

Zabbix

- [Zabbix Install with Docker](#)
 - [Installing Zabbix Monitor with Docker](#)

Zabbix Install with Docker

Installing Zabbix Monitor with Docker

<https://www.youtube.com/embed/piVlj8UPLFI>

UPDATED: June 6, 2024 with newer instructions

I've been asked about Zabbix for a while now. I have covered some other solutions for monitoring equipment / services in the past. CheckMK, using Dashy Widgets, Glances, NetData, and several others are some really great options for monitoring, and as with any software, the right thing for you will depend greatly on your needs. As I started looking at Zabbix, I was contacted by Marc over at [OneMarcFifty](https://www.youtube.com/channel/UC1vVlj8UPLFI) about doing a collaborative video series with him on Zabbix. He asked if I would be interested in covering the install, and he would cover some more in-depth setup of getting monitored systems enrolled, as well as setting up email alerts for anything the system finds.

How could I say, "No" to such a great opportunity? Of course I was interested. Marc does some absolutely amazing content, and I have watched his channel for a couple of years now. He has some incredibly great content on all kinds of tech topics, and his explanations are just terrific, so definitely jump over to his channel for the second part of this tutorial once you've got Zabbix up and running. You can find the video right here.

Installation

What you'll need

- A system with Docker-CE and Docker-Compose installed (we'll call this the Host Server for this tutorial)
- SSH Access to the Host Server
- Git, Curl, Wget installed on the Host Server
- About 30 Minutes of your time

Installation of Docker via a Simple Script

You can easily install Docker-CE, Docker-Compose, Portainer-CE, and NGinX Proxy manager by using this quick install script I created and maintain on Github. Just use the command:

```
wget -O install-docker.sh https://gitlab.com/bmcgonag/docker\_installs/-/raw/main/install\_docker\_nproxyman.sh
```

To download the script to your desired host.

Change the permissions to make the script executable:

```
chmod +x ./install-docker.sh
```

and then run the script with the command:

```
./install-docker.sh
```

When run, the script will prompt you to select your host operating system, then will ask you which bits of software you want to install.

Simply enter 'y' for each thing you want to install. In this case, you really only need to install Docker-CE and Docker-Compose. Feel free to answer 'n' to the other software options.

At some point, you may be asked for your super user (sudo) password as well.

Allow the script to complete installation.

At this point, you might want to log out and back in, as this will allow you to use the `docker` and `docker-compose` commands without the need of sudo in front of them.

Alternatively, you can try the following commands:

```
newgrp
```

```
newgrp docker
```

Now you can test your ability to run a docker command by doing

```
docker ps
```

If you don't get any errors, you are good to go.

Installing Zabbix

Now that we have our server software setup, we'll start on our Zabbix installation. Fortunately, Zabbix provides a nice set of ready to use docker-compose options for us. We'll clone the Zabbix-

Docker repository from github, and then make a few modifications to some specific files to set our Environment Variables, and then we'll be ready to run.

First, we need to make sure we have git, curl, and wget installed on our Host Server. We can install them on Debian / Ubuntu with

```
sudo apt install git curl wget -y
```

On Fedora, CentOS, Redhat we should be able to use:

```
sudo dnf install git curl wget -y
```

On Arch, you should be able to use

```
sudo pacman -S git curl wget
```

For OpenSuse, you should use

```
sudo zypper install git curl wget
```

Once those few dependencies are installed, we'll use the git command to pull down the Zabbix-Docker repository to our local machine. First, however, let's move into the "docker" folder. If you don't already have a top level folder to keep all of your docker applications in, I highly recommend you set one up, as you can then simply compress and backup the top level folder and have all of your docker applications and data backed up with one command.

```
cd docker
```

Now let's clone that repository:

```
git clone https://github.com/zabbix/zabbix-docker.git
```

This will create a folder called "zabbix-docker" with all of the files from the repository in it. We'll move into that folder with

```
cd zabbix-docker
```

In this folder you'll find many docker-compose template files. These are labeled in a way that is fairly easy to understand. Each file that has "local" in the name, means that the compose file will attempt to build new images when run, versus the files that only have "latest" and not "local" in the name. These files will pull down pre-built images from dockerhub for us to use in our system.

In this updated version we'll be using the three files labeled "compose.yaml", "compose_databases.yaml", and "compose_zabbix_components.yaml".

Next, we need to set a few environment variables. These variables are super useful, especially in large projects like Zabbix, because you can set a value one time, and it is reused throughout the project. This reduces issues with misspellings, mismatched values throughout a docker-compose file, etc.

We'll move into the environment variable folder of the project with the command:

```
cd env_vars
```

In this folder, are a group of hidden files. In Linux / Unix based systems, hidden files are set by the use of a dot / period in front of the file name. In order to see these files in our directory we use the flag "a" with the "ls" command. We can also use the flag "l" to list out the file permissions, and make them list vertically down the screen. So, let's list out the files:

```
ls -al
```

We need to set a few variables. Most of these files we will leave untouched, however.

First, we'll edit the following files since we are using the mysql version of the docker-compose they provide.

- MYSQL_PASSWORD
- MYSQL_ROOT_PASSWORD

We don't need to change the POSTGRES_PASSWORD, POSTGRES_USER values unless you prefer to use Postgres DB, in which case change the password file at the very least.

In order to edit each file, you'll use the following command structure:

```
nano <filename including the leading dot ".">
```

Make the change to the value, then save using CTRL+O, then Enter to confirm, and CTRL+X to exit the nano editor.

We'll use the .MYSQL_PASSWORD as our first example:

```
nano .MYSQL_PASSWORD
```

You should then see this following

```
zabbix
```

Which you'll change to a long password (32 characters or more) with upper and lower case characters and numbers all mixed in.

Then save, with CTRL+O, and Enter to confirm, then CTRL+X to exit the nano editor.

Repeat this process for the .MYSQL_ROOT_PASSWORD as well, using a different password.

Next, we may want to make a change in a couple of the other files in this directory. I make a few modifications in the video, but they are not necessary. If you want to make changes, be certain you understand how those changes effect the system overall.

For instance, in the "compose_zabbix_components.yaml" file, I need / want to change the web-nginx port mappings, because it is set to 8080 and 8443 on the host by default, and I'm already running an application (Zammad) that uses these ports on the host.

You can see if your port 8080 is being used on your host by running the command

```
sudo lsof -i -P -n | grep LISTEN
```

and look at the ports listed. Each port listed is already in use on the system.

To change the ports, you can do

```
nano compose_zabbix_components.yaml
```

Once in the file, use the CTRL + W hotkey combo to open the search feature. Then, type in 'web-nginx' and hit Enter to search.

When you reach the web-nginx section, look for the 'ports:' sub-section and edit the left side of the port mapping. It will be a long string of text surrounded by curly braces.

```
web-nginx:
  ports:
    - "${ZABBIX_WEB_NGINX_HTTP_PORT}:8080"
    - "${ZABBIX_WEB_NGINX_HTTPS_PORT}:8443"
```

Replace `${ZABBIX_WEB_NGINX_HTTP_PORT}` with a port number not in use on your machine, and above 8000 preferably. I used 8052 for mine. You can also replace `${ZABBIX_WEB_NGINX_HTTPS_PORT}` with a port number if you wish, I used 9043.

When done my section looked like this:

```
web-nginx:
  ports:
    - "$8052:8080"
    - "9043:8443"
```

Save your changes with CTRL + O, then press Enter to confirm, and use CTRL + X to exit the nano editor.

Once, you've made changes to the necessary environment variables and ports, we are ready to run our docker-compose.yml file.

Now we can run:

```
docker-compose up -d && docker-compose logs -f
```

Be patient. The initial startup takes a good bit. I'd say as long as you don't see any errors, let it run. Go get some coffee, or just chill on YouTube watching the AwesomeOpenSource channel for a bit.

This really runs two commands: The part before the "&&" will pull down the zabbix images, and create new containers for us. The part after the "&&" will show us the logs of Zabbix starting up after the containers are started.

You may see some warnings in the logs about the limitation settings for CPU, Memory, etc. This is because docker-compose is not an orchestrator. In docker-compose v3 and later, the limit values are ignored. Such values are used in docker-swarm only.

This is a minimal startup of the base services needed to start up Zabbix server and be able to connect external clients to it. But there are many more services that can be started up. I show this in the video. Once you've started Zabbix successfully, setup your reverse proxy for a fully qualified domain name (FQDN), and have https working for your site, we can start these other services.

To start them, we'll go back to the terminal and enter the command:

```
docker compose --profile all pull
```

This will pull down several more images to be used in our server, including the dockerized version of the agent, so we can monitor our docker based Zabbix server.

Once those have all pulled down, we'll start them up (without having to stop anything that's already running) by doing the command:

```
docker compose --profile all up -d
```

You'll see the new containers being started up, be patient while everything starts. This can all take a while, and it's especially hardware and resource dependent.

Once started, you should be able to do:

```
docker compose ps
```

and see a whole list of running containers on your system.

Next we need to setup the agent container to be able to talk to the server container, and get it setup as Host in Zabbix. Check out the video on how to setup a Host through the Zabbix Web UI, but I'll give you the commands to use for getting the right information from the two containers, here, as it's a bit more involved.

Setting Up the Zabbix Container Agent to Monitor the Zabbix Container Server

First, we need a couple of IP addresses from our containers. In particular, you'll want the one from the zabbix-agent container, and the zabbix-server container. We can get these by using a couple

of docker commands. First, we'll list out our containers with

```
docker ps
```

This will show us all of our running containers. We are really interested only in the ones with zabbix in the name. Find the container called "zabbix-docker-zabbix-server-1", and then locate its container id, and highlight and copy it. You can right click and select "Copy", or use CTRL + Shift + C in the terminal to copy.

now, enter

```
docker inspect <id you just copied>
```

It should look something like

```
docker inspect 8cf2731bef1d
```

It's worth noting, your ID will be different.

When you get the output of the inspect command, at the end will be a segment for network information. There you'll see a key for the ip address of this container inside the docker network. We need that address.

You should see some information like this:

```
"zabbix-docker_frontend": {
  "IPAMConfig": null,
  "Links": null,
  "Aliases": [
    "zabbix-docker-zabbix-server-1",
    "zabbix-server"
  ],
  "MacAddress": "02:42:ac:10:ee:03",
  "NetworkID": "fd61396dda648b01b00a4f23d2640577823230923e8b25d71bf1b2339ac",
  "EndpointID": "ef3a483717dc831856d9f14gnei48vansu74ifj439afjaf7766e5ffe810389e9315",
  "Gateway": "172.16.239.1",
  "IPAddress": "172.16.239.3",
  "IPPrefixLen": 24,
  "IPv6Gateway": "",
  "GlobalIPv6Address": "",
  "GlobalIPv6PrefixLen": 0,
  "DriverOpts": null,
  "DNSNames": [
    "zabbix-docker-zabbix-server-1",
```

```
        "zabbix-server",
        "8cf2722bef1d"
    ]
}
```

We need the IP address listed here as `"IPAddress": "172.16.239.3",`.

We need to do the same thing for our container "zabbix-docker-zabbix-agent-1". Get the ID, do the docker inspect on the ID, and grab it's IPAddress value from the JSON output.

You should put these somewhere so you can grab them quickly again. I made a simple text file, and marked them as

Agent IP: 172.16.239.6

ServerIP: 172.16.239.3

Now we need to jump into the agent docker container, and edit a file that will tell the agent where to communicate with the server.

```
docker exec -it zabbix-docker-zabbix-agent-1 /bin/bash
```

This will get you into the container at a new command prompt. From here we'll edit our file located at `/etc/zabbix/zabbix_agentd.conf`

To do this, we'll have to use the VI editor, as the container does not have nano installed, but no big deal.

```
vi /etc/zabbix/zabbix_agentd.conf
```

Now, press the 'i' key on the keyboard, to go into 'insert' mode in VI. Scroll down to the section where you'll see a line that says

```
SourceIP=127.0.0.1
```

Remove the '127.0.0.1' and replace it with the agent IP that we got earlier.

```
SourceIP=172.16.239.6
```

Next, we'll scroll down until we see a section for the 'Server'.

```
### Option: Server
#   List of comma delimited IP addresses, optionally in CIDR notation, or DNS names of Zabbix servers and
#   Zabbix proxies.
#   Incoming connections will be accepted only from the hosts listed
#   here.
#   If IPv6 support is enabled then '127.0.0.1', '::127.0.0.1', '::ffff:127.0.0.1' are treated
```

```
equally
#    and '::/0' will allow any IPv4 or IPv6
address.
#    '0.0.0.0/0' can be used to allow any IPv4
address.
#    Example:
Server=127.0.0.1,192.168.1.0/24,::1,2001:db8::/32,zabbix.example.com

#
# Mandatory: yes, if StartAgents is not explicitly set to
0
# Default:
# Server=

Server=zabbix-server
```

We want to remove 'zabbix-server' and instead enter the IP address for the server we copied.

```
Server=172.16.239.3
```

Next, we want to do the exact same thing for the 'ServerActive' section just a little bit down from the 'Server' section.

```
ServerActive=zabbix-server:10051
```

We want to remove 'zabbix-server:10051' and enter the Server IP we copied, again.

```
ServerActive=172.16.239.3
```

Now we can save by pressing the escape key, Esc, to exit insert mode. Then, we need to type `:wq` which tells VI to write (save) the changes, and (q)uit out of VI.

Now we are back at the prompt inside our agent container, so let's type `exit` and leave our container area, and we'll be back at our normal prompt.

We'll restart the agent with

```
docker restart zabbix-docker-zabbix-agent-1
```

Give it a minute or so, and go back to your Zabbix Server Web UI, and you should see that the server host has been updated (this assumes you've gone in the Web UI and changed the IP address in the host entry under the Monitoring menu). You need to change the IP from 127.0.0.1 in the Host entry, to be the IP of the agent you copied earlier. Save / Update that host, and give it a minute.

Use Ansible to Install the Client / Agent

If you've seen my video on Ansible, here's my take on getting the Zabbix Agent installed on multiple client machines at once.

```
---
- hosts: all
  become: true
  tasks:

    - name: Install the Zabbix Agent on Ubuntu
      apt:
        name: zabbix-agent
        update_cache: true

    - name: Configure the Zabbix agent
      replace:
        path: /etc/zabbix/zabbix_agentd.conf
        regexp: '{{item.regexp}}'
        replace: '{{item.replace}}'
      with_items:
        - {regexp: "^Server=127.0.0.1$", replace: "Server={{zabbix_server}}"}
        - {regexp: "^ServerActive=127.0.0.1$", replace: "ServerActive={{zabbix_server}}"}
        - {regexp: "^Hostname=Zabbix server$", replace: "Hostname={{ansible_hostname}}"}
        - {regexp: "^# SourceIP=$", replace: "SourceIP={{ansible_facts['wt0']['ipv4']['address']}}"}
      vars:
        zabbix_server: 100.87.204.118

    - name: Start the Zabbix agent service
      service:
        name: zabbix-agent
        state: started
        enabled: yes

    - name: Set API token
      ansible.builtin.set_fact:
        ansible_zabbix_auth_key: "<you create this through your zabbix web admin interface>"

    - name: Create a new host or rewrite an existing host's info
      vars:
        ansible_network_os: community.zabbix.zabbix
        ansible_connection: httpapi
```

```
ansible_httpapi_port: 443
ansible_httpapi_use_ssl: true
ansible_httpapi_validate_certs: false
ansible_zabbix_url_path: ""
ansible_host: monitor.sysmainit.com
become: false
community.zabbix.zabbix_host:
  host_name: "{{ansible_hostname}}"
  visible_name: "{{ansible_hostname}}"
  description:
  host_groups:
    - SysMainIT
    - Linux servers
  link_templates:
    - Linux by Zabbix agent
  status: enabled
  state: present
  inventory_mode: manual
  inventory_zabbix:
    tag: "SysMainIT"
    alias: "SysMainIT"
    notes: ""
    location: ""
    site_rack: ""
    os: "Ubuntu 22.04 LTS"
    hardware: "Digital Ocean VPS"
    interfaces:
      - type: 1
        main: 1
        useip: 1
        ip: "{{ansible_facts['wt0']['ipv4']['address']}}"
        dns: ""
        port: "10050"

- name: Restart the Zabbix agent service
  service:
    name: zabbix-agent
    state: restarted
    enabled: yes
```

Make sure to update any necessary details in the above. There may be easier, better, or more efficient ways to do this, but I did use this to push out the agent to about 20 Ubuntu machines at once, and it was very quick.

You will of course need a hosts file for this playbook to run from, but once you've got that setup, and have SSH access for your ansible user.

Troubleshooting

One issue I came across during my testing of how to get all of this up and running, was that on one of my servers, my docker-compose version was not new enough. It would continuously give me an error about "profiles" not being supported. The key is to get a newer version of docker-compose installed. Depending on the version of your Linux Distribution, you may have to add a more recent repository in order to update your docker-compose version, or do more of a manual install of docker-compose. Since there are so many variables on what OS may need which setup, it's not feasible for me to try and guide you further on the topic. I ended up on docker-compose version 1.29.x and it worked without issue. ON 1.25.x I was getting the issue with the "profiles" section of the docker-compose file.

The Zabbix Web Interface

You should now be able to access the Zabbix Web Interface using your web browser of choice. You'll go to the IP address of your host machine. In my case I installed it on a machine with the IP 192.168.10.42. So in my web browser I type:

<http://192.168.10.42>

If you changed the 80:8080 port mapping, make sure to add the port to your IP. If I had change it from 80:8080 to 8022:8080, I would then enter the following into my browser url bar:

<http://192.168.10.42:8022>

Once you see the login page (be patient, as it could take a few minutes for Zabbix to come up the first time), use the username "Admin" and password "zabbix" to login.

You should immediately navigate to the User Settings >> Profile in the left navigation bar to change your admin user password to a long, strong password.

You are now ready to begin enrolling devices into your Zabbix monitoring solution. This is a massive system with an incredible amount of power. Take your time, get to know what all it's capable of doing, and what information you can gain from it. A system like this is worth the time you'll put into making it do as much as you can.

In the video, I go through setting up one client machine. It's a manual process, but worth watching. Make sure to watch Marc's follow up video over [@OneMarcFifty](#) where he will help you unlock more of the power available in this awesome open source system.

<https://www.youtube.com/embed/DFdDEf5iib4>

Support My Channel and Content

Support my Channel and ongoing efforts through Patreon:

<https://www.patreon.com/awesomeopensource>

Buy me a Beer / Coffee:

<https://paypal.me/BrianMcGonagill>