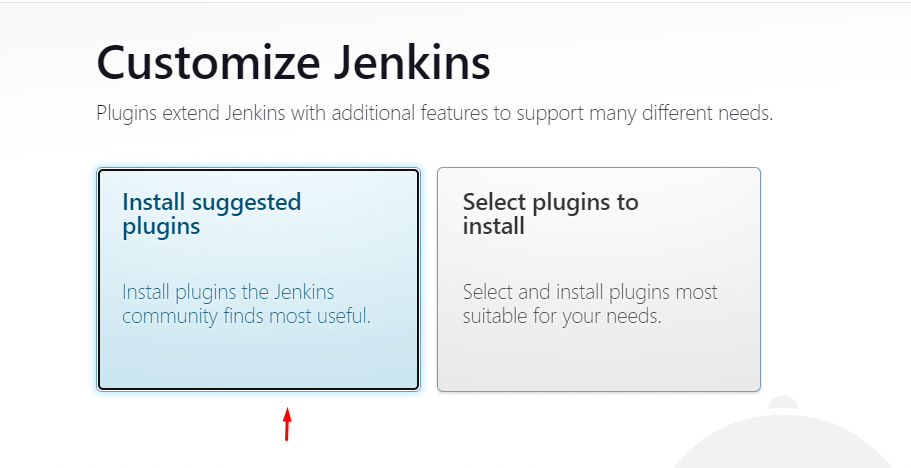
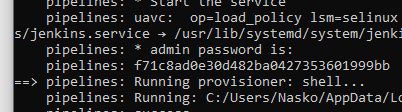
DevOps Exam

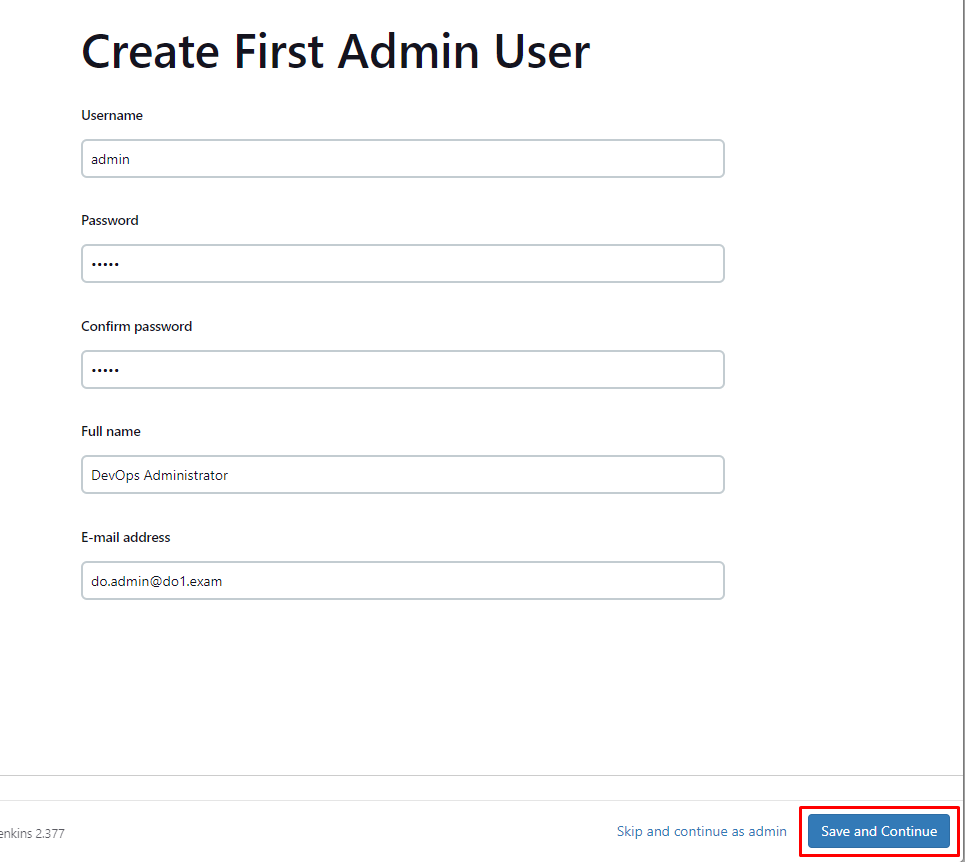
After we’ve bring the environment up with the vagrant up command, we’ll need to Unlock Jenkins with the initial admin password.

We should go to <http://192.168.150.201:8080/> and type the initial admin password, which we can get from the console:

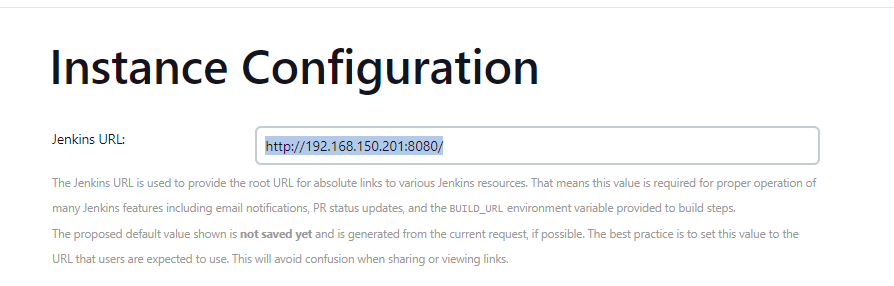


Afterwards we need to install all suggested plugins.

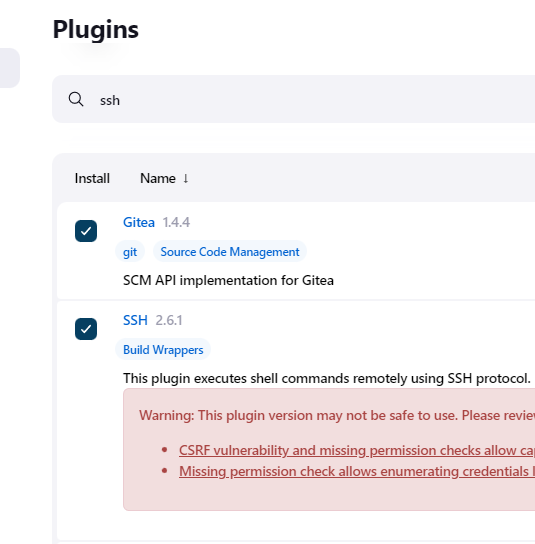
After the plugins are installed, the First admin user has been created:



The instance configuration should be configured to the following URL: <http://192.168.150.201:8080/>

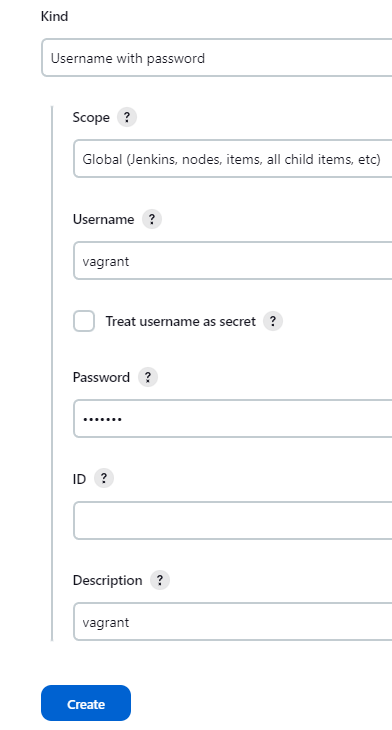


We should go to Manage Jenkins > Manage Plugins > Available plugins in order to have some additional plugins installed. I will install Gitea and SSH.

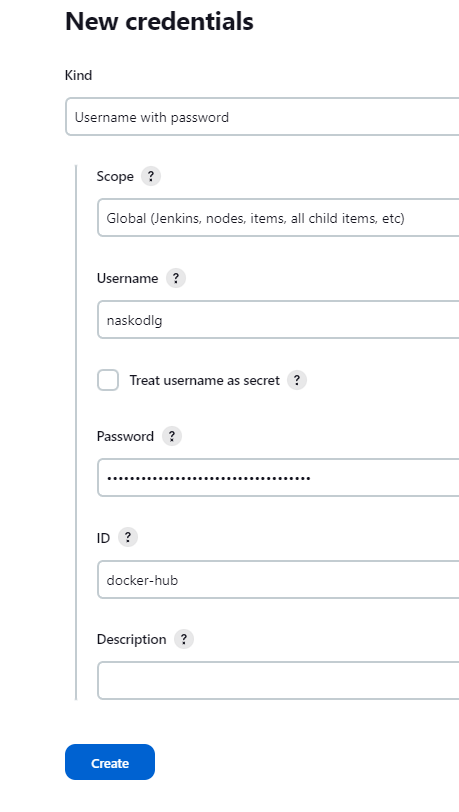


In order to download and install them, I’ll select Download now and install after restart.

After the restart, we need to login with the admin user and create the credentials.

Go to Manage Jenkins > Manage Credentials > System > Global credentials and select Add credentials. Here I’ll add the vagrant user => 

After that I’ve added the Dockerhub credentials:



We should add the SSH Remote host.

For the hostname should be containers.do1.exam, the Port should be 22 and we should use the Vagrant credentials.

The connection is tested and it’s successful.



Now we should add slave (agent) node on the Docker host. Let’s go to Manage Jenkins > Manage Nodes and Clouds and select New Node.

I’ll name the node as docker-node and its type will be Permanent Agent. It will have 2 numbers of executors and its remote root directory will be /home/vagrant. Its usage will be set to “Only build jobs with label expressions matching this node” . The launch method is set to Launch agents via SSH, the host is containers.do1.exam with the vagrant credentials.

There’s an error with the node launching – its not in the known hosts file on the pipelines machine. Let’s execute the following commands on the pipelines:

sudo passwd jenkins

Set the password to Jenkins

sudo vi /etc/passwd

We should set

jenkins:x:986:986:Jenkins Automation Server:/var/lib/jenkins:/bin/false

to

jenkins:x:986:986:Jenkins Automation Server:/var/lib/jenkins:/bin/bash

and save the file with :wq

Execute the following afterwards:

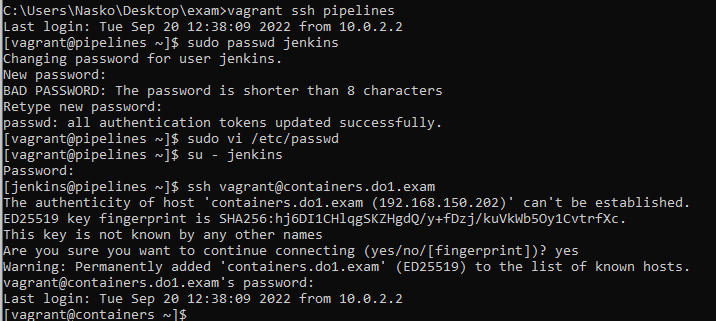
su – jenkins

We will be required to login with the jenkins password

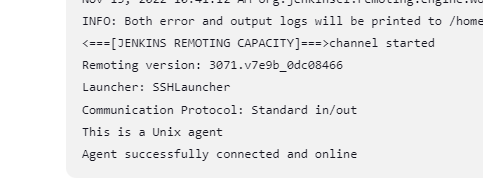
After we login, we should execute

ssh [vagrant@containers.do1.exam](mailto:vagrant@containers.do1.exam)

We should input yes when we’re asked if we want to add the hosts to the known hosts file.



Now after I’ve relaunched the agent, its up and running:



We should execute the following commands in order for vagrant to have the ownership of .vagrant on the containers machine.

sudo chown vagrant:vagrant /home/vagrant/.docker -R

sudo chmod g+rwx "/home/vagrant/.docker" -R

Otherwise, the logging and publishing to dockerhub won’t work.

I’ve created the Pipeline – it can be found in the Jenkinsfile.

Now we should create the Jenkins job. Go to the dashboard, select New item, name the Item – I’ve named it as Pipeline-1 and select Pipeline type.

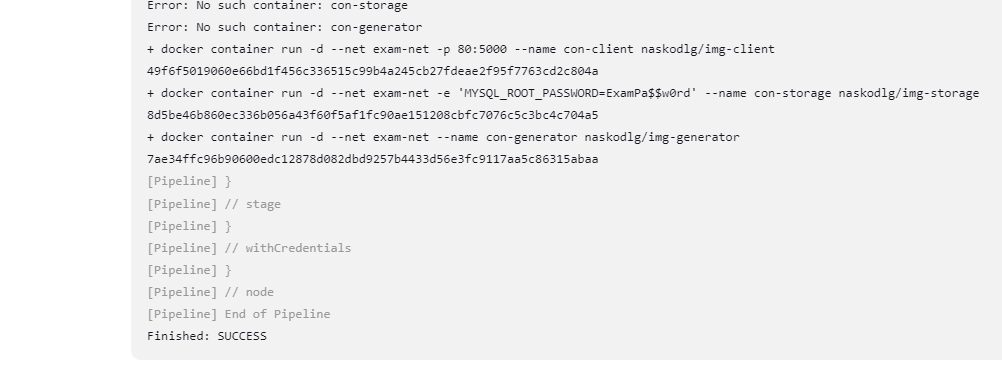
We should select GitHub Project and set the Project url to the following: <http://192.168.150.202:3000/vagrant/exam.git/>

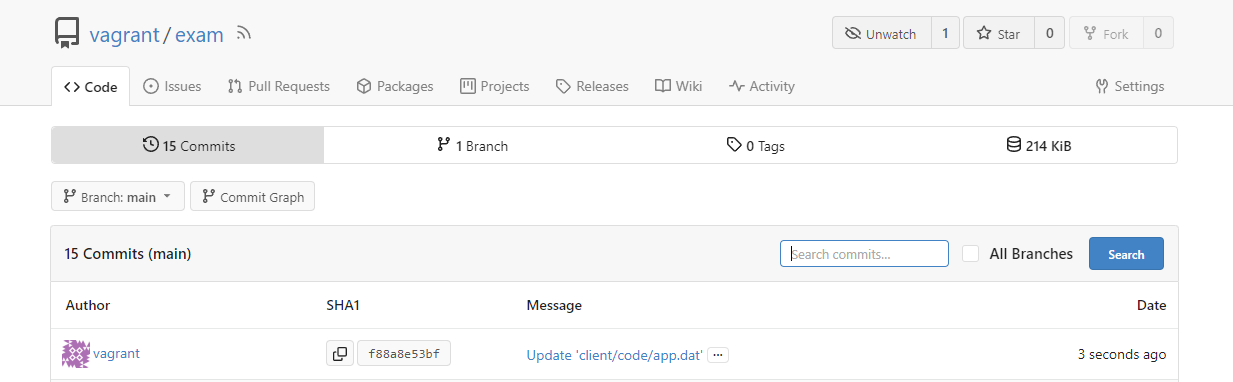
At the Build triggers we should select the GitHub hook trigger for GITScm polling and Poll SCM.

On the Pipeline, the definition should be set to Pipeline script from SCM, SCM should be set to Git. The repository URL should be <http://192.168.150.202:3000/vagrant/exam.git> and we should use the vagrant credentials to login.

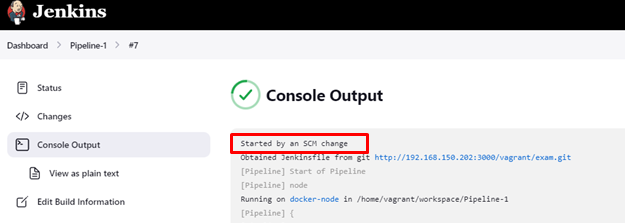
I’ve uploaded the Jenkisnfile to the Gitea which can be found running up already by the automation on <http://192.168.150.202:3000/> , logged with the vagrant user, whos password is vagrant. The Repository has been already cloned with the setup-gitea.sh, the user creation is also happening automatically there. The Branch Specifier should be set to \*/main  
Script path is Jenkinsfile.

The pipeline has been run manually and executed sucessfuly:





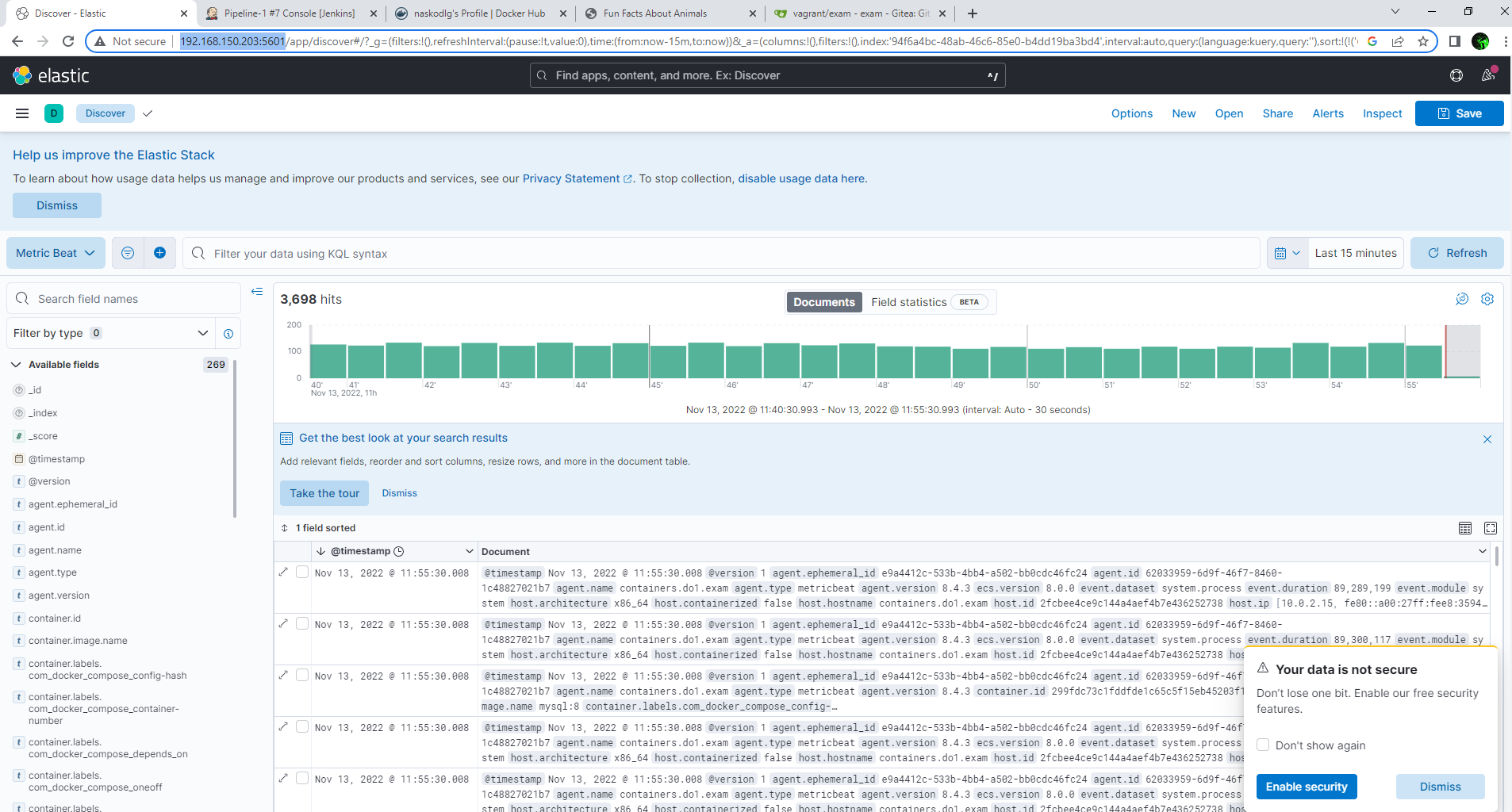
client/code/app.dat has been changed – and a Jenkins job has been started automatically – the web hook works:



For the monitoring I am using the ELK stack. It has been installed automatically via the add-elk.sh file.

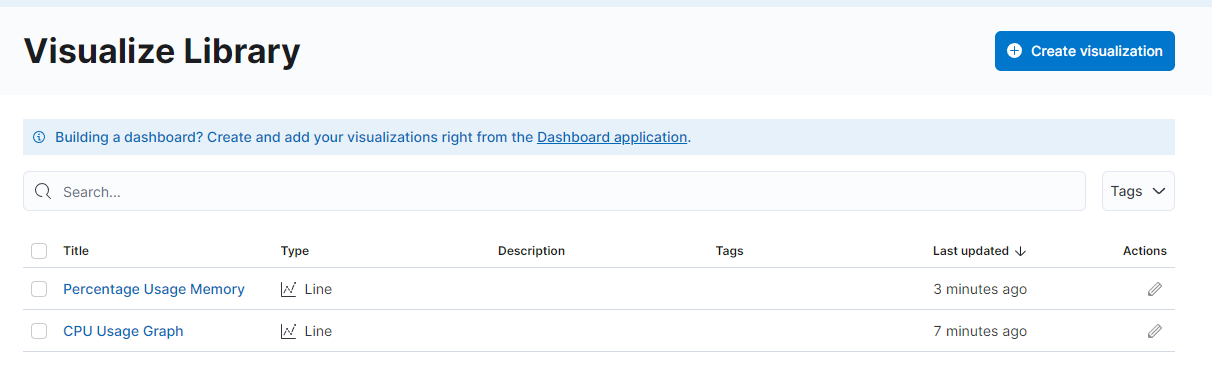
The installation of all components for ELK are working and there are deployed monitoring agents on the pipelines and containers VMs.

Kibana is working on <http://192.168.150.203:5601/> :

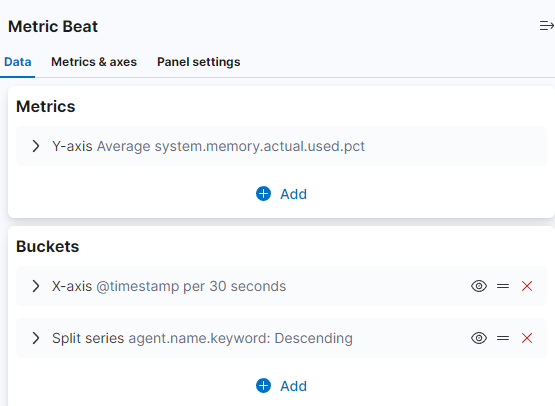


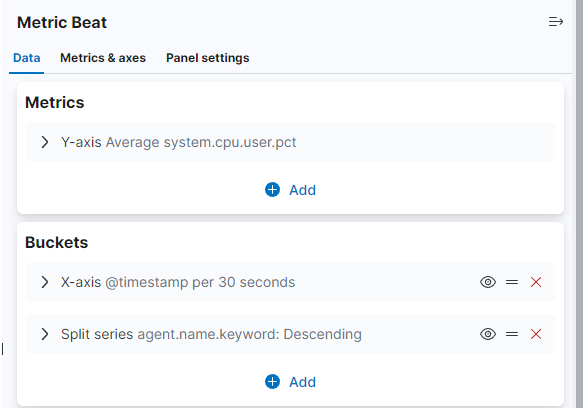
There are CPU and RAM metrics from the pipelines and containers machines.

I’ve created a visualization the metrics of the containers and pipelines machines in Kibana

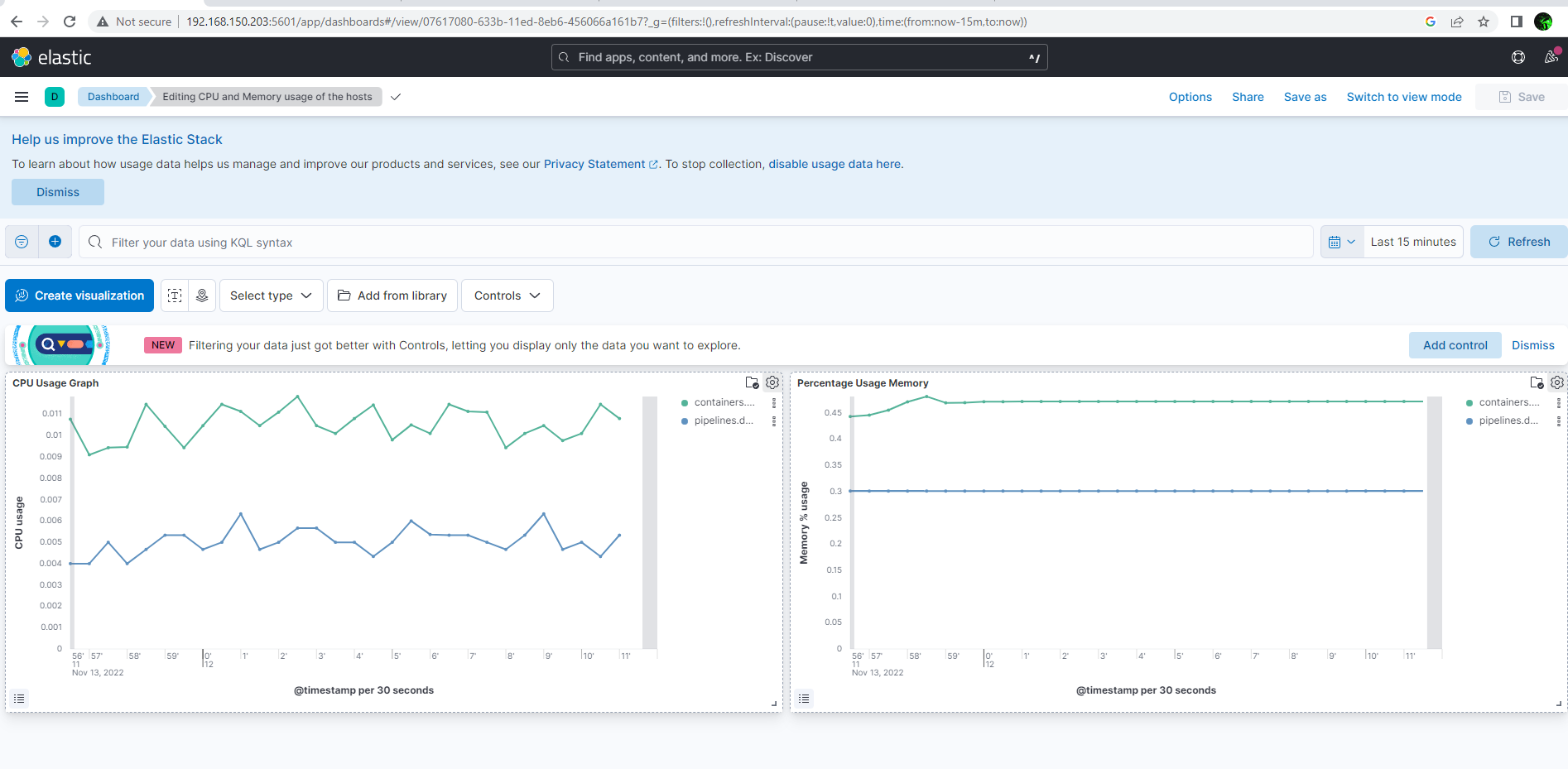


Here are their setups:





I’ve went, created the Dashboard and Added the metrics from the already created visualizations via “Add from library”. Here’s the monitoring dashboard of the 2 hosts – containers and pipelines:



The app-net for the containers is being created in the scope of the Pipeline, which can be found in the Jenkinsfile. The 3 containers are being attached to the exam-net, the application is working containerized in production mode, published on port 80 on <http://192.168.150.202:80>. The application is being deployed with the pipeline, which can be found in the Jenkinsfile.

