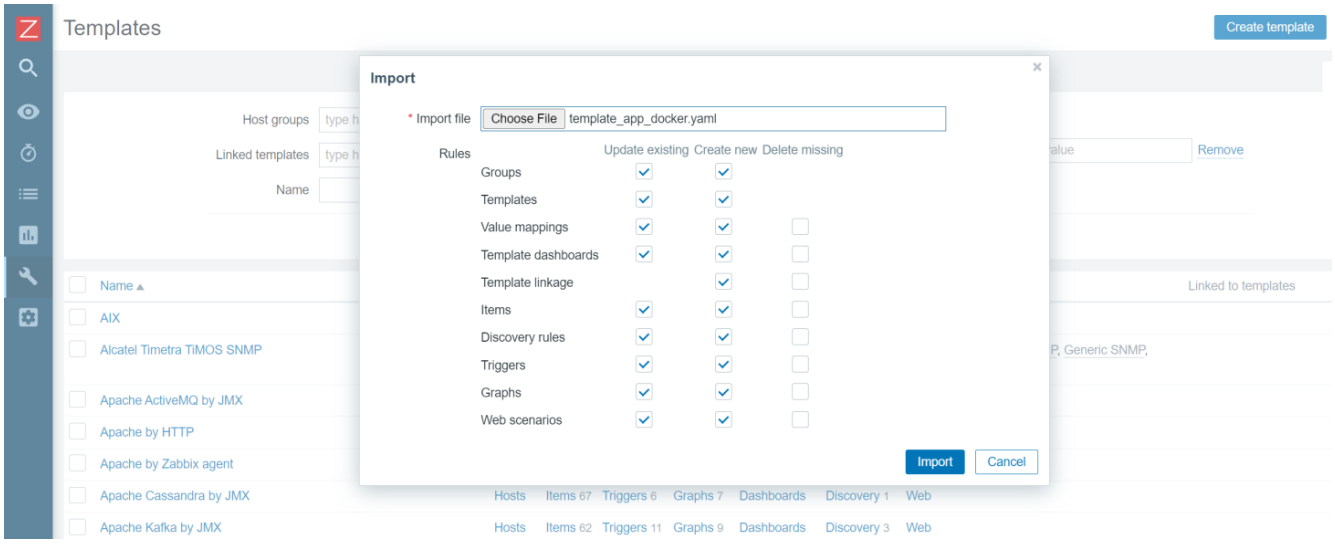
Docker Container Monitoring With Zabbix

> Credit **Dmitry Lambert** Original URL: https://blog.zabbix.com/docker-container-monitoring-with-zabbix/20175/

Importing the official Docker template



Importing the Docker by Zabbix agent 2 template

Since we will be using the official *Docker by Zabbix agent 2* template, first, we need to make sure that the template is actually available in our Zabbix instance. The template is available for **Zabbix versions 5.0, 5.4, and 6.0**. If you cannot find this template under *Configuration – Templates*, chances are that you haven't imported it into your environment after upgrading Zabbix to one of the aforementioned versions. Remember that Zabbix does not modify or import any templates during the upgrade process, so we will have to import the template manually. If that is so, simply download the template and import it into your Zabbix instance by using the Import button in the *Configuration – Templates* section.

Installing and configuring Zabbix agent 2

Before we get started with configuring our host, we first have to install Zabbix agent 2 and configure it according to the template guidelines. Follow the steps in the download section of the Zabbix website and install the zabbix-agent2 package. Feel free to use any other agent deployment methods if you want to (like compiling the agent from the source files)

Installing Zabbix agent2 from packages takes just a few simple steps:

Install the Zabbix repository package:

```
rpm -Uvh https://repo.zabbix.com/zabbix/6.0/rhel/8/x86_64/zabbix-release-6.0-1.el8.noarch.rpm
```

Install the Zabbix agent 2 package:

```
dnf install zabbix-agent2
```

Configure the *Server* parameter by populating it with your Zabbix server/proxy address

```
vi /etc/zabbix/zabbix_agent2.conf
```

```
### 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=192.168.50.49
```

Plugin specific Zabbix agent 2 configuration

Zabbix agent 2 provides plugin-specific configuration parameters. Mostly these are optional parameters related to a specific plugin. You can find the full list of plugin-specific configuration parameters in the Zabbix documentation. In the newer versions of Zabbix agent 2, the plugin-specific parameters are defined in separate plugin configuration files, located in */etc/zabbix/zabbix_agent2.d/plugins.d/*, while in older versions, they are defined directly in the *zabbix_agent2.conf* file.

For the Zabbix agent 2 Docker plugin, we have to provide the Docker daemon unix-socket location. This can be done by specifying the following plugin parameter:

```
### Option: Plugins.Docker.Endpoint
# Docker API endpoint.
#
# Mandatory: no
# Default: unix:///var/run/docker.sock
# Plugins.Docker.Endpoint=unix:///var/run/docker.sock
```

The default socket location will be correct for your Docker environment – in that case, you can leave the configuration file as-is.

Once we have made the necessary changes in the Zabbix agent 2 configuration files, start and enable the agent:

```
systemctl enable zabbix-agent2 --now
```

Check if the Zabbix agent2 is running:

```
tail -f /var/log/zabbix/zabbix_agent2.log
```

Before we move on to Zabbix frontend, I would like to point your attention to the Docker socket file permission – the zabbix user needs to have access to the Docker socket file. The zabbix user should be added to the docker group to resolve the following error messages.

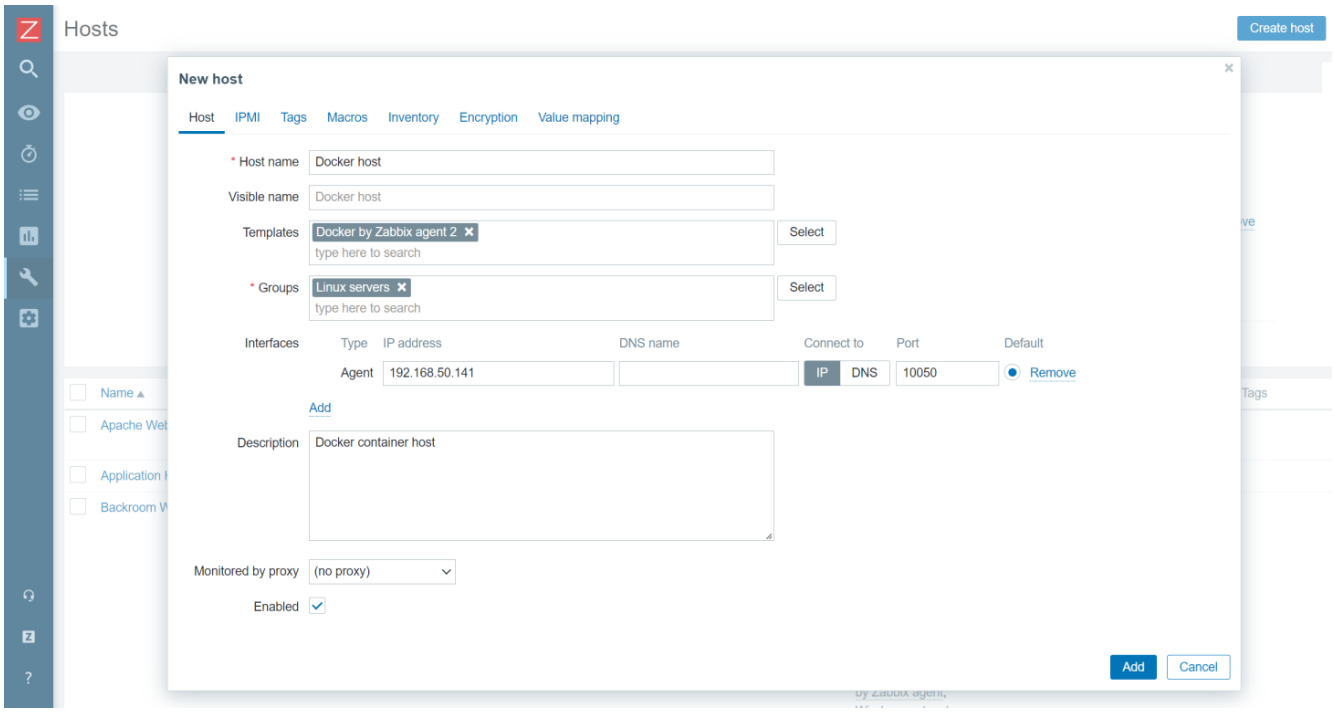
```
[Docker] cannot fetch data: Get http://1.28/info: dial unix /var/run/docker.sock: connect:
permission denied
```

```
ZBX_NOTSUPPORTED: Cannot fetch data.
```

You can add the zabbix user to the Docker group by executing the following command:

```
usermod -aG docker zabbix
```

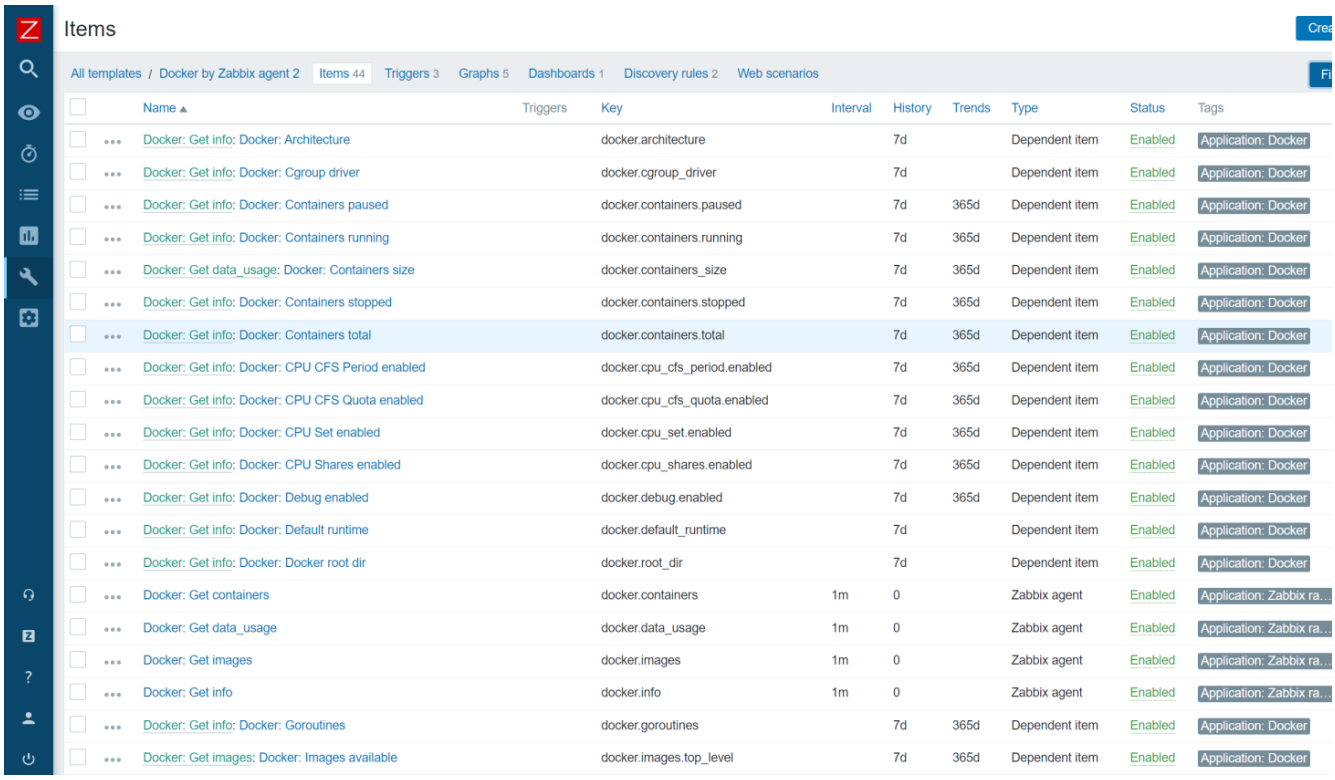
Configuring the docker host



Configuring the host representing our Docker environment

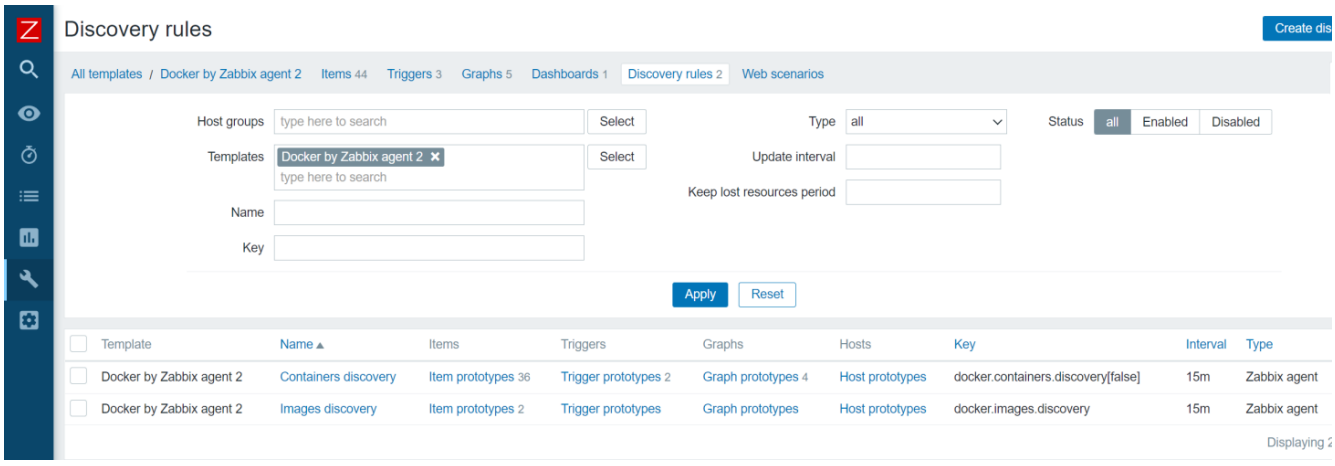
After importing the template, we have to create a host which will represent our Docker instance. Give the host a name and assign it to a Host group – I will assign it to the Linux servers host group. Assign the *Docker by Zabbix agent 2* template to the host. Since the template uses Zabbix agent 2 to collect the metrics, we also have to add an agent interface on this host. The address of the interface should point to the machine running your Docker containers. Finish up the host configuration by clicking the Add button.

Docker by Zabbix agent 2 template



Regular docker template items

The template contains a set of regular items for the general Docker instance metrics, such as the number of available images, Docker architecture information, the total number of containers, and more.

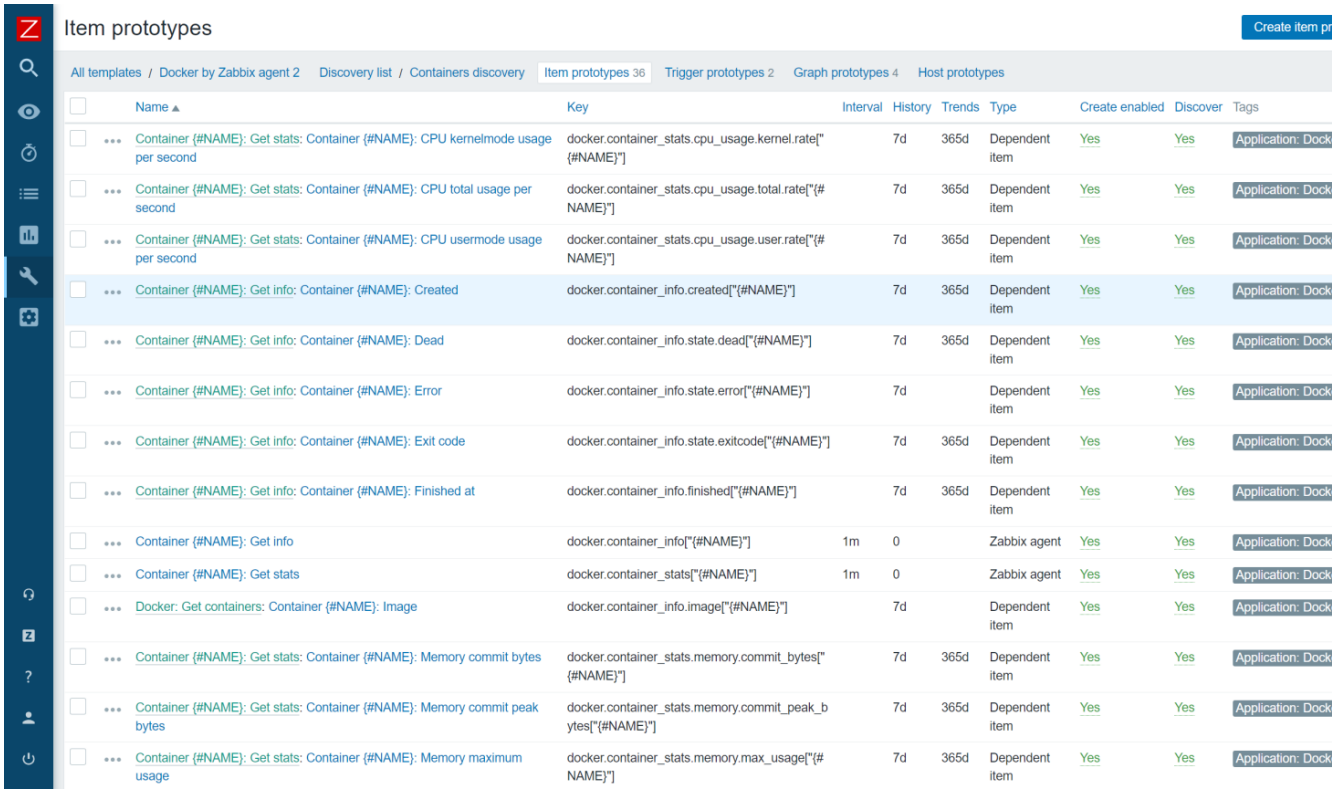


Docker template Low-level discovery rules

On top of that, the template also gathers container and image-specific information by using low-level discovery rules.

Once Zabbix discovers your containers and images, these low-level discovery rules will then be used to create items, triggers, and graphs from prototypes for each of your containers and images. This way, we can monitor container or image-specific metrics, such as container memory, network

information, container status, and more.



Docker templates Low-level discovery item prototypes

Verifying the host and template configuration

To verify that the agent and the host are configured correctly, we can use Zabbix get command-line tool and try to poll our agent. If you haven't installed Zabbix get, do so on your Zabbix server or Zabbix proxy host:

```
dnf install zabbix-get
```

Now we can use *zabbix-get* to verify that our agent can obtain the Docker-related metrics. Execute the following command:

```
zabbix_get -s docker-host -k docker.info
```

Use the *-s* parameter to specify your agent host's host name or IP address. The *-k* parameter specifies the item key for which we wish to obtain the metrics by polling the agent with Zabbix get.

```
zabbix_get -s 192.168.50.141 -k docker.info

{"Id":"SJYT:SATE:7XZE:7GEC:XFUD:KZO5:NYFI:L7M5:4RGO:P2KX:QJFD:TAVY","Containers":2,"Containers
Running":2,"ContainersPaused":0,"ContainersStopped":0,"Images":2,"Driver":"overlay2","MemoryLi
mit":true,"SwapLimit":true,"KernelMemory":true,"KernelMemoryTCP":true,"CpuCfsPeriod":true,"Cpu
CfsQuota":true,"CPUShares":true,"CPUSet":true,"PidsLimit":true,"IPv4Forwarding":true,"BridgeNf
Iptables":true,"BridgeNfIP6tables":true,"Debug":false,"NFd":39,"OomKillDisable":true,"NGorouti
nes":43,"LoggingDriver":"json-
file","CgroupDriver":"cgroupfs","NEventsListener":0,"KernelVersion":"5.4.17-
2136.300.7.el8uek.x86_64","OperatingSystem":"Oracle Linux Server
8.5","OSVersion":"8.5","OSType":"linux","Architecture":"x86_64","IndexServerAddress":"https://
index.docker.io/v1/","NCPU":1,"MemTotal":1776848896,"DockerRootDir":"/var/lib/docker","Name":"
localhost.localdomain","ExperimentalBuild":false,"ServerVersion":"20.10.14","ClusterStore":"",
"ClusterAdvertise":"","DefaultRuntime":"runc","LiveRestoreEnabled":false,"InitBinary":"docker-
init","SecurityOptions":["name=seccomp,profile=default"],"Warnings":null}
```

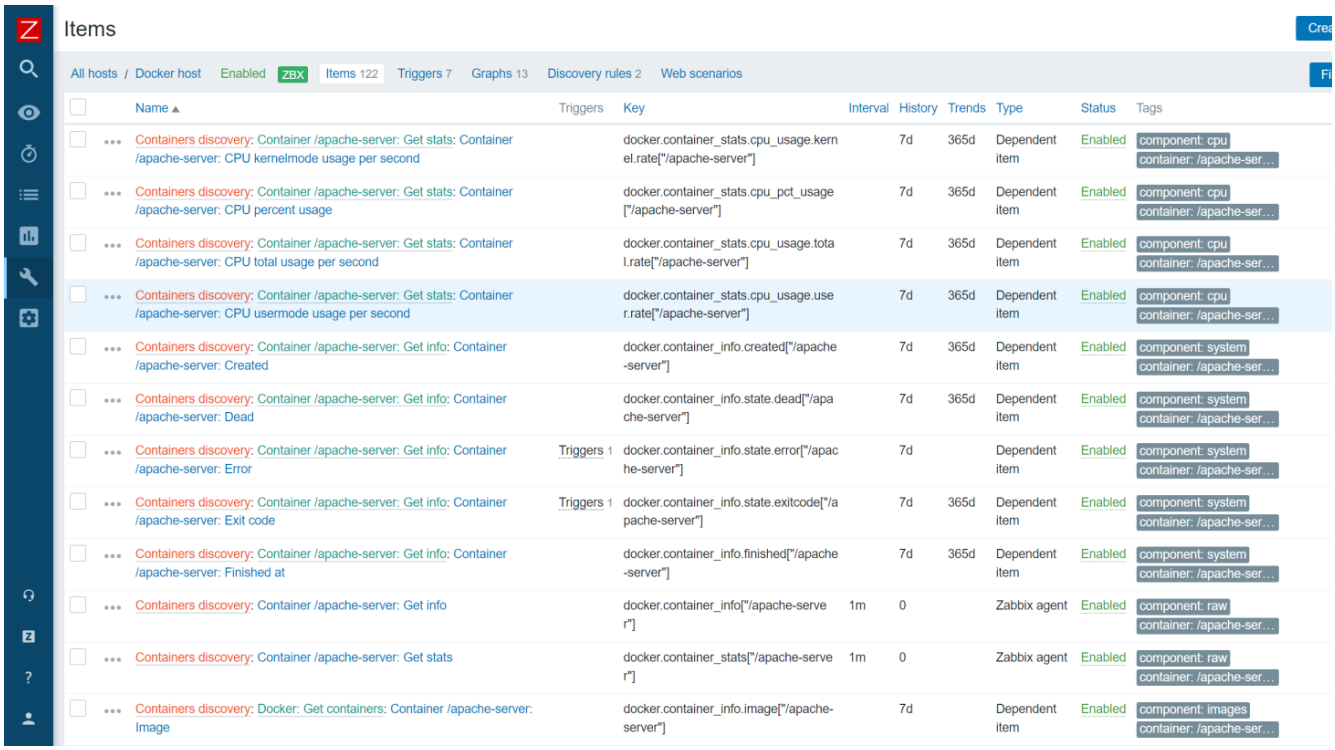
In addition, we can also use the low-level discovery key – *docker.containers.discovery[false]* to check the result of the low-level discovery.

```
zabbix_get -s 192.168.50.141 -k docker.containers.discovery[false]

[{"{#ID}":"a1ad32f5ee680937806bba62a1aa37909a8a6663d8d3268db01edb1ac66a49e2","{#NAME}":"/apach
e-
server"},{"{#ID}":"120d59f3c8b416aaeeba50378dee7ae1eb89cb7ffc6cc75afdfedb9bc8cae12e","{#NAME}"
:"/mysql-server"}]
```

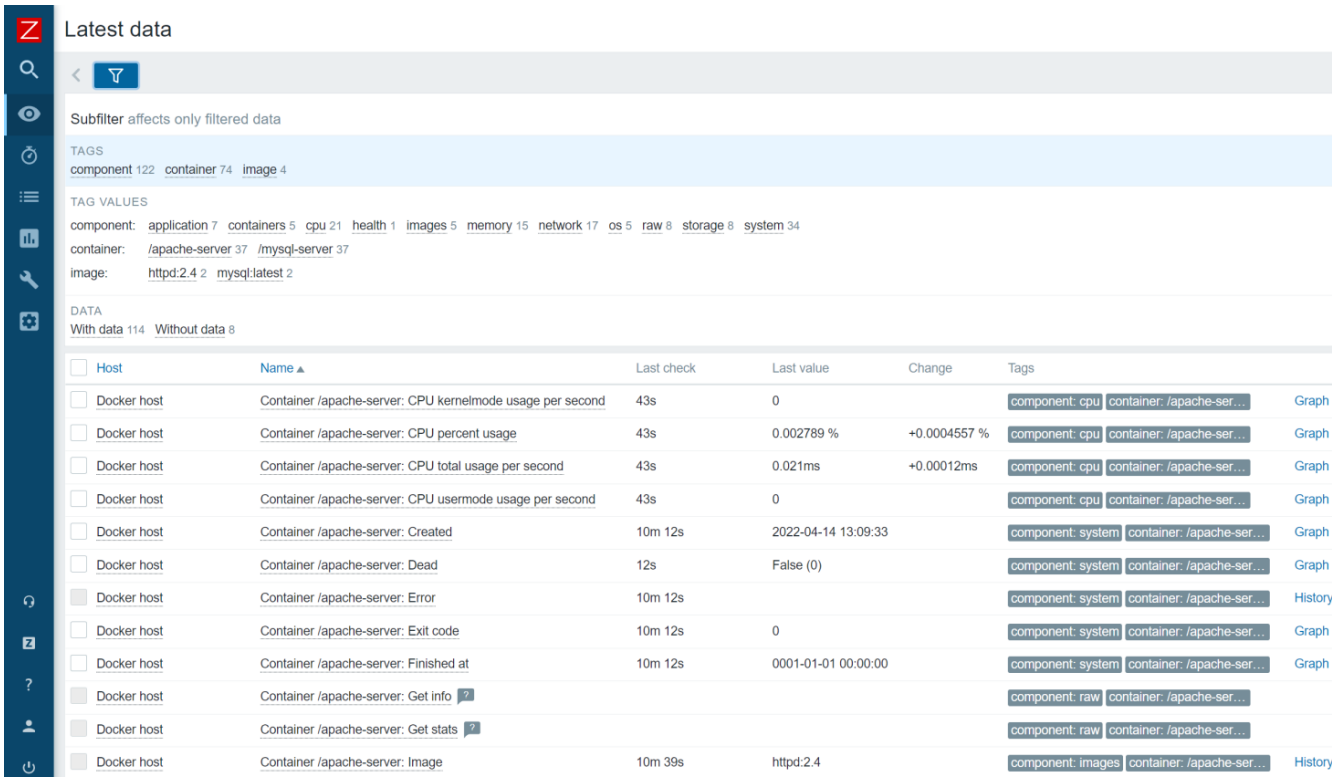
We can see that Zabbix will discover and start monitoring two containers – *apache-server* and *mysql-server.* Any agent low-level discovery rule or item can be checked with Zabbix get.

Docker template in action



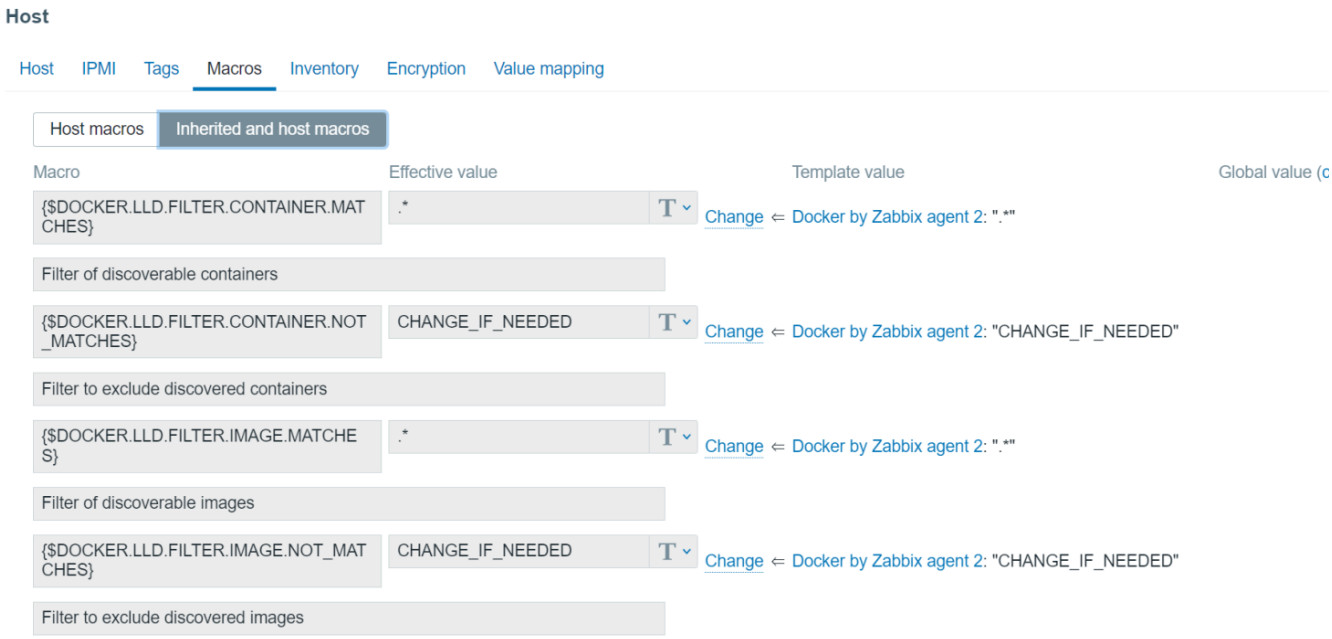
Discovered items on our Docker host

Now that we have configured our agent and host, applied the Docker template, and verified that everything is working, we should be able to see the discovered entities in the frontend.



Collected Docker container metrics

In addition, our metrics should have also started coming in. We can check the *Latest data* section and verify that they are indeed getting collected.



Macros inherited from the Docker template

Lastly, we have a few additional options for further modifying the template and the results of our low-level discovery. If you open the *Macros* section of your host and select *Inherited and host macros*, you will notice that there are 4 macros inherited from the Docker template. These macros are responsible for filtering in/out the discovered containers and images. Feel free to modify these values if you wish to filter in/out the discovery of these entities as per your requirements.

Notice that the container discovery item also has one parameter, which is defined as *false* on the template:

- *docker.containers.discovery[false]* – Discover only running containers
- *docker.containers.discovery[true]* – Discover all containers, no matter their state.

Revision #3
Created 13 June 2022 16:43:30 by Dino Edwards
Updated 18 December 2022 11:45:48 by Dino Edwards