

DATA2410-1 22V Datanettverk og Skytjenester  
GROUP Portfolio Assignment 2 - Docker and Zabbix Real Use-Case

Ulrik Bakken Singsaas | s351917  
Adrian Bakstad | s341507  
Thea Emilie Haugen | s351879  
Andrea Bjørge | s344175

April 25, 2022

# Contents

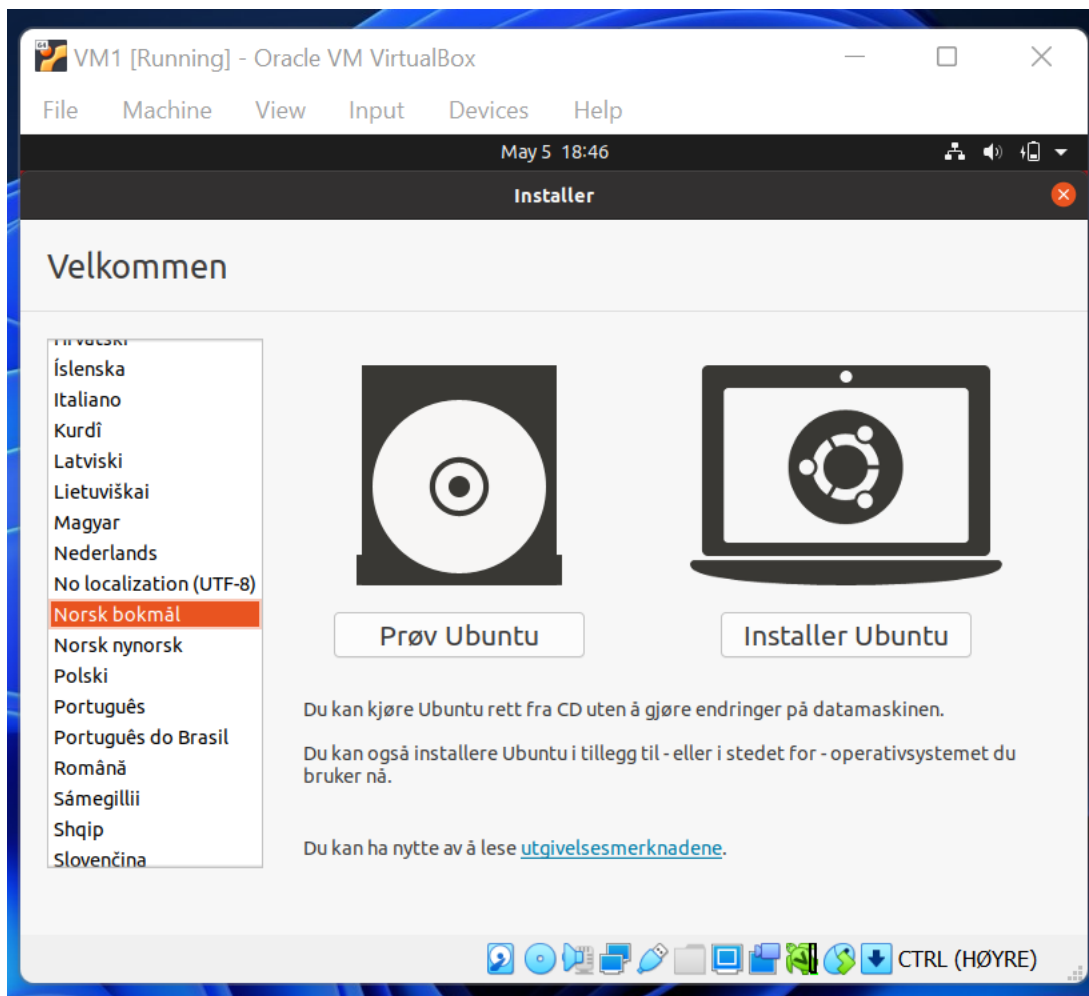
<b>1. Introduction</b>	<b>2</b>
1.1. Virtual Machines with VirtualBox . . . . .	2
<b>2. VM1: Docker containers setup</b>	<b>4</b>
2.1. Quad Container Setup on intel1 / VM1 . . . . .	4
<b>3. VM2 and VM3: Install zabbix-agent and zabbix-proxy</b>	<b>7</b>
3.1. VM2 . . . . .	7
3.1.1. Installing Zabbix Proxy . . . . .	7
3.1.2. Configuring MariaDB database for the proxy to use . . . . .	7
3.1.3. Configuring Zabbix Proxy . . . . .	8
3.1.4. Starting and enabling the Zabbix Proxy . . . . .	9
3.1.5. Registering Zabbix Proxy in the Zabbix frontend . . . . .	9
3.2. VM 3 . . . . .	10
<b>4. VM2: Nginx proxy</b>	<b>12</b>
<b>5. VM1: Zabbix frontend</b>	<b>13</b>
5.1. Items . . . . .	14
5.2. Triggers . . . . .	14
<b>6. References</b>	<b>16</b>

## 1. Introduction

### 1.1. Virtual Machines with VirtualBox

The first thing we needed was a VM running Ubuntu Focal Fossa. Cloning this VM would let us create the 3 virtual machines needed for the assignment, each with 4GB of RAM and 10GB of disk space.

Therefore, We started by downloading the image for Ubuntu Focal Fossa (20.04) from: <https://releases.ubuntu.com/20.04/ubuntu-20.04.4-desktop-amd64.iso> and creating VM1.



We set up VM1 on a bridged network, configuring it first to make sure it was working. Then, we cloned it and changed the mac address to create VM2. We cloned VM1 one more time to create VM3. Changing the mac addresses lets each VM have each their local IP on the bridged network. By doing it this way, we made sure that all three VMs could communicate with each other, whilst also being able to communicate with the host machine.

The architecture diagram in the assignment description can be interpreted to mean that we should use an internal network for all of the VMs whilst giving VM2 a second bridged network adapter. This would ensure that only the nginx proxy could reach the outside of the internal VM network. However, this would make our assignment more complicated than necessary, because the assignment didn't specify what network method to use for the VMs.

## 2. VM1: Docker containers setup

Originally, we attempted to use docker containers on the intel1-server to implement our solution. However, the server ran out of storage space, so we created virtual machines through VirtualBox as a substitute.

These are the commands we did to install docker, and in the block beneath we display our docker compose file configs/docker-compose.yml, which was run with the docker-compose up command on VM1, in the same directory as this file

```
sudo apt-get install -y docker-compose
```

### 2.1. Quad Container Setup on intel1 / VM1

After setting up the three VMs, we used the file docker-compose.yml, to set up the four containers with the required config instructions for the assignment within VM1. This file can be found in the configs folder.

```
# Docker Container setup for VM 1
version: "3.0"
services:
  # mysql container setup
  mysql-server:
    image: haakco/mysql80
    ports:
      - 3306
    hostname: mysql-server
    restart: unless-stopped
    volumes:
      - mysql-server-data:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=123
      - MYSQL_DATABASE=zabbix
      - MYSQL_USER=zabbix
      - MYSQL_PASSWORD=123
    cap_add:
      - SYS_NICE
    networks:
      zabbix-net:
        ipv4_address: 172.200.1.1
        # not really supposed to have two commands put putting it on one line broke things and this seems to work
        # tried this, but it did not work:
        # command: bash -c "--default-authentication-plugin=mysql_native_password && --datadir=/var/lib/mysql/c
    command: --default-authentication-plugin=mysql_native_password
    command: --datadir=/var/lib/mysql/data

  # zabbix server setup
  zabbix-server:
    image: zabbix/zabbix-server-mysql
    ports:
      - 10051:10051
    hostname: zabbix-server
    restart: unless-stopped
    volumes:
      - zabbix-server-config:/etc/zabbix
      - docs:/usr/share/doc/
    environment:
      - DB_SERVER_HOST=mysql-server
      - MYSQL_DATABASE=zabbix
      - MYSQL_USER=zabbix
```

```

    - MYSQL_PASSWORD=123
depends_on:
    - mysql-server
networks:
    zabbix-net:
        ipv4_address: 172.200.1.2

# zabbix web container
zabbix-web:
    image: zabbix/zabbix-web-nginx-mysql
    ports:
        - 80:8080
    hostname: zabbix-web
    restart: unless-stopped
    volumes:
        - zabbix-web-config:/etc/zabbix
    environment:
        - DB_SERVER_HOST=mysql-server
        - MYSQL_DATABASE=zabbix
        - MYSQL_USER=zabbix
        - MYSQL_PASSWORD=123
        - ZBX_SERVER_HOST=zabbix-server
    depends_on:
        - mysql-server
        - zabbix-server
    networks:
        zabbix-net:
            ipv4_address: 172.200.1.3

# zabbix agent container
zabbix-agent:
    image: zabbix/zabbix-agent
    ports:
        - 10050:10050
    hostname: zabbix-agent
    restart: unless-stopped
    volumes:
        - zabbix-agent-config:/etc/zabbix
    environment:
        - ZBX_SERVER_HOST=zabbix-server
    depends_on:
        - zabbix-server
    networks:
        zabbix-net:
            ipv4_address: 172.200.1.4

# custom network for the containers
networks:
    zabbix-net:
        driver: bridge
        ipam:
            config:
                - subnet: 172.200.1.0/16

# volumes for the containers
volumes:

```

```
mysql-server-data:
  external: true
zabbix-server-config:
  external: true
zabbix-web-config:
  external: true
zabbix-agent-config:
  external: true
docs:
  external: true
```

When setting up the docker containers, we mostly used the auto generated docker files, but we made some minor adjustments by setting the environment variables for these files before they were generated. The environment variables were created outside the `docker-compose.yml` file, and later referenced in the `docker-compose.yml` file. We created volume links as external volumes, to make it possible to edit the files outside the docker containers via the docker volume functionality, since these volume files are synchronized with the volume files we mapped them to. The `docs` volume was used to get the `.sql` file to create the server. Setting up volumes for outside access made debugging easier while working on the project. For example we utilized this setup to look at the `zabbix_server.conf` config file to see if the environment variables in the `docker-compose.yml` file was correctly written in the config file.

The following block of code describes how we set up the volumes according to the description above.

```
sudo docker volume create mysql-server-data
sudo docker volume create zabbix-server-config
sudo docker volume create zabbix-web-config
sudo docker volume create zabbix-agent-config
sudo docker volume create docs
```

After the docker containers were up and running, we decided to set up host profiles for active and passive checks between the zabbix agent and server in the docker stack. This was to ensure that everything connected properly. The web frontend is hosted on VM1 port 80 as per the assignment description, so to reach it, we simply typed the address in a web browser. In later steps of the assignment, we set up an nginx proxy that allowed us to reach the frontend through the localhost-address.

zabbix-agent	Items 66	Triggers 34	Graphs 13	Discovery 3	Web		Linux by Zabbix agent active	Enabled	None	
zabbix_server	Items 67	Triggers 34	Graphs 13	Discovery 3	Web	172.200.1.4:10050	Linux by Zabbix agent	Enabled	ZBX	None

Figure 1: Active and Passive checks on the

Here is a screen of our docker compose log, showing that all the checks except for one is working between the agent and server. Our thoughts was that this one check from the template probably wasn't suited for being run in a docker environment as some things can be different there.04\_all

```
zabbix-web_1 | 192.168.50.247 - - [12/May/2022:11:09:34 +0000] "POST /zabbix.php?action=notifications.get&sid=b6bbe17608fd0db2&output=ajax
ion=host.list" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:100.0) Gecko/20100101 Firefox/100.0" 192.168.50.206"
zabbix-web_1 | 192.168.50.247 - - [12/May/2022:11:09:40 +0000] "POST /jsrpc.php?output=json-rpc HTTP/1.0" 200 62 "http://192.168.50.247:808
; Win64; x64; rv:100.0) Gecko/20100101 Firefox/100.0" 192.168.50.206"
zabbix-web_1 | 192.168.50.247 - - [12/May/2022:11:09:50 +0000] "POST /jsrpc.php?output=json-rpc HTTP/1.0" 200 62 "http://192.168.50.247:808
; Win64; x64; rv:100.0) Gecko/20100101 Firefox/100.0" 192.168.50.206"
zabbix-web_1 | 192.168.50.247 - - [12/May/2022:11:10:01 +0000] "POST /jsrpc.php?output=json-rpc HTTP/1.0" 200 62 "http://192.168.50.247:808
; Win64; x64; rv:100.0) Gecko/20100101 Firefox/100.0" 192.168.50.206"
zabbix-web_1 | 192.168.50.247 - - [12/May/2022:11:10:04 +0000] "POST /zabbix.php?action=notifications.get&sid=b6bbe17608fd0db2&output=ajax
ion=host.list" "Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:100.0) Gecko/20100101 Firefox/100.0" 192.168.50.206"
zabbix-agent_1 | 78:20220512:111009.996 active check "system.sw.packages" is not supported: Cannot obtain package information.
zabbix-web_1 | 192.168.50.247 - - [12/May/2022:11:10:11 +0000] "POST /jsrpc.php?output=json-rpc HTTP/1.0" 200 63 "http://192.168.50.247:808
; Win64; x64; rv:100.0) Gecko/20100101 Firefox/100.0" 192.168.50.206"
zabbix-server_1 | 188:20220512:111014.632 item "zabbix-agent:system.sw.packages" became not supported: Cannot obtain package information.
zabbix-web_1 | 192.168.50.247 - - [12/May/2022:11:10:21 +0000] "POST /jsrpc.php?output=json-rpc HTTP/1.0" 200 63 "http://192.168.50.247:808
; Win64; x64; rv:100.0) Gecko/20100101 Firefox/100.0" 192.168.50.206"
```

Figure 2: Logs from docker compose after setting up hosts on frontend

So this should fully explain how we did the first part of this assignment, how we installed docker and how we configured our docker-compose stack, and made our docker bridge network inside of VM1, whilst also going a little further with using the frontend with setting up the hosts there.

## 3. VM2 and VM3: Install zabbix-agent and zabbix-proxy

### 3.1. VM2

We followed this guide to complete task 1 in part III of the assignment description: <https://bestmonitoringtools.com/install-zabbix-proxy-on-ubuntu/> There were a few things we did differently from the guide. These will be described below.

#### 3.1.1. Installing Zabbix Proxy

We started by installing Zabbix Proxy on VM2 with the following commands:

NB: These links are not the same given in the tasks as using the release packages gave a lot of problems with wrong versions, sometimes we would get version 4 or 5, and sometimes a 6.2 beta version, whilst all we needed was the 6.0.x versions as these were the ones that the docker compose stack received. We got our downloads from this zabbix repo:

```
apt-get install wget

wget https://repo.zabbix.com/zabbix/6.0/ubuntu/pool/main/z/zabbix-release/zabbix-release_6.0-1%2Bubuntu20.04_all.deb

dpkg -i zabbix-release_6.0-1+ubuntu20.04_all.deb

# needed this as well since we got the wrong version of the proxy by just having the release package
wget https://repo.zabbix.com/zabbix/6.0/ubuntu/pool/main/z/zabbix/zabbix-proxy-mysql_6.0.1-1%2Bubuntu20.04_amd64.deb

dpkg -i zabbix-proxy-mysql_6.0.1-1+ubuntu20.04_amd64.deb

apt-get install -f

apt-get install zabbix-proxy-mysql

apt-get install zabbix-sql-scripts
```

The only notable difference between the guide and what we did in this part of the task, is that we used Zabbix-proxy version 6.0.1 instead of Zabbix-proxy version 6.0, because the assignment descriptions asks us to use version 6.1.1

#### 3.1.2. Configuring MariaDB database for the proxy to use

After we finished installing Zabbix Proxy, we installed and configured the database, using MariaDB, according to both the guide mentioned above, and the assignment description.

The following block of code describes the installation of MariaDB on VM2:

```
sudo apt install software-properties-common -y

curl -LsS -O https://downloads.mariadb.com/MariaDB/mariadb_repo_setup

sudo bash mariadb_repo_setup --mariadb-server-version=10.6

sudo apt update

sudo apt -y install mariadb-common mariadb-server-10.6 mariadb-client-10.6
```

After installing MariaDB we began configuring the database by running the following commands to start and enable MariaDB, and configure it to start on boot:

```
sudo systemctl start mariadb

sudo systemctl enable mariadb
```

The next step in the configuration of the database was to reset the password. We did that with the commands in the following code block: The following code blocks set the password for new **root password** = 123.

```
sudo mysql_secure_installation

Enter current password for root (enter for none): Press Enter

Switch to unix_socket authentication [Y/n] y

Change the root password? [Y/n] y

New password: <Enter root DB password>

Re-enter new password: <Repeat root DB password>

Remove anonymous users? [Y/n] y

Disallow root login remotely? [Y/n] y

Remove test database and access to it? [Y/n] y

Reload privilege tables now? [Y/n] y
```

The only notable difference between the guide and what we did was that we set the root password to '123', in stead of 'rootDBpass'.

After we set the root password, it was time to create the database by running the commands in the following block of code:

```
sudo mysql -uroot -p'123' -e "create database zabbix_proxy character set utf8mb4 collate utf8mb4_bin;"

sudo mysql -uroot -p'123' -e "grant all privileges on *.* to zabbix@localhost identified by 'zabbixDBpass';"
```

The last step in the configuration of the database was to import the initial schema and data with the following command:

```
sudo cat /usr/share/doc/zabbix-sql-scripts/mysql/proxy.sql | mysql -uzabbix -p'zabbixDBpass' zabbix_proxy
```

In the installation and configuration of the database, we followed the guide quite exactly. Therefore, there are very few differences between what we did to install and configure the database, and what is stated in the installation guide.

### 3.1.3. Configuring Zabbix Proxy

Once the installation and configuration of the database was complete, it was time to configure the Zabbix Proxy. The first step when configuring the Zabbix Proxy was to open the config file with the following command:

```
sudo gedit /etc/zabbix/zabbix_proxy.conf
```

In the file we changed the following values:

```
DBPassword=zabbixDBpass
ConfigFrequency=100
Server=192.168.50.95
Hostname=Zabbix Proxy
DBName=zabbix_proxy
DBUser=zabbix
```

After editing the necessary values, we saved and exited the file. We configured the ConfigFrequency to be 100 seconds. This parameter determines how often the proxy retrieves data from the configuration file, and is useful to cut down on the waiting time between updates on the status of the Zabbix Proxy. The notable differences between



our config file, and the config file in the guide is that we have a different IP address for the server, and a different hostname for the proxy itself.

Name ▲	Mode	Encryption	Compression	Last seen (age)	Host count	Item count
Zabbix Proxy	Active	None PSK	On	3s	1	73

zabbix-agent

Items 66

Triggers 34

Graphs 13

Discovery 3

Web

Linux by Zabbix agent active

Enabled

None

zabbix\_server

Items 67

Triggers 34

Graphs 13

Discovery 3

Web

172.200.1.4:10050

Linux by Zabbix agent

Enabled

ZBX

None

### 3.1.4. Starting and enabling the Zabbix Proxy

Next, we started and enabled the Zabbix Proxy to boot on startup with the following commands:

```
#makes the proxy start by default
sudo systemctl enable zabbix-proxy

# not really needed but i like to do this just in case
sudo systemctl start zabbix-proxy
```

It is important to take note of the Hostname in the zabbix-proxy config file. We need it in order to connect the proxy to the server in the web frontend. While in the zabbix-proxy config file, it is also important to make sure that the DBPassword is set to the correct value.

### 3.1.5. Registering Zabbix Proxy in the Zabbix frontend

Proxy

Encryption

\* Proxy name

Zabbix Proxy

Proxy mode

Active

Passive

Proxy address

Description

Update

Clone

Delete

Cancel

Then we set up a new host to make the active changes work, configured like this:

Proof that it is working:

Name	Items	Triggers	Graphs	Discovery	Web	Interface	Proxy	Templates	Status	Availability	Agent encryption	Info
Zabbix server	Items 103	Triggers 57	Graphs 19	Discovery 4	Web	127.0.0.1:10050		Linux by Zabbix agent, Zabbix server health	Enabled	20x	None	
zabbix_server	Items 42	Triggers 14	Graphs 8	Discovery 3	Web	172.200.1.4:10050		Linux by Zabbix agent	Enabled	20x	None	

### 3.2. VM 3

The following code block must be run as root on VM3 to install the Zabbix-agent.

```
wget https://repo.zabbix.com/zabbix/6.1/ubuntu/pool/main/z/zabbix-release/zabbix-release_6.1-1%2Bubuntu20.04_all.deb
```

*#had to use this one for right version*

```
wget https://repo.zabbix.com/zabbix/6.0/ubuntu/pool/main/z/zabbix/zabbix-agent_6.0.1-1%2Bubuntu20.04_amd64.deb
```

```
sudo dpkg -i zabbix-release_6.1-1ubuntu20.04_all.deb
```

```
sudo apt-get install -f
```

```
sudo apt-get install zabbix-agent
```

The following code block creates the psk encryption key.

```
openssl rand -hex 32 > zabbix_agent.psk
```

```
cat zabbix_agent.psk
```

```
e8126679667a8594bc8d3d76121b6ba2a5fb4b6d41bea2cd62190c163fbc6c6b
```

The following code block moves the psk encryption key to /opt/zabbix folder.

```
sudo mkdir /opt/zabbix  
  
sudo chmod 777 /opt/zabbix  
  
sudo mv zabbix_agent.psk /opt/zabbix/
```

used these commands to edit the `zabbix-agent.conf` file

```
sudo vim /etc/zabbix/zabbix_agentd.conf
```

The following code block contains the lines we changed in the `zabbix-agent.conf` file; to enable psk encryption.

```
TLSConnect=psk  
TLSAccept=psk  
TLSPSKIdentity=cbt_psk_01,  
TLSPSKFile=/opt/zabbix_agent.psk  
# the local ip of our bridged networked VM2  
Server=192.168.50.247
```

Since we are running this in a docker container and not on an actual VM, we don't have `systemd` available. Therefore, we cannot *enable* the service, only start it, and have it as a run command in a dockerfile. This means we need to make sure that the service is started every time we run the container the dockerfile is made for.

## 4. VM2: Nginx proxy

We start by creating the configuration file for the nginx-proxy. This file makes sure that the nginx-proxy listens on port 8080, and redirects all incoming traffic from that port to the Zabbix-server using `proxy_pass` followed by the Zabbix-server's IP address and port number.

```
# file: /etc/nginx/sites-enabled/zabbix.conf

server {
    listen 8080;
    server_name localhost;

    location / {
        proxy_pass http://192.168.50.95:8080;
        proxy_set_header Host $http_host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

After setting up the configuration file for the Nginx-proxy, we start up a terminal on VM2 and install nginx with the following commands.

```
sudo apt-get update

sudo apt-get install nginx
```

Once nginx has been installed, we disable the default virtual host by unlinking it with the following command.

```
sudo unlink /etc/nginx/sites-enabled/default
```

Then we add `reverse-proxy.conf` to the directory `/etc/nginx/sites-available/` with the following commands.

```
sudo nano /etc/nginx/sites-available/reverse-proxy.conf
```

To complete the proxy, we activate the directives by linking to `/sites-enabled/` using the following command.

```
sudo ln -s /etc/nginx/sites-available/reverse-proxy.conf /etc/nginx/sites-enabled/reverse-proxy.conf
```

Lastly, to see if it works, we run an nginx configuration test and restart the service.

```
sudo service nginx configtest
* Testing nginx configuration [ OK ]

sudo service nginx restart
* Restarting nginx nginx [ OK ]
```

This verifies that nginx works as intended.

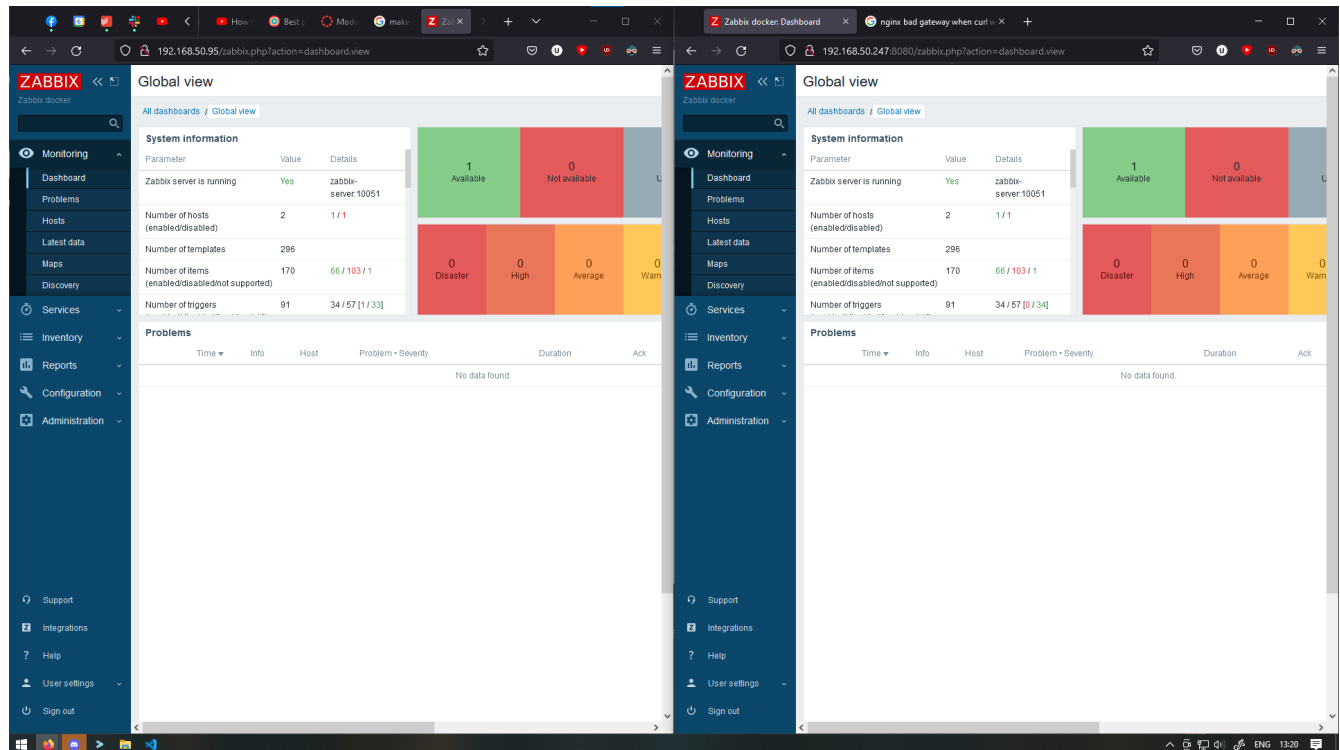
To reach other docker containers, we have to use the IP address of the host machine (intel1 in this case) 172.19.0.1

```
curl 172.19.0.1
<!DOCTYPE html>
<html>
----- # commented put contents of the zabbix page
```

## 5. VM1: Zabbix frontend

To access the Zabbix frontend we connect to the nginx-proxy on VM2 via it's local IP and and port 8080 as specified. This redirected us to the VM1 zabbix-web docker container. Here we added the host according to the descriptions, and then made the items as per point a) and b), and the triggers as per point c) and d)

Name	Items	Triggers	Graphs	Discovery	Web	Interface	Proxy	Templates	Status	Availability	Agent encryption
zabbix-agent	Items 66	Triggers 34	Graphs 13	Discovery 3	Web			Linux by Zabbix agent active	Enabled		None
zabbix_server	Items 67	Triggers 34	Graphs 13	Discovery 3	Web	172.200.1.4:10050		Linux by Zabbix agent	Enabled	ZBX	None
zabbix_server_agent_vm3	Items 73	Triggers 29	Graphs 14	Discovery 3	Web	192.168.50.151:10050	Zabbix Proxy	Linux by Zabbix agent	Enabled	ZBX	PSK None PSK CERT



We created a new template zabbix-monitoring in the host group zabbix-monitoring with the following items and triggers:

Templates

Tags

Macros

Value mapping

\* Template name

zabbix-monitoring

Visible name

zabbix-monitoring

Templates

type here to search

Select

\* Groups

zabbix-monitoring

type here to search

Select

Description

Add

Cancel

## 5.1. Items

We created the item for total disk space usage in the directory `/var` with interval of 1 hour:

```
fsv.fs[/var,used]
```

We created an that will monitor the docker process usage with interval of 1 minute:

```
proc.cpu.util[dockerd]
```

<input type="checkbox"/>	Name ▲	Triggers	Key	Interval	History
<input type="checkbox"/>	... Docker process usage		proc.cpu.util[dockerd]	1m	90d
<input type="checkbox"/>	... Total Used Disk Space		vfs.fs.size[/var,used]	60m	90d

## 5.2. Triggers

We created a trigger that triggers when uptime is longer than 240 days:

```
last(/zabbix_server_agent_vm3/system.uptime)>240d
```

All hosts / zabbix\_server\_agent\_vm3 Enabled ZBX Items 73 Triggers 28 Graphs 14 Discovery rules 3 Web scenarios

Trigger Tags Dependencies

\* Name uptime over 240 days

Event name uptime over 240 days

Operational data

Severity Not classified Information Warning Average High Disaster

\* Expression last(/zabbix\_server\_agent\_vm3/system.uptime)>240d Add

[Expression constructor](#)

OK event generation Expression Recovery expression None

PROBLEM event generation mode Single Multiple

OK event closes All problems All problems if tag values match

We created a trigger that triggers when disk I/O is higher than 20% average for 5 minutes:

```
avg(/zabbix_server_agent_vm3/system.cpu.util[,iowait],5m)>20
```

## Triggers

All hosts / zabbix\_server\_agent\_vm3 Enabled ZBX Items 73 Triggers 29 Graphs 14 Discovery rules 3 Web scenarios

Trigger Tags Dependencies

\* Name Disk i/o is overloaded

Event name Disk i/o is overloaded

Operational data

Severity Not classified Information Warning Average High Disaster

\* Expression avg(/zabbix\_server\_agent\_vm3/system.cpu.util[,iowait],5m)>20 Add

[Expression constructor](#)

OK event generation Expression Recovery expression None

PROBLEM event generation mode Single Multiple

OK event closes All problems All problems if tag values match

After making those triggers we made sure that the triggers were correctly created:

<input type="checkbox"/>	Information	OK	Linux by Zabbix agent: System name has changed		<code>last(/zabbix_server_agent_vm3/system.hostname,#1)&lt;=&gt;last(/zabbix_server_agent_vm3/system_server_agent_vm3/system.hostname))&gt;0</code>
<input type="checkbox"/>	Warning	OK	Linux by Zabbix agent: System time is out of sync		<code>fuzzytime(/zabbix_server_agent_vm3/system.localtime,{\$SYSTEM.FUZZYTIME.MAX})=0</code>
<input type="checkbox"/>	Not classified	OK	uptime over 240 days		<code>last(/zabbix_server_agent_vm3/system.uptime)&gt;240d</code>
<input type="checkbox"/>	Average	OK	Linux by Zabbix agent: Zabbix agent is not available		<code>max(/zabbix_server_agent_vm3/zabbix[host.agent.available],{\$AGENT.TIMEOUT})=0</code>
			zabbix_server_agent_vm3: Getting closer to process limit		
<input type="checkbox"/>	Not classified	OK	Disk i/o is overloaded		<code>avg(/zabbix_server_agent_vm3/system.cpu.util[,iowait],5m)&gt;20</code>
<input type="checkbox"/>	Warning	OK	Linux by Zabbix agent: Getting closer to process limit	{ITEM.LASTVALUE1} active, {ITEM.LASTVALUE2} limit	<code>last(/zabbix_server_agent_vm3/proc.num)/last(/zabbix_server_agent_vm3/kern</code>
<input type="checkbox"/>	Warning	OK	Linux by Zabbix agent: has been restarted		<code>last(/zabbix_server_agent_vm3/system.uptime)&lt;10m</code>
<input type="checkbox"/>	Warning	OK	Linux by Zabbix agent: High CPU utilization Depends on: zabbix_server_agent_vm3: Load average is too high	Current utilization: {ITEM.LASTVALUE1}	<code>min(/zabbix_server_agent_vm3/system.cpu.util,5m)&gt;{\$CPU.UTIL.CRIT}</code>

## 6. References

TODO: actually add references