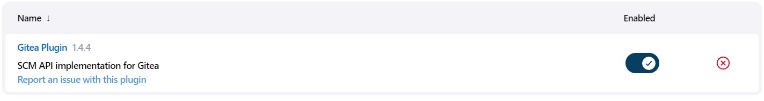
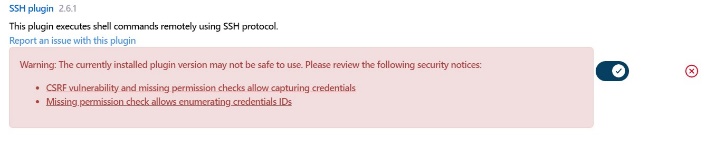
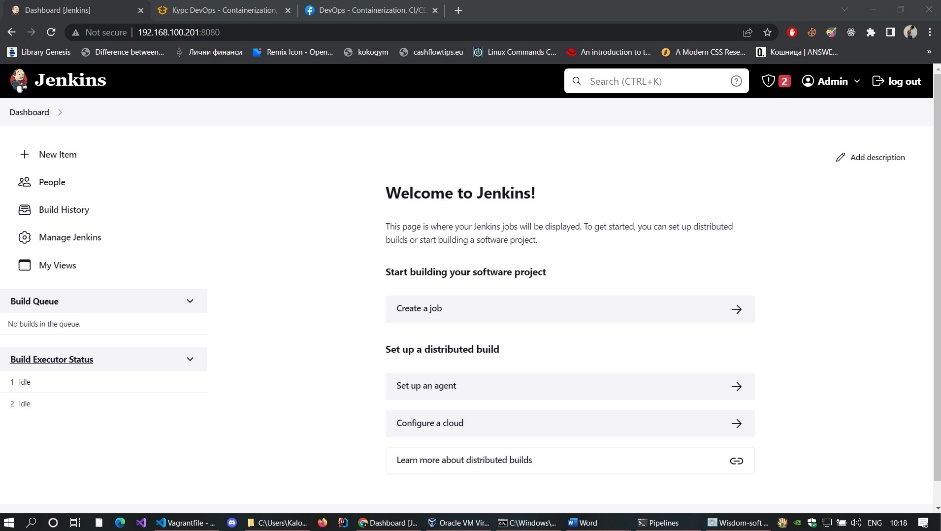
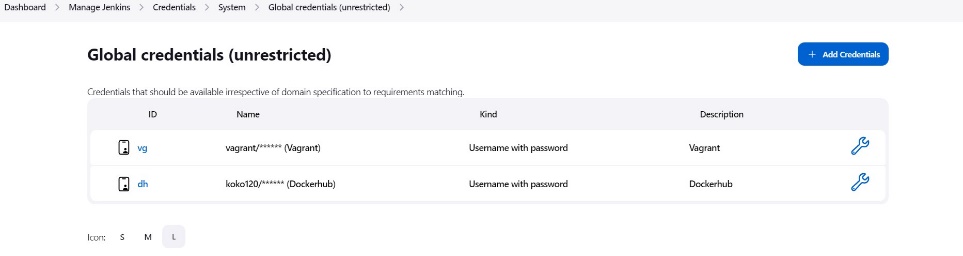
EXAM – Калоян Георгиев

Създаваме 3 виртуални машини с Vagrantfile-a, с определените имена и мрежа, спираме им Firewall, прибавяме им нужния софтуер и им настройваме параметри.

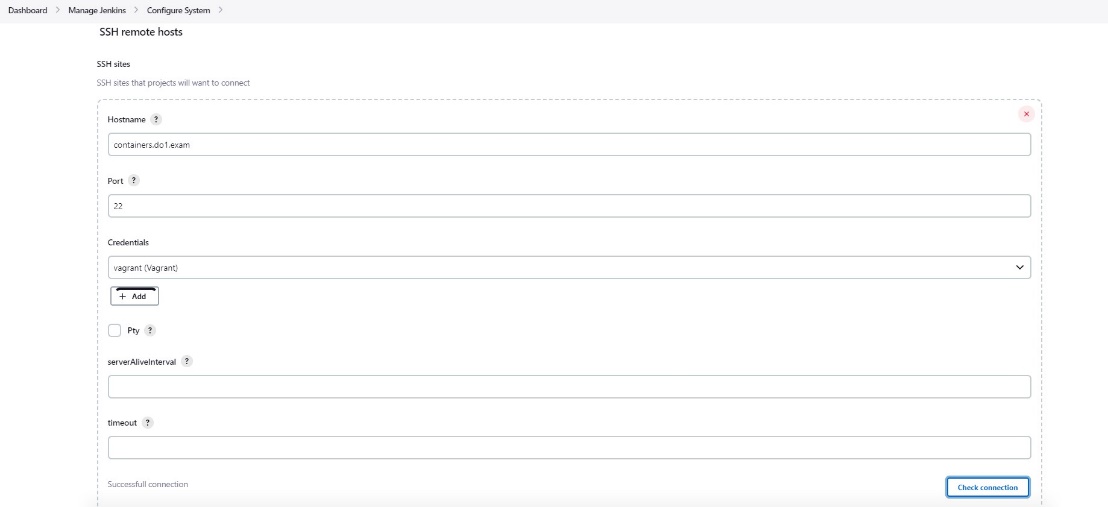
Влизаме в pipelines машината с “vagrant ssh pipelines”; “sudo cat /var/lib/jenkins/secrets/initialAdminPassword” и копираме паролалта в Jenkins, инсталираме suggested plugins; Създаваме First Admin User; Оставяме адреса на <http://192.168.100.201:8080/> ; Start using Jenkins; Инсталиране plugins SSH и Gitea от Dashboard -> Manage Jenkins -> Manage plugins -> Available plugins -> SSH и Gitea; Изтегляме и рестартираме Jenkins;



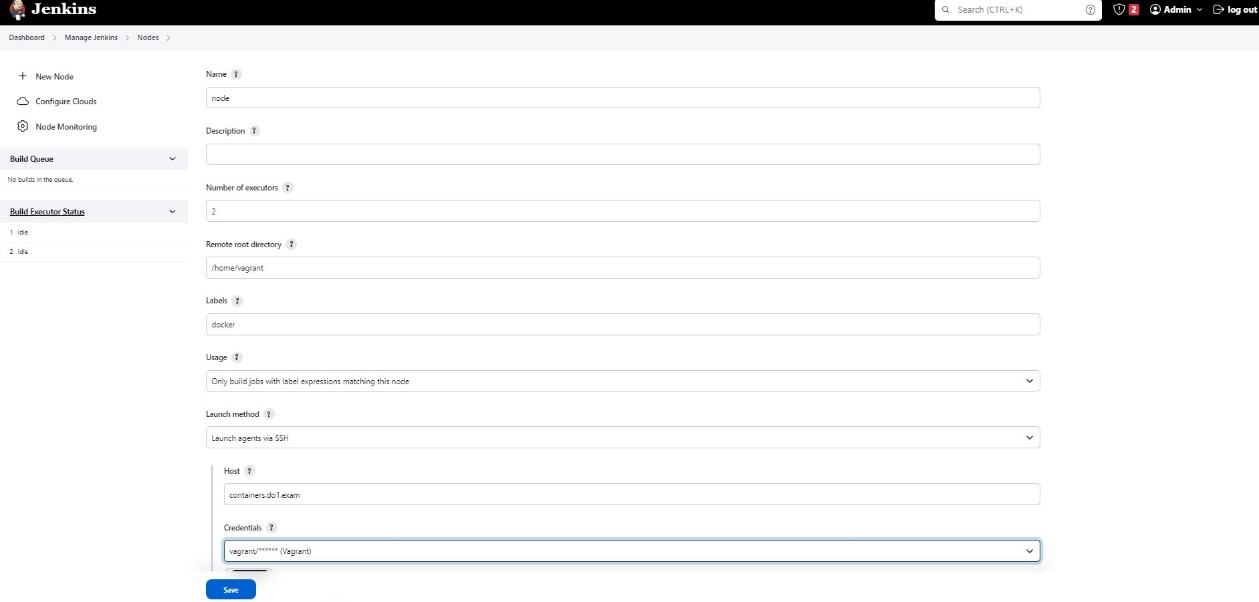
Създаваме credentials: 1. Vagrant, 2. Dockerhub – правим access token от Security в профила в Dockerhub и го поставяме като парола на този credential а username – като в dockerhub



Configure system + save



Създаваме node:



Влизаме в pipelines машината слагаме парола на jenkins, слагаме възможност за /bin/bash, влизаме в потребителя, свързваме се с container машината за да я добавим в known hosts и излизаме:

1 cat /var/lib/jenkins/secrets/initialAdminPassword

2 sudo cat /var/lib/jenkins/secrets/initialAdminPassword

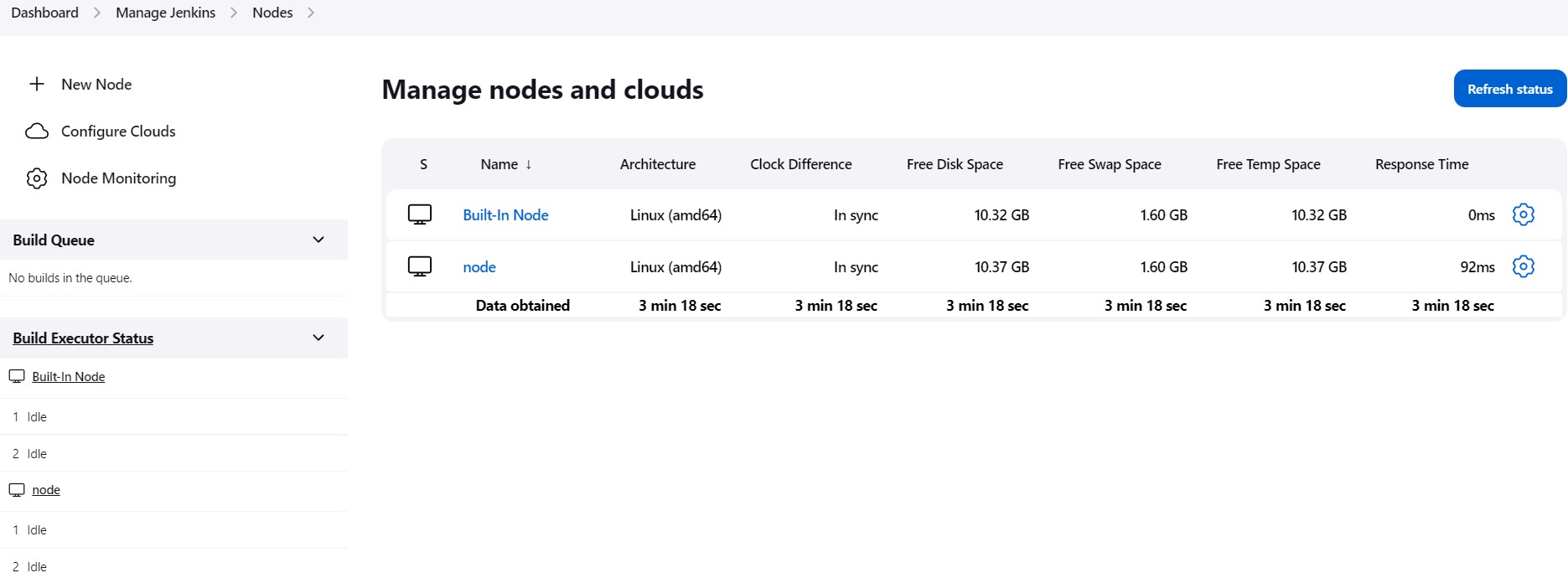
3 clear

4 sudo passwd jenkins

5 sudo vi /etc/passwd

6 su - jenkins

ssh containers.do1.exam; yes; exit;exit



Засега отиваме към Monitoring:

Променяме нужната конфигурация на Elasticsearch

sudo vi /etc/elasticsearch/elasticsearch.yml

:set number

insert

56: Uncomment network.host: ["localhost", "192.168.100.203"]

61: Uncomment http.port: 9200

17: cluster.name: examcluster

23: node.name: monitoring.do1.exam

98: xpack.security.enabled: false

esc

:wq

sudo vi /etc/elasticsearch/jvm.options.d/jvm.options

insert

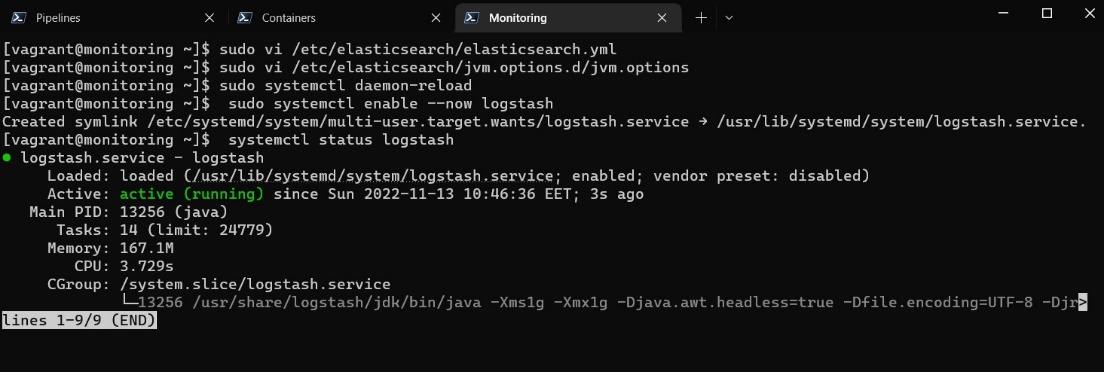
-Xms2g

-Xmx2g

esc

:wq

Пускаме го:



Променяме нужната конфигурация на Logstash:

sudo vi /etc/logstash/jvm.options

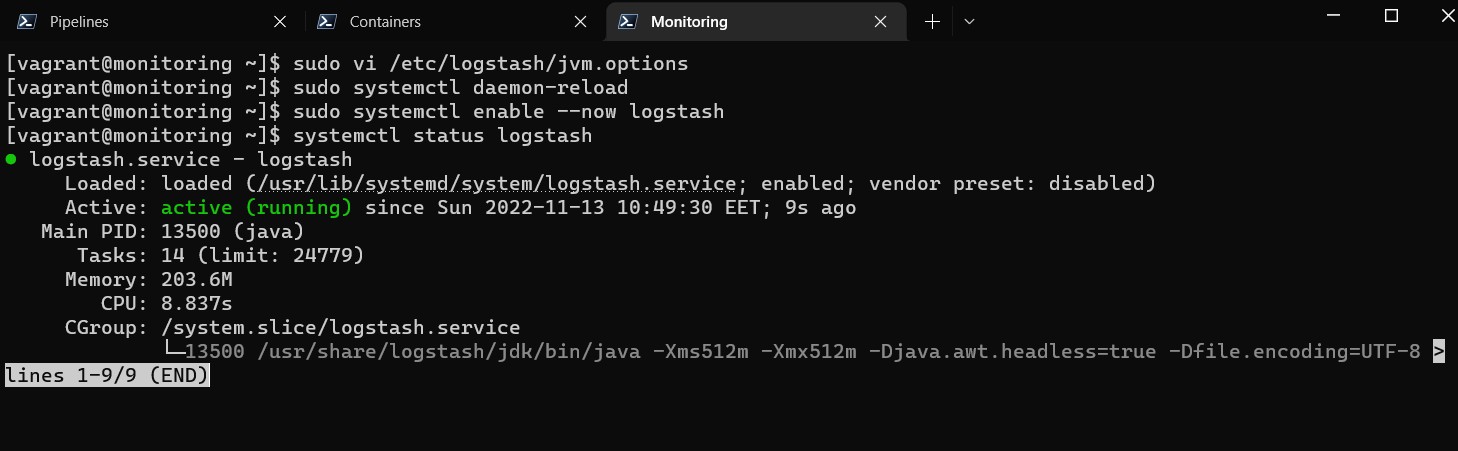
insert

-Xms512m

-Xmx512m

esc

:wq



Променяме нужната конфигурация на Kibana:

sudo vi /etc/kibana/kibana.yml

:set number

insert

6: Uncomment server.port: 5601

10: Uncomment server.host: "192.168.100.203“

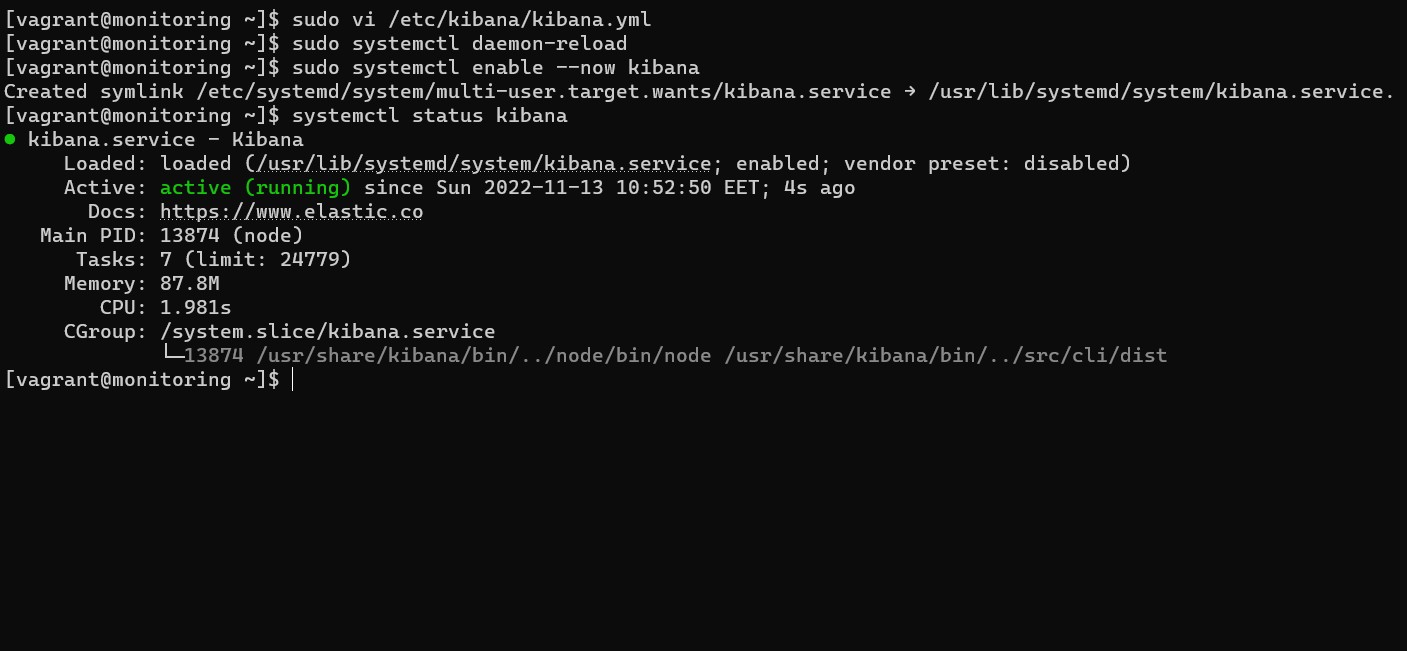
32: Uncomment server.name: "monitoring.do1.exam”

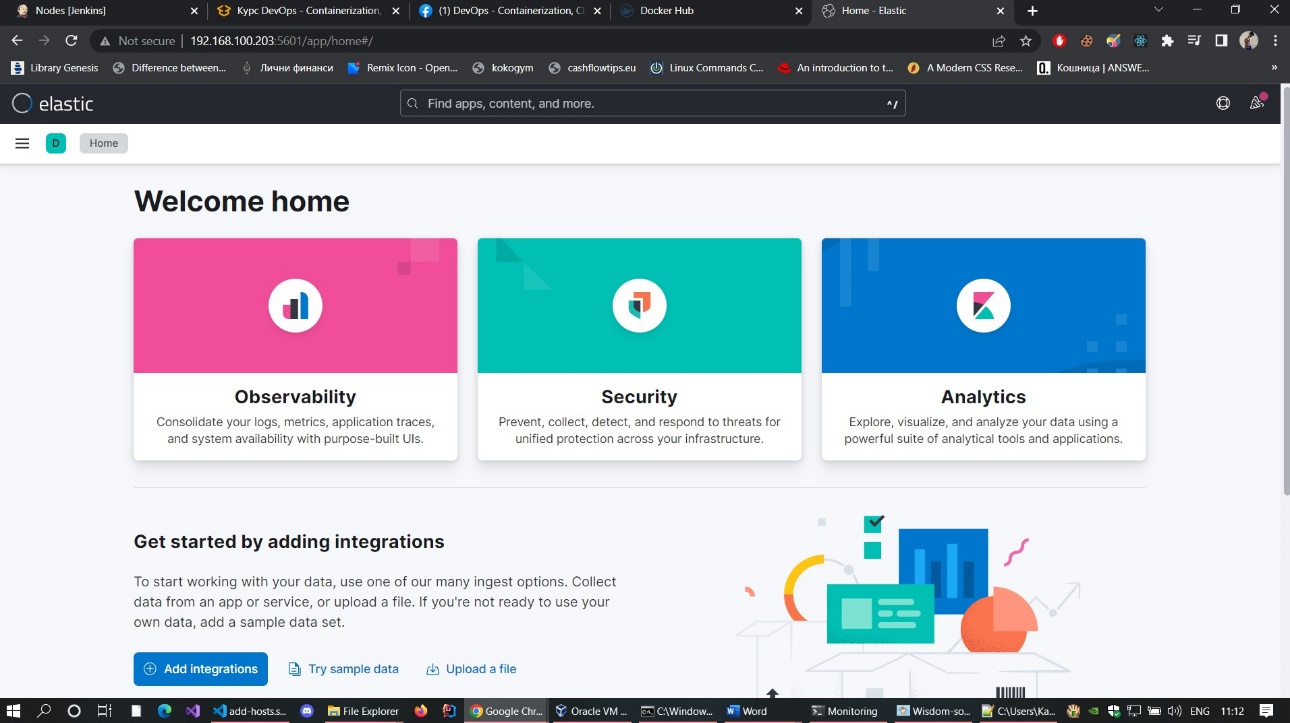
43: Uncomment elasticsearch.hosts: ["http://localhost:9200"]

esc

:wq

И пускаме:





В Logstash правим beats.conf файл за бийтовете:

sudo vi /etc/logstash/conf.d/beats.conf

insert

input {

beats {

port => 5044

}

}

output {

elasticsearch {

hosts => ["http://localhost:9200"]

index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"

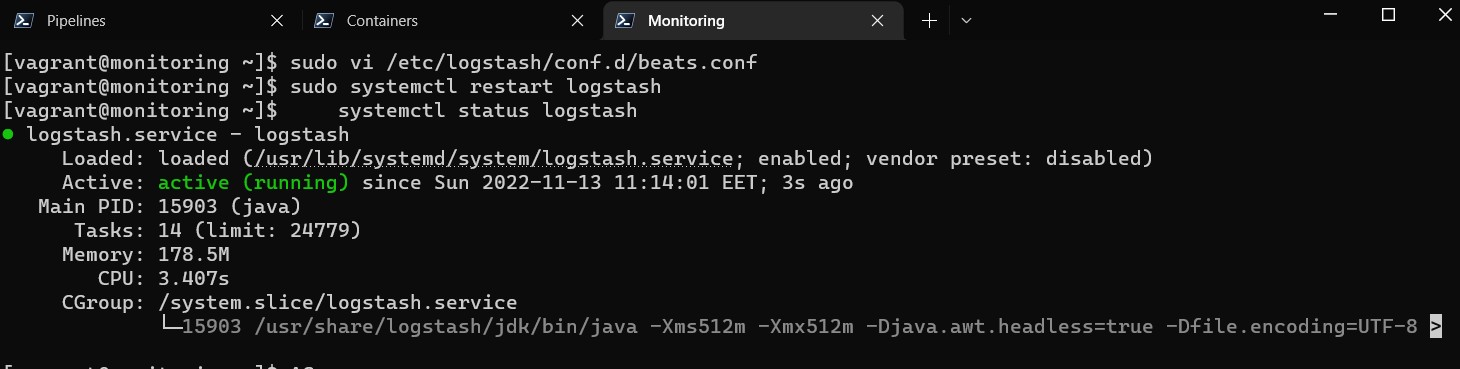
}

}

esc

:wq

Рестартираме Logstash



На двете други машини:

sudo vi /etc/metricbeat/metricbeat.yml

:set number

insert

92: disable ElasticSearch output

105: enable Logstash output with 192.168.100.203 of the Monitoring machine with port 5044

esc

:wq

sudo metricbeat test config

sudo metricbeat setup --index-management -E output.logstash.enabled=false -E 'output.elasticsearch.hosts=["<ip-of-monitoring-machine>:9200"]'

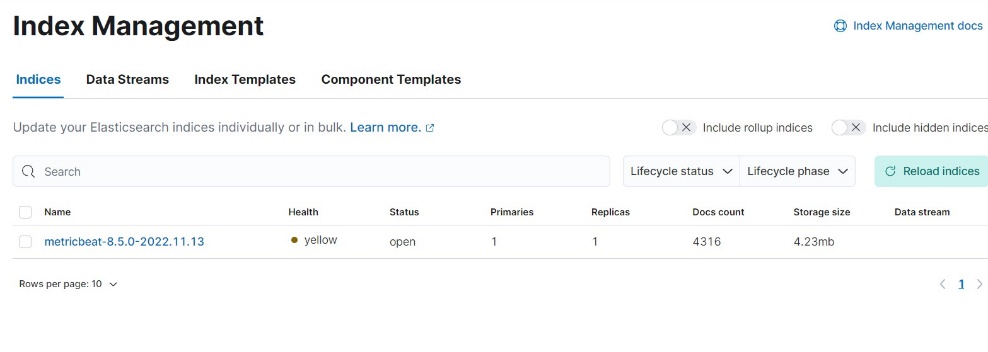
sudo systemctl daemon-reload

sudo systemctl enable --now metricbeat

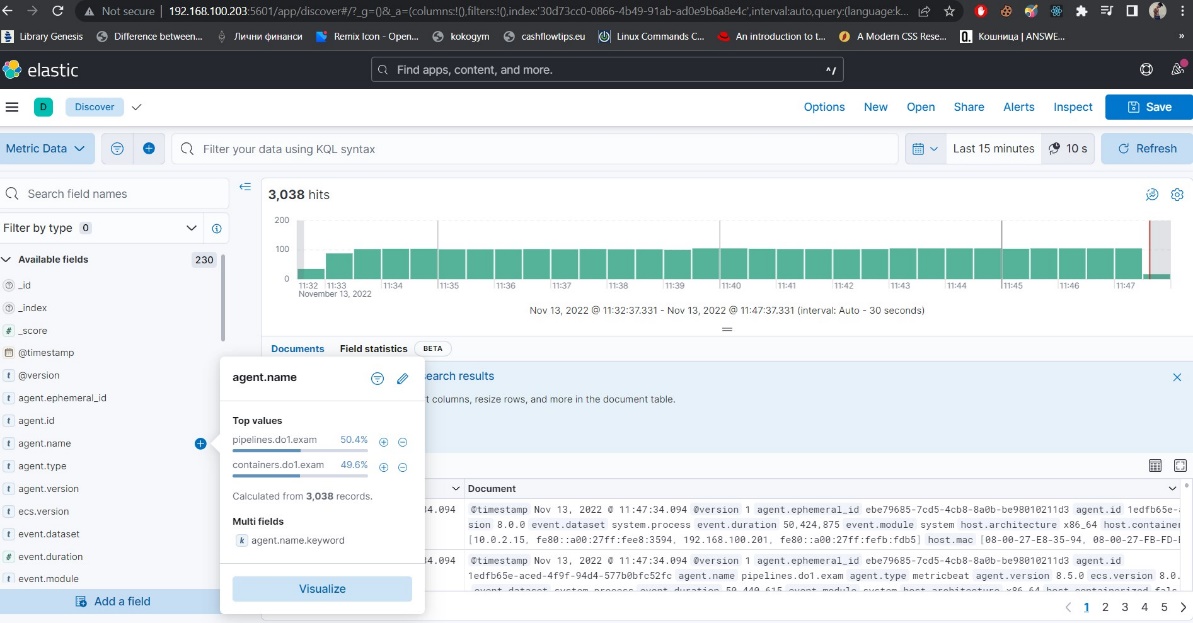
systemctl status metricbeat

sudo journalctl -xe --unit metricbeat

Виждаме успешна връзка, отиваме в Kibana -> Stack Management -> Index Management виждаме:



Отиваме в Data View и правим data view Metric Data с Index pattern: metricbeat-8.5.0-\*, отиваме на discover и виждаме:

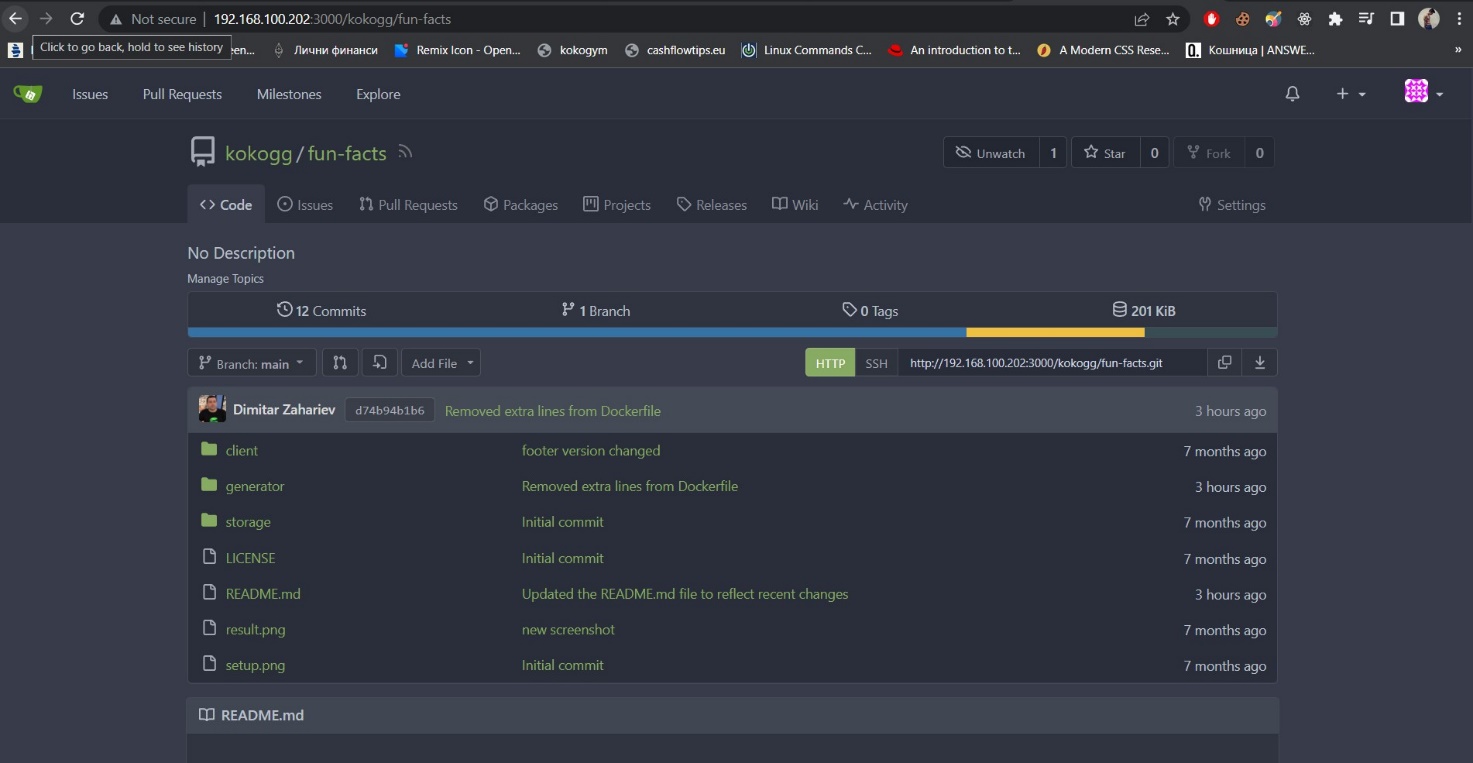


Връщаме се към Gitea да направим pipeline, оставяме visualizations за после:

В containers инсталираме gitea:

cp /vagrant/docker-compose.yml . && docker compose up -d

Влизаме в нея на 192.168.100.202:3000;регистрираме се; add migration from [**https://github.com/shekeriev/fun-facts**](https://github.com/shekeriev/fun-facts)

****

Отиваме в Jenkins Dashboard- > new Item -> name Exam > pipeline -> ok -> tick GitHub project

Pipeline code for getting the project from Gitea, building the images, running the application in test mode:

pipeline

{

agent

{

label 'docker'

}

environment

{

DOCKERHUB\_CREDENTIALS=credentials('docker-hub')

}

stages

{

stage('Clone')

{

steps

{

git branch: 'main', url: 'http://192.168.100.202:3000/kokogg/fun-facts'

}

}

stage('Build')

{

steps

{

sh 'cd client && docker image build -t img-client .'

sh 'cd storage && docker image build -t img-storage .'

sh 'cd generator && docker image build -t img-generator .'

}

}

stage('Network')

{

steps

{

sh 'docker network ls | grep fun-facts || docker network create fun-facts'

}

}

stage('Run')

{

steps

{

sh 'docker container rm -f con-client || true'

sh 'docker container run -d --name con-client --net fun-facts -p 8080:5000 img-client'

// sh 'docker container run -d --name con-client --net fun-facts -p 8080:5000 img-client'

sh 'docker container rm -f con-storage || true'

sh 'docker container run -d --name con-storage --net fun-facts -e MYSQL\_ROOT\_PASSWORD=ExamPa\\$\\$w0rd img-storage'

//sh 'docker container run -d --name con-storage --net fun-facts -e MYSQL\_ROOT\_PASSWORD=ExamPa$$w0rd img-storage'

sh 'docker container rm -f con-generator || true'

sh 'docker container run -d --name con-generator --net fun-facts img-generator'

}

}/\*

stage('Login')

{

steps

{

sh 'echo \\$DOCKERHUB\_CREDENTIALS\_PSW | docker login -u \\$DOCKERHUB\_CREDENTIALS\_USR --password-stdin'

}

}

stage('Push')

{

steps

{

sh 'docker image tag fun-facts koko120/fun-facts'

sh 'docker push koko120/fun-facts'

}

}\*/

}

post

{

always

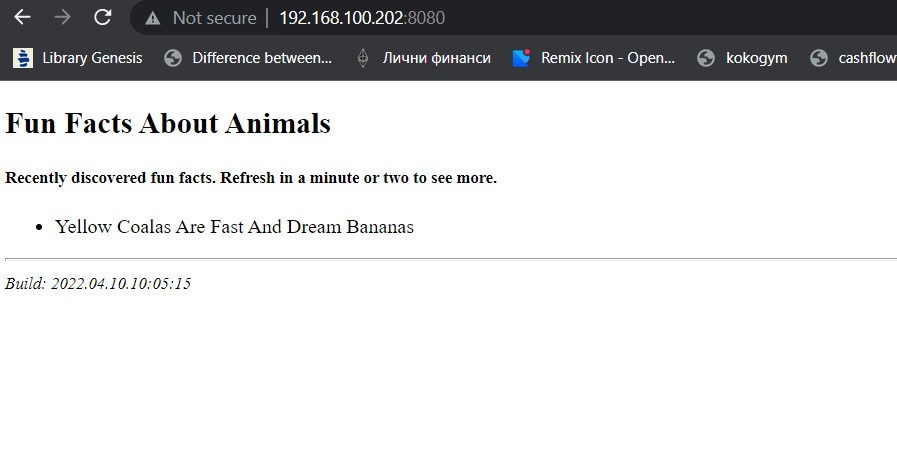
{

cleanWs()

}

}

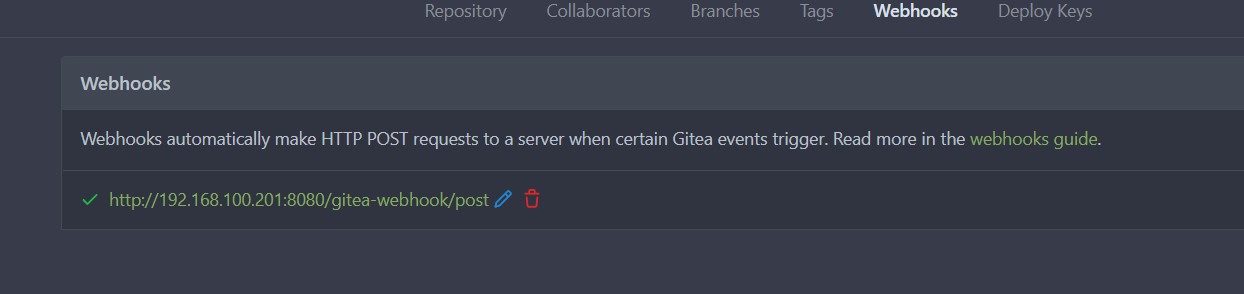
}



Tick GitHub hook trigger for GITScm polling and Poll SCM; Отиваме в gitea, settings, webohooks, add webhook, gitea;

Target URL:

<http://192.168.100.201:8080/gitea-webhook/post>



Правим промяна в ***client/code/app.dat*** и commit

