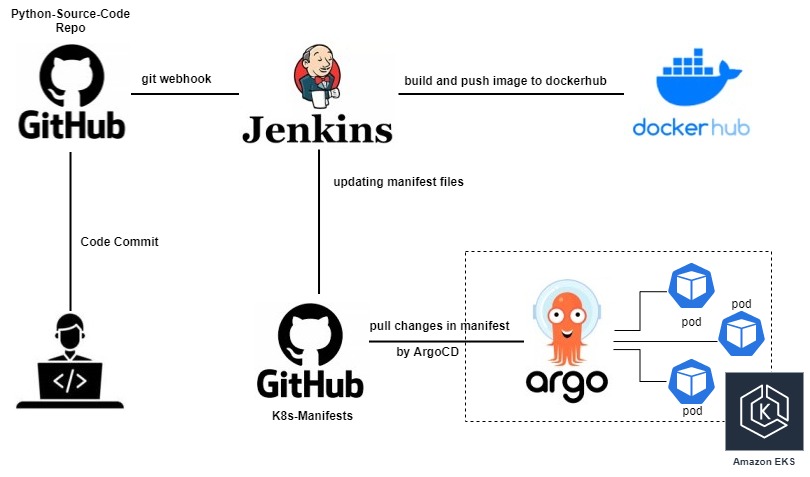
**How to Deploy a Flask Application to a EKS Cluster using Jenkins and ArgoCD.**

[](https://user-images.githubusercontent.com/119833411/242811691-f36ed79a-b1ea-49c0-ab8d-f3b40b4f00d2.jpg)

**Prerequisites.**

* **Jenkins Server up and running.**
* **Docker and git installed inside Jenkins Server.**
* **Docker Hub account.**
* **AWS Account.**
* **GitHub Account.**
* **EKS Cluster running**
* **Basic Understanding of Jenkins, Docker and Kubernetes.**

**Reference:-**

* **For Jenkins Step By Step installation:-** <https://gist.github.com/sampathshivakumar/54449ea95540ad0fd0f0cf44beb54ff9>
* **Fork the below two Repositories.**
* **Python-Source-Code Repository:** <https://github.com/Shivarajunayak/Python-Source-Code.git>
* **K8s-Manifests Repository:** <https://github.com/Shivarajunayak/K8s-Manifests.git>
* **Note:- Most of things are hardcoded in jenkins and deployment files,So Do Not change the following.**
* **Repository names that you fork**.
* **Jenkins job names**
* **docker-hub repo name**
* **Don't change anything expect your username,passwords.**

**After forking Repositories make these changes immediately.**

# In Python-Source-Code/Jenkinsfile

replace "dockersampath/packages" with "<your-dockerhub-username>/<repo-name>"

Note:- better don't change your repo name in docker hub.

# In K8s-Manifests/Jenkinsfile

replace bellow two line:-

sh "git config user.email sampathshivakumar@gmail.com" ---> sh "git config user.email <your-email-id>"

sh "git config user.name sampathshivakumar" ---> sh "git config user.name <git-user-name>"

sh "sed -i 's+dockersampath/packages.\*+dockersampath/packages:${DOCKERTAG}+g' deployment.yaml"

sh "sed -i 's+<your-docker-hub-usename>/packages.\*+<your-docker-hub-usename>/packages:${DOCKERTAG}+g' deployment.yaml"

# In K8s-Manifests/deployment.yaml

replace

- image: dockersampath/packages:5 ---> - image: <your-docker-hub-username>/packages:5

**Don't worry we start from scratch.**

**Jenkins installation.**

**Select Amazon Linux-2 AMI.**

**t2.micro**

**Enter the following command as root.**

# To download Jenkins repo.

wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/redhat-stable/jenkins.repo

# To import key.

rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key

# Install java.

amazon-linux-extras install java-openjdk11 -y

# Check Version of Java.

java --version

#Install Jenkins.

yum install jenkins

# You can enable the Jenkins service to start at boot with the command.

systemctl enable jenkins

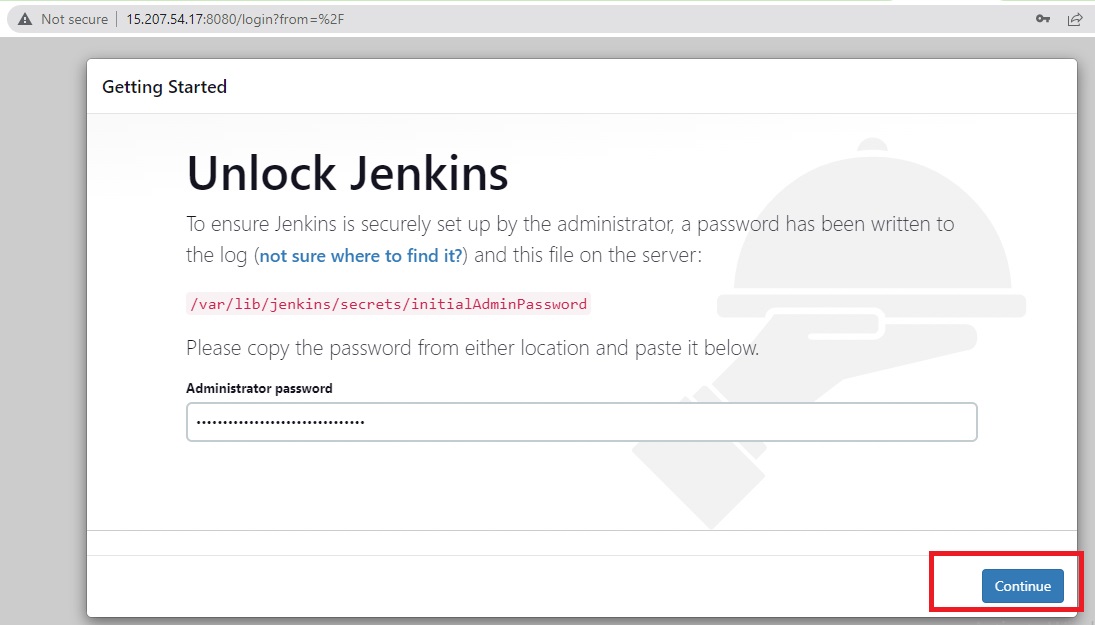
# You can start the Jenkins service with the command.

systemctl start jenkins

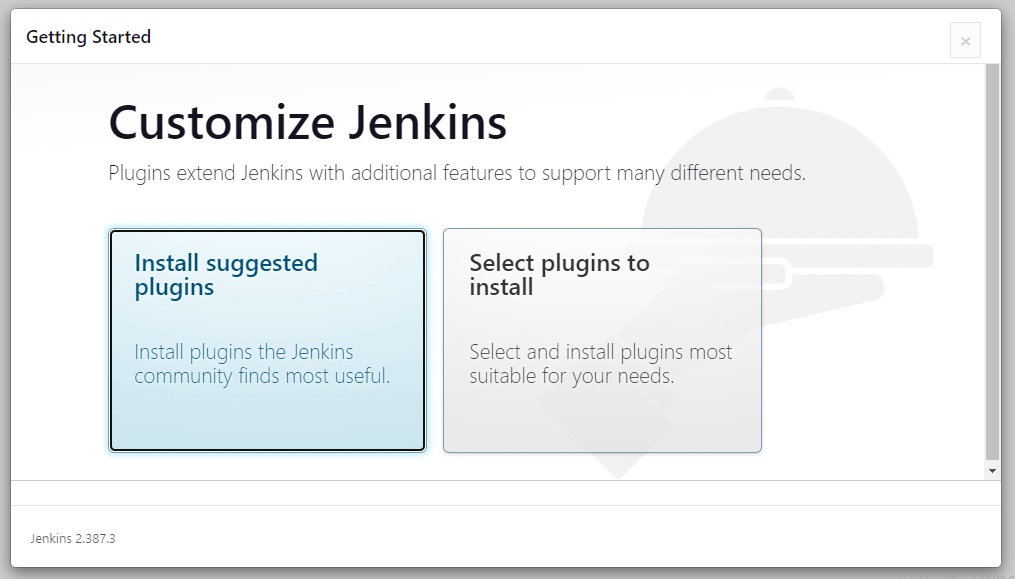
# You can check the status of the Jenkins service using the command.

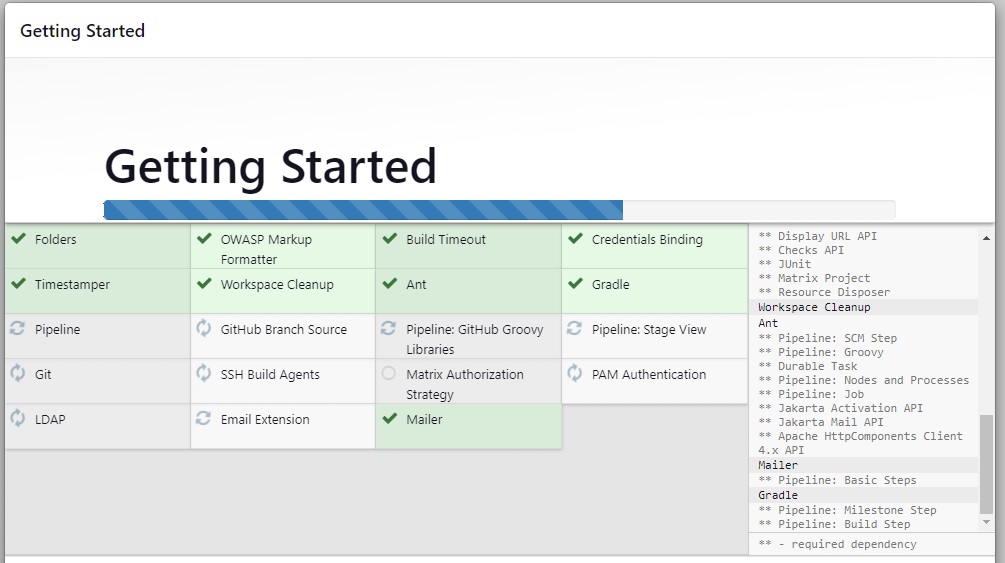
systemctl status jenkins

**Unlocking Jenkins.**

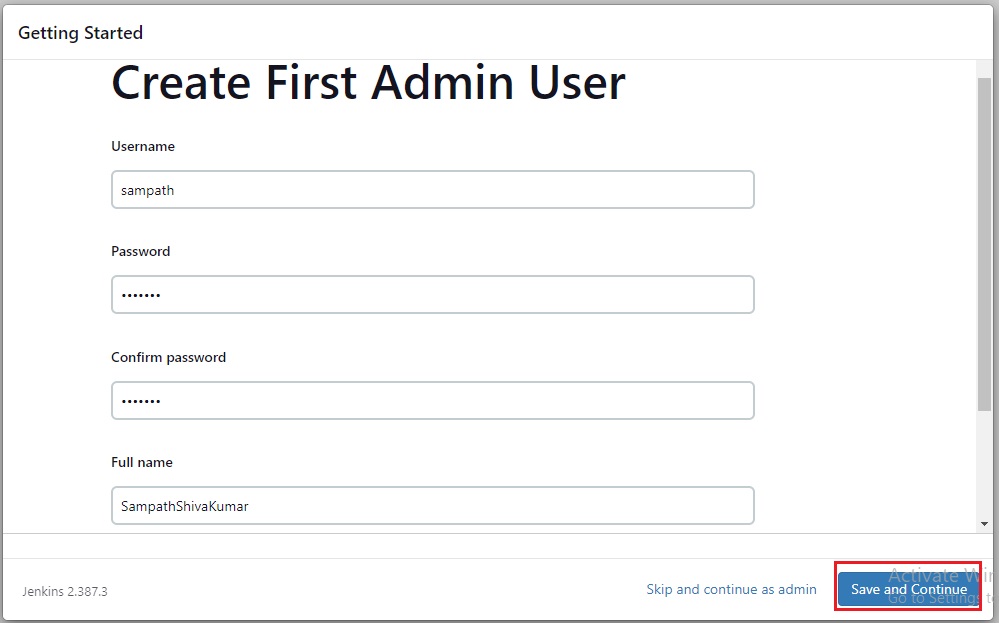
\*\*Browse to [http://localhost:8080](http://localhost:8080/) and unlock jenkins by inputing password. [](https://user-images.githubusercontent.com/119833411/242836388-6569bbc6-a713-4b6f-b889-51fec7476fab.jpg)

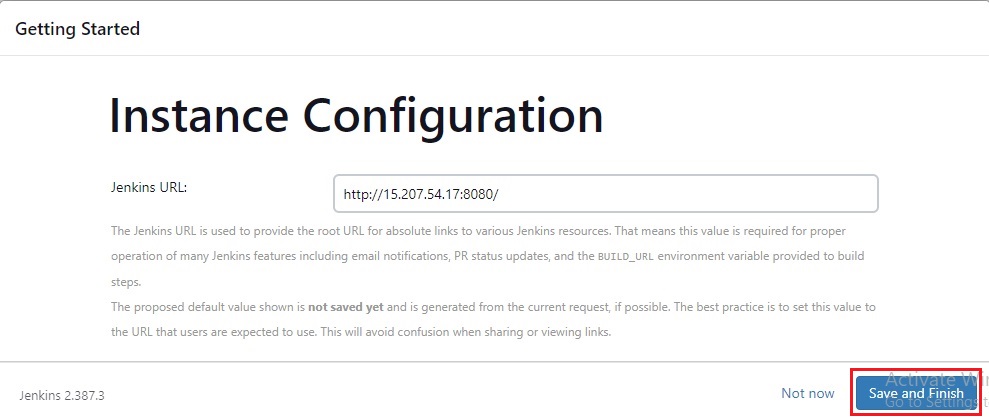
**Install selected plugins.**

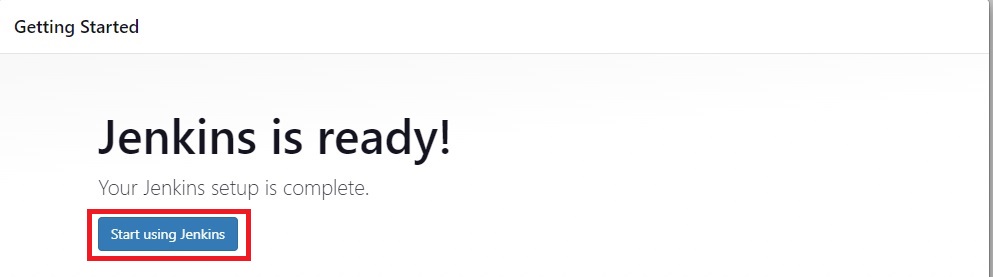
[](https://user-images.githubusercontent.com/119833411/242836593-b1014a6a-1438-40b5-981f-060de2c1db81.jpg)

[](https://user-images.githubusercontent.com/119833411/242836672-69c6f5c6-90bc-4de5-a130-849990ab0383.jpg)

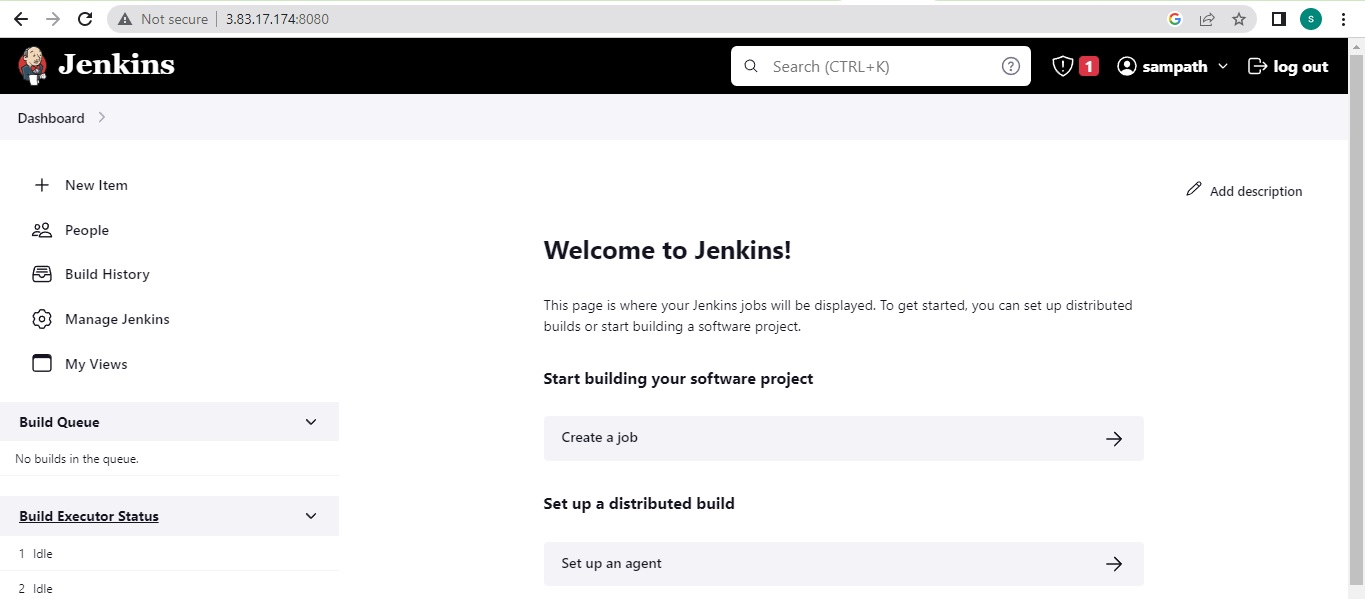
**Create Username and Password.**

[](https://user-images.githubusercontent.com/119833411/242836880-28499db9-f69a-4d5d-b418-0352344dc2a4.jpg)

[](https://user-images.githubusercontent.com/119833411/242836948-577c3974-c1e5-4d42-ac74-980802ef53aa.jpg)

[](https://user-images.githubusercontent.com/119833411/242836981-c83b2869-187e-472b-9385-91f5bdf198e3.jpg)

**You should see Jenkins Dashboard.**

[](https://user-images.githubusercontent.com/119833411/242839563-a122c57f-2a91-4251-a603-29df52b41fd0.jpg)

**Now let's download the Docker in the same server.**

# Become root user

sudo su -

# Apply updates

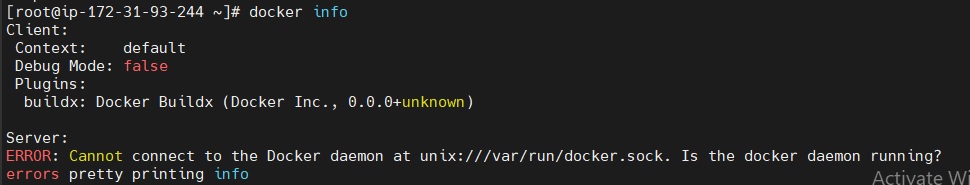
yum update -y

# Install Docker

yum install docker -y

# Let’s check the version and info of the docker

docker info

[](https://user-images.githubusercontent.com/119833411/242841711-428ded15-6fa6-4aa1-90fe-16c9e6e279c0.jpg)

# Enable Docker during boot time

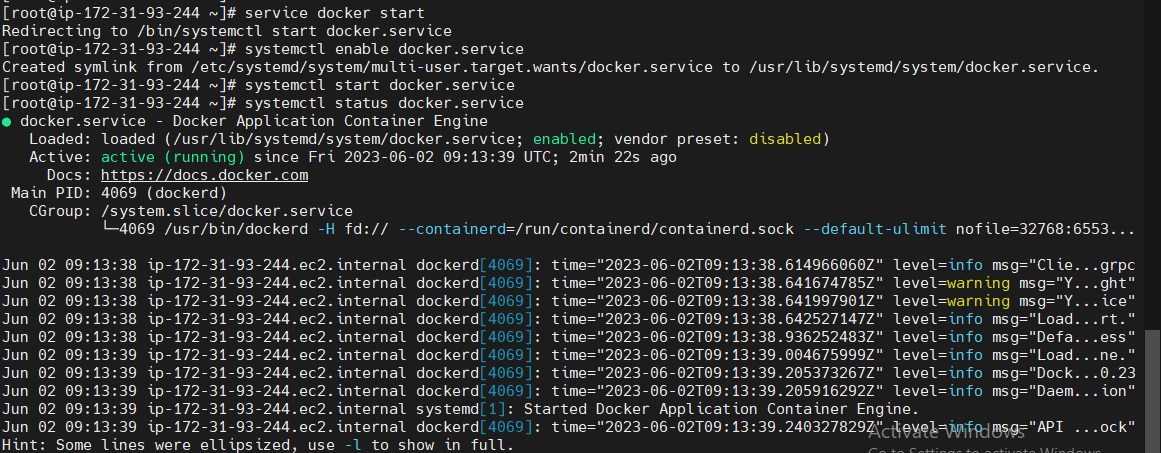
systemctl enable docker.service

# Start Docker

systemctl start docker.service

# Check the status of Docker

systemctl status docker.service

**You should see** [](https://user-images.githubusercontent.com/119833411/242842985-e1e5ee70-608a-41d5-a02c-f151bce7a6b2.jpg)

**Commands to control the docker service**

sudo systemctl start docker.service --> To start the service

sudo systemctl stop docker.service --> To stop the service

sudo systemctl restart docker.service --> To restart the service

sudo systemctl status docker.service --> To get the service status

**Now both Jenkins and Docker are Installed in our server.**

**Install Git also in the same server.**

# Install git

yum install git -y

# Check the version of git

git --version

**Integrate Docker with Jenkins.**

**You have to run docker commands using jenkins user, while runing jenkins job.**

# Add Jenkins user to Docker Group.

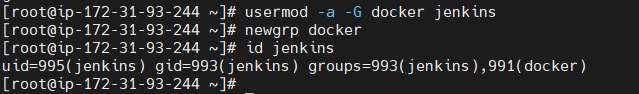
usermod -a -G docker jenkins

# Reload a Linux user's group assignments to docker

newgrp docker

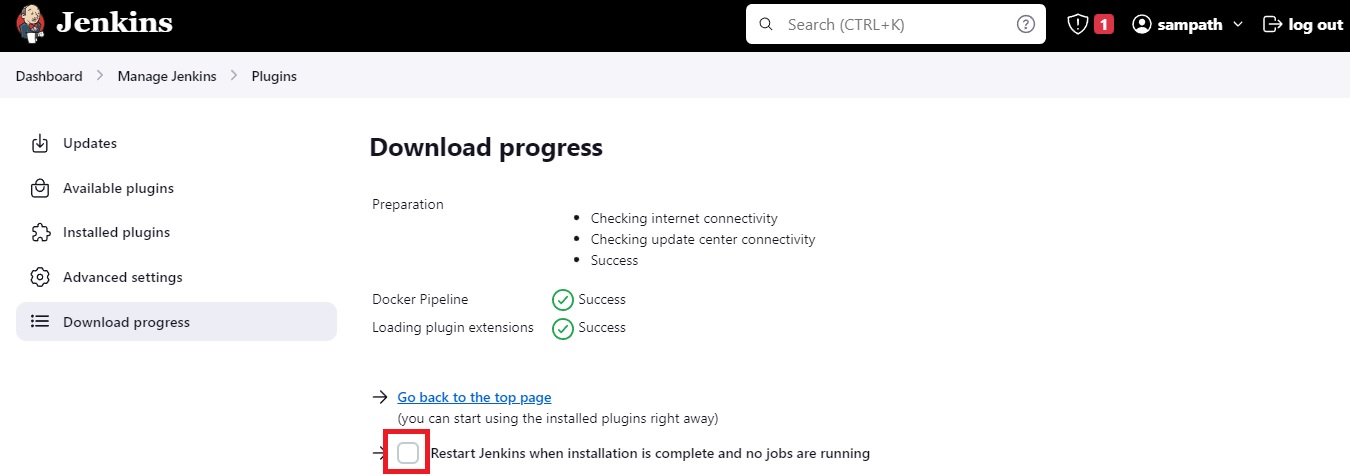
# Check the user id. to see group we added

id jenkins

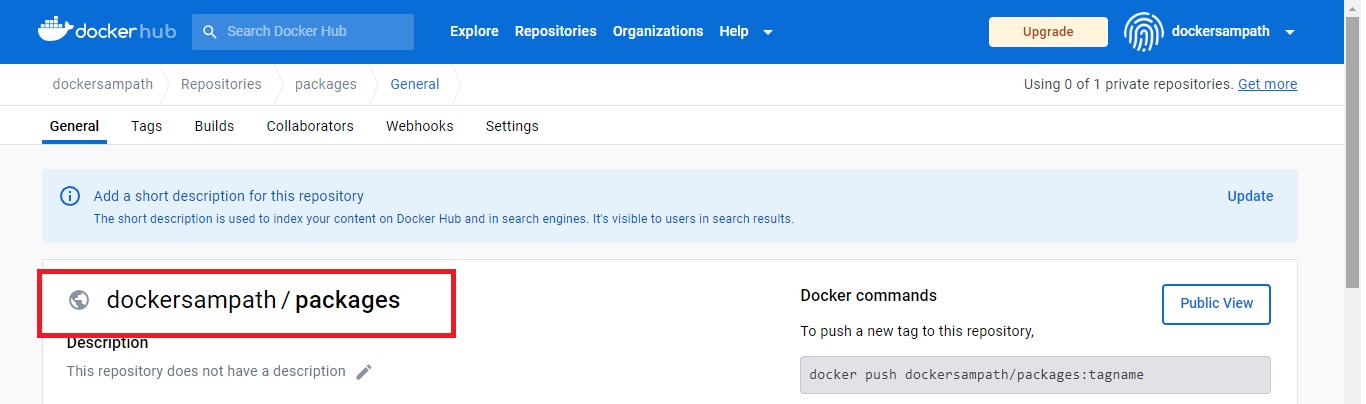
[](https://user-images.githubusercontent.com/119833411/242848265-38db7258-9009-437d-8bd0-890aa89b929d.jpg)

**Install the following plugins in Jenkins using Jenkis Dashboard.**

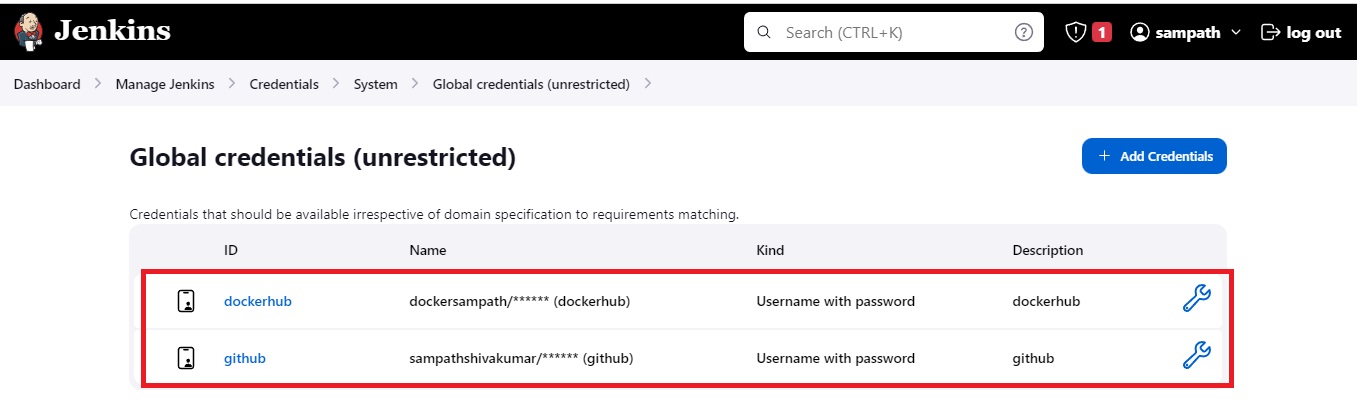
* **Docker Pipeline.** [](https://user-images.githubusercontent.com/119833411/242859779-3a2e74d4-15dd-46bf-88e0-2b8e1f771bdb.jpg)

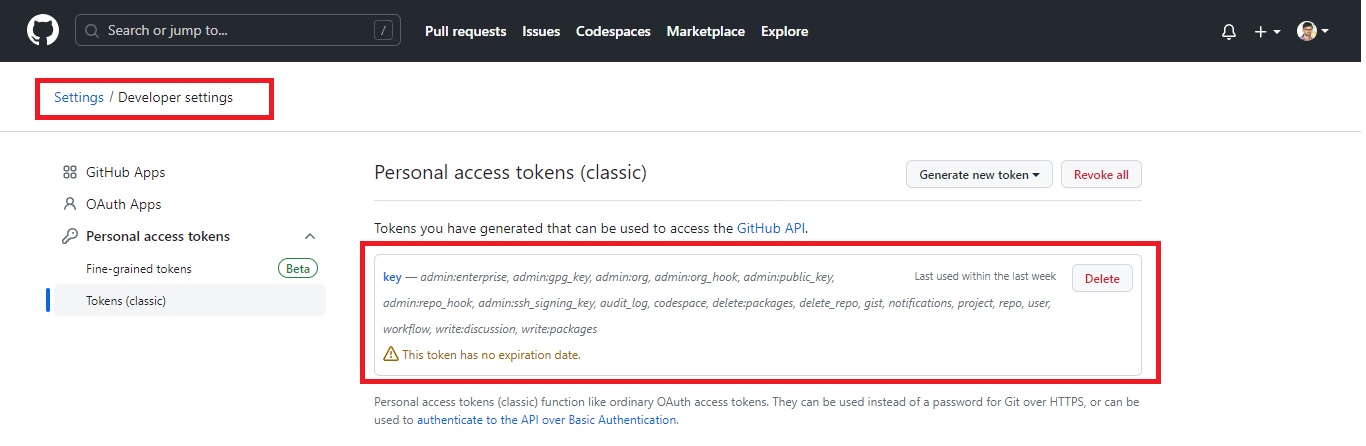
[](https://user-images.githubusercontent.com/119833411/242860023-1ca39b42-bc60-4084-91de-0dc57f3befa2.jpg)

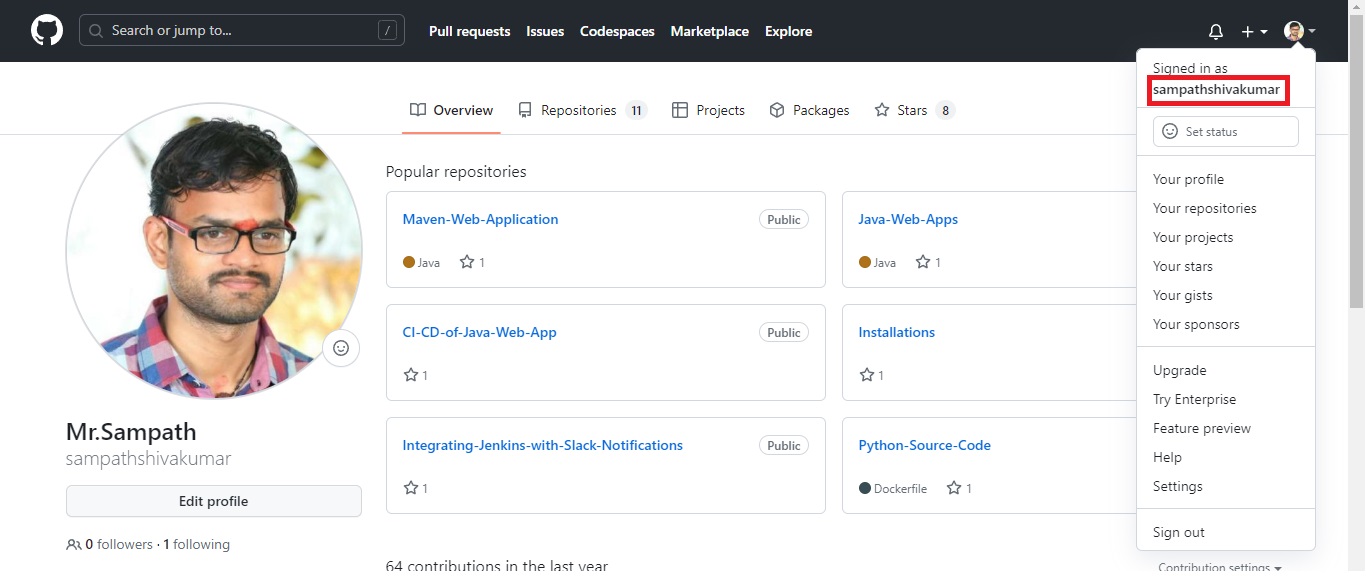
**Create a Repository name "packages" in your Docker Hub.**

[](https://user-images.githubusercontent.com/119833411/242860628-c6cd7366-32d0-4c88-8ec3-ae244595afdc.jpg)

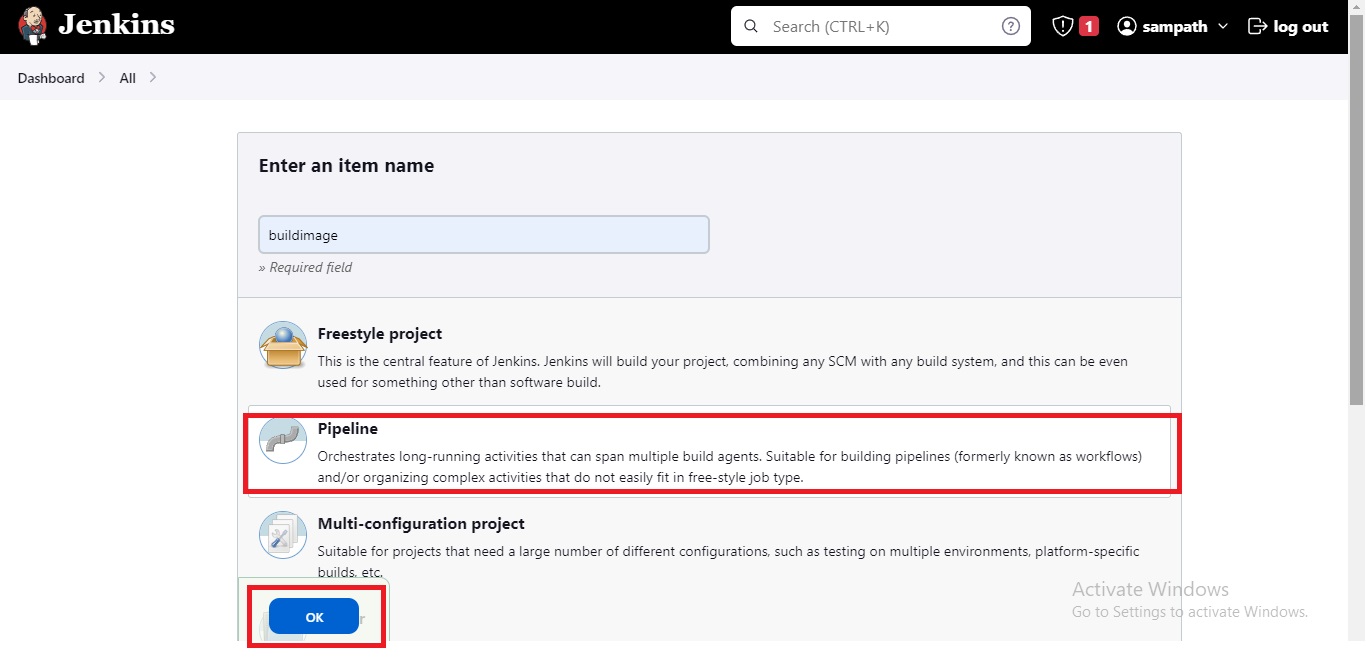
**Add your GitHub and Docker Hub credentials in your Jenkins credentials.**

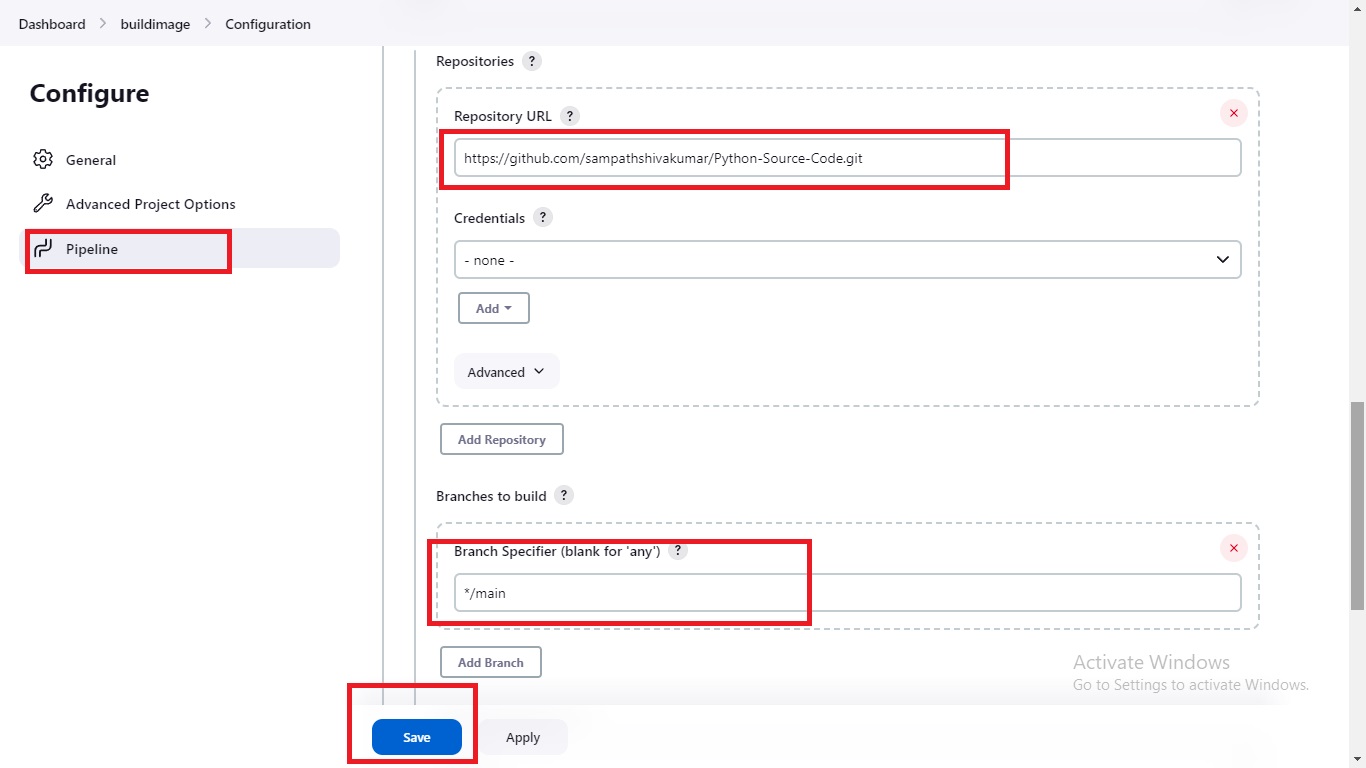
* **Save GitHub credentials with ID "github"**
* **Save Docker Hub credentials with ID "dockerhub"**
* **Dont change "ID", "Repository" or any names as we are involving save name in jenkins-pipelines file.** [](https://user-images.githubusercontent.com/119833411/242862908-c4c4d2e6-8d1e-40b5-8991-0294739ac990.jpg)

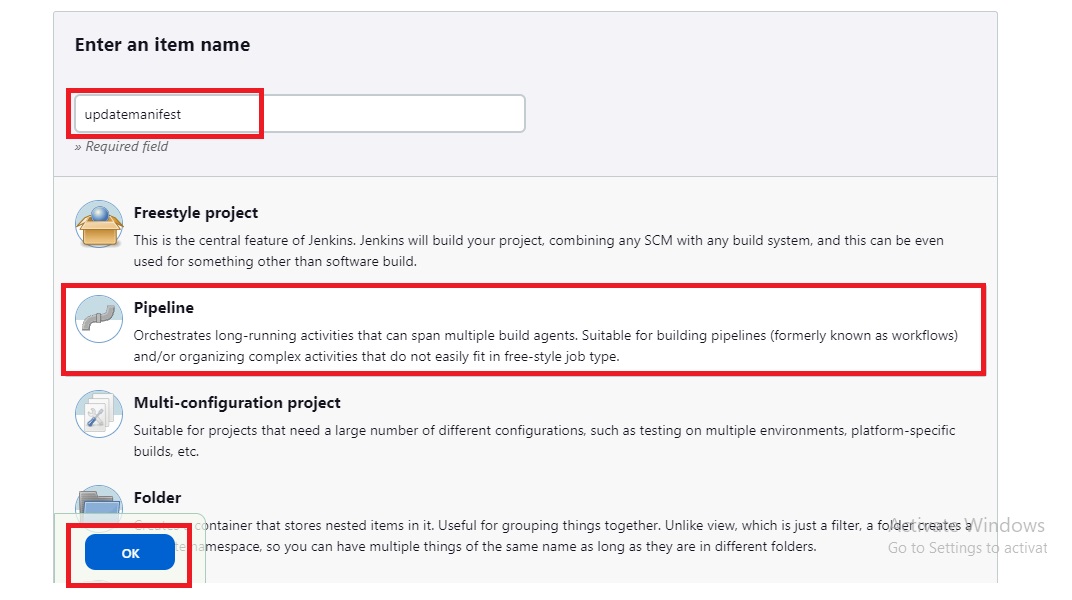
**Note:-For GitHub use Personal access tokens as password.** [](https://user-images.githubusercontent.com/119833411/242864773-dc4a91a9-fe36-46c8-97b2-a7e7fc36487e.jpg)

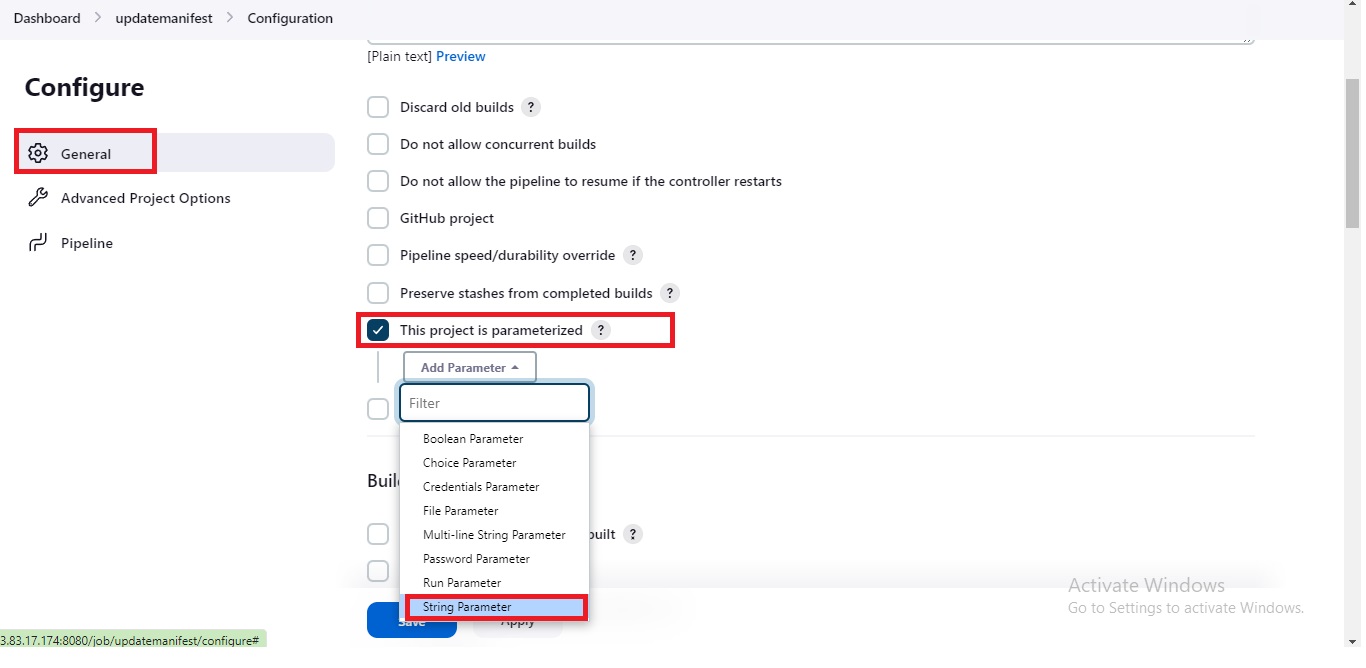
**Note:-GitHub username means not your email. You can find it here on your GitHub page.** [](https://user-images.githubusercontent.com/119833411/242863701-a527b312-707a-4276-831a-cf9d216ed8ca.jpg)

**Lets create two jobs on Jenkins.**

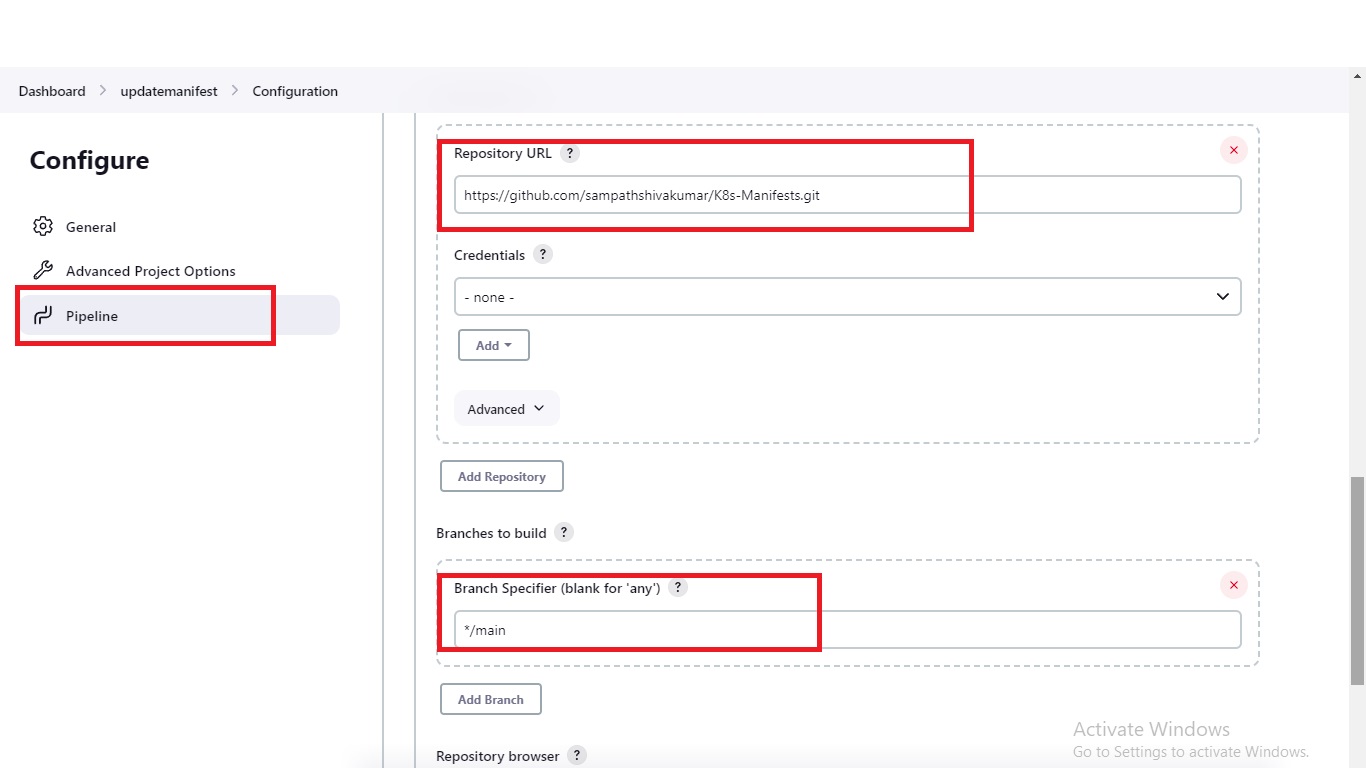
**Job-1** [](https://user-images.githubusercontent.com/119833411/242865351-5134b474-bee7-446d-bc9f-2718de178918.jpg) **Select Pipeline from scm and give Python-Source-Code-git repo url and change branch to main and click save.**

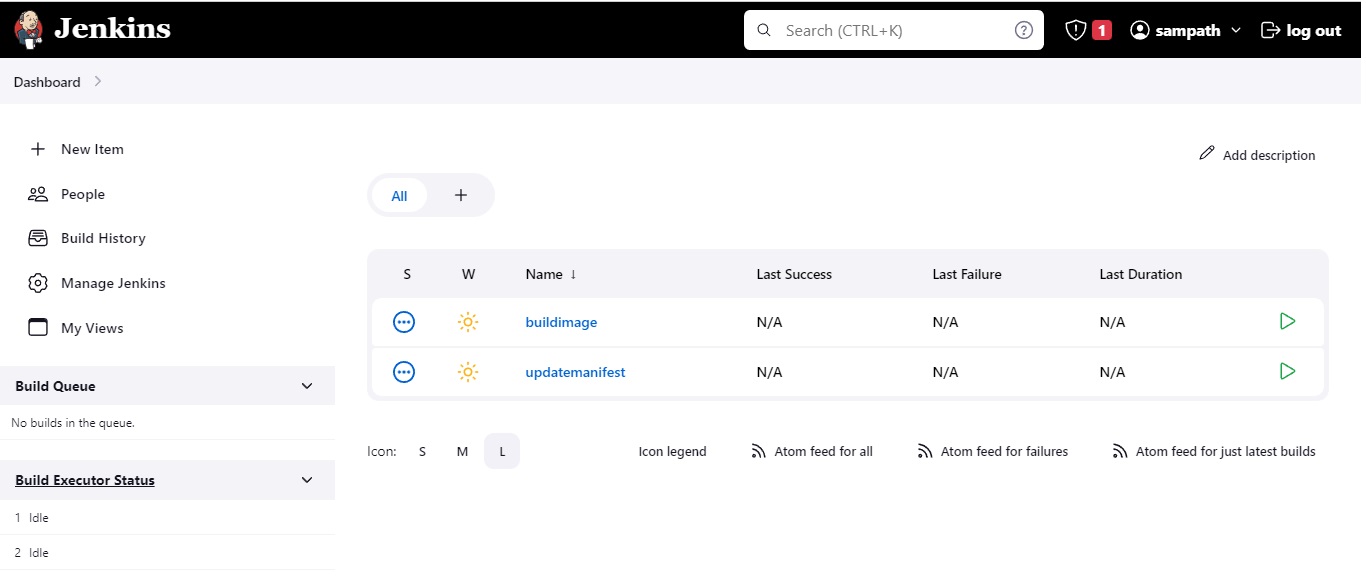
[](https://user-images.githubusercontent.com/119833411/242866173-2952025e-44b6-4d93-b8a9-f154c52fc81f.jpg)

**Job-2** [](https://user-images.githubusercontent.com/119833411/242866830-7f49c014-a1e1-468e-bf8c-fc079468be34.jpg)

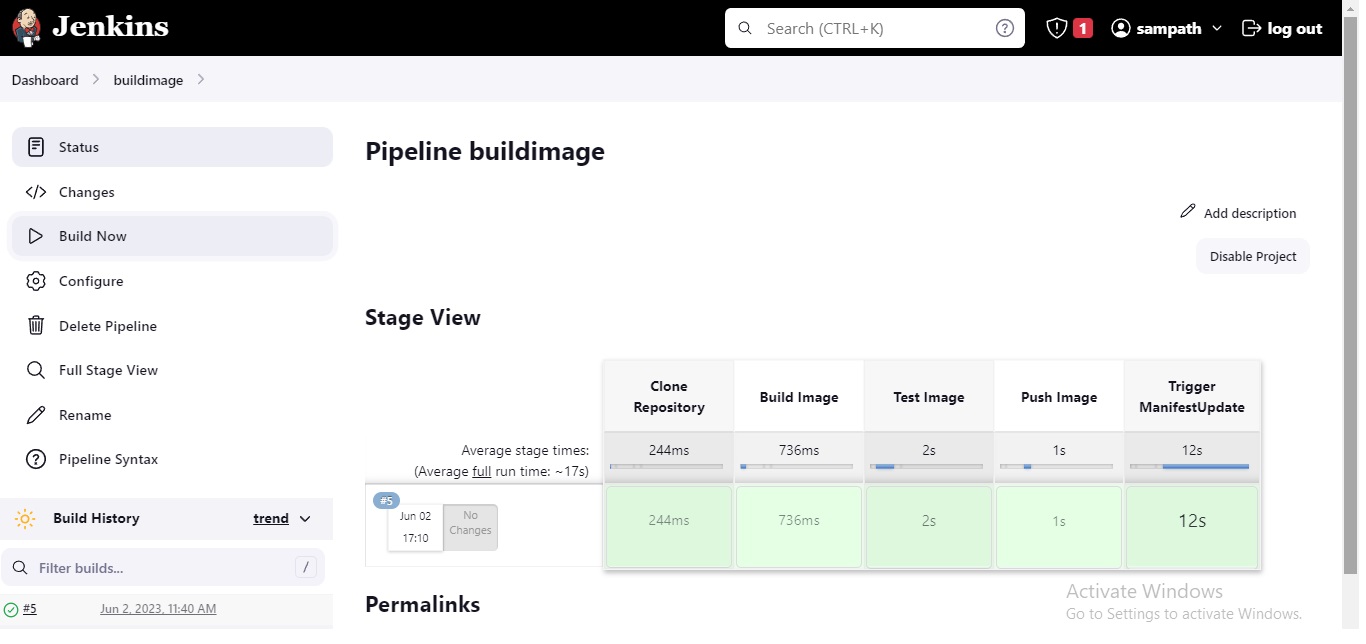
**Select This project is parameterized, give name "DOCKERTAG",Default Value as "latest"**  
[](https://user-images.githubusercontent.com/119833411/242868082-1832a3c8-f613-42db-bd13-910326383f26.jpg)

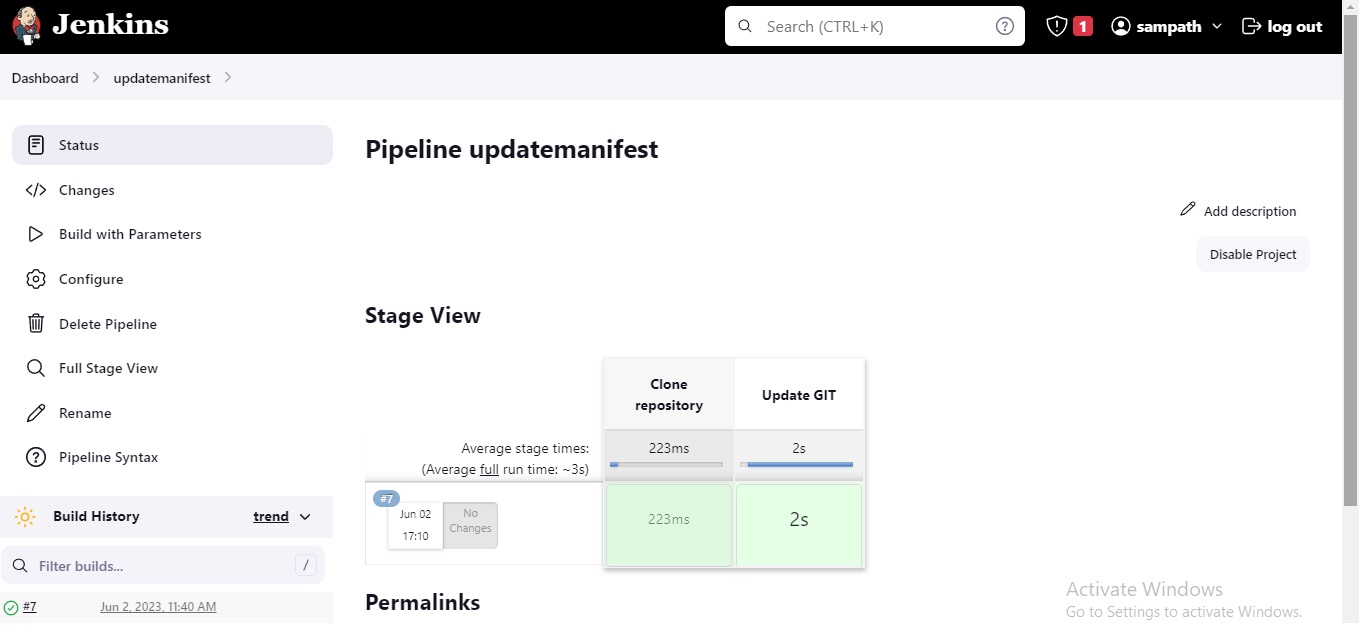
**Select Pipeline from scm and give K8s-Manifests-git-repo url and change branch to main and click save.**

[](https://user-images.githubusercontent.com/119833411/242867452-aa0daab2-7e16-4575-a518-6bda248a4ec4.jpg)

[](https://user-images.githubusercontent.com/119833411/242867716-c94a3fee-60a2-47eb-9bd1-682dfe09810e.jpg)

**Select Job-1 "buildimage" and click on "Build Now"**

**If you have configured everything correctly you should see all stages of pipeline executed successfully** [](https://user-images.githubusercontent.com/119833411/242874408-b555f535-7ee5-4203-a486-8596898534fa.jpg)

**Job-2 should also get trigged automatically** [](https://user-images.githubusercontent.com/119833411/242874598-62f53920-63b6-45a6-9673-27e07aeabb8d.jpg)

**Congratulations you have done 90 % of Project as of now. Now we just need to setup EKS Cluster and install ArgoCD init.**

**Launch a new instance and install AWS CLI, eksctl, kubectl in to Create, and interact with EKS Cluster in AWS.**

**Select Amazon Linux-2 AMI.**

**t2.micro**

**AWS CLI Installation**

curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"

unzip awscliv2.zip

sudo ./aws/install

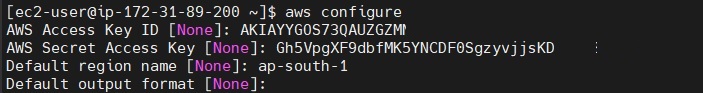
# Check AWS CLI version

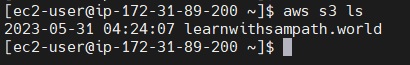
aws --version

**Now Configure the AWS CLI with your AWS "Access key" and "Secret access key".**

# trype aws configure and press enter, then follow the flow.

aws configure

[](https://user-images.githubusercontent.com/119833411/242883724-79ab8535-ffa1-4106-b14b-18b3cb2f5d31.jpg)

**Now Check by listing s3 buckets in your AWS account using AWS CLI.** [](https://user-images.githubusercontent.com/119833411/242884217-fd079905-d3ce-4d26-a096-ff2954d4ef83.jpg)

**Done, its showing my Bucket successfully, AWS CLI is configured correctly**

**Now lets install eksctl.**

ARCH=amd64

PLATFORM=$(uname -s)\_$ARCH

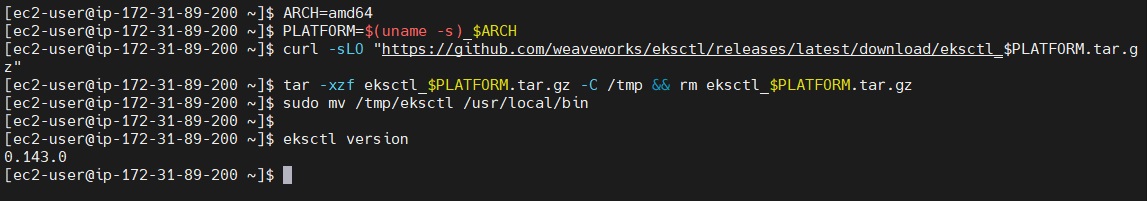
curl -sLO "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$PLATFORM.tar.gz"

tar -xzf eksctl\_$PLATFORM.tar.gz -C /tmp && rm eksctl\_$PLATFORM.tar.gz

sudo mv /tmp/eksctl /usr/local/bin

# Check the eksctl version.

eksctl version

[](https://user-images.githubusercontent.com/119833411/242885445-b399b593-1033-4236-81c0-ff04f1180f63.jpg)

**Lets Install kubectl**

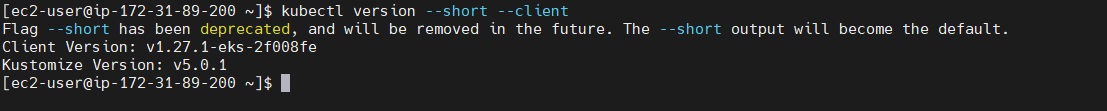
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.27.1/2023-04-19/bin/linux/amd64/kubectl

chmod +x ./kubectl

mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH

echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc

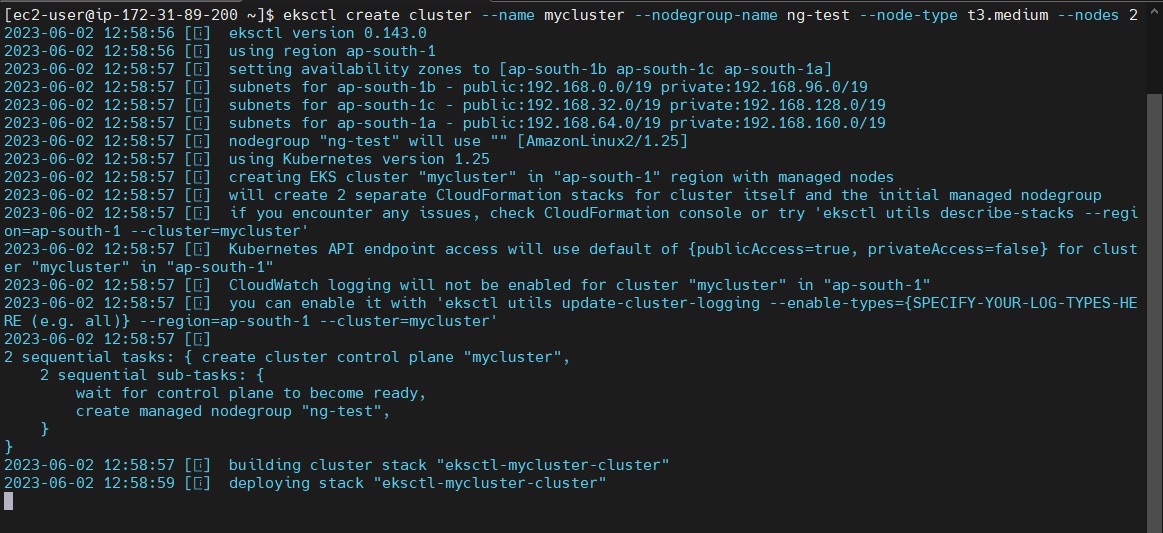
kubectl version --short --client

[](https://user-images.githubusercontent.com/119833411/242886396-f6c71df7-c4a5-48c4-aa06-6417cce8530f.jpg)

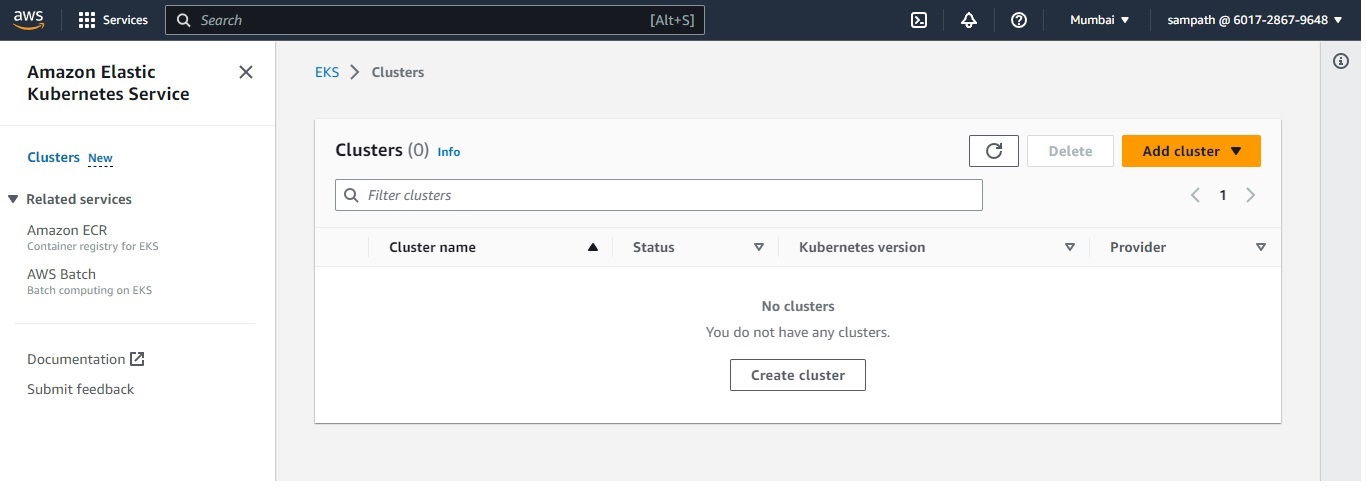
**Command to Create EKS Cluster using eksctl command**

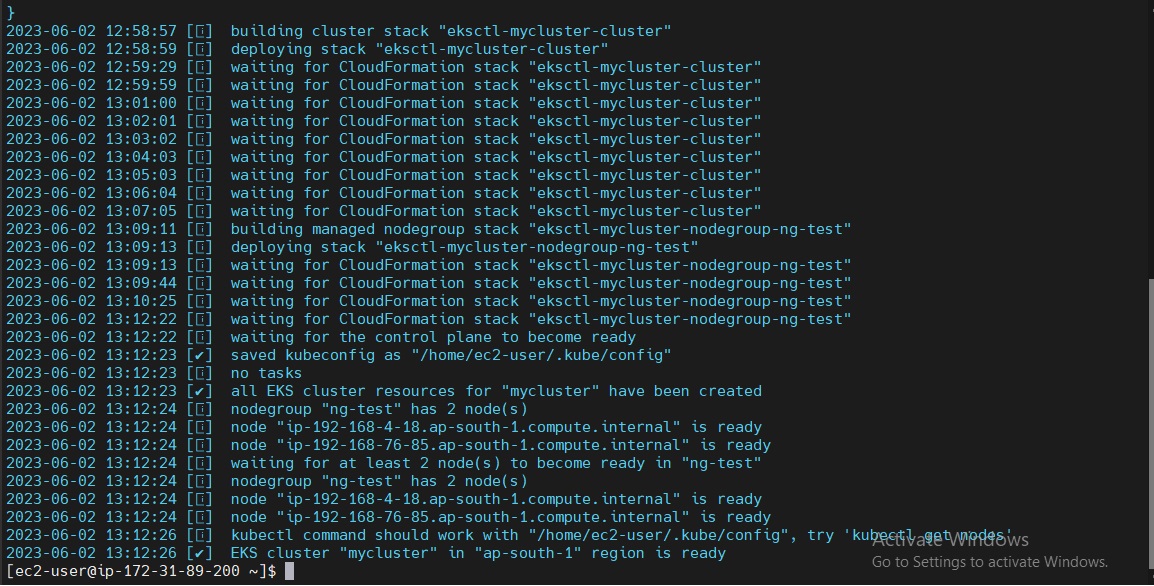
eksctl create cluster --name <name-of-cluster> --nodegroup-name <nodegrpname> --node-type <instance-type> --nodes <no-of-nodes>

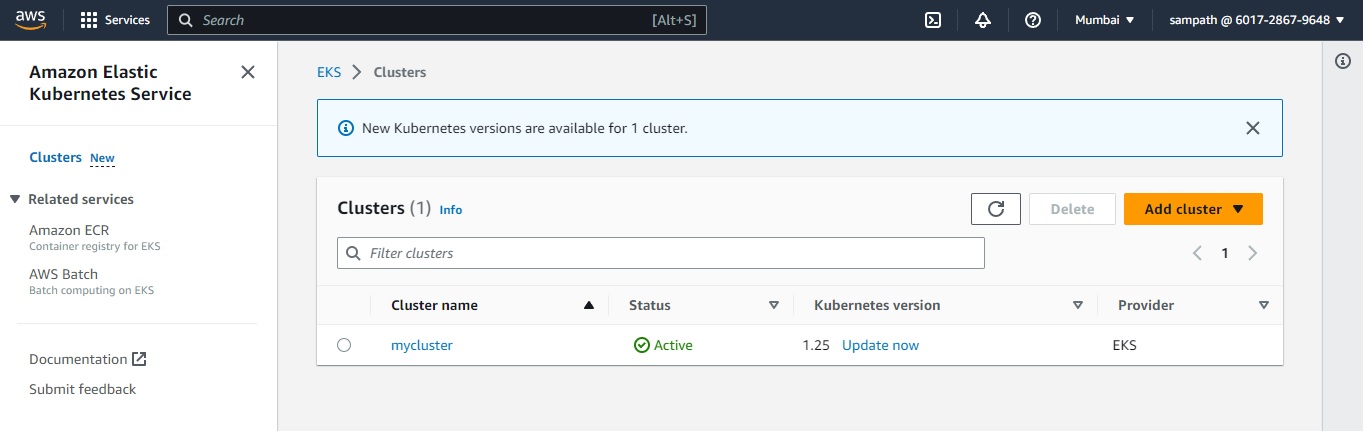
eksctl create cluster --name mycluster --nodegroup-name ng-test --node-type t3.medium --nodes 2

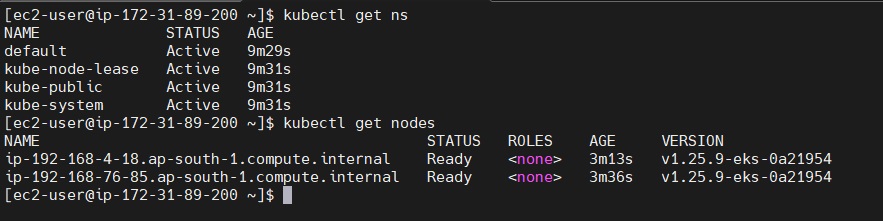
[](https://user-images.githubusercontent.com/119833411/242890376-9b735583-3f95-4a23-8d05-bdffa44de083.jpg)

**It will take 5-10 mins to create cluster.**

**We can see as of now there is no cluster in EKS** [](https://user-images.githubusercontent.com/119833411/242893160-df3871a3-2d98-481d-8357-72aa9f753013.jpg)

[](https://user-images.githubusercontent.com/119833411/242893386-7e6012df-47ac-462a-b8d8-b4eb382ed534.jpg)

**It's Done. Now lets see the cluster** [](https://user-images.githubusercontent.com/119833411/242893531-260960a1-d62b-418a-bbad-3d1be91e4545.jpg)

**Lets test some kubectl commands** [](https://user-images.githubusercontent.com/119833411/242893906-f33f3b00-a87e-4188-bfaf-c8e28cd224ed.jpg)

**EKS Cluster is up and ready.**

**Now lets install ArgoCD in EKS Cluster.**

# This will create a new namespace, argocd, where Argo CD services and application resources will live.

kubectl create namespace argocd

kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

**Download Argo CD CLI**

curl -sSL -o argocd-linux-amd64 https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64

sudo install -m 555 argocd-linux-amd64 /usr/local/bin/argocd

rm argocd-linux-amd64

**Access The Argo CD API Server**

# By default, the Argo CD API server is not exposed with an external IP. To access the API server,

choose one of the following techniques to expose the Argo CD API server:

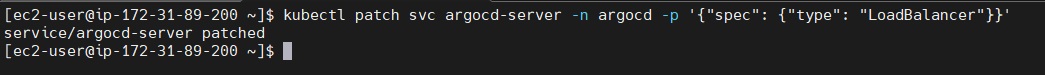
\* Service Type Load Balancer

\* Port Forwarding

**Lets go with Service Type Load Balancer.**

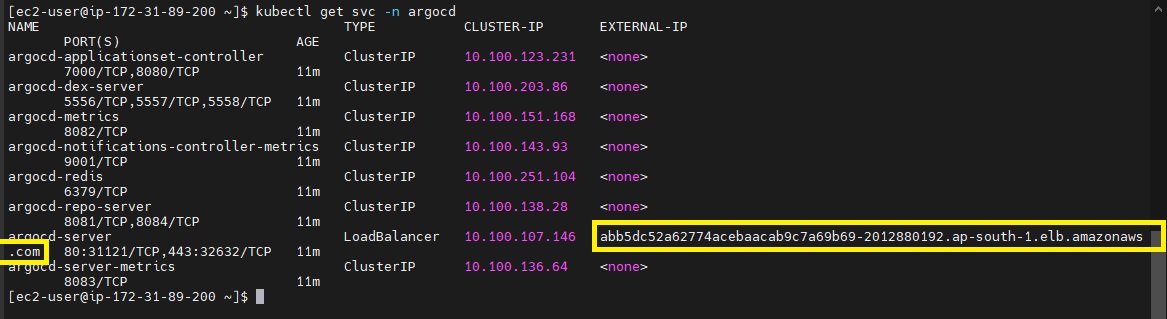
# Change the argocd-server service type to LoadBalancer.

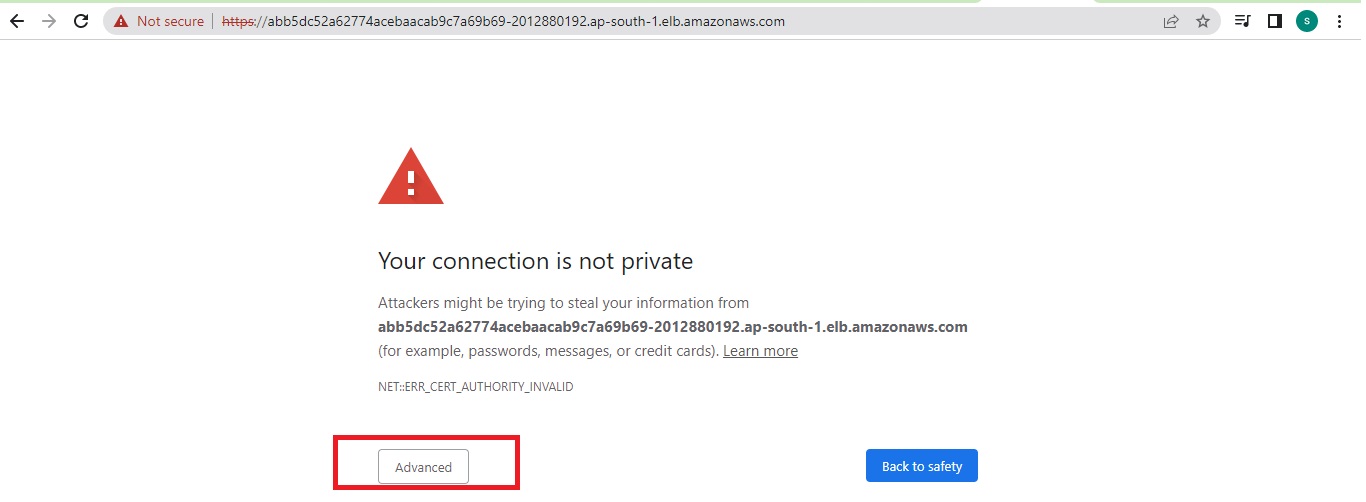
kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'

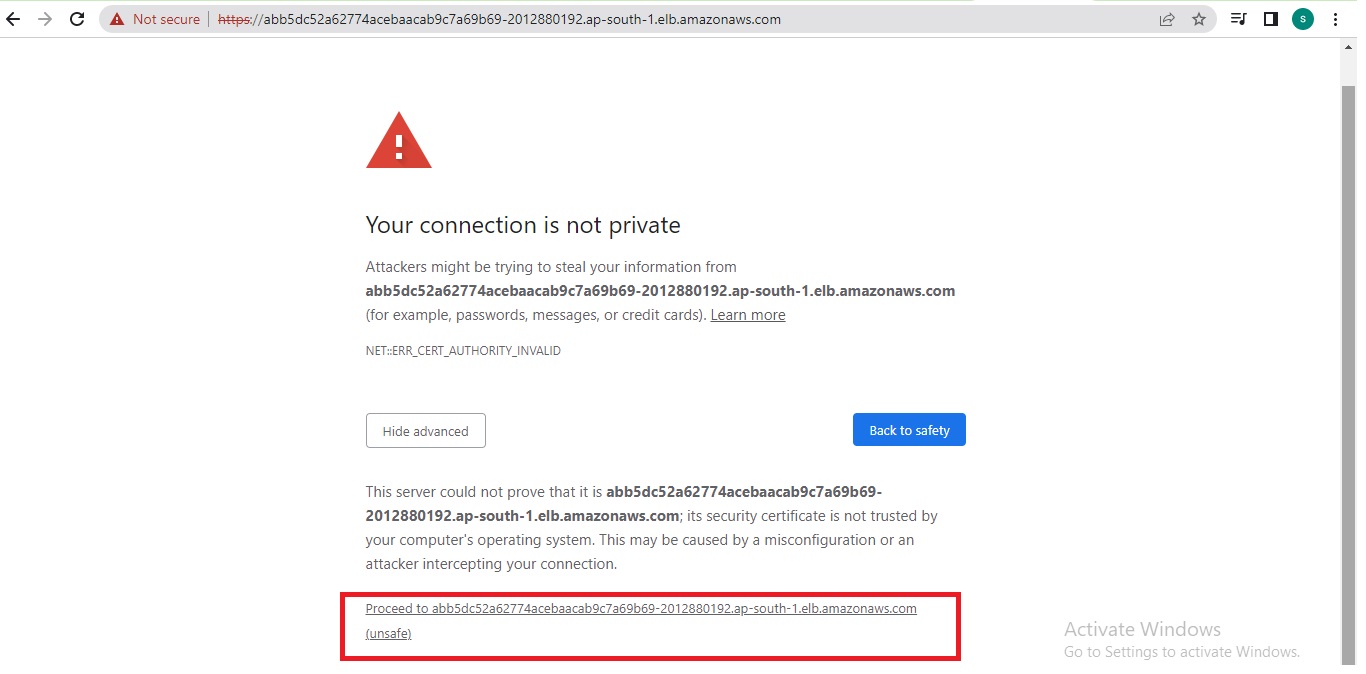
[](https://user-images.githubusercontent.com/119833411/242896467-24a636cf-6ca7-4fcc-9b9f-5f20f8eedcc7.jpg)

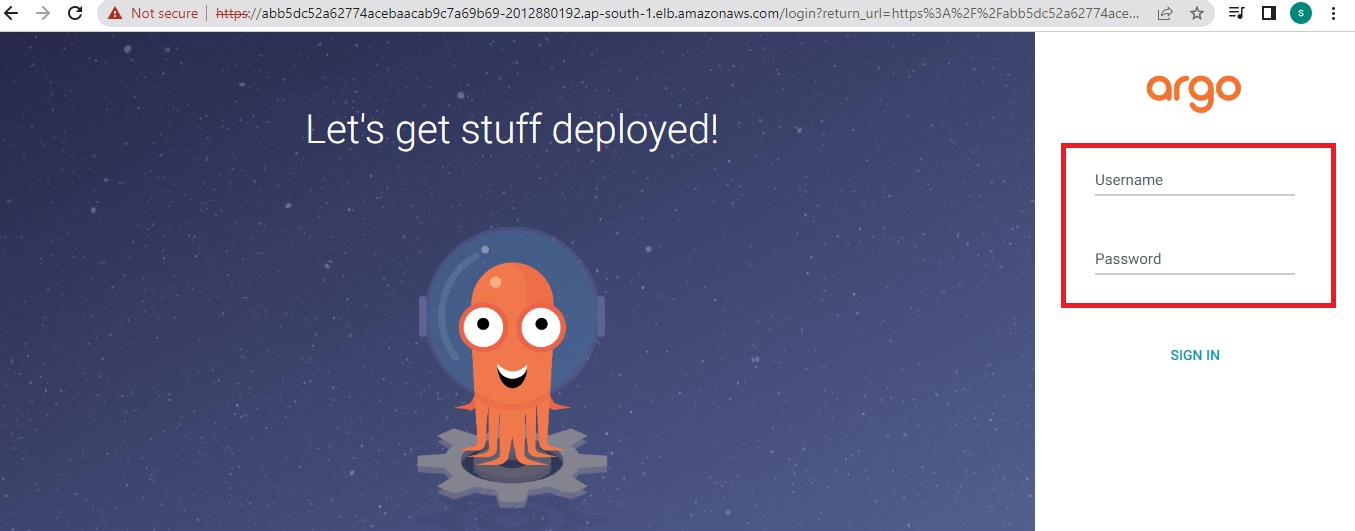
**Get the load balancer url**

kubectl get svc -n argocd

[](https://user-images.githubusercontent.com/119833411/242898489-60d0d141-a677-4124-b383-c49355a719b1.jpg)

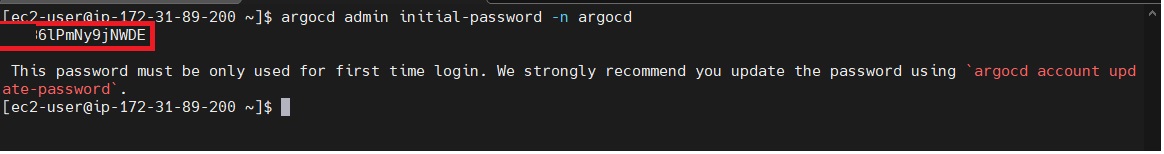
**Lets access the ArgoCD GUI** [](https://user-images.githubusercontent.com/119833411/242898787-2379b513-377b-4556-bc77-10539bc6ca40.jpg)

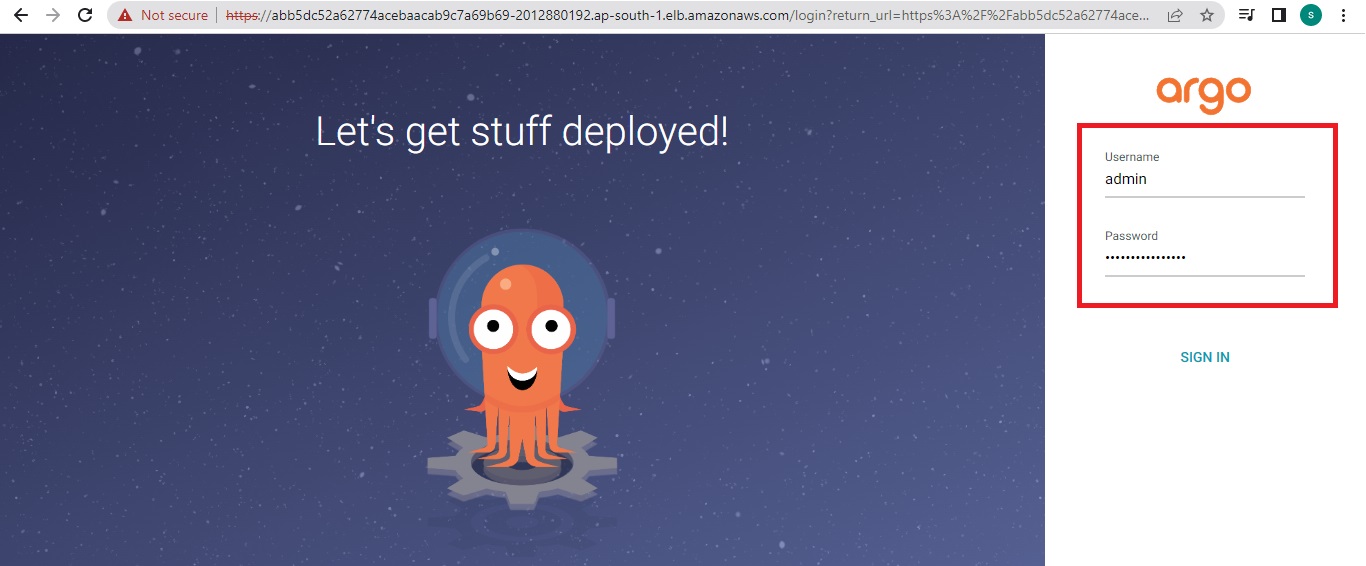
[](https://user-images.githubusercontent.com/119833411/242899028-fd0885fe-08b0-42c9-9d82-12ed16900bd7.jpg)

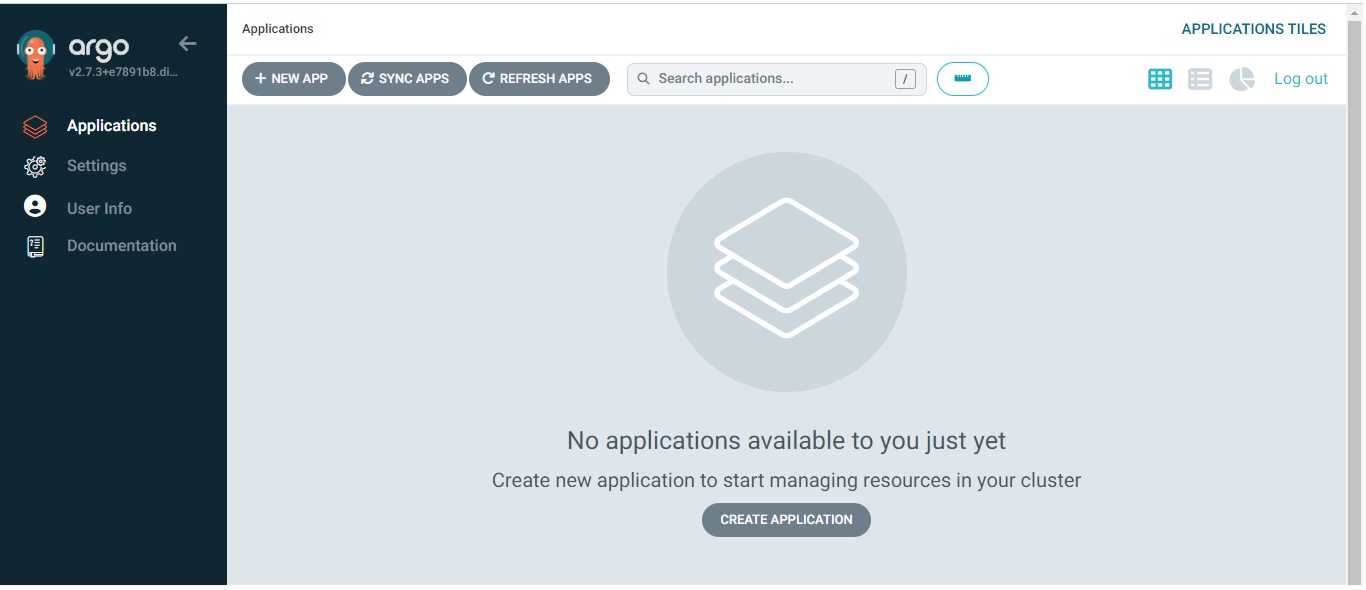
**Enter the user name and Password** [](https://user-images.githubusercontent.com/119833411/242899289-fffac2b0-9bb9-4ab1-8c87-e45f11ed2297.jpg)

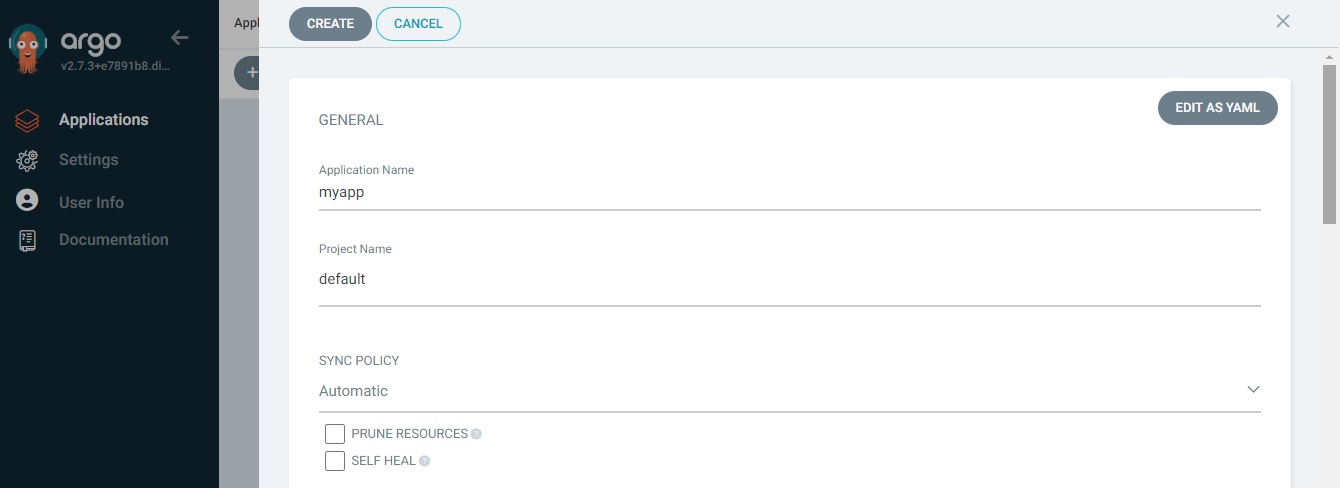
**Login Using The CLI**

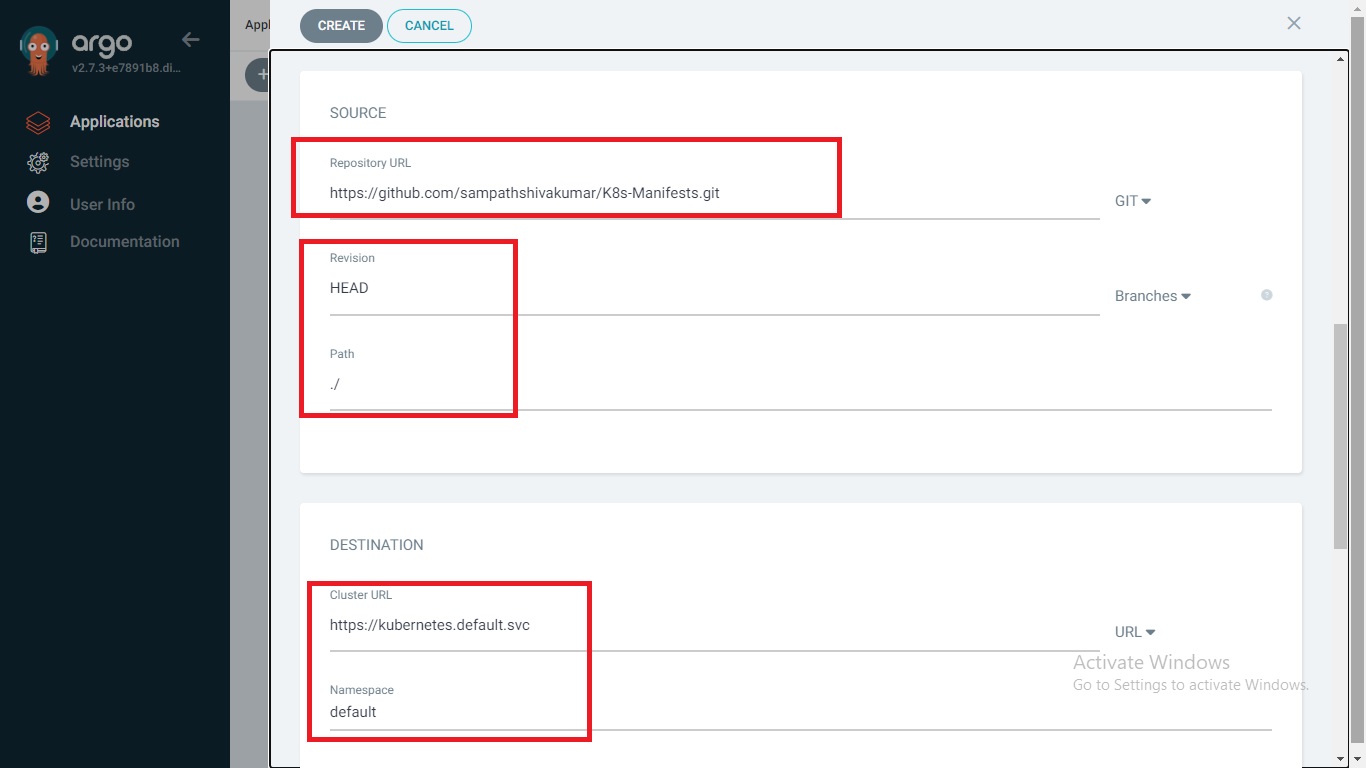
argocd admin initial-password -n argocd

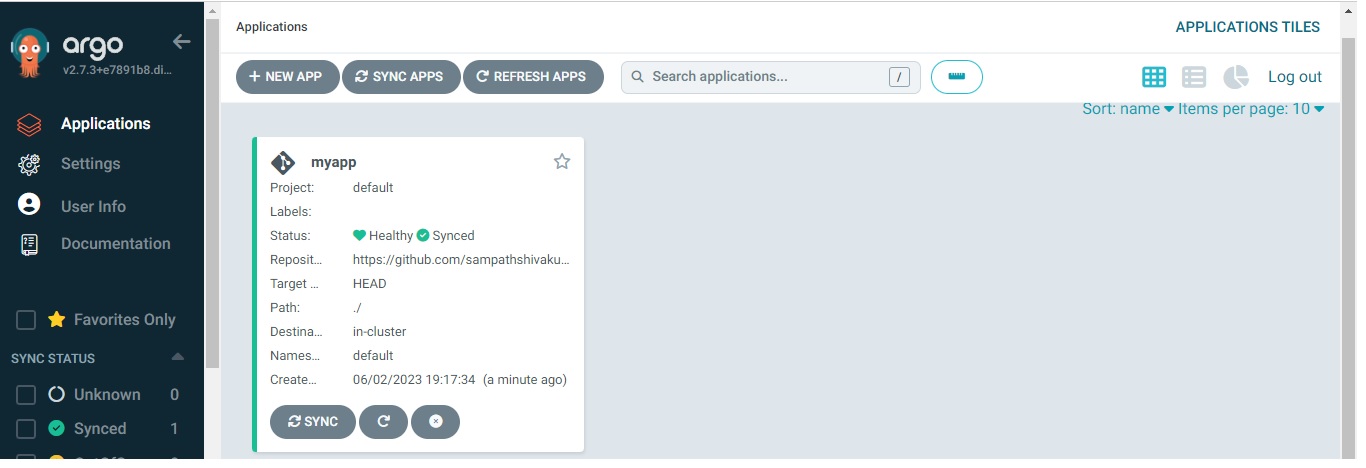
[](https://user-images.githubusercontent.com/119833411/242899843-be3c87a5-0e42-473f-b65f-9fe1fb6ee53b.jpg)

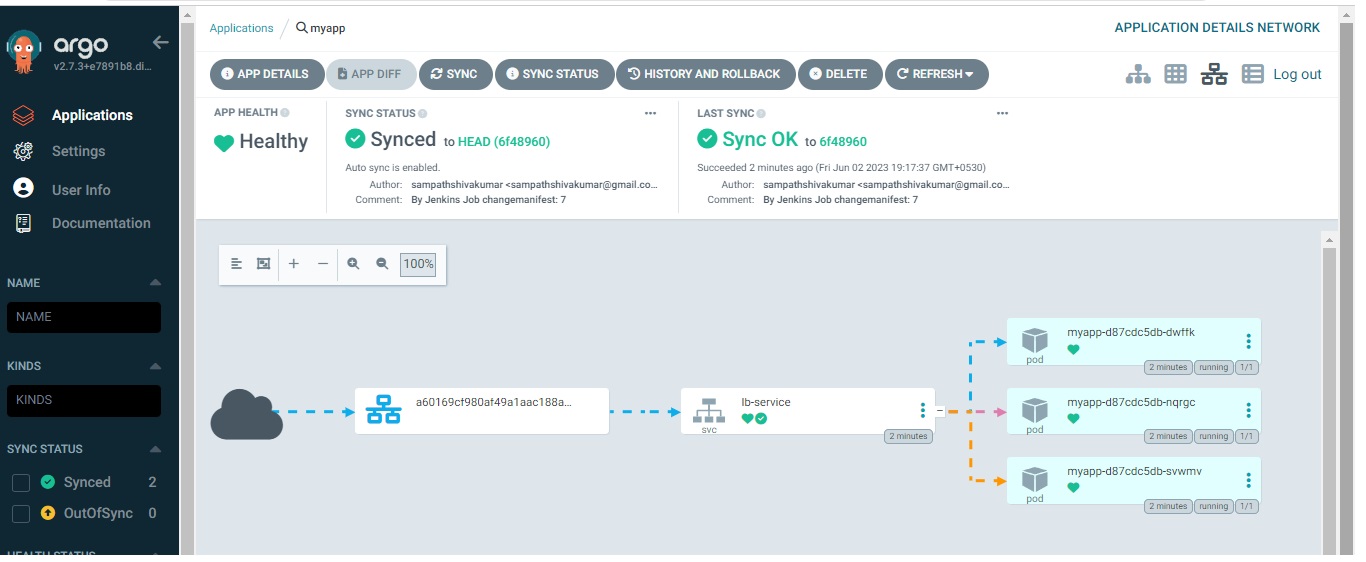
[](https://user-images.githubusercontent.com/119833411/242900089-a7d93f6b-2a47-43b7-85bc-a3291eb8a627.jpg)

**ArgoCD Dashboard** [](https://user-images.githubusercontent.com/119833411/242900283-876ce7c5-c43f-4dd8-b534-cb03f4d284fc.jpg)

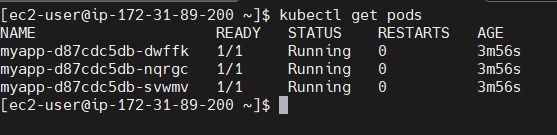
**Click on New App** [](https://user-images.githubusercontent.com/119833411/242900868-984aedd4-9ecf-4331-8843-2c68f24cd920.jpg)

**Enter Repository URL,set path to ./ , Cluster URL to**[**https://kubernetes.default.svc**](https://kubernetes.default.svc/)**, namespace to default and click save.** [](https://user-images.githubusercontent.com/119833411/242901522-b6e4c2a5-6b2a-4928-a14c-d80c7deae138.jpg)

**You should see the below, once your done** [](https://user-images.githubusercontent.com/119833411/242902353-8a5b04ca-92d2-442a-a2a1-d4f720d5352d.jpg)

**Click on it** [](https://user-images.githubusercontent.com/119833411/242902594-76382e85-69b7-4939-959f-8e7df815e952.jpg)

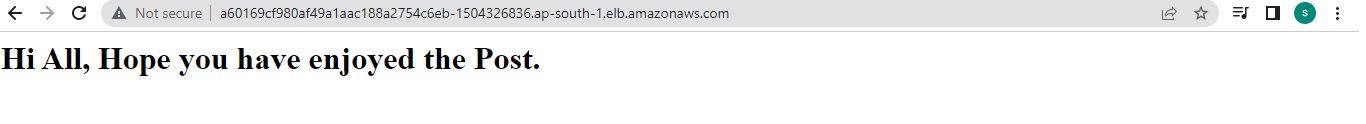
**You can see the pods running in EKS Cluster**

[](https://user-images.githubusercontent.com/119833411/242902936-a2a7c2ec-305c-4512-8c3c-b15b5a754120.jpg)

**We can see the out of pods using load balancer url**

kubectl get svc

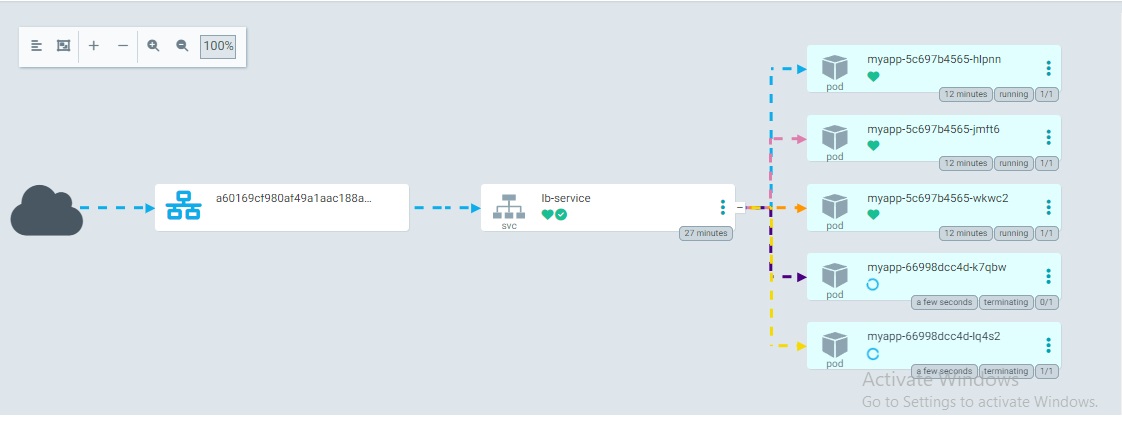
[](https://user-images.githubusercontent.com/119833411/242903766-d1c7e7f9-7b4a-4bf6-965e-dfc2f7ca24e9.jpg)

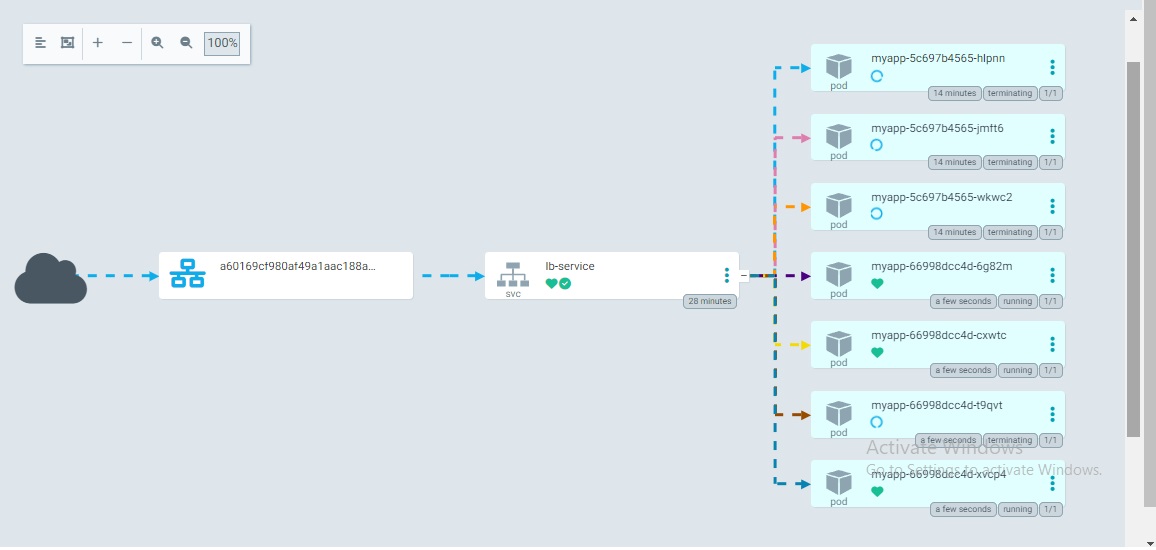
[](https://user-images.githubusercontent.com/119833411/242905695-a474acec-b1bf-4ffd-9783-6728d6140913.jpg)

**ArgoCD will automatically syn for every 3 mins to manifest repo to pull and apply changes to EKS Cluster.**

**If your are intrested you can apply github web hook to automatically trigger jenkins job when developer commit changes in git repo. So that ArgoCD can pull those changes and apply in EKS Cluster.**

**Lets Do some changes in code and the output will automatically change or not.**

**Yes it observed some changes** [](https://user-images.githubusercontent.com/119833411/242908792-05584fd1-eec6-4bfe-afb5-c59d60d9d61b.jpg)

[](https://user-images.githubusercontent.com/119833411/242909094-5c44d4aa-c614-487b-92f5-e91d59816e03.jpg)

**Here is the output**

**[](https://user-images.githubusercontent.com/119833411/242909372-25383eca-fb02-47ba-8d13-813bc6f0d76e.jpg)**

**We have successfully Deployed a Flask Application to a EKS Cluster using Jenkins and ArgoCD.**

**Clean up Cluster**

eksctl delete cluster --name <name-of-cluster>

eksctl delete cluster --name mycluster

**Thank you for reading this post! I**