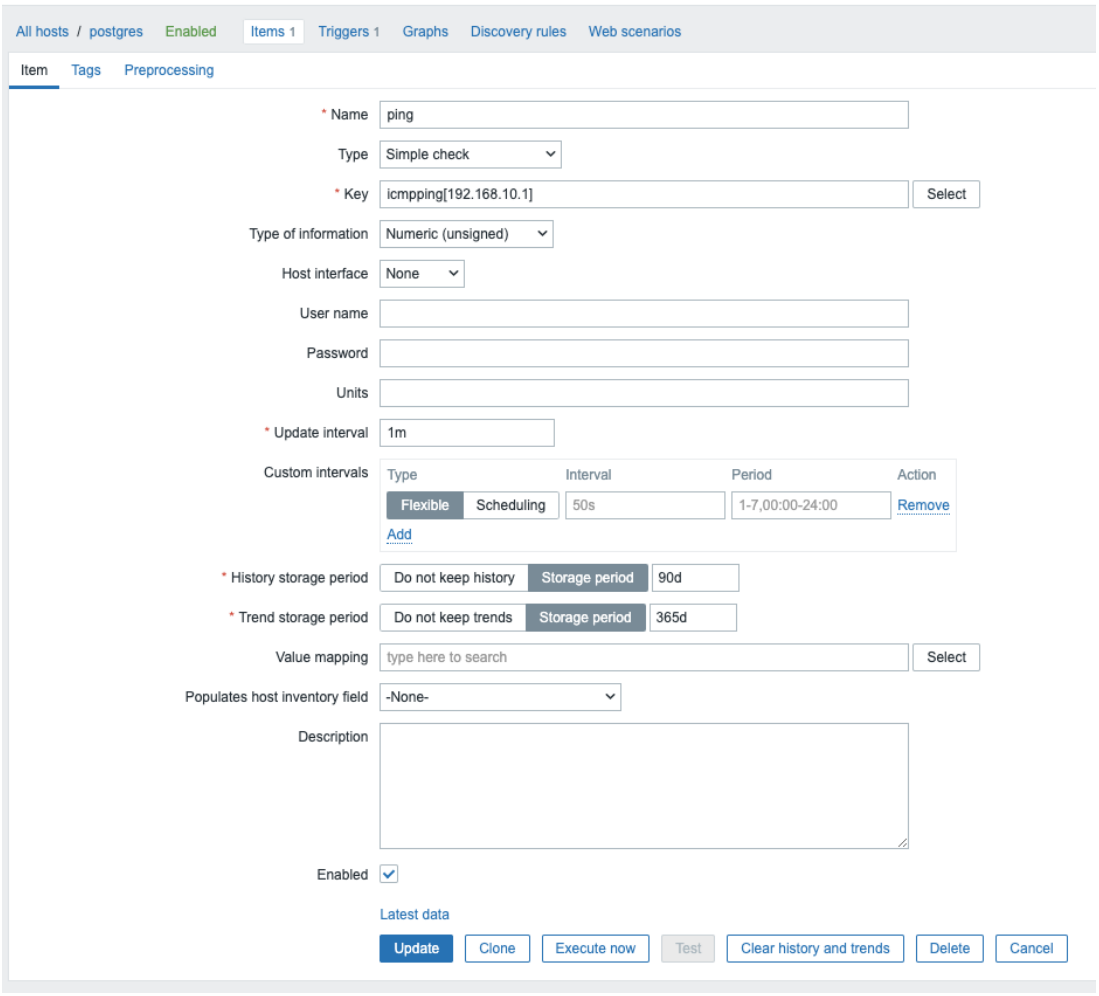
As final test you can try to remove the group `read-write` same as we did before with the `Deny` group. This time only the `read` group will be visible for our user and Brian will not be able to edit our host `postgres` anymore.

Let's try out tags

Now let's add tags into the mix. Imagine that we only like to see problems with a tag `read-write` and value `off`. Go to `User groups` select our `Admin Group` again and go to the tab `Problem tag filter` and fill in the needed tag `read-write` and value `off`.



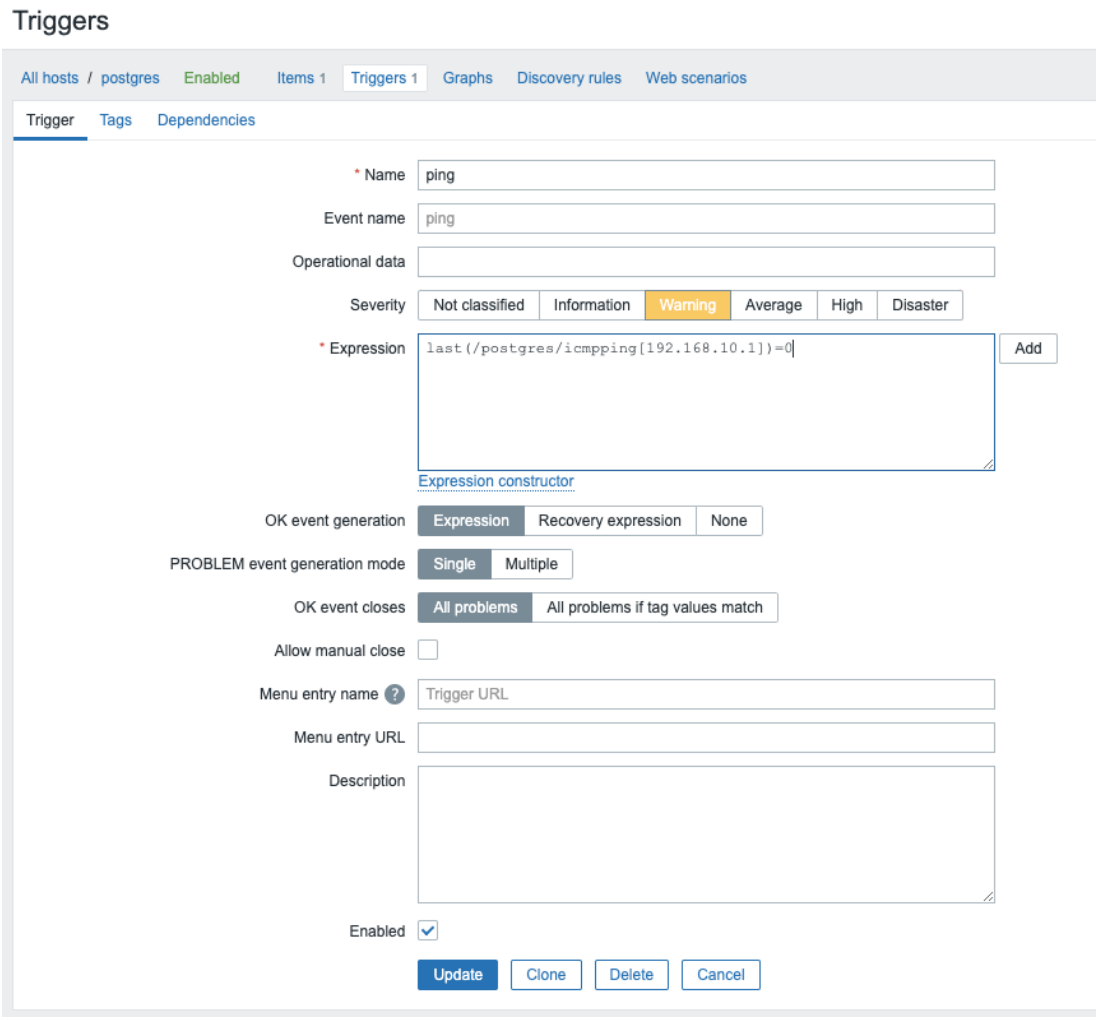
Now we need to create a problem for this we will add an item and a trigger to our host `postgres`. Go to the menu `Data collection` -> `Hosts` and click on items behind our host `postgres`. On the top right you will see a button `Create item` click on it and fill in the same data as in the screenshot below. Don't worry if you don't understand anything we will come to items later.



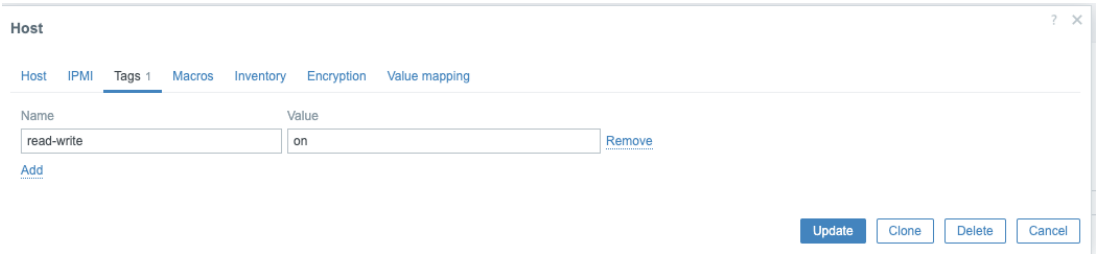
Note

In this item we just tell our Zabbix server to do a ping to IP `192.168.10.1` make sure this IP doesn't exist in your lan so try to ping it first to be sure you don't get a reply back. If you do get a reply back change the IP with some address that is not pingable for you.

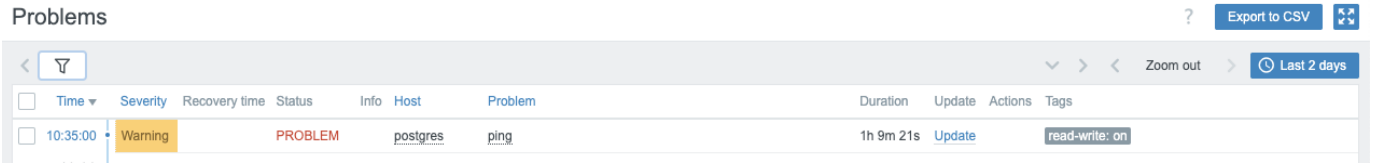
Next step once you have filled in all the data is to save the item and click on top on Triggers. You will also notice now that there is a `1` next to Items. This indicates that we have made 1 item on our host `postgres`. Now that we are in the trigger tab click in the top right corner on the button `Create trigger`. Once again copy over all the data from the screenshot and save the trigger. If you changed the IP in the item make sure you use same IP in the trigger.



Next let's add a tag on our host `postgres` that tells Zabbix to mark everything on the host with a tag `read-write` and value `on`. Remember we added a value `off` in our `User group` problem tag filter tab. So we only want to see everything with a tag `read-write` and value `off`.



When you go now to the `Problem` page in the menu `Monitoring` you should see after some time a warning that there is a problem on our host postgres. You will also see that the problem got a tag read-write with value `on`.

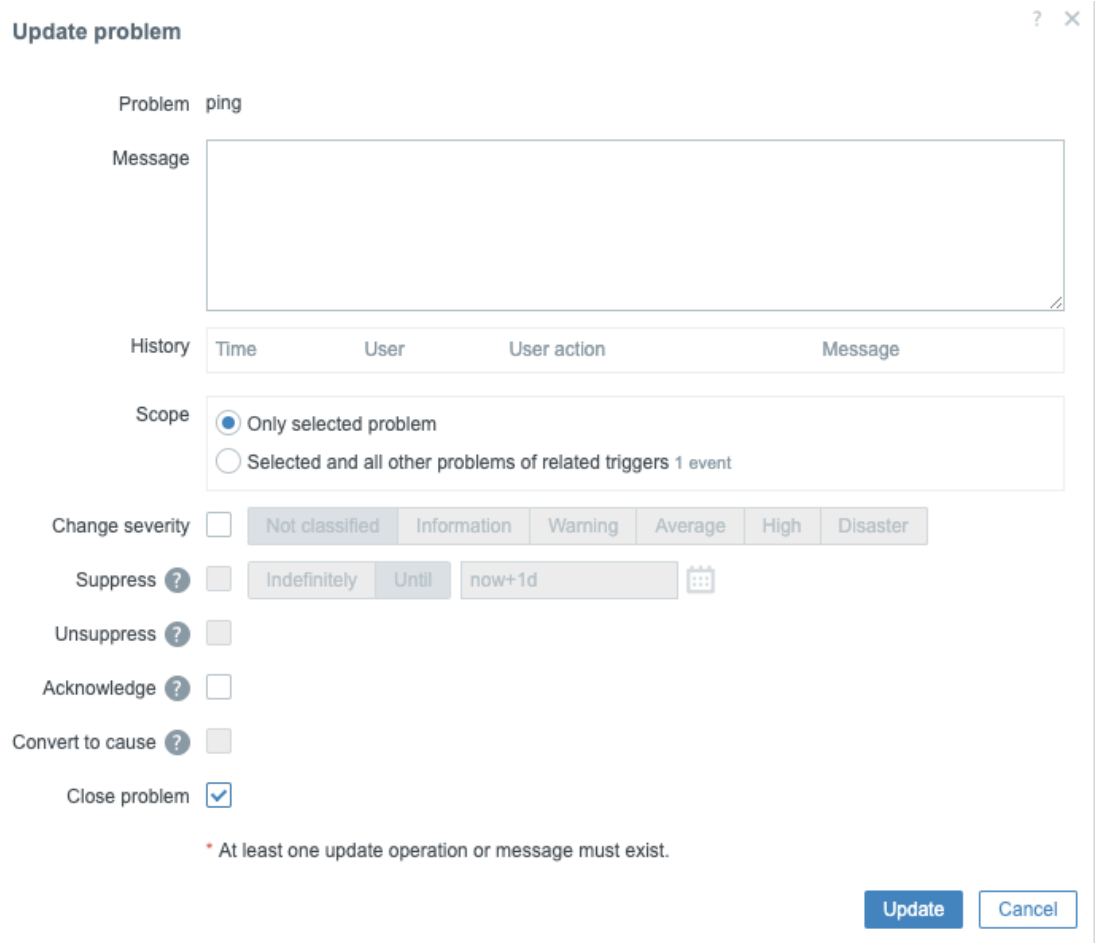


You can clearly see that under our `Zabbix super admin` user the problem is visible. Now do the same but as user `Brian`. You will notice that there is no visible problem for our user even he has `read-write` access to the hostgroup where our server `postgres` belongs to.

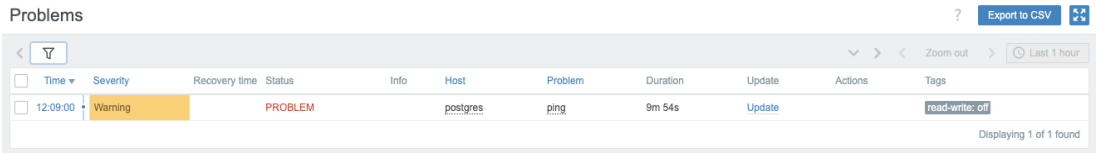
Now as user `Brian` I would like to see the problem so let's go to our menu `Data collection` and click on our host `postgres`. Go to the `Tags` tab and change the value from our tag `read-write` from on to `off`. So now everything on our host should get the tags `read-write` with value `off`. So now Brian should be

able to see the problem right ? Sadly `Brian` is still not able to see the problem in our Problem page. This is because the problem was already created in Zabbix and has already received the tag. So the only way to fix this is to close the problem first and let Zabbix create a new problem again.

As `Super Admin` log back in and go to our trigger `Ping` and mark the box `Allow manual close` and press `Update`. Go back to the dashboard and behind the problem ping you will see `Update`. Click on it and selec the option `Close problem` and press `Update`.



Log back in as our user `Brian` and go to the problem dashboard. We will see that the problem is back. Even we closed the problem before Zabbix opened a new problem because the issue was not resolved. This time our issue has the tag with the correct value.



Note

A Zabbix user needs to be created with a user role. You cannot create one without.

Warning

Be careful if you use the API at the time of writing it's possible to create a Zabbix user with the API without a role. When created by the API the user can even be saved by the frontend afterwards !

Info

More information can be found in the online Zabbix documentation here

7.3 Zabbix hosts

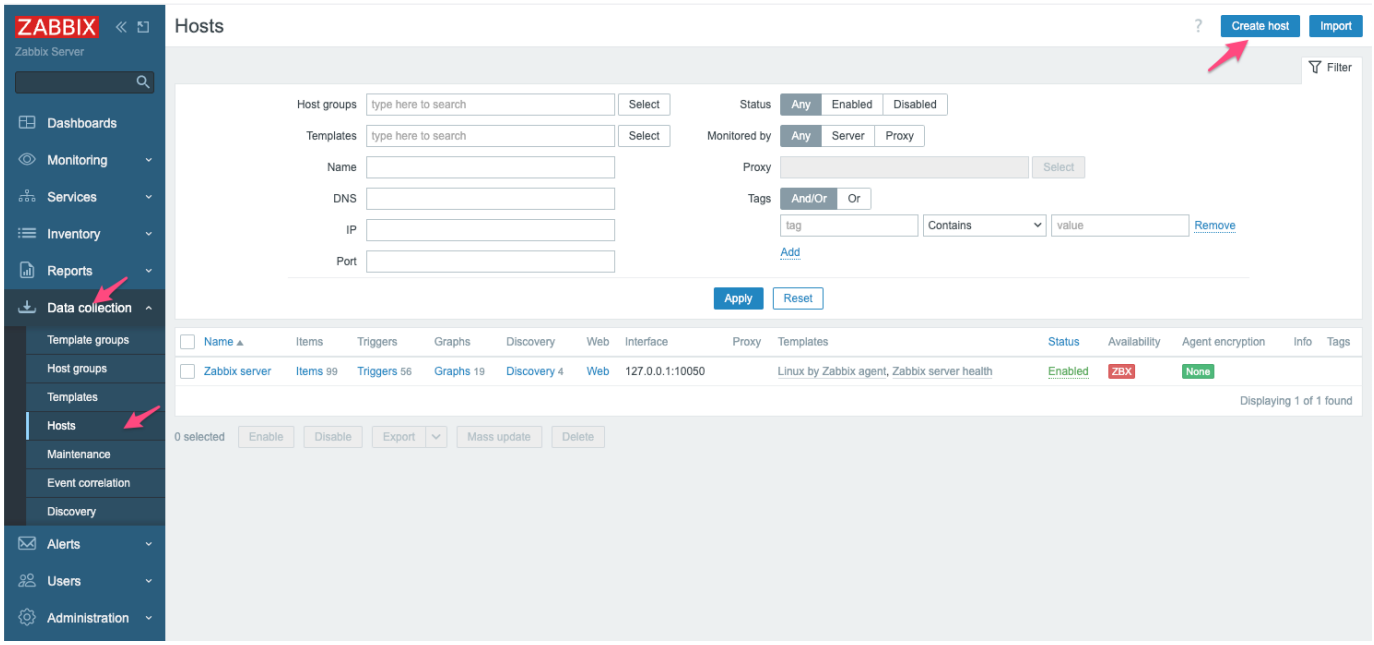
To understand how Zabbix works, it's important to know that `Hosts` in Zabbix are a reference to anything we would like to monitor. It can be a physical host, a virtual machine, an application, a device, or even just a dummy host used to calculate data from existing hosts into something new.

It's probably one of the first tasks that we will do as an Admin when we first login to Zabbix because we need a host if we would like to monitor some metrics. It's however, important to know that hosts cannot be created without being in a hostgroup.

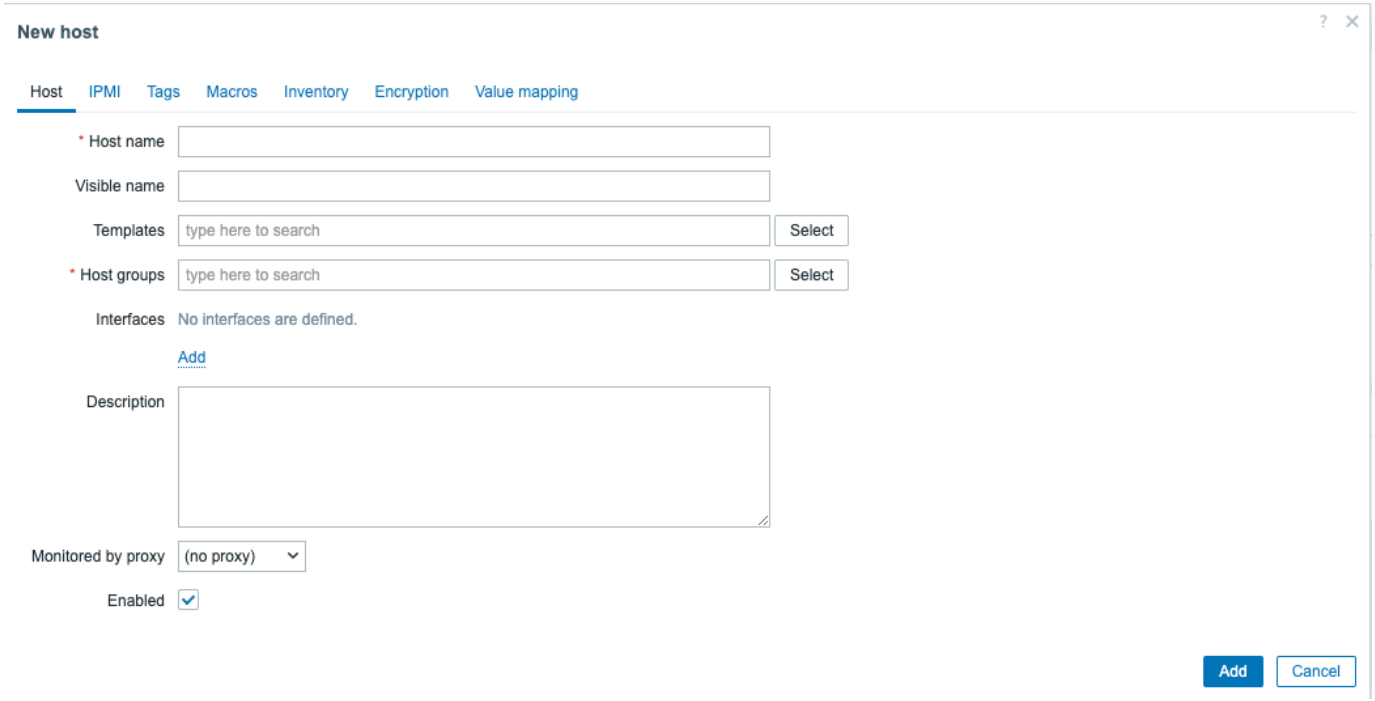
With this said, let's see how to create our first host.

Let's go to the menu on your `left` and select Data Collection -> Hosts. We see that there is already a host configured and that the availability icon is "RED". Don't worry about it, this is normal. We have no Zabbix agent installed or configured.

To add a new host to our system, we have to press `Create host`, this button can be found in the upper right corner of our screen.



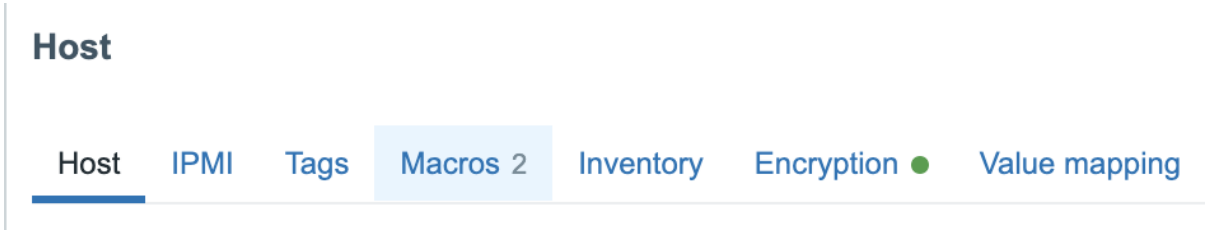
We now get a modal form where we need to fill in some information about our host. The fields marked with a *red* asterisk "*" are the fields that are mandatory.



Parameter	Description
Host name	Here we need to enter the **Host name** of the machine we would like to add. The name can contain alphanumerics, spaces, dots, dashes, and underscores. HOWEVER you are not allowed to use leading and trailing spaces. The **Host name** in the frontend is what we need later for the configuration of our Zabbix agent, so make sure you remember it.
Visible name	The **host name**, as we have seen, is needed to configure our **Zabbix agent**. So in case you like to give it a unique name or one that is randomly generated, ... you can add a visible name here. This name will then be used on the frontend instead of what we call the technical name **host name**. This name has support for UTF-8, so special characters are supported. This name will be used in all the places like maps, the latest data, inventory, ...
Templates	Templates are like blueprints that we can use on our hosts to add items, triggers, etc. We explain more about it in the topic **Zabbix templates**. You can start typing the name of the template, and Zabbix will start to show a list with matches, or you can press the ```Select``` box and choose one from the list.
Host groups	Every host must belong to atleast one ```host group```. This is because permissions are set on host groups. You can type the name of the **host group**, and a list of matching groups will start to appear. Another way is to select a host group from an existing list by pressing the **Select** button. Or you can create a new group by just typing the name and pressing on the box that shows the name of the group you typed with (new) behind it
Interfaces	Zabbix supports several host interfaces, like the Zabbix agent, SNMP, JMX, and IPMI. By default, when we create a host, no interface is added. To add an interface, press **Add** and fill in the needed information, like IP or DNS, depending on the host interface chosen. When an interface is in use (items created that use the interface), then the interface cannot be removed.
Description	A place to enter a short description about our host.
Monitored by proxy	If we have proxies configured, we can select them here if we like to monitor our host through a proxy.
Enabled	Mark the checkbox to enable the host. This will keep it monitored by Zabbix. When unchecked, the host will not be monitored.

7.3.1 Host menu details

Before we add a host ourselves, there are a few things we need to know first. When we click on a host that we have already configured, there are a few things that we will notice. First of all, we see a `blue` line under `Host`. This means that we are on the current `tab` of the host page. As you can see, there are multiple tabs that we can click on, like IPMI, Tags, Macros,...



The next thing we see is that next to the tab `Macros`, there is a number 2. This is because there are two macros configured in the macro tab. So when we add information to tabs like macros or tags ... , Zabbix will show how many items we have added to these tabs by showing next to the tab name the number.

When looking at the `encryption` tab, we notice the green dot. This shows us that an option on the tab has been activated. Now that we know this, let's get a quick overview of every tab and see what it does.

IPMI

So looking at the `IPMI` tab, there are a few things we need to fill in when working with an IPMI interface. IPMI stands for Intelligent Platform Management Interface and is basically a set of standards to manage hardware platforms. In short, it allows us to monitor and manage our servers hardware even if the server is not turned on yet. IPMI is better known as ILO on HP servers and DRAC on Dell servers.

Parameter	Description
Autentication algorithm	Select the authentication algorithm that we have configured on our IPMI server this can be Default, none, MD2, MD5, Straight, OEM, RMCP+
Privelege level	Here we select the privelege level: Callback, User, Operator, Admin or OEM.
Username	The user for authentication that was created on the host. User Macros can be used
Password	The password for our user on the host. User macros can be used.

Note

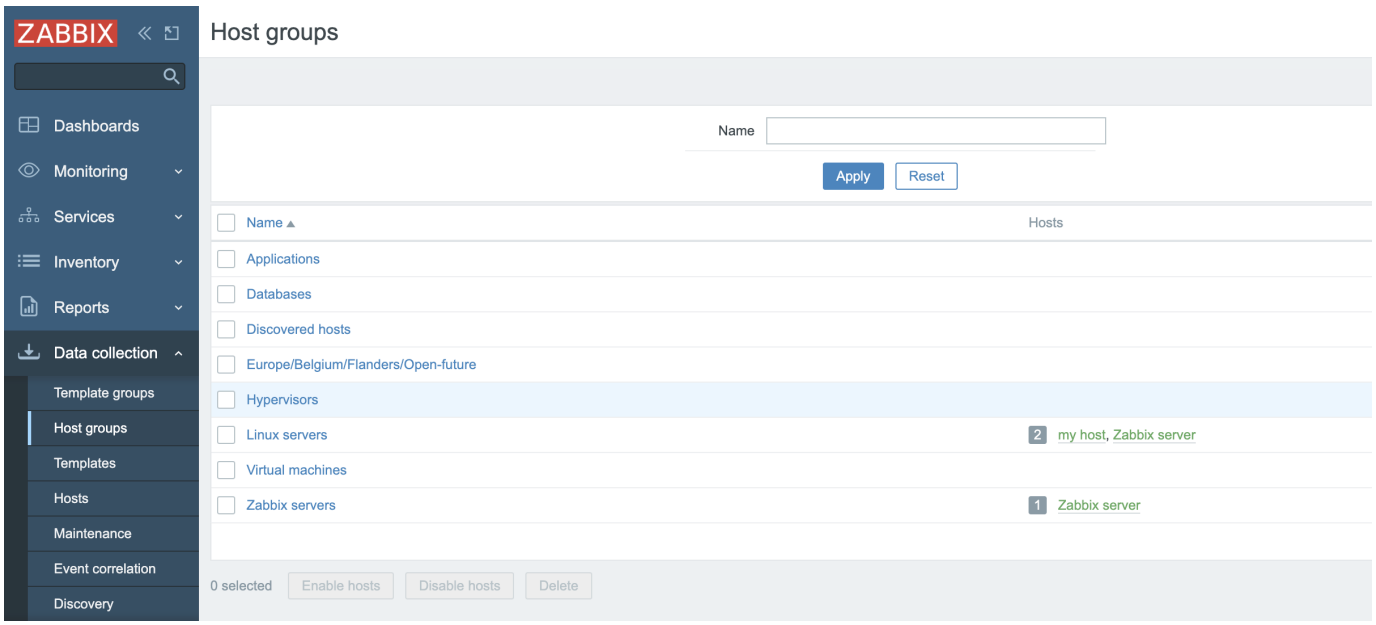
We will cover IPMI in more detail later in the Chapter IPMI Monitoring

Tags

To Do

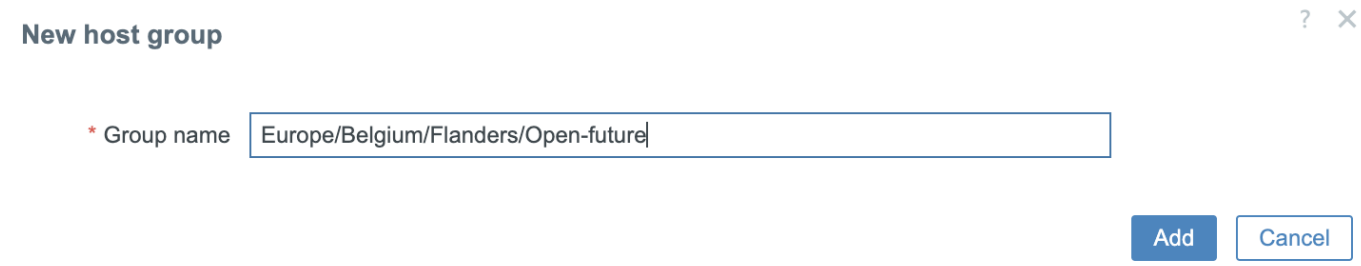
7.4 Host groups

Let's have look at the concepts of host groups and what the benifits are that they provide. We have seen that `Host groups` can be created directly when we create a new `Zabbix host`. Another way to create them is by a `Super Admin` going to Data collection -> Host groups. Next press the button `Create host group` in the upper right corner of the screen. Host groups exists to make a logical group so we can add all hosts that belong together in one group or more. Ex all Linux server, all PostgreSQL server, or all the servers that belong to one team.

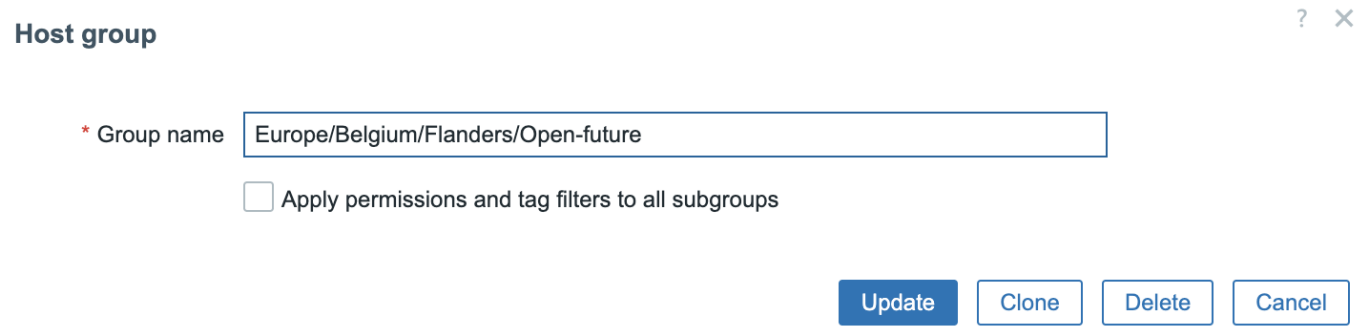


When going to our menu `data collection` you notice that there are `Host groups` and `Template groups`. If you come from an older Zabbix version you will be happy to read that Zabbix made a specific group for Templates. If you are new to Zabbix don't panic :). In older versions Zabbix had mixed Templates and host in one group. This mixing was sometimes confusing especially for new users, as Zabbix doesnt link templates to groups.

When you click on the menu `Data collection` -> Host groups. You will notice that some groups are already made. You will also see that there are some names behind the host groups with numbers in front. These names are the names from the hosts that are in the group. The number in fron is the number of hosts that are in the `host group.` To make life more easy you can click on the names of the hosts and Zabbix will bring you directly to the configuration screen for this host.



Zabbix allows the creation of nested groups. As you can see we are using forward slashes in our group name. When you make use of nested group you can use the '/' to separate groups.



Once our group or set of nested groups is made you can click again from the `host group` overview on the group. You will notice that there is now a box that says `Apply permissions and tag filters to all subgroups`. When pressing this button, all right that are this group will be applied to the sub-groups. So if we have a user `John` for example in a user group that has rights to see everything in the `Host group` with the name `Europe/Belgium` and we apply the option to the subgroups then our user `John` will suddenly see also the hosts in all our nestet groups and the tags on this host.

Note

When creating nested groups, Parent groups don't have to exist. So we can have only the group `open-future` without any of the parent groups. It's up to the user to create them or not. Also group names cannot have `/` in their names. We cannot escape the `/` character. Also leading and trailing slashes and multiple slashes in a row are not allowed.

Tip

Have you tried to put emoticons in fields like host group yet ?

7.5 Interfaces

7.6 templates

7.7 Items

7.8 Zabbix triggers

7.9 Macros

7.10 Data Flow

7.10.1 Data Collection

7.10.2 Simple Checks

7.11 Zabbix Agent

7.11.1 Zabbix agent Linux

7.11.2 Zabbix agent windows

8. Problem detection

8.1 Triggers

9. Taking action when problems come

9.1 Event based Actions

10. Managing permissions

10.1 Managing Permissions

11. Visualising Problems

11.1 Visualising our problems

12. Automating configuration

12.1 Automating configuration

13. VMWare monitoring

13.1 VMWare monitoring

14. Monitoring websites

14.1 Monitoring websites

15. Monitoring SNMP, IPMI and JAVA

15.1 Monitoring SNMP,IPMI and JAVA

15.2 JAVA monitoring

15.3 IPMI monitoring

16. Authentication

16.1 Authentication with HTTP

16.2 Authentication with LDAP

16.3 Authentication with SAML

16.4 Zabbix MFA support

https://support.zabbix.com/browse/ZBXNEXT-6876

17. Monitoring with Proxies

17.1 Monitoring with Proxies

18. Securing Zabbix

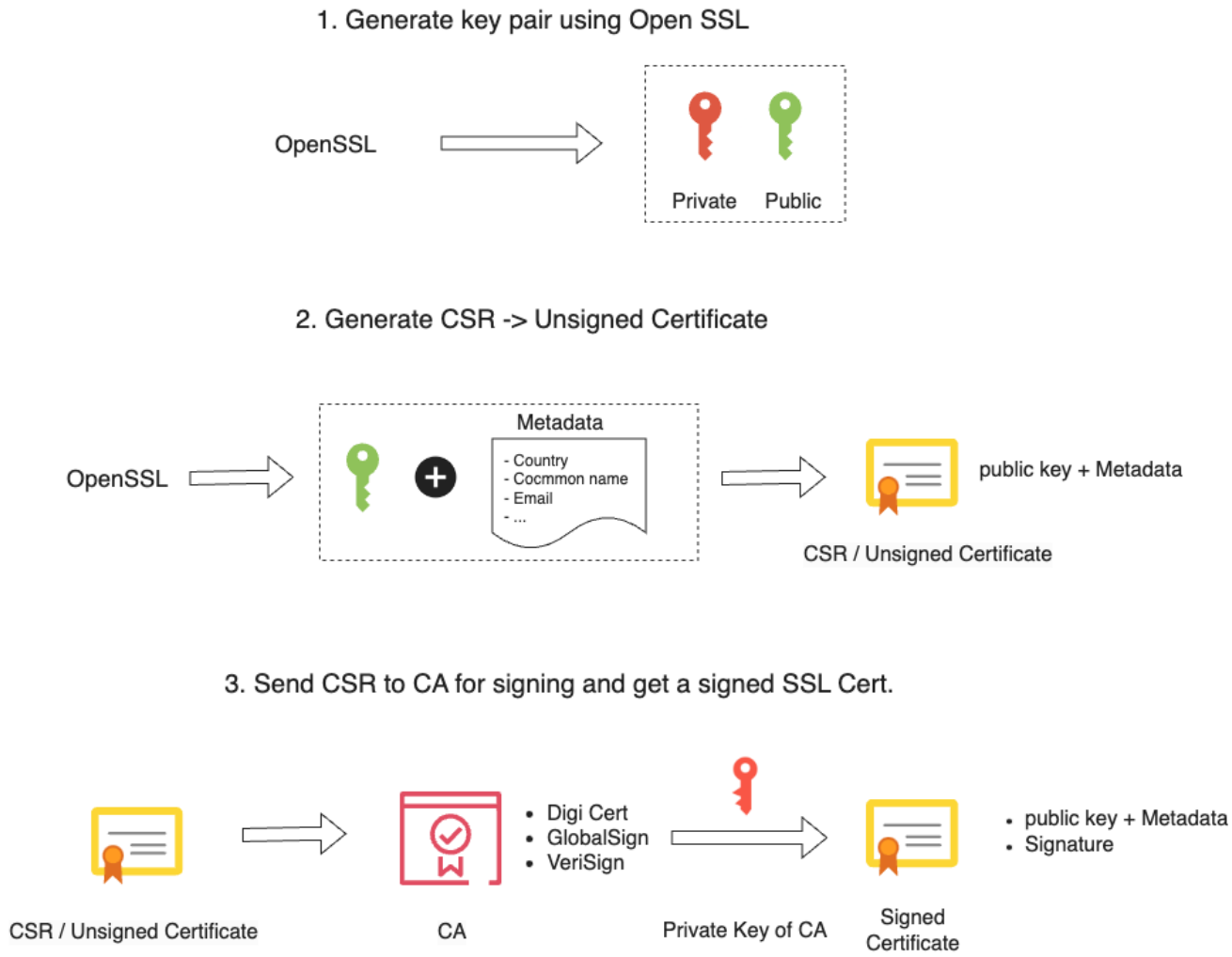
18.1 Securing Zabbix Frontend

The frontend is what we use to login into our system. The Zabbix frontend will connect to our Zabbix server and our database. But we also send information from our laptop to the frontend. It's important that when we enter our credentials that we can do this in a safe way. So it makes sense to make use of certificates and one way to do this is by making use of Self-Signed certificates.

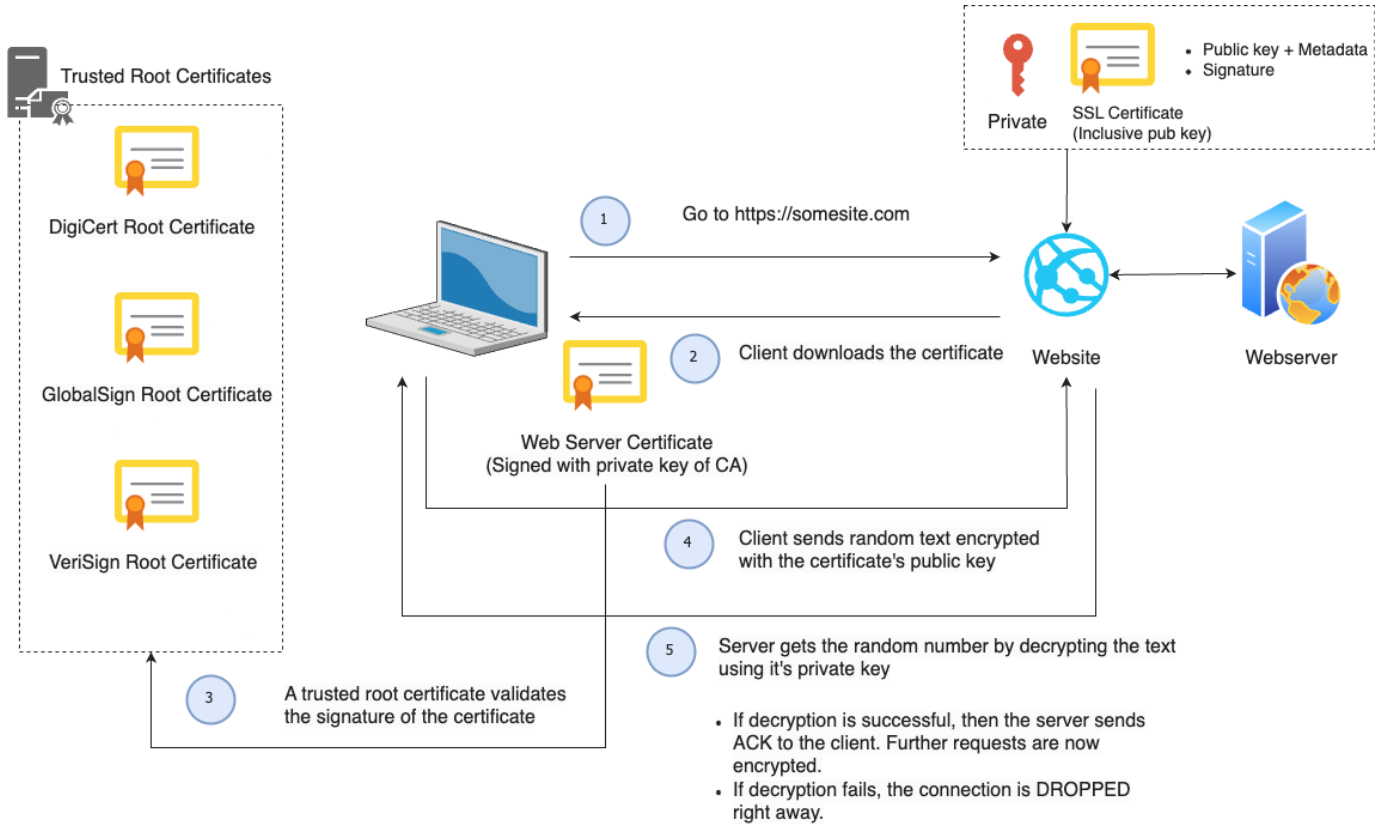
To give you a better understanding of why your browser will warn you when using self signed certificates, we have to know that when we request an SSL certificate from an official Certificate Authority (CA) that you submit a Certificate Signing Reauest (CSR) to them. They in return provide you with a Signed SSL certificate. For this they make use of their root certificate and private key. Our browser comes with a copy of the root certificate (CA) from various authorities or it can access it from the OS. This is why our self signed certificates are not trusted by our browser, we don't have any CA validation. Our only workaround is to create our own root certificate and private key.

18.1.1 Understanding the concepts

How to create an SSL certificate



18.1.2 How SSL works - Client - Server flow



Note

Borrowed the designs from https://www.youtube.com/watch?v=WqgzYuHtnIM this video explains well how SSL works.

18.1.3 Securing the Frontend with Self signed SSL on Nginx

To configure this there are a few steps that we need to follow:

```
- Generate a private key for the CA ( Certificate Authority )
- Generate a root certficate
- Generating CA-Authenticated Certificates
- Generate a Certificate Signing Request (CSR)
- Generate an X509 V3 certificate extension configuration file
- Generate the certificate using our CSR, the CA private key, the CA certificate, and the config file
- Copy the SSL certificates to our Virtual Host
- Adapt your Nginx Zabbix config
```

Generate a private key for the CA

First step is to make a folder named SSL so we can create our certificates and safe them:

```
>- mkdir ~/ssl
>- cd ~/ssl
>- openssl ecparam -out myCA.key -name prime256v1 -genkey
```

Let's explain all the options;

- openssl : The tool to use the OpenSSL library, this library provides us with cryptographic functions and utilities.
- out myCA.key : This part of the command specifies the output file name for the generated private key.
- name prime256v1: Name of the elliptic curve; X9.62/SECG curve over a 256 bit prime field
- ecparam: This command is used to manipulate or generate EC parameter files.
- genkey: This option will generate a EC private key using the specified parameters.

Generate a Root Certificate

```
openssl req -x509 -new -nodes -key myCA.key -sha256 -days 1825 -out myCA.pem
```

Let's explain all the options;

- openssl: The command-line tool for OpenSSL.
- req: This command is used for X.509 certificate signing request (CSR) management.
- x509: This option specifies that a self-signed certificate should be created.
- new: This option is used to generate a new certificate.
- nodes: This option indicates that the private key should not be encrypted. It will generates a private key without a passphrase, making it more convenient but potentially less secure.
- key myCA.key: This specifies the private key file (myCA.key) to be used in generating the certificate.
- sha256: This option specifies the hash algorithm to be used for the certificate. In this case, SHA-256 is chosen for stronger security.
- days 1825: This sets the validity period of the certificate in days. Here, it’s set to 1825 days (5 years).
- out myCA.pem: This specifies the output file name for the generated certificate. In this case, “myCA.pem.”

The information you enter is not so important but it's best to fill it in as good as possible. Just make sure you enter for CN you IP or DNS.

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:BE
State or Province Name (full name) []:vlaams-brabant
Locality Name (eg, city) [Default City]:leuven
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:192.168.0.134
Email Address []:
```

Generating CA-Authenticated Certificates

It's probably good practice to use de dns name of your webiste in the name fo the private key.
As we use in this case no DNS but an IP address I will use the fictive dns zabbix.mycompany.internal.

```
openssl genrsa -out zabbix.mycompany.internal.key 2048
```

Generate a Certificate Signing Request (CSR)

```
openssl req -new -key zabbix.mycompany.internal.key -out zabbix.mycompany.internal.csr
```

You will be asked the same set of questions as above. Once again your answers hold minimal significance and in our case no one will inspect the certificate so they matter even less.

```
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
```

```
-----
Country Name (2 letter code) [XX]:BE
State or Province Name (full name) []:vlaams-brabant
Locality Name (eg, city) [Default City]:leuven
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:192.168.0.134
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Generate an X509 V3 certificate extension configuration file.

```
# vi zabbix.mycompany.internal.ext
```

Add the following lines in your certificate extension file. Replace IP or DNS with your own values.

```
authorityKeyIdentifier=keyid,issuer
basicConstraints=CA:FALSE
keyUsage = digitalSignature, nonRepudiation, keyEncipherment, dataEncipherment
subjectAltName = @alt_names

[alt_names]
IP.1 = 192.168.0.133
#DNS.1 = MYDNS (You can use DNS if you have a dns name if you use IP then use the above line)
```

Generate the certificate using our CSR, the CA private key, the CA certificate, and the config file

```
openssl x509 -req -in zabbix.mycompany.internal.csr -CA myCA.pem -CAkey myCA.key \
-CAcreateserial -out zabbix.mycompany.internal.crt -days 825 -sha256 -extfile zabbix.mycompany.internal.ext
```

Copy the SSL certificates to our Virtual Host

```
cp zabbix.mycompany.internal.crt /etc/pki/tls/certs/.
cp zabbix.mycompany.internal.key /etc/pki/tls/private/.
```

IMPORT THE CA IN LINUX (RHEL)

We need to update the CA certificate's, run the below command to update the CA certs.

```
cp myCA.pem /etc/pki/ca-trust/source/anchors/myCA.crt
update-ca-trust extract
```

IMPORT THE CA IN OSX

- Open the macOS Keychain app.
- Navigate to File > Import Items
- Choose your private key file (i.e., myCA.pem)
- Search for the "Common Name" you provided earlier.
- Double-click on your root certificate in the list.
- Expand the Trust section.
- Modify the "When using this certificate:" dropdown to "Always Trust".
- Close the certificate window.

IMPORT THE CA IN WINDOWS

- Open the "Microsoft Management Console" by pressing Windows + R, typing mmc, and clicking Open.
- Navigate to File > Add/Remove Snap-in.
- Select Certificates and click Add.
- Choose Computer Account and proceed by clicking Next.
- Select Local Computer and click Finish.

- Click OK to return to the MMC window.
- Expand the view by double-clicking Certificates (local computer).
- Right-click on Certificates under “Object Type” in the middle column, select All Tasks, and then Import.
- Click Next, followed by Browse. Change the certificate extension dropdown next to the filename field to All Files (.) and locate the myCA.pem file.
- Click Open, then Next.
- Choose “Place all certificates in the following store.” with “Trusted Root Certification Authorities store” as the default. Proceed by clicking Next, then Finish, to finalize the wizard.
- If all went well you should find your certficate under Trusted Root Certification Authorities > Certificates

Warning

You also need to import the myCA.crt file in your OS we are not an official CA so we have to import it in our OS and tell it to trust this Certificate. This action depends on the OS you use.

As you are using OpenSSL, you should also create a strong Diffie-Hellman group, which is used in negotiating Perfect Forward Secrecy with clients. You can do this by typing:

```
openssl dhparam -out /etc/ssl/certs/dhparam.pem 2048
```

ADAPT YOUR NGINX ZABBIX CONFIG

Add the following lines to your Nginx configuration, modifying the file paths as needed.
Replace the the already existing lines with port 80 with this configuration. This will enable SSL and HTTP2.

```
# vi /etc/nginx/conf.d/zabbix.conf
```

```
server {
        listen          443 http2 ssl;
        listen          [::]:443 http2 ssl;
        server_name     <ip qddress>;
        ssl_certificate /etc/ssl/certs/zabbix.mycompany.internal.crt;
        ssl_certificate_key /etc/pki/tls/private/zabbix.mycompany.internal.key;
        ssl_dhparam /etc/ssl/certs/dhparam.pem;
```

To redirect traffic from port 80 to 443 we can add the following lines above our https block:

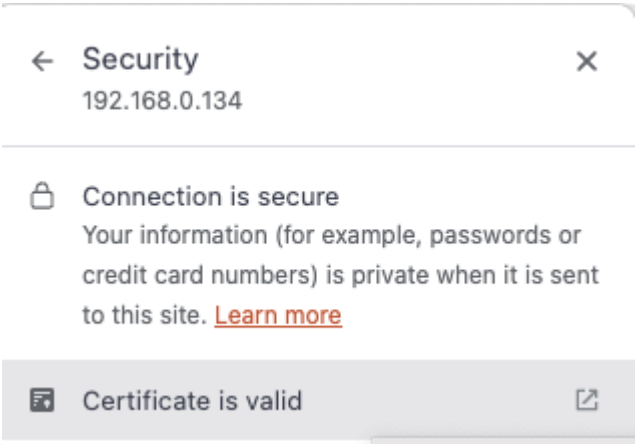
```
server {
       listen         80;
       server_name    _; #dns or ip is also possible
       return         301 https://$host$request_uri;
}
```

RESTART ALL SERVICES AND ALLOW HTTPS TRAFFIC

```
systemctl restart php-fpm.service
systemctl restart nginx

firewall-cmd --add-service=https --permanent
firewall-cmd --reload
```

When we go to our url `http://<IP or DNS>/` we get redirected to our `https://` page and when we check we can see that our site is secure:



Note

- To be even more secure have a loot at https://cipherlist.eu/ this page maintains a list of strong ciphers that you can use so secure your Nginx even more.
- You can test your nginx config with 'nginx -t' before you restart.
- For HTTP/2 to work you need atleast nginx 1.9.5 or later

18.1.4 Securing the Frontend with Let's Encrypt on Nginx

19. Maintaining Zabbix

19.1 Maintaining Zabbix

20. Monitoring Windows

20.1 Monitoring Windows

21. Zabbix API

21.1 Zabbix API