**Abstract Statement Project - 3**

You are expected to prepare design and implementation plan to onboard two tenants on k8s and OpenShift with sample application comprising of two microservices (PHP with PostgreSQL), use ArgoCD(on K8s) and S2I pipeline(on OpenShift) for deploying Sample project, use ELK stack for monitoring application with below flow.

* Use Terraform scripts to provision infrastructure such as Kubernetes cluster, EC2 instances **(Completed )**
* Use Ansible to configure provisioned Infrastructure **(Completed )**
* Use GitHub/Gitlab repos to store source code and pipeline files and Apply appropriate Branching and Security Policies**(Completed )**
* Docker File and the use docker-compose to automate image creation process - store images in Docker hub **( completed)**
* Use of helm charts for k8s package management **( Completed )**
* Deploy ingress Controller in K8s to access the applications **(Completed )**
* Launch Microservices using S2I concept of OpenShift **(Completed)**
* Use Persistent storage **(Completed)**
* Use Network policies to isolate applications **(Completed)**
* Monitor applications using ELK/EFK stack on OpenShift **(completed)**

**Team :3**

**Members – Anburaj, Gopi , Charles**

* Use Terraform scripts to provision infrastructure such as Kubernetes cluster, EC2 instances

Pre-requisites

For Terraform on ubuntu os

apt install python3-pip

apt install  python3-boto3

 apt install  python3-botocore

For Aws communication

 curl "<https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip>" -o "awscliv2.zip"

    apt install unzip

    unzip awscliv2.zip

  sudo ./aws/install

  aws configure ( supply keys)

root@ip-172-31-81-209:~/Team3# pwd

/root/Team3

root@ip-172-31-81-209:~/Team3# **cat mainec2.tf ( Ec2 instance create via terraform)**

variable "elb-names" {

type = list

default = ["Team3-kuber-master", "Team3-kuber-n1","Team3-Kuber-n2"]

}

variable "list" {

type = list

default = ["t2.medium","t2.micro","t2.micro"]

}

provider "aws" {

region = "us-east-1"

# version = "5.60.0"

# access\_key = "PUT-YOUR-ACCESS-KEY-HERE"

# secret\_key = "PUT-YOUR-SECRET-KEY-HERE"

}

resource "aws\_instance" "Team3-proj" {

ami = "ami-0e54eba7c51c234f6"

count= 3

instance\_type = var.list[count.index]

tags= {

Name= var.elb-names[count.index]

}

# Block device mapping for the EBS volume

root\_block\_device {

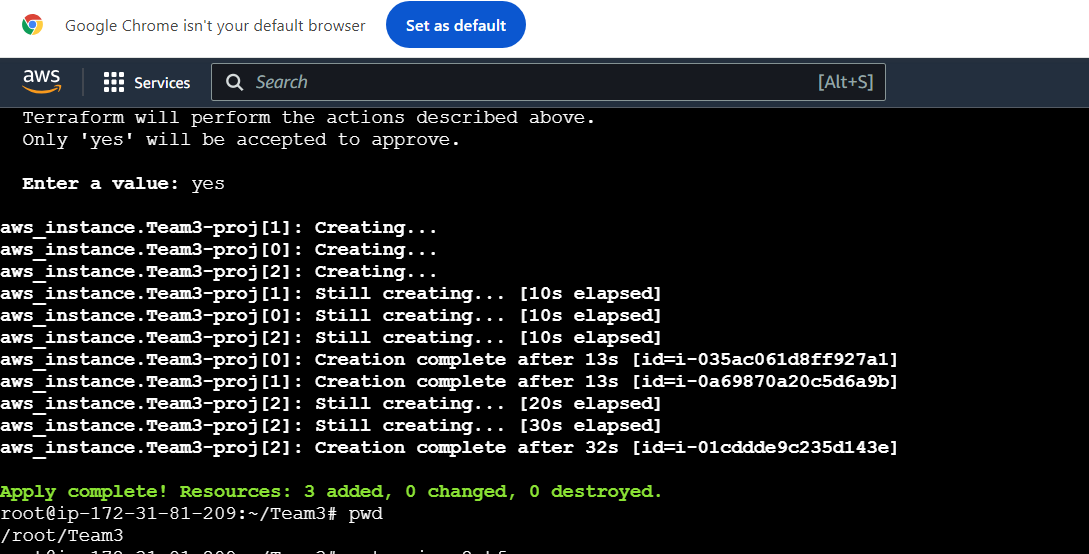
volume\_size = 20 # Size in GB

volume\_type = "gp2" # General Purpose SSD

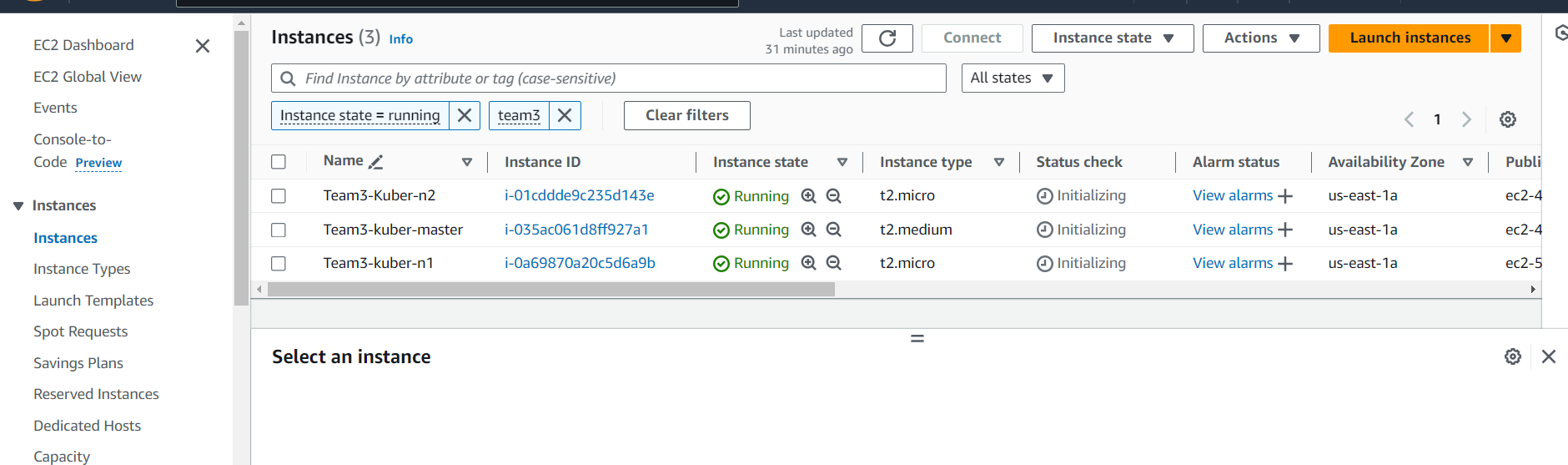
}

}

root@ip-172-31-81-209:~/Team3#



Aws Gui output of EC2



Project line2 : Kubernetes cluster setup through anisble

[**Installing Ansible on Ubuntu**](https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html#id6)[**ℑ**](https://docs.ansible.com/ansible/latest/installation_guide/installation_distros.html#installing-ansible-on-ubuntu)

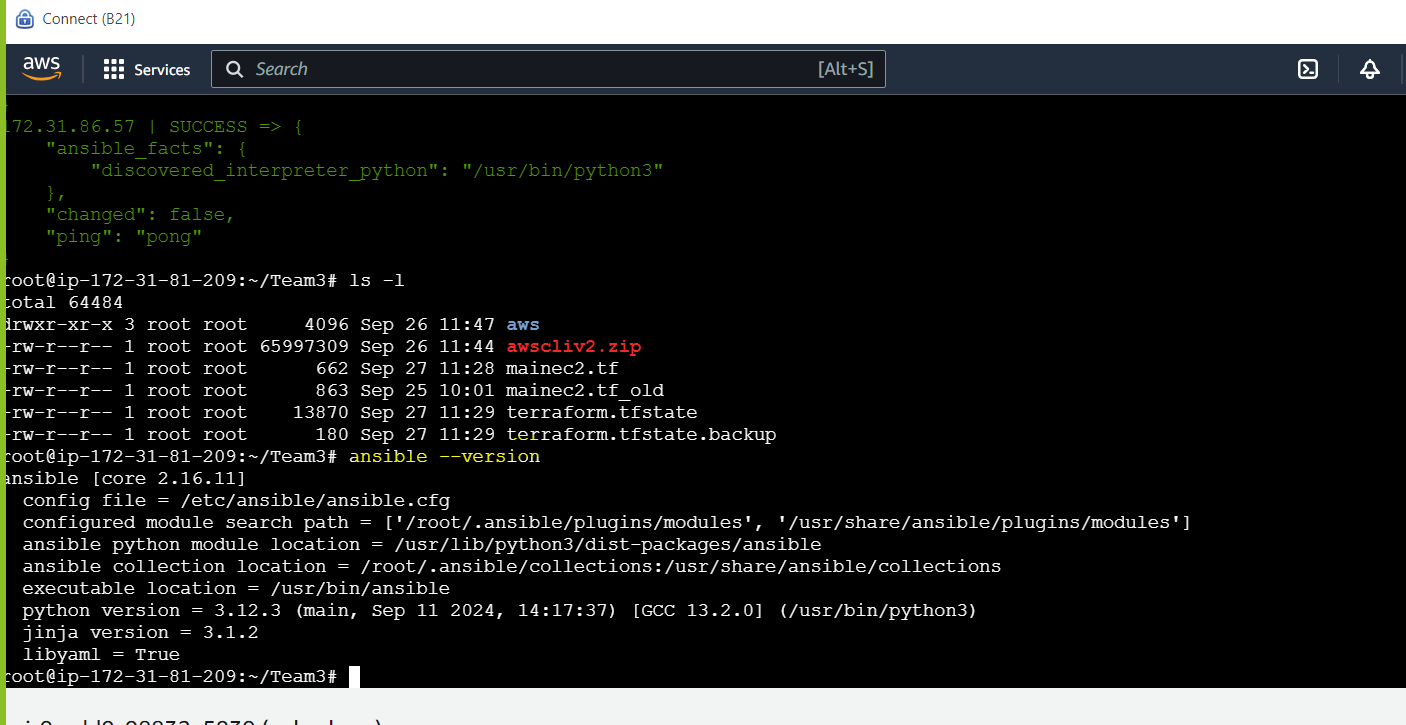
To configure the PPA on your system and install Ansible run these commands:

$ sudo apt update

$ sudo apt install software-properties-common

$ sudo add-apt-repository --yes --update ppa:ansible/ansible

$ sudo apt install ansible



**4. Configure SSH with Password Authentication on Managed Nodes**

SSH into the managed nodes (M1 and M2) and configure SSH password authentication.

**4.1. SSH into the Managed Nodes**

SSH into each managed node (M1 and M2) from your local machine or master node:

bash

Copy code

ssh ubuntu@<managed-node-ip>

**4.2. Enable SSH Password Authentication**

1. Open the SSH configuration file:

bash

Copy code

sudo vi /etc/ssh/sshd\_config

1. **Comment out the following line** (add # at the beginning):

bash

Copy code

# Include /etc/ssh/sshd\_config.d/\*.conf

1. **Change PasswordAuthentication to yes** to enable password-based authentication:

bash

Copy code

PasswordAuthentication yes

1. **Permit Root Login** (not recommended for security reasons, but enabled here for demo purposes):

bash

Copy code

PermitRootLogin yes

1. Save and exit the file.

**4.3. Set a Root Password**

1. Set a password for the root user:

bash

Copy code

sudo passwd root

1. Provide and confirm a password when prompted.

**4.4. Restart the SSH Service**

1. Restart the SSH service for the changes to take effect:

bash

Copy code

sudo systemctl restart ssh

**4.5. Test SSH Access with Password**

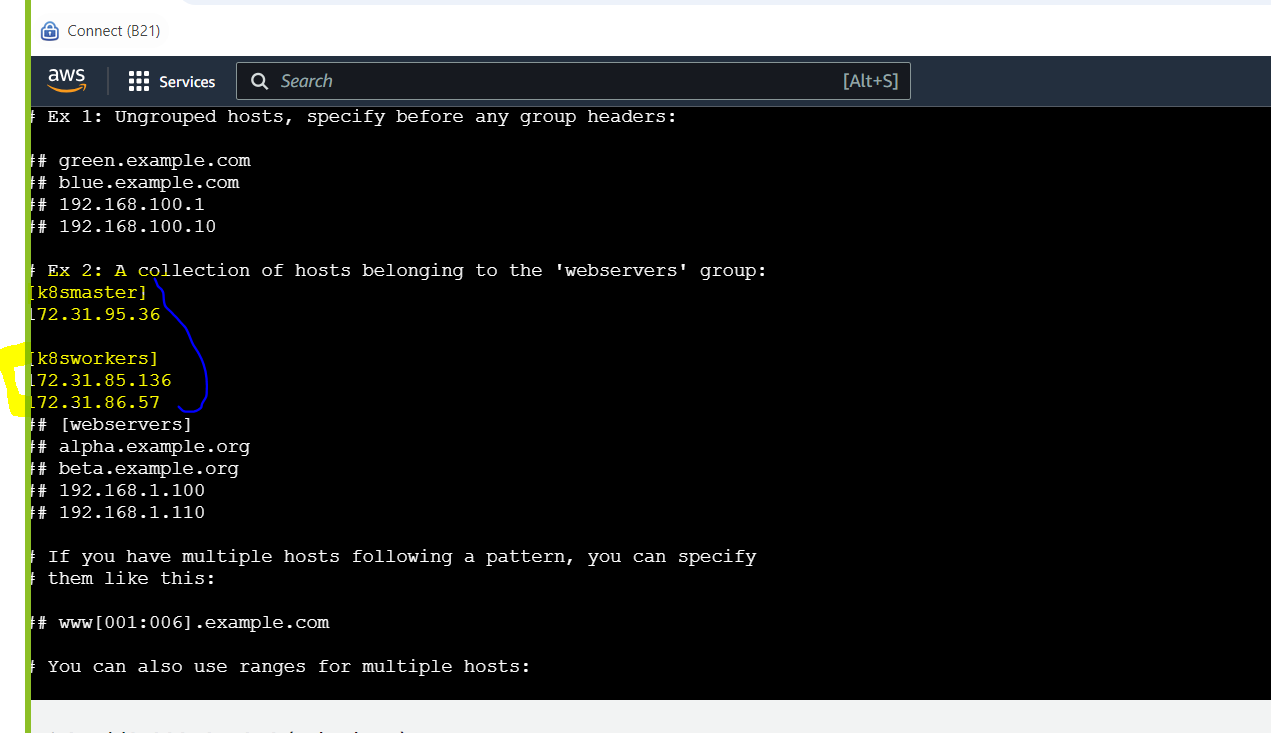
From the master node or your local machine, test SSH access using the root user and the password:

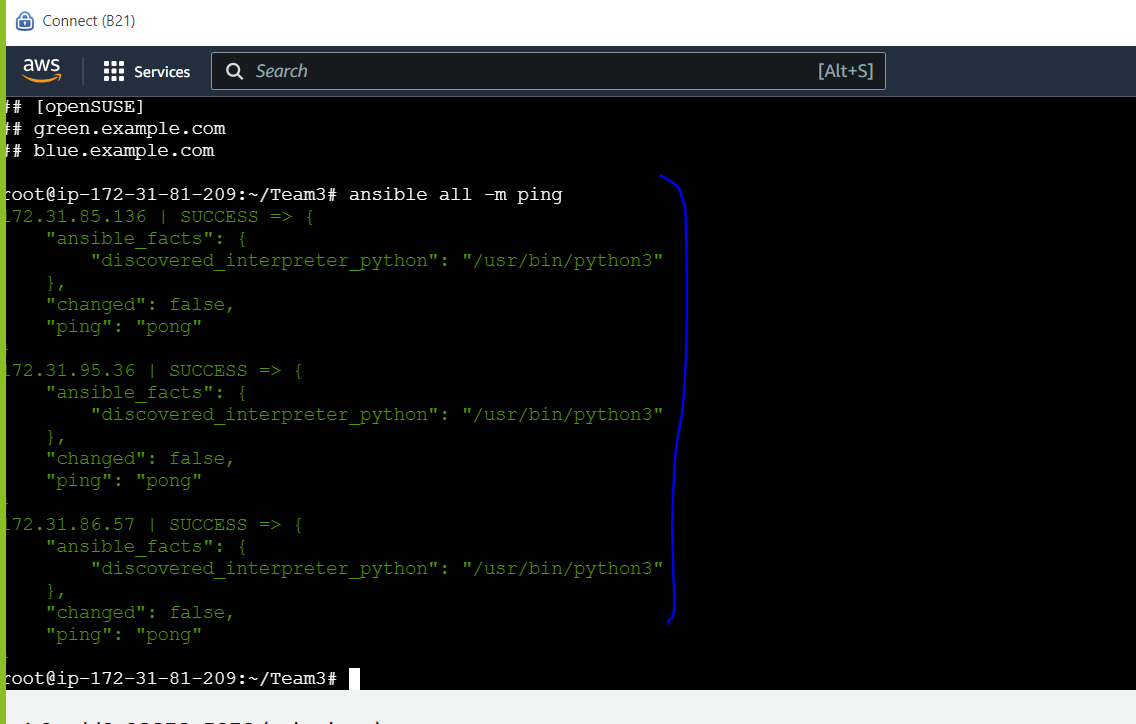
bash

Copy code

ssh root@<managed-node-ip>

Proof





Setup pipeline

Install Jenkins and helm chat in Kubernetes master

This is the Debian package repository of Jenkins to automate installation and upgrade. To use this repository, first add the key to your system (for the Weekly Release Line):

sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \  
   <https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key>  
   <https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key>

Then add a Jenkins apt repository entry:

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/nullUpdate your local package index, then finally install Jenkins (execute the commands one by one):

sudo apt-get update  
sudo apt-get install fontconfig openjdk-17-jre  
sudo apt-get install Jenkins

vi /usr/lib/systemd/system/jenkins.service

set as below in above files

root@master:~# grep -i root /usr/lib/systemd/system/jenkins.service

# $JENKINS\_WEBROOT.

**User=root**

**Group=root**

Environment="JENKINS\_WEBROOT=%C/jenkins/war"

root@master:~#

root@master:~# systemctl daemon-reload

root@master:~# systemctl restart Jenkins

root@master:~# kubectl get nodes

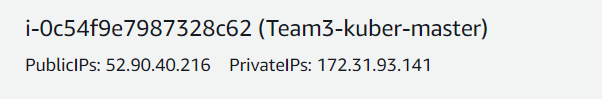
NAME STATUS ROLES AGE VERSION

master Ready control-plane 29h v1.29.9

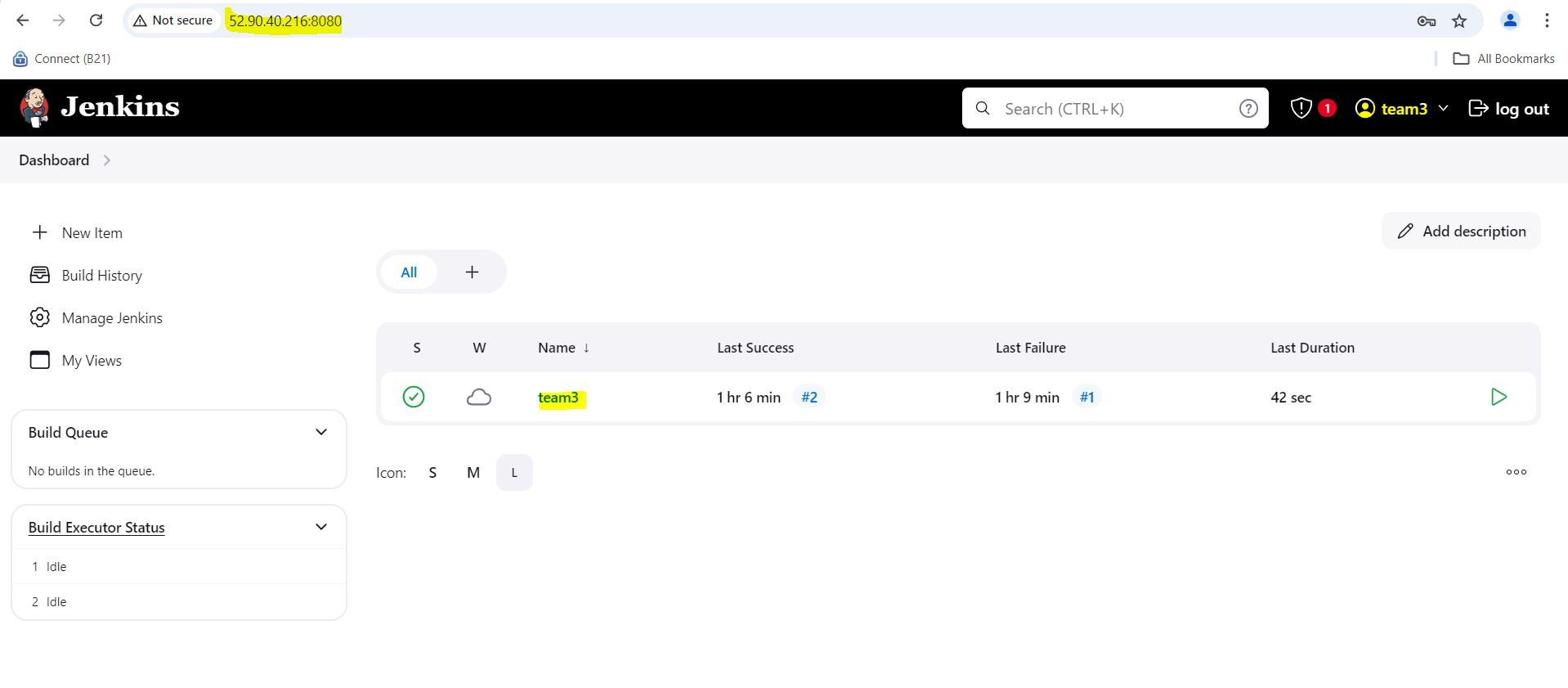
worker1 Ready <none> 29h v1.29.9

worker2 Ready <none> 29h v1.29.9

root@master:~#

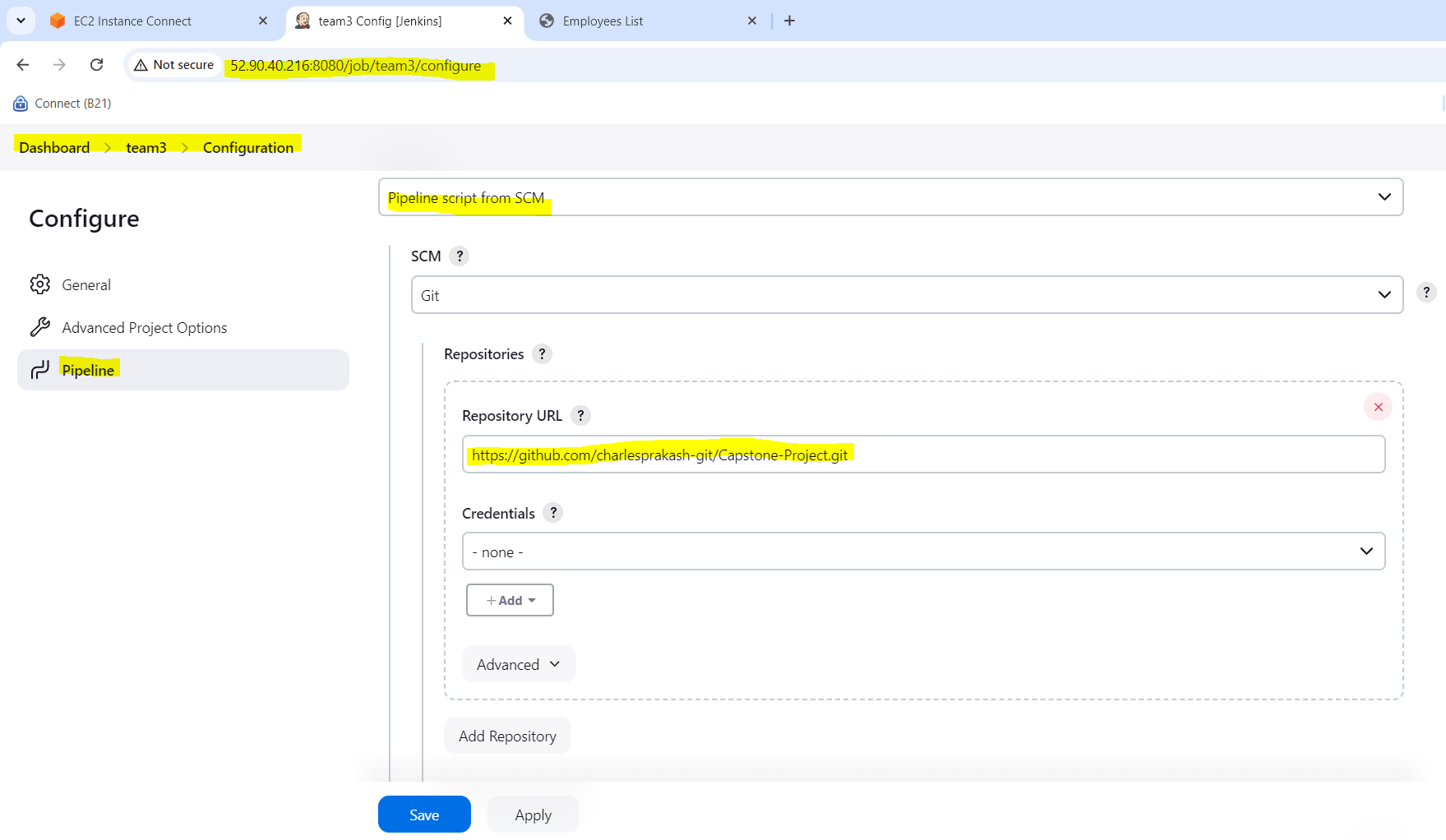


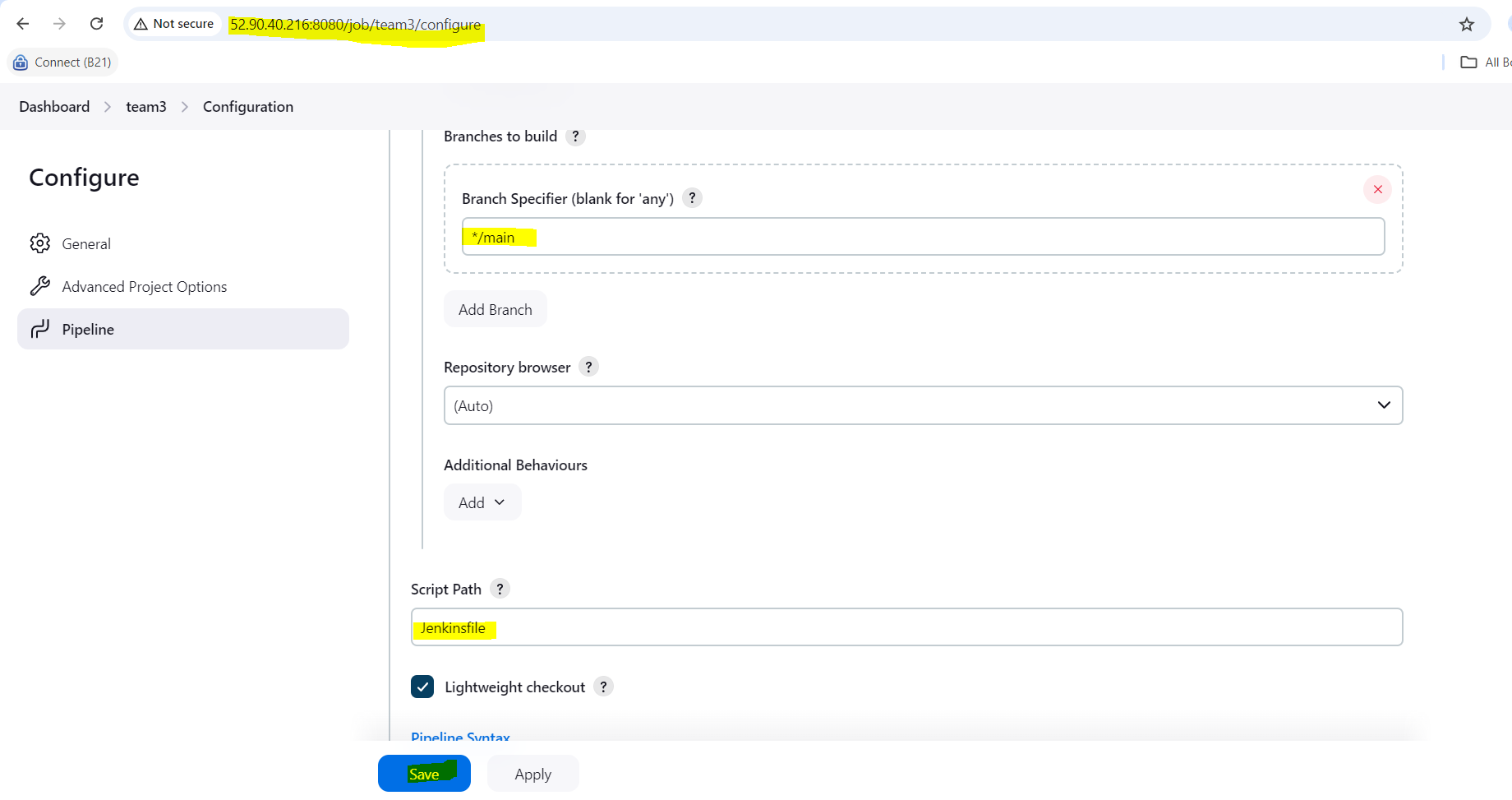
Jekin will work in publicip on 8080 port



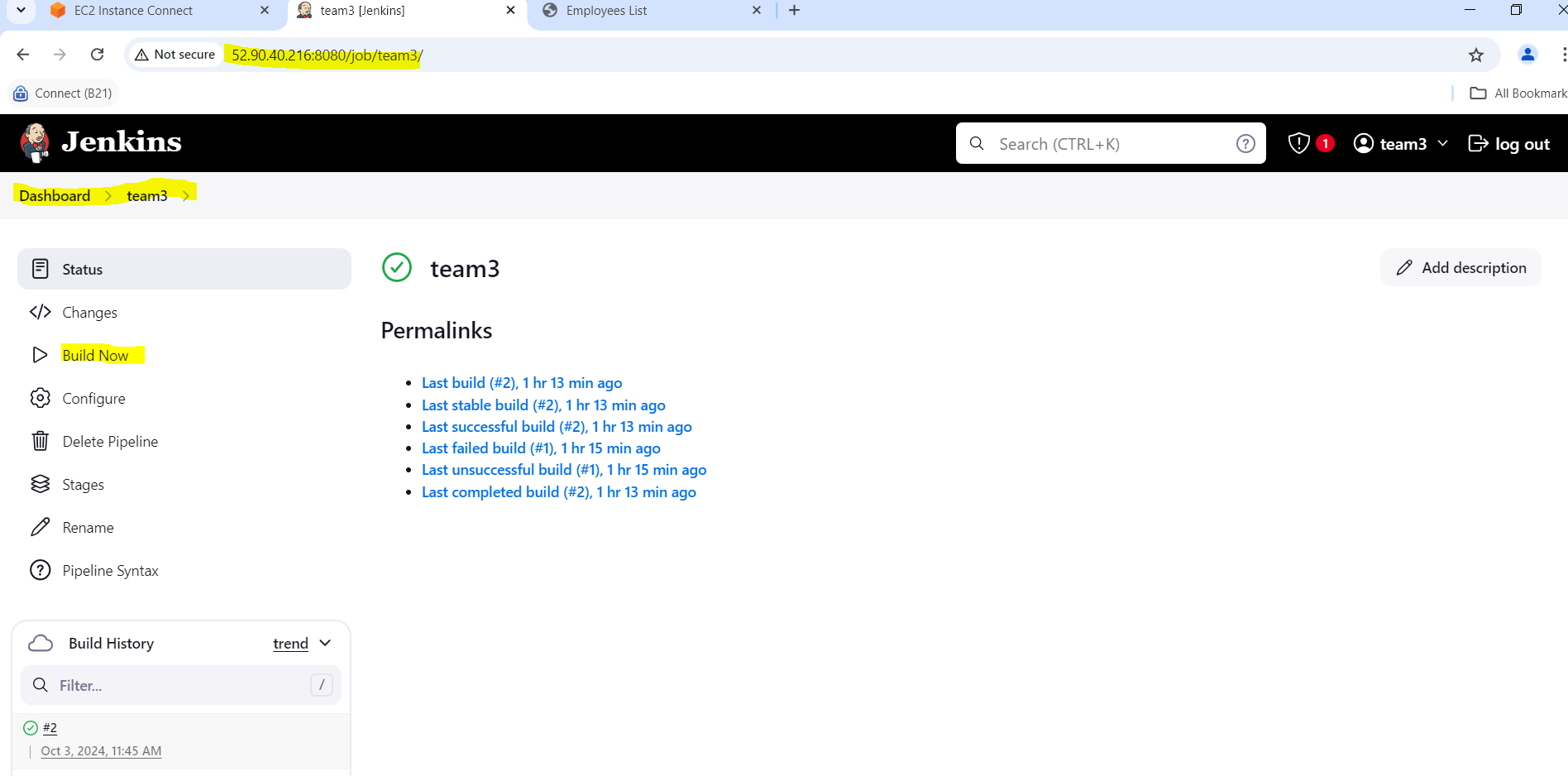
g

**Set pipeline in in Jenkins as show below**

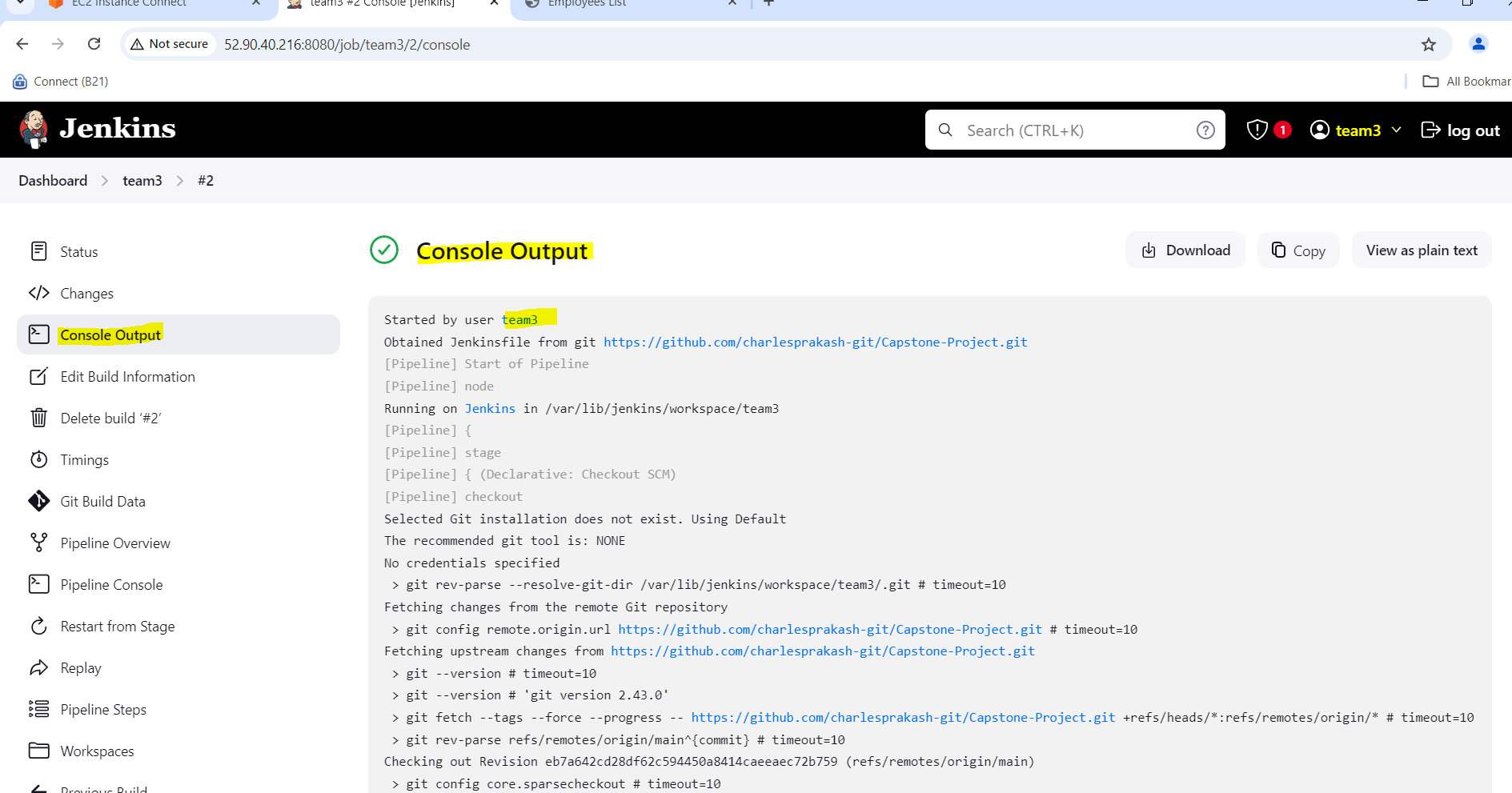


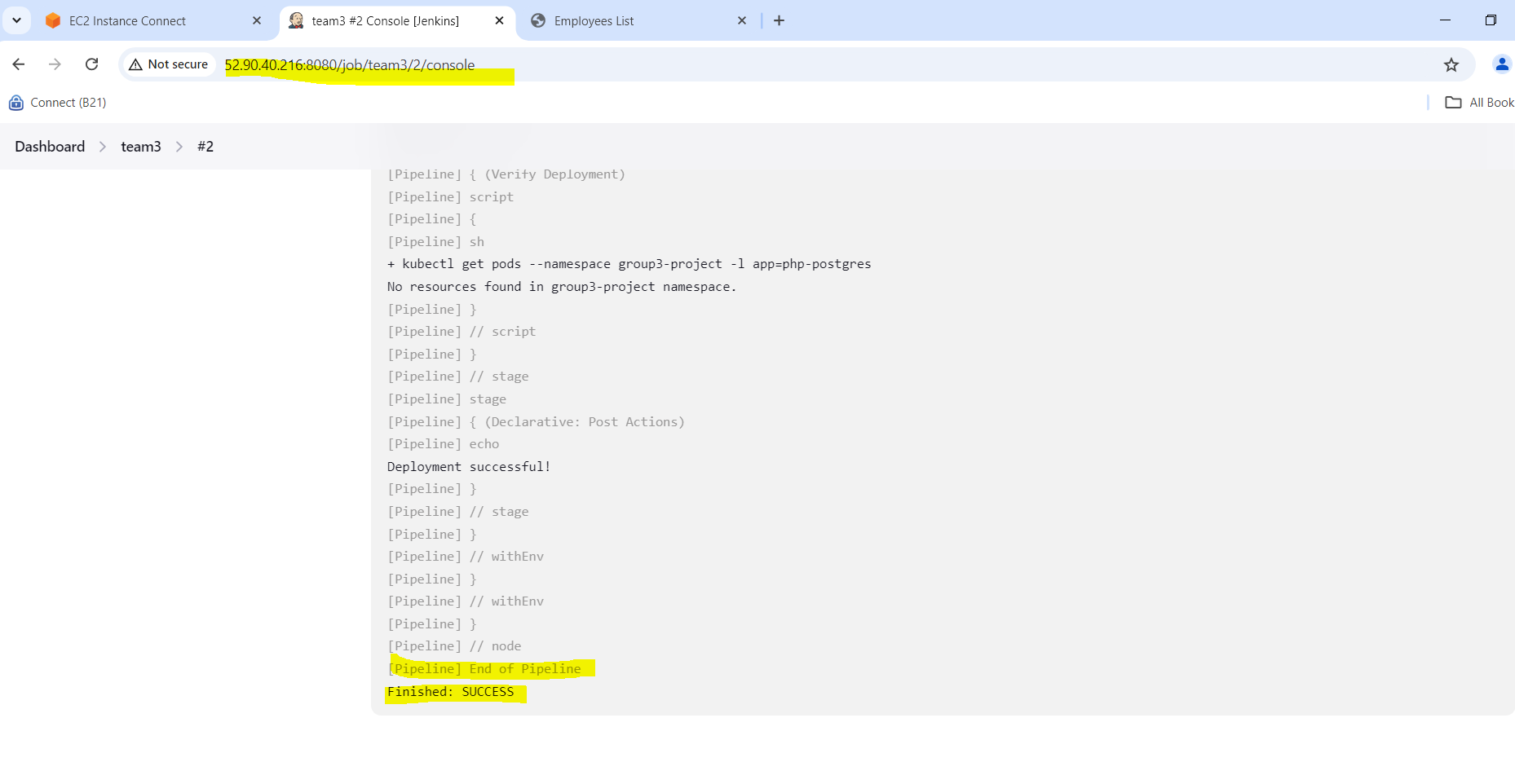


**Run build from pipeline**

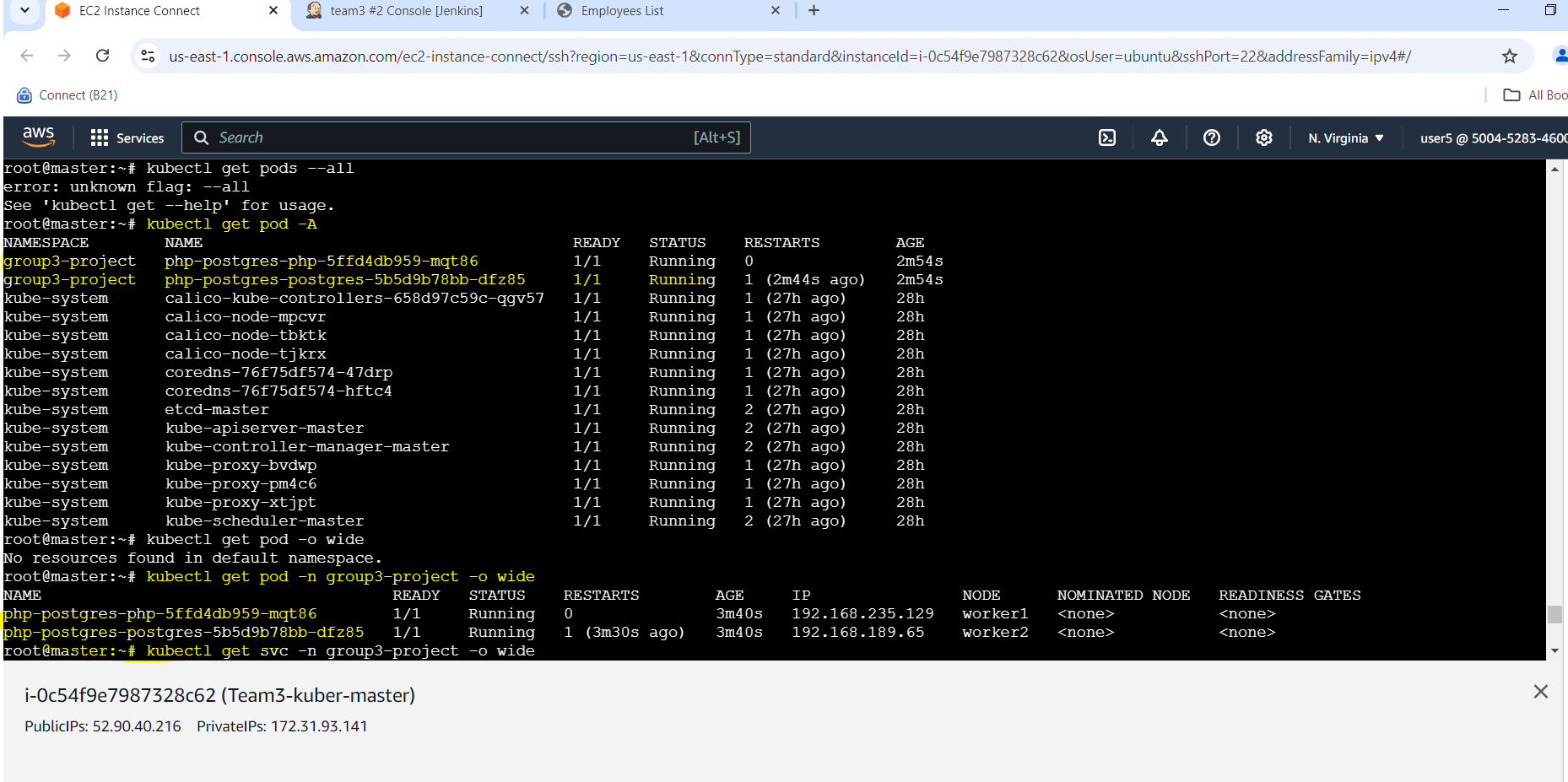


Check console ouput





**Check in Kubernetes master – pod and service successfully deployed**



**Check apps services**

root@master:~# kubectl get svc -n group3-project -o wide

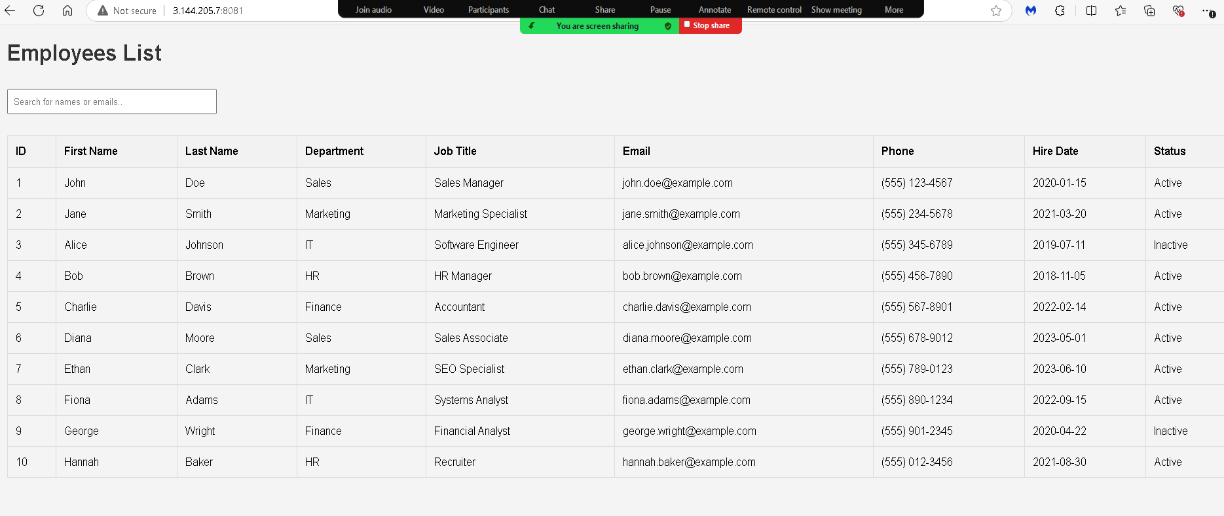
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE SELECTOR

php-postgres-php NodePort 10.102.2.156 <none> **80:32000/TCP** 3m56s app=php

php-postgres-postgres ClusterIP 10.110.32.159 <none> 5432/TCP 3m56s app=postgres

root@master:~#

Application deployed in public ip of Kubernetes master and exposed 32000 port and it working fine as below



**Pipeline(2) – ( OpenShift with image builder / Pv)**

• Launch Microservices using S2I concept of OpenShift

• Use Persistent storage

• Use Network policies to isolate applications

• Monitor applications using ELK/EFK stack on OpenShift

OpenShift installation steps:

**Step1**

**install OC**

wget https://mirror.openshift.com/pub/openshift-v4/x86\_64/clients/ocp/stable/openshift-client-linux.tar.gz

tar -xvzf openshift-client-linux.tar.gz

cp kubectl oc /usr/bin

**Step 2**

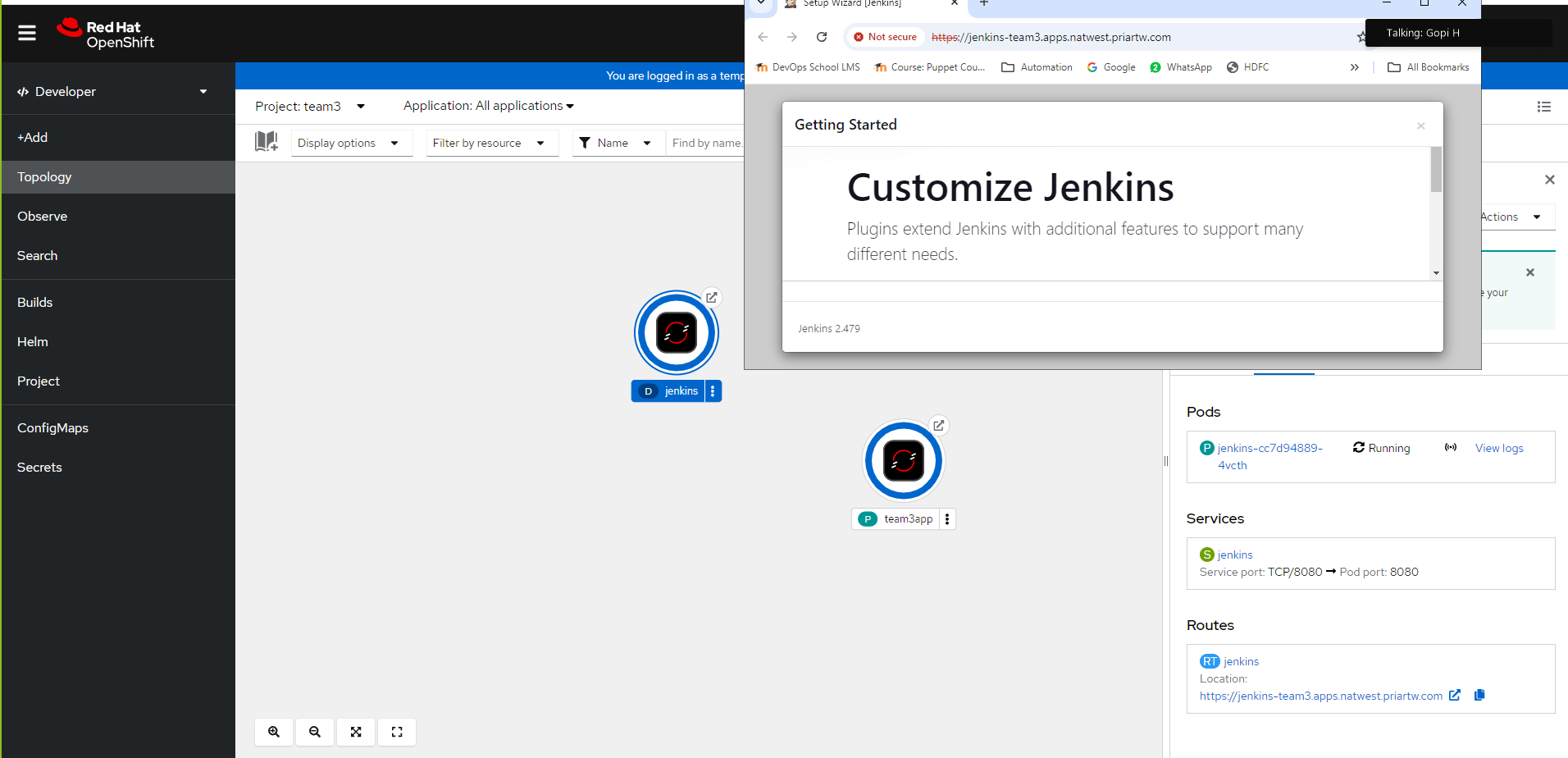
**oc new-project team3**

oc adm policy add-scc-to-user anyuid -z default -n team3

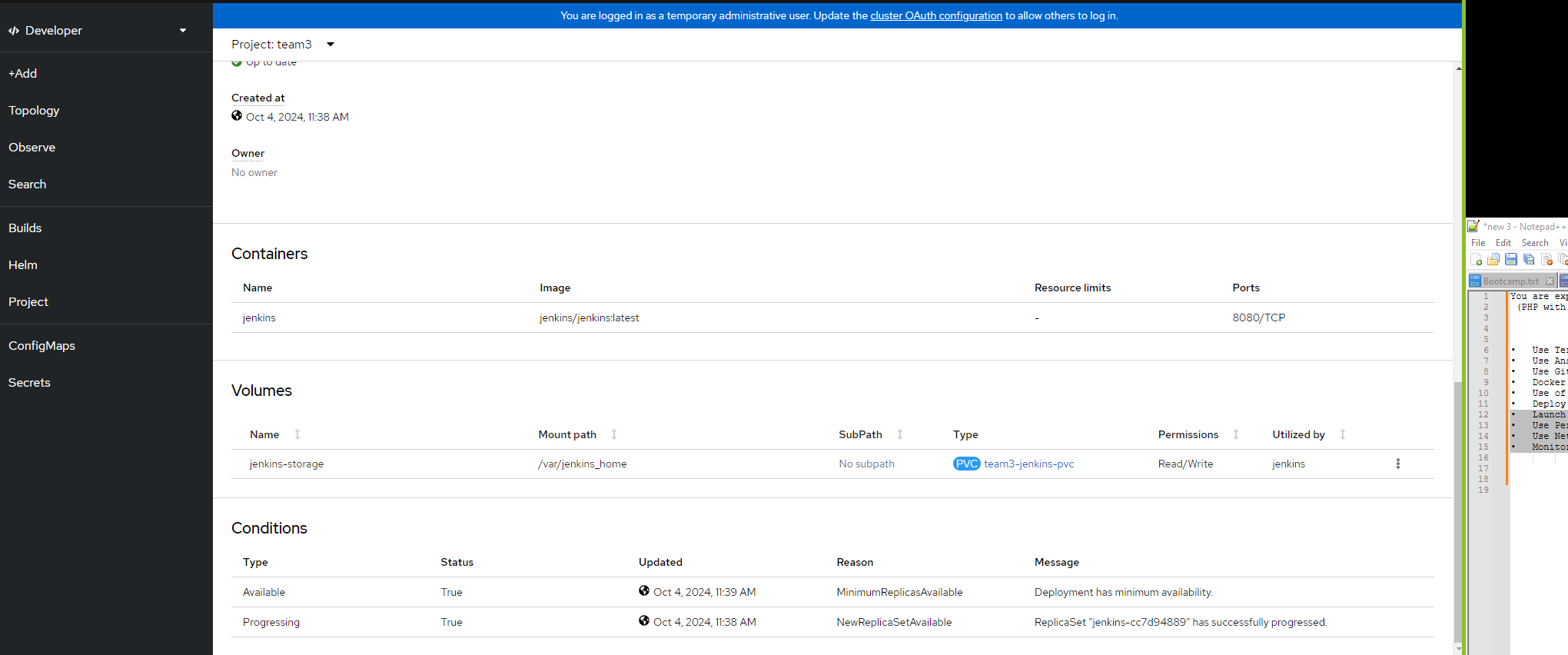
oc adm policy add-scc-to-user hostmount-anyuid -z default -n team3

oc adm policy add-scc-to-user privileged -z default -n team3

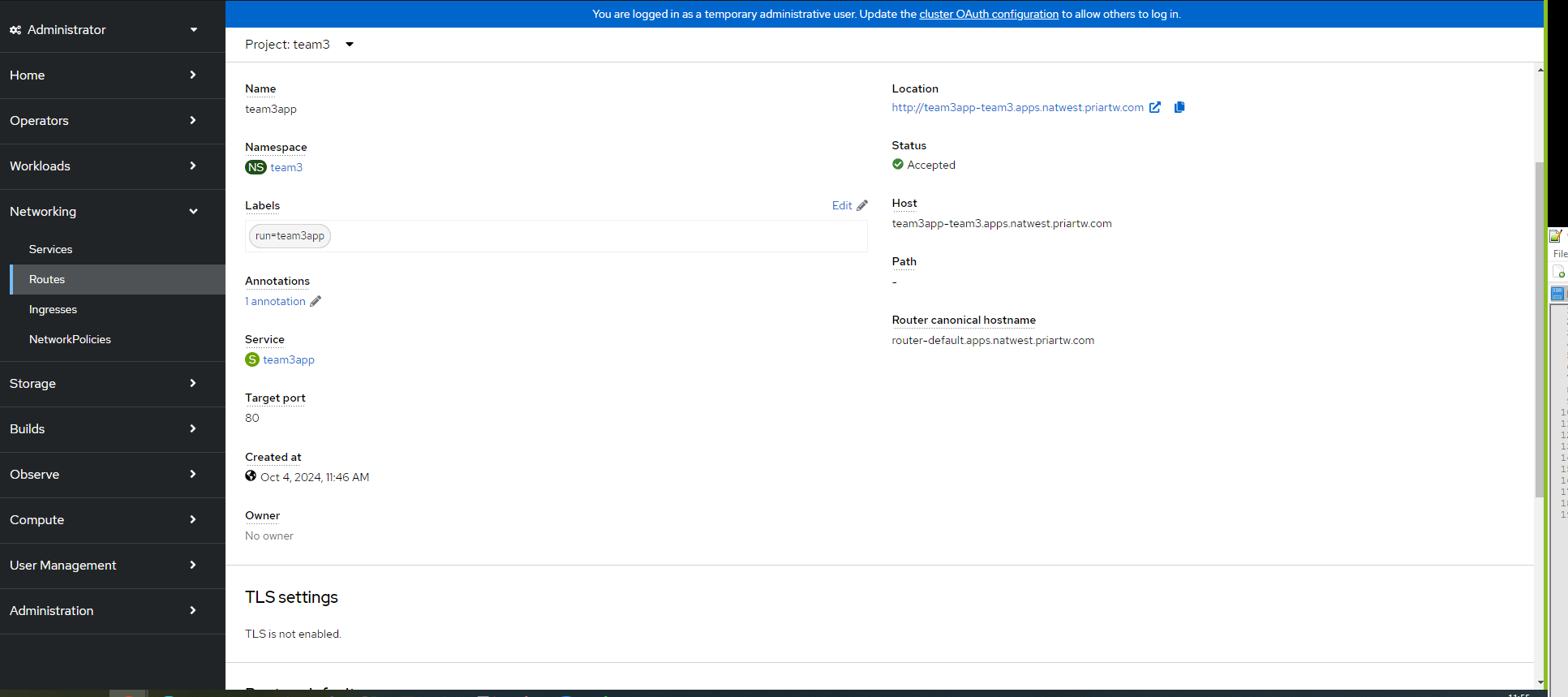
below screenshot shown Jenkins pod’s access over route.



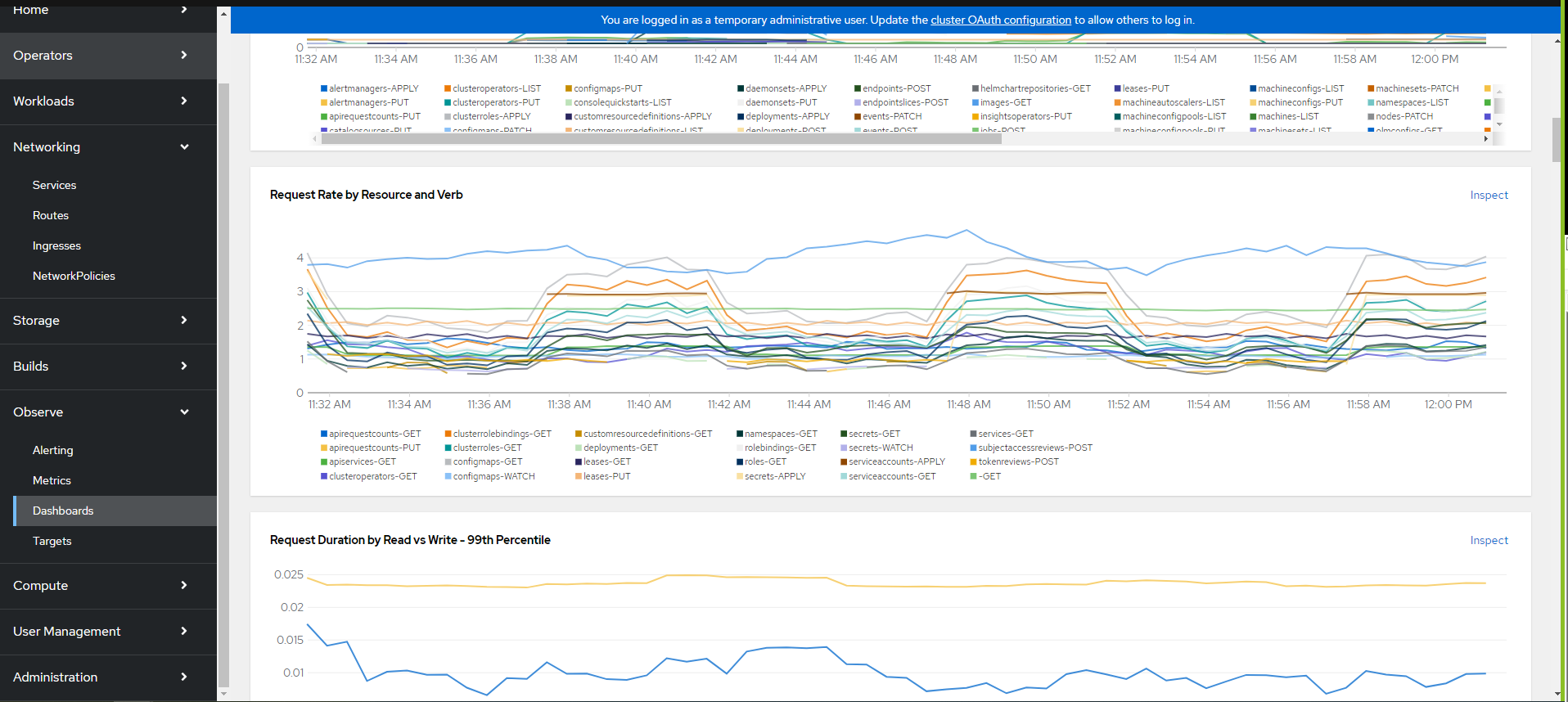
Created PV and PVCs for safe guard data



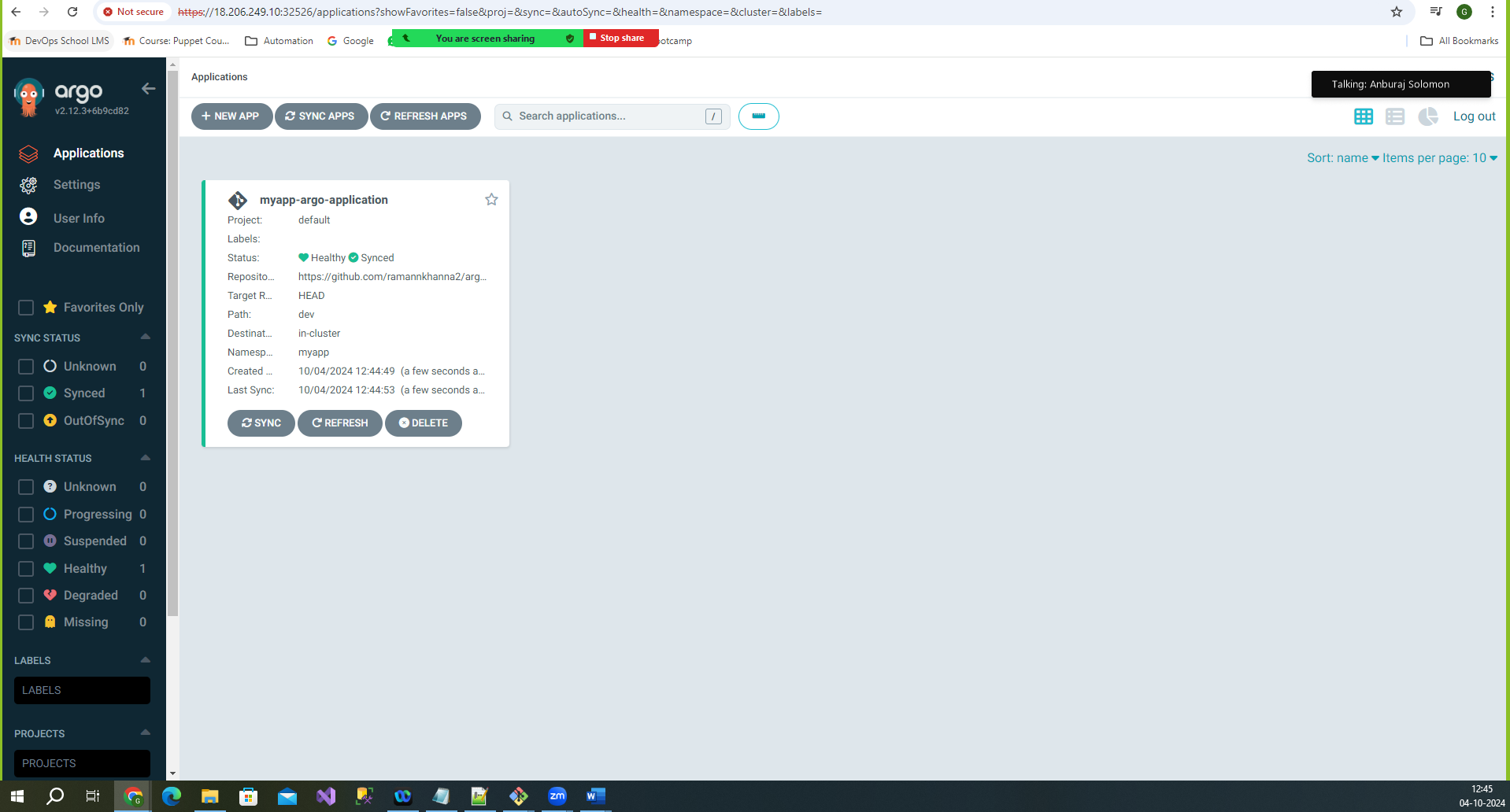
Use Network policies to isolate applications



Monitor applications using ELK/EFK stack on OpenShift



Argo CD setup on K8S for continuous deployment



Argo cd is monitoring latest modification and spun up five pod’s as shown below.

