Phase1: Environment Setup

Network Environment Configuration:

1. Root user login AWS via email

LINK: <https://ca-central-1.console.aws.amazon.com/vpcconsole/home?region=ca-central-1#Home>

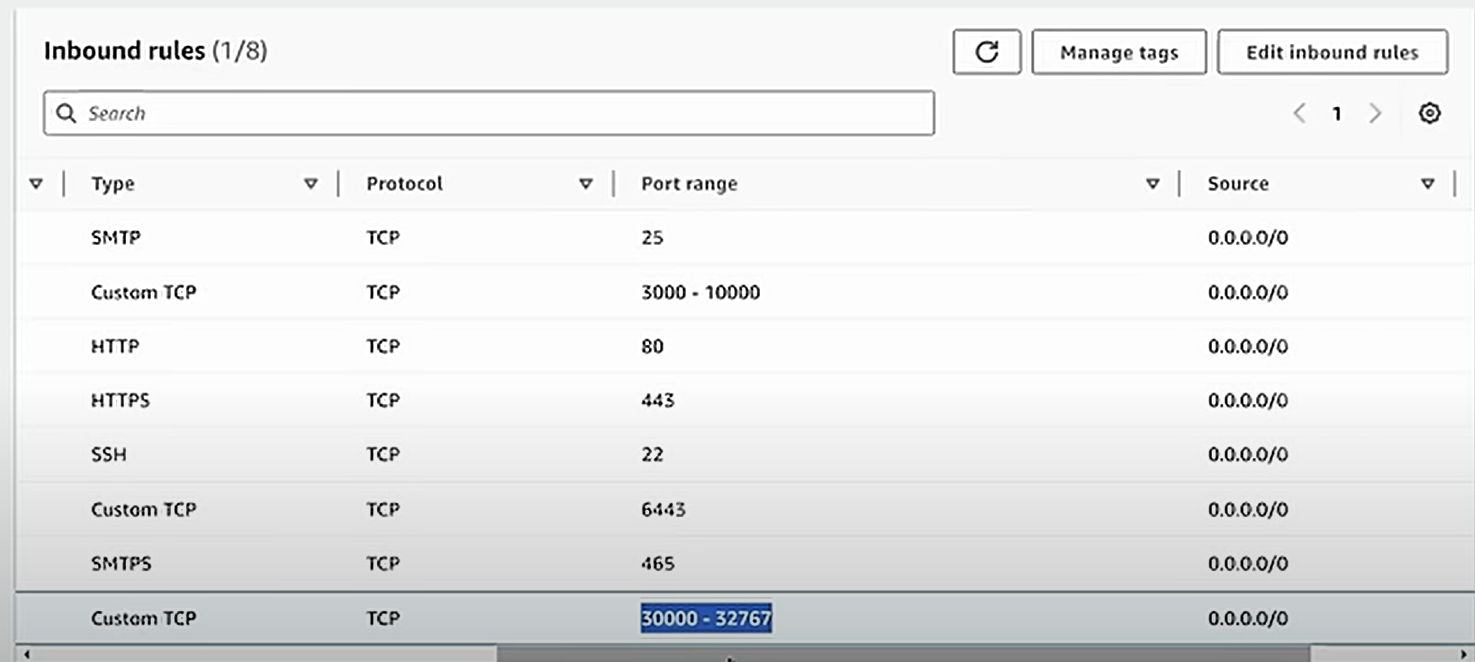
1. Create Private Network Environment:

* Go to VPC and rename the default VPC to use it further- Lets name DevOps-VPC

1. Add Security Groups:

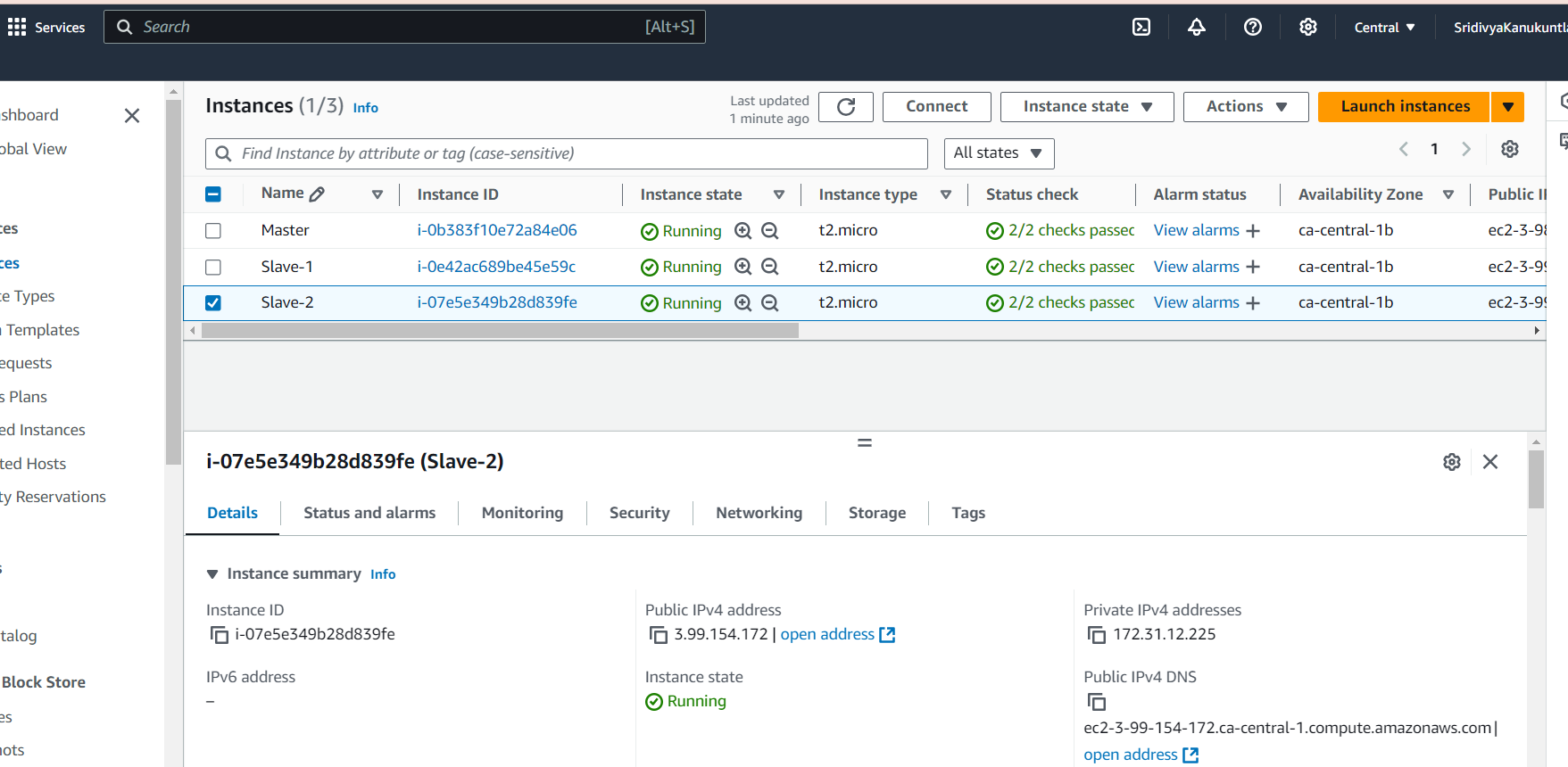
* Go to **EC2 Instance > Security Groups** create custom inbound rules that will act as firewalls for resources.

Create or edit the inbound rules as shown below:

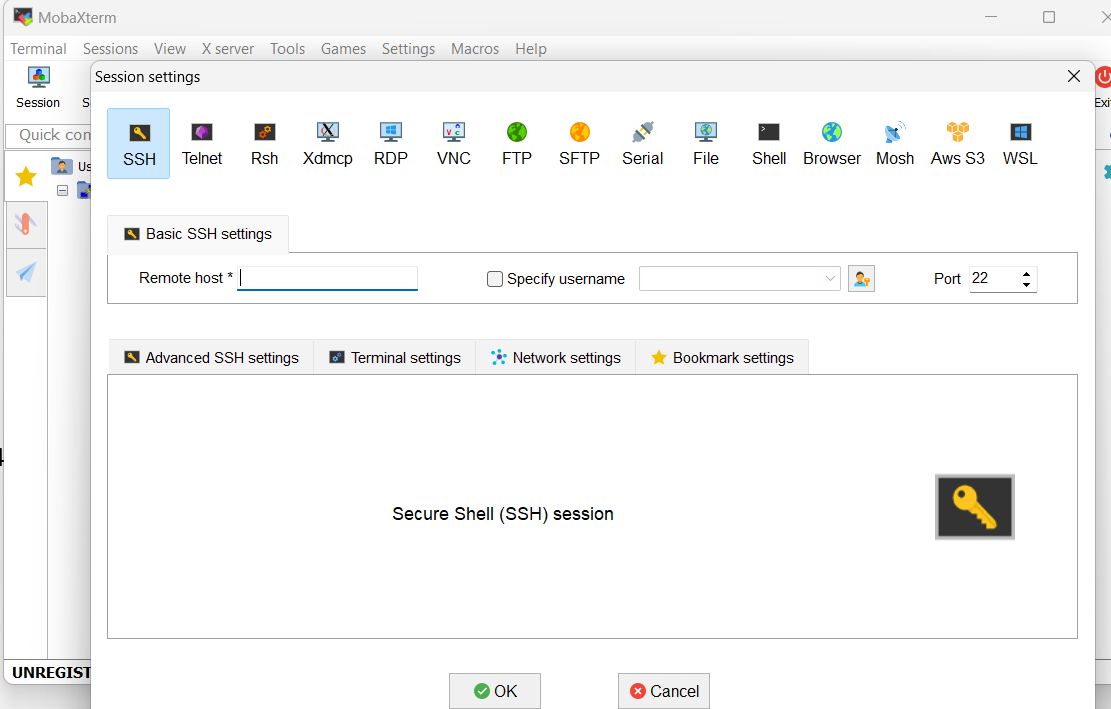


1. **EC2 Instance Setup**:

* After adding security group create 3 EC2 instances: 1 Master, 2 slave.
* Select Ubuntu, apply custom security group that we have created in the above step, create a private key ex: key.pem, and 25 GB storage.
* Click Launch Instance and Name the instances as master, slave1, slave2:



1. **Connect EC2 Instances Using MobaXterm**:

* For this install VM –MobaXterm and add the EC2 instance in the MobaXterm - SSH- Remote hostname:
* Enter the remote host name: IP of master Instance
* Take duplicate sessions and use Slave1 and slave 2 ipaddresses respectively and under advanced SSH setting select the key.pem as private key

1. Connecting to Master and Slave instances from VM

* After connecting as shown in step 4 now enter into sudo mode on all the 3 sessions:

$sudu su

## **Setting up Kubernetes Cluster:**

* Now we will run commands to run Kubernetes Cluster.
* Let’s group the commands under script 1.sh and run them at once: execute 1.sh on master ,slave1 and slave 2 servers:

ubuntu@ip-172-31-21-20:~$ sudo su

root@ip-172-31-21-20:/home/ubuntu# vi 1.sh

root@ip-172-31-21-20:/home/ubuntu# chmod +x 1.sh

root@ip-172-31-21-20:/home/ubuntu# ./1.sh

cat 1.sh

Update System Packages [On Master & Worker Node]

* sudo apt-get update

Install Docker[On Master & Worker Node]

* sudo apt install docker.io -y
* sudo chmod 666 /var/run/docker.sock

Install Required Dependencies for Kubernetes[On Master & Worker Node]

* sudo apt-get install -y apt-transport-https ca-certificates curl gnupg
* sudo mkdir -p -m 755 /etc/apt/keyrings

Add Kubernetes Repository and GPG Key[On Master & Worker Node]

* curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
* echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

Update Package List[On Master & Worker Node]

* sudo apt update

**Install Kubernetes Components [On Master & Worker Node]**

* sudo apt install -y kubeadm=1.28.1-1.1 kubelet=1.28.1-1.1 kubectl=1.28.1-1.1
* kubeadm- to setup kubernetes cluster
* kubelet-responsible to create pods on which we deploy applications
* kubectl-acts as CLI to interact with kubernetes cluster

**Initialize Kubernetes Master Node [On MasterNode]**

* sudo kubeadm init --pod-network-cidr=10.244.0.0/16

10.244.0.0/16 will allow around 6k IP’s

After running this command on master, the output consists a command to run on Slaves, which will make the slave nodes as Worker nodes and on these slave nodes deployments will be done

**O/P:**   
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

You should now deploy a pod network to the cluster.

Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:

https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 172.31.15.12:6443 --token bd29zb.wtnsmj2p88dn51c8 \

--discovery-token-ca-cert-hash sha256:d084000d3d983e3cf423c15098438b231a789af6276b373558e0253c481521b9

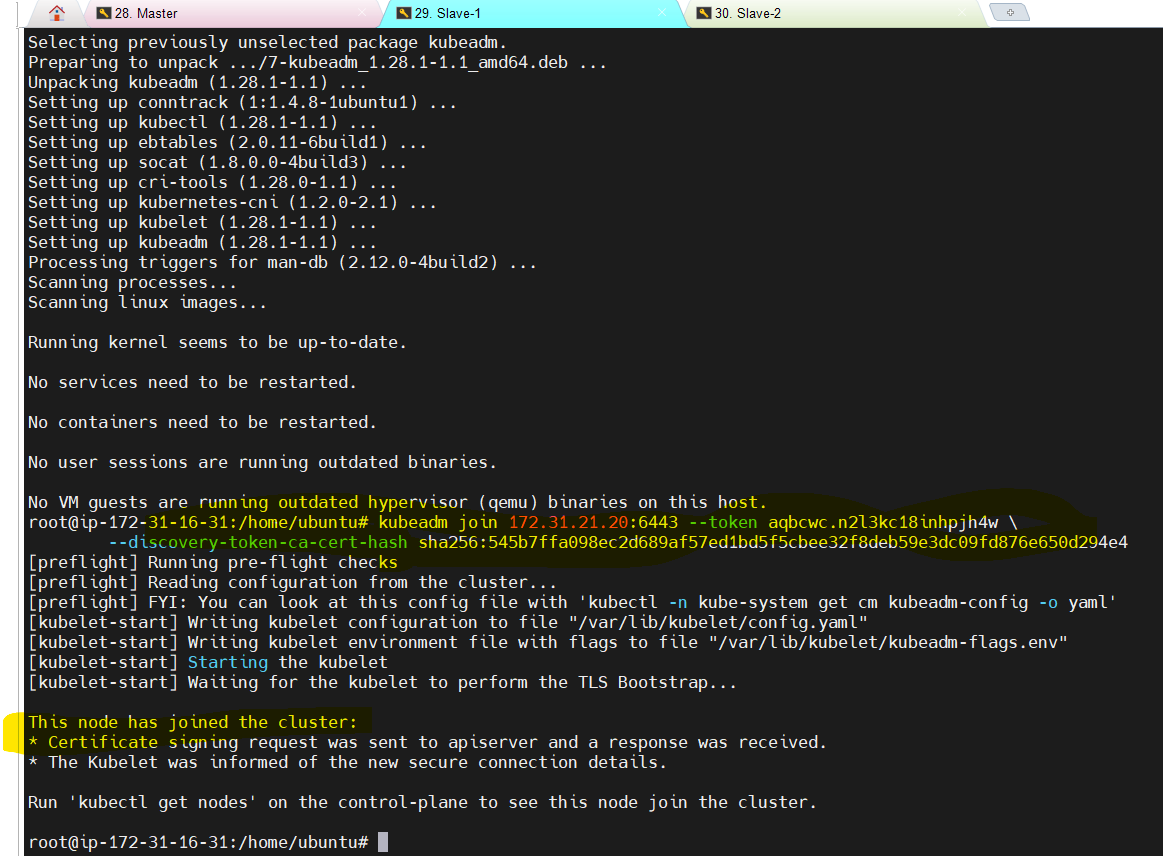
Now based on the output let’s run the below commands:

**Join Slave Nodes to Cluster**:

Run the kubeadm join from the master node command output on slave 1 and slave2 which makes them join the cluster:

kubeadm join 172.31.15.12:6443 --token bd29zb.wtnsmj2p88dn51c8 \

--discovery-token-ca-cert-hash sha256:d084000d3d983e3cf423c15098438b231a789af6276b373558e0253c481521b9



**Configure Kubernetes Cluster [On MasterNode]**

* mkdir -p $HOME/.kube
* sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
* sudo chown $(id -u):$(id -g) $HOME/.kube/config

**Deploy Networking Solution (Calico) [On MasterNode]**

* kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

**Deploy Ingress Controller (NGINX) [On MasterNode]**

* kubectl apply -f <https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.49.0/deploy/static/provider/baremetal/deploy.yaml>

root@ip-172-31-21-20:/home/ubuntu# mkdir -p $HOME/.kube

root@ip-172-31-21-20:/home/ubuntu# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

root@ip-172-31-21-20:/home/ubuntu# sudo chown $(id -u):$(id -g) $HOME/.kube/config

root@ip-172-31-21-20:/home/ubuntu# kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml

**O/P:**

poddisruptionbudget.policy/calico-kube-controllers created

serviceaccount/calico-kube-controllers created

serviceaccount/calico-node created

configmap/calico-config created

customresourcedefinition.apiextensions.k8s.io/bgpconfigurations.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/bgppeers.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/blockaffinities.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/caliconodestatuses.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/clusterinformations.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/felixconfigurations.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/globalnetworkpolicies.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/globalnetworksets.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/hostendpoints.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/ipamblocks.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/ipamconfigs.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created

customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created

clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created

clusterrole.rbac.authorization.k8s.io/calico-node created

clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created

clusterrolebinding.rbac.authorization.k8s.io/calico-node created

daemonset.apps/calico-node created

deployment.apps/calico-kube-controllers created

**root@ip-172-31-21-20:/home/ubuntu# kubectl apply –f**

[**https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.49.0/deploy/static/provider/baremetal/deploy.yaml**](https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.49.0/deploy/static/provider/baremetal/deploy.yaml)

namespace/ingress-nginx created

serviceaccount/ingress-nginx created

configmap/ingress-nginx-controller created

clusterrole.rbac.authorization.k8s.io/ingress-nginx created

clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx created

role.rbac.authorization.k8s.io/ingress-nginx created

rolebinding.rbac.authorization.k8s.io/ingress-nginx created

service/ingress-nginx-controller-admission created

service/ingress-nginx-controller created

deployment.apps/ingress-nginx-controller created

validatingwebhookconfiguration.admissionregistration.k8s.io/ingress-nginx-admission created

serviceaccount/ingress-nginx-admission created

clusterrole.rbac.authorization.k8s.io/ingress-nginx-admission created

clusterrolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created

role.rbac.authorization.k8s.io/ingress-nginx-admission created

rolebinding.rbac.authorization.k8s.io/ingress-nginx-admission created

job.batch/ingress-nginx-admission-create created

job.batch/ingress-nginx-admission-patch created

Run on master node:

root@ip-172-31-21-20:/home/ubuntu# kubectl get nodes

NAME STATUS ROLES AGE VERSION

ip-172-31-16-31 Ready <none> 2m39s v1.28.1

ip-172-31-21-20 Ready control-plane 6m36s v1.28.1

ip-172-31-28-137 Ready <none> 2m31s v1.28.1

1. **Cluster Auditing with Kubeaudit:**

**Install Kubeaudit:**

* To scan kubernetes cluster let’s use Kubeaudit on Master node:

<https://github.com/Shopify/kubeaudit/releases>

* Select the amd64 link and use wget in master node:

wget <https://github.com/Shopify/kubeaudit/releases/download/v0.22.2/kubeaudit_0.22.2_linux_amd64.tar.gz>

then—

$root@ip-172-31-15-12:/home/ubuntu# tar -xvzf kubeaudit\_0.22.2\_linux\_amd64.tar.gz

README.md

Kubeaudit

**Run Audit:**

$ kubeaudit all

This gives a report to be used infra and others

The output might have errors and warnings as we are still in the phase of setting up the cluster and need to work on it.

############

Troubleshoot If kube-audit throws any error :

kubeaudit: command not found

The error “kubeaudit: command not found” suggests that the kubeaudit binary is either not installed or not properly linked in your system’s PATH. Since you have the kubeaudit\_0.22.2\_linux\_amd64.tar.gz file in the current directory, you’ll need to extract it and ensure the binary is executable and accessible.

***Here’s how to proceed:***

**Extract the kubeaudit Archive:**

tar -xzf kubeaudit\_0.22.2\_linux\_amd64.tar.gz

**Move the Binary to a Directory in PATH:**

Move the extracted kubeaudit binary to /usr/local/bin or another directory in your PATH.

sudo mv kubeaudit /usr/local/bin/

**Make the Binary Executable (if needed):**

sudo chmod +x /usr/local/bin/kubeaudit

**Verify Installation:**

Confirm that kubeaudit is accessible by running:

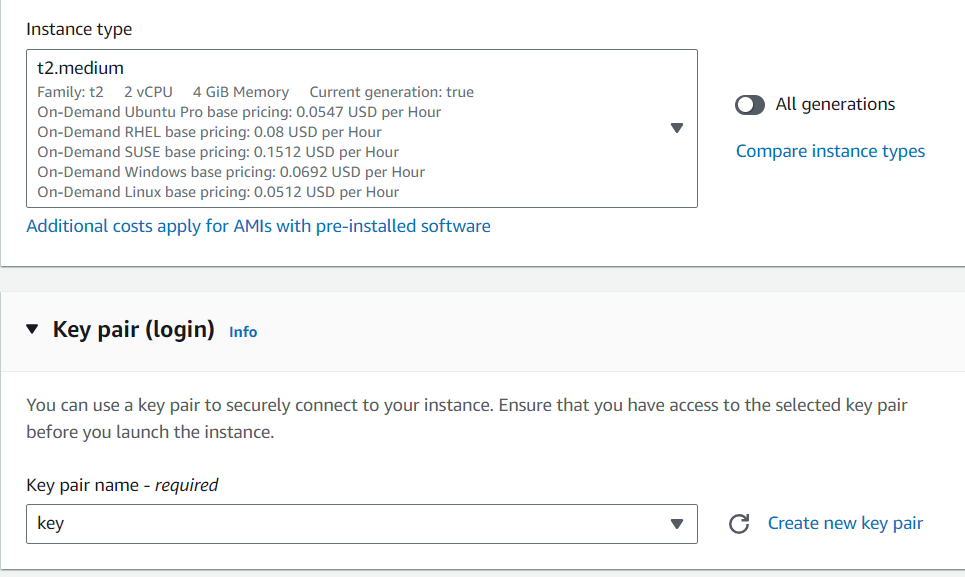
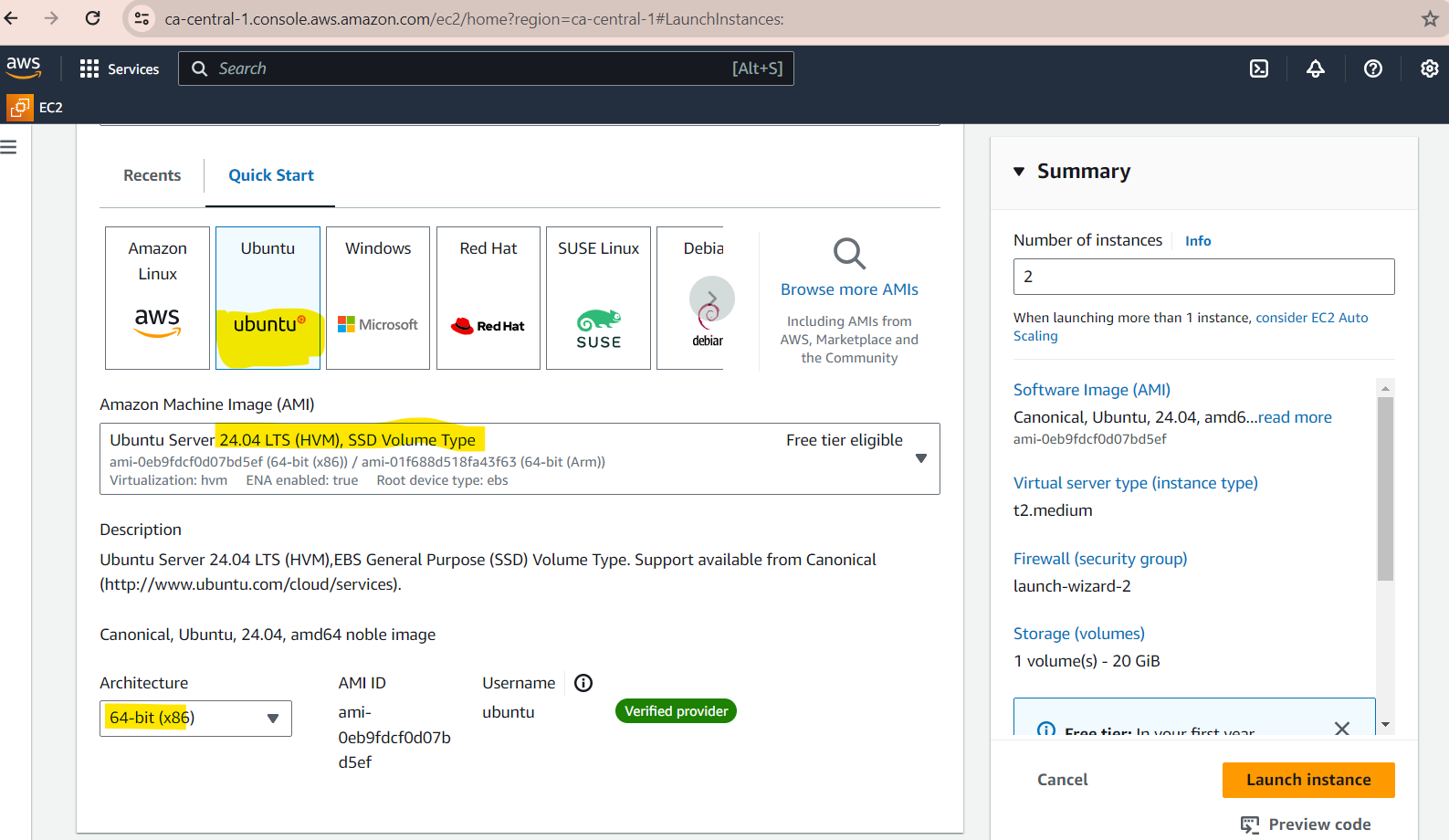
kubeaudit all

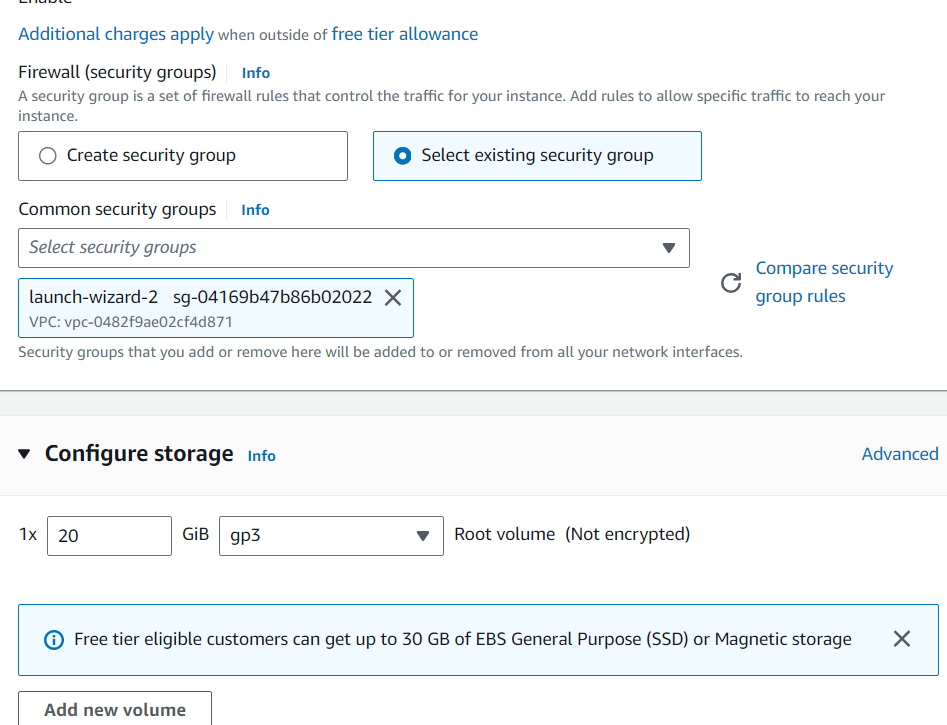
##########

## Virtual Machines Setup for Tools:

1. **Create Instances for SonarQube and Nexus**:

* Launch two Ubuntu instances, select t2.large, apply security groups, and connect using MobaXterm

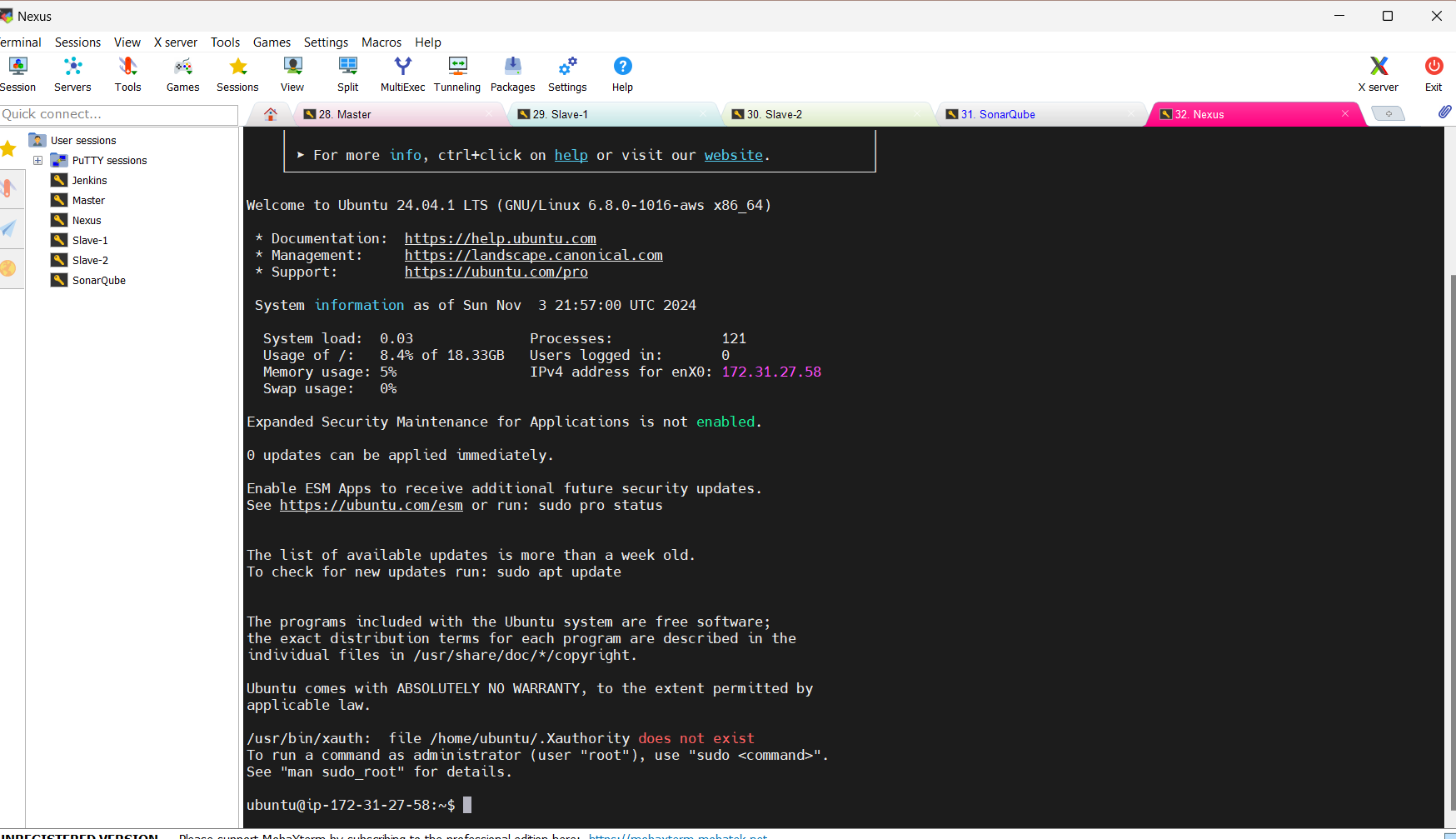




1. **Create Instance for Jenkins**:

* Launch an Ubuntu instance, select t3.large, apply security groups, and connect via MobaXterm.

Overview:



1. Now setup SONAR QUBE and NEXUS
2. On SONAR QUBE and NEXUs server:

sudo apt update

Then create a script with commands to get Docker

### Docker Installation

vi dock.sh

#!/bin/bash

# Update package manager repositories

sudo apt-get update

# Install necessary dependencies

sudo apt-get install -y ca-certificates curl

# Create directory for Docker GPG key

sudo install -m 0755 -d /etc/apt/keyrings

# Download Docker's GPG key

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

# Ensure proper permissions for the key

sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add Docker repository to Apt sources

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \

$(. /etc/os-release && echo "$VERSION\_CODENAME") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Update package manager repositories

sudo apt-get update

sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

With above script Package for Docker is made available and installed on both SonarQube and Nexus

Now validate if unubtu user is able to run docker commands:

root@ip-172-31-26-9:/home/ubuntu# sudo chmod 666 /var/run/docker.sock

root@ip-172-31-26-9:/home/ubuntu# docker pull hello-world

Using default tag: latest

latest: Pulling from library/hello-world

c1ec31eb5944: Pull complete

Digest: sha256:d211f485f2dd1dee407a80973c8f129f00d54604d2c90732e8e320e5038a0348

Status: Downloaded newer image for hello-world:latest

docker.io/library/hello-world:latest

### Creating Docker Containers for Sonar Qube and Nexus

To pull docker image and create container from it

Execute in detached mode using –d:

*On SonarQube*

docker run -d --name sonar -p 9000:9000 sonarqube:lts-community

**root@ip-172-31-26-9:/home/ubuntu# docker run -d --name sonar -p 9000:9000 sonarqube:lts-community**

Unable to find image 'sonarqube:lts-community' locally

lts-community: Pulling from library/sonarqube

6414378b6477: Pull complete

17da8ec43a12: Pull complete

d12988e90d61: Pull complete

f4d133ca2b7f: Pull complete

143733ae87a4: Pull complete

d021fe0fda1e: Pull complete

bc52ff982b3d: Pull complete

4f4fb700ef54: Pull complete

Digest: sha256:aac4b3bdd2ae29c95e1a4c07bc50f0cc81d1914019bc609e92dee25e72ff2e6d

Status: Downloaded newer image for sonarqube:lts-community

098f57044a22501bf0e1e4dd9558ef6e99e69200cf054d121437c9a098ecc27c

**root@ip-172-31-26-9:/home/ubuntu# docker ps**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

098f57044a22 sonarqube:lts-community "/opt/sonarqube/dock…" 3 minutes ago Up 3 minutes 0.0.0.0:9000->9000/tcp, :::9000->9000/tcp sonar

*On Nexus Server:*

root@ip-172-31-27-58:/home/ubuntu# docker run -d --name nexus -p 8081:8081 sonatype/nexus3:latest

Unable to find image 'sonatype/nexus3:latest' locally

latest: Pulling from sonatype/nexus3

ccc2996f86eb: Pull complete

e20932f8434f: Pull complete

7940a458dc60: Pull complete

c0e276cf3782: Pull complete

b59a6e4250e9: Pull complete

a3afbeb554d9: Pull complete

fc3ab5b0b39e: Pull complete

Digest: sha256:3995b631ed941eb21d071d2d6b2c55ba98a502fb1afef6bd78e43f94583ffbaf

Status: Downloaded newer image for sonatype/nexus3:latest

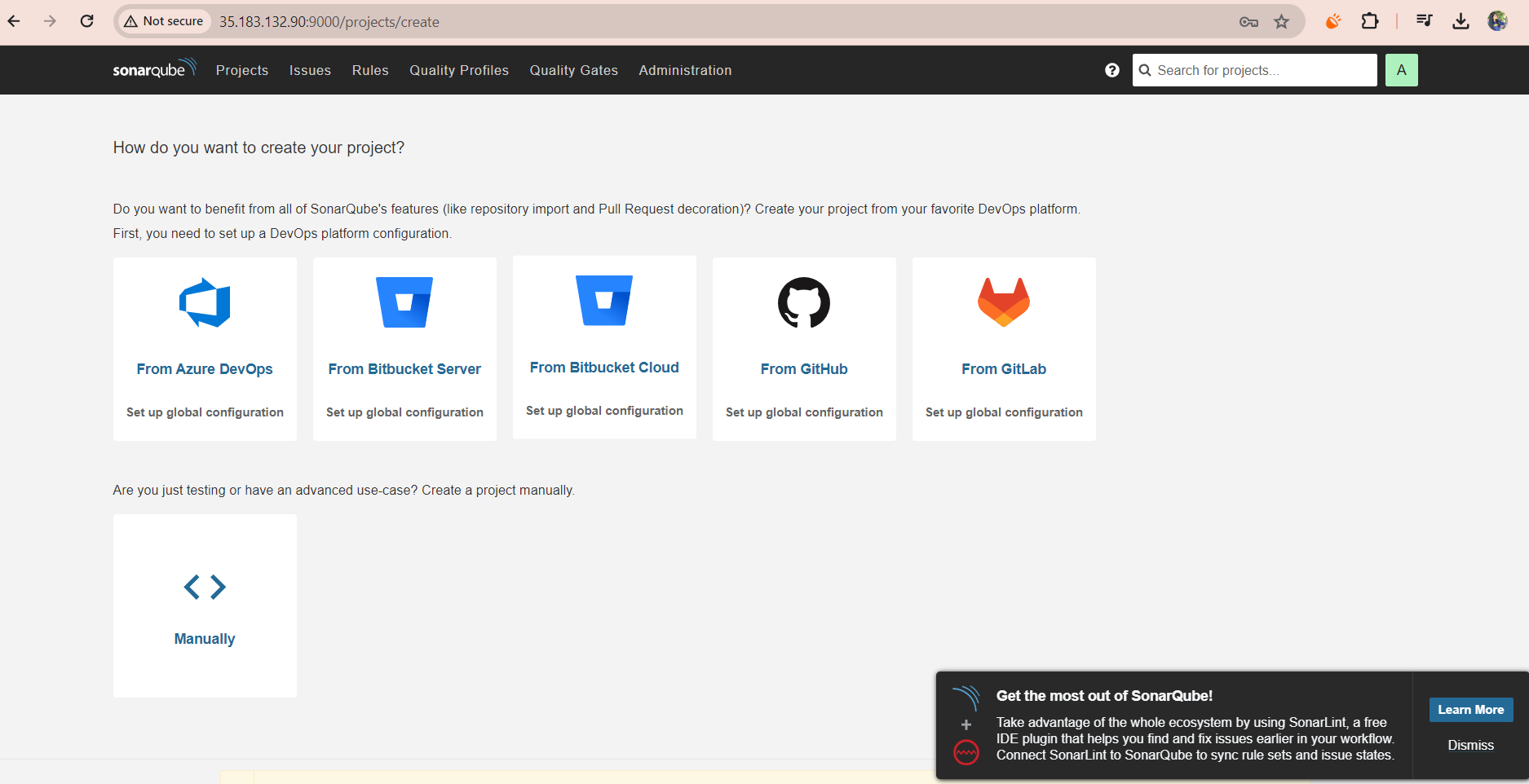
f50ec2234ca133354508e99e3098824d133177c34fa34a481e55c3ae548bf49e

root@ip-172-31-27-58:/home/ubuntu# docker ps

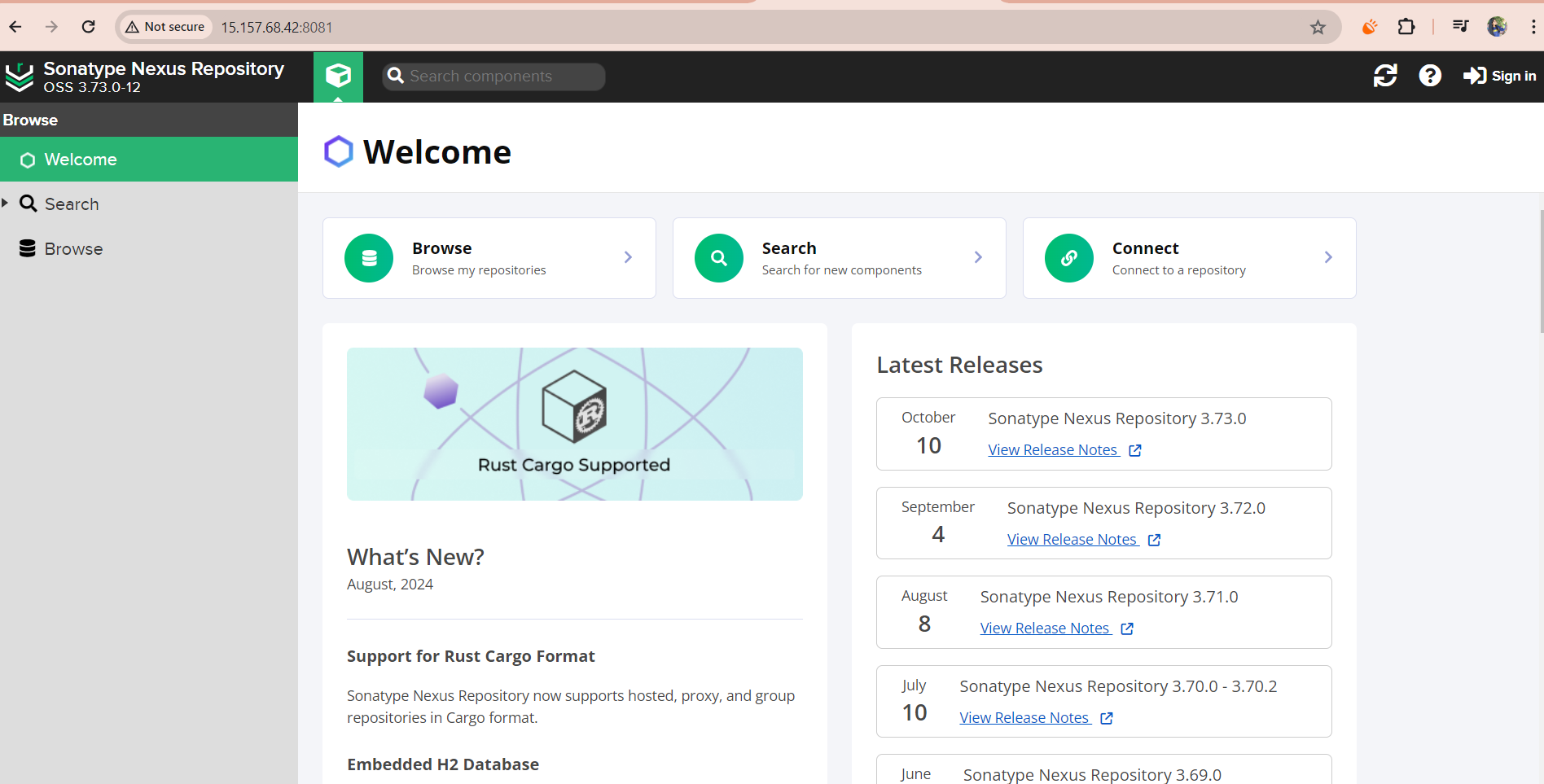
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

f50ec2234ca1 sonatype/nexus3:latest "/opt/sonatype/nexus…" About a minute ago Up About a minute 0.0.0.0:8081->8081/tcp, :::8081->8081/tcp nexus

SonarQube: in chrome use SonarQube host <http://35.183.132.90:9000/projects/create>



Nexus: use Nexus ip and default port 8081

http://15.157.68.42:8081/#browse/browse  


To sign in get the password from nexus password directory:

root@ip-172-31-27-58:/home/ubuntu# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

f50ec2234ca1 sonatype/nexus3:latest "/opt/sonatype/nexus…" About a minute ago Up About a minute 0.0.0.0:8081->8081/tcp, :::8081->8081/tcp nexus

root@ip-172-31-27-58:/home/ubuntu# docker exec -it f50ec2234ca1 /bin/bash

bash-4.4$ ls

nexus sonatype-work start-nexus-repository-manager.sh

bash-4.4$ cd sonatype-work/

bash-4.4$ ls

nexus3

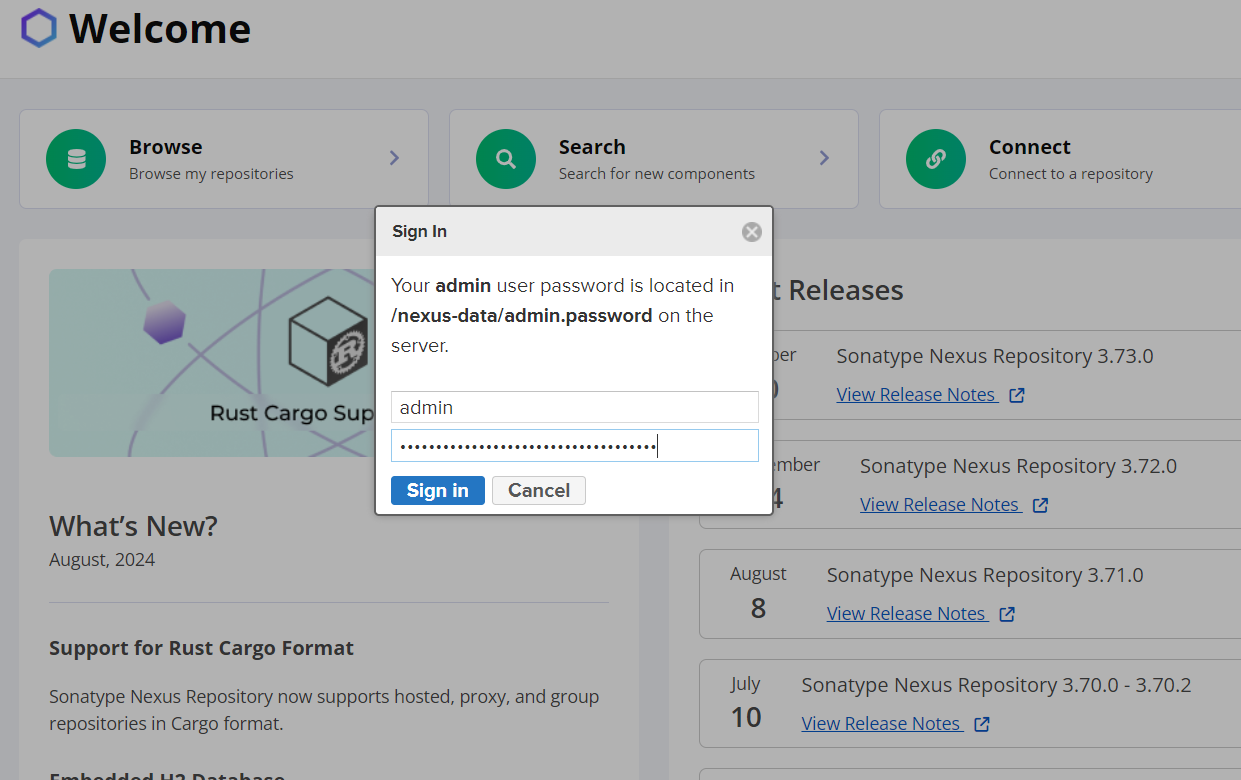
bash-4.4$ cd nexus3/

bash-4.4$ ls

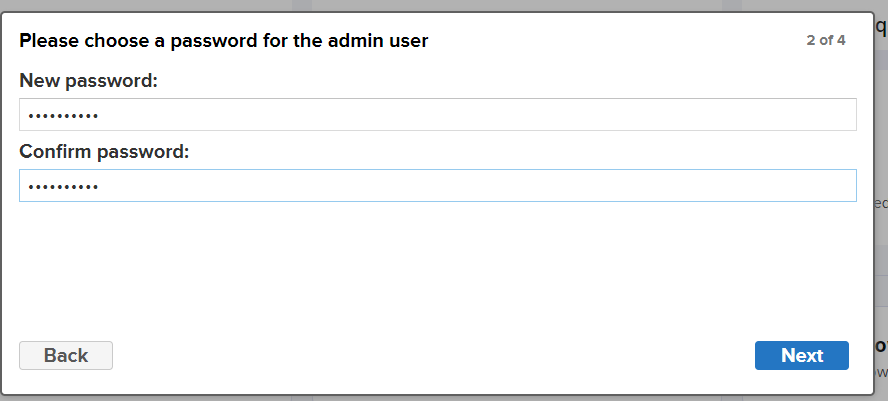
admin.password cache elasticsearch generated-bundles javaprefs karaf.pid lock port tmp

blobs db etc instances kar keystores log restore-from-backup

bash-4.4$ cat admin.password

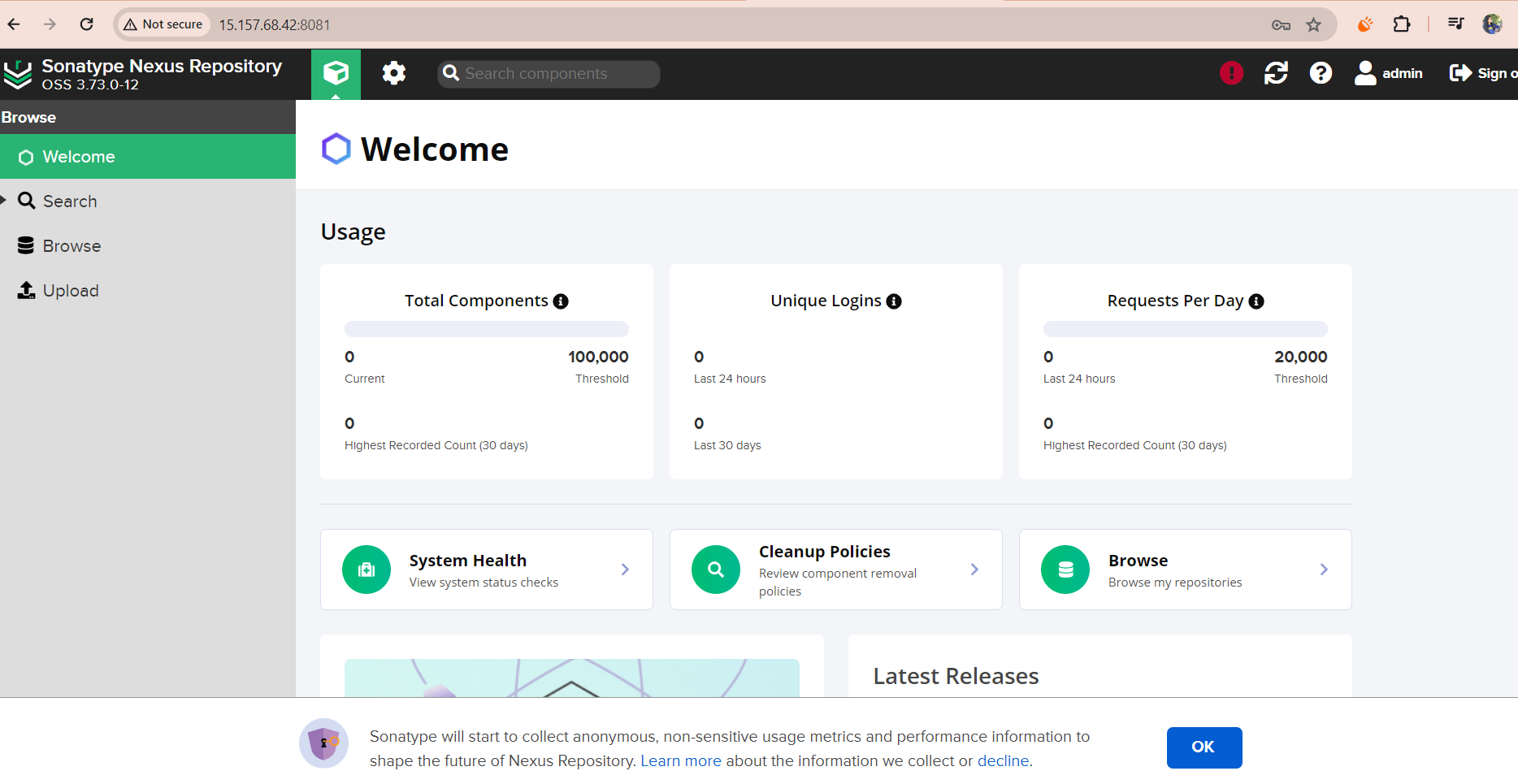
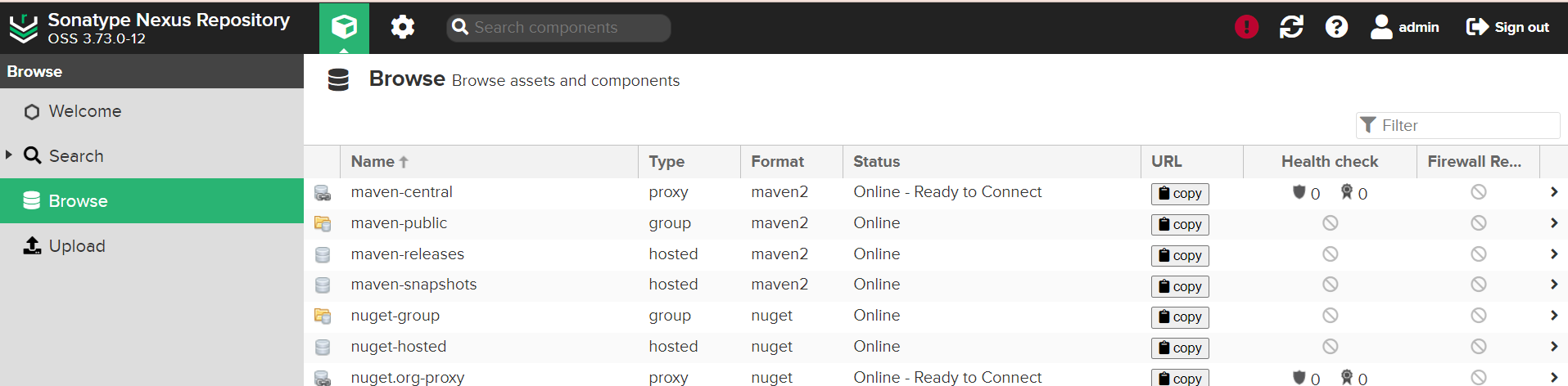


Set new password-



And select enable anonymous access (users to download components without access).

Here is the UI:

### Now set up Jenkins:

Pre-requisite is to install Java, select version 17 or above

sudo apt install openjdk-17-jre-headless –y

To install jenkins go to <https://www.jenkins.io/doc/book/installing/linux/#debianubuntu>

Copy the commands jenk.sh as:

#!/bin/bash

sudo apt install openjdk-17-jre-headless –y

#To download Jenkins

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \

https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \

https://pkg.jenkins.io/debian-stable binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null

sudo apt-get update

sudo apt-get install jenkins –y

Once Jenkins is installed next install docker using same docker script we used in sonar and nexus servers

vi dock.sh

Then sudo chmod +x dock.sh

./dock.sh

ubuntu@ip-172-31-18-47:~$ sudo chmod 666 /var/run/docker.sock

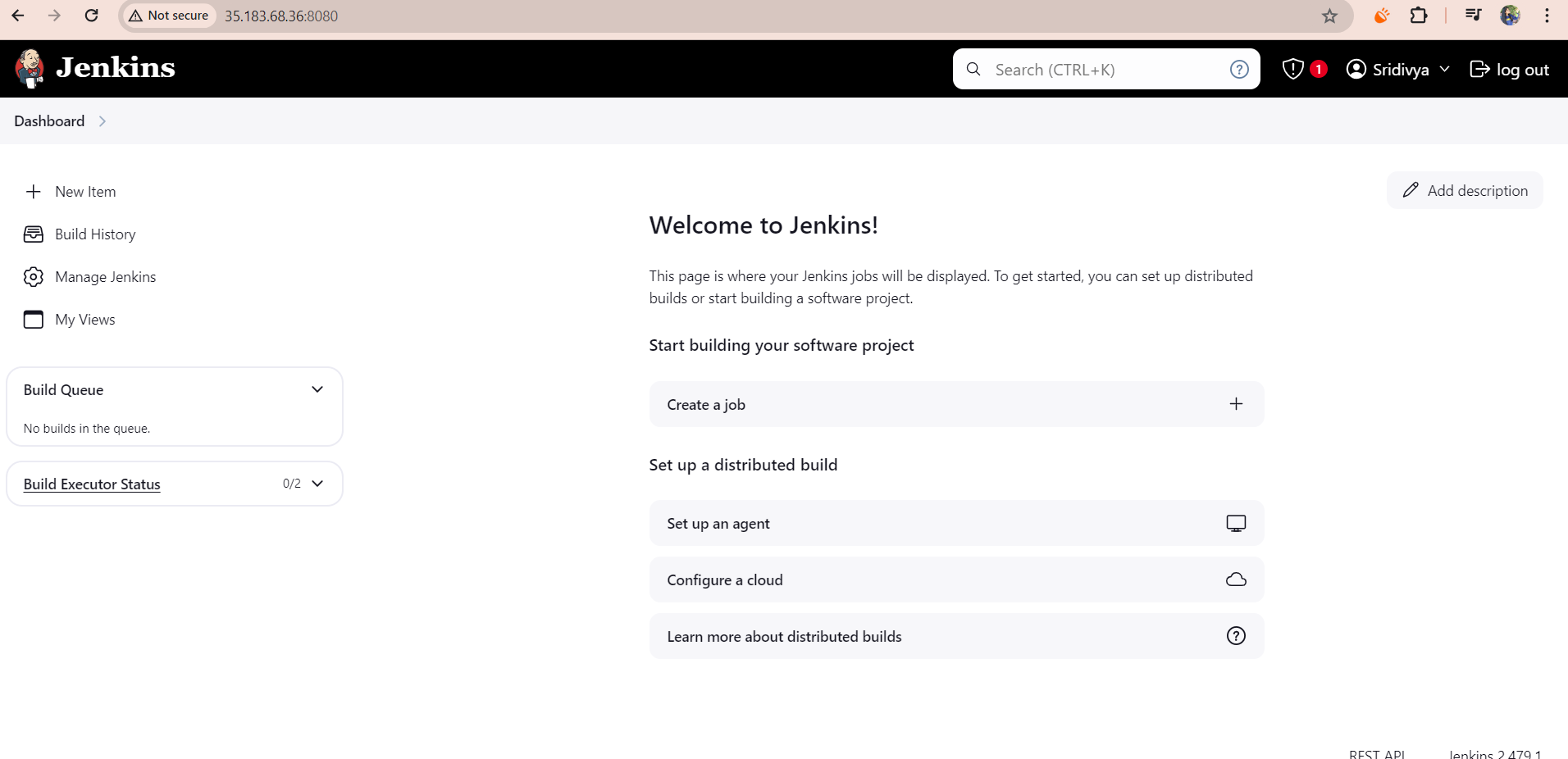
ubuntu@ip-172-31-18-47:~$ docker pull hello-world

Now for JENKIMS GUI: use Jenkins IP and default port is 8080

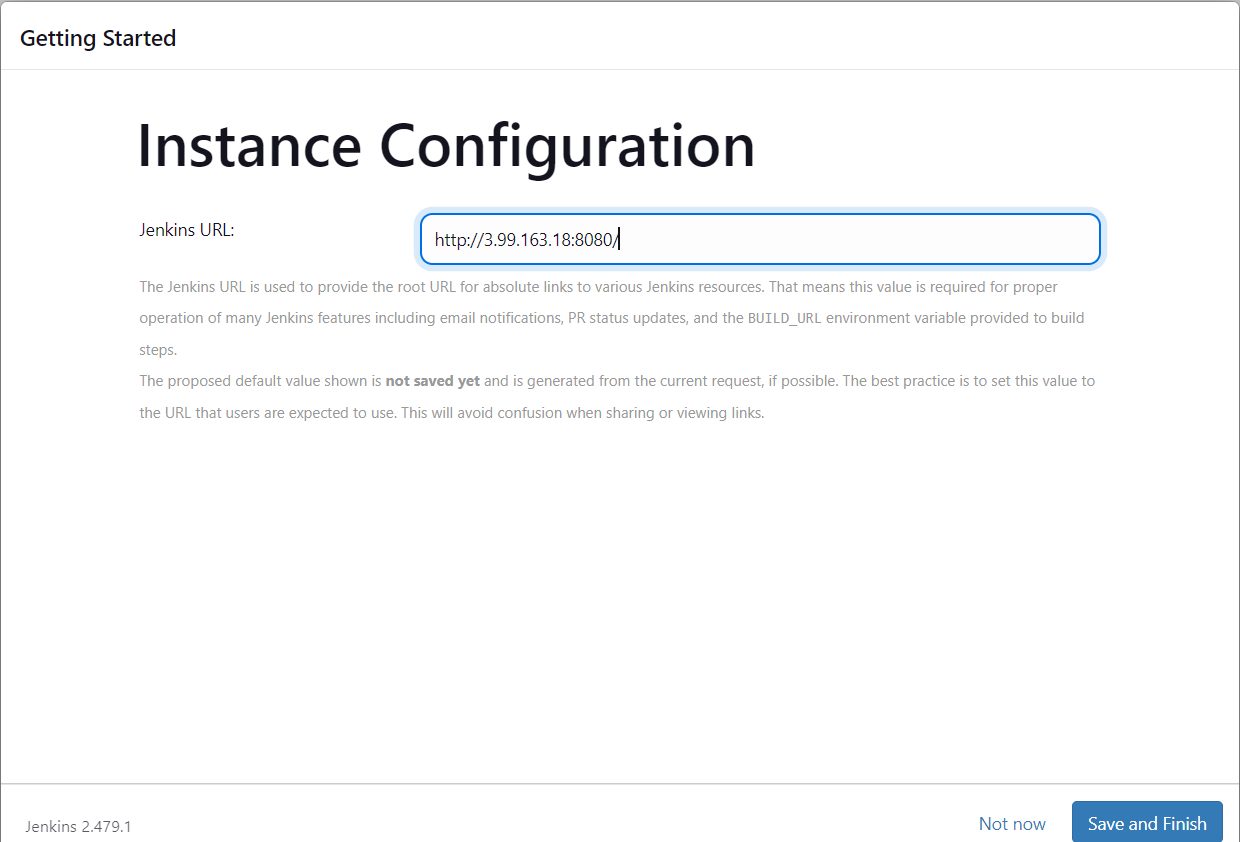
Use password from -sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Now Install suggested Plugins:

<http://35.183.68.36:8080/>



After installing plugins create username password-



## If VM’s are stopped then restart the docker too:

root@ip-172-31-27-58:/home/ubuntu# docker rm nexus

nexus

root@ip-172-31-27-58:/home/ubuntu# docker run -d --name nexus -p 8081:8081 sonatype/nexus3:latest

use IP:8081 in web browser to login and it asks password for which we should check under sonar-tyoe-nexus3-admin-passowrd

similarly for sonarQube:

root@ip-172-31-26-9:/home/ubuntu# docker rm sonar

sonar

root@ip-172-31-26-9:/home/ubuntu# docker run -d --name sonar -p 9000:9000 sonarqube:lts-community

1681f8c96db38c57b007ac0767808930475f5b4647114ffc67bc48e7d12c31ce

Use ip:9000 and creds: admin:admin