DevSecOps-Demo Project

Step by step for pratices

**Prerequisites**

Before diving into the project, make sure you have the following prerequisites in place:

1. **Local Environment Setup**:

* **Terraform and AWS CLI**: Install and configure Terraform and AWS CLI on your local machine. Basic knowledge of these tools is necessary.
* **Basic Knowledge**: Ensure basic knowledge of Terraform, AWS CLI, and familiarity with cloud concepts.

2. **Jenkins Server Deployment:**

* **Git**: Basic knowledge of Git commands is required.
* **AWS EC2**: Understanding of AWS EC2 instances and security groups.

**3. Jenkins Configuration:**

* **Jenkins**: Familiarity with Jenkins and basic Jenkins pipeline concepts.
* **Docker, Sonarqube, Terraform, Kubectl, AWS CLI, Trivy**: Basic knowledge of these tools is necessary.

**4. ArgoCD Setup:**

* **Kubernetes**: Basic knowledge of Kubernetes concepts.
* **ArgoCD**: Familiarity with ArgoCD and understanding of continuous deployment concepts.

**5. Pipeline Configuration:**

* **Jenkins Plugins:** Understanding of Jenkins plugins, especially AWS Credentials, and Pipeline: AWS Steps.
* **Tools Configuration:** Basic knowledge of configuring tools like Docker, NodeJS, OWASP Dependency-Check, and SonarQube in Jenkins.

**GitHub Repository for the Project-** <https://github.com/nhuluong2024/DevSecOps-Demo>

**We will install Terraform & AWS CLI to deploy our Jenkins Server(EC2) on AWS**

Install & Configure Terraform and AWS CLI on your local machine to create Jenkins Server on AWS Cloud

**Terraform Installation Script**

wget -O- https://apt.releases.hashicorp.com/gpg | sudo gpg - dearmor -o /usr/share/keyrings/hashicorp-archive-keyring.gpg  
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb\_release -cs) main" | sudo tee /etc/apt/sources.list.d/hashicorp.list  
sudo apt update  
sudo apt install terraform -y

**AWSCLI Installation Script**

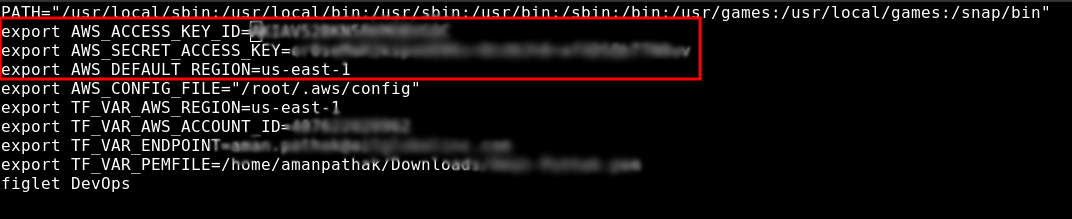
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"  
sudo apt install unzip -y  
unzip awscliv2.zip  
sudo ./aws/install

Now, Configure both the tools

**Configure Terraform**

Edit the file /etc/environment using the below command and add the highlighted lines and add your keys at the blur space.

sudo vim /etc/environment

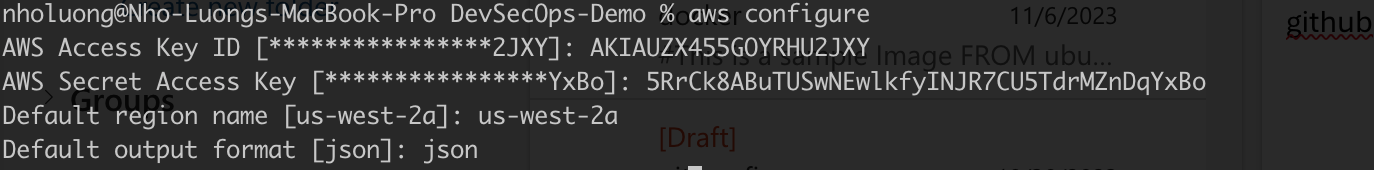


After doing the changes, restart your machine to reflect the changes of your environment variables.

**Configure AWS CLI**

Run the below command, and add your keys

aws configure

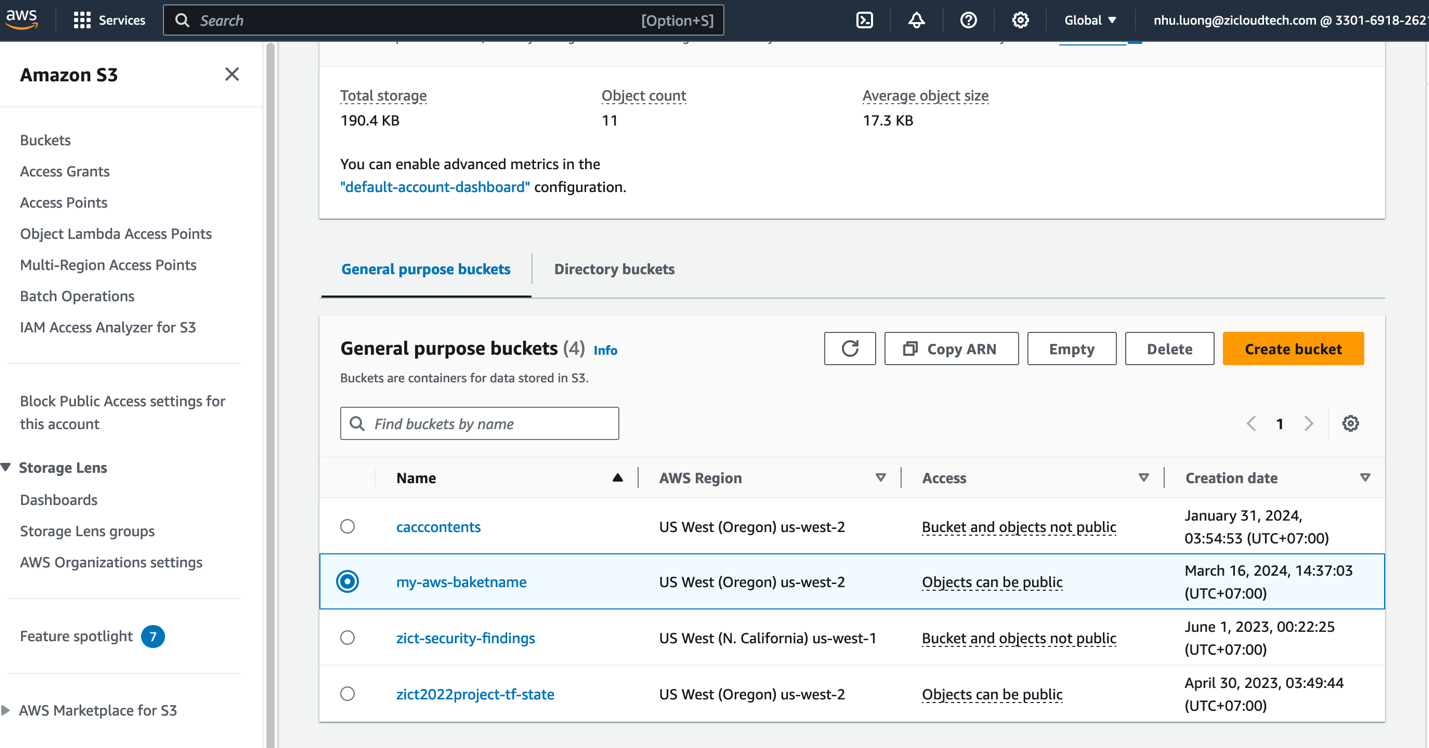


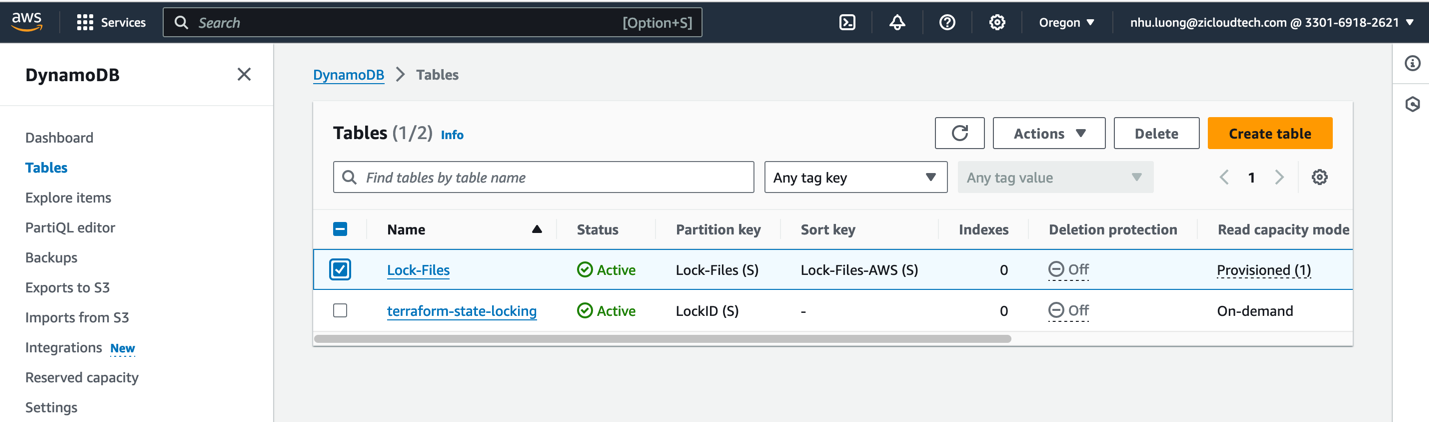
# ****Deploy the Jenkins Server(EC2) using Terraform****

Clone the Git repository

Navigate to the **Jenkins-Server-TF**

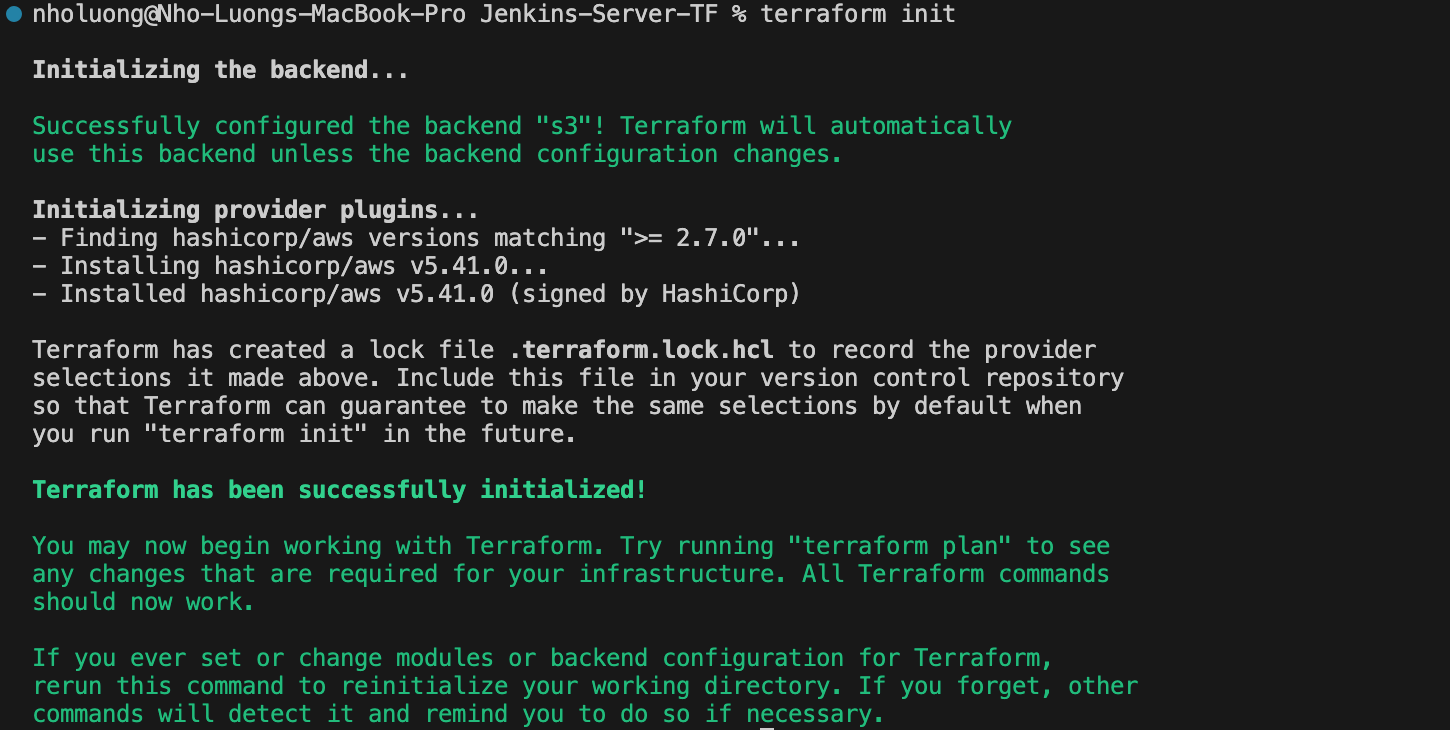
Do some modifications to the backend.tffile such as changing the **bucket** name and **dynamodb** table(make sure you have created both manually on AWS Cloud).





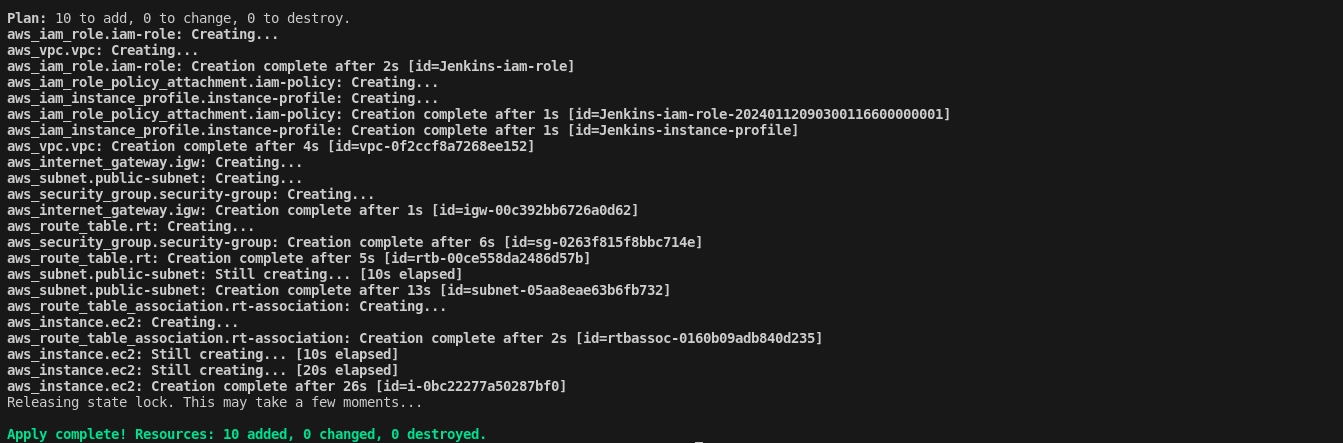
Initialize the backend by running the below command

terraform init

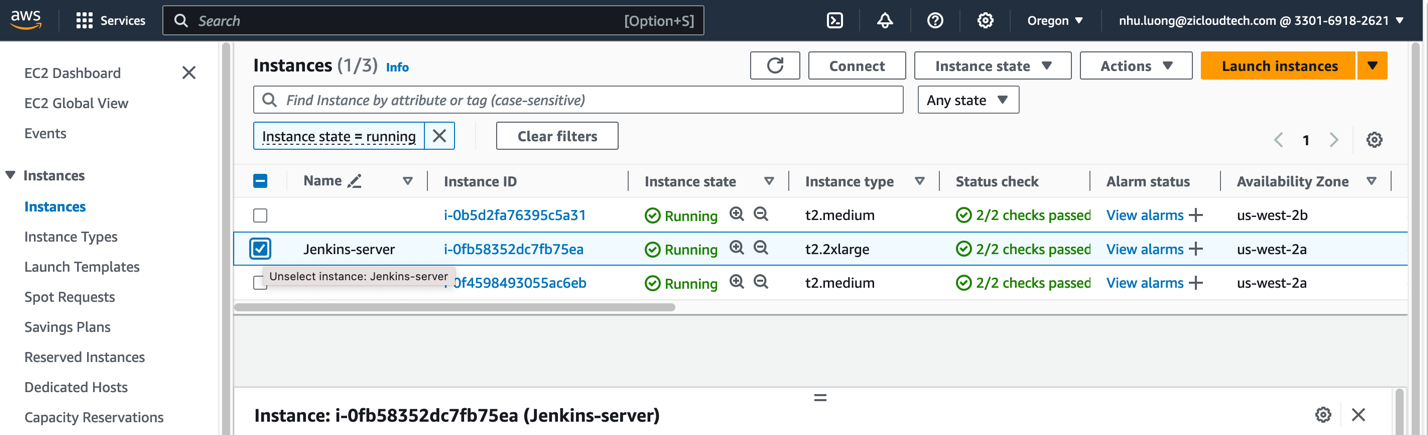


Now, run the below command to create the infrastructure on AWS Cloud which will take 3 to 4 minutes maximum

terraform apply -var-file=variables.tfvars --auto-approve

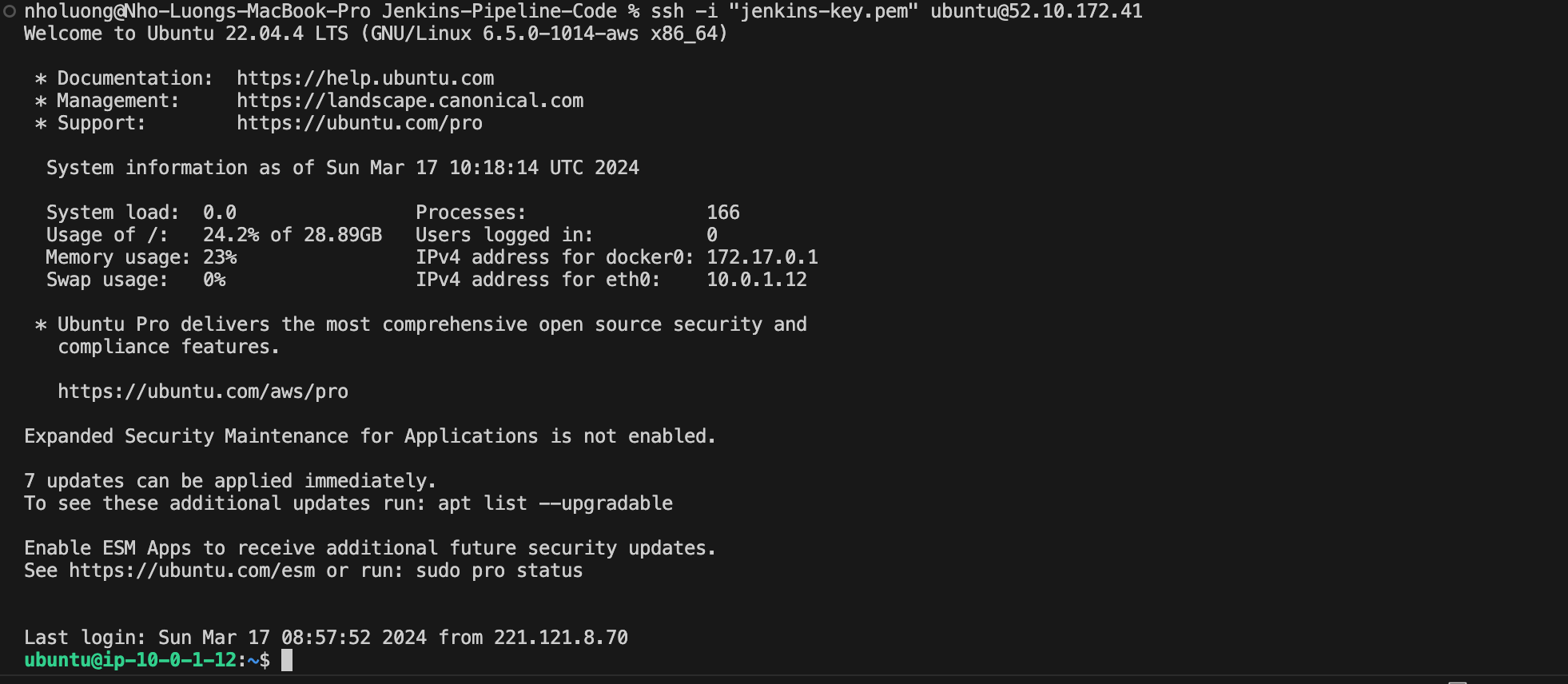


Now, connect to your Jenkins-Server by clicking on Connect.



# ****Configure the Jenkins****

Now, we logged into our **Jenkins server.**



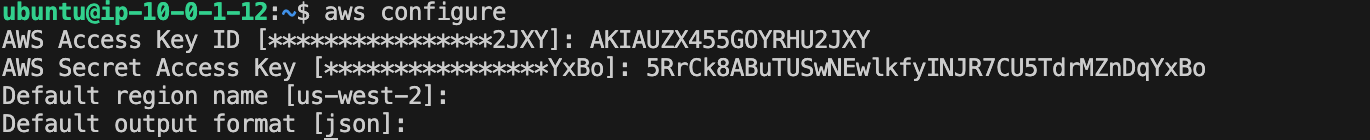
We have installed some services such as Jenkins, Docker, Sonarqube, Terraform, Kubectl, AWS CLI, and Trivy.

Let’s validate whether all our installed or not.

jenkins --version  
docker --version  
docker ps  
terraform --version  
kubectl version  
aws --version  
trivy --version

# ****We will deploy the EKS Cluster using Jenkins****

Now, go back to your Jenkins Server **terminal** and configure the AWS.



Click on **Manage Jenkins**

Select the **Available plugins** and install the following plugins and click on **Install**

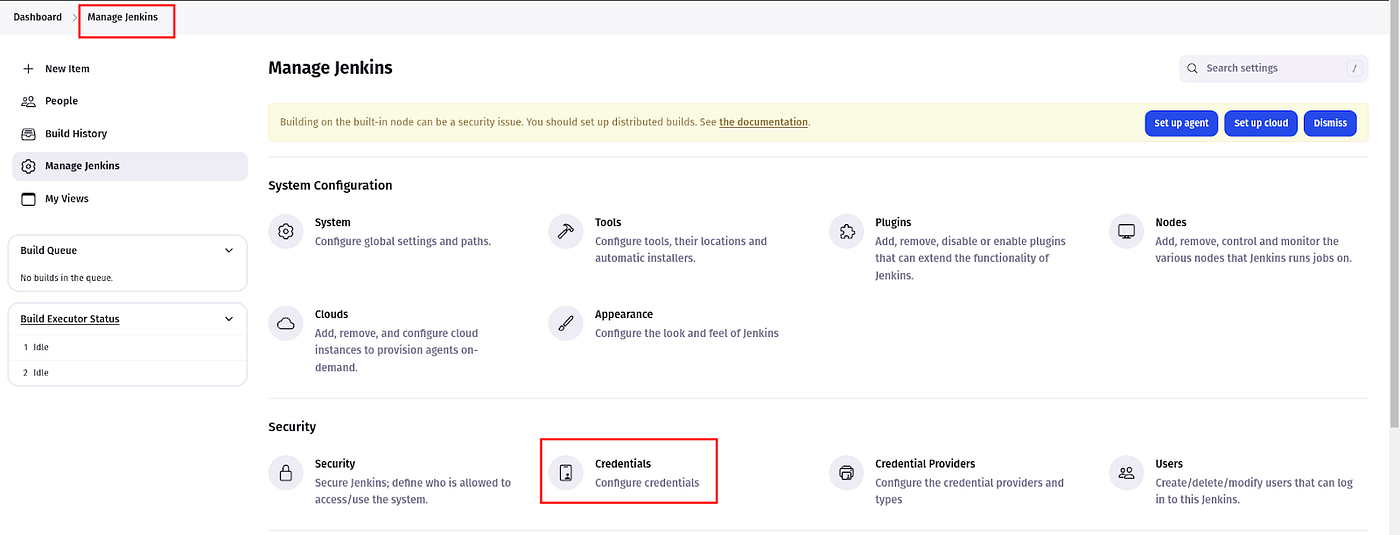
AWS Credentials  
Pipeline: AWS Steps

Docker  
Docker Commons  
Docker Pipeline  
Docker API  
docker-build-step  
Eclipse Temurin installer  
NodeJS  
OWASP Dependency-Check  
SonarQube Scanner

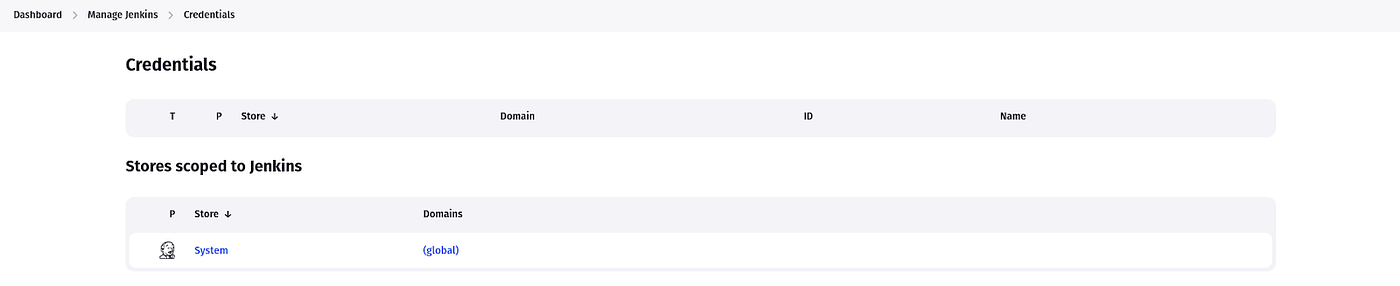
Once, both the plugins are installed, restart your Jenkins service.

Now, we have to set our AWS credentials on Jenkins

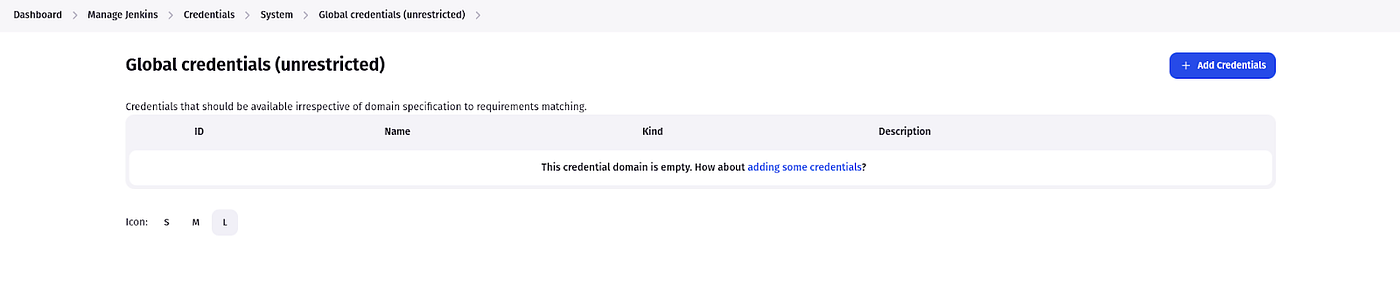
Go to **Manage Plugins** and click on **Credentials**



Click on **global.**



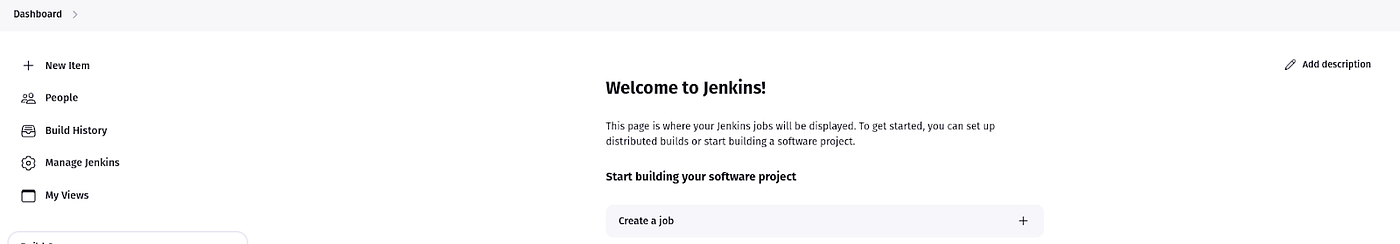
Click on **Add credentials**



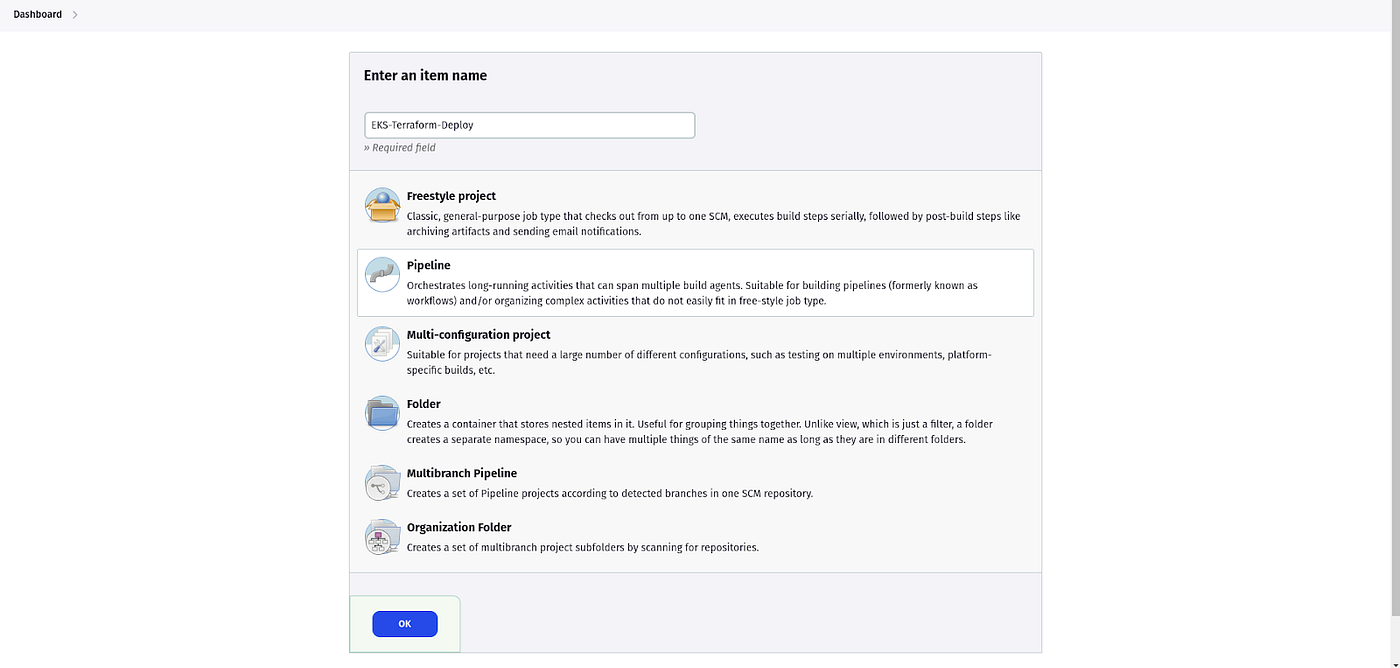
Select **AWS Credentials** as **Kind** and add **the ID** same as shown in the below snippet except for your AWS Access Key & Secret Access key and click on **Create.**



Now, Go to the **Dashboard** and click **Create a job**



Select the **Pipeline** and provide the name to your **Jenkins Pipeline** then click on **OK.**



Now, Go to the GitHub Repository in which the Jenkins Pipeline code is located to deploy the EKS service using Terraform.

After pasting the Jenkinsfile code, click on **Save** & **Apply.**



Click on **Build**



You can see our **Pipeline**

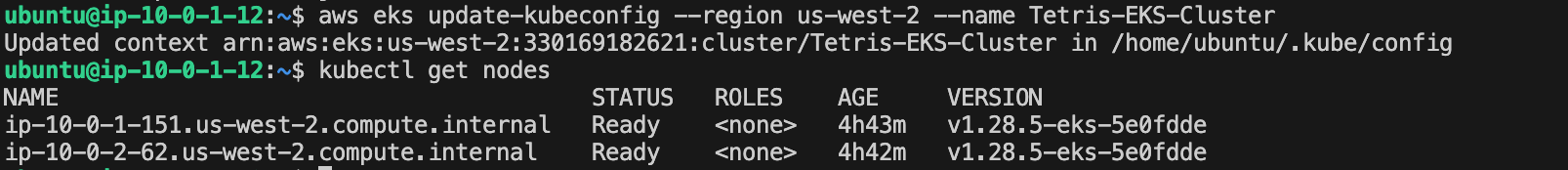
Now, we will configure the **EKS** **Cluster** on the **Jenkins Server**

Run the below command to configure the EKS Cluster

aws eks update-kubeconfig --region us-west-2 --name Tetris-EKS-Cluster

To validate whether the EKS Cluster was successfully configured or not. Run the below command

kubectl get nodes



# ****We will install ArgoCD Controller on EKS Cluster and make publicly available****

Create tetris namespace

kubectl create namespace tetris

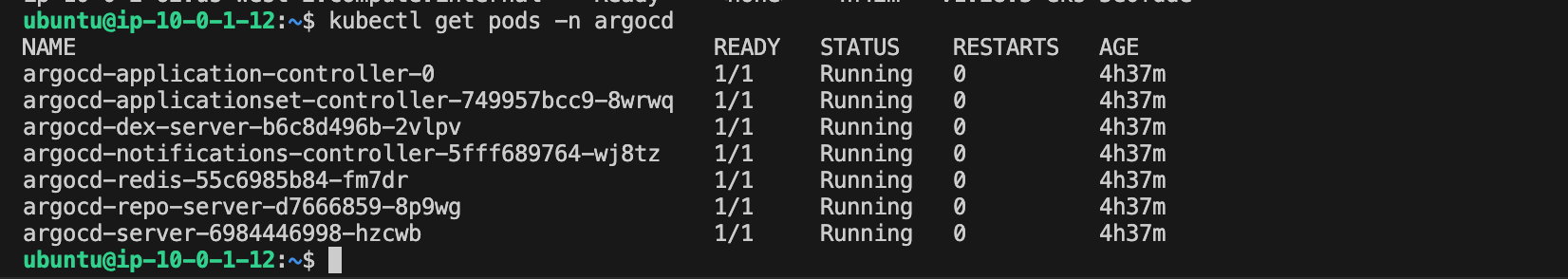
Now, we will configure the argoCD controller on our EKS cluster.

Create a new namespace named argocd and apply the manifest file on the EKS cluster.

kubectl create namespace argocd  
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/v2.4.7/manifests/install.yaml

Validate whether the argocd controller is deployed or not using the below command.

kubectl get pods -n argocd

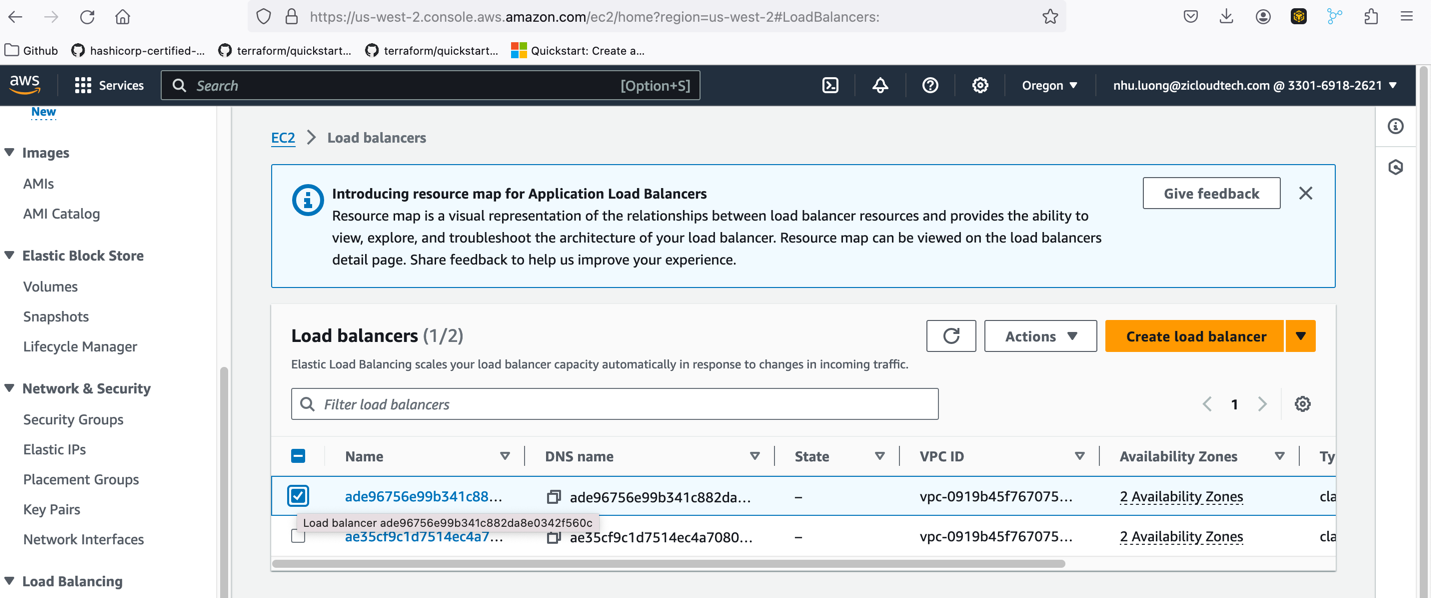


As we have to access our ArgoCD controller through GUI, we need to set up the LoadBalancer for it.

Run the below command to set up the **load balancer** which will expose the argoCD controller publicly.

kubectl patch svc argocd-server -n argocd -p ‘{“spec”: {“type”: “LoadBalancer”}}’

Now, you will see **Load Balancer** on the AWS console.



Copy the LoadBalancer DNS and paste it into your favorite browser.

<https://ade96756e99b341c882da8e0342f560c-902802644.us-west-2.elb.amazonaws.com/applications/tetris?view=tree&resource=>

Now, we need an admin password to log in to our argoCD.

There is one pre-requisite which is **jq** to get the password by using filtration.

sudo apt install jq -y

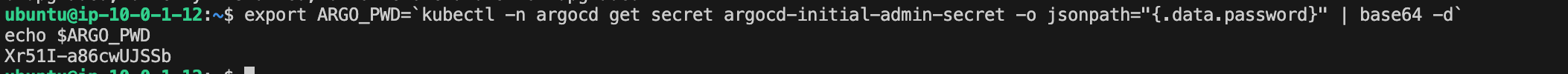


Store the ArgoCD DNS name in the variable

export ARGOCD\_SERVER=`kubectl get svc argocd-server -n argocd -o json | jq — raw-output ‘.status.loadBalancer.ingress[0].hostname’`

Run the below command to get the password.

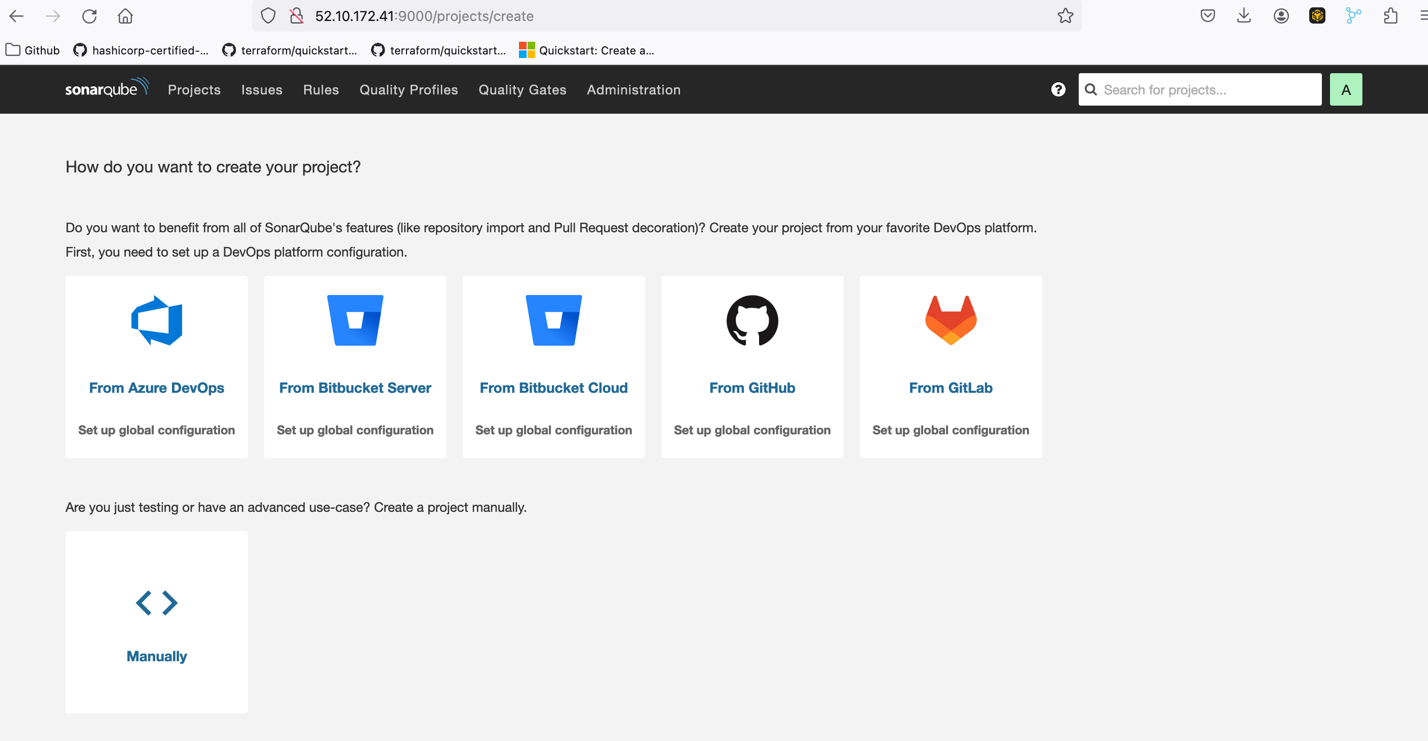
export ARGO\_PWD=`kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d`  
echo $ARGO\_PWD



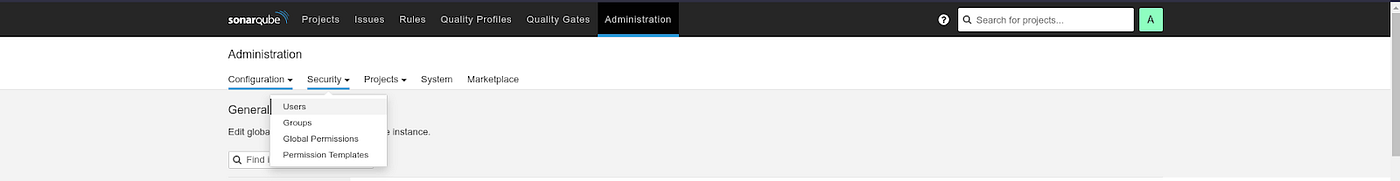
To do that, copy your Jenkins Server public IP and paste it on your favorite browser with a 9000 port

The username and password will be **admin**

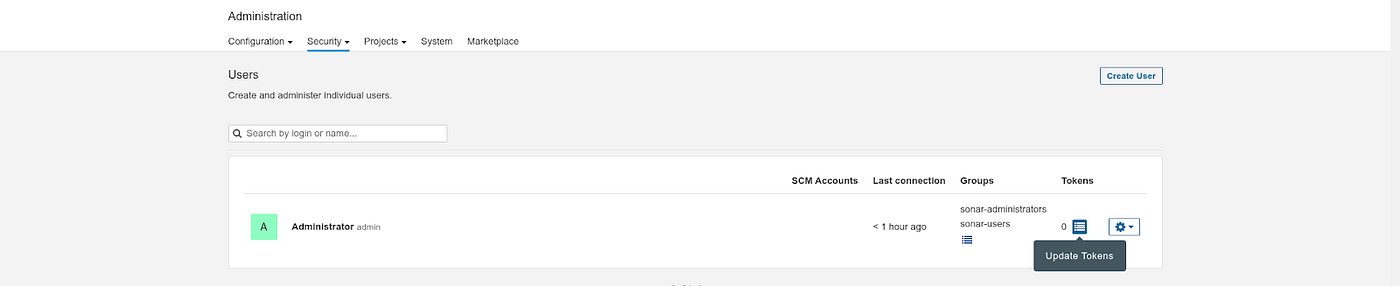
Click on **Log In.**



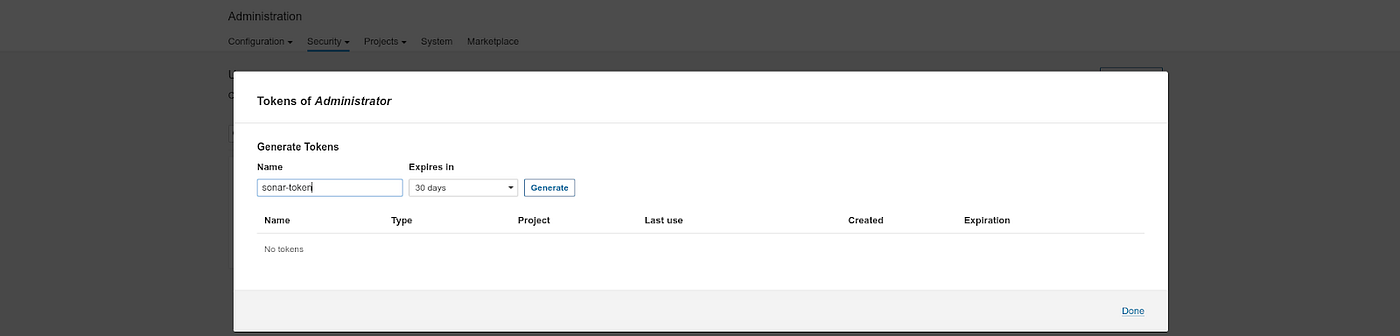
Click on **Administration** then **Security,** and select **Users**



Click on **Update tokens**



Click on **Generate**

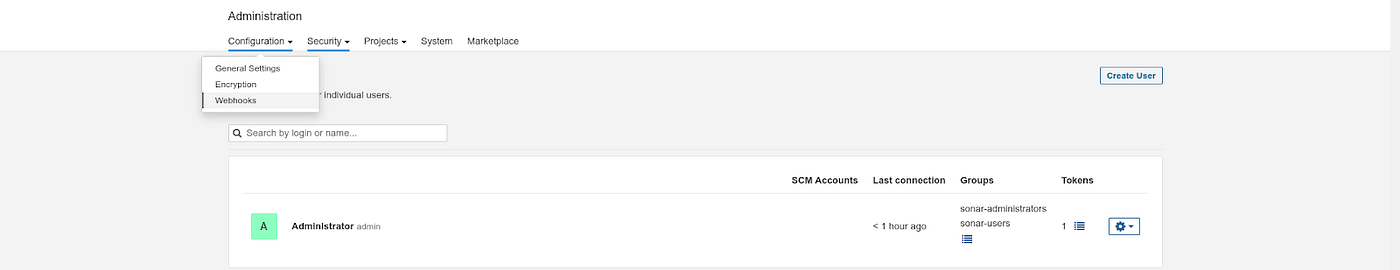


Copy the **token** and keep it somewhere safe and click on **Done.**

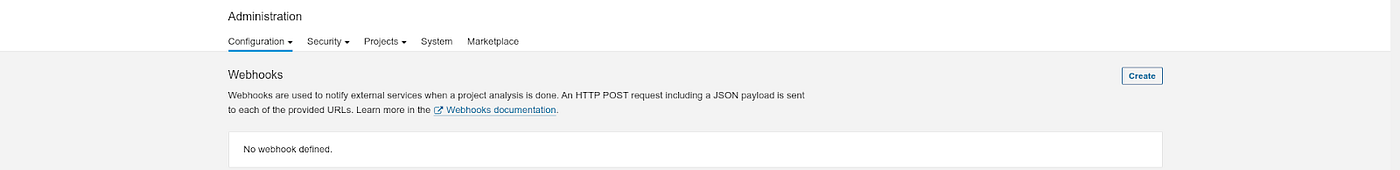


Now, we have to configure **webhooks** for quality checks.

Click on **Administration** then, **Configuration** and select **Webhooks**

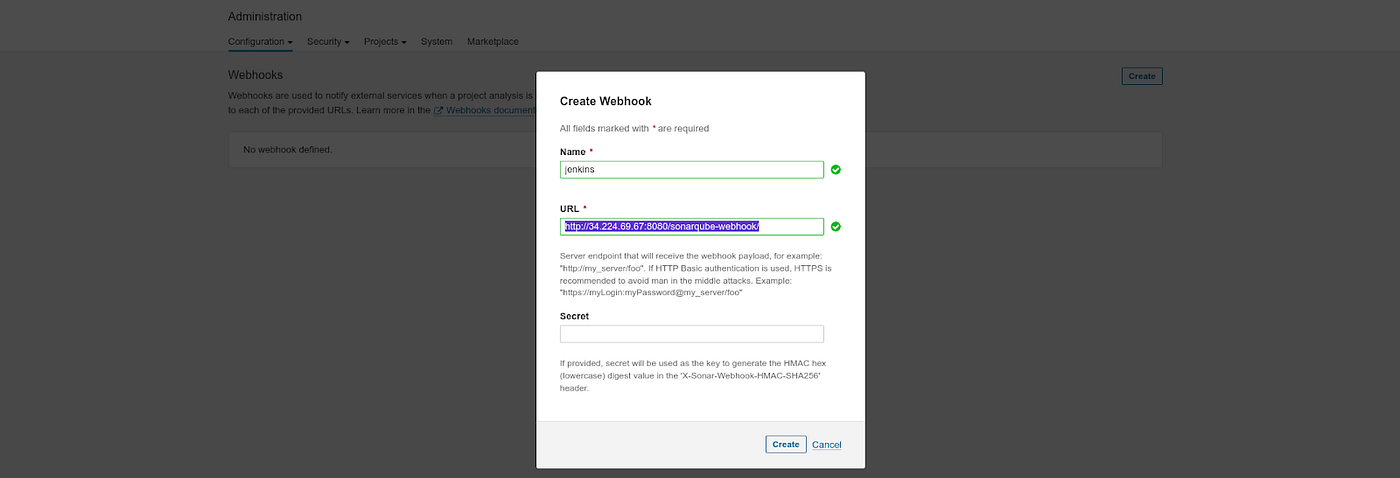


Click on **Create**

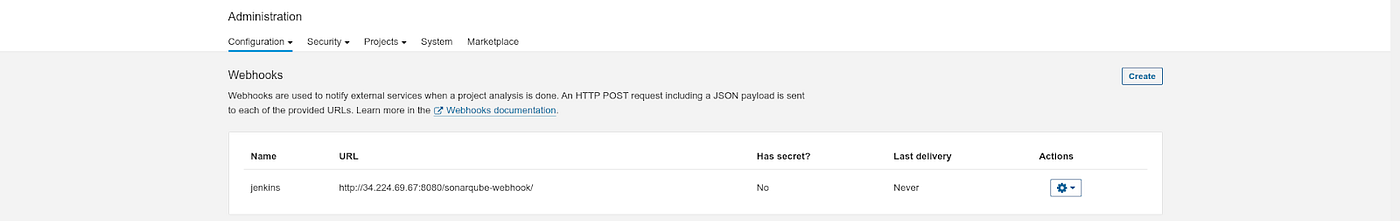


Provide the name to your project and in the **URL,** provide the jenkins server public ip with port 8080 and add sonarqube-webhook in suffix and click on **Create.**

[http://<jenkins-server-public-ip>:8080/sonarqube-webhook/](http://34.224.69.67:8080/sonarqube-webhook/)



Here, you can see the **webhook.**

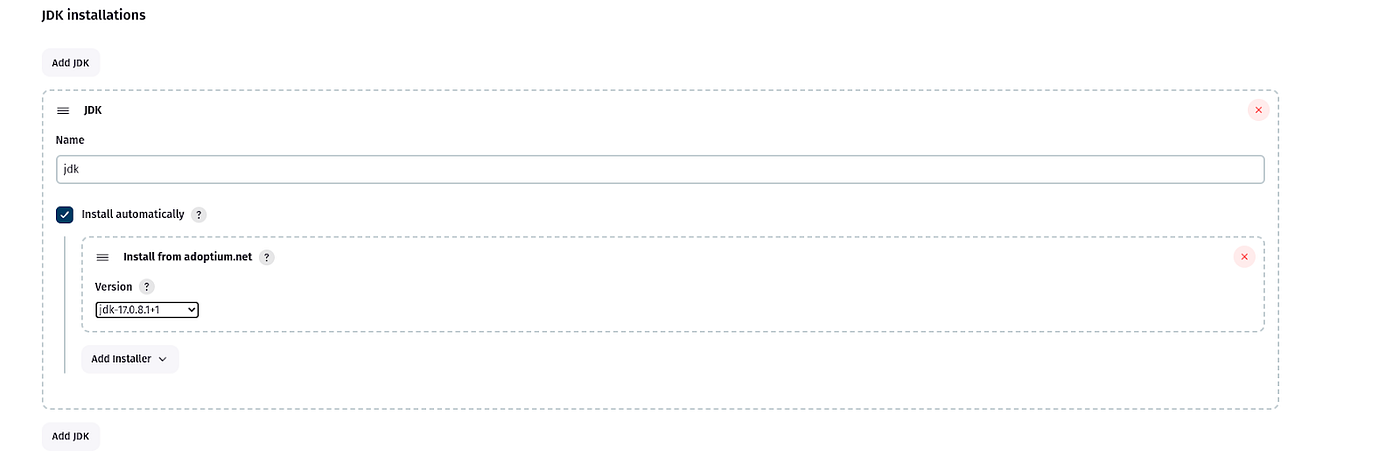


Now, we have to configure the installed plugins.

Go to **Dashboard -> Manage Jenkins -> Tools**

We are configuring jdk

Search for jdk and provide the configuration like below snippet.



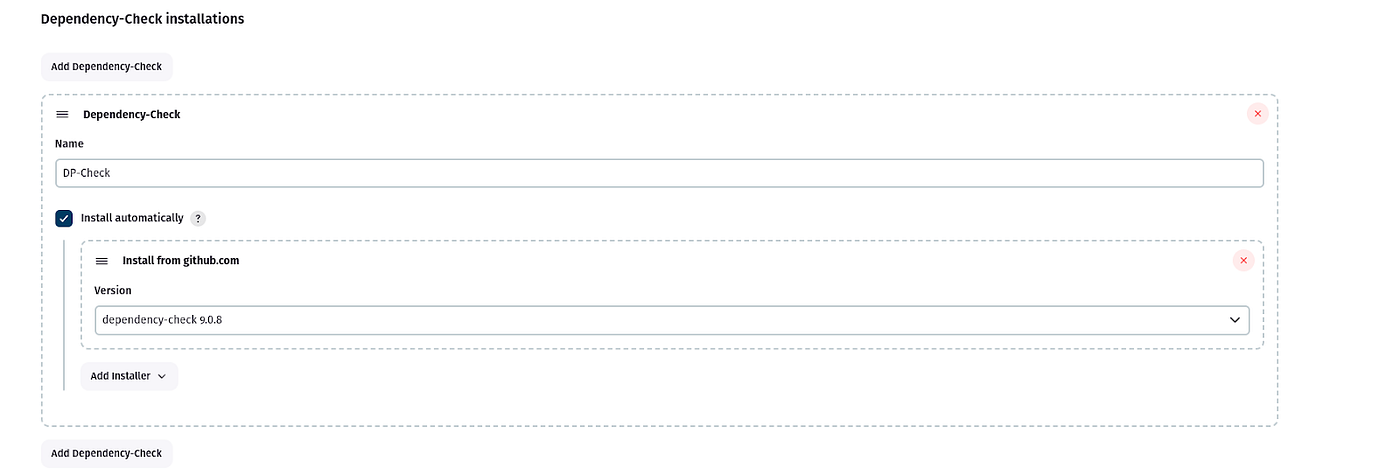
Now, we will configure nodejs

Search for node and provide the configuration like the below snippet.



Now, we will configure the OWASP Dependency check

Search for Dependency-Check and provide the configuration like the below snippet.



Now, we will configure the docker

Search for docker and provide the configuration like the below snippet.



Now, we will configure sonarqube

Search for sonarqube and provide the configuration like the below snippet.



This is it, now click on **Apply** and **Save.**

Now, we have to store our generated token for Sonarqube in Jenkins

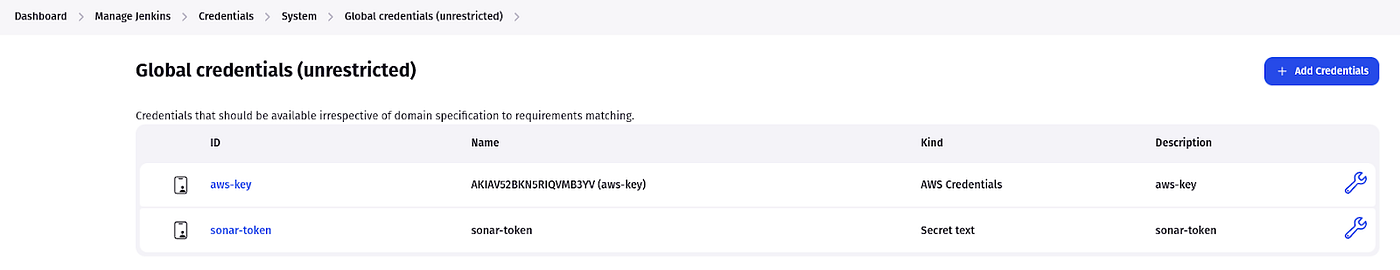
Go to **Dashboard -> Manage Jenkins -> Credentials**

Select the kind as **Secret text** and paste your token in **Secret** and keep other things as it is.

Click on **Create**



Token is created



Now, we have to set the path for Sonarqube in Jenkins

Go to **Dashboard -> Manage Jenkins -> System**

Search for **SonarQube installations**

Provide the **name** as it is, then in **Server URL** copy the sonarqube public IP (same as jenkins) with port 9000 and select the sonar token that we have added recently and click on **Apply** & **Save**.



Now, In our Jenkinsfile we are pushing our docker image to Dockerhub and then, we have to update the same image name with the new tag in the deployment file which is present on GitHub.

To do that, we need to store our Docker & GitHub credentials in Jenkins.

Go to **Dashboard -> Manage Jenkins -> Credentials**

Add your docker hub username and password in the respective field with ID **docker.**

Click on **Create**



Add GitHub credentials

Select the kind as **Secret text** and paste your GitHub Personal access token(not password) in **Secret** and keep other things as it is.

Click on **Create**

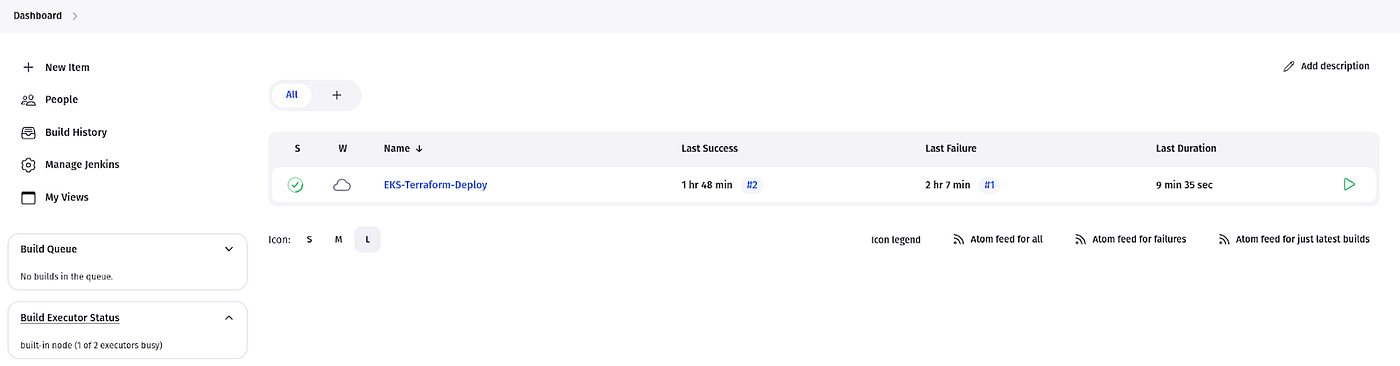
**Note:** If you haven’t generated your token then, you have it generated first then paste it in the Jenkins



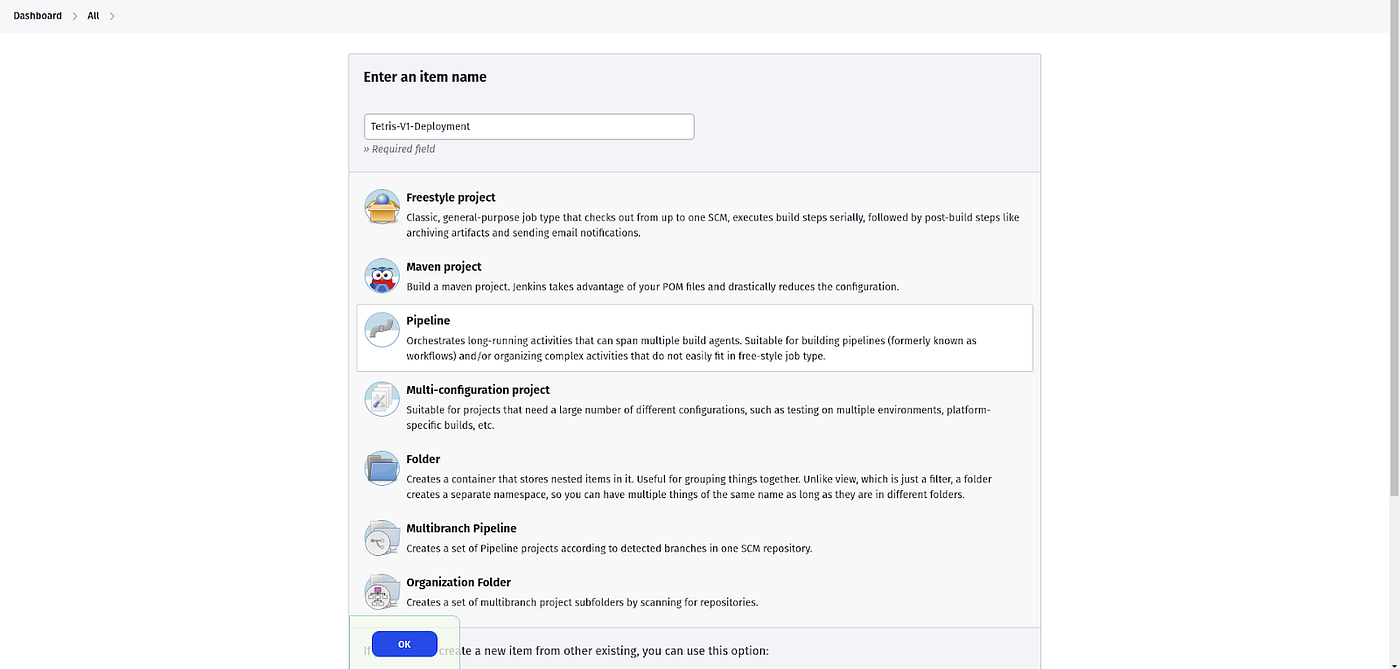
Now, we are ready to create our Jenkins Pipeline to deploy our Tetris Application.

Go to **Jenkins Dashboard**

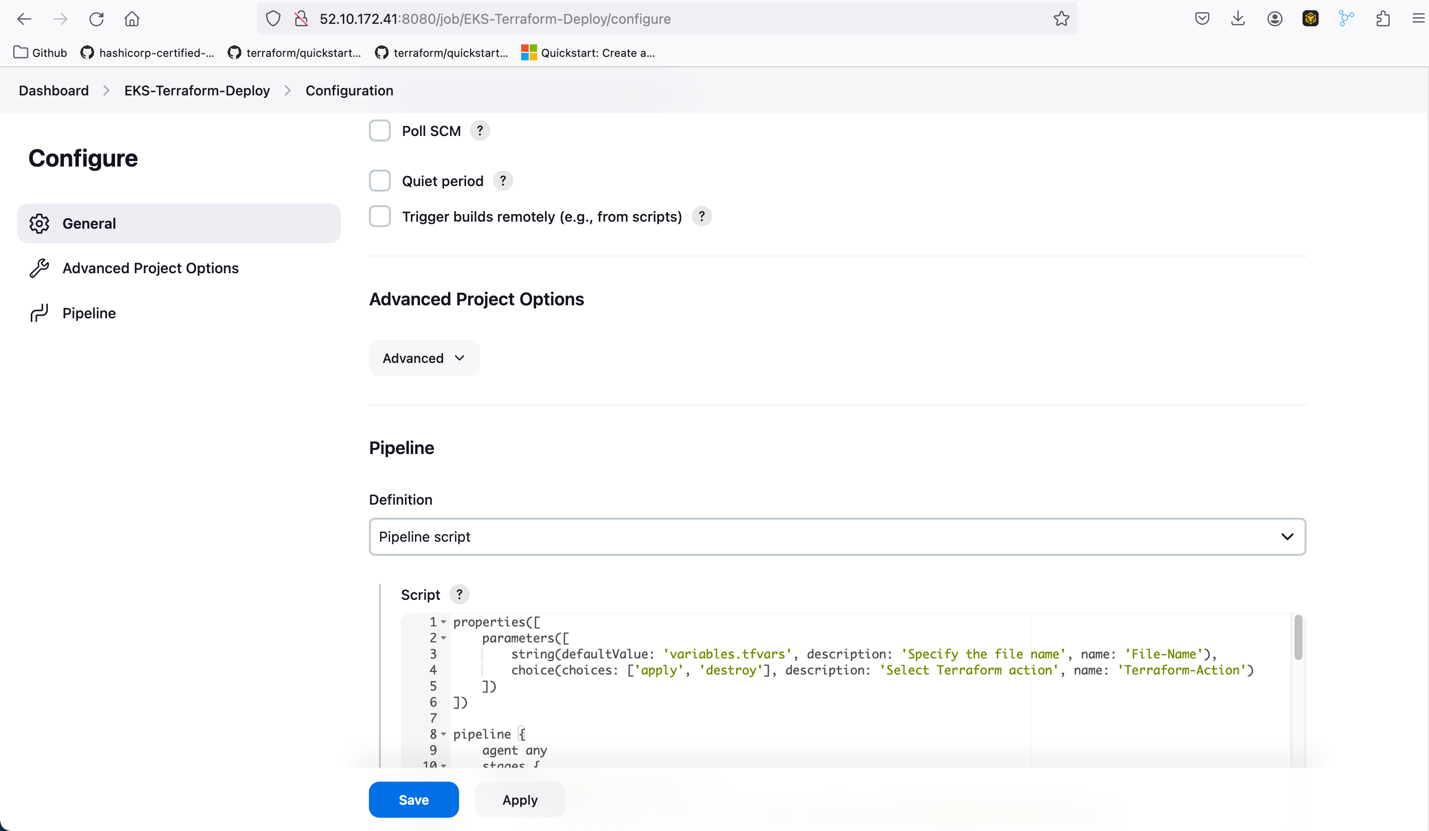
Click on **New Item**



Provide the name of your **Pipeline** and click on **OK.**

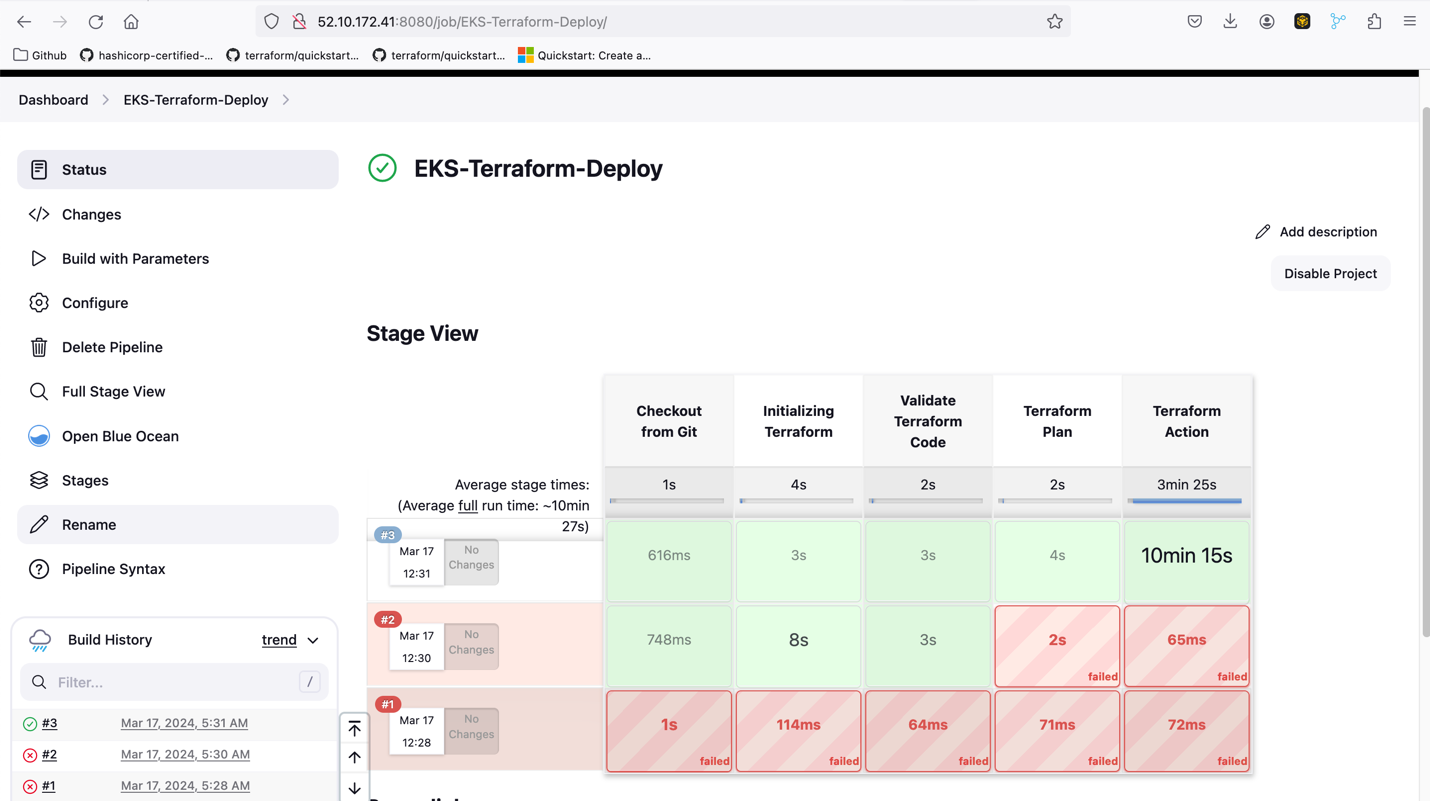


This is the Jenkinsfile to deploy Tetris Application Version 1 on EKS.

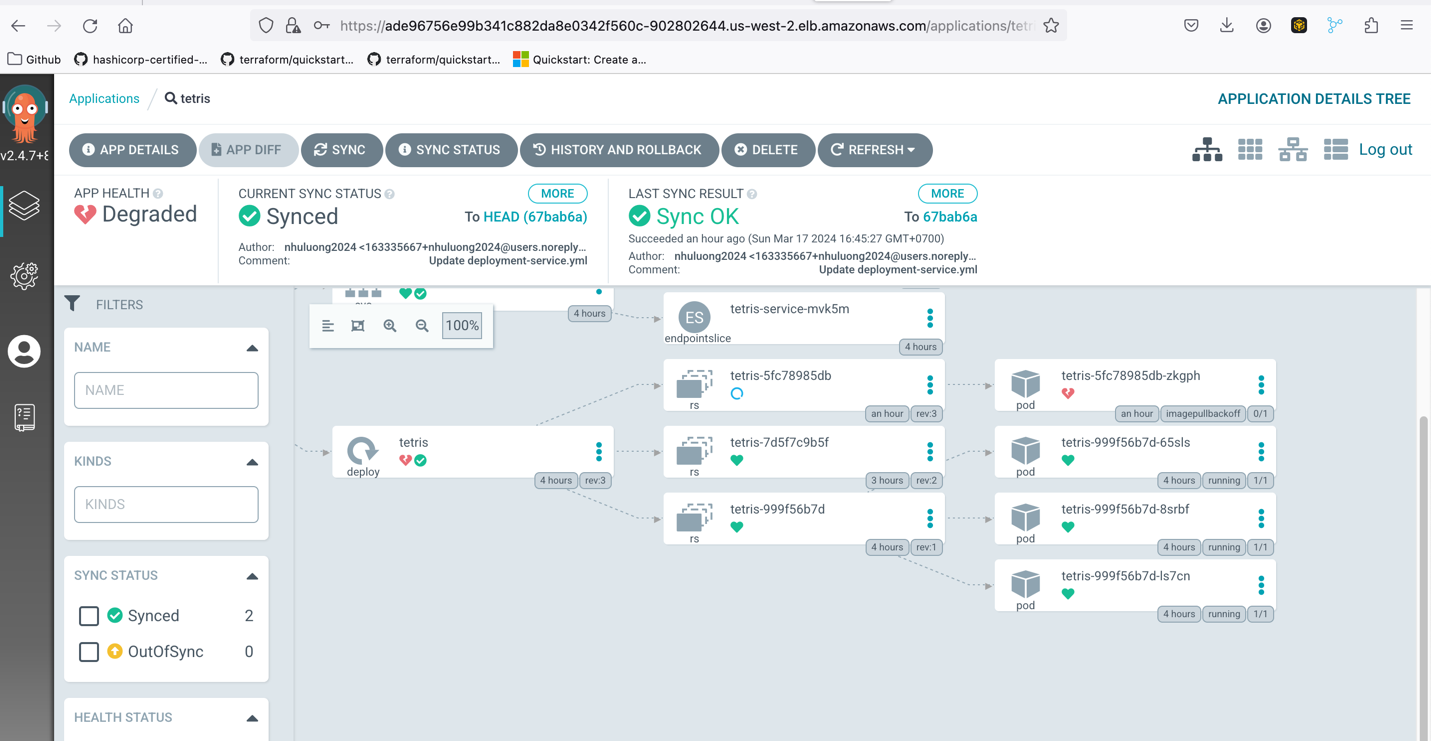


Now, click on the build.

Our pipeline was successful.



Now, Go to the **ArgoCD Console.** You will see your application has been deployed or deploying

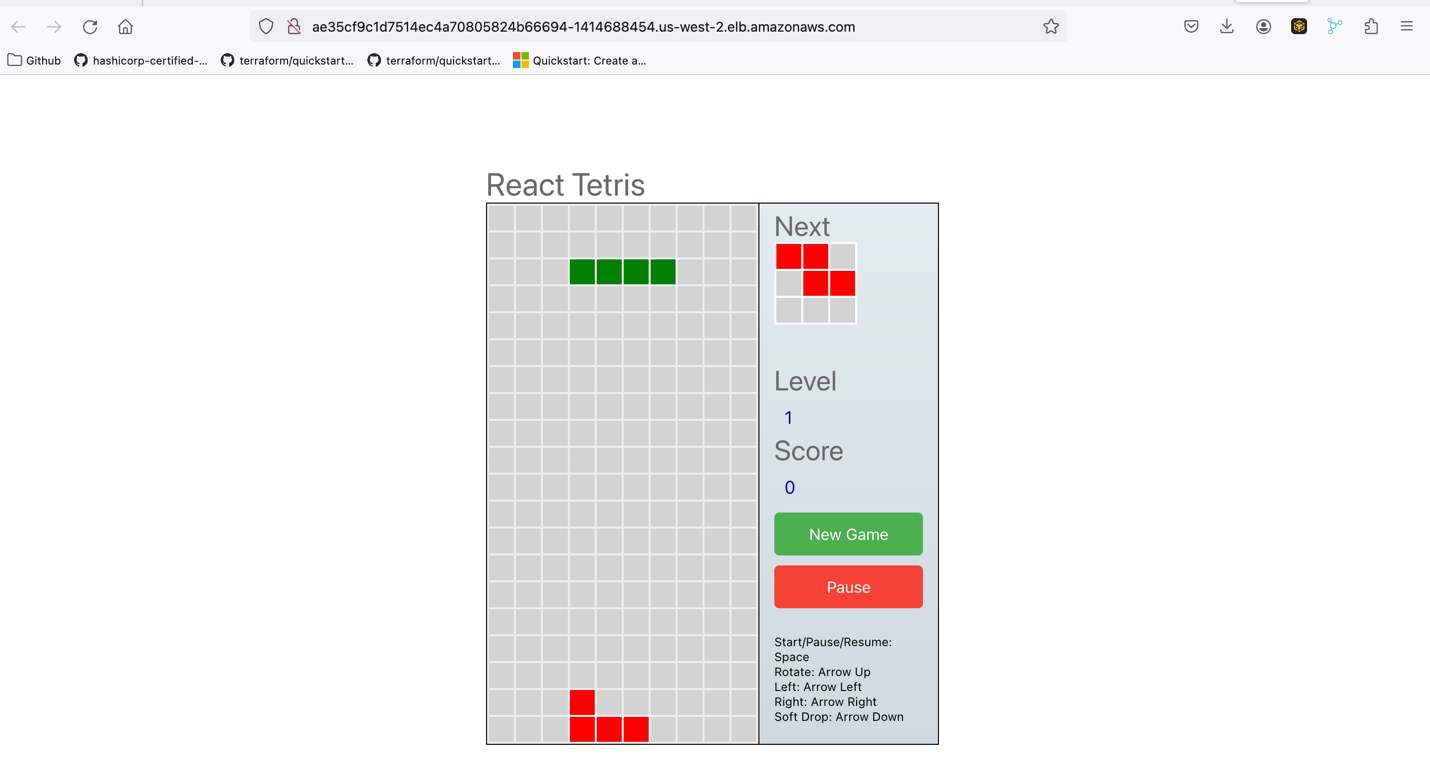


This is our Sonarqube which will show you the Code Smells and vulnerabilities in your source code.

<http://52.10.172.41:9000/projects/create>

Copy the DNS name of your load balancer from ArgoCD Console or you can go to AWS Console and copy the Load Balancer and hit the DNS on your favorite browser to enjoy the **Tetris Game**.

<http://ae35cf9c1d7514ec4a70805824b66694-1414688454.us-west-2.elb.amazonaws.com/>



# ****We will deploy our Tetris Application Version 2****

Now, suppose we have done some modifications to our previous version to make it more good in the sense of GUI or anything else. Then, we will have to deploy our **Version** **2** of our same application.

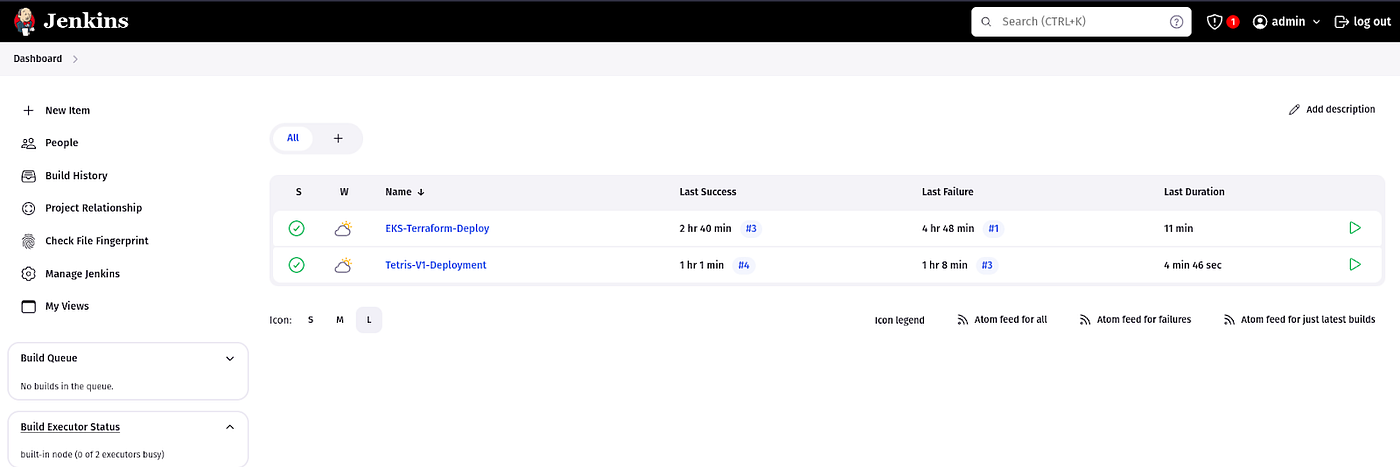
To do that, we will create a new pipeline. We can do it in the existing pipeline as well but this way you will be able to understand clearly.

We have a separate code for our Tetris Version 2. In which Dockerfile is present, so we will build the image and push it on docker and then update the same manifest file instead of v1 we will replace it with v2 manually first.

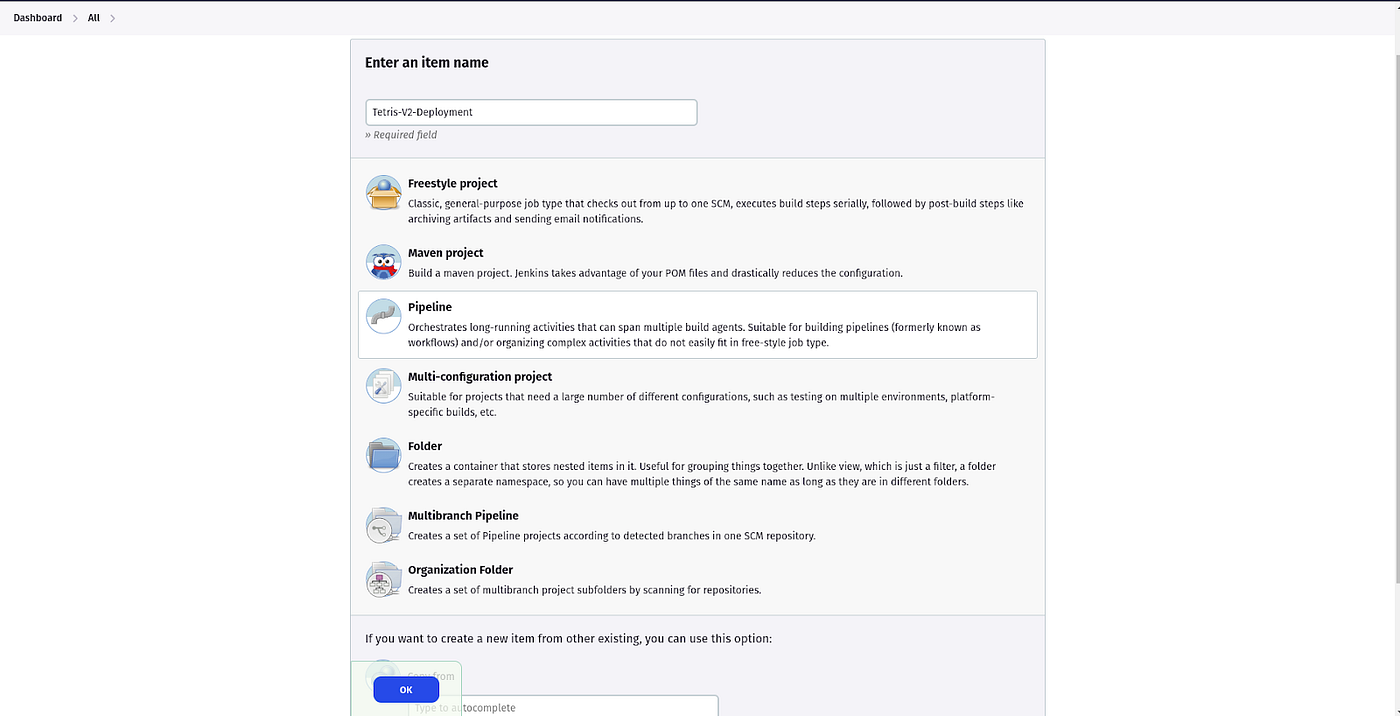
**Hope you get the high overview, what are we going to do next?**

Let’s make it and finish our project.

Go to **Jenkins** -> **Dashboard** and click on **New item**

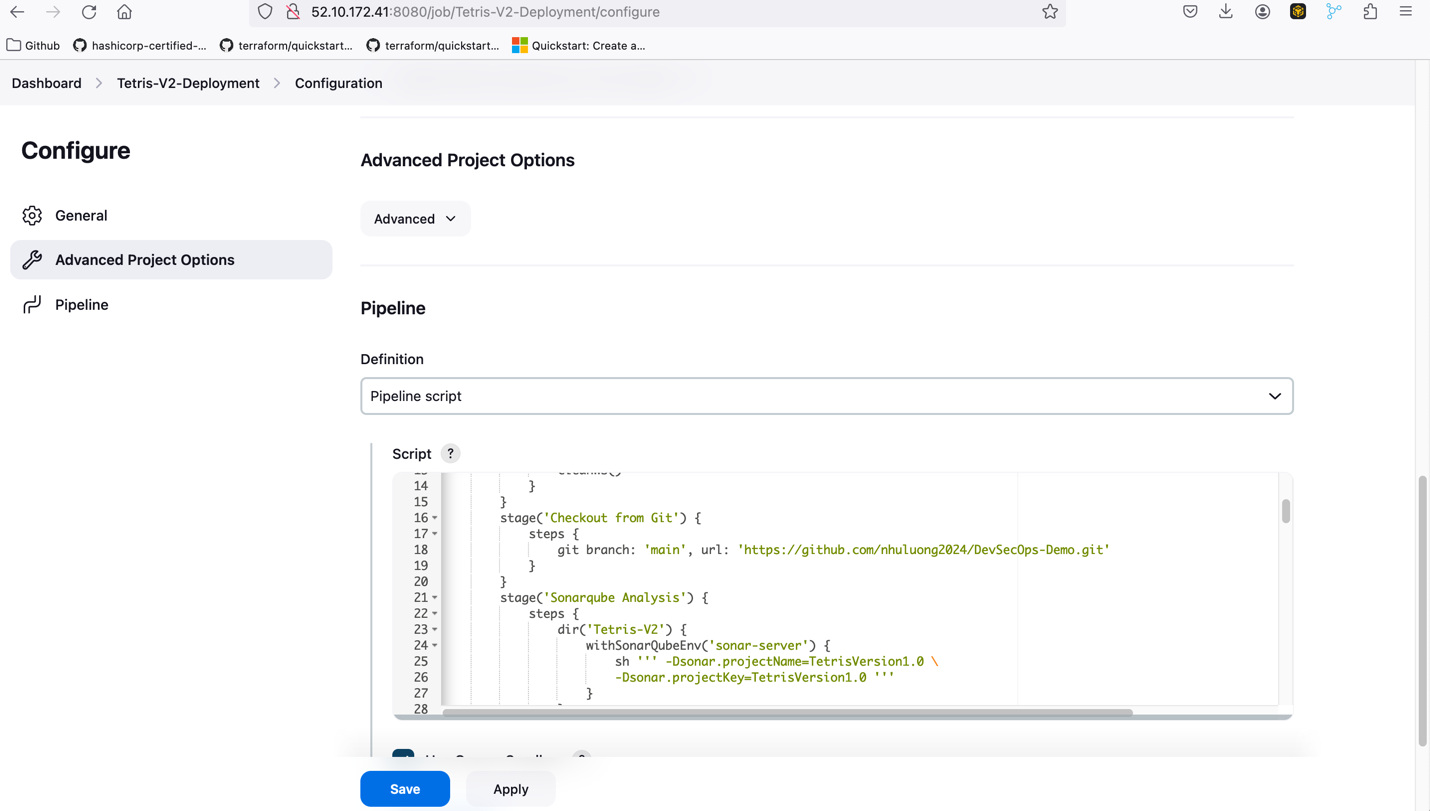


Provide the name to your **Pipeline name** and click on **OK.**



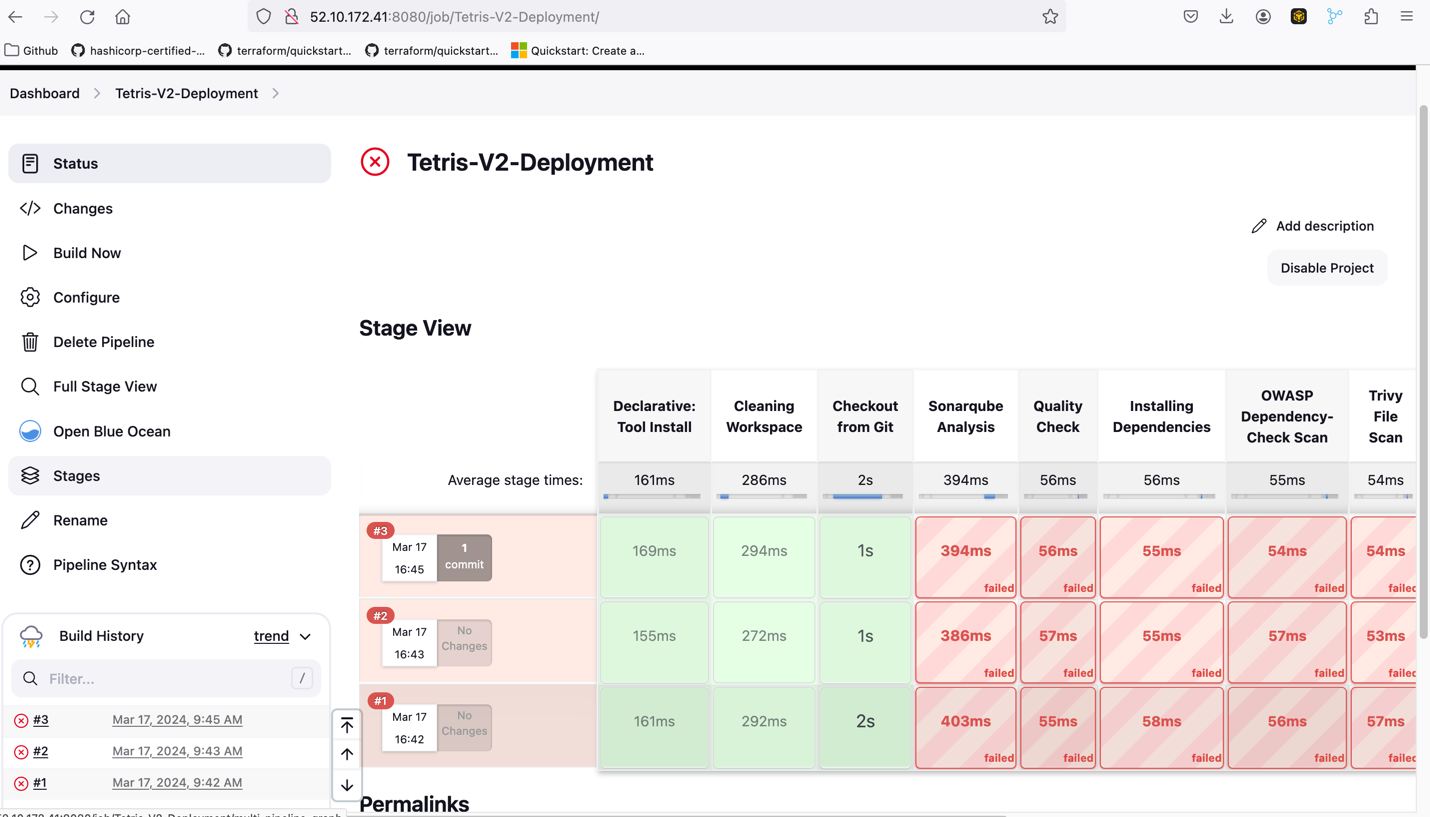
This is the Jenkinsfile to deploy Tetris Application Version 2 on EKS.

Click **Apply & Save.**



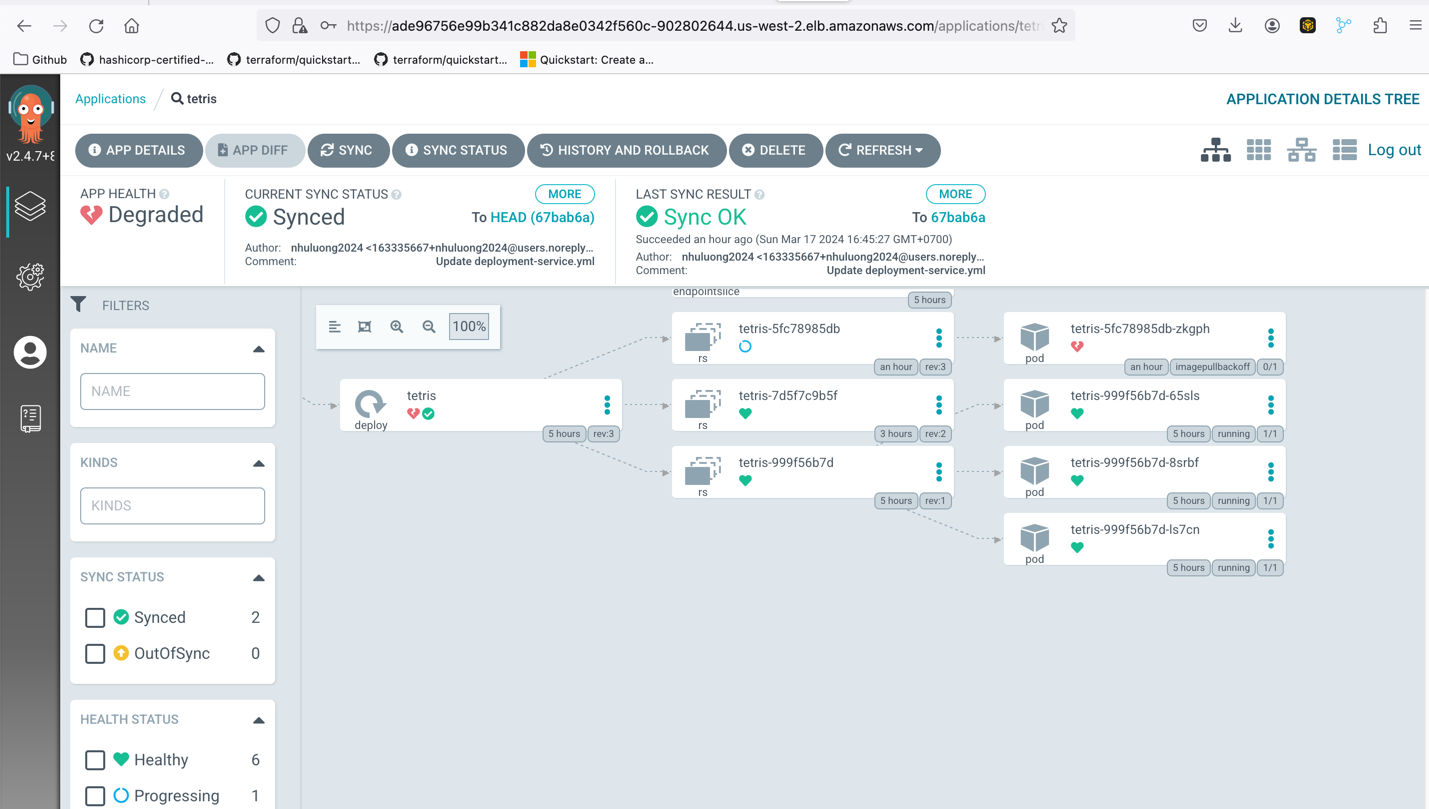
Before going to **build** the pipeline, update the manifest file.

Now, Once you click on the build to deploy our Tetris Application Version 2.   
You will see our **pipeline**

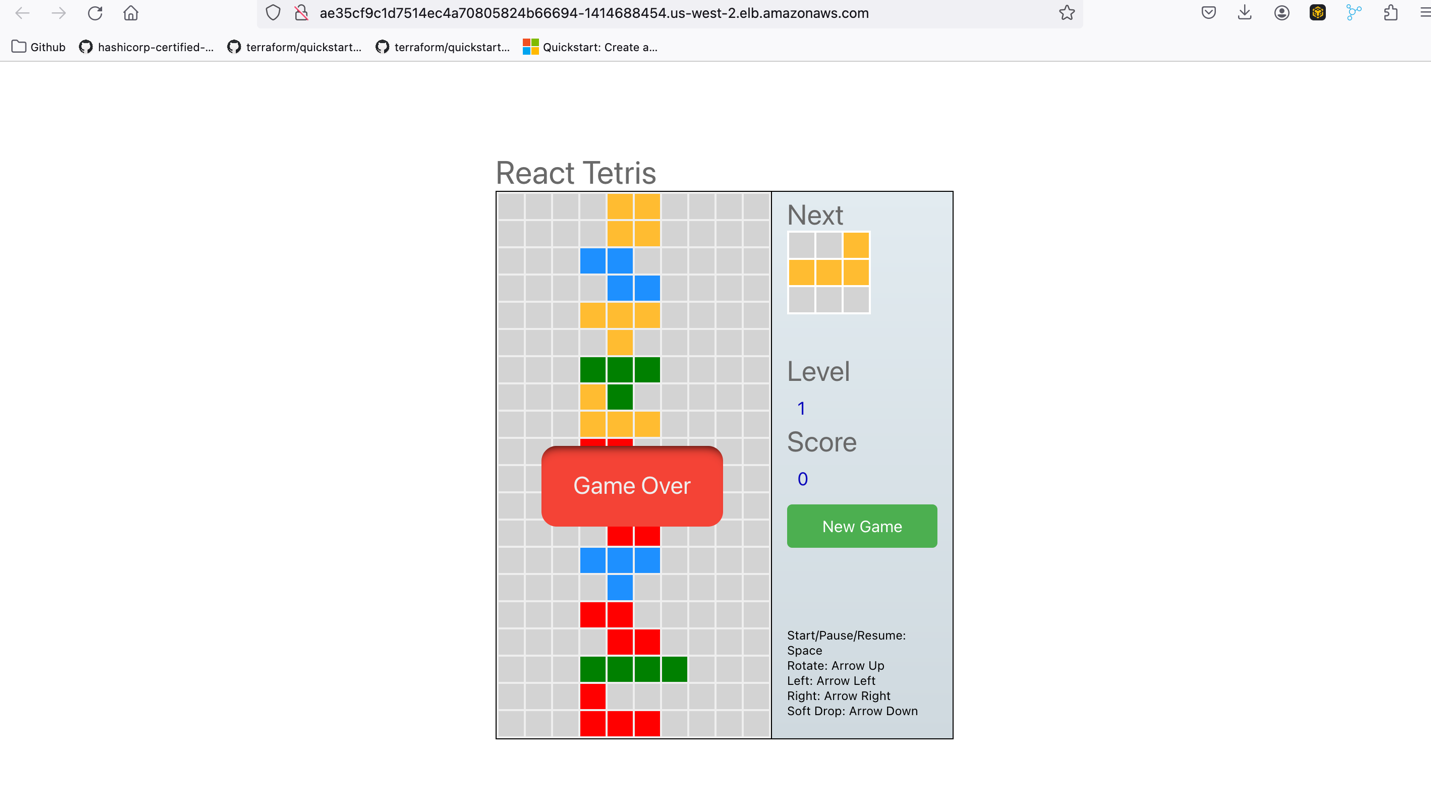


ArgoCD deployed our Application. Now, you can click on the **service** to get the **LoadBalancer** **DNS** name

<https://ade96756e99b341c882da8e0342f560c-902802644.us-west-2.elb.amazonaws.com/applications/tetris?view=tree&resource=>



Now, you can enjoy the game.



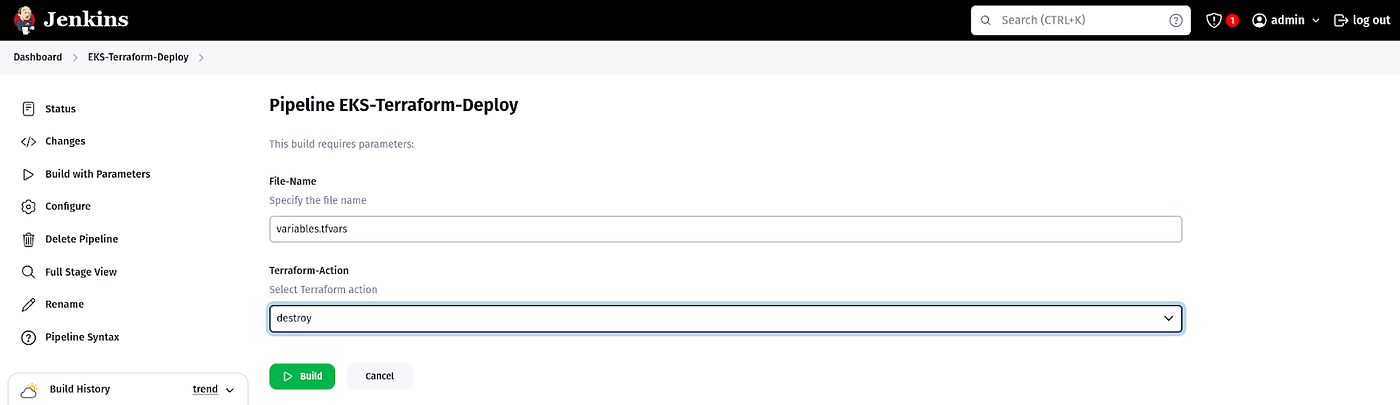
# ****Cleanup****

# ****Step 9: We will destroy the created AWS Resources****

Delete both the created **LoadBalancer** manually.

Select the **EKS-Terraform-Deploy** Pipeline.

Click on **Build with Parameters** and select the **destroy** and click on **Build.**



The Pipeline ran successfully which means the EKS Cluster has been deleted.

Now, we have to delete our **Jenkins Server**.

To do that, just run the below command on your local machine from where you create Jenkins Server.

terraform destroy -var-file=variables.tfvars --auto-approve

Happy Learning