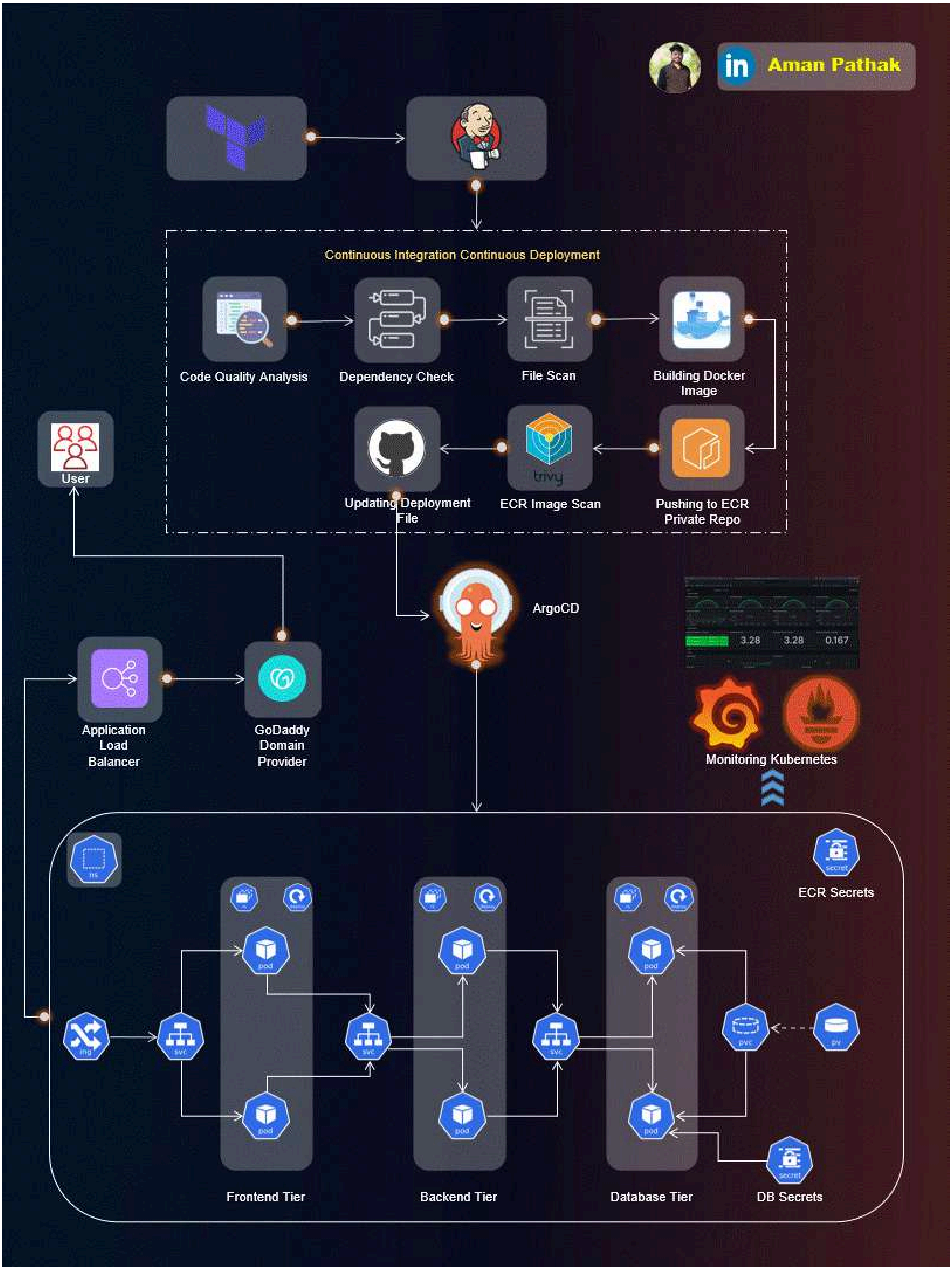
Advanced End-to-End DevSecOps Kubernetes Three-Tier Project using AWS EKS, ArgoCD, Prometheus, Grafana, and Jenkins

Aman Pathak · Follow

Published in Stackademic

24 min read · Jan 18, 2024

Listen Share



Project Introduction:

Welcome to the End-to-End DevSecOps Kubernetes Project guide! In this comprehensive project, we will walk through the process of setting up a robust Three-Tier architecture on AWS using Kubernetes, DevOps best practices, and security measures. This project aims to provide hands-on experience in deploying, securing, and monitoring a scalable application environment.

Project Overview:

In this project, we will cover the following key aspects:

1. **IAM User Setup**: Create an IAM user on AWS with the necessary permissions to facilitate deployment and management activities.
2. **Infrastructure as Code (IaC):** Use Terraform and AWS CLI to set up the Jenkins server (EC2 instance) on AWS.
3. **Jenkins Server Configuration**: Install and configure essential tools on the Jenkins server, including Jenkins itself, Docker, Sonarqube, Terraform, Kubectl, AWS CLI, and Trivy.
4. **EKS Cluster Deployment**: Utilize eksctl commands to create an Amazon EKS cluster, a managed Kubernetes service on AWS.
5. **Load Balancer Configuration**: Configure AWS Application Load Balancer (ALB) for the EKS cluster.
6. **Amazon ECR Repositories**: Create private repositories for both frontend and backend Docker images on Amazon Elastic Container Registry (ECR).
7. **ArgoCD Installation:** Install and set up ArgoCD for continuous delivery and GitOps.
8. **Sonarqube Integration**: Integrate Sonarqube for code quality analysis in the DevSecOps pipeline.
9. **Jenkins Pipelines:** Create Jenkins pipelines for deploying backend and frontend code to the EKS cluster.
10. **Monitoring Setup**: Implement monitoring for the EKS cluster using Helm, Prometheus, and Grafana.
11. **ArgoCD Application Deployment**: Use ArgoCD to deploy the Three-Tier application, including database, backend, frontend, and ingress components.
12. **DNS Configuration**: Configure DNS settings to make the application accessible via custom subdomains.

13. **Data Persistence:** Implement persistent volume and persistent volume claims for database pods to ensure data persistence.
14. **Conclusion and Monitoring:** Conclude the project by summarizing key achievements and monitoring the EKS cluster’s performance using Grafana.

Prerequisites:

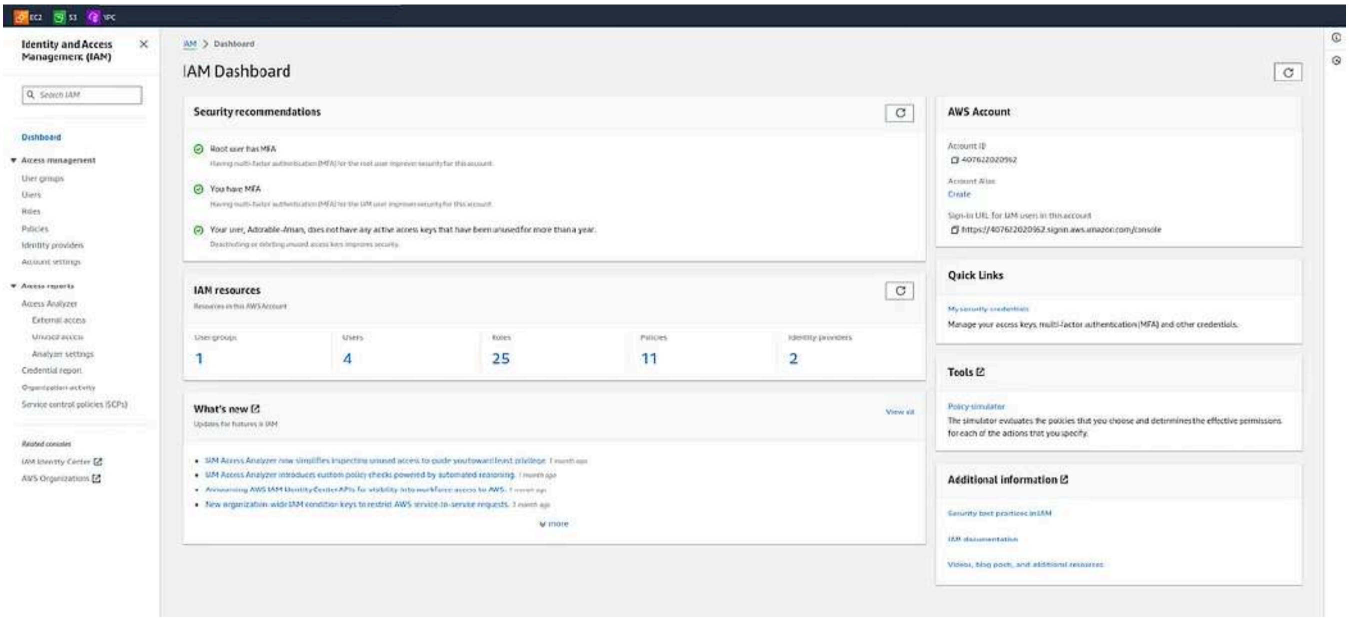
Before starting the project, ensure you have the following prerequisites:

- An AWS account with the necessary permissions to create resources.
- Terraform and AWS CLI installed on your local machine.
- Basic familiarity with Kubernetes, Docker, Jenkins, and DevOps principles.

Step 1: We need to create an IAM user and generate the AWS Access key

Create a new IAM User on AWS and give it to the AdministratorAccess for testing purposes (not recommended for your Organization's Projects)

Go to the AWS IAM Service and click on **Users.**



IAM Dashboard

Security recommendations

Root user has MFA

You have MFA

Your user, Adorable-Aman, does not have any active access keys that have been unused for more than a year.

Deactivating or deleting unused access keys improves security.

IAM resources

Resources in this AWS Account

User groups	Users	Roles	Policies	Identity providers
1	4	25	11	2

What's new

Updates for features in IAM

- IAM Access Analyzer now simplifies inspecting unused access to guide you toward least privilege. 1 month ago
- IAM Access Analyzer introduces custom policy checks powered by automated reasoning. 1 month ago
- Announcing AWS IAM Identity Center APIs for visibility into workforce access to AWS. 1 month ago
- New organization-wide IAM condition keys to restrict AWS service-to-service requests. 3 month ago

AWS Account

Account ID
407622020962

Account Alias
Create

Sign-in URL for IAM users in this account
https://407622020962.signin.aws.amazon.com/console

Quick Links

My security credentials

Manage your access keys, multi-factor authentication (MFA) and other credentials.

Tools

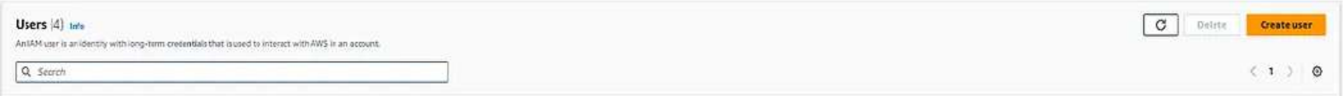
Policy simulator

The simulator evaluates the policies that you choose and determines the effective permissions for each of the actions that you specify.

Additional information

Security best practices in IAM
IAM documentation
Videos, blog posts, and additional resources

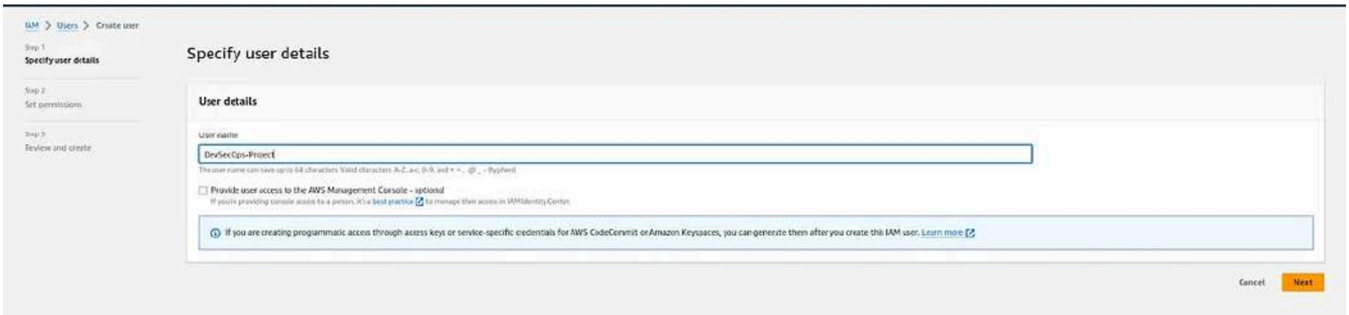
Click on **Create user**



Users (4) Info

An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Provide the name to your user and click on **Next.**



IAM > Users > Create user

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Specify user details

User details

User name
DevSecOps-Project

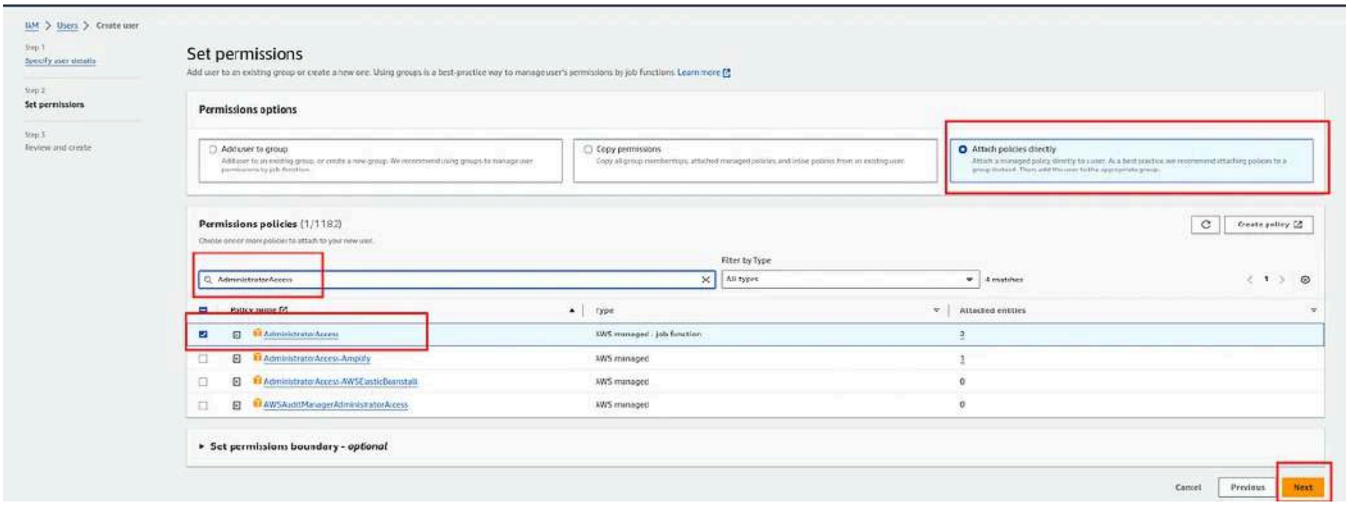
The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , . @ _ - (hyphen)

□ Provide user access to the AWS Management Console - optional
If you're providing console access to a person, it's a best practice to manage their access in IAM Identity Center.

If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more

Select the **Attach policies directly** option and search for **AdministratorAccess** then select it**.**

Click on the **Next.**



IAM > Users > Create user

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Set permissions

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. Learn more

Permissions options

○ Add user to group
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

○ Copy permissions
Copy all group memberships, attached managed policies, and inline policies from an existing user.

● Attach policies directly
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

Permissions policies (1/1182)
Choose one or more policies to attach to your new user.

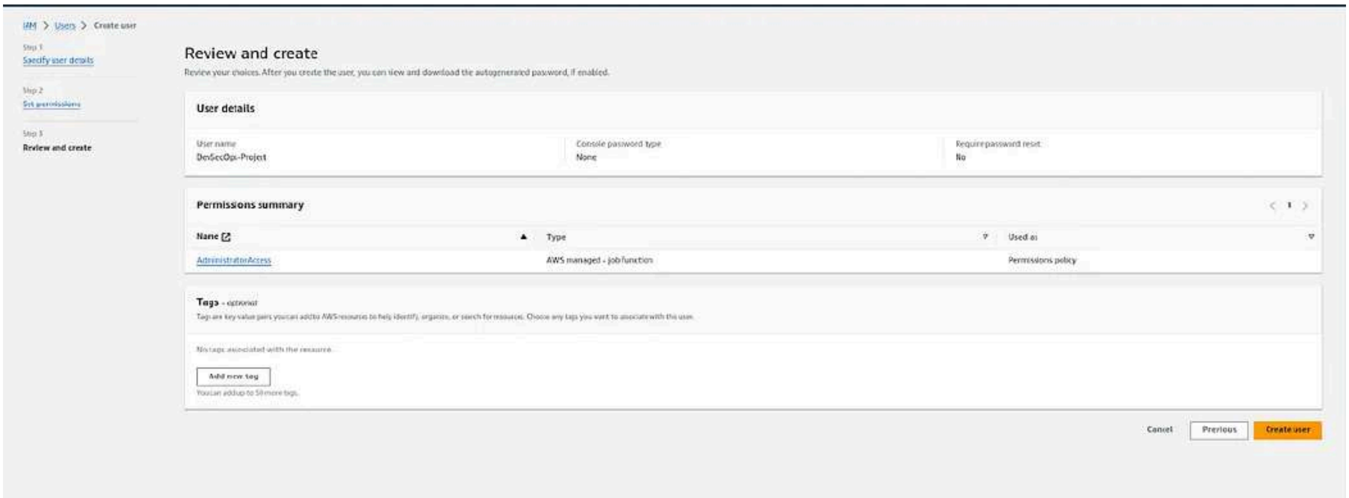
AdministratorAccess

Filter by Type
All types
4 matches

	Policy name	Type	Attached entities
☑	AdministratorAccess	AWS managed - job function	2
□	AdministratorAccess-Amplify	AWS managed	1
□	AdministratorAccess-AWSElasticBeanstalk	AWS managed	0
□	AWSAuditManagerAdministratorAccess	AWS managed	0

▸ Set permissions boundary - *optional*

Click on **Create user**



IAM > Users > Create user

Step 1
Specify user details

Step 2
Set permissions

Step 3
Review and create

Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

User details

User name	Console password type	Require password reset
DevSecOps-Project	None	No

Permissions summary

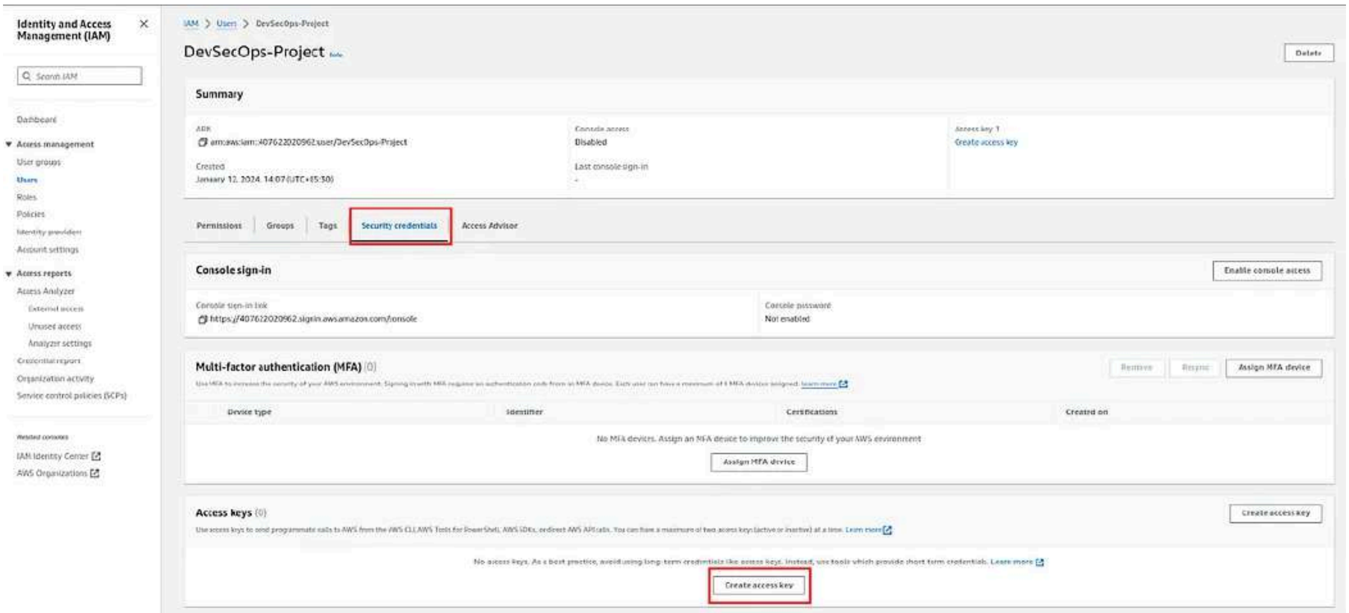
Name	Type	Used as
AdministratorAccess	AWS managed - job function	Permissions policy

Tags - optional
Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

You can add up to 50 more tags.

Now, Select your created user then click on **Security credentials** and generate access key by clicking on **Create access key.**



IAM > Users > DevSecOps-Project

DevSecOps-Project

Summary

ARN: arn:aws:iam::407622020962:user/DevSecOps-Project

Console access: Disabled

Access key 1: Create access key

Created: January 12, 2024, 14:07 (UTC+05:30)

Last console sign-in: -

Console sign-in

Console sign-in link: https://407622020962.signin.aws.amazon.com/console

Console password: Not enabled

Multi-factor authentication (MFA) (0)

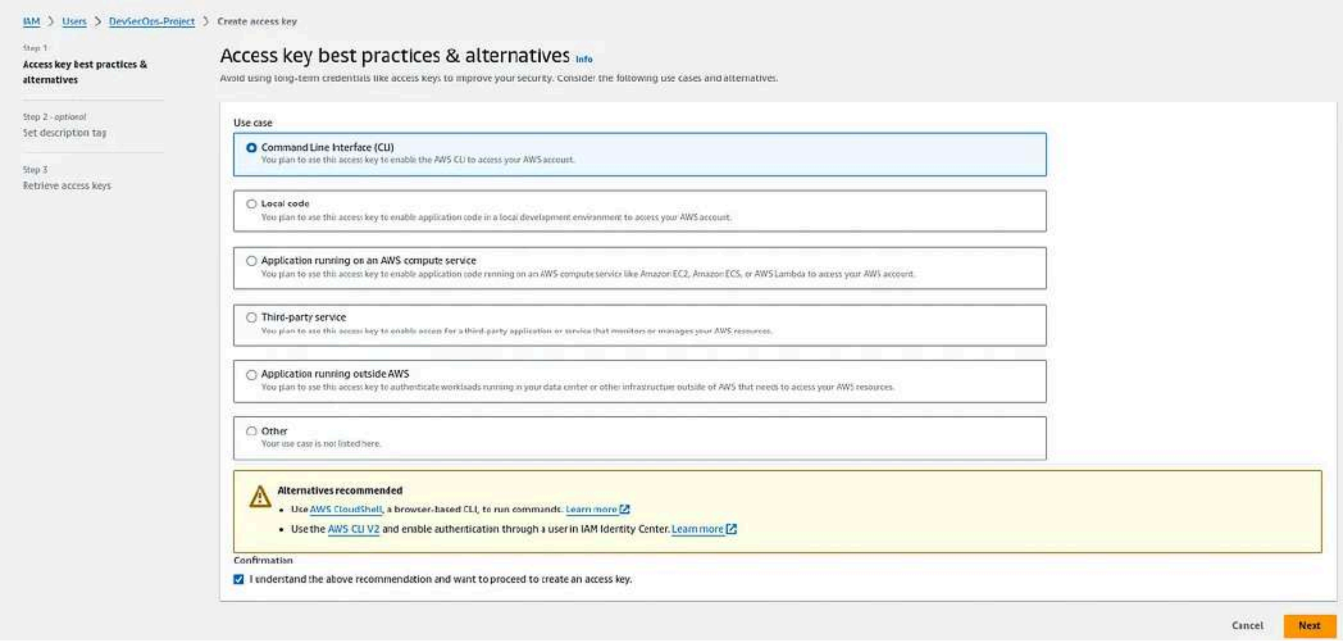
Device type	Identifier	Certifications	Created on

No MFA devices. Assign an MFA device to improve the security of your AWS environment

Access keys (0)

No access keys. As a best practice, avoid using long-term credentials like access keys. Instead, use tools which provide short term credentials. Learn more

Select the **Command Line Interface (CLI)** then select the checkmark for the confirmation and click on **Next.**



IAM > Users > DevSecOps-Project > Create access key

Access key best practices & alternatives

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

Use case

● Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.

○ Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.

○ Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

○ Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

○ Application running outside AWS
You plan to use this access key to authenticate workloads running in your data center or other infrastructure outside of AWS that needs to access your AWS resources.

○ Other
Your use case is not listed here.

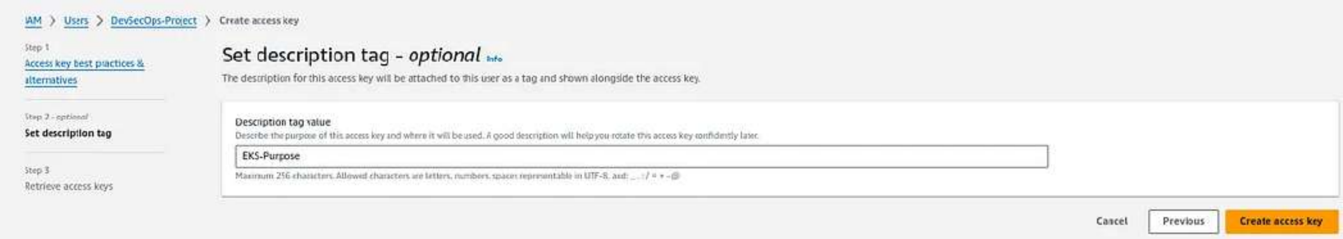
Alternatives recommended

- Use AWS CloudShell, a browser-based CLI, to run commands. Learn more
- Use the AWS CLI V2 and enable authentication through a user in IAM Identity Center. Learn more

Confirmation

☑ I understand the above recommendation and want to proceed to create an access key.

Provide the **Description** and click on the **Create access key.**



IAM > Users > DevSecOps-Project > Create access key

Set description tag - *optional*

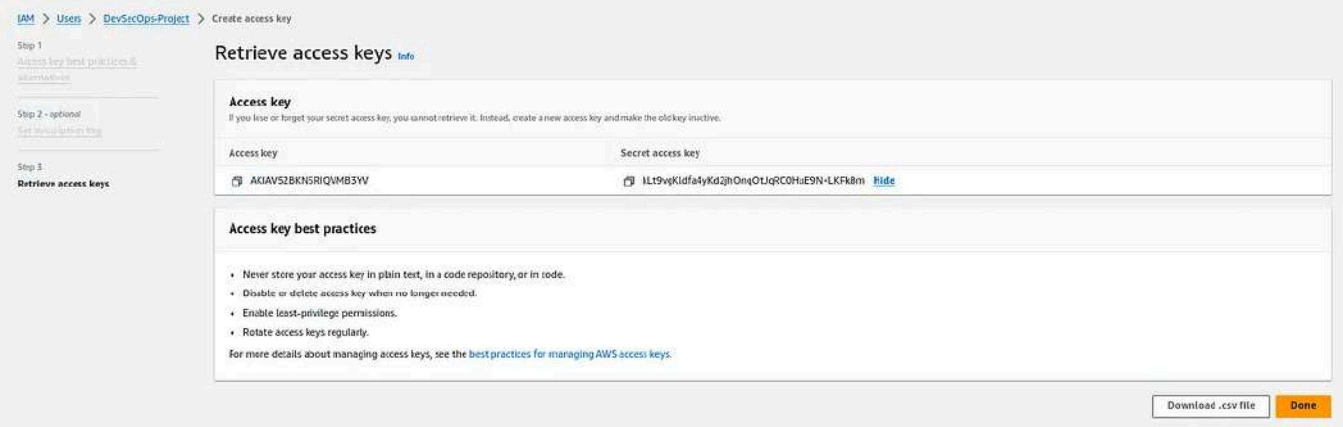
The description for this access key will be attached to this user as a tag and shown alongside the access key.

Description tag value
Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

EKS-Purpose

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . : / = + - @

Here, you will see that you got the credentials and also you can download the CSV file for the future.



IAM > Users > DevSecOps-Project > Create access key

Step 1
Access key best practices & alternatives

Step 2 - optional
Set description tag

Step 3
Retrieve access keys

Retrieve access keys Info

Access key
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIAV52BKN5RIQVMB3YV	kLt9vqKldfa4yKd2jhOnqOtJqRC0HaE9N+LKFk8m Hide

Access key best practices

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the best practices for managing AWS access keys.

Step 2: 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/sh
echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https:/
sudo apt update
sudo apt install terraform -y
```

AWSCLI Installation Script

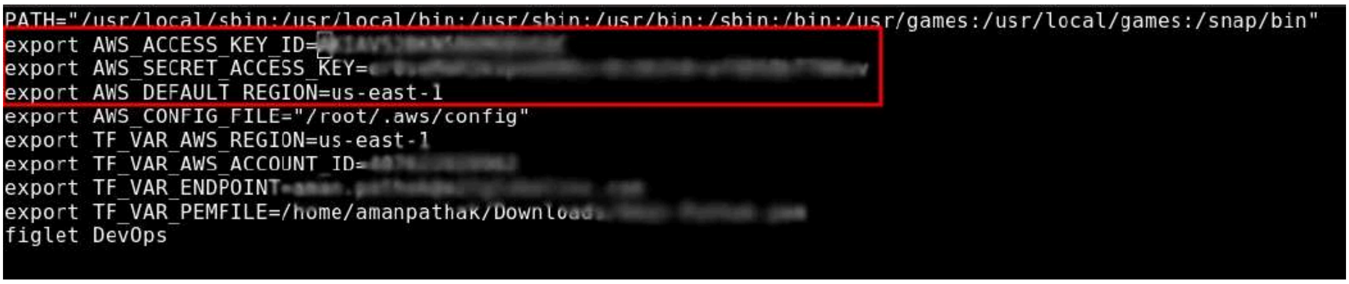
```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zi
sudo apt install unzip -y
unzip awscliv2.zip
sudo ./aws/install
```

Now, Configure both the tools

Configure Terraform

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

```
sudo vim /etc/environment
```



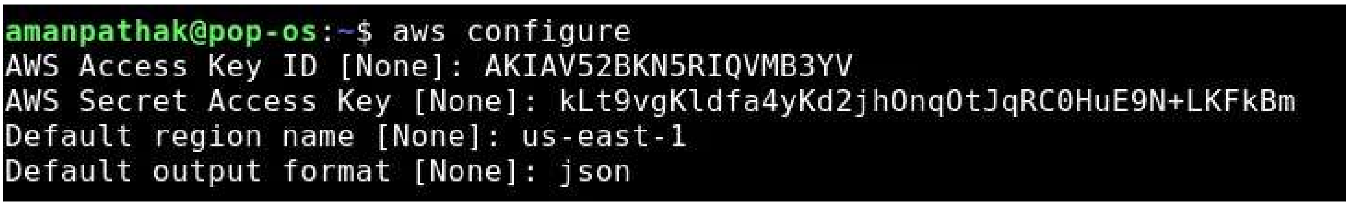
```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin"
export AWS_ACCESS_KEY_ID=[redacted]
export AWS_SECRET_ACCESS_KEY=[redacted]
export AWS_DEFAULT_REGION=us-east-1
export AWS_CONFIG_FILE="/root/.aws/config"
export TF_VAR_AWS_REGION=us-east-1
export TF_VAR_AWS_ACCOUNT_ID=[redacted]
export TF_VAR_ENDPOINT=[redacted]
export TF_VAR_PEMFILE=/home/amanpathak/Download[redacted]
figlet DevOps
```

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
```



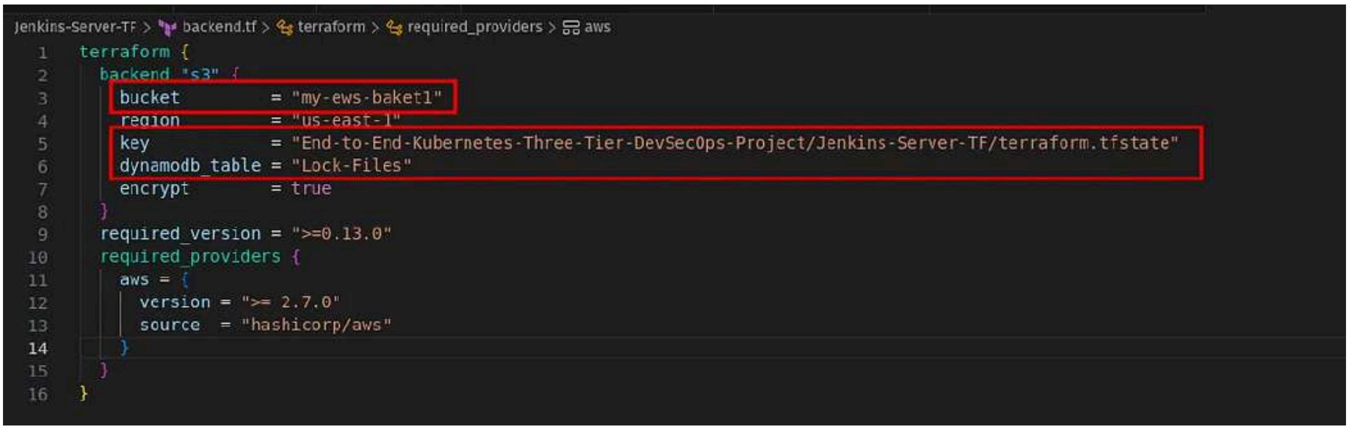
```
amanpathak@pop-os:~$ aws configure
AWS Access Key ID [None]: AKIAV52BKN5RIQVMB3YV
AWS Secret Access Key [None]: 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
Default region name [None]: us-east-1
Default output format [None]: json
```

Step 3: Deploy the Jenkins Server(EC2) using Terraform

Clone the Git repository- https://github.com/AmanPathak-DevOps/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project

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

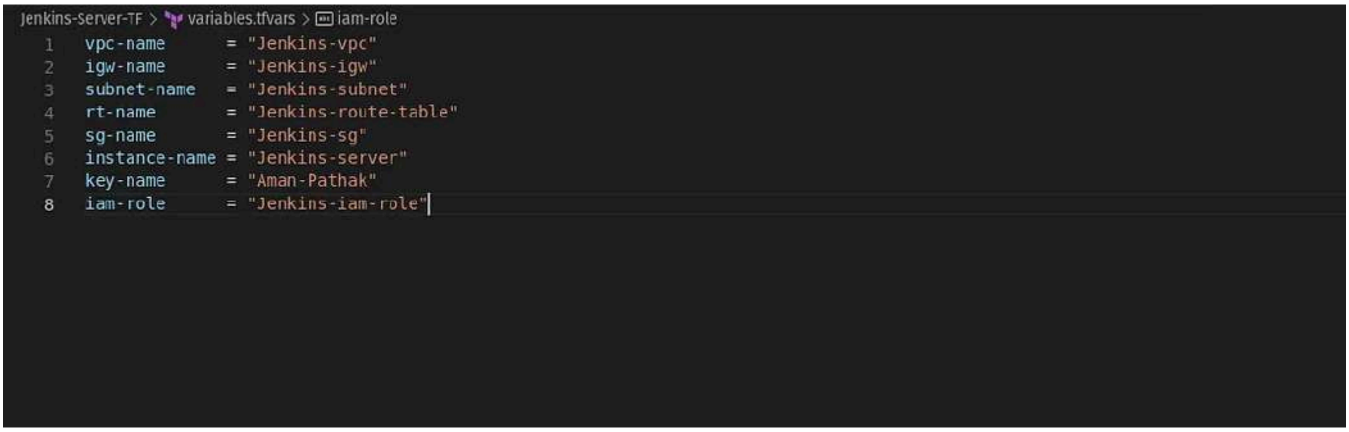
Do some modifications to the backend.tf file such as changing the **bucket** name and **dynamodb** table(make sure you have created both manually on AWS Cloud).



Jenkins-Server-TF > backend.tf > terraform > required_providers > aws

```
terraform {
  backend "s3" {
    bucket         = "my-ews-baket1"
    region         = "us-east-1"
    key            = "End-to-End-Kubernetes-Three-Tier-DevSecOps-Project/Jenkins-Server-TF/terraform.tfstate"
    dynamodb_table = "Lock-Files"
    encrypt        = true
  }
  required_version = ">=0.13.0"
  required_providers {
    aws = {
      version = ">= 2.7.0"
      source  = "hashicorp/aws"
    }
  }
}
```

Now, you have to replace the Pem File name as you have some other name for your Pem file. To provide the Pem file name that is already created on AWS

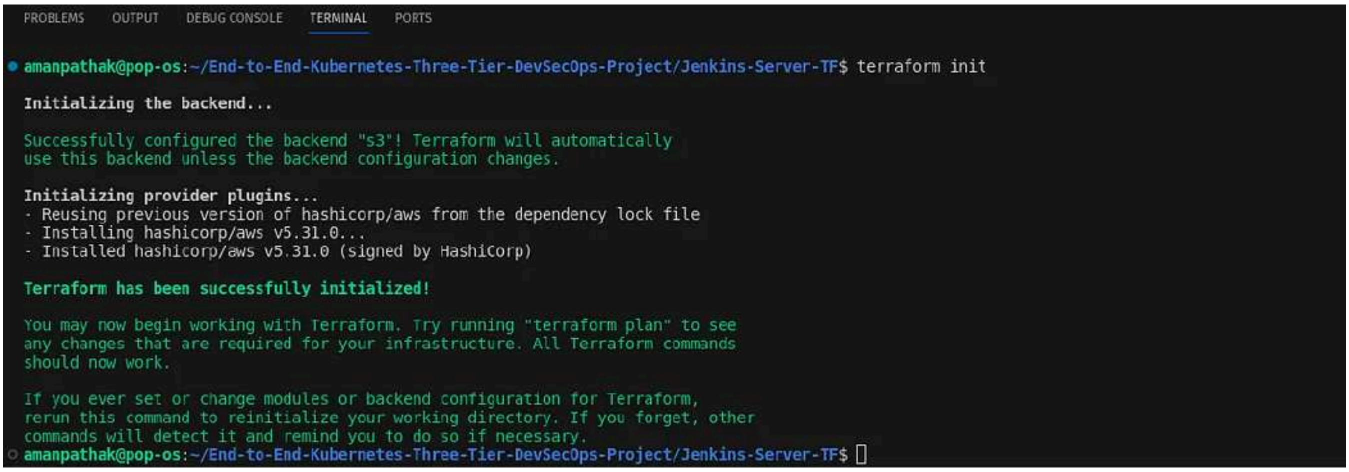


Jenkins-Server-TF > variables.tfvars > iam-role

```
vpc-name      = "Jenkins-vpc"
igw-name      = "Jenkins-igw"
subnet-name   = "Jenkins-subnet"
rt-name       = "Jenkins-route-table"
sg-name       = "Jenkins-sg"
instance-name = "Jenkins-server"
key-name      = "Aman-Pathak"
iam-role      = "Jenkins-iam-role"
```

Initialize the backend by running the below command

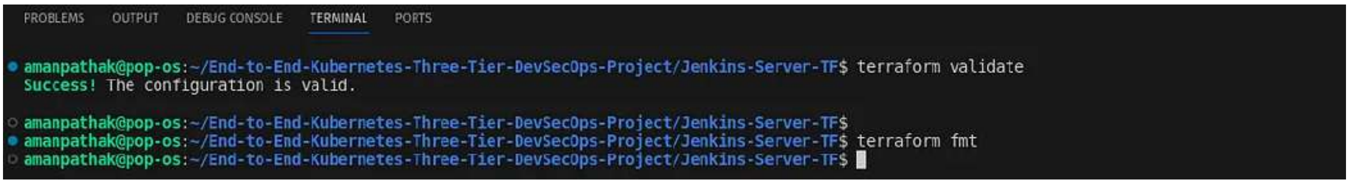
```
terraform init
```



```
amanpathak@pop-os:~/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project/Jenkins-Server-TF$ terraform init

Initializing the backend...

Successfully configured the backend "s3"! Terraform will automatically
use this backend unless the backend configuration changes.

Initializing provider plugins...
- Reusing previous version of hashicorp/aws from the dependency lock file
- Installing hashicorp/aws v5.31.0...
- Installed hashicorp/aws v5.31.0 (signed by HashiCorp)

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
amanpathak@pop-os:~/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project/Jenkins-Server-TF$
```

Run the below command to check the syntax error

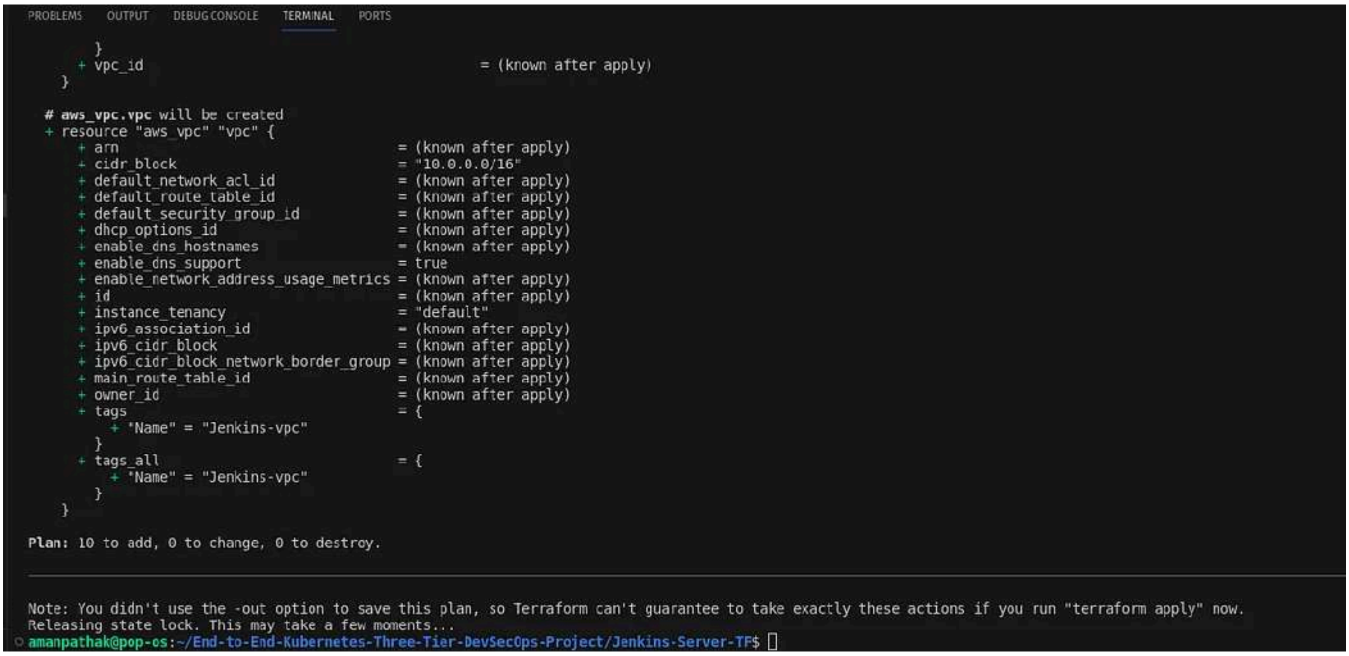
```
terraform validate
```



```
amanpathak@pop-os:~/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project/Jenkins-Server-TF$ terraform validate
Success! The configuration is valid.

amanpathak@pop-os:~/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project/Jenkins-Server-TF$
amanpathak@pop-os:~/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project/Jenkins-Server-TF$ terraform fmt
amanpathak@pop-os:~/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project/Jenkins-Server-TF$
```

Run the below command to get the blueprint of what kind of AWS services will be created.

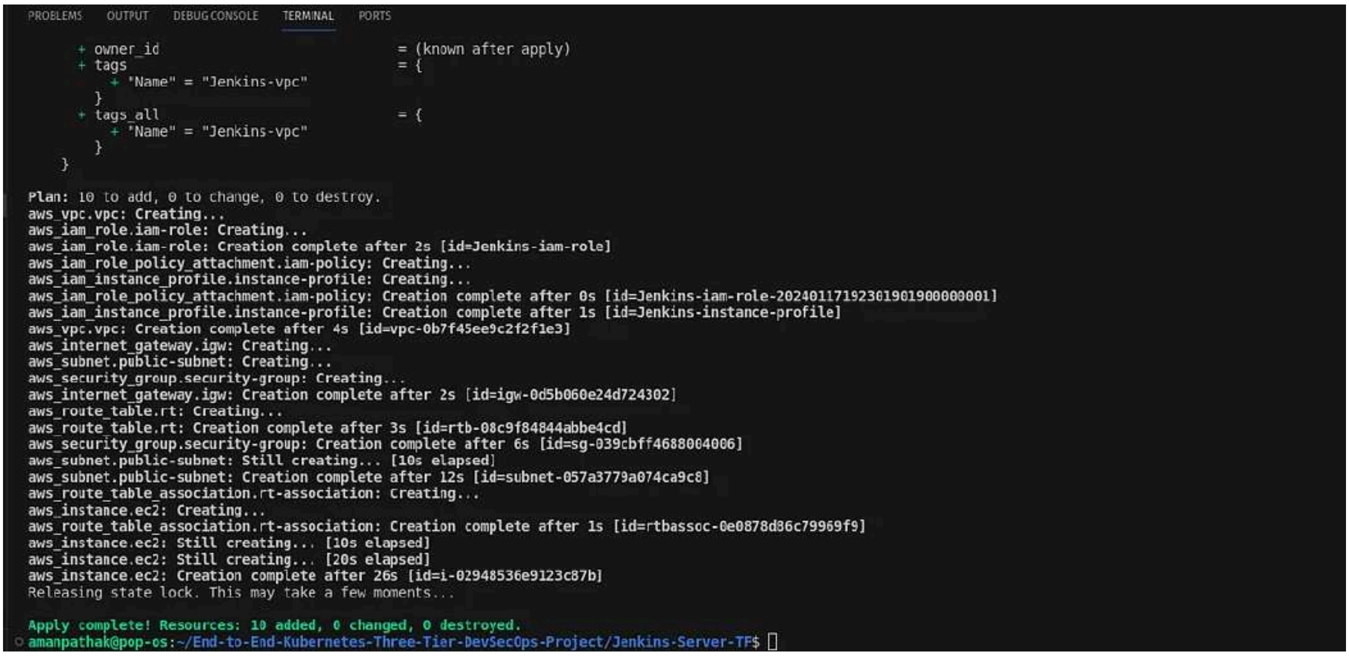
```
terraform plan -var-file=variables.tfvars
```



```
        }
      + vpc_id                                          = (known after apply)
    }

  # aws_vpc.vpc will be created
  + resource "aws_vpc" "vpc" {
      + arn                                  = (known after apply)
      + cidr_block                           = "10.0.0.0/16"
      + default_network_acl_id               = (known after apply)
      + default_route_table_id               = (known after apply)
      + default_security_group_id            = (known after apply)
      + dhcp_options_id                      = (known after apply)
      + enable_dns_hostnames                 = (known after apply)
      + enable_dns_support                   = true
      + enable_network_address_usage_metrics = (known after apply)
      + id                                   = (known after apply)
      + instance_tenancy                     = "default"
      + ipv6_association_id                  = (known after apply)
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + tags                                 = {
          + "Name" = "Jenkins-vpc"
        }
      + tags_all                             = {
          + "Name" = "Jenkins-vpc"
        }
    }

Plan: 10 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply" now.
Releasing state lock. This may take a few moments...
amanpathak@pop-os:~/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project/Jenkins-Server-TF$
```

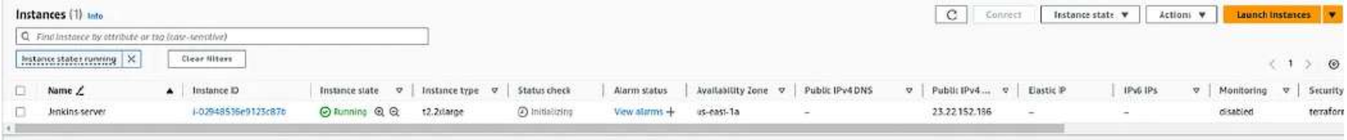
Now, run the below command to create the infrastructure on AWS Cloud which will take 3 to 4 minutes maximum

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



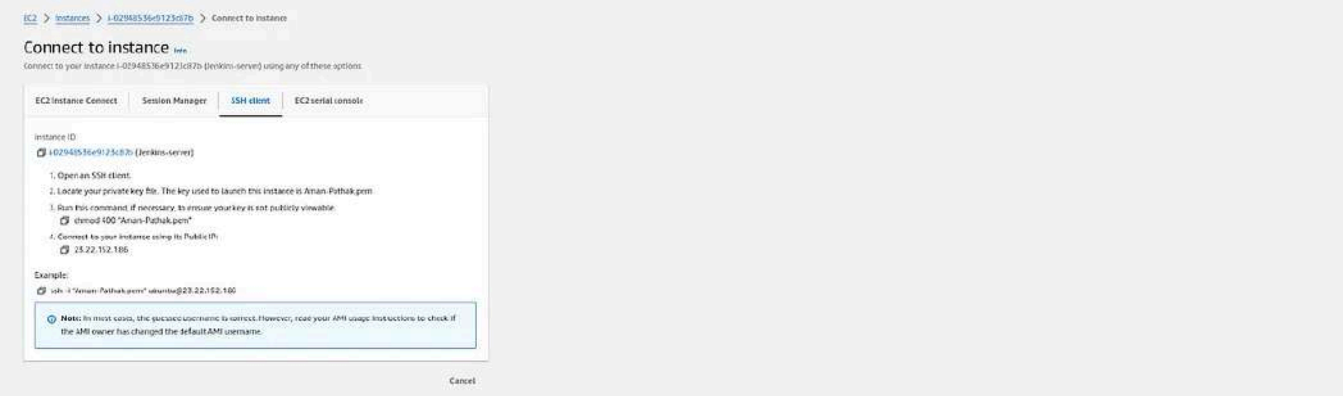
```
      + owner_id                         = (known after apply)
      + tags                             = {
          + "Name" = "Jenkins-vpc"
        }
      + tags_all                         = {
          + "Name" = "Jenkins-vpc"
        }
    }

Plan: 10 to add, 0 to change, 0 to destroy.
aws_vpc.vpc: Creating...
aws_iam_role.iam-role: Creating...
aws_iam_role.iam-role: Creation complete after 2s [id=Jenkins-iam-role]
aws_iam_role_policy_attachment.iam-policy: Creating...
aws_iam_instance_profile.instance-profile: Creating...
aws_iam_role_policy_attachment.iam-policy: Creation complete after 0s [id=Jenkins-iam-role-20240117192301901900000001]
aws_iam_instance_profile.instance-profile: Creation complete after 1s [id=Jenkins-instance-profile]
aws_vpc.vpc: Creation complete after 4s [id=vpc-0b7f45ee9c2f2f1e3]
aws_internet_gateway.igw: Creating...
aws_subnet.public-subnet: Creating...
aws_security_group.security-group: Creating...
aws_internet_gateway.igw: Creation complete after 2s [id=igw-0d5b060e24d724302]
aws_route_table.rt: Creating...
aws_route_table.rt: Creation complete after 3s [id=rtb-08c9f84844abbe4cd]
aws_security_group.security-group: Creation complete after 6s [id=sg-039cbff4688004006]
aws_subnet.public-subnet: Still creating... [10s elapsed]
aws_subnet.public-subnet: Creation complete after 12s [id=subnet-057a3779a074ca9c8]
aws_route_table_association.rt-association: Creating...
aws_instance.ec2: Creating...
aws_route_table_association.rt-association: Creation complete after 1s [id=rtbassoc-0e0878d86c79969f9]
aws_instance.ec2: Still creating... [10s elapsed]
aws_instance.ec2: Still creating... [20s elapsed]
aws_instance.ec2: Creation complete after 26s [id=i-02948536e9123c87b]
Releasing state lock. This may take a few moments...

Apply complete! Resources: 10 added, 0 changed, 0 destroyed.
amanpathak@pop-os:~/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project/Jenkins-Server-TF$ 
```

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



☐	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP	IPv6 IPs	Monitoring	Security
☐	Jenkins-server	i-02948536e9123c87b	Running	t2.2xlarge	Initializing	View alarms +	us-east-1a	–	23.22.152.186	–	–	disabled	terrafor

Copy the **ssh** command and paste it on your local machine.



Connect to instance Info

Connect to your instance i-02948536e9123c87b (Jenkins-server) using any of these options

Instance ID

i-02948536e9123c87b (Jenkins-server)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is Aman-Pathak.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   chmod 400 "Aman-Pathak.pem"
4. Connect to your instance using its Public IP:
   23.22.152.186

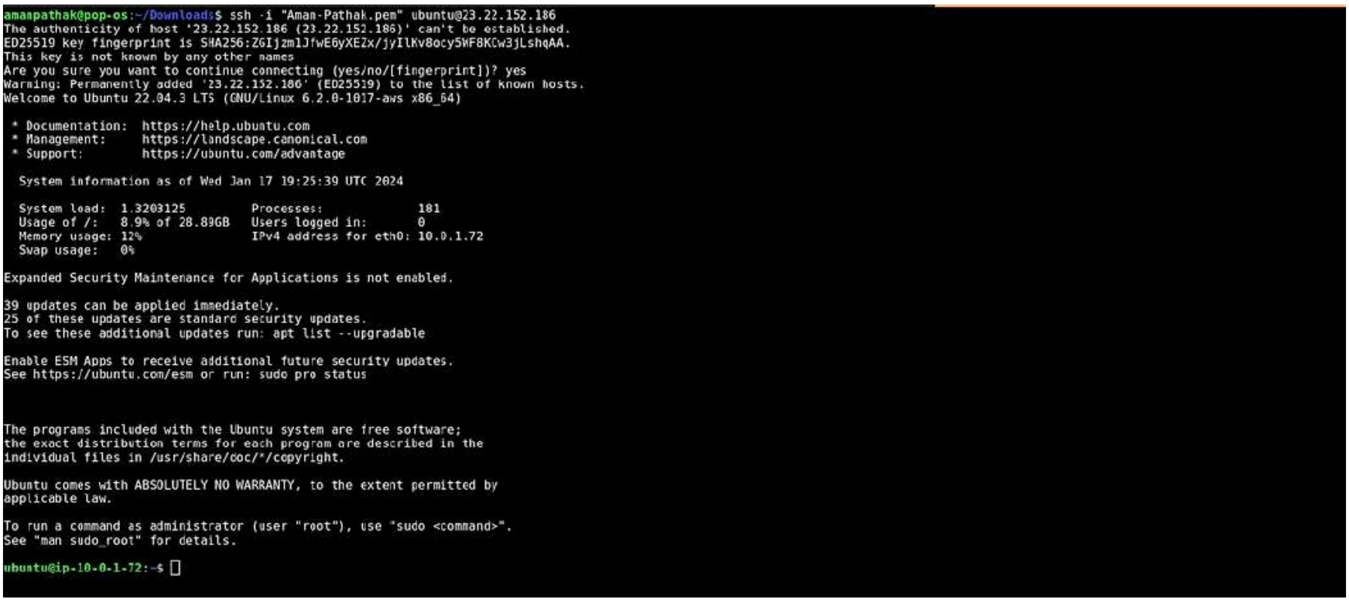
Example:

ssh -i "Aman-Pathak.pem" ubuntu@23.22.152.186

Note: In most cases, the guessed username is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI username.

Step 4: Configure the Jenkins

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

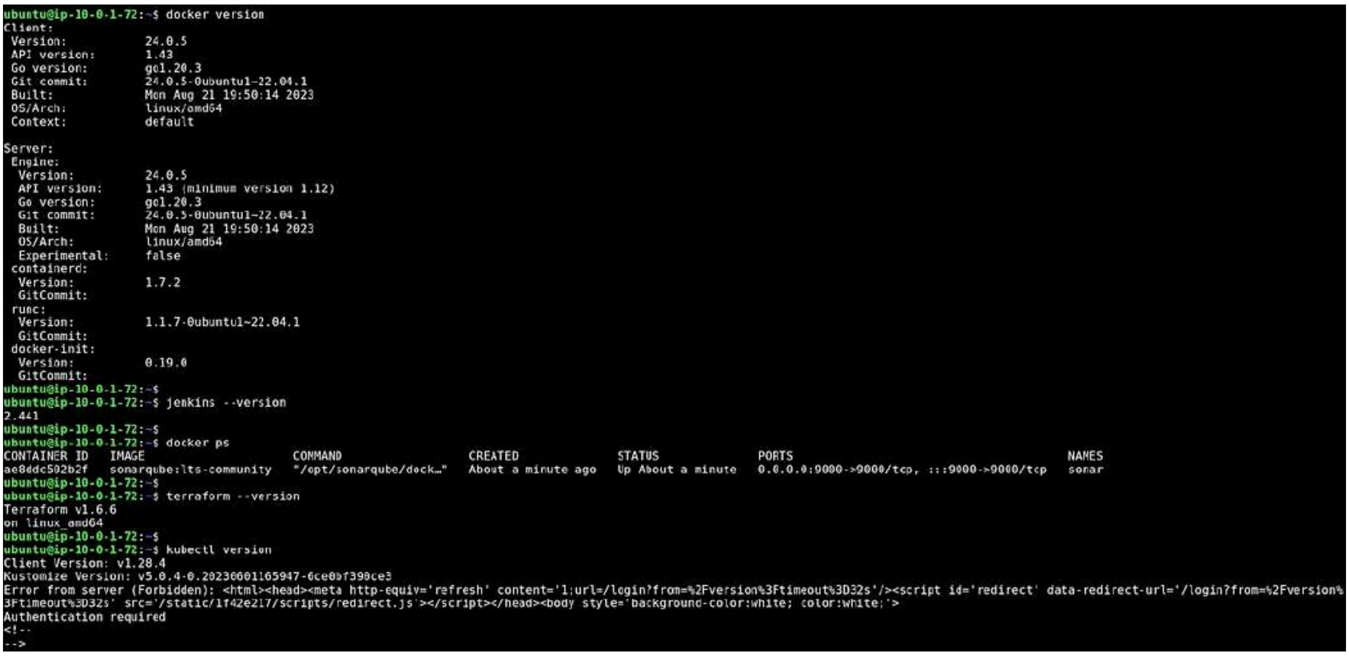


```
amanpathak@pop-os:~/Downloads$ ssh -i "Aman-Pathak.pem" ubuntu@23.22.152.186
The authenticity of host '23.22.152.186 (23.22.152.186)' can't be established.
ED25519 key fingerprint is SHA256:ZGIjzmlJfwE6yXEZx/jyIlKv8ocy5WF8KCw3jLshqAA.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '23.22.152.186' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 6.2.0-1017-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Jan 17 19:25:39 UTC 2024

  System load:  1.3203125         Processes:             181
  Usage of /:   8.9% of 28.89GB   Users logged in:       0
  Memory usage: 12%               IPv4 address for eth0: 10.0.1.72
  Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.

39 updates can be applied immediately.
25 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-10-0-1-72:~$ 
```

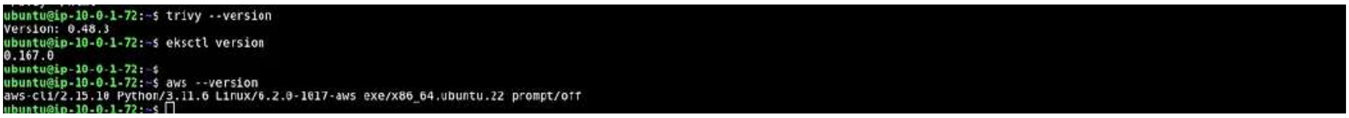
We have installed some services such as Jenkins, Docker, Sonarqube, Terraform, Kubectl, AWS CLI, and Trivy.

Let’s validate whether all our installed or not.

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

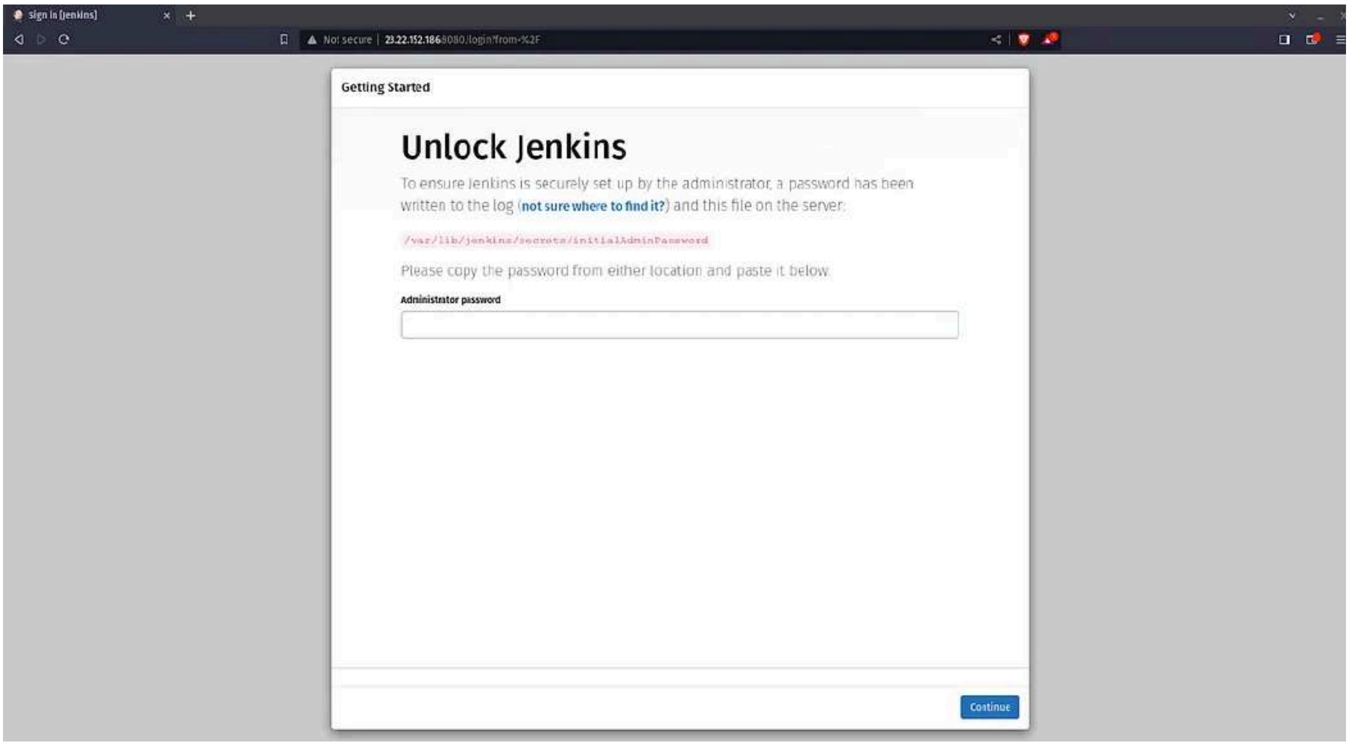


```
ubuntu@ip-10-0-1-72:~$ docker version
Client:
 Version:           24.0.5
 API version:       1.43
 Go version:        go1.20.3
 Git commit:        24.0.5-0ubuntu1~22.04.1
 Built:             Mon Aug 21 19:50:14 2023
 OS/Arch:           linux/amd64
 Context:           default

Server:
 Engine:
  Version:          24.0.5
  API version:      1.43 (minimum version 1.12)
  Go version:       go1.20.3
  Git commit:       24.0.5-0ubuntu1~22.04.1
  Built:            Mon Aug 21 19:50:14 2023
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.7.2
  GitCommit:
 runc:
  Version:          1.1.7-0ubuntu1~22.04.1
  GitCommit:
 docker-init:
  Version:          0.19.0
  GitCommit:
ubuntu@ip-10-0-1-72:~$
ubuntu@ip-10-0-1-72:~$ jenkins --version
2.441
ubuntu@ip-10-0-1-72:~$
ubuntu@ip-10-0-1-72:~$ docker ps
CONTAINER ID   IMAGE                   COMMAND                  CREATED              STATUS              PORTS                                       NAMES
ae8ddc502b2f   sonarqube:lts-community   "/opt/sonarqube/dock…"   About a minute ago   Up About a minute   0.0.0.0:9000->9000/tcp, :::9000->9000/tcp   sonar
ubuntu@ip-10-0-1-72:~$
ubuntu@ip-10-0-1-72:~$ terraform --version
Terraform v1.6.6
on linux_amd64
ubuntu@ip-10-0-1-72:~$
ubuntu@ip-10-0-1-72:~$ kubectl version
Client Version: v1.28.4
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Error from server (Forbidden): <html><head><meta http-equiv='refresh' content='1;url=/login?from=%2Fversion%3Ftimeout%3D32s'/><script id='redirect' data-redirect-url='/login?from=%2Fversion%
3Ftimeout%3D32s' src='/static/1f42e217/scripts/redirect.js'></script></head><body style='background-color:white; color:white;'>
Authentication required
<!--
-->
```



```
ubuntu@ip-10-0-1-72:~$ trivy --version
Version: 0.48.3
ubuntu@ip-10-0-1-72:~$ eksctl version
0.167.0
ubuntu@ip-10-0-1-72:~$
ubuntu@ip-10-0-1-72:~$ aws --version
aws-cli/2.15.10 Python/3.11.6 Linux/6.2.0-1017-aws exe/x86_64.ubuntu.22 prompt/off
ubuntu@ip-10-0-1-72:~$
```

Now, we have to configure Jenkins. So, copy the public IP of your Jenkins Server and paste it on your favorite browser with an 8080 port.



Getting Started

Unlock Jenkins

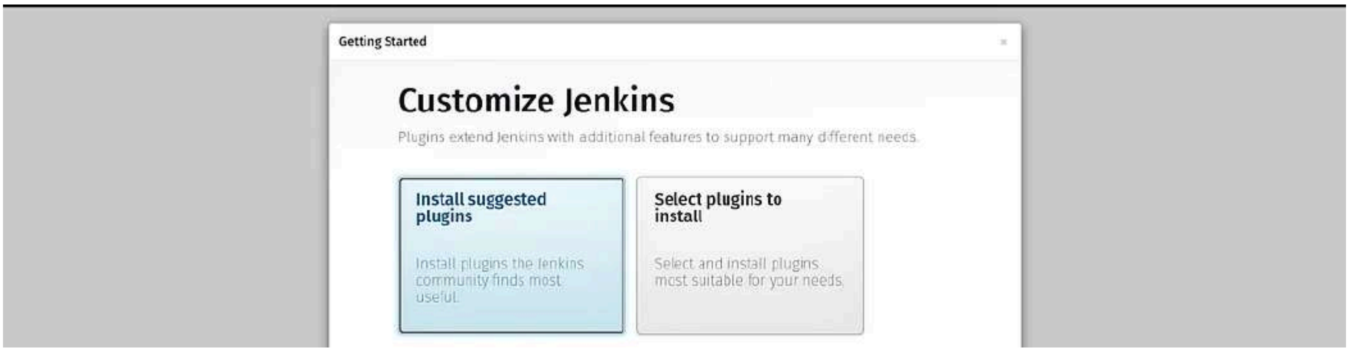
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Click on **Install suggested plugins**



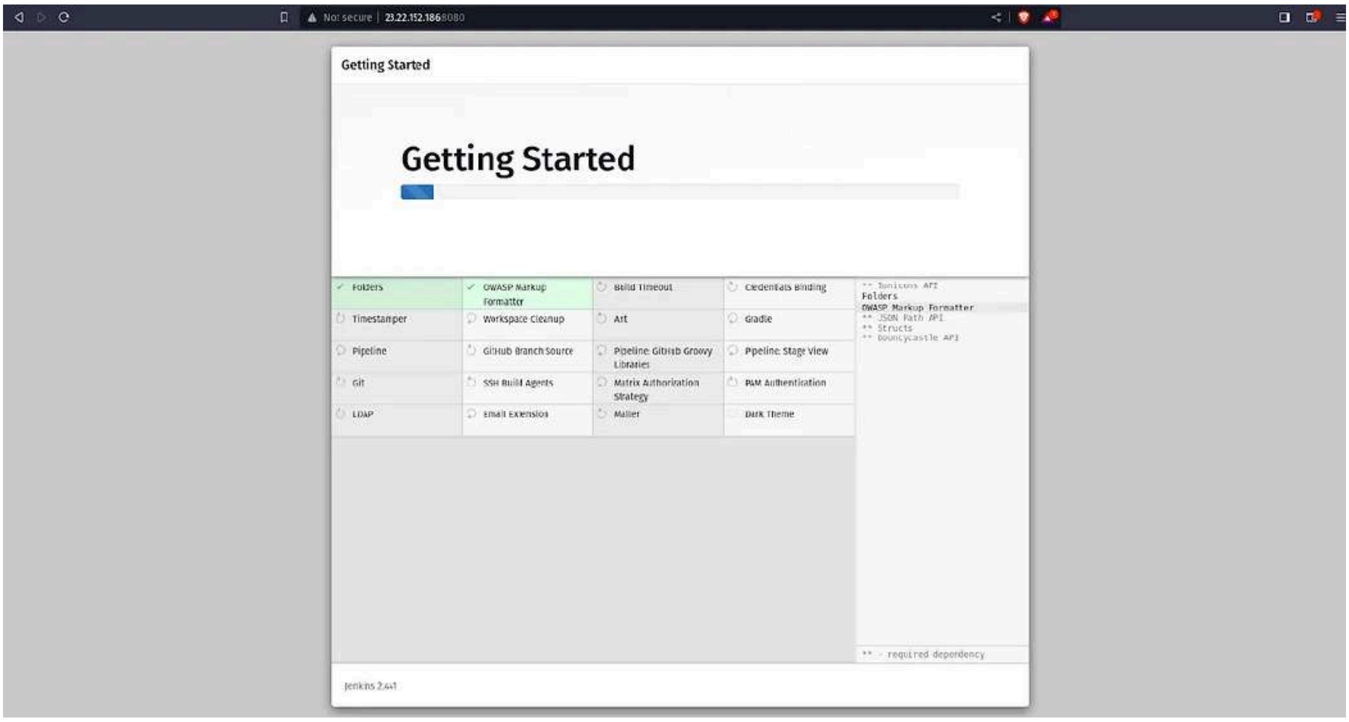
Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.

The plugins will be installed



Getting Started

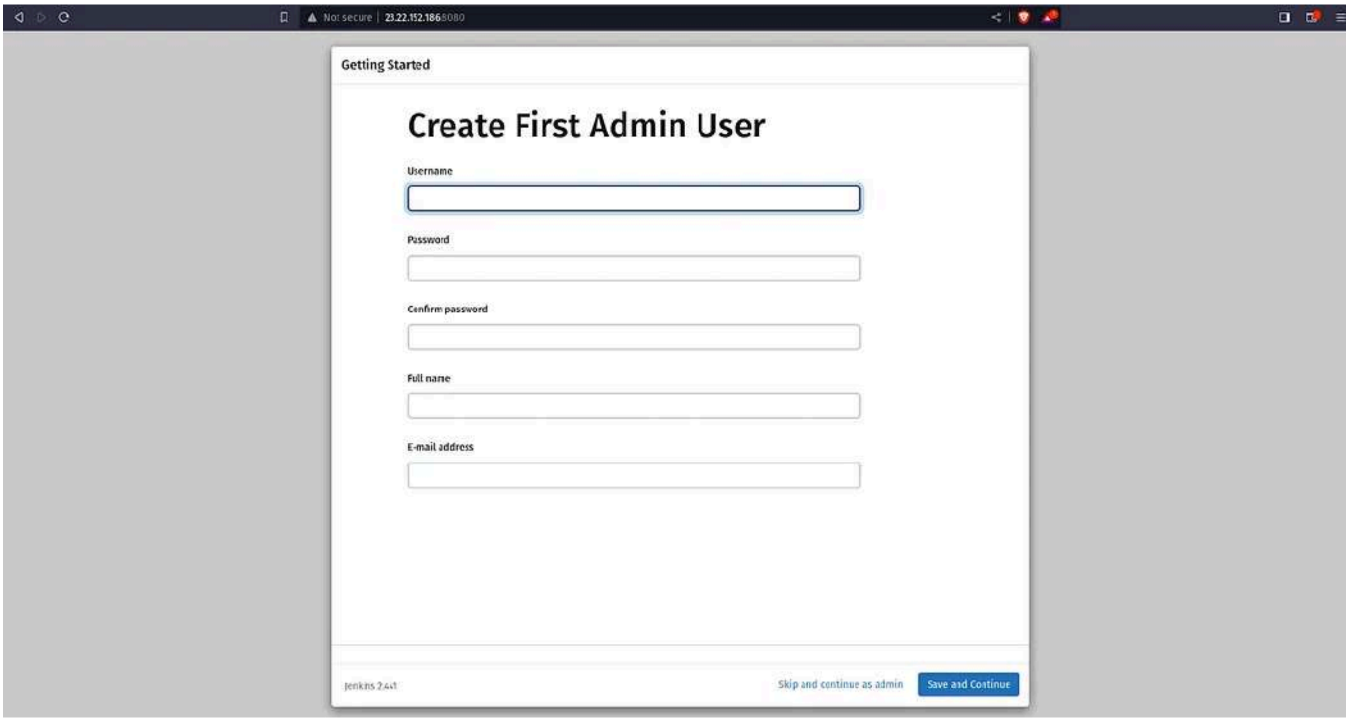
✓ Folders	✓ OWASP Markup Formatter	Build Timeout	Credentials Binding
Timestamper	Workspace Cleanup	Ant	Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	Pipeline: Stage View
Git	SSH Build Agents	Matrix Authorization Strategy	PAM Authentication
LDAP	Email Extension	Mailer	Dark Theme

```
** Ionicons API
Folders
OWASP Markup Formatter
** JSON Path API
** Structs
** bouncycastle API
```

```
** - required dependency
```

Jenkins 2.441

After installing the plugins, continue as admin



Create First Admin User

Username

Password

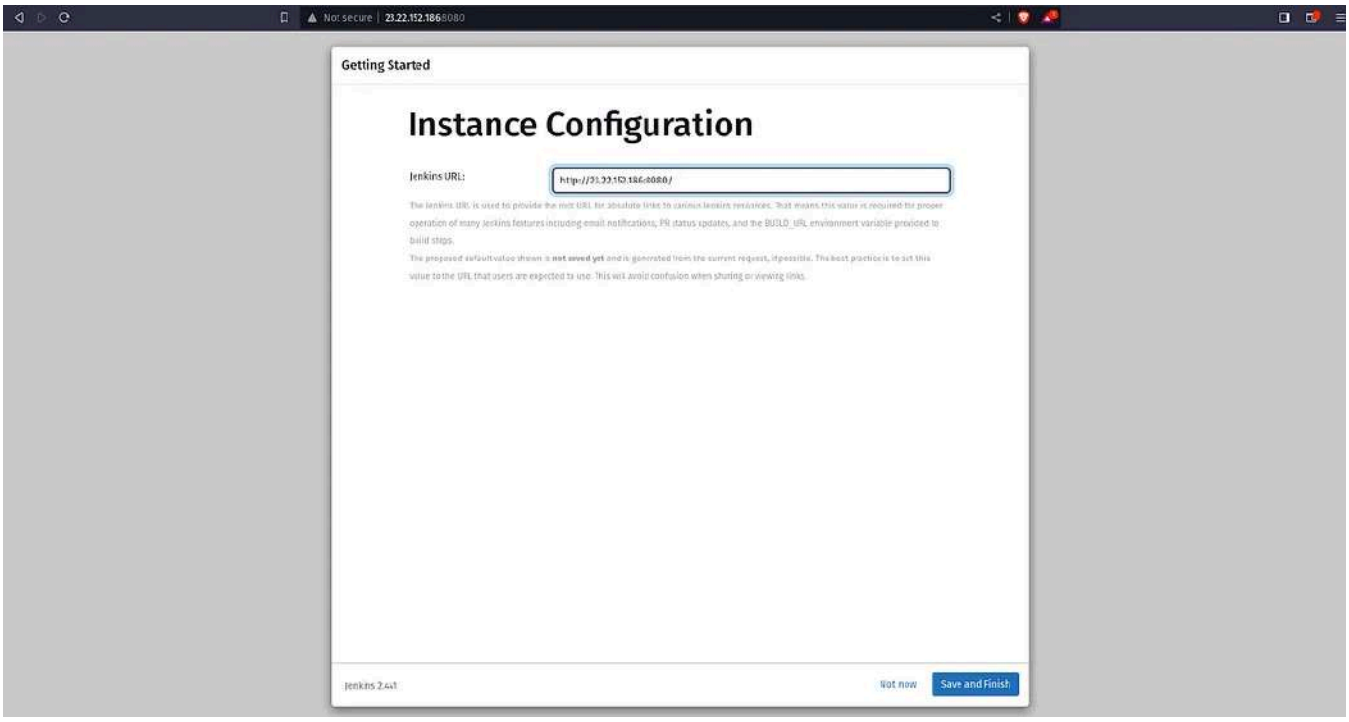
Confirm password

Full name

E-mail address

Jenkins 2.441

Click on **Save and Finish**



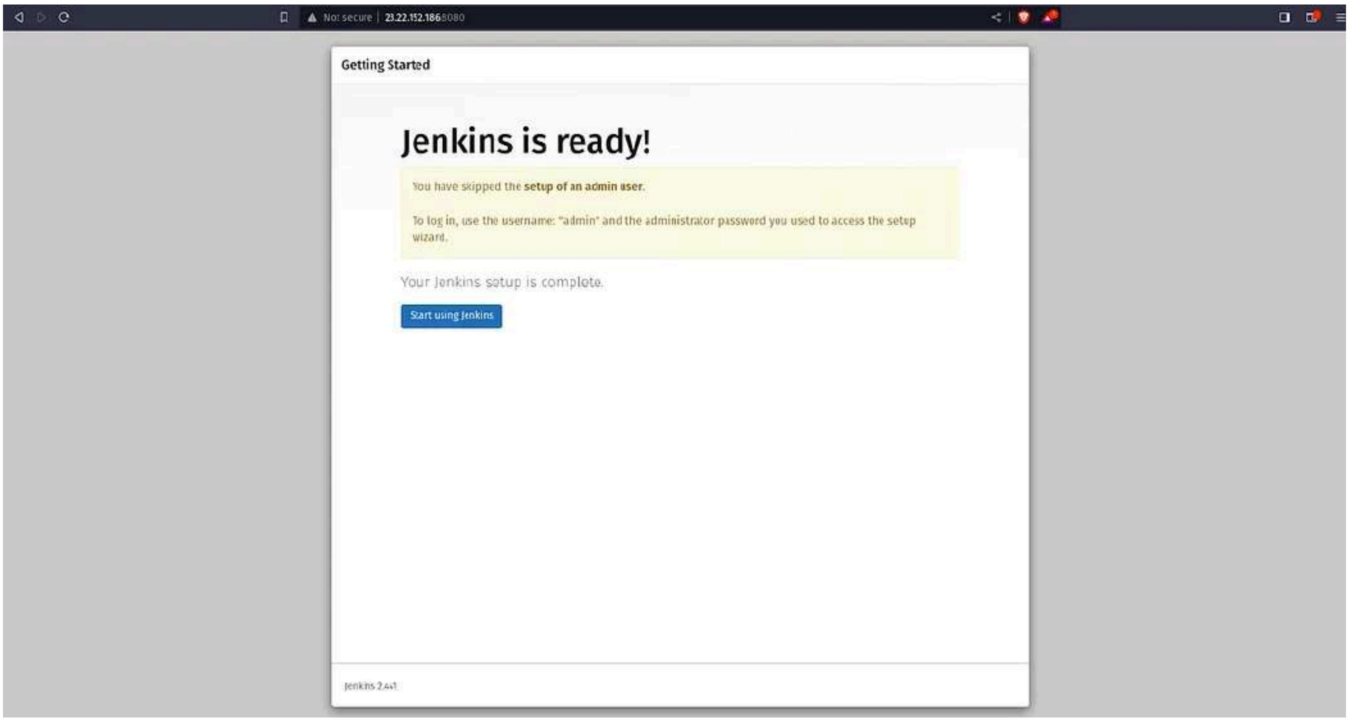
Instance Configuration

Jenkins URL: http://23.22.152.186:8080/

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Click on **Start using Jenkins**



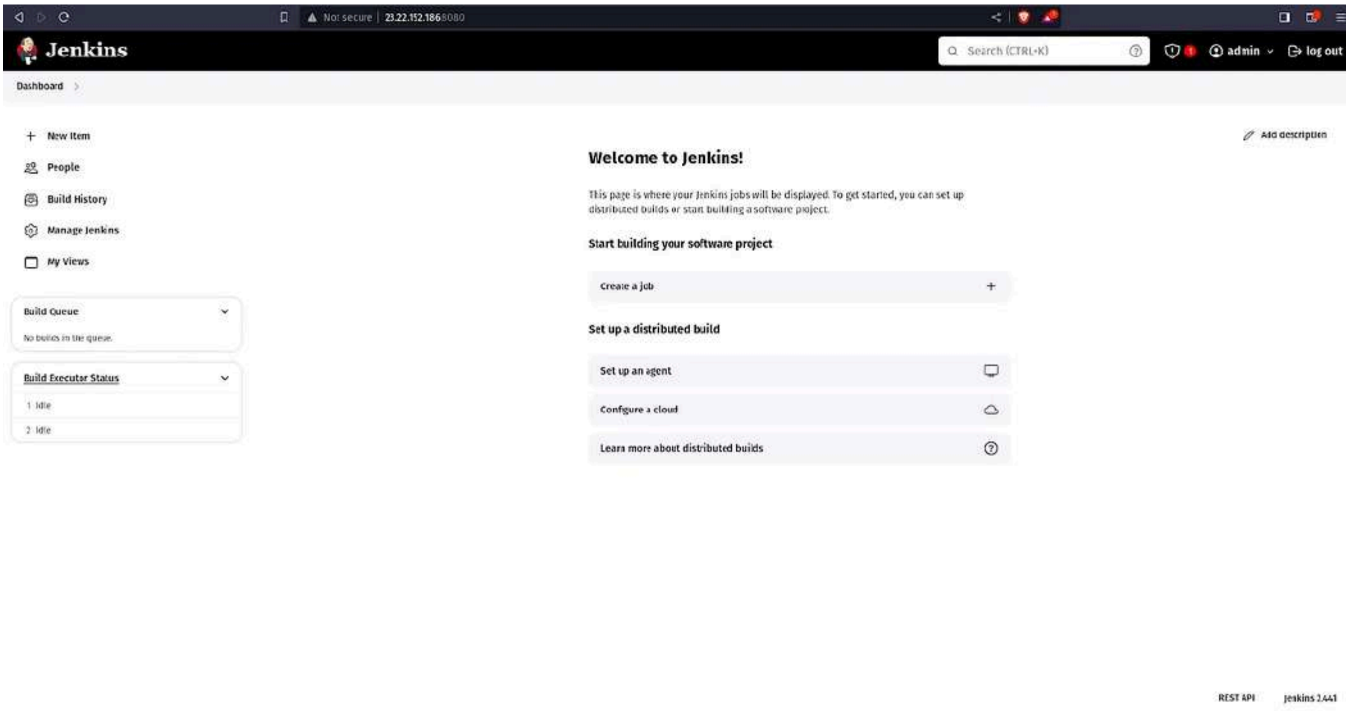
Jenkins is ready!

You have skipped the **setup of an admin user**.

To log in, use the username: "admin" and the administrator password you used to access the setup wizard.

Your Jenkins setup is complete.

The Jenkins Dashboard will look like the below snippet



Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

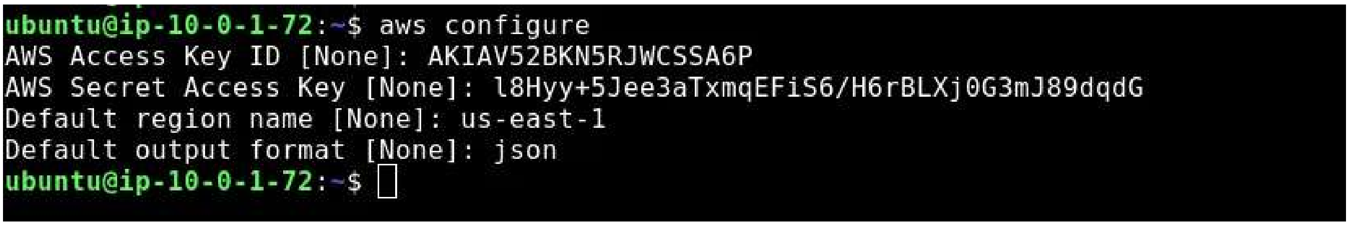
Start building your software project

Set up a distributed build

No builds in the queue.

Step 5: We will deploy the EKS Cluster using eksctl commands

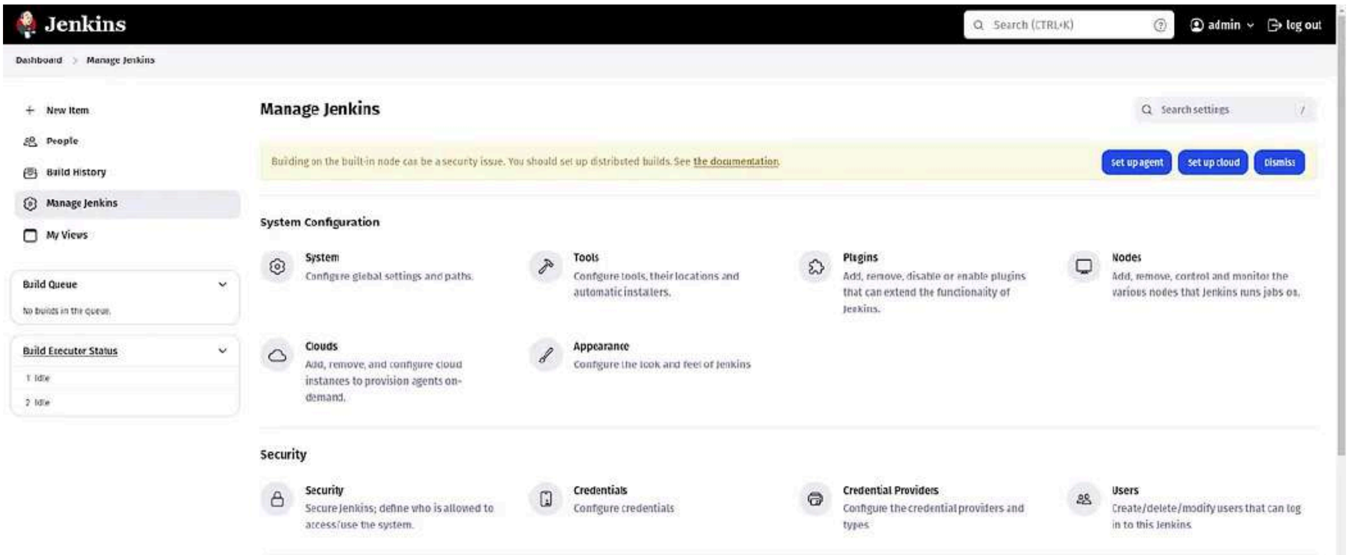
Now, go back to your Jenkins Server **terminal** and configure the AWS.



```
ubuntu@ip-10-0-1-72:~$ aws configure
AWS Access Key ID [None]: AKIAV52BKN5RJWCSSA6P
AWS Secret Access Key [None]: l8Hyy+5Jee3aTxmqEFiS6/H6rBLXj0G3mJ89dqdG
Default region name [None]: us-east-1
Default output format [None]: json
ubuntu@ip-10-0-1-72:~$
```

Go to **Manage Jenkins**

Click on **Plugins**



Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

No builds in the queue.

System Configuration

System — Configure global settings and paths.

Tools — Configure tools, their locations and automatic installers.

Plugins — Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Nodes — Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Clouds — Add, remove, and configure cloud instances to provision agents on-demand.

Appearance — Configure the look and feel of Jenkins

Security

Security — Secure Jenkins; define who is allowed to access/use the system.

Credentials — Configure credentials

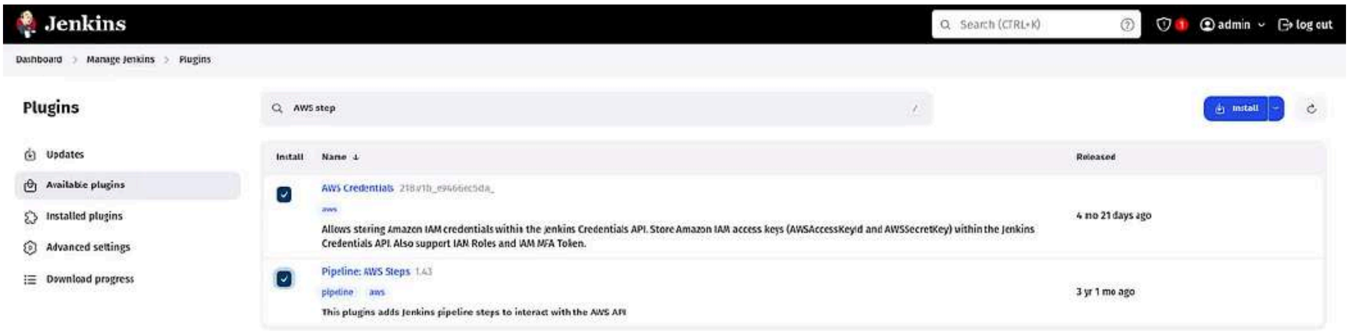
Credential Providers — Configure the credential providers and types

Users — Create/delete/modify users that can log in to this Jenkins

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

AWS Credentials

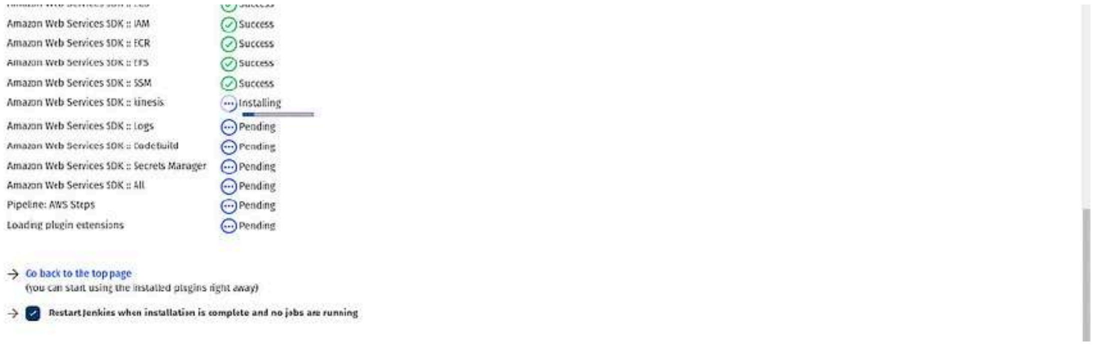
Pipeline: AWS Steps



AWS step

Install	Name ↓	Released
☑	AWS Credentials 218.v1b_e9466ec5da_ aws Allows storing Amazon IAM credentials within the Jenkins Credentials API. Store Amazon IAM access keys (AWSAccessKeyId and AWSSecretKey) within the Jenkins Credentials API. Also support IAM Roles and IAM MFA Token.	4 mo 21 days ago
☑	Pipeline: AWS Steps 1.43 pipeline aws This plugins adds Jenkins pipeline steps to interact with the AWS API	3 yr 1 mo ago

Once, both the plugins are installed, restart your Jenkins service by checking the **Restart Jenkins** option.

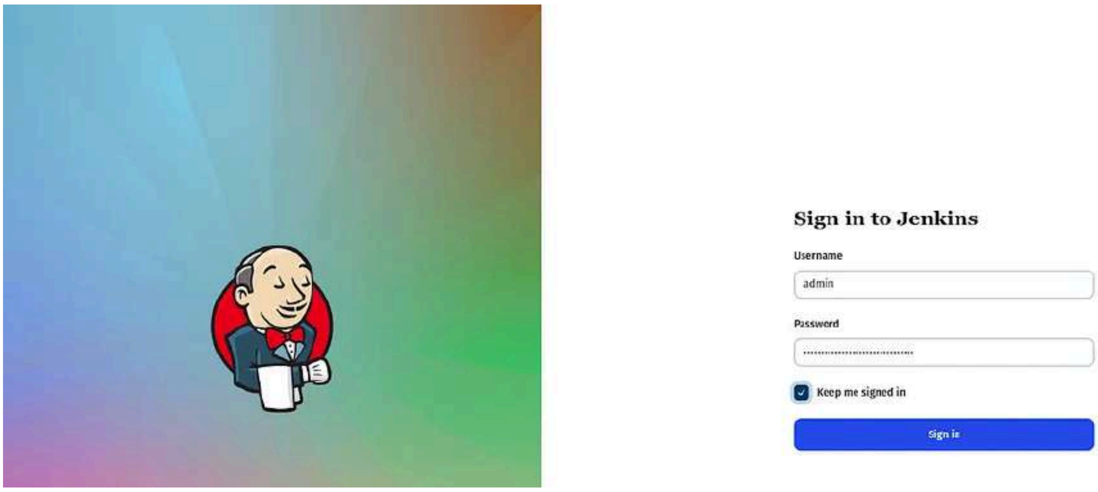


Amazon Web Services SDK :: IAM	Success
Amazon Web Services SDK :: ECR	Success
Amazon Web Services SDK :: EFS	Success
Amazon Web Services SDK :: SSM	Success
Amazon Web Services SDK :: kinesis	Installing
Amazon Web Services SDK :: Logs	Pending
Amazon Web Services SDK :: CodeBuild	Pending
Amazon Web Services SDK :: Secrets Manager	Pending
Amazon Web Services SDK :: All	Pending
Pipeline: AWS Steps	Pending
Loading plugin extensions	Pending

→ Go back to the top page
(you can start using the installed plugins right away)

→ ☑ Restart Jenkins when installation is complete and no jobs are running

Login to your Jenkins Server Again



Sign in to Jenkins

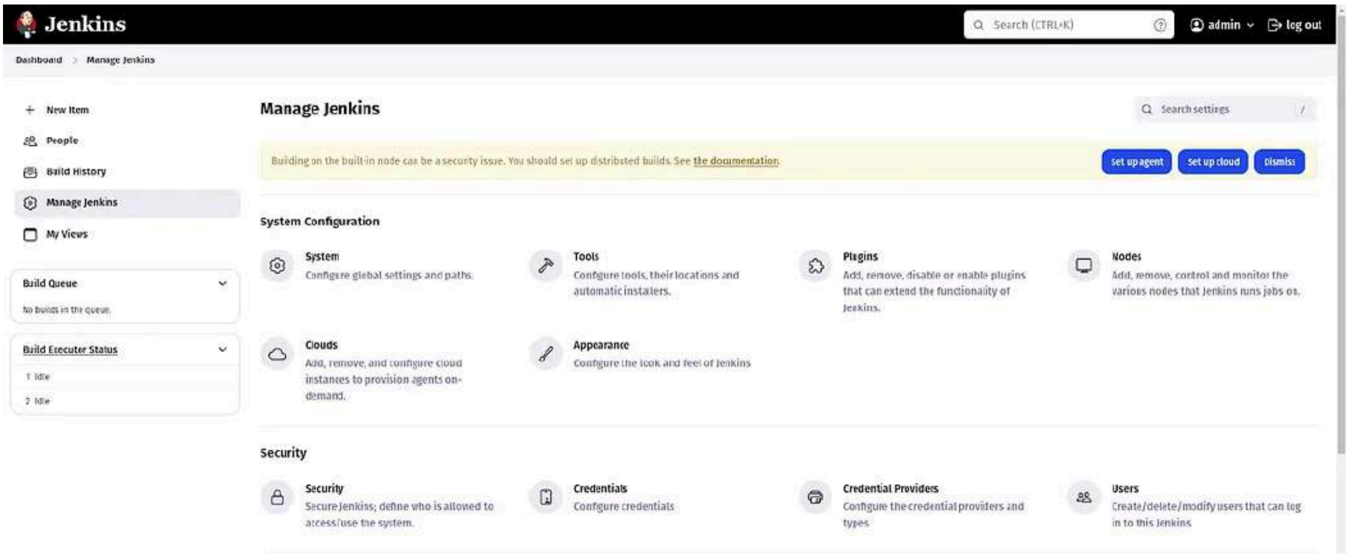
Username: admin

Password: ••••••••••••••••••••••••••••••

☑ Keep me signed in

Now, we have to set our AWS credentials on Jenkins

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



Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See the documentation.

System Configuration

System
Configure global settings and paths.

Tools
Configure tools, their locations and automatic installers.

Plugins
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.

Nodes
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

Clouds
Add, remove, and configure cloud instances to provision agents on-demand.

Appearance
Configure the look and feel of Jenkins

Security

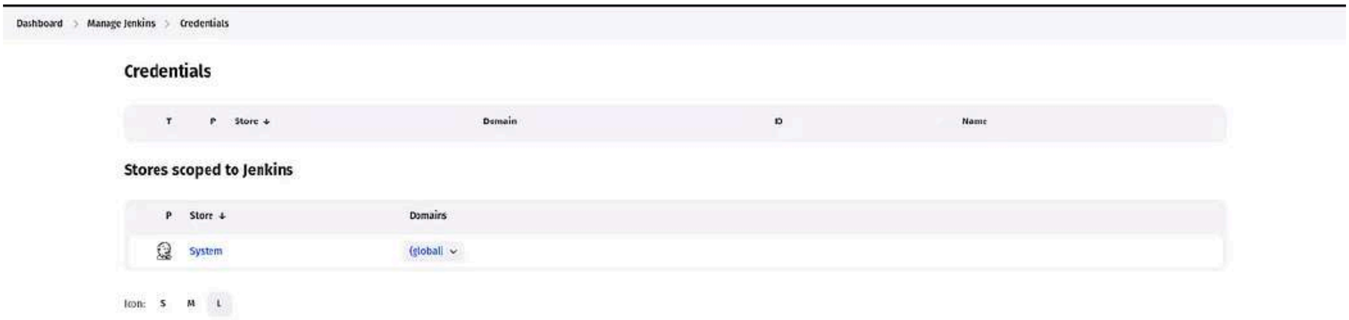
Security
Secure Jenkins; define who is allowed to access/use the system.

Credentials
Configure credentials

Credential Providers
Configure the credential providers and types

Users
Create/delete/modify users that can log in to this Jenkins

Click on **global.**



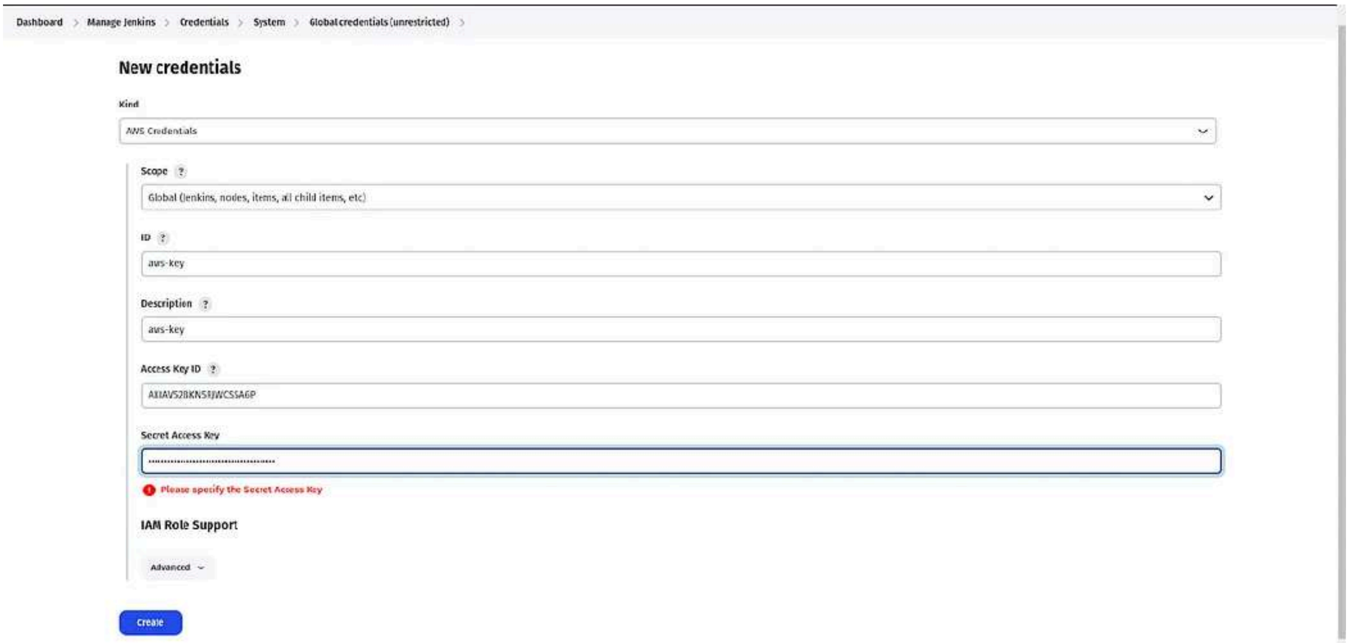
Credentials

T	P	Store ↓	Domain	ID	Name

Stores scoped to Jenkins

P	Store ↓	Domains
	System	(global)

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.**


Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
Advanced
Create

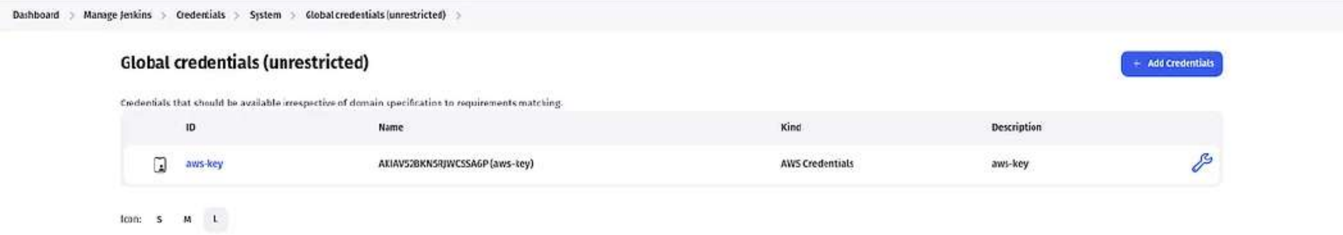
New credentials

Kind	AWS Credentials
Scope	Global (Jenkins, nodes, items, all child items, etc)
ID	aws-key
Description	aws-key
Access Key ID	AKIAVS2BKN5IJWCSSA6P
Secret Access Key	••

Please specify the Secret Access Key

IAM Role Support

The Credentials will look like the below snippet.


Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
+ Add Credentials
Icon: S M L

Global credentials (unrestricted)

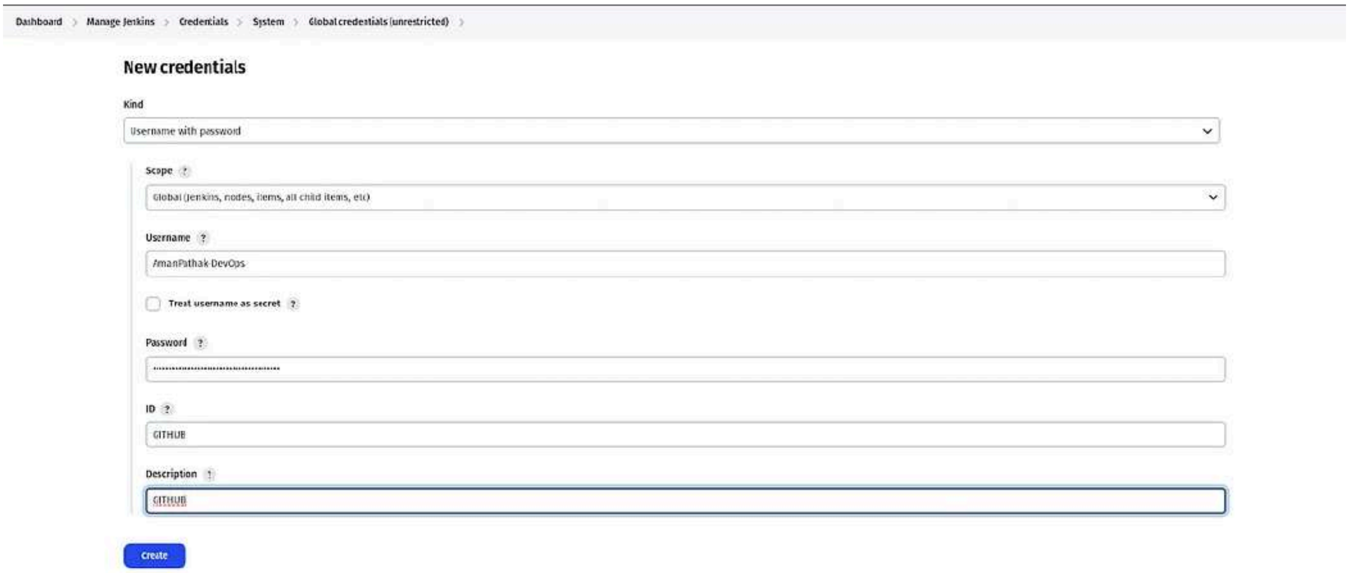
Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
aws-key	AKIAV52BKN5RJWCSSAGP (aws-key)	AWS Credentials	aws-key

Now, We need to add GitHub credentials as well because currently, my repository is Private.

This thing, I am performing this because in Industry Projects your repository will be private.

So, add the username and personal access token of your GitHub account.


Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
Create

New credentials

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: AmanPathak-DevOps

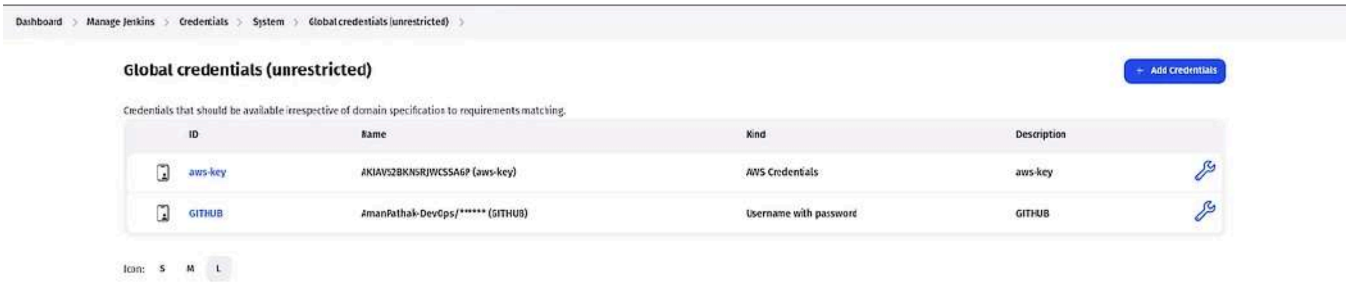
□ Treat username as secret

Password: ••

ID: GITHUB

Description: GITHUB

Both credentials will look like this.


Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >
+ Add credentials
Icon: S M L

Global credentials (unrestricted)

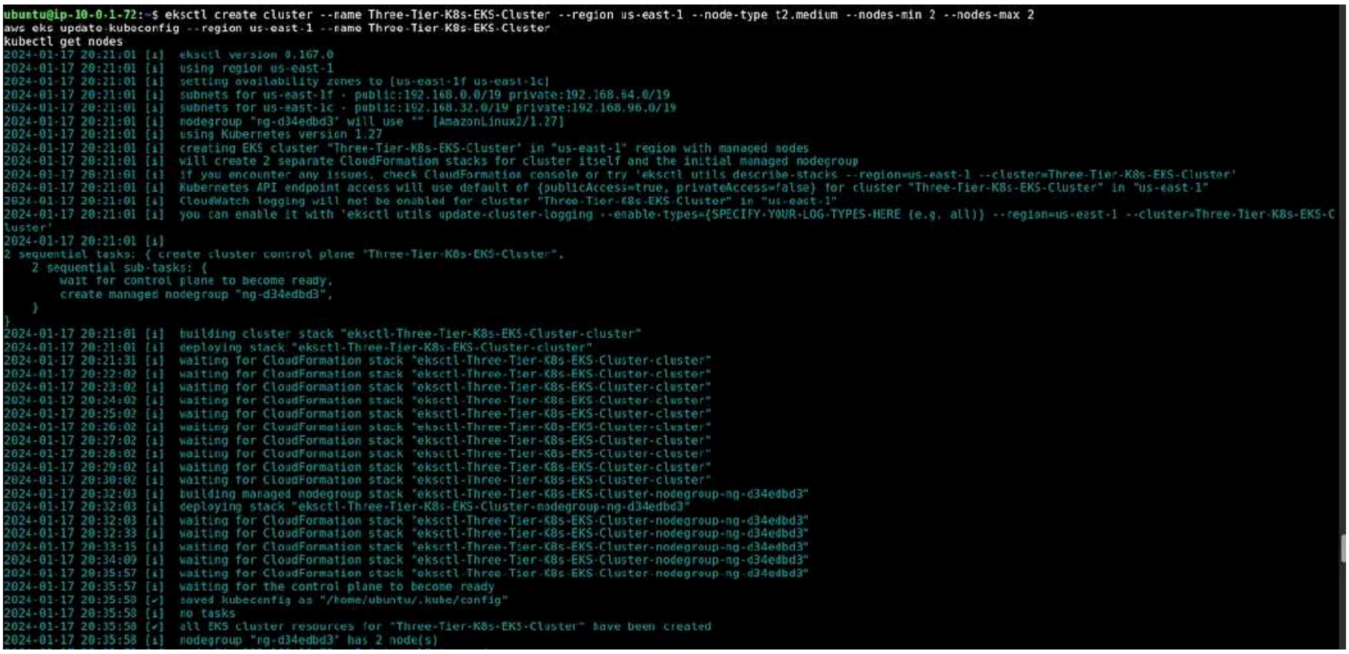
Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
aws-key	AKIAV52BKN5RJWCSSA6P (aws-key)	AWS Credentials	aws-key
GITHUB	AmanPathak-DevOps/****** (GITHUB)	Username with password	GITHUB

Create an eks cluster using the below commands.

```
eksctl create cluster --name Three-Tier-K8s-EKS-Cluster --region us-east-1 --nc
```

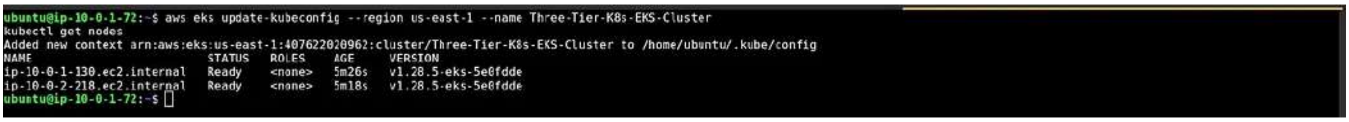
```
aws eks update-kubeconfig --region us-east-1 --name Three-Tier-K8s-EKS-Cluster
```



```
ubuntu@ip-10-0-1-72:~$ eksctl create cluster --name Three-Tier-K8s-EKS-Cluster --region us-east-1 --node-type t2.medium --nodes-min 2 --nodes-max 2
aws eks update-kubeconfig --region us-east-1 --name Three-Tier-K8s-EKS-Cluster
kubectl get nodes
2024-01-17 20:21:01 [i]  eksctl version 0.167.0
2024-01-17 20:21:01 [i]  using region us-east-1
2024-01-17 20:21:01 [i]  setting availability zones to [us-east-1f us-east-1c]
2024-01-17 20:21:01 [i]  subnets for us-east-1f - public:192.168.0.0/19 private:192.168.64.0/19
2024-01-17 20:21:01 [i]  subnets for us-east-1c - public:192.168.32.0/19 private:192.168.96.0/19
2024-01-17 20:21:01 [i]  nodegroup "ng-d34edbd3" will use "" [AmazonLinux2/1.27]
2024-01-17 20:21:01 [i]  using Kubernetes version 1.27
2024-01-17 20:21:01 [i]  creating EKS cluster "Three-Tier-K8s-EKS-Cluster" in "us-east-1" region with managed nodes
2024-01-17 20:21:01 [i]  will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup
2024-01-17 20:21:01 [i]  if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-1 --cluster=Three-Tier-K8s-EKS-Cluster'
2024-01-17 20:21:01 [i]  Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "Three-Tier-K8s-EKS-Cluster" in "us-east-1"
2024-01-17 20:21:01 [i]  CloudWatch logging will not be enabled for cluster "Three-Tier-K8s-EKS-Cluster" in "us-east-1"
2024-01-17 20:21:01 [i]  you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=us-east-1 --cluster=Three-Tier-K8s-EKS-C
luster'
2024-01-17 20:21:01 [i]
2 sequential tasks: { create cluster control plane "Three-Tier-K8s-EKS-Cluster",
    2 sequential sub-tasks: {
        wait for control plane to become ready,
        create managed nodegroup "ng-d34edbd3",
    }
}
2024-01-17 20:21:01 [i]  building cluster stack "eksctl-Three-Tier-K8s-EKS-Cluster-cluster"
2024-01-17 20:21:01 [i]  deploying stack "eksctl-Three-Tier-K8s-EKS-Cluster-cluster"
2024-01-17 20:21:31 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-cluster"
2024-01-17 20:22:02 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-cluster"
2024-01-17 20:23:02 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-cluster"
2024-01-17 20:24:02 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-cluster"
2024-01-17 20:25:02 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-cluster"
2024-01-17 20:26:02 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-cluster"
2024-01-17 20:27:02 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-cluster"
2024-01-17 20:28:02 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-cluster"
2024-01-17 20:29:02 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-cluster"
2024-01-17 20:30:02 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-cluster"
2024-01-17 20:32:03 [i]  building managed nodegroup stack "eksctl-Three-Tier-K8s-EKS-Cluster-nodegroup-ng-d34edbd3"
2024-01-17 20:32:03 [i]  deploying stack "eksctl-Three-Tier-K8s-EKS-Cluster-nodegroup-ng-d34edbd3"
2024-01-17 20:32:03 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-nodegroup-ng-d34edbd3"
2024-01-17 20:32:33 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-nodegroup-ng-d34edbd3"
2024-01-17 20:33:15 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-nodegroup-ng-d34edbd3"
2024-01-17 20:34:09 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-nodegroup-ng-d34edbd3"
2024-01-17 20:35:57 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-nodegroup-ng-d34edbd3"
2024-01-17 20:35:57 [i]  waiting for the control plane to become ready
2024-01-17 20:35:58 [✔]  saved kubeconfig as "/home/ubuntu/.kube/config"
2024-01-17 20:35:58 [i]  no tasks
2024-01-17 20:35:58 [✔]  all EKS cluster resources for "Three-Tier-K8s-EKS-Cluster" have been created
2024-01-17 20:35:58 [i]  nodegroup "ng-d34edbd3" has 2 node(s)
```

Once your cluster is created, you can validate whether your nodes are ready or not by the below command

```
kubectl get nodes
```

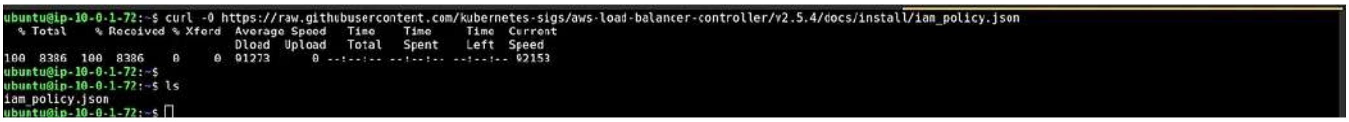


```
ubuntu@ip-10-0-1-72:~$ aws eks update-kubeconfig --region us-east-1 --name Three-Tier-K8s-EKS-Cluster
kubectl get nodes
Added new context arn:aws:eks:us-east-1:407622020962:cluster/Three-Tier-K8s-EKS-Cluster to /home/ubuntu/.kube/config
NAME                          STATUS   ROLES    AGE     VERSION
ip-10-0-1-130.ec2.internal    Ready    <none>   5m26s   v1.28.5-eks-5e0fdde
ip-10-0-2-218.ec2.internal    Ready    <none>   5m18s   v1.28.5-eks-5e0fdde
ubuntu@ip-10-0-1-72:~$ 
```

Step 6: Now, we will configure the Load Balancer on our EKS because our application will have an ingress controller.

Download the policy for the LoadBalancer prerequisite.

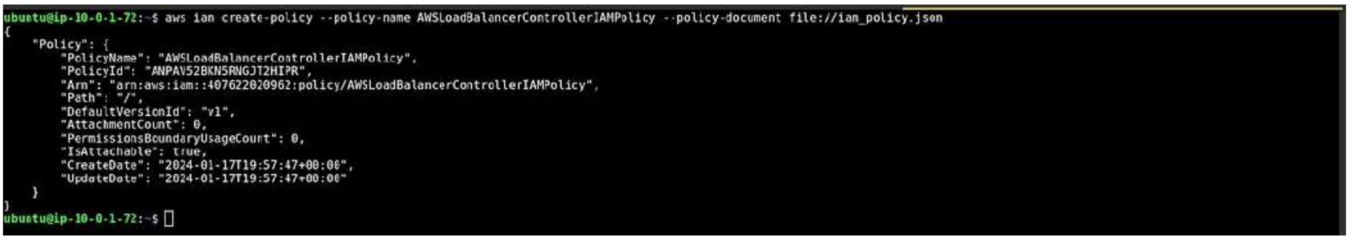
```
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-cor
```



```
ubuntu@ip-10-0-1-72:~$ curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.5.4/docs/install/iam_policy.json
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  8386  100  8386    0     0  91273      0 --:--:-- --:--:-- --:--:-- 92153
ubuntu@ip-10-0-1-72:~$
ubuntu@ip-10-0-1-72:~$ ls
iam_policy.json
ubuntu@ip-10-0-1-72:~$ 
```

Create the IAM policy using the below command

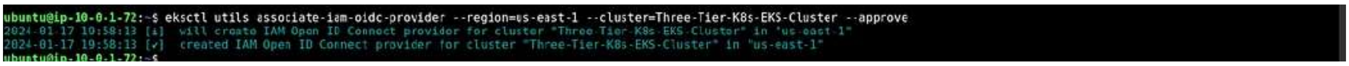
```
aws iam create-policy --policy-name AWSLoadBalancerControllerIAMPolicy --policy
```



```
ubuntu@ip-10-0-1-72:~$ aws iam create-policy --policy-name AWSLoadBalancerControllerIAMPolicy --policy-document file://iam_policy.json
{
    "Policy": {
        "PolicyName": "AWSLoadBalancerControllerIAMPolicy",
        "PolicyId": "ANPAV52BKN5RNGJT2HIPR",
        "Arn": "arn:aws:iam::407622020962:policy/AWSLoadBalancerControllerIAMPolicy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2024-01-17T19:57:47+00:00",
        "UpdateDate": "2024-01-17T19:57:47+00:00"
    }
}
ubuntu@ip-10-0-1-72:~$ 
```

Create OIDC Provider

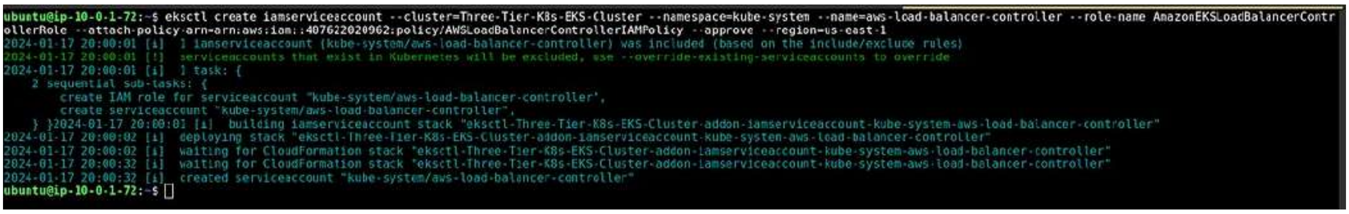
```
eksctl utils associate-iam-oidc-provider --region=us-east-1 --cluster=Three-Tie
```



```
ubuntu@ip-10-0-1-72:~$ eksctl utils associate-iam-oidc-provider --region=us-east-1 --cluster=Three-Tier-K8s-EKS-Cluster --approve
2024-01-17 19:58:13 [i]  will create IAM Open ID Connect provider for cluster "Three-Tier-K8s-EKS-Cluster" in "us-east-1"
2024-01-17 19:58:13 [✓]  created IAM Open ID Connect provider for cluster "Three-Tier-K8s-EKS-Cluster" in "us-east-1"
ubuntu@ip-10-0-1-72:~$
```

Create a Service Account by using below command and replace your account ID with your one

```
eksctl create iamserviceaccount --cluster=Three-Tier-K8s-EKS-Cluster --namespac
```



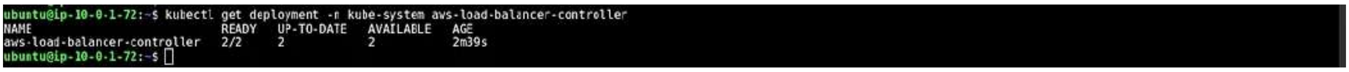
```
ubuntu@ip-10-0-1-72:~$ eksctl create iamserviceaccount --cluster=Three-Tier-K8s-EKS-Cluster --namespace=kube-system --name=aws-load-balancer-controller --role-name AmazonEKSLoadBalancerControllerRole --attach-policy-arn=arn:aws:iam::407622020962:policy/AWSLoadBalancerControllerIAMPolicy --approve --region=us-east-1
2024-01-17 20:00:01 [i]  1 iamserviceaccount (kube-system/aws-load-balancer-controller) was included (based on the include/exclude rules)
2024-01-17 20:00:01 [!]  serviceaccounts that exist in Kubernetes will be excluded, use --override-existing-serviceaccounts to override
2024-01-17 20:00:01 [i]  1 task: {
    2 sequential sub-tasks: {
        create IAM role for serviceaccount "kube-system/aws-load-balancer-controller",
        create serviceaccount "kube-system/aws-load-balancer-controller",
    } }2024-01-17 20:00:01 [i]  building iamserviceaccount stack "eksctl-Three-Tier-K8s-EKS-Cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2024-01-17 20:00:02 [i]  deploying stack "eksctl-Three-Tier-K8s-EKS-Cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2024-01-17 20:00:02 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2024-01-17 20:00:32 [i]  waiting for CloudFormation stack "eksctl-Three-Tier-K8s-EKS-Cluster-addon-iamserviceaccount-kube-system-aws-load-balancer-controller"
2024-01-17 20:00:32 [i]  created serviceaccount "kube-system/aws-load-balancer-controller"
ubuntu@ip-10-0-1-72:~$ 
```

Run the below command to deploy the AWS Load Balancer Controller

```
sudo snap install helm --classic
helm repo add eks https://aws.github.io/eks-charts
helm repo update eks
helm install aws-load-balancer-controller eks/aws-load-balancer-controller -n k
```

After 2 minutes, run the command below to check whether your pods are running or not.

```
kubectl get deployment -n kube-system aws-load-balancer-controller
```



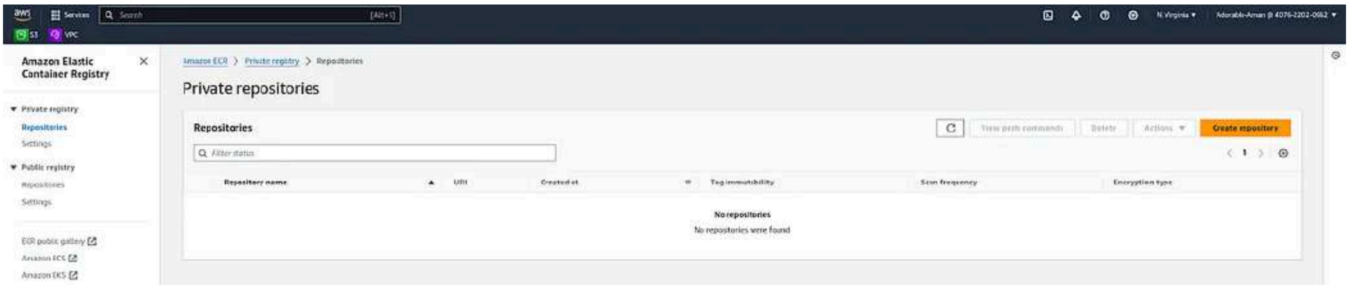
```
ubuntu@ip-10-0-1-72:~$ kubectl get deployment -n kube-system aws-load-balancer-controller
NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
aws-load-balancer-controller   2/2     2            2           2m39s
ubuntu@ip-10-0-1-72:~$ 
```

If the pods are getting Error or CrashLoopBackOff, then use the below command

```
helm upgrade -i aws-load-balancer-controller eks/aws-load-balancer-controller \
  --set clusterName=<cluster-name> \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller \
  --set region=us-west-1 --set vpcId=<vpc#> -n kube-system
```

Step 7: We need to create Amazon ECR Private Repositories for both Tiers (Frontend & Backend)

Click on Create repository



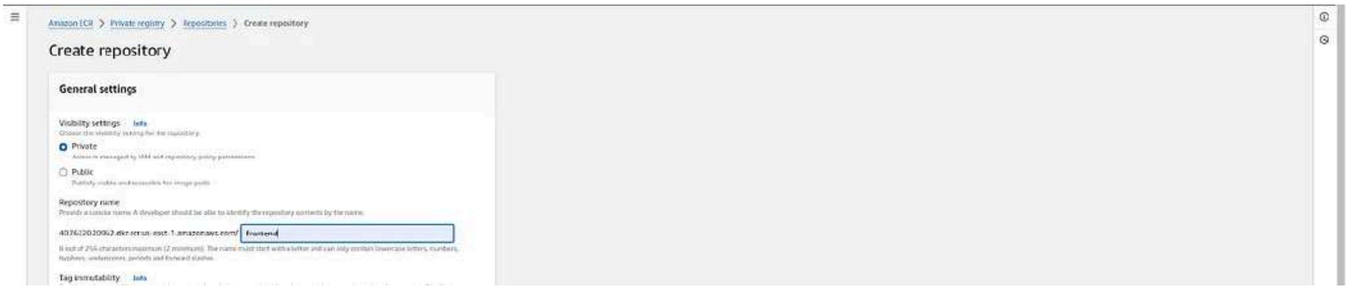
Private repositories

Repositories

Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type

No repositories
No repositories were found

Select the Private option to provide the repository and click on **Save**.



Amazon ECR > Private registry > Repositories > Create repository

Create repository

General settings

Visibility settings Info

● Private

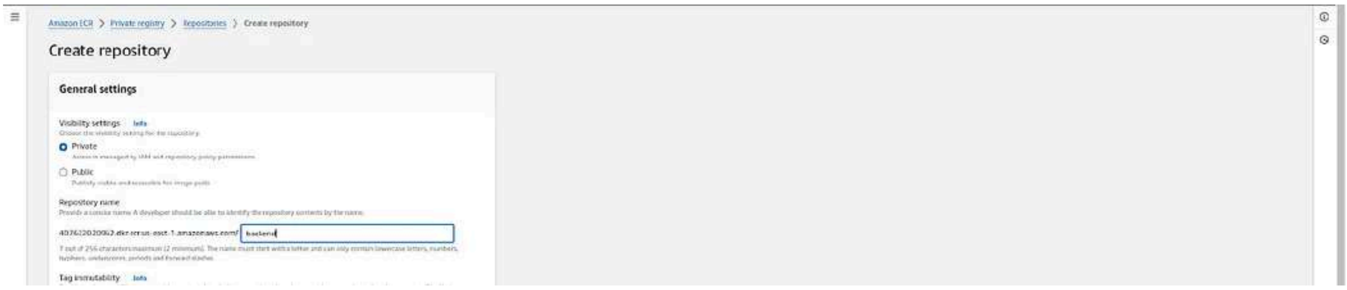
○ Public

Repository name

407622020962.dkr.ecr.us-east-1.amazonaws.com/ frontend

Tag immutability Info

Do the same for the backend repository and click on **Save**



Amazon ECR > Private registry > Repositories > Create repository

Create repository

General settings

Visibility settings Info

● Private

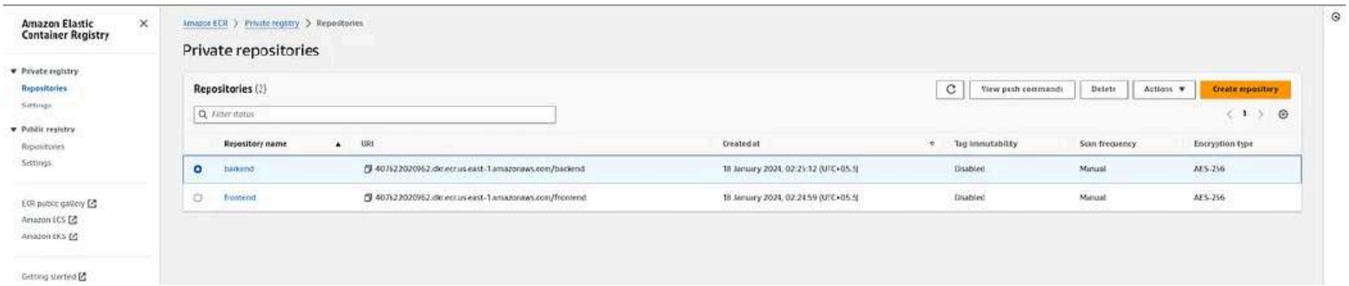
○ Public

Repository name

407622020962.dkr.ecr.us-east-1.amazonaws.com/ backend

Tag immutability Info

Now, we have set up our ECR Private Repository and



Amazon ECR > Private registry > Repositories

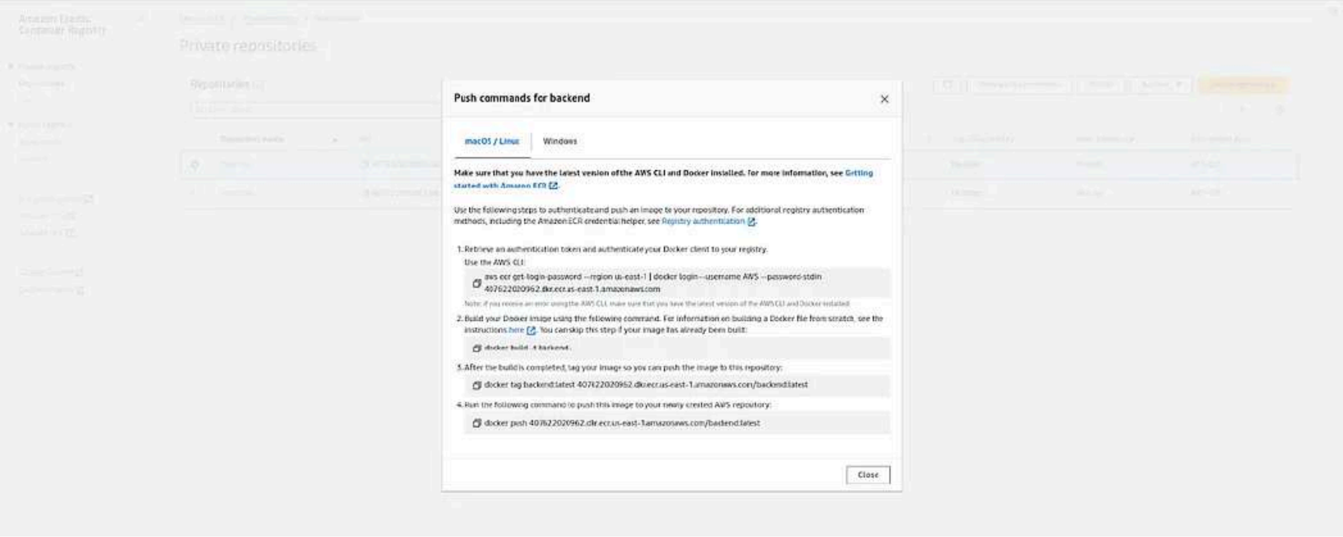
Private repositories

Repositories (2)

	Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type
●	backend	407622020962.dkr.ecr.us-east-1.amazonaws.com/backend	18 January 2024, 02:25:12 (UTC+05.5)	Disabled	Manual	AES-256
○	frontend	407622020962.dkr.ecr.us-east-1.amazonaws.com/frontend	18 January 2024, 02:24:59 (UTC+05.5)	Disabled	Manual	AES-256

Now, we need to configure ECR locally because we have to upload our images to Amazon ECR.

Copy the **1st** command for login



Push commands for backend

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see Getting started with Amazon ECR.

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see Registry authentication.

1. Retrieve an authentication token and authenticate your Docker client to your registry.
   Use the AWS CLI:

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 407622020962.dkr.ecr.us-east-1.amazonaws.com
```

   Note: if you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.

2. Build your Docker image using the following command. For information on building a Docker file from scratch, see the instructions here. You can skip this step if your image has already been built:

```
docker build -t backend .
```

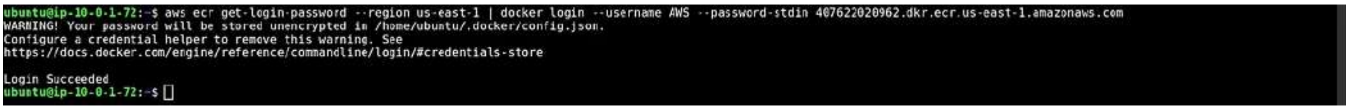
3. After the build is completed, tag your image so you can push the image to this repository:

```
docker tag backend:latest 407622020962.dkr.ecr.us-east-1.amazonaws.com/backend:latest
```

4. Run the following command to push this image to your newly created AWS repository:

```
docker push 407622020962.dkr.ecr.us-east-1.amazonaws.com/backend:latest
```

Now, run the copied command on your **Jenkins Server.**

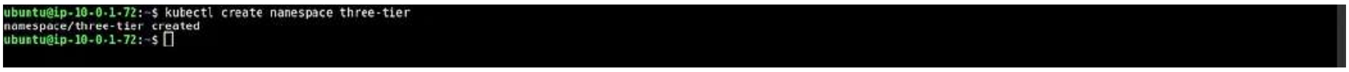


```
ubuntu@ip-10-0-1-72:~$ aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 407622020962.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu@ip-10-0-1-72:~$ 
```

Step 8: Install & Configure ArgoCD

We will be deploying our application on a three-tier namespace. To do that, we will create a three-tier namespace on EKS

```
kubectl create namespace three-tier
```



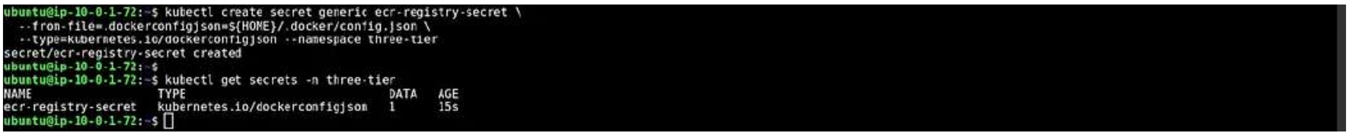
```
ubuntu@ip-10-0-1-72:~$ kubectl create namespace three-tier
namespace/three-tier created
ubuntu@ip-10-0-1-72:~$ 
```

As you know, Our two ECR repositories are private. So, when we try to push images to the ECR Repos it will give us the error **Imagepullerror.**

To get rid of this error, we will create a secret for our ECR Repo by the below command and then, we will add this secret to the deployment file.

Note: The Secrets are coming from the .docker/config.json file which is created while login the ECR in the earlier steps

```
kubectl create secret generic ecr-registry-secret \
  --from-file=.dockerconfigjson=${HOME}/.docker/config.json \
  --type=kubernetes.io/dockerconfigjson --namespace three-tier
kubectl get secrets -n three-tier
```

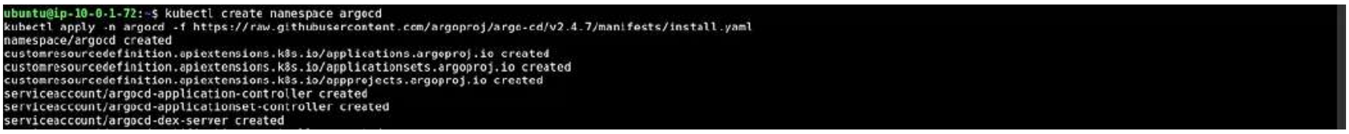


```
ubuntu@ip-10-0-1-72:~$ kubectl create secret generic ecr-registry-secret \
  --fron-file=.dockerconfigjson=${HOME}/.docker/config.json \
  --type=kubernetes.io/dockerconfigjson --namespace three-tier
secret/ecr-registry-secret created
ubuntu@ip-10-0-1-72:~$
ubuntu@ip-10-0-1-72:~$ kubectl get secrets -n three-tier
NAME                  TYPE                             DATA   AGE
ecr-registry-secret   kubernetes.io/dockerconfigjson   1      15s
ubuntu@ip-10-0-1-72:~$ 
```

Now, we will install argoCD.

To do that, create a separate namespace for it and apply the argocd configuration for installation.

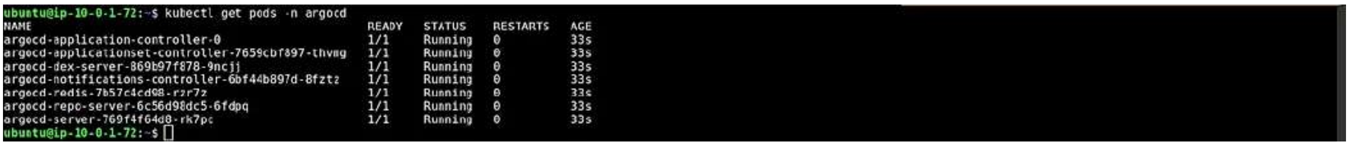
```
kubectl create namespace argocd
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/v
```



```
ubuntu@ip-10-0-1-72:~$ kubectl create nanespace argocd
kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/v2.4.7/manifests/install.yaml
namespace/argocd created
customresourcedefinition.apiextensions.k8s.io/applications.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/applicationsets.argoproj.io created
customresourcedefinition.apiextensions.k8s.io/appprojects.argoproj.io created
serviceaccount/argocd-application-controller created
serviceaccount/argocd-applicationset-controller created
serviceaccount/argocd-dex-server created
```

All pods must be running, to validate run the below command

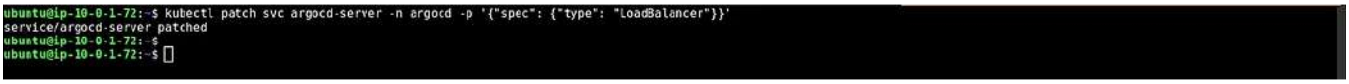
```
kubectl get pods -n argocd
```



```
ubuntu@ip-10-0-1-72:~$ kubectl get pods -n argocd
NAME                                                READY   STATUS    RESTARTS   AGE
argocd-application-controller-0                     1/1     Running   0          33s
argocd-applicationset-controller-7659cbf897-thvmg   1/1     Running   0          33s
argocd-dex-server-869b97f878-9ncjj                  1/1     Running   0          33s
argocd-notifications-controller-6bf44b897d-8fztz    1/1     Running   0          33s
argocd-redis-7b57c4cd98-rzr7z                       1/1     Running   0          33s
argocd-repo-server-6c56d98dc5-6fdpq                 1/1     Running   0          33s
argocd-server-769f4f64d8-rk7pc                      1/1     Running   0          33s
ubuntu@ip-10-0-1-72:~$ 
```

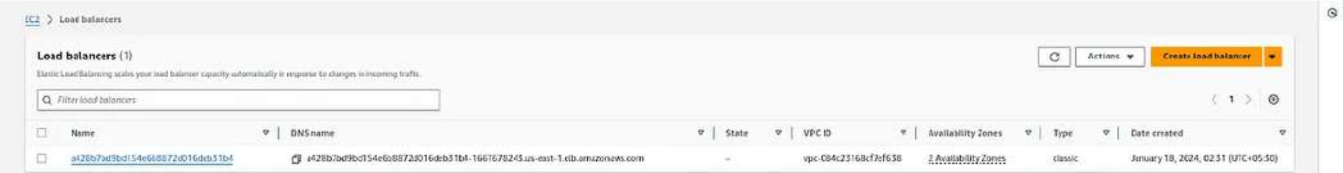
Now, expose the argoCD server as LoadBalancer using the below command

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



```
ubuntu@ip-10-0-1-72:~$ kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'
service/argocd-server patched
ubuntu@ip-10-0-1-72:~$
ubuntu@ip-10-0-1-72:~$ 
```

You can validate whether the Load Balancer is created or not by going to the AWS Console



EC2 > Load balancers

Load balancers (1)

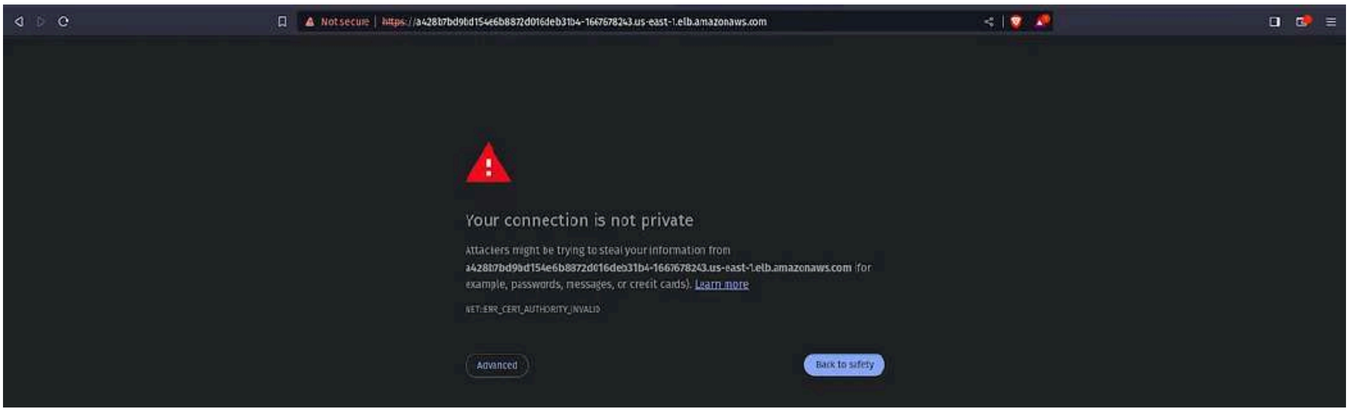
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
a428b7bd9bd154e6b8872d016deb31b4	a428b7bd9bd154e6b8872d016deb31b4-1667678243.us-east-1.elb.amazonaws.com	-	vpc-084c23168cf7ef638	2 Availability Zones	classic	January 18, 2024, 02:31 (UTC+05:30)

To access the argoCD, copy the LoadBalancer DNS and hit on your favorite browser.

You will get a warning like the below snippet.

Click on **Advanced.**



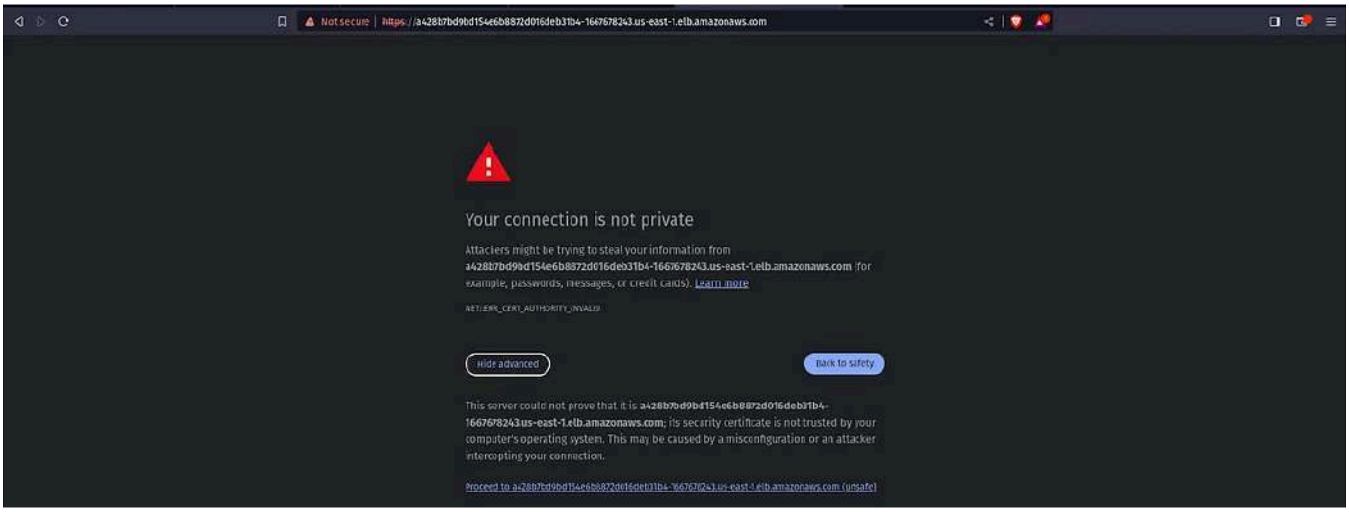
Not secure | https://a428b7bd9bd154e6b8872d016deb31b4-1667678243.us-east-1.elb.amazonaws.com

Your connection is not private

Attackers might be trying to steal your information from **a428b7bd9bd154e6b8872d016deb31b4-1667678243.us-east-1.elb.amazonaws.com** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_AUTHORITY_INVALID

Click on the below link which is appearing under **Hide advanced**



Not secure | https://a428b7bd9bd154e6b8872d016deb31b4-1667678243.us-east-1.elb.amazonaws.com

Your connection is not private

Attackers might be trying to steal your information from **a428b7bd9bd154e6b8872d016deb31b4-1667678243.us-east-1.elb.amazonaws.com** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_AUTHORITY_INVALID

Hide advanced
Back to safety

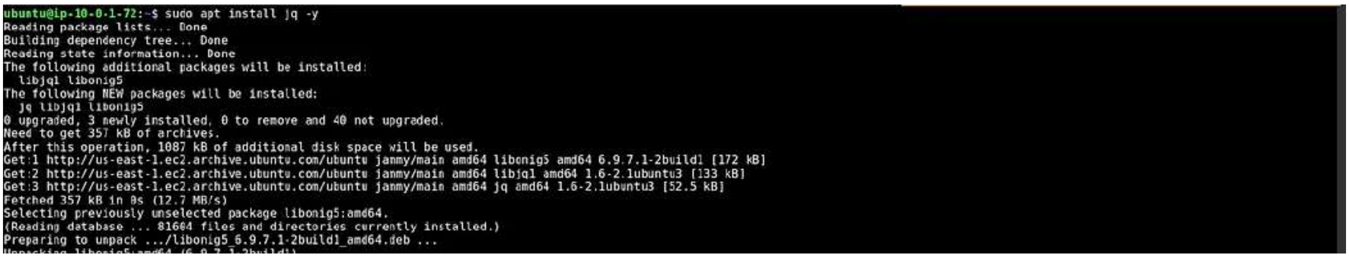
This server could not prove that it is **a428b7bd9bd154e6b8872d016deb31b4-1667678243.us-east-1.elb.amazonaws.com**; its security certificate is not trusted by your computer's operating system. This may be caused by a misconfiguration or an attacker intercepting your connection.

Proceed to a428b7bd9bd154e6b8872d016deb31b4-1667678243.us-east-1.elb.amazonaws.com (unsafe)

Now, we need to get the password for our argoCD server to perform the deployment.

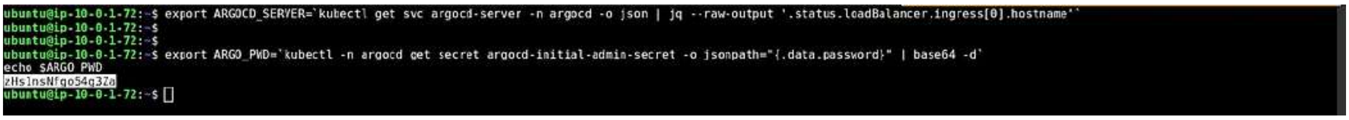
To do that, we have a pre-requisite which is **jq.** Install it by the command below.

```
sudo apt install jq -y
```



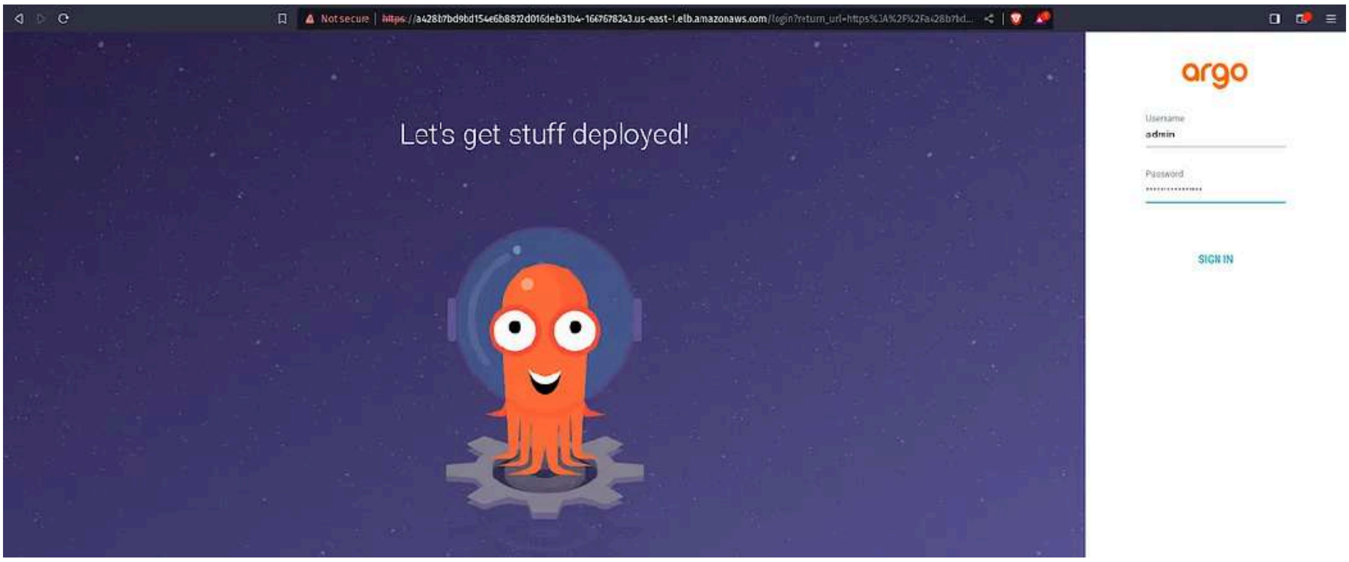
```
ubuntu@ip-10-0-1-72:~$ sudo apt install jq -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libjq1 libonig5
The following NEW packages will be installed:
  jq libjq1 libonig5
0 upgraded, 3 newly installed, 0 to remove and 40 not upgraded.
Need to get 357 kB of archives.
After this operation, 1087 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libonig5 amd64 6.9.7.1-2build1 [172 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 libjq1 amd64 1.6-2.1ubuntu3 [133 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 jq amd64 1.6-2.1ubuntu3 [52.5 kB]
Fetched 357 kB in 0s (12.7 MB/s)
Selecting previously unselected package libonig5:amd64.
(Reading database ... 81604 files and directories currently installed.)
Preparing to unpack .../libonig5_6.9.7.1-2build1_amd64.deb ...
Unpacking libonig5:amd64 (6.9.7.1-2build1)
```

```
export ARGOCD_SERVER='kubectl get svc argocd-server -n argocd -o json | jq - ra
export ARGO_PWD='kubectl -n argocd get secret argocd-initial-admin-secret -o js
echo $ARGO_PWD
```



```
ubuntu@ip-10-0-1-72:~$ export ARGOCD_SERVER=`kubectl get svc argocd-server -n argocd -o json | jq --raw-output '.status.loadBalancer.ingress[0].hostname'`
ubuntu@ip-10-0-1-72:~$
ubuntu@ip-10-0-1-72:~$
ubuntu@ip-10-0-1-72:~$ export ARGO_PWD=`kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" | base64 -d`
echo $ARGO_PWD
zHslnsNfqo54q3Za
ubuntu@ip-10-0-1-72:~$ 
```

Enter the username and password in argoCD and click on **SIGN IN.**

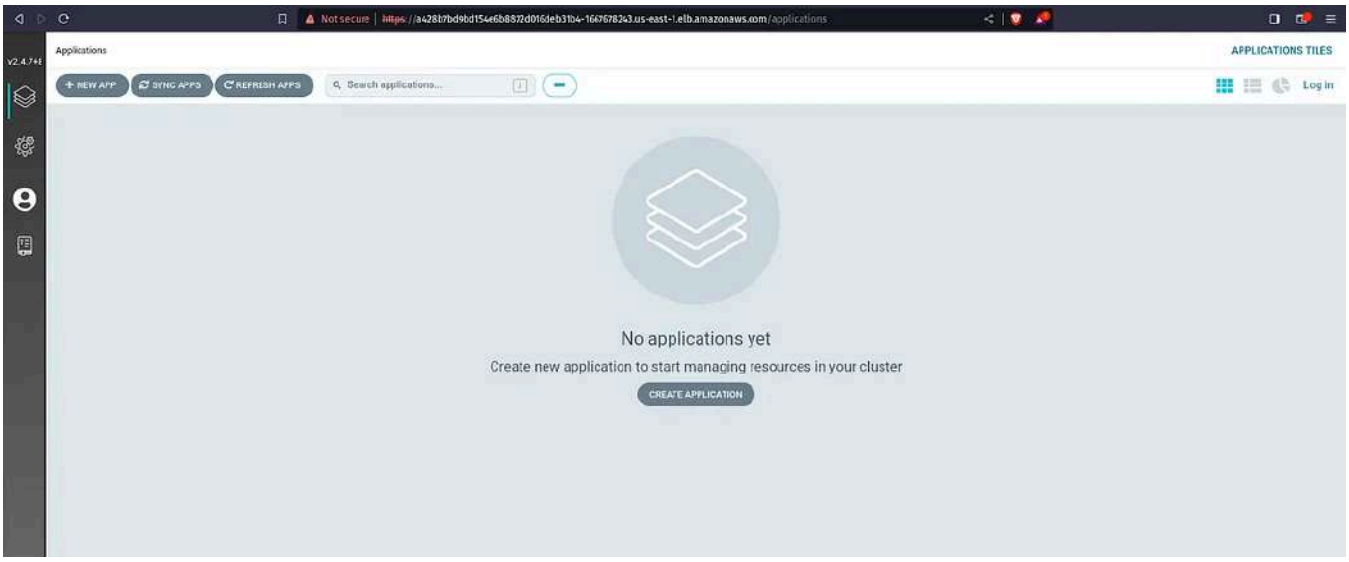


Not secure | https://a428b7bd9bd154e6b8872d016deb31b4-1667678243.us-east-1.elb.amazonaws.com/login?return_url=https%3A%2F%2Fa428b7bd...

Username
admin

Password
••••••••••••••••

Here is our ArgoCD **Dashboard.**

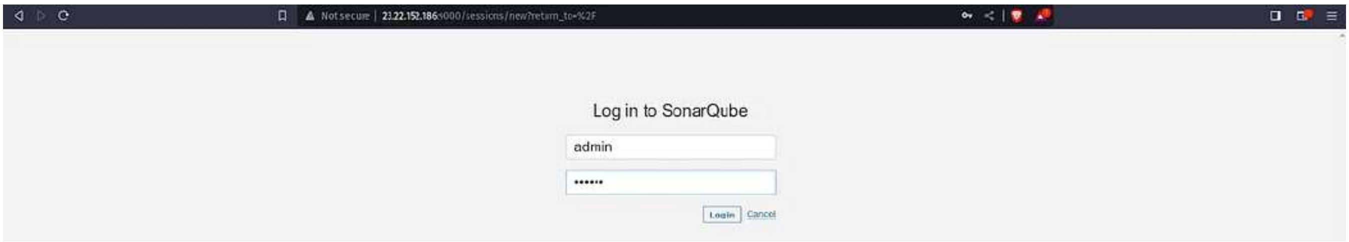

Not secure | https://a428b7bd9bd154e6b8872d016deb31b4-1667678243.us-east-1.elb.amazonaws.com/applications
v2.4.7+
Applications
APPLICATIONS TILES
+ NEW APP
SYNC APPS
REFRESH APPS
Search applications...
Log in
No applications yet
Create new application to start managing resources in your cluster
CREATE APPLICATION

Step 9: Now, we have to configure Sonarqube for our DevSecOps Pipeline

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**

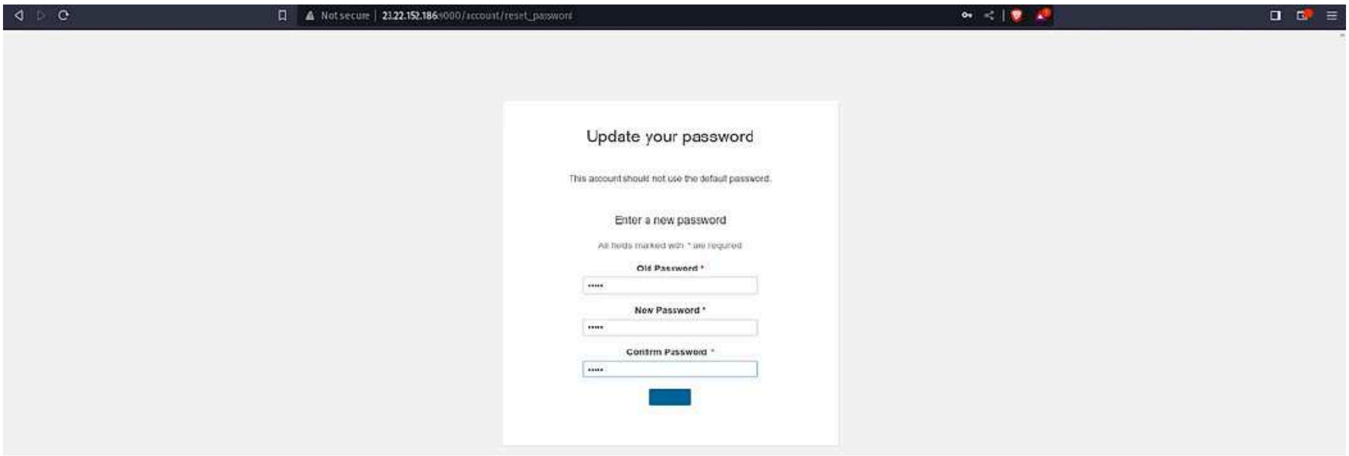
Click on **Log In.**



admin

••••••

Update the password



Update your password

This account should not use the default password.

Enter a new password

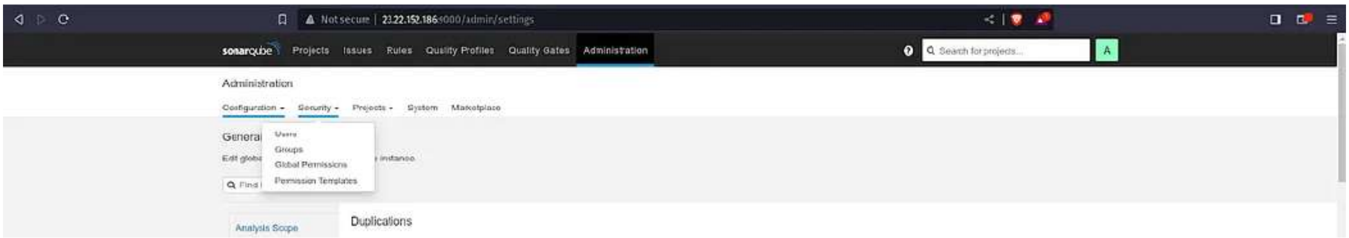
All fields marked with * are required

Old Password * •••••

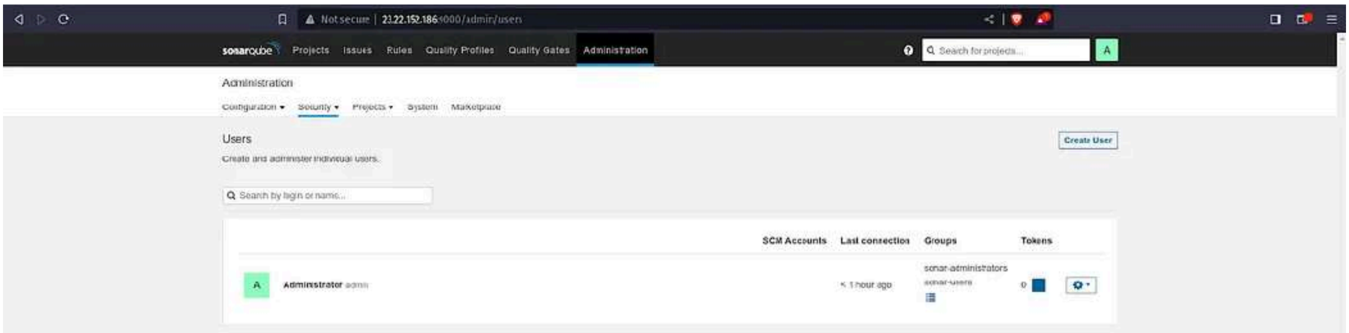
New Password * •••••

Confirm Password * •••••

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



Click on **Update tokens**

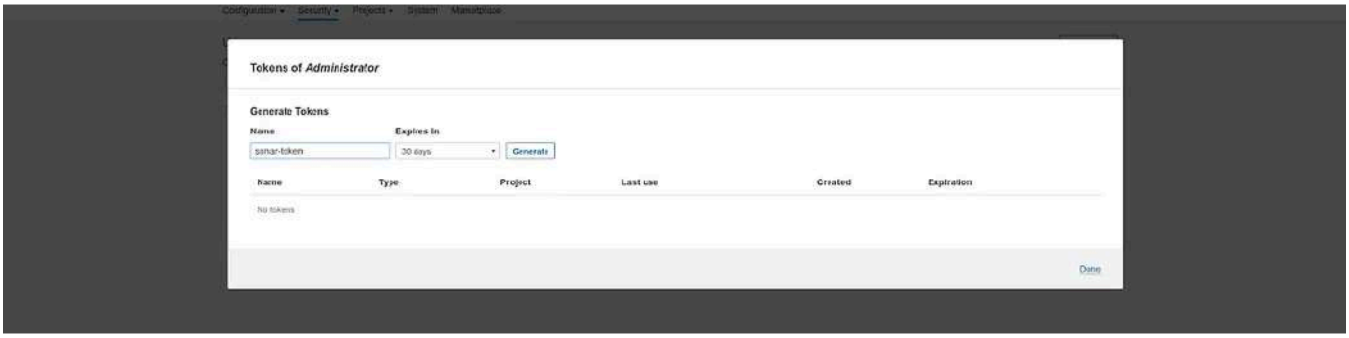


Users

Create and administer individual users.

	SCM Accounts	Last connection	Groups	Tokens
Administrator admin		< 1 hour ago	sonar-administrators sonar-users	0

Click on **Generate**



Tokens of *Administrator*

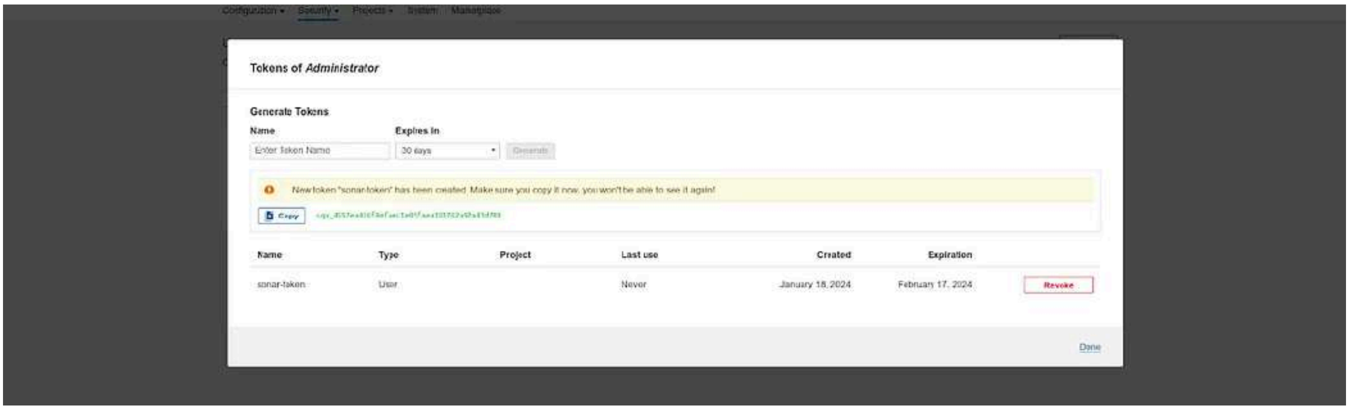
Generate Tokens

Name: sonar-token

Expires In: 30 days

Name	Type	Project	Last use	Created	Expiration
No tokens					

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



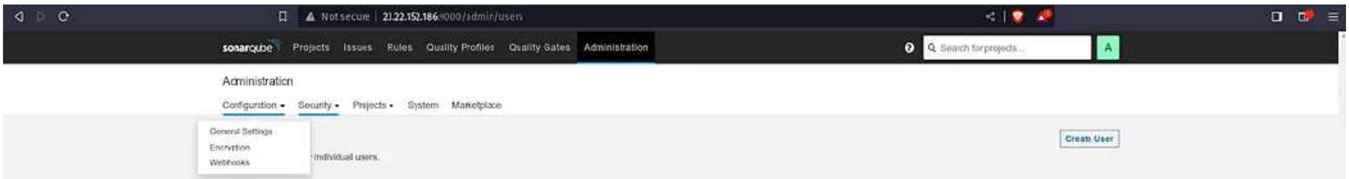
Tokens of *Administrator*

New token "sonar-token" has been created. Make sure you copy it now, you won't be able to see it again!

Name	Type	Project	Last use	Created	Expiration
sonar-token	User		Never	January 18, 2024	February 17, 2024

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

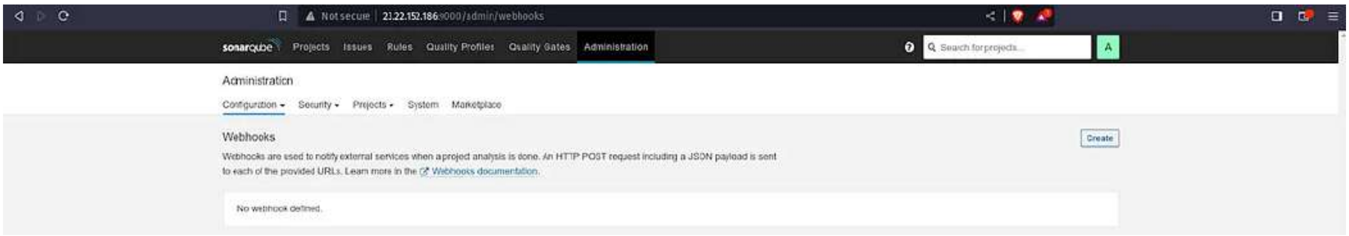
Click on **Administration** then, Configuration and select **Webhooks**


Not secure | 23.22.152.186:9000/admin/users
sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration
Search for projects...
Configuration ▾ Security ▾ Projects ▾ System Marketplace
General Settings
Encryption
Webhooks
Create User

Administration

individual users.

Click on **Create**


Not secure | 23.22.152.186:9000/admin/webhooks
sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration
Search for projects...
Configuration ▾ Security ▾ Projects ▾ System Marketplace
Create

Administration

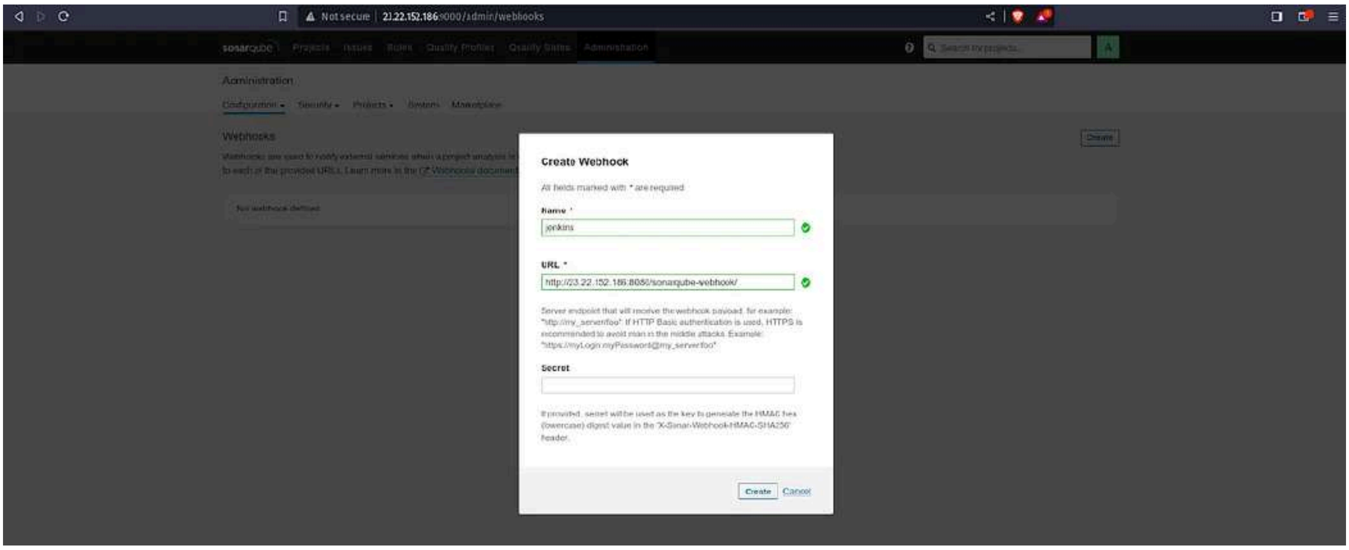
Webhooks

Webhooks are used to notify external services when a project analysis is done. An HTTP POST request including a JSON payload is sent to each of the provided URLs. Learn more in the Webhooks documentation.

No webhook defined.

Provide the name of your project and in the URL, provide the Jenkins server public IP with port 8080 add sonarqube-webhook in the suffix, and click on Create.

http://<jenkins-server-public-ip>:8080/sonarqube-webhook/



Create Webhook

All fields marked with * are required

Name *

jenkins

URL *

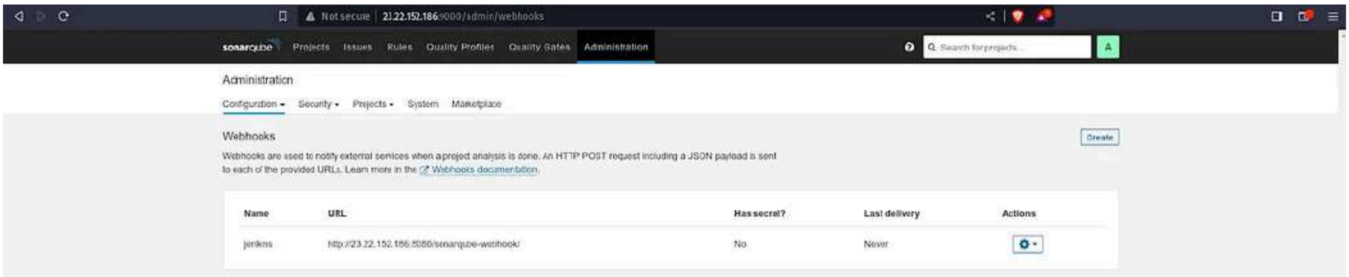
http://23.22.152.186:8080/sonarqube-webhook/

Server endpoint that will receive the webhook payload, for example: "http://my_server/foo". If HTTP Basic authentication is used, HTTPS is recommended to avoid man in the middle attacks. Example: "https://myLogin:myPassword@my_server/foo"

Secret

If provided, secret will be used as the key to generate the HMAC hex (lowercase) digest value in the 'X-Sonar-Webhook-HMAC-SHA256' header.

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


Not secure | 23.22.152.186:9000/admin/webhooks
sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration
Search for projects...
Configuration ▾ Security ▾ Projects ▾ System Marketplace
Create

Administration

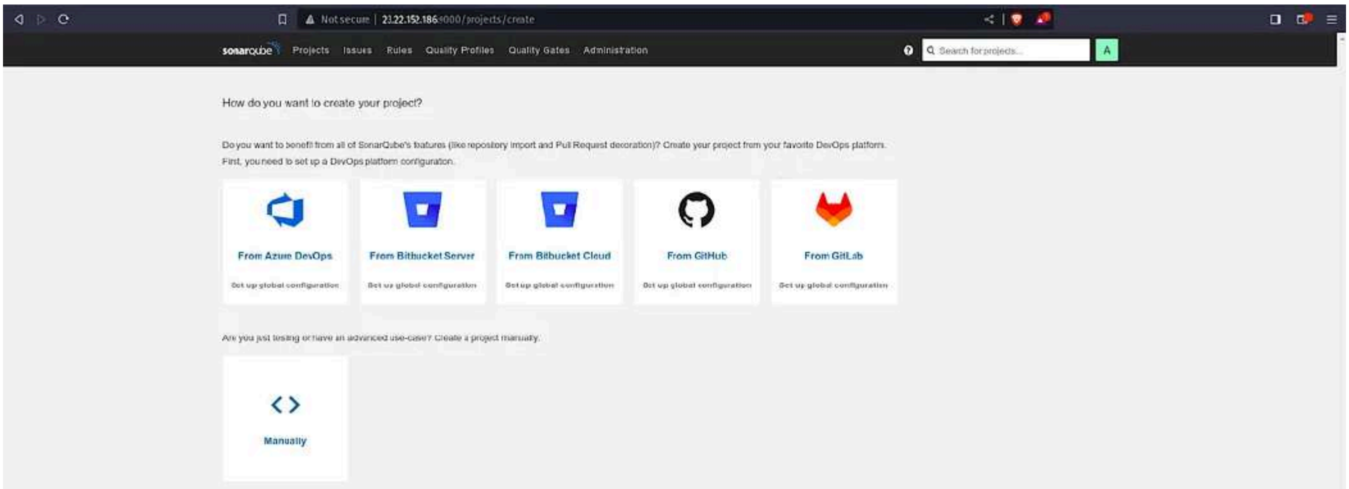
Webhooks

Webhooks are used to notify external services when a project analysis is done. An HTTP POST request including a JSON payload is sent to each of the provided URLs. Learn more in the Webhooks documentation.

Name	URL	Has secret?	Last delivery	Actions
jenkins	http://23.22.152.186:8080/sonarqube-webhook/	No	Never	⚙ ▾

Now, we have to create a Project for frontend code.

Click on **Manually.**

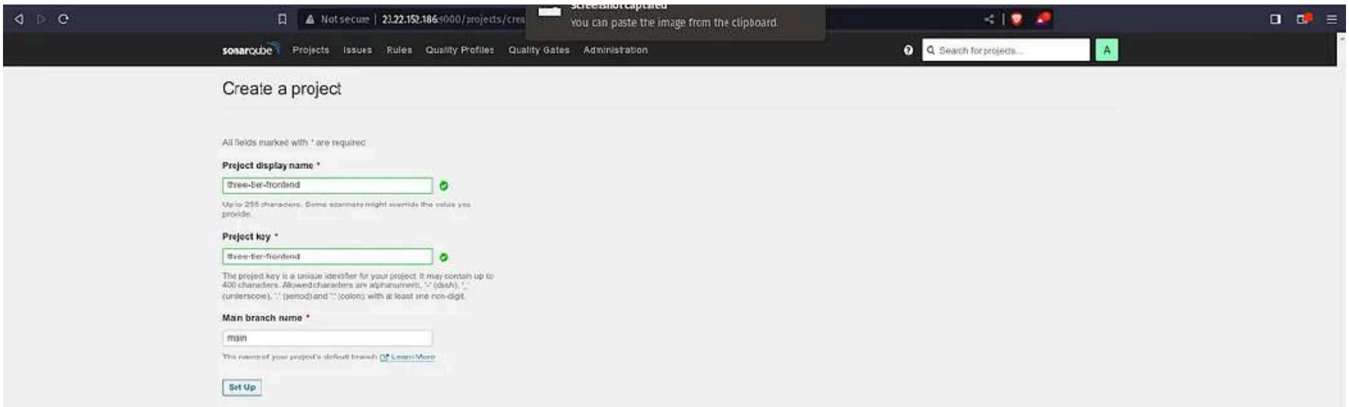

Not secure | 23.22.152.186:9000/projects/create
sonarqube
Projects
Issues
Rules
Quality Profiles
Quality Gates
Administration
Search for projects...
A
From Azure DevOps
Set up global configuration
From Bitbucket Server
Set up global configuration
From Bitbucket Cloud
Set up global configuration
From GitHub
Set up global configuration
From GitLab
Set up global configuration
Manually

How do you want to create your project?

Do you want to benefit from all of SonarQube's features (like repository import and Pull Request decoration)? Create your project from your favorite DevOps platform.
First, you need to set up a DevOps platform configuration.

Are you just testing or have an advanced use-case? Create a project manually.

Provide the display name to your **Project** and click on **Setup**



Create a project

All fields marked with * are required

Project display name *

three-tier-frontend

Up to 255 characters. Some scanners might override the value you provide.

Project key *

three-tier-frontend

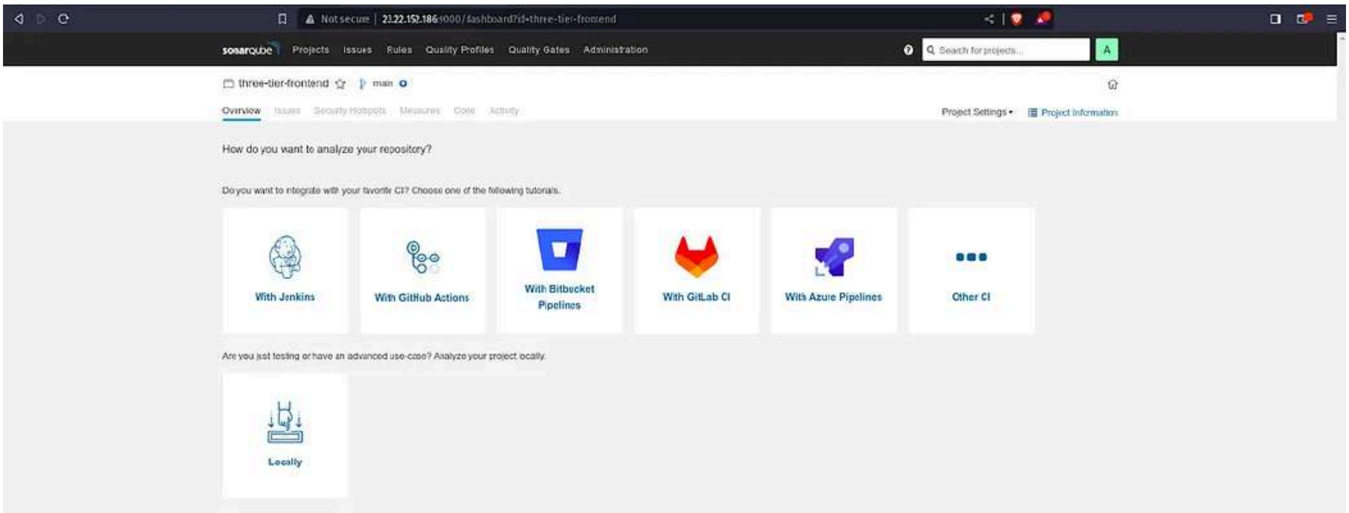
The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Main branch name *

main

The name of your project's default branch Learn More

Click on **Locally.**

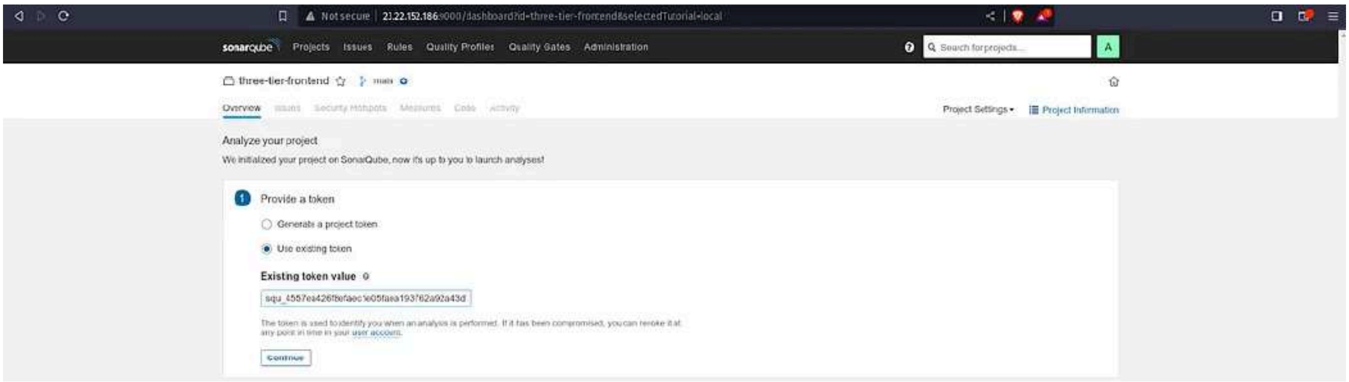


How do you want to analyze your repository?

Do you want to integrate with your favorite CI? Choose one of the following tutorials.

Are you just testing or have an advanced use-case? Analyze your project locally.

Select the **Use existing token** and click on **Continue.**



21.22.152.186:9000/dashboard?id=three-tier-frontend&selectedTutorial=local

three-tier-frontend

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

○ Generate a project token

● Use existing token

Existing token value

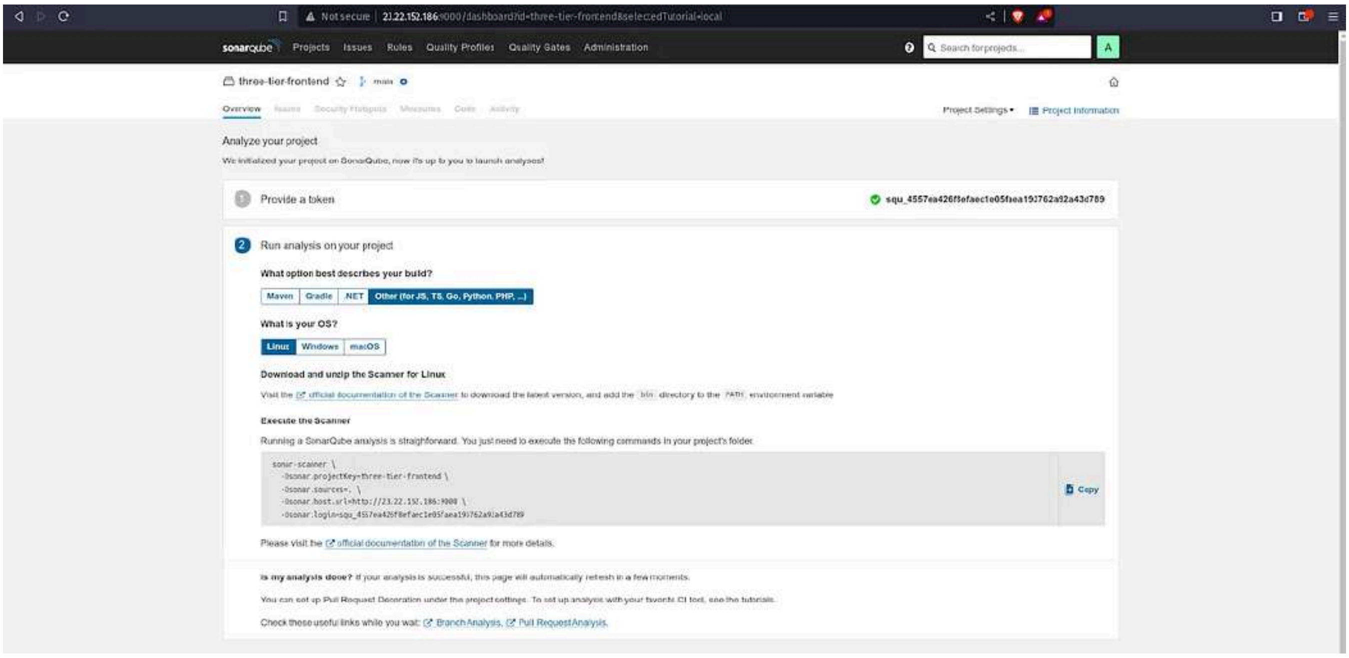
squ_4557ea426f8efaec1e05faea193762a92a43d

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your user account.

Select **Other** and **Linux** as OS.

After performing the above steps, you will get the command which you can see in the below snippet.

Now, use the command in the Jenkins Frontend Pipeline where Code Quality Analysis will be performed.



21.22.152.186:9000/dashboard?id=three-tier-frontend&selectedTutorial=local

three-tier-frontend

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token ✓ squ_4557ea426f8efaec1e05faea193762a92a43d789

2 Run analysis on your project

What option best describes your build?

Maven | Gradle | .NET | **Other (for JS, TS, Go, Python, PHP, ...)**

What is your OS?

Linux | Windows | macOS

Download and unzip the Scanner for Linux

Visit the official documentation of the Scanner to download the latest version, and add the `bin` directory to the `PATH` environment variable

Execute the Scanner

Running a SonarQube analysis is straighforward. You just need to execute the following commands in your project's folder

```
sonar-scanner \
  -Dsonar.projectKey=three-tier-frontend \
  -Dsonar.sources=. \
  -Dsonar.host.url=http://23.22.152.186:9000 \
  -Dsonar.login=squ_4557ea426f8efaec1e05faea193762a92a43d789
```

Please visit the official documentation of the Scanner for more details.

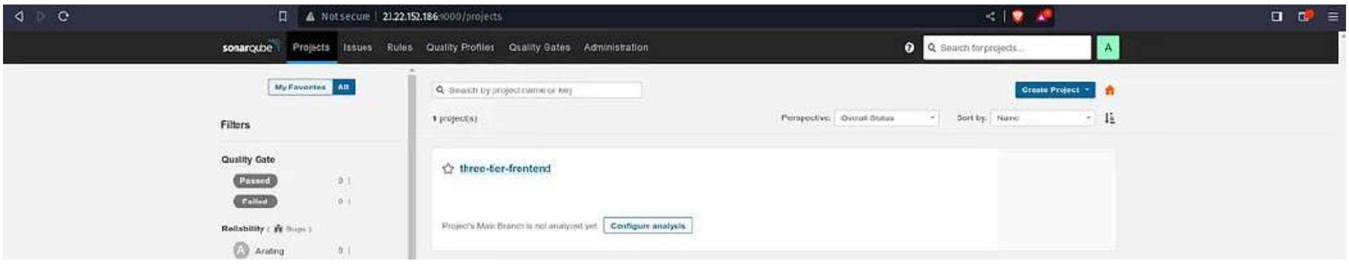
Is my analysis done? If your analysis is successful, this page will automatically refresh in a few moments.

You can set up Pull Request Decoration under the project settings. To set up analysis with your favorite CI tool, see the tutorials.

Check these useful links while you wait: Branch Analysis, Pull Request Analysis.

Now, we have to create a Project for backend code.

Click on **Create Project.**



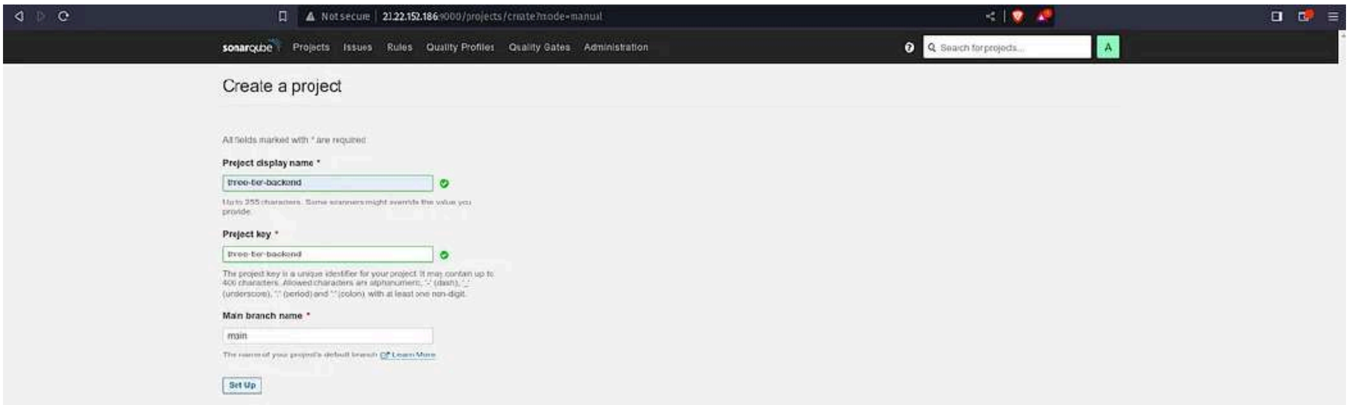
21.22.152.186:9000/projects

1 project(s)

three-tier-frontend

Project's Main Branch is not analyzed yet

Provide the name of your project name and click on **Set up.**



Create a project

All fields marked with * are required

Project display name *: three-tier-backend

Up to 255 characters. Some scanners might override the value you provide.

Project key *: three-tier-backend

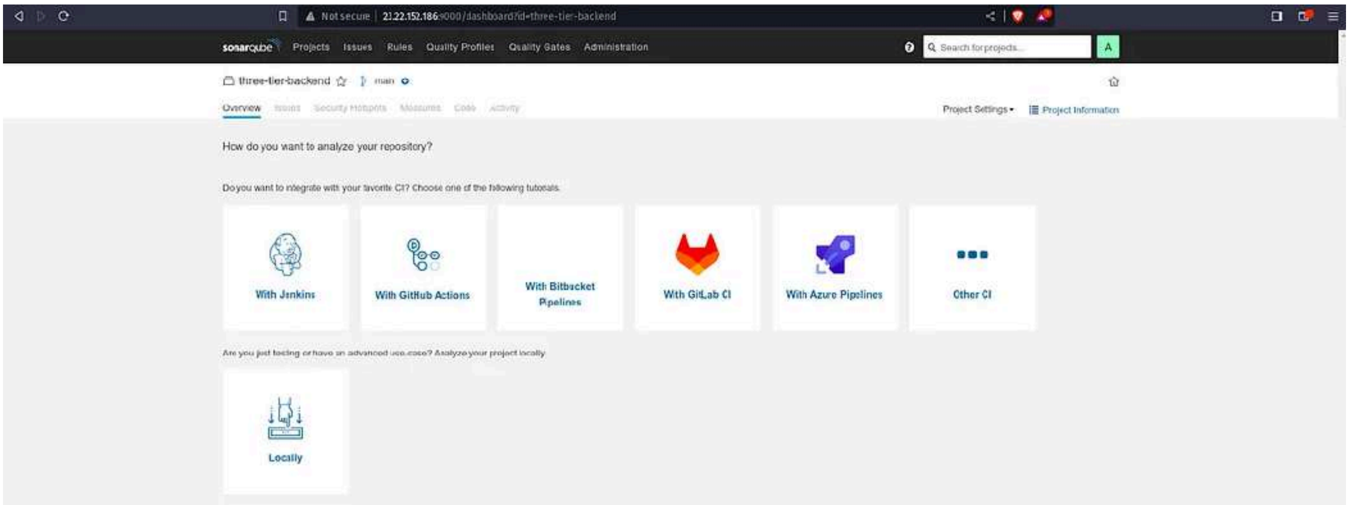
The project key is a unique identifier for your project. It may contain up to 400 characters. Allowed characters are alphanumeric, '-' (dash), '_' (underscore), '.' (period) and ':' (colon), with at least one non-digit.

Main branch name *: main

The name of your project's default branch Learn More

Set Up

Click on **Locally.**

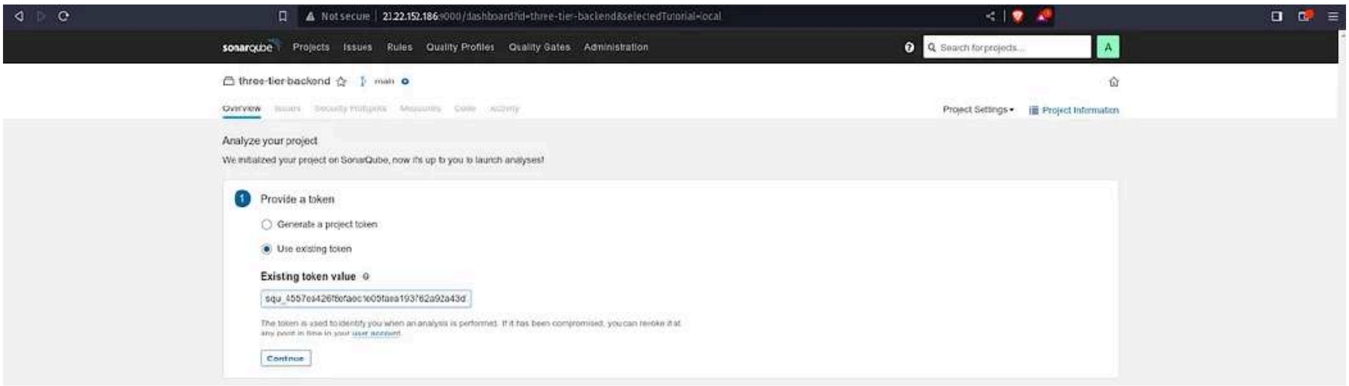


How do you want to analyze your repository?

Do you want to integrate with your favorite CI? Choose one of the following tutorials.

Are you just testing or have an advanced use-case? Analyze your project locally.

Select the **Use existing token** and click on **Continue.**



Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token

○ Generate a project token

● Use existing token

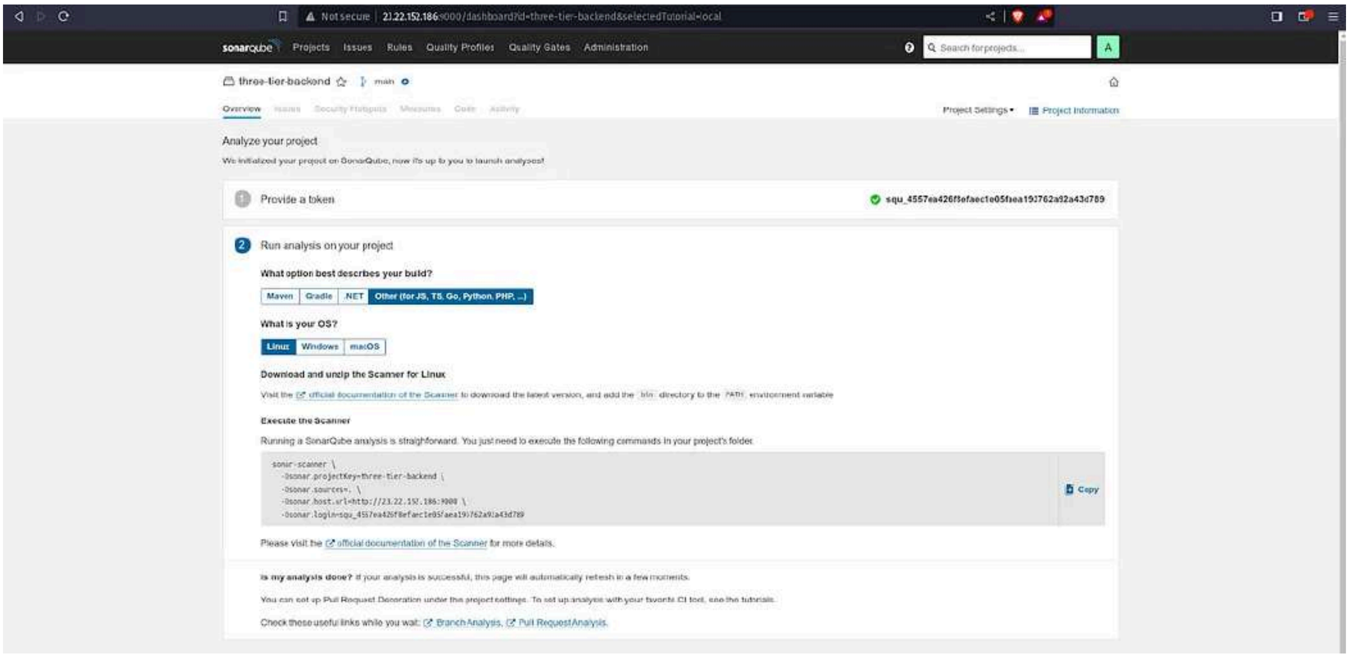
Existing token value: squ_4557ea426f8efaec1e05faea193762a92a43d

The token is used to identify you when an analysis is performed. If it has been compromised, you can revoke it at any point in time in your user account.

Select **Other** and **Linux** as OS.

After performing the above steps, you will get the command which you can see in the below snippet.

Now, use the command in the Jenkins Backend Pipeline where Code Quality Analysis will be performed.



Not secure | 21.22.152.186:9000/dashboard?id=three-tier-backend&selectedTutorial=local

Analyze your project

We initialized your project on SonarQube, now it's up to you to launch analyses!

1 Provide a token ✓ squ_4557ea426f8efaec1e05faea191762a92a43d789

2 Run analysis on your project

What option best describes your build?

Maven | Gradle | .NET | Other (for JS, TS, Go, Python, PHP, ...)

What is your OS?

Linux | Windows | macOS

Download and unzip the Scanner for Linux

Visit the official documentation of the Scanner to download the latest version, and add the `bin` directory to the `PATH` environment variable

Execute the Scanner

Running a SonarQube analysis is straightforward. You just need to execute the following commands in your project's folder.

```
sonar-scanner \
  -Dsonar.projectKey=three-tier-backend \
  -Dsonar.sources=. \
  -Dsonar.host.url=http://21.22.152.186:9000 \
  -Dsonar.login=squ_4557ea426f8efaec1e05faea191762a92a43d789
```

Please visit the official documentation of the Scanner for more details.

Is my analysis done? If your analysis is successful, this page will automatically refresh in a few moments.

You can set up Pull Request Decoration under the project settings. To set up analysis with your favorite CI tool, see the tutorials.

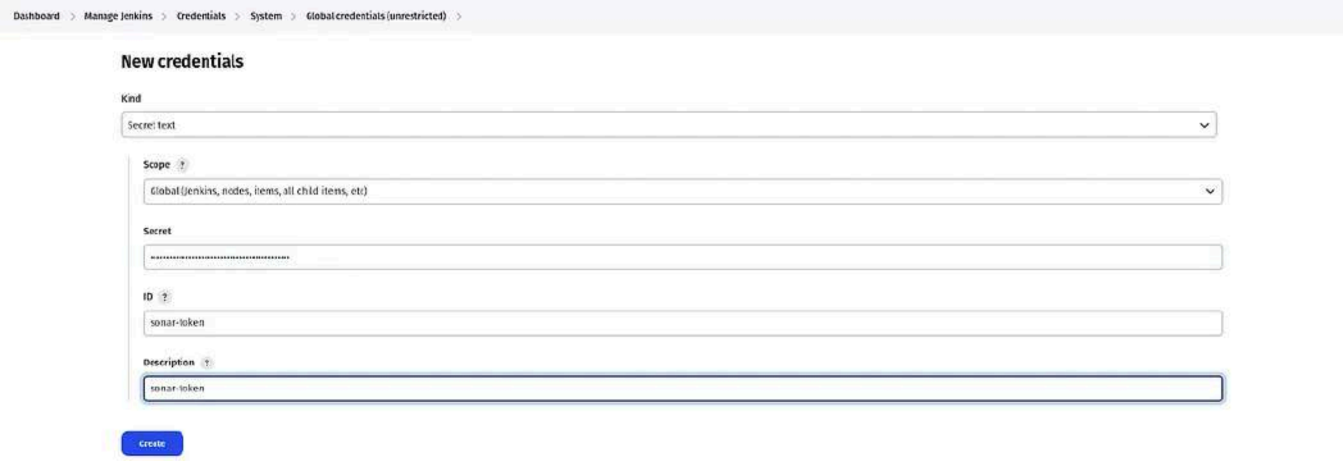
Check these useful links while you wait: Branch Analysis, Pull Request Analysis.

Now, we have to store the sonar credentials.

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

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

Click on **Create**



Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret: ••

ID: sonar-token

Description: sonar-token

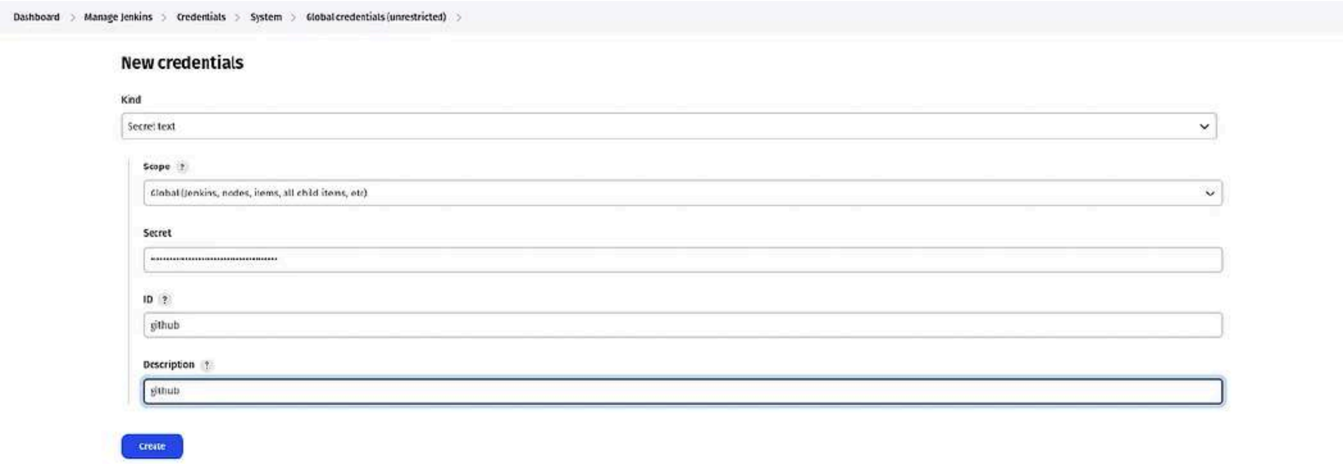
Now, we have to store the GitHub Personal access token to push the deployment file which will be modified in the pipeline itself for the ECR image.

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 into the Jenkins



Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret: ••

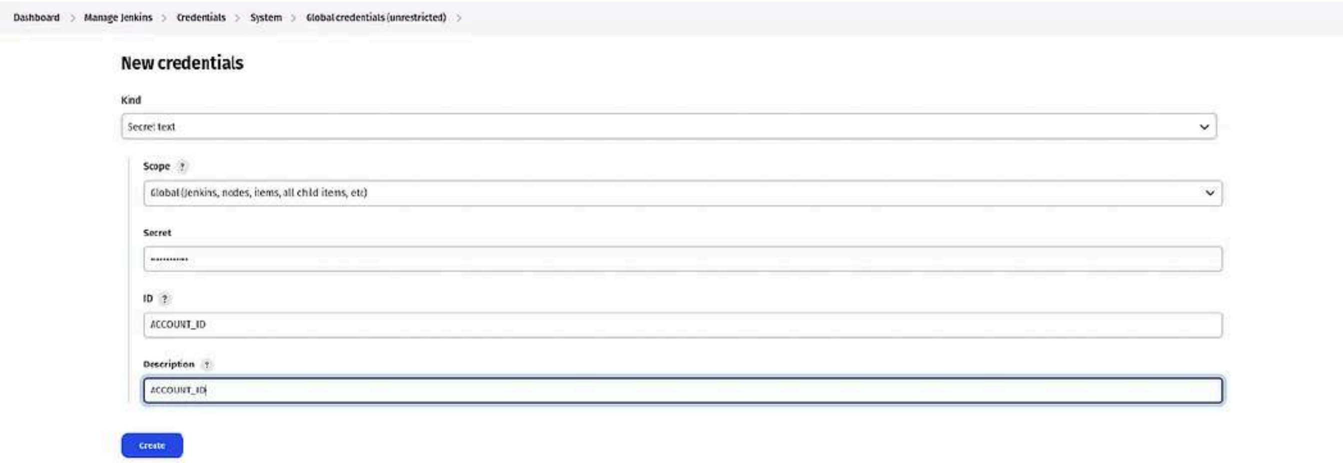
ID: github

Description: github

Now, according to our Pipeline, we need to add an Account ID in the Jenkins credentials because of the ECR repo URI.

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

Click on **Create**



Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret: ••••••••••••

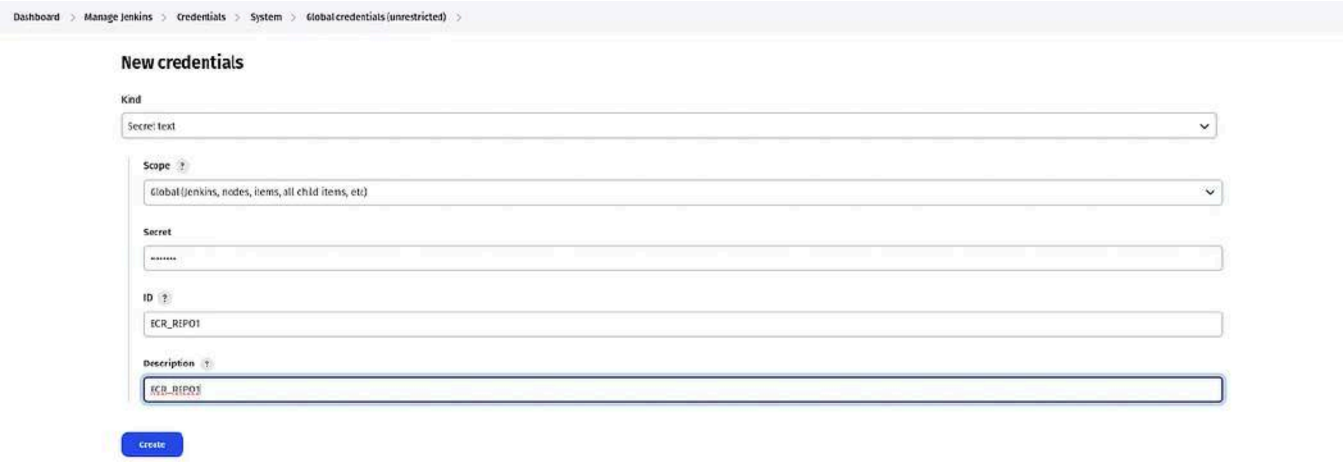
ID: ACCOUNT_ID

Description: ACCOUNT_ID

Now, we need to provide our ECR image name for frontend which is **frontend** only.

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

Click on **Create**



Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind: Secret text

Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret: •••••••

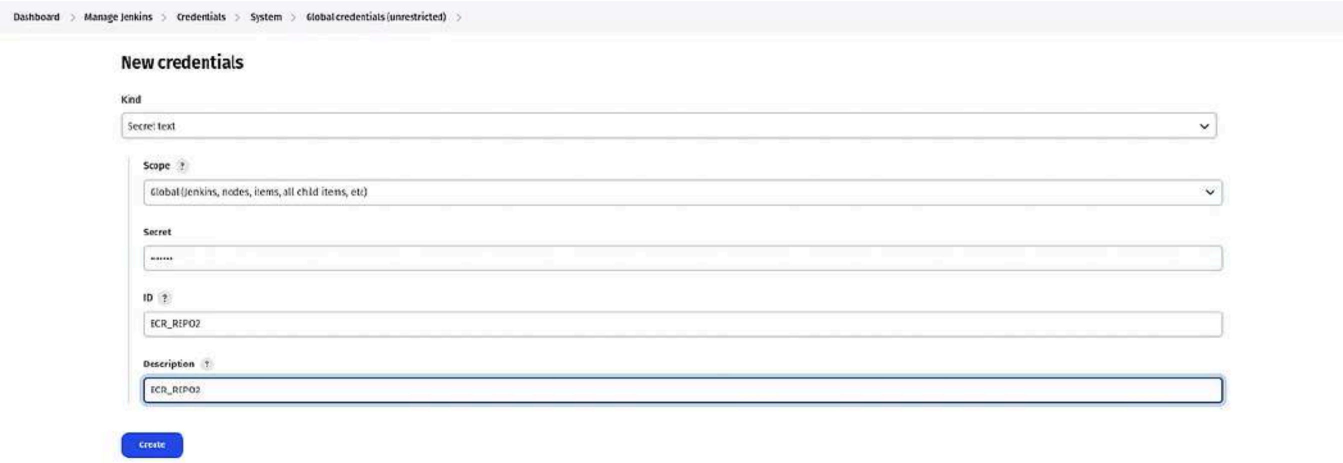
ID: ECR_REPO1

Description: ECR_REPO1

Now, we need to provide our ECR image name for the backend which is **backend** only.

Select the kind as **Secret text,** paste your backend repo name in Secret, and keep other things as it is.

Click on **Create**



Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

New credentials

Kind: Secret text

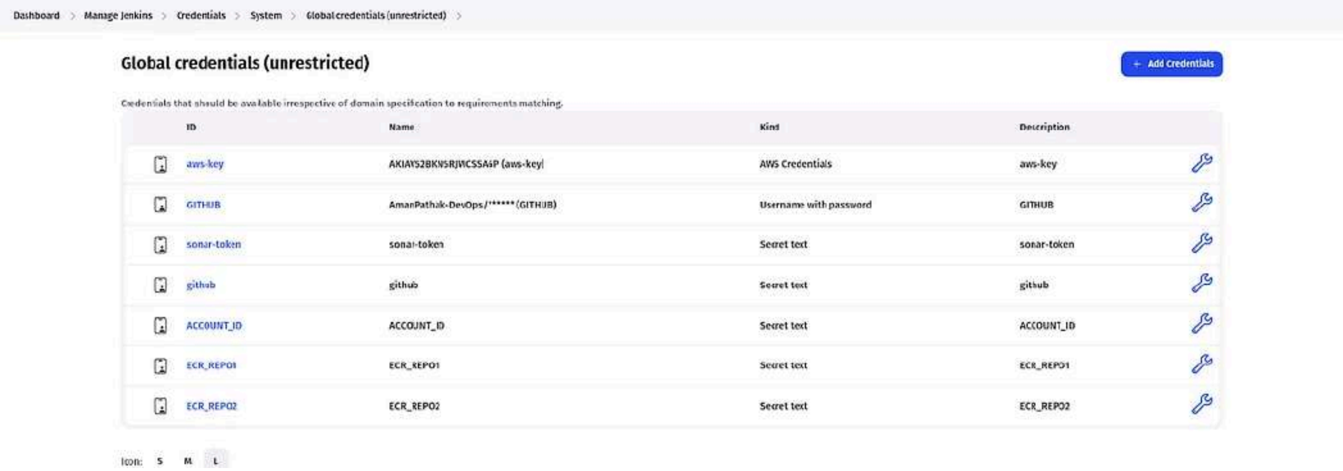
Scope: Global (Jenkins, nodes, items, all child items, etc)

Secret: ••••••

ID: ECR_REPO2

Description: ECR_REPO2

Final Snippet of all Credentials that we needed to implement this project.



Dashboard > Manage Jenkins > Credentials > System > Global credentials (unrestricted) >

Global credentials (unrestricted)

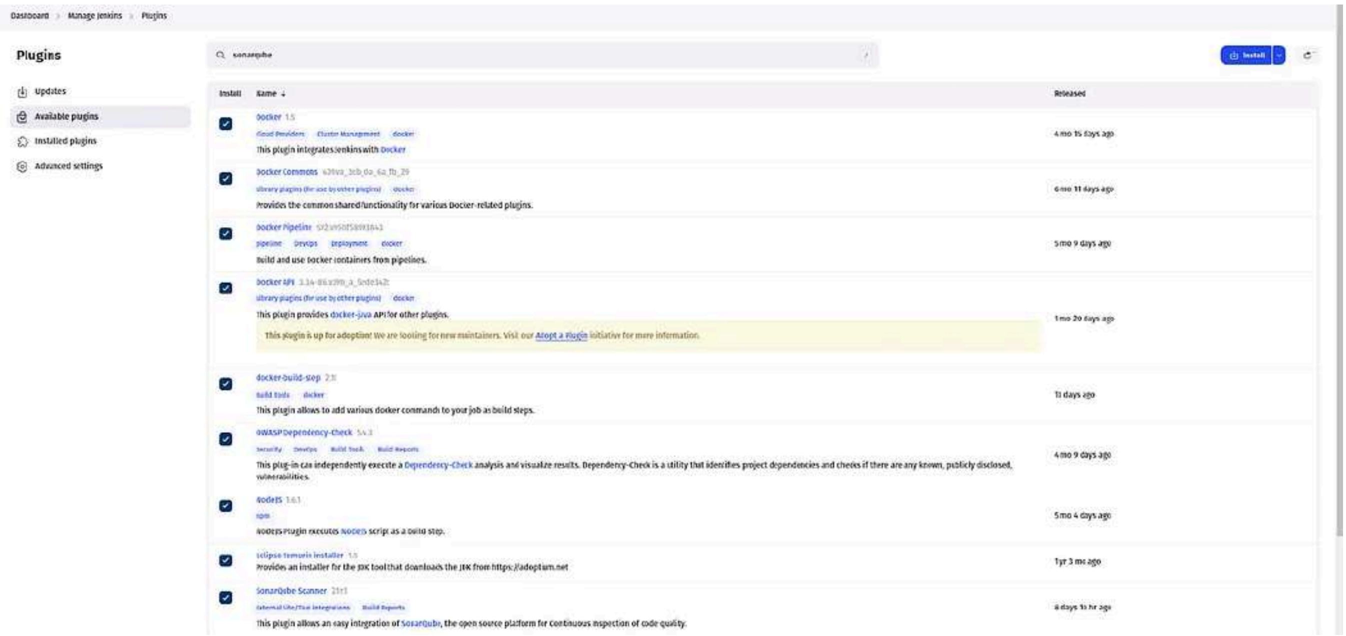
Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
aws-key	AKIAYS2BKN5RJVICSSA6P (aws-key)	AWS Credentials	aws-key
GITHUB	AmanPathak-DevOps/****** (GITHUB)	Username with password	GITHUB
sonar-token	sonar-token	Secret text	sonar-token
github	github	Secret text	github
ACCOUNT_ID	ACCOUNT_ID	Secret text	ACCOUNT_ID
ECR_REPO1	ECR_REPO1	Secret text	ECR_REPO1
ECR_REPO2	ECR_REPO2	Secret text	ECR_REPO2

Step 10: Install the required plugins and configure the plugins to deploy our Three-Tier Application

Install the following plugins by going to **Dashboard -> Manage Jenkins -> Plugins -> Available Plugins**

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



Dashboard > Manage Jenkins > Plugins

Plugins

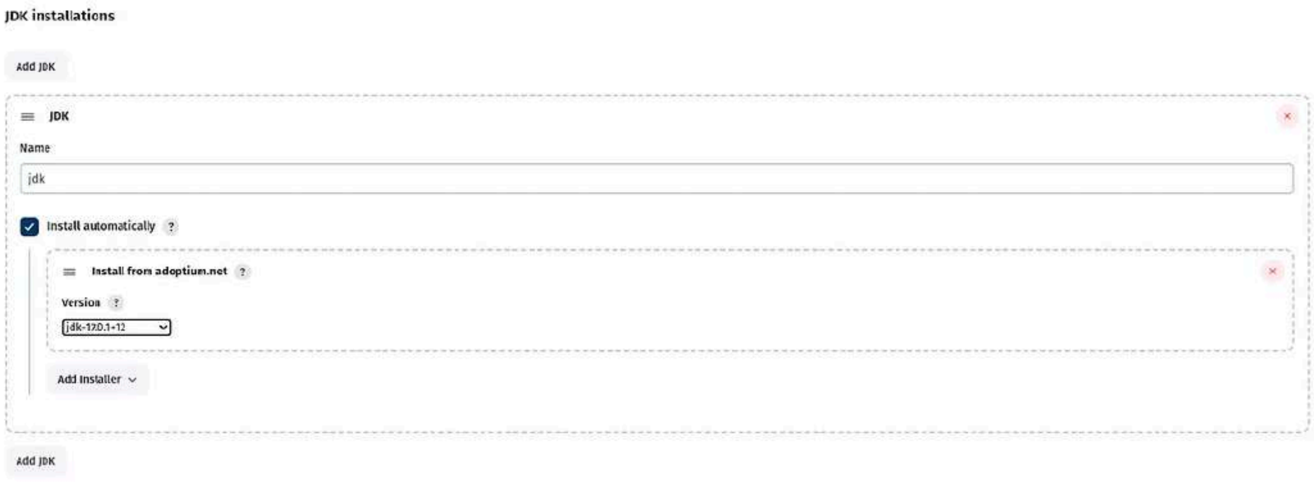
Install	Name	Released
☑	Docker 1.5 Cloud Providers Cluster Management docker This plugin integrates Jenkins with Docker	4 mo 15 days ago
☑	Docker Commons 439.va_3cb_0a_6a_fb_29 Library plugins (for use by other plugins) docker Provides the common shared functionality for various Docker-related plugins.	6 mo 11 days ago
☑	Docker Pipeline 572.v950f58993843 pipeline DevOps Deployment docker Build and use Docker containers from pipelines.	5 mo 9 days ago
☑	Docker API 3.3.4-86.v39b_a_5ede342c Library plugins (for use by other plugins) docker This plugin provides docker-java API for other plugins. This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	1 mo 20 days ago
☑	docker-build-step 2.11 Build Tools docker This plugin allows to add various docker commands to your job as build steps.	11 days ago
☑	OWASP Dependency-Check 5.4.3 Security DevOps Build Tools Build Reports This plug-in can independently execute a Dependency-Check analysis and visualize results. Dependency-Check is a utility that identifies project dependencies and checks if there are any known, publicly disclosed, vulnerabilities.	4 mo 9 days ago
☑	NodeJS 1.6.1 npm NodeJS Plugin executes NodeJS script as a build step.	5 mo 4 days ago
☑	Eclipse Temurin installer 1.5 Provides an installer for the JDK tool that downloads the JDK from https://adoptium.net	1 yr 3 mo ago
☑	SonarQube Scanner 2.17.1 External Site/Tool Integrations Build Reports This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.	8 days 15 hr ago

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 the below snippet.



JDK installations

Add JDK

JDK

Name: jdk

☑ Install automatically

Install from adoptium.net

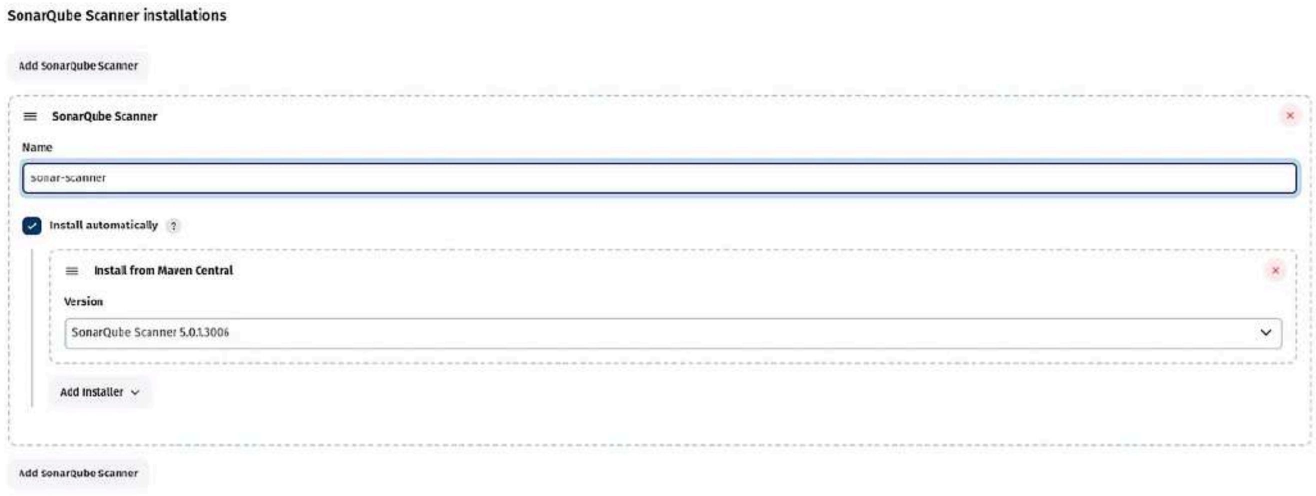
Version: jdk-17.0.1+12

Add Installer

Add JDK

Now, we will configure the sonarqube-scanner

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



SonarQube Scanner installations

Add SonarQube Scanner

SonarQube Scanner

Name: sonar-scanner

☑ Install automatically

Install from Maven Central

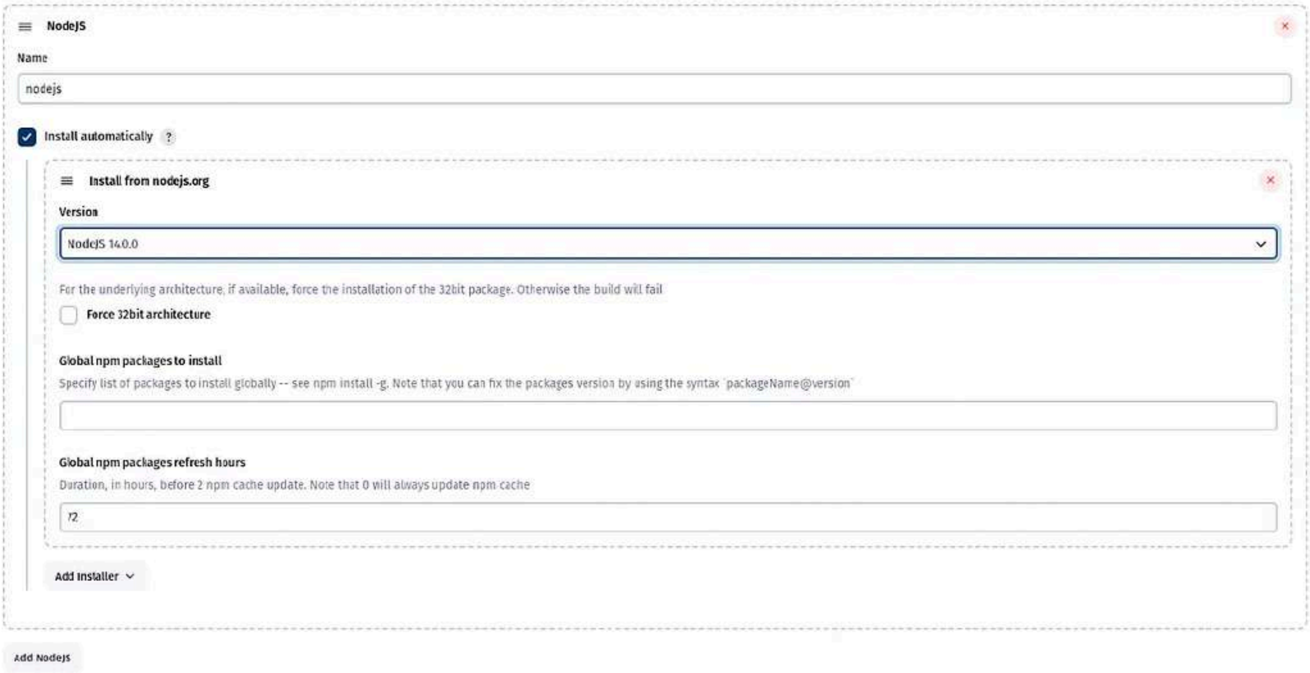
Version: SonarQube Scanner 5.0.1.3006

Add Installer

Add SonarQube Scanner

Now, we will configure **nodejs**

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



NodeJS

Name

nodejs

☑ Install automatically

Install from nodejs.org

Version

NodeJS 14.0.0

For the underlying architecture, if available, force the installation of the 32bit package. Otherwise the build will fail

□ Force 32bit architecture

Global npm packages to install

Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax `packageName@version`

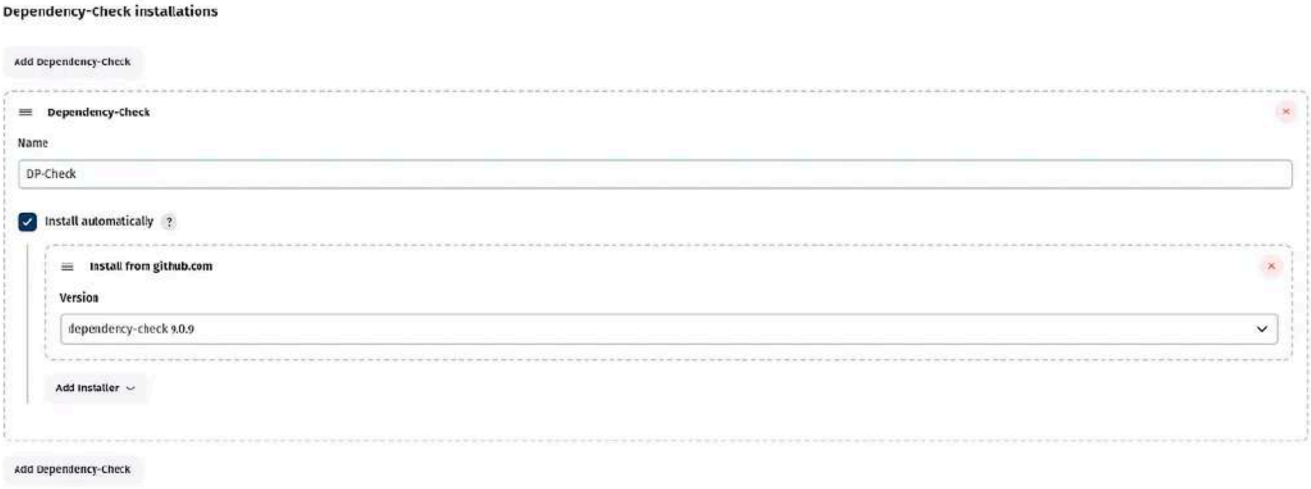
Global npm packages refresh hours

Duration, in hours, before 2 npm cache update. Note that 0 will always update npm cache

72

Now, we will configure the OWASP Dependency check

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



Dependency-Check installations

Dependency-Check

Name

DP-Check

☑ Install automatically

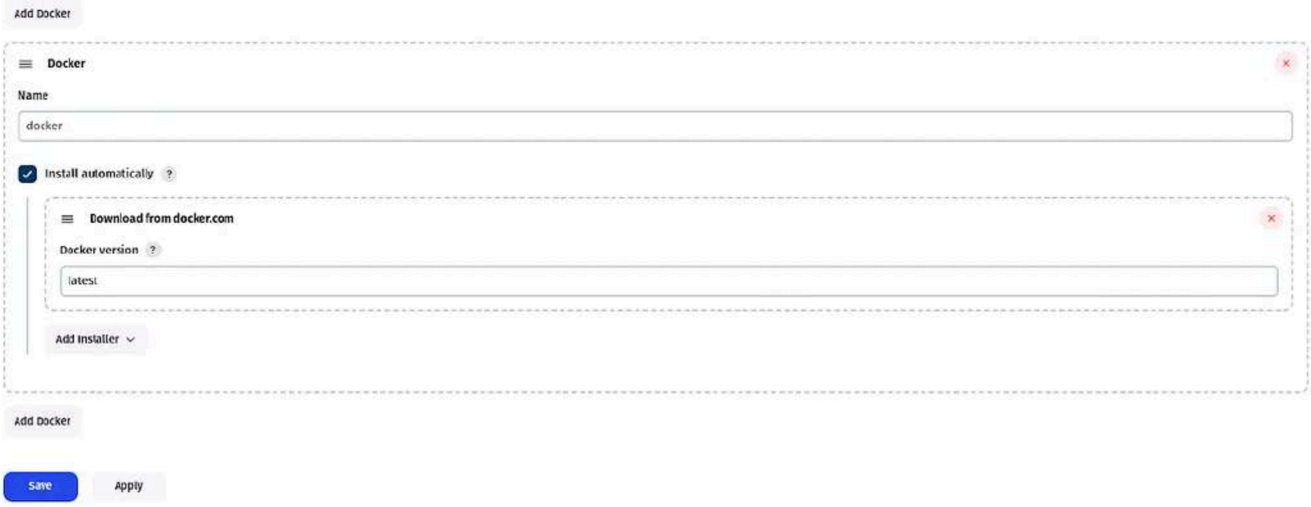
Install from github.com

Version

dependency-check 9.0.9

Now, we will configure the docker

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



Docker

Name: docker

☑ Install automatically

Download from docker.com

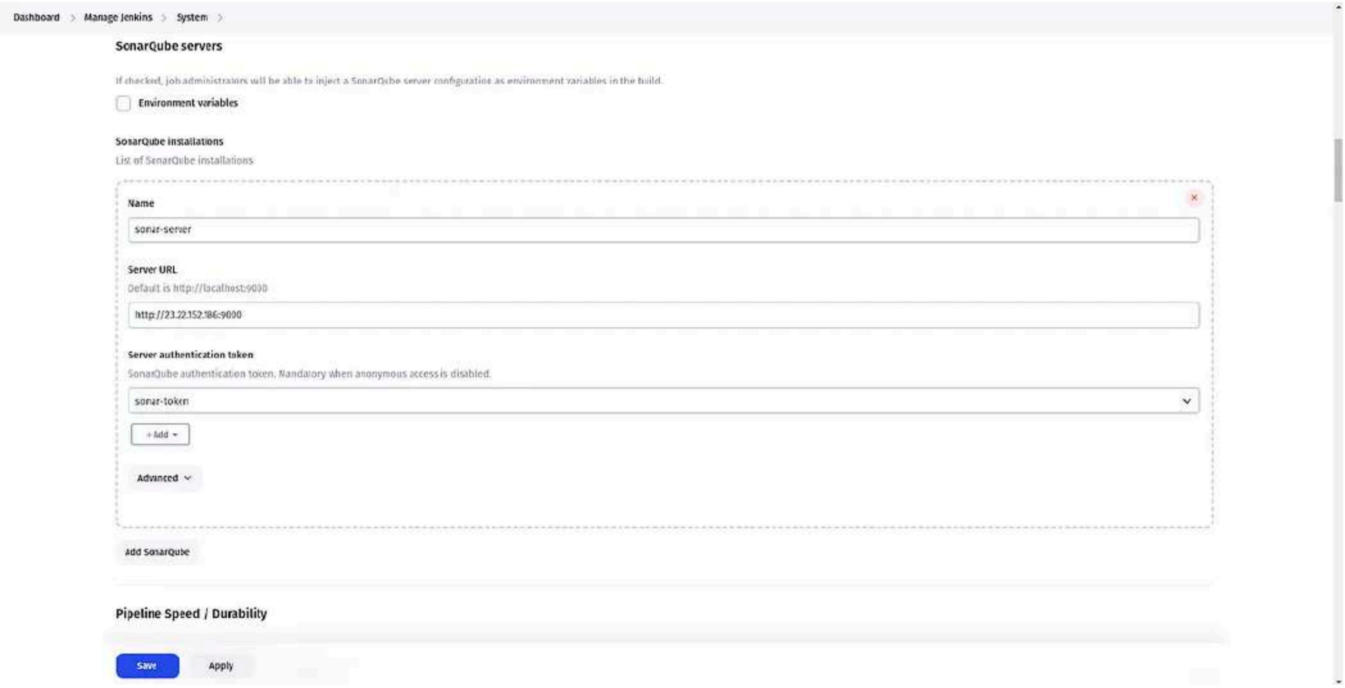
Docker version: latest

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 the Server URL copy the sonarqube public IP (same as Jenkins) with port 9000 select the sonar token that we have added recently, and click on Apply & Save.


Dashboard > Manage Jenkins > System >
+ Add
Advanced
Add SonarQube
Pipeline Speed / Durability
Save
Apply

SonarQube servers

If checked, job administrators will be able to inject a SonarQube server configuration as environment variables in the build.

□ Environment variables

SonarQube installations

List of SonarQube installations

Name: sonar-server

Server URL: http://23.22.152.186:9000

Default is http://localhost:9000

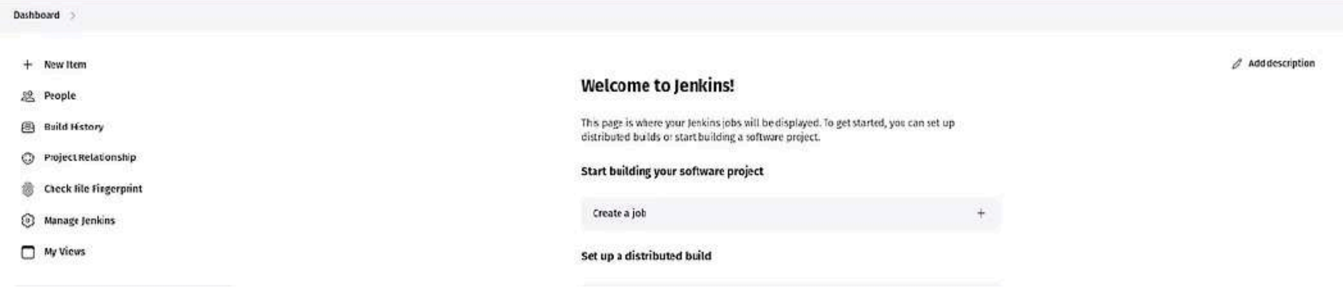
Server authentication token: sonar-token

SonarQube authentication token. Mandatory when anonymous access is disabled.

Now, we are ready to create our Jenkins Pipeline to deploy our Backend Code.

Go to Jenkins **Dashboard**

Click on **New Item**

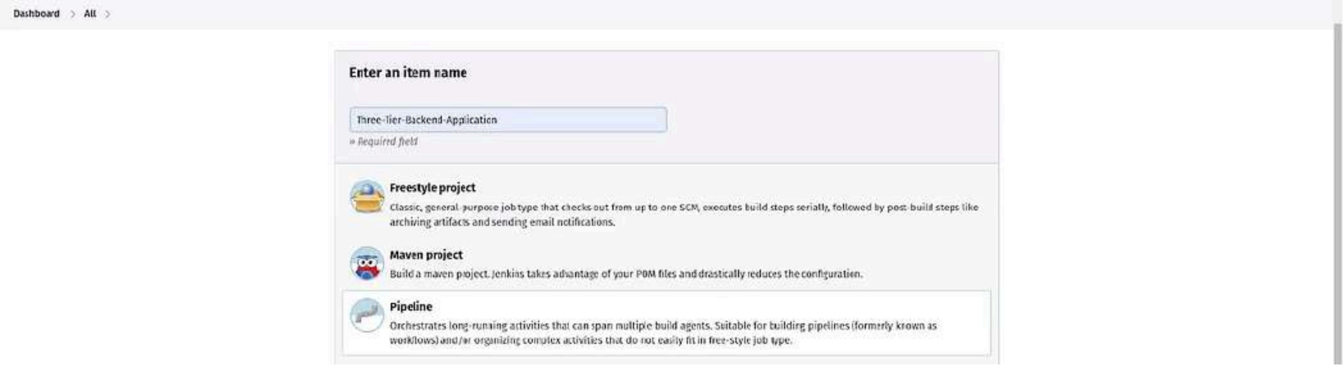


Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

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



Enter an item name

Three-Tier-Backend-Application

» Required field

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

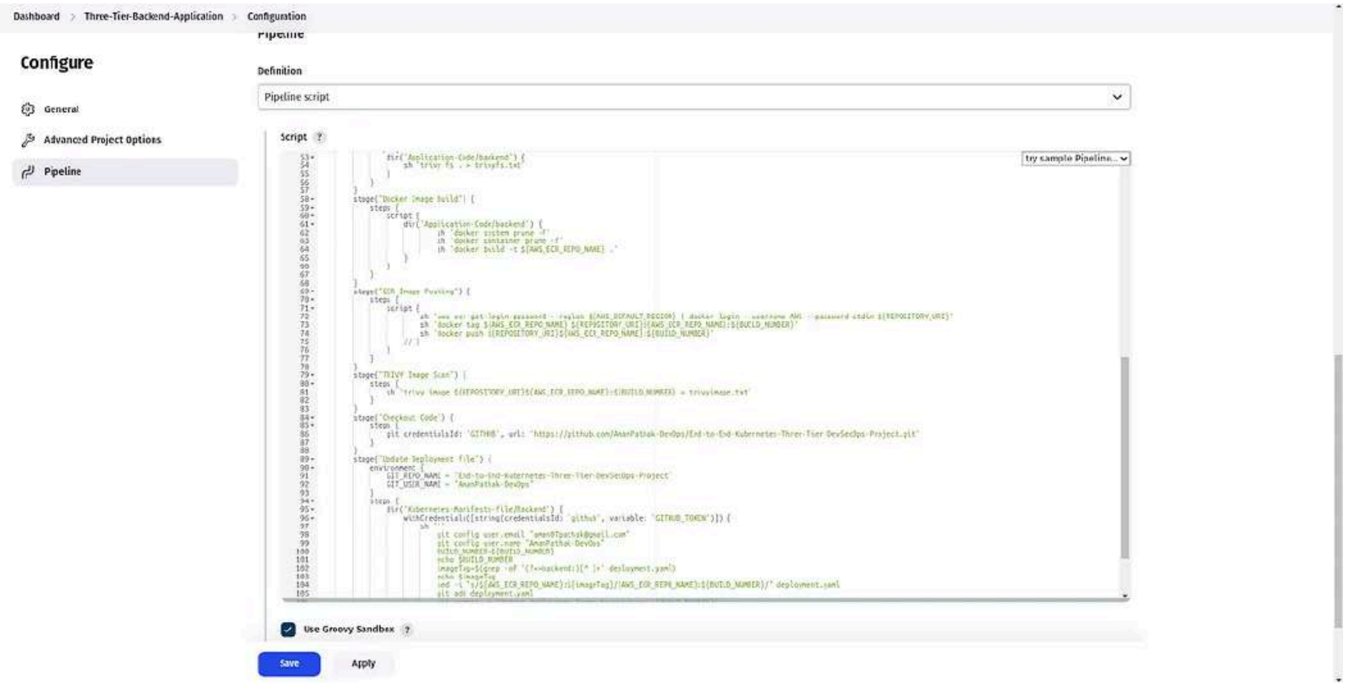
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

This is the Jenkins file to deploy the Backend Code on **EKS.**

Copy and paste it into the **Jenkins**

https://github.com/AmanPathak-DevOps/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project/blob/master/Jenkins-Pipeline-Code/Jenkinsfile-Backend

Click **Apply** & **Save.**



Pipeline

Definition

Pipeline script

Script

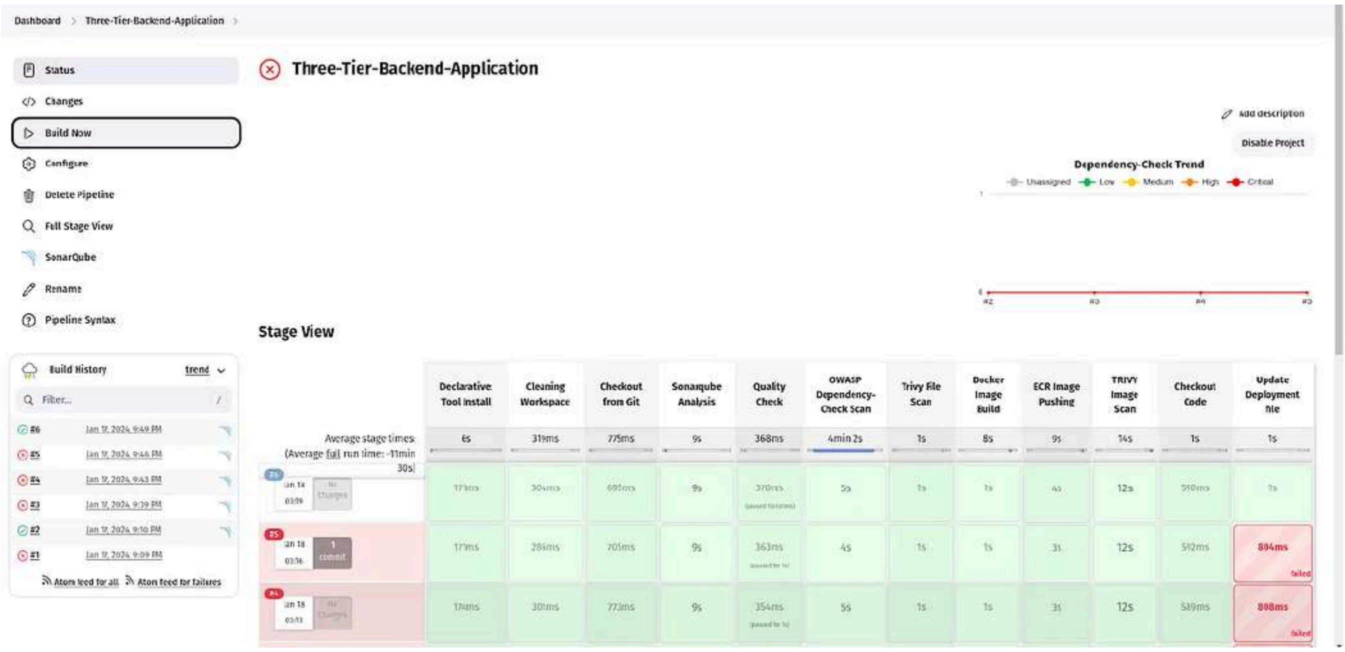
```
            dir('Application-Code/backend') {
                sh 'trivy fs . > trivyfs.txt'
            }
        }
    }
    stage("Docker Image Build") {
        steps {
            script {
                dir('Application-Code/backend') {
                        sh 'docker system prune -f'
                        sh 'docker container prune -f'
                        sh 'docker build -t ${AWS_ECR_REPO_NAME} .'
                }
            }
        }
    }
    stage("ECR Image Pushing") {
        steps {
            script {
                    sh 'aws ecr get-login-password --region ${AWS_DEFAULT_REGION} | docker login --username AWS --password-stdin ${REPOSITORY_URI}'
                    sh 'docker tag ${AWS_ECR_REPO_NAME} ${REPOSITORY_URI}${AWS_ECR_REPO_NAME}:${BUILD_NUMBER}'
                    sh 'docker push ${REPOSITORY_URI}${AWS_ECR_REPO_NAME}:${BUILD_NUMBER}'
                // }
            }
        }
    }
    stage("TRIVY Image Scan") {
        steps {
            sh 'trivy image ${REPOSITORY_URI}${AWS_ECR_REPO_NAME}:${BUILD_NUMBER} > trivyimage.txt'
        }
    }
    stage('Checkout Code') {
        steps {
            git credentialsId: 'GITHUB', url: 'https://github.com/AmanPathak-DevOps/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project.git'
        }
    }
    stage('Update Deployment file') {
        environment {
            GIT_REPO_NAME = "End-to-End-Kubernetes-Three-Tier-DevSecOps-Project"
            GIT_USER_NAME = "AmanPathak-DevOps"
        }
        steps {
            dir('Kubernetes-Manifests-file/Backend') {
                withCredentials([string(credentialsId: 'github', variable: 'GITHUB_TOKEN')]) {
                    sh '''
                        git config user.email "aman07pathak@gmail.com"
                        git config user.name "AmanPathak-DevOps"
                        BUILD_NUMBER=${BUILD_NUMBER}
                        echo $BUILD_NUMBER
                        imageTag=$(grep -oP '(?<=backend:)[^ ]+' deployment.yaml)
                        echo $imageTag
                        sed -i "s/${AWS_ECR_REPO_NAME}:${imageTag}/${AWS_ECR_REPO_NAME}:${BUILD_NUMBER}/" deployment.yaml
                        git add deployment.yaml
```

☑ Use Groovy Sandbox

Now, click on the **build.**

Our **pipeline** was **successful** after a few common mistakes.

Note: Do the changes in the Pipeline according to your project.



Three-Tier-Backend-Application

Build	Date
#6	Jan 17, 2024, 9:49 PM
#5	Jan 17, 2024, 9:46 PM
#4	Jan 17, 2024, 9:43 PM
#3	Jan 17, 2024, 9:39 PM
#2	Jan 17, 2024, 9:10 PM
#1	Jan 17, 2024, 9:09 PM

Stage View

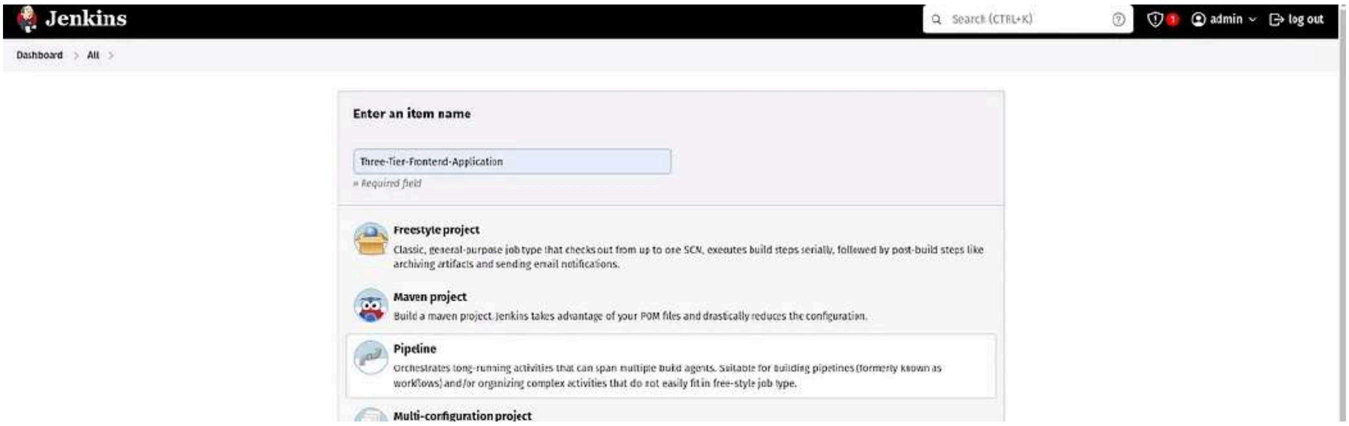
	Declarative Tool Install	Cleaning Workspace	Checkout from Git	Sonarqube Analysis	Quality Check	OWASP Dependency-Check Scan	Trivy File Scan	Docker Image Build	ECR Image Pushing	TRIVY Image Scan	Checkout Code	Update Deployment file
Average stage times: (Average full run time: ~11min 30s)	6s	319ms	775ms	9s	368ms	4min 2s	1s	8s	9s	14s	1s	1s
#6 Jan 18 03:19 No Changes	173ms	304ms	695ms	9s	370ms (paused for ...)	5s	1s	1s	4s	12s	510ms	1s
#5 Jan 18 03:16 1 commit	171ms	286ms	705ms	9s	363ms (paused for 1s)	4s	1s	1s	3s	12s	512ms	804ms failed
#4 Jan 18 03:13 No Changes	174ms	301ms	773ms	9s	354ms (paused for 1s)	5s	1s	1s	3s	12s	519ms	808ms failed

Now, we are ready to create our Jenkins Pipeline to deploy our Frontend Code.

Go to Jenkins **Dashboard**

Click on **New Item**

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



Enter an item name

Three-Tier-Frontend-Application

» Required field

Freestyle project
Classic, general-purpose job type that checks out from up to one SCM, executes build steps serially, followed by post-build steps like archiving artifacts and sending email notifications.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

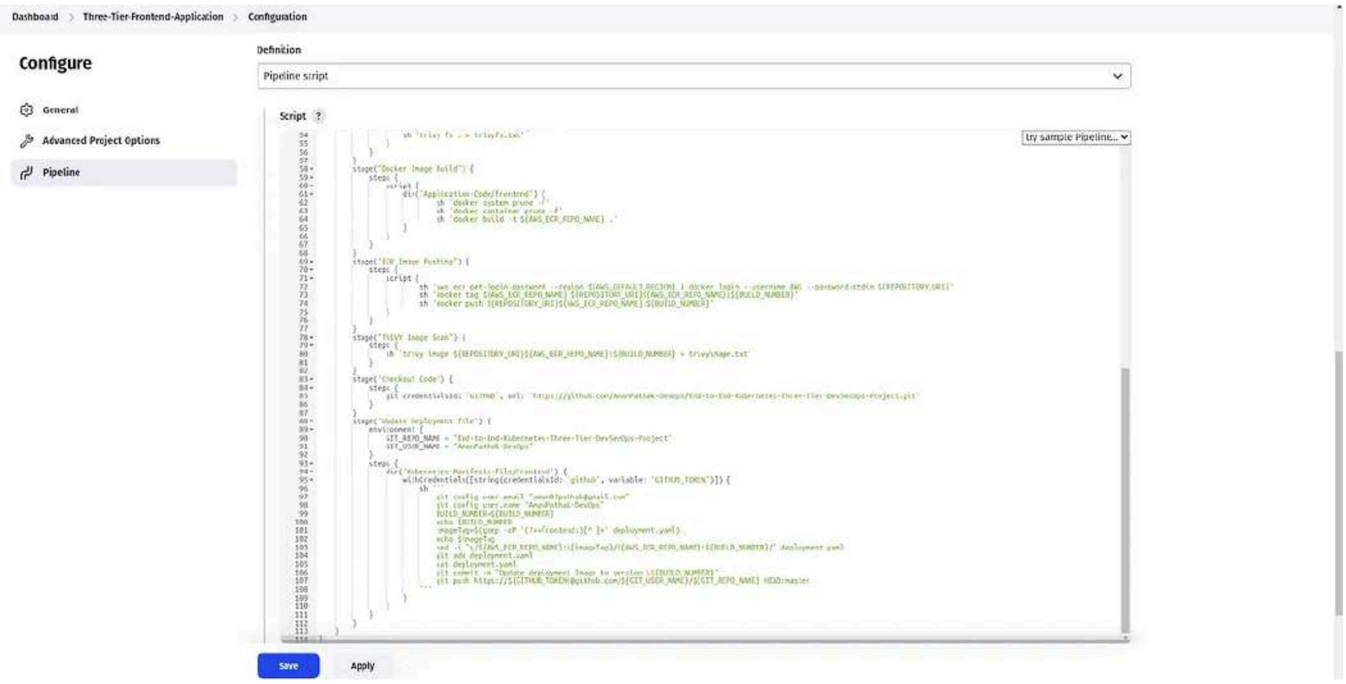
Multi-configuration project

This is the Jenkins file to deploy the Frontend Code on **EKS.**

Copy and paste it into the **Jenkins**

https://github.com/AmanPathak-DevOps/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project/blob/master/Jenkins-Pipeline-Code/Jenkinsfile-Frontend

Click **Apply** & **Save**.

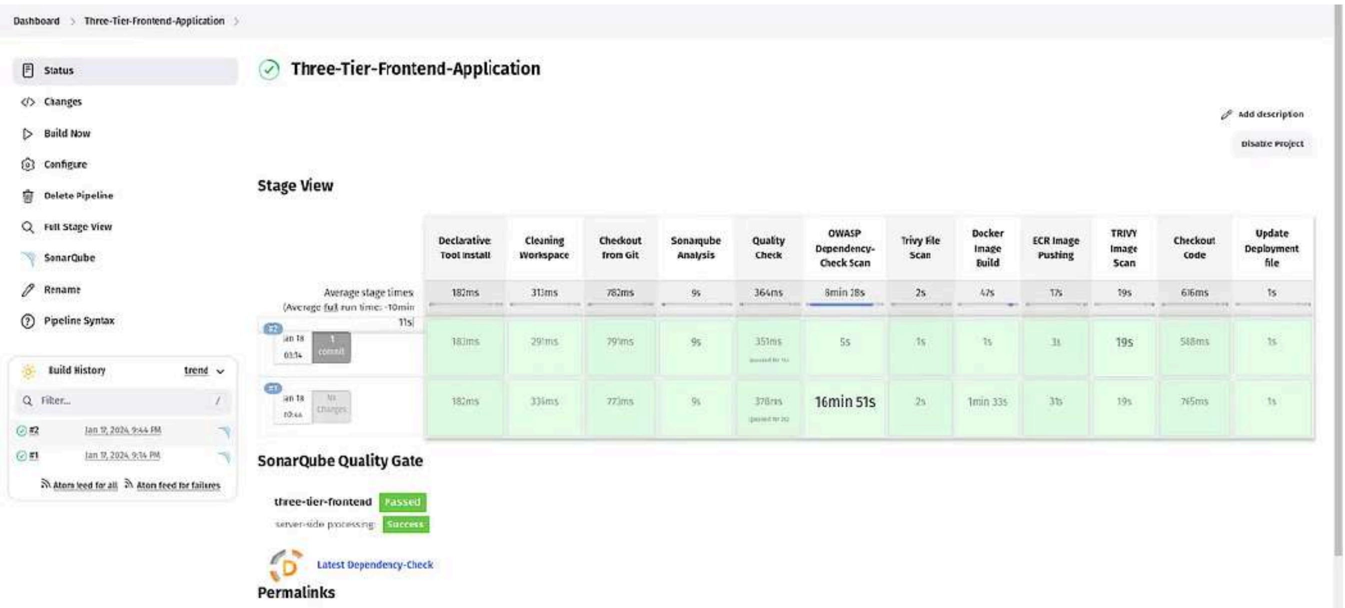


```
                sh 'trivy fs . > trivyfs.txt'
            }
        }
    }
    stage('Docker Image Build') {
        steps {
            script {
                dir('Application-Code/frontend') {
                    sh 'docker system prune -f'
                    sh 'docker container prune -f'
                    sh 'docker build -t ${AWS_ECR_REPO_NAME} .'
                }
            }
        }
    }
    stage('ECR Image Pushing') {
        steps {
            script {
                    sh 'aws ecr get-login-password --region ${AWS_DEFAULT_REGION} | docker login --username AWS --password-stdin ${REPOSITORY_URI}'
                    sh 'docker tag ${AWS_ECR_REPO_NAME} ${REPOSITORY_URI}${AWS_ECR_REPO_NAME}:${BUILD_NUMBER}'
                    sh 'docker push ${REPOSITORY_URI}${AWS_ECR_REPO_NAME}:${BUILD_NUMBER}'
            }
        }
    }
    stage("TRIVY Image Scan") {
        steps {
            sh 'trivy image ${REPOSITORY_URI}${AWS_ECR_REPO_NAME}:${BUILD_NUMBER} > trivyimage.txt'
        }
    }
    stage('Checkout Code') {
        steps {
            git credentialsId: 'GITHUB', url: 'https://github.com/AmanPathak-DevOps/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project.git'
        }
    }
    stage('Update Deployment file') {
        environment {
            GIT_REPO_NAME = "End-to-End-Kubernetes-Three-Tier-DevSecOps-Project"
            GIT_USER_NAME = "AmanPathak-DevOps"
        }
        steps {
            dir('Kubernetes-Manifests-file/Frontend') {
                withCredentials([string(credentialsId: 'github', variable: 'GITHUB_TOKEN')]) {
                    sh '''
                        git config user.email "aman07pathak@gmail.com"
                        git config user.name "AmanPathak-DevOps"
                        BUILD_NUMBER=${BUILD_NUMBER}
                        echo $BUILD_NUMBER
                        imageTag=$(grep -oP '(?<=frontend:)[^ ]+' deployment.yaml)
                        echo $imageTag
                        sed -i "s/${AWS_ECR_REPO_NAME}:${imageTag}/${AWS_ECR_REPO_NAME}:${BUILD_NUMBER}/" deployment.yaml
                        git add deployment.yaml
                        git deployment.yaml
                        git commit -m "Update deployment Image to version \${BUILD_NUMBER}"
                        git push https://${GITHUB_TOKEN}@github.com/${GIT_USER_NAME}/${GIT_REPO_NAME} HEAD:master
                    '''
                }
            }
        }
    }
}
}
```

Now, click on the **build.**

Our **pipeline** was **successful** after a few common mistakes.

Note: Do the changes in the Pipeline according to your project.



	Declarative Tool Install	Cleaning Workspace	Checkout from Git	Sonarqube Analysis	Quality Check	OWASP Dependency-Check Scan	Trivy File Scan	Docker Image Build	ECR Image Pushing	TRIVY Image Scan	Checkout Code	Update Deployment file
Average stage times: (Average full run time: ~10min 11s)	182ms	313ms	782ms	9s	364ms	8min 28s	2s	47s	17s	19s	616ms	1s
#2 Jan 18 03:14 1 commit	183ms	291ms	791ms	9s	351ms	5s	1s	1s	3s	19s	568ms	1s
#1 Jan 18 02:44 No Changes	182ms	336ms	773ms	9s	378ms	16min 51s	2s	1min 33s	31s	19s	765ms	1s

#2 Jan 17, 2024, 9:44 PM
#1 Jan 17, 2024, 9:34 PM

SonarQube Quality Gate

three-tier-frontend Passed
server-side processing: Success

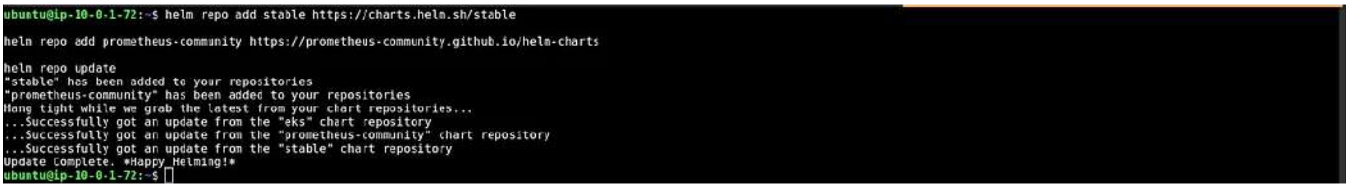
Latest Dependency-Check

Setup 10: We will set up the Monitoring for our EKS Cluster. We can monitor the Cluster Specifications and other necessary things.

We will achieve the monitoring using Helm

Add the prometheus repo by using the below command

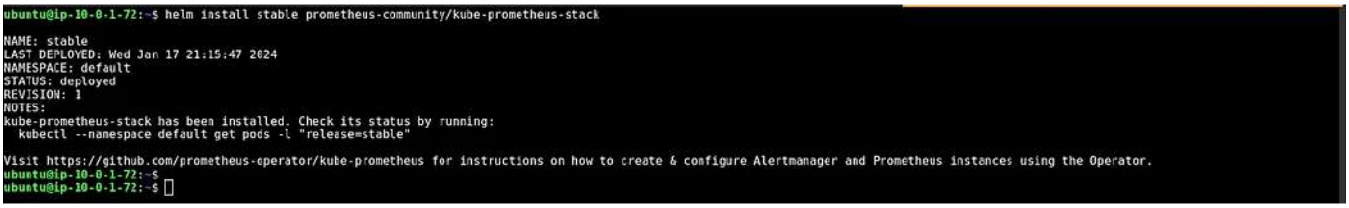
```
helm repo add stable https://charts.helm.sh/stable
```



```
ubuntu@ip-10-0-1-72:~$ helm repo add stable https://charts.helm.sh/stable

helm repo add prometheus-community https://prometheus-community.github.io/helm-charts

helm repo update
"stable" has been added to your repositories
"prometheus-community" has been added to your repositories
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "eks" chart repository
...Successfully got an update from the "prometheus-community" chart repository
...Successfully got an update from the "stable" chart repository
Update Complete. *Happy Helming!*
ubuntu@ip-10-0-1-72:~$ 
```

Install the Prometheus

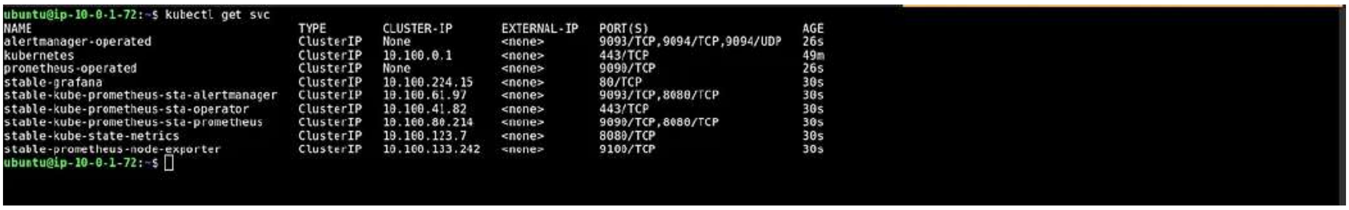
```
helm repo add prometheus-community https://prometheus-community.github.io/helm-
helm install prometheus prometheus-community/prometheus
helm repo add grafana https://grafana.github.io/helm-charts
helm repo update
helm install grafana grafana/grafana
```



```
ubuntu@ip-10-0-1-72:~$ helm install stable prometheus-community/kube-prometheus-stack

NAME: stable
LAST DEPLOYED: Wed Jan 17 21:15:47 2024
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
kube-prometheus-stack has been installed. Check its status by running:
  kubectl --namespace default get pods -l "release=stable"

Visit https://github.com/prometheus-operator/kube-prometheus for instructions on how to create & configure Alertmanager and Prometheus instances using the Operator.
ubuntu@ip-10-0-1-72:~$
ubuntu@ip-10-0-1-72:~$ 
```

Now, check the service by the below command

```
kubectl get svc
```



```
ubuntu@ip-10-0-1-72:~$ kubectl get svc
NAME                                     TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
alertmanager-operated                    ClusterIP   None             <none>        9093/TCP,9094/TCP,9094/UDP   26s
kubernetes                               ClusterIP   10.100.0.1       <none>        443/TCP                      49m
prometheus-operated                      ClusterIP   None             <none>        9090/TCP                     26s
stable-grafana                           ClusterIP   10.100.224.15    <none>        80/TCP                       30s
stable-kube-prometheus-sta-alertmanager  ClusterIP   10.100.61.97     <none>        9093/TCP,8080/TCP            30s
stable-kube-prometheus-sta-operator      ClusterIP   10.100.41.82     <none>        443/TCP                      30s
stable-kube-prometheus-sta-prometheus    ClusterIP   10.100.80.214    <none>        9090/TCP,8080/TCP            30s
stable-kube-state-metrics                ClusterIP   10.100.123.7     <none>        8080/TCP                     30s
stable-prometheus-node-exporter          ClusterIP   10.100.133.242   <none>        9100/TCP                     30s
ubuntu@ip-10-0-1-72:~$ 
```

Now, we need to access our Prometheus and Grafana consoles from outside of the cluster.

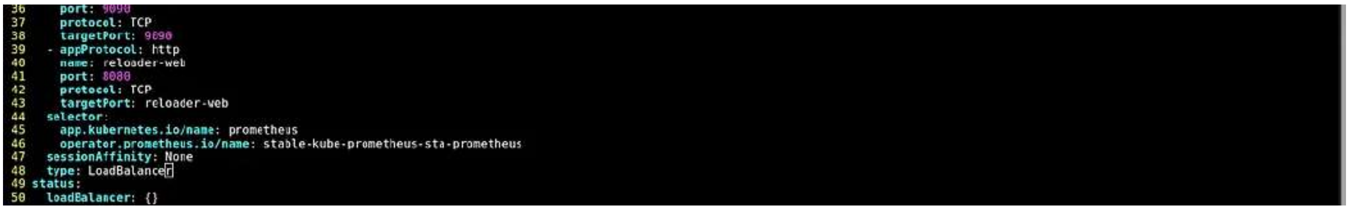
For that, we need to change the Service type from ClusterType to **LoadBalancer**

Edit the **stable-kube-prometheus-sta-prometheus** service

```
kubectl edit svc stable-kube-prometheus-sta-prometheus
```

```
ubuntu@ip-10-0-1-72:~$ kubectl edit svc stable-kube-prometheus-sta-prometheus
```

Modification in the 48th line from ClusterType to **LoadBalancer**



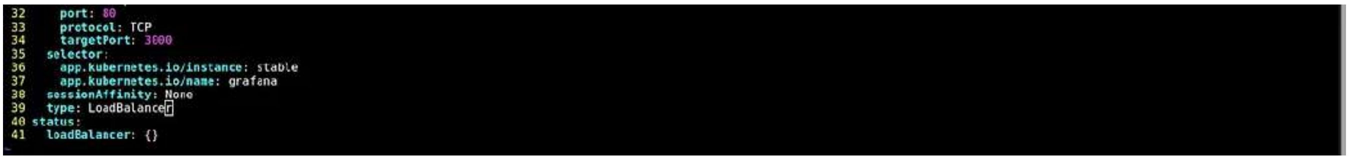
```
36      port: 9090
37      protocol: TCP
38      targetPort: 9090
39    - appProtocol: http
40      name: reloader-web
41      port: 8080
42      protocol: TCP
43      targetPort: reloader-web
44    selector:
45      app.kubernetes.io/name: prometheus
46      operator.prometheus.io/name: stable-kube-prometheus-sta-prometheus
47    sessionAffinity: None
48    type: LoadBalancer
49  status:
50    loadBalancer: {}
```

Edit the **stable-grafana** service

```
kubectl edit svc stable-grafana
```

```
ubuntu@ip-10-0-1-72:~$ kubectl edit svc stable-grafana
```

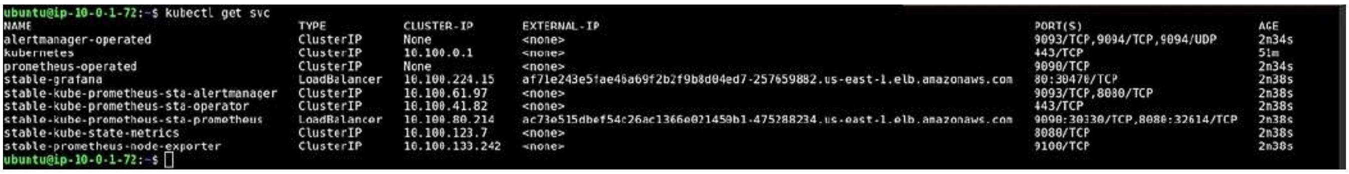
Modification in the 39th line from ClusterType to **LoadBalancer**



```
32      port: 80
33      protocol: TCP
34      targetPort: 3000
35    selector:
36      app.kubernetes.io/instance: stable
37      app.kubernetes.io/name: grafana
38    sessionAffinity: None
39    type: LoadBalancer
40  status:
41    loadBalancer: {}
```

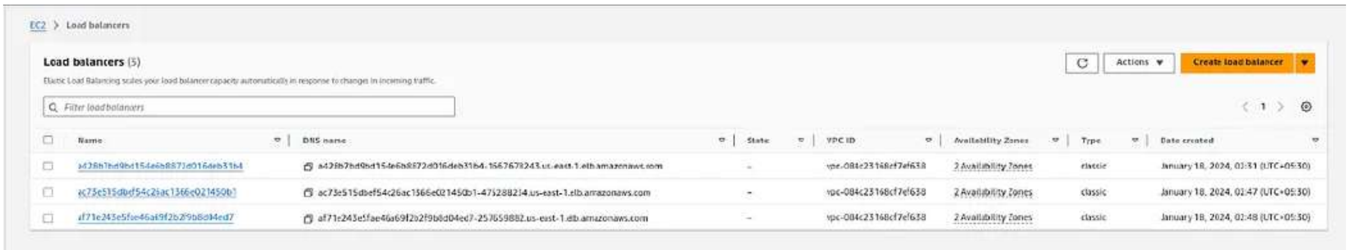
Now, if you list again the service then, you will see the LoadBalancers DNS names

```
kubectl get svc
```



```
ubuntu@ip-10-0-1-72:~$ kubectl get svc
NAME                                     TYPE           CLUSTER-IP       EXTERNAL-IP                                                                   PORT(S)                          AGE
alertmanager-operated                    ClusterIP      None             <none>                                                                        9093/TCP,9094/TCP,9094/UDP       2m34s
kubernetes                               ClusterIP      10.100.0.1       <none>                                                                        443/TCP                          51m
prometheus-operated                      ClusterIP      None             <none>                                                                        9090/TCP                         2m34s
stable-grafana                           LoadBalancer   10.100.224.15    af71e243e5fae46a69f2b2f9b8d04ed7-257659882.us-east-1.elb.amazonaws.com       80:30476/TCP                     2m38s
stable-kube-prometheus-sta-alertmanager  ClusterIP      10.100.61.97     <none>                                                                        9093/TCP,8080/TCP                2m38s
stable-kube-prometheus-sta-operator      ClusterIP      10.100.41.82     <none>                                                                        443/TCP                          2m38s
stable-kube-prometheus-sta-prometheus    LoadBalancer   10.100.80.214    ac73e515dbef54c26ac1366e021450b1-475288234.us-east-1.elb.amazonaws.com       9090:30330/TCP,8080:32614/TCP    2m38s
stable-kube-state-metrics                ClusterIP      10.100.123.7     <none>                                                                        8080/TCP                         2m38s
stable-prometheus-node-exporter          ClusterIP      10.100.133.242   <none>                                                                        9100/TCP                         2m38s
ubuntu@ip-10-0-1-72:~$
```

You can also validate from your console.



EC2 > Load balancers

Load balancers (3)

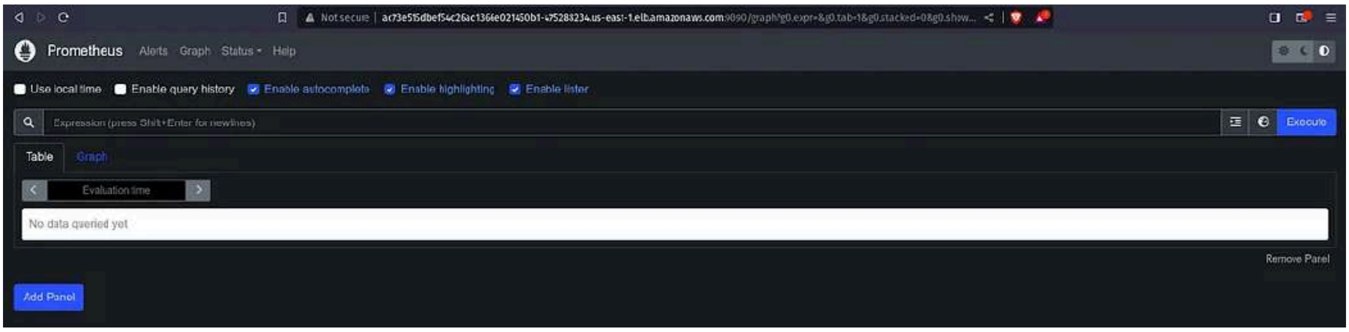
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Filter load balancers

□	Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
□	a428b7bd9bd154e6b8872d016deb31b4	a428b7bd9bd154e6b8872d016deb31b4-1567678243.us-east-1.elb.amazonaws.com	-	vpc-084c23168cf7ef638	2 Availability Zones	classic	January 18, 2024, 01:31 (UTC+05:30)
□	ac73e515dbef54c26ac1366e021450b1	ac73e515dbef54c26ac1366e021450b1-475288234.us-east-1.elb.amazonaws.com	-	vpc-084c23168cf7ef638	2 Availability Zones	classic	January 18, 2024, 01:47 (UTC+05:30)
□	af71e243e5fae46a69f2b2f9b8d04ed7	af71e243e5fae46a69f2b2f9b8d04ed7-257659882.us-east-1.elb.amazonaws.com	-	vpc-084c23168cf7ef638	2 Availability Zones	classic	January 18, 2024, 02:48 (UTC+05:30)

Now, access your Prometheus Dashboard

Paste the <Prometheus-LB-DNS>:9090 in your favorite browser and you will see like this



Not secure | ac73e515dbef54c26ac1366e021450b1-475288234.us-east-1.elb.amazonaws.com:9090/graph?g0.expr=&g0.tab=1&g0.stacked=0&g0.show...

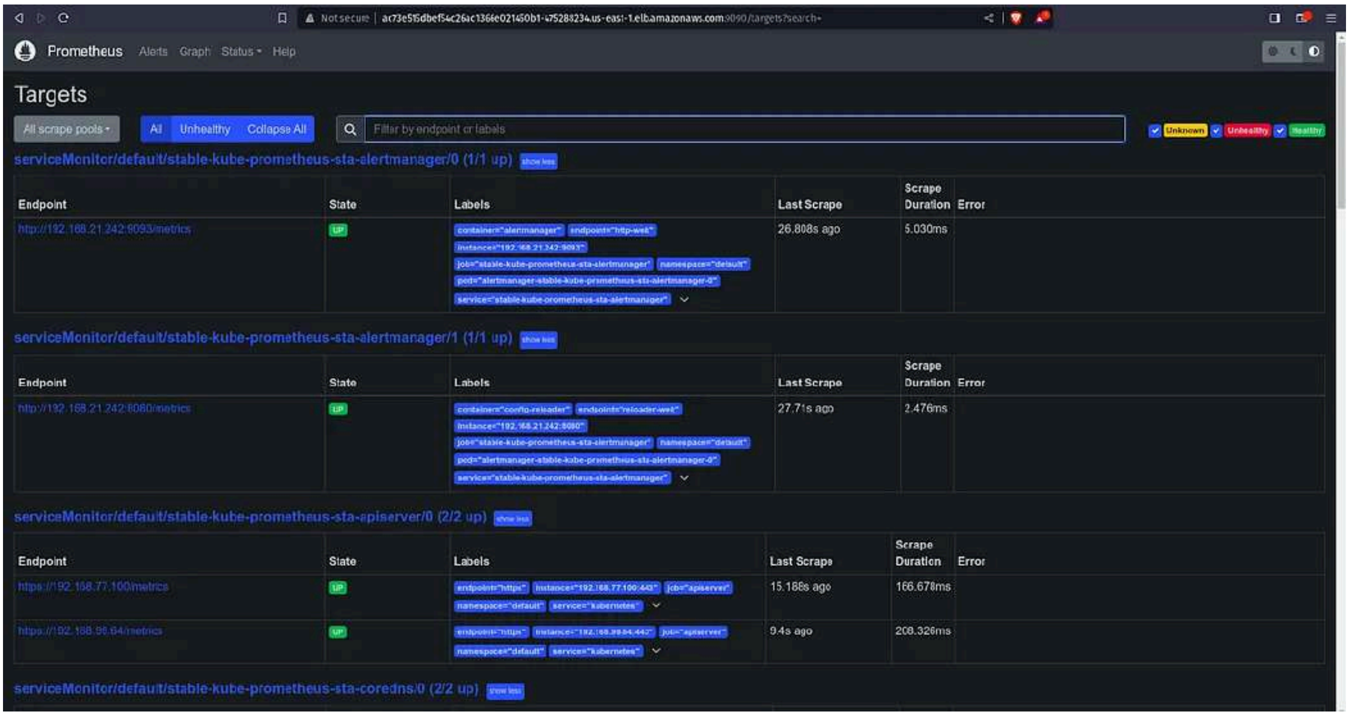
□ Use local time □ Enable query history ☑ Enable autocomplete ☑ Enable highlighting ☑ Enable linter

Expression (press Shift+Enter for newlines)

No data queried yet

Click on **Status** and select **Target.**

You will see a lot of Targets


Not secure | ac73e515dbef54c26ac1366e021450b1-475288234.us-east-1.elb.amazonaws.com:9090/targets?search=
Prometheus
Alerts
Graph
Status
Help
All scrape pools
All
Unhealthy
Collapse All
Filter by endpoint or labels
Unknown
Unhealthy
Healthy
show less

Targets

serviceMonitor/default/stable-kube-prometheus-sta-alertmanager/0 (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.21.242:9093/metrics	UP	container="alertmanager" endpoint="http-web" instance="192.168.21.242:9093" job="stable-kube-prometheus-sta-alertmanager" namespace="default" pod="alertmanager-stable-kube-prometheus-sta-alertmanager-0" service="stable-kube-prometheus-sta-alertmanager"	26.808s ago	5.030ms	

serviceMonitor/default/stable-kube-prometheus-sta-alertmanager/1 (1/1 up)

Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
http://192.168.21.242:8080/metrics	UP	container="config-reloader" endpoint="reloader-web" instance="192.168.21.242:8080" job="stable-kube-prometheus-sta-alertmanager" namespace="default" pod="alertmanager-stable-kube-prometheus-sta-alertmanager-0" service="stable-kube-prometheus-sta-alertmanager"	27.71s ago	2.476ms	

serviceMonitor/default/stable-kube-prometheus-sta-apiserver/0 (2/2 up)

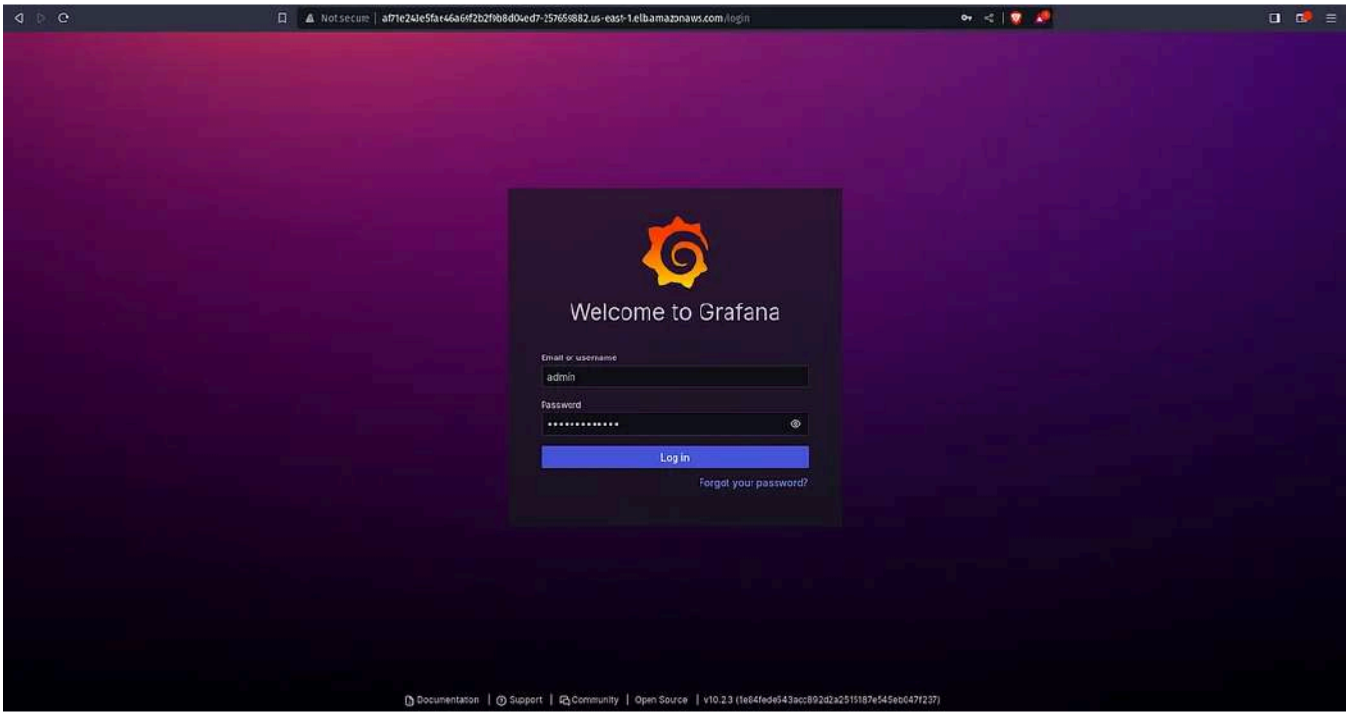
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error
https://192.168.77.100/metrics	UP	endpoint="https" instance="192.168.77.100:443" job="apiserver" namespace="default" service="kubernetes"	15.188s ago	166.678ms	
https://192.168.98.64/metrics	UP	endpoint="https" instance="192.168.98.64:443" job="apiserver" namespace="default" service="kubernetes"	9.4s ago	208.326ms	

serviceMonitor/default/stable-kube-prometheus-sta-coredns/0 (2/2 up)

Now, access your **Grafana Dashboard**

Copy the ALB DNS of Grafana and paste it into your favorite browser.

The username will be **admin** and the password will be **prom-operator** for your Grafana LogIn.

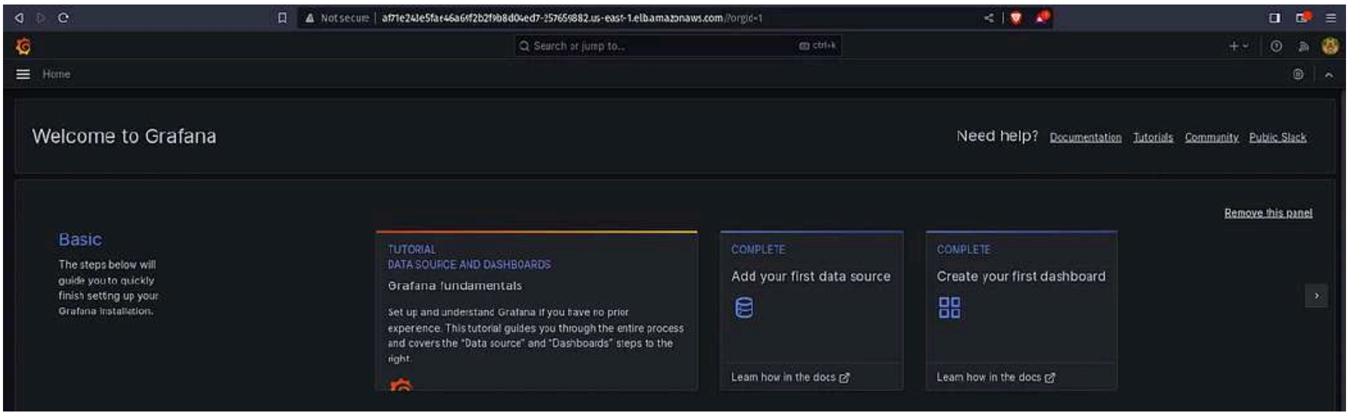

Not secure | af7e24e5fae46a6f2b2f9b8d04ed7-257659882.us-east-1.elb.amazonaws.com/login
Log in
Forgot your password?
Documentation | Support | Community | Open Source | v10.2.3 (1e84fede543acc892d2a2515187e545eb047f237)

Welcome to Grafana

Email or username: admin

Password: •••••••••••••

Now, click on **Data Source**


Not secure | af71e24e5fa46a6f2b2f9b8d04ed7-257659882.us-east-1.elb.amazonaws.com/?orgId=1
Search or jump to...
ctrl+k
Home
Need help? Documentation Tutorials Community Public Slack
Remove this panel
TUTORIAL
DATA SOURCE AND DASHBOARDS
COMPLETE
COMPLETE
Learn how in the docs
Learn how in the docs

Welcome to Grafana

Basic

The steps below will guide you to quickly finish setting up your Grafana installation.

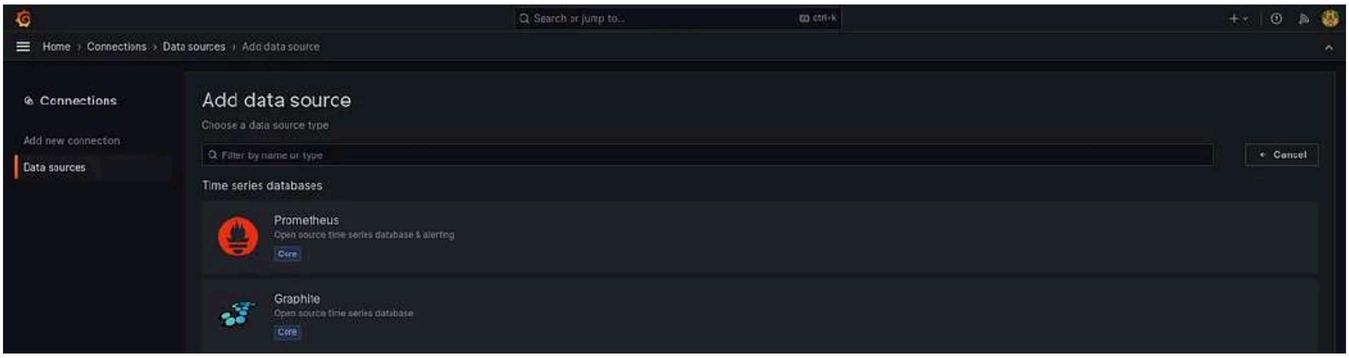
Grafana fundamentals

Set up and understand Grafana if you have no prior experience. This tutorial guides you through the entire process and covers the "Data source" and "Dashboards" steps to the right.

Add your first data source

Create your first dashboard

Select **Prometheus**



Add data source

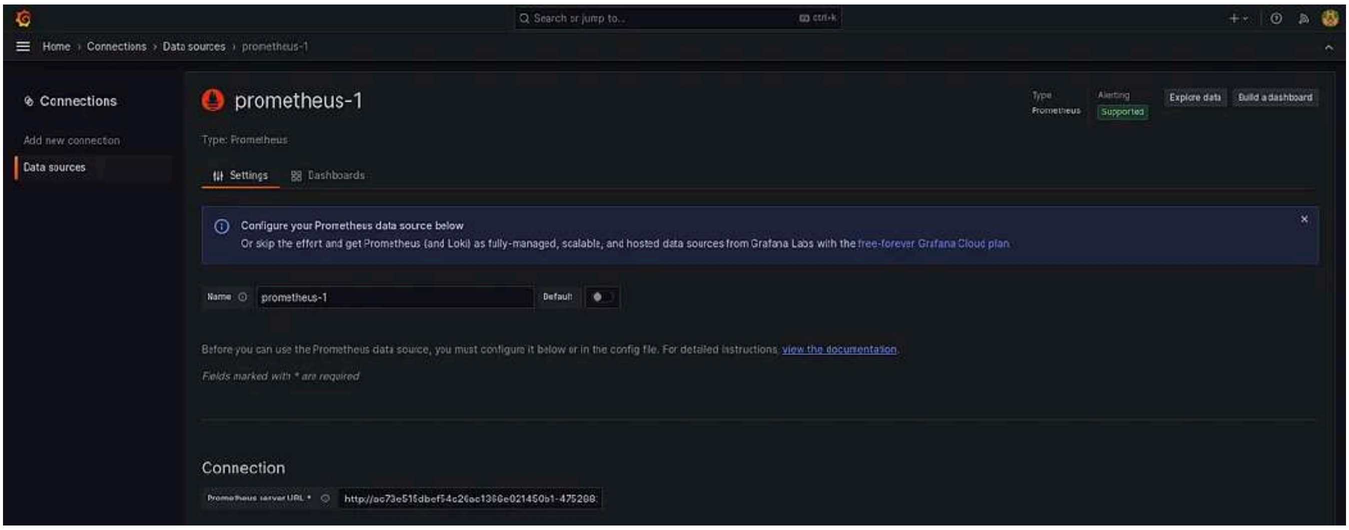
Choose a data source type

Time series databases

Prometheus
Open source time series database & alerting

Graphite
Open source time series database

In the **Connection,** paste your <Prometheus-LB-DNS>:9090.



prometheus-1

Type: Prometheus

Type: Prometheus

Alerting: Supported

Configure your Prometheus data source below
Or skip the effort and get Prometheus (and Loki) as fully-managed, scalable, and hosted data sources from Grafana Labs with the free-forever Grafana Cloud plan.

Name: prometheus-1 Default

Before you can use the Prometheus data source, you must configure it below or in the config file. For detailed instructions, view the documentation.

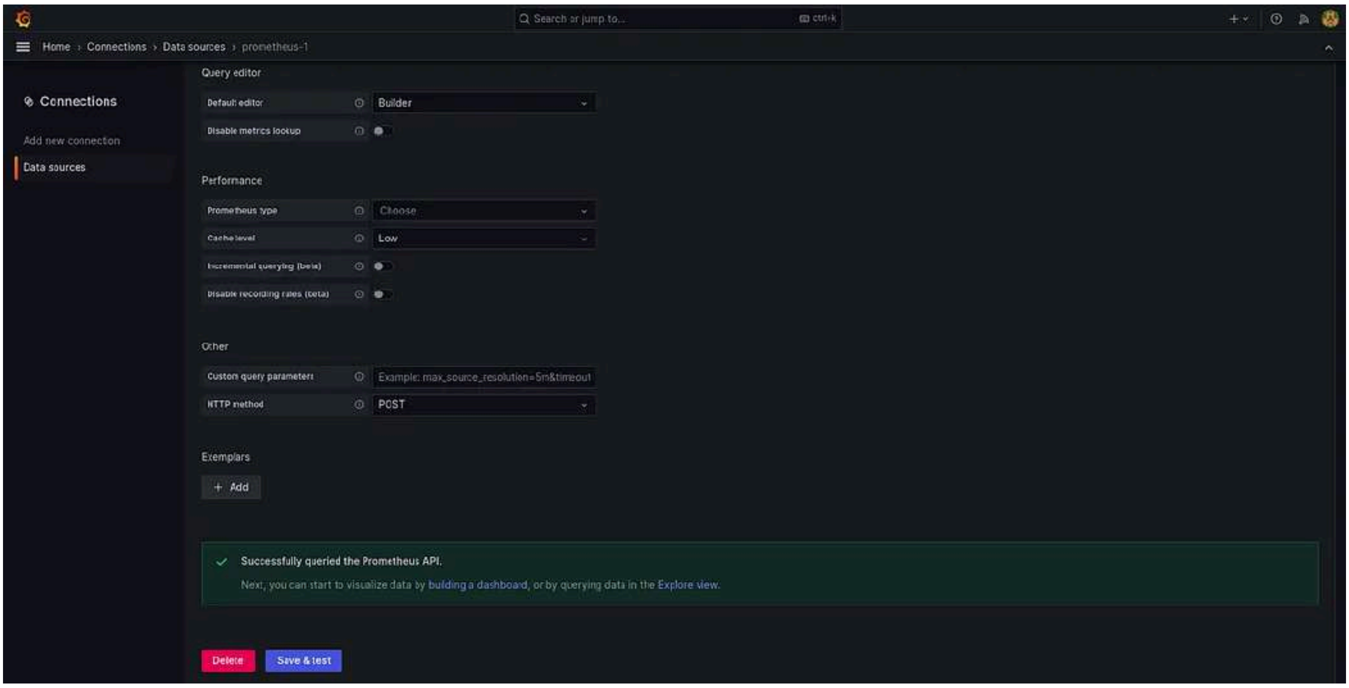
*Fields marked with * are required*

Connection

Prometheus server URL *: http://ac73e516dbef54c26ac1366e021450b1-475208

If the URL is correct, then you will see a green notification/

Click on **Save** & **test.**



Query editor

- Default editor: Builder
- Disable metrics lookup: ●

Performance

- Prometheus type: Choose
- Cache level: Low
- Incremental querying (beta): ●
- Disable recording rules (beta): ●

Other

- Custom query parameters: Example: max_source_resolution=5m&timeout
- HTTP method: POST

Exemplars

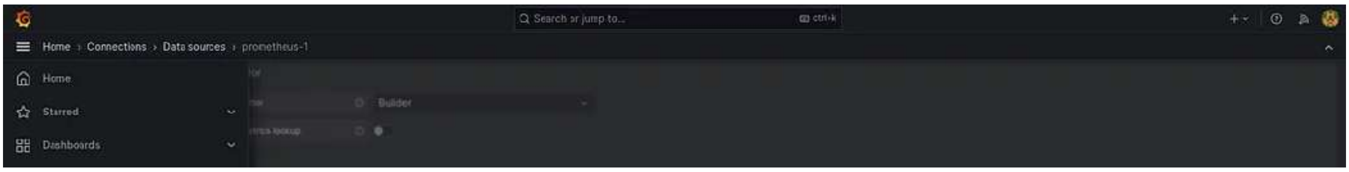
+ Add

✓ Successfully queried the Prometheus API.

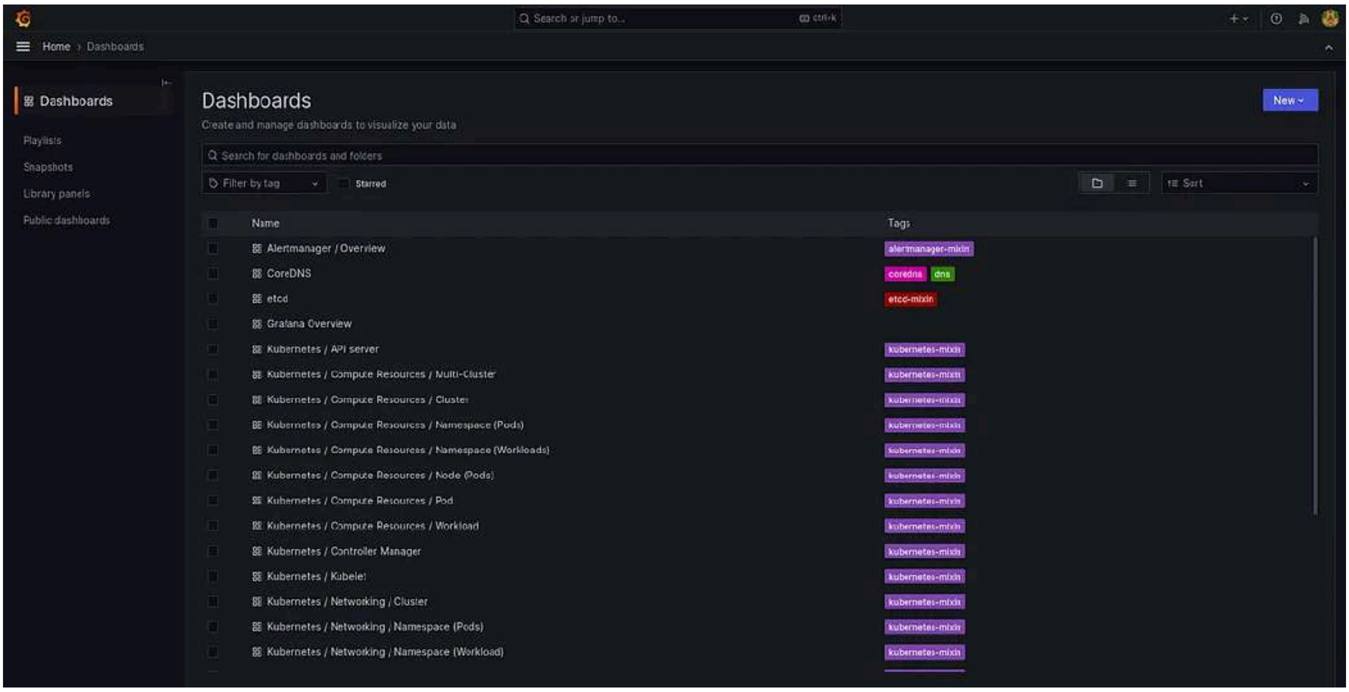
Next, you can start to visualize data by building a dashboard, or by querying data in the Explore view.

Now, we will create a dashboard to visualize our Kubernetes Cluster Logs.

Click on **Dashboard.**



Once you click on **Dashboard.** You will see a lot of Kubernetes components monitoring.



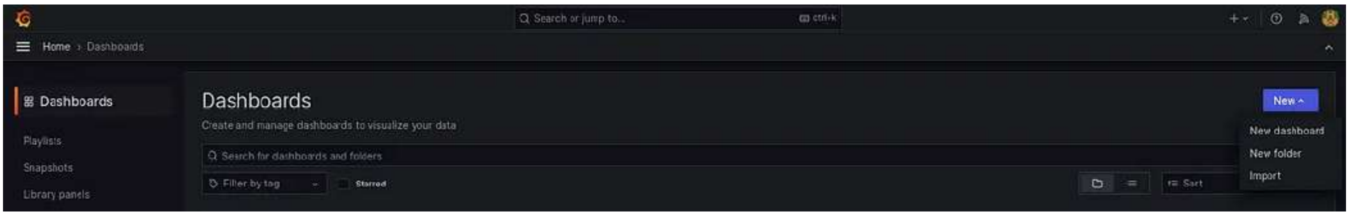
Dashboards

Create and manage dashboards to visualize your data

Name	Tags
Alertmanager / Overview	alertmanager-mixin
CoreDNS	coredns dns
etcd	etcd-mixin
Grafana Overview	
Kubernetes / API server	kubernetes-mixin
Kubernetes / Compute Resources / Multi-Cluster	kubernetes-mixin
Kubernetes / Compute Resources / Cluster	kubernetes-mixin
Kubernetes / Compute Resources / Namespace (Pods)	kubernetes-mixin
Kubernetes / Compute Resources / Namespace (Workloads)	kubernetes-mixin
Kubernetes / Compute Resources / Node (Pods)	kubernetes-mixin
Kubernetes / Compute Resources / Pod	kubernetes-mixin
Kubernetes / Compute Resources / Workload	kubernetes-mixin
Kubernetes / Controller Manager	kubernetes-mixin
Kubernetes / Kubelet	kubernetes-mixin
Kubernetes / Networking / Cluster	kubernetes-mixin
Kubernetes / Networking / Namespace (Pods)	kubernetes-mixin
Kubernetes / Networking / Namespace (Workload)	kubernetes-mixin

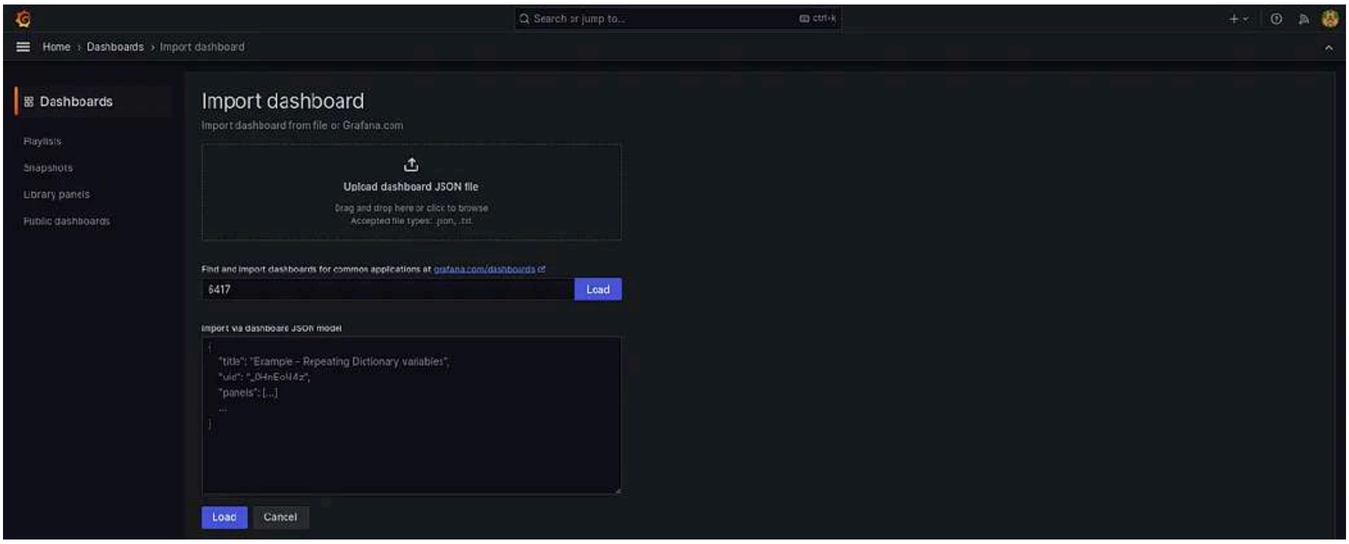
Let’s try to import a type of Kubernetes Dashboard.

Click on **New** and select **Import**



Provide **6417** ID and click on **Load**

Note: 6417 is a unique ID from Grafana which is used to Monitor and visualize Kubernetes Data



Import dashboard

Import dashboard from file or Grafana.com

Upload dashboard JSON file

Drag and drop here or click to browse

Accepted file types: .json, .txt

Find and import dashboards for common applications at grafana.com/dashboards

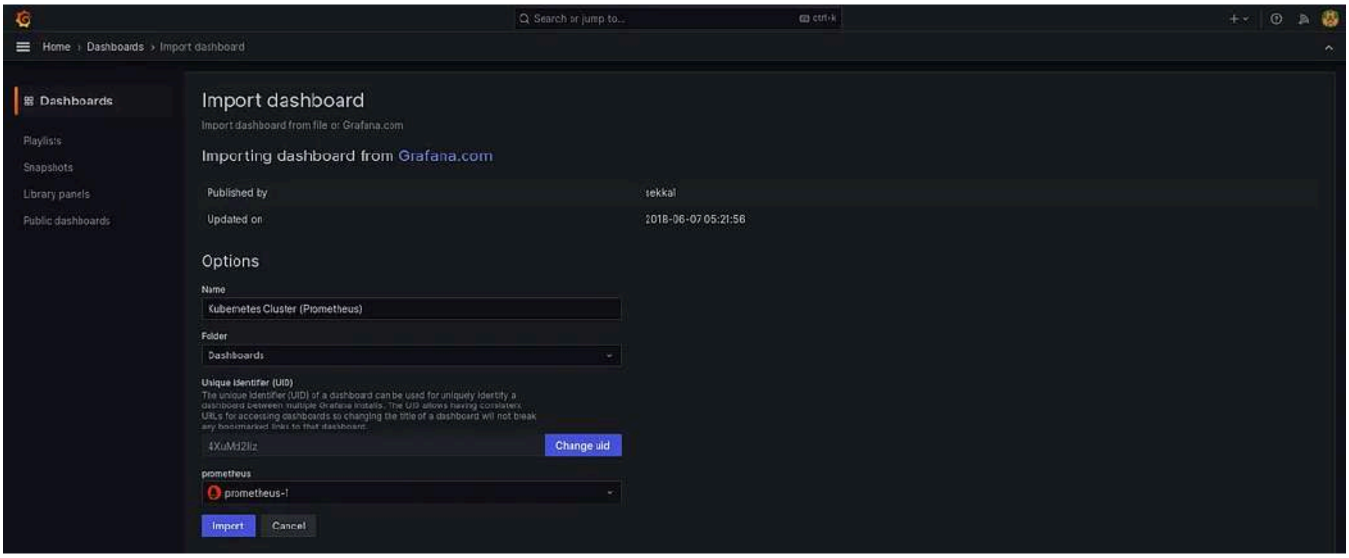
6417

Load

Import via dashboard JSON model

```
{
  "title": "Example - Repeating Dictionary variables",
  "uid": "_0HnEoN4z",
  "panels": [...]
  ...
}
```

Select the **data source** that you have created earlier and click on **Import.**



Import dashboard

Import dashboard from file or Grafana.com

Importing dashboard from Grafana.com

Published by	sekkal
Updated on	2018-06-07 05:21:56

Options

Name: Kubernetes Cluster (Prometheus)

Folder: Dashboards

Unique Identifier (UID)

The unique identifier (UID) of a dashboard can be used for uniquely identify a dashboard between multiple Grafana installs. The UID allows having consistent URLs for accessing dashboards so changing the title of a dashboard will not break any bookmarked links to that dashboard.

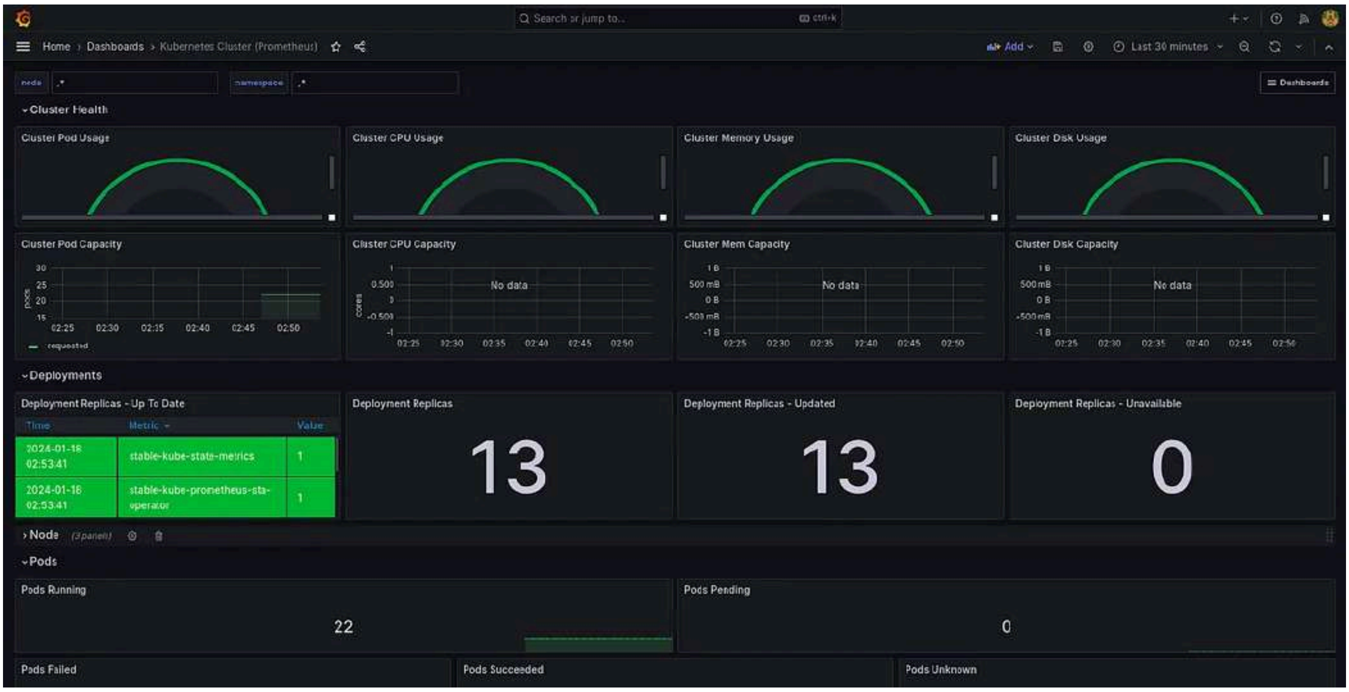
4XuMd2Iiz

prometheus: prometheus-1

Here, you go.

You can view your Kubernetes Cluster Data.

Feel free to explore the other details of the Kubernetes Cluster.

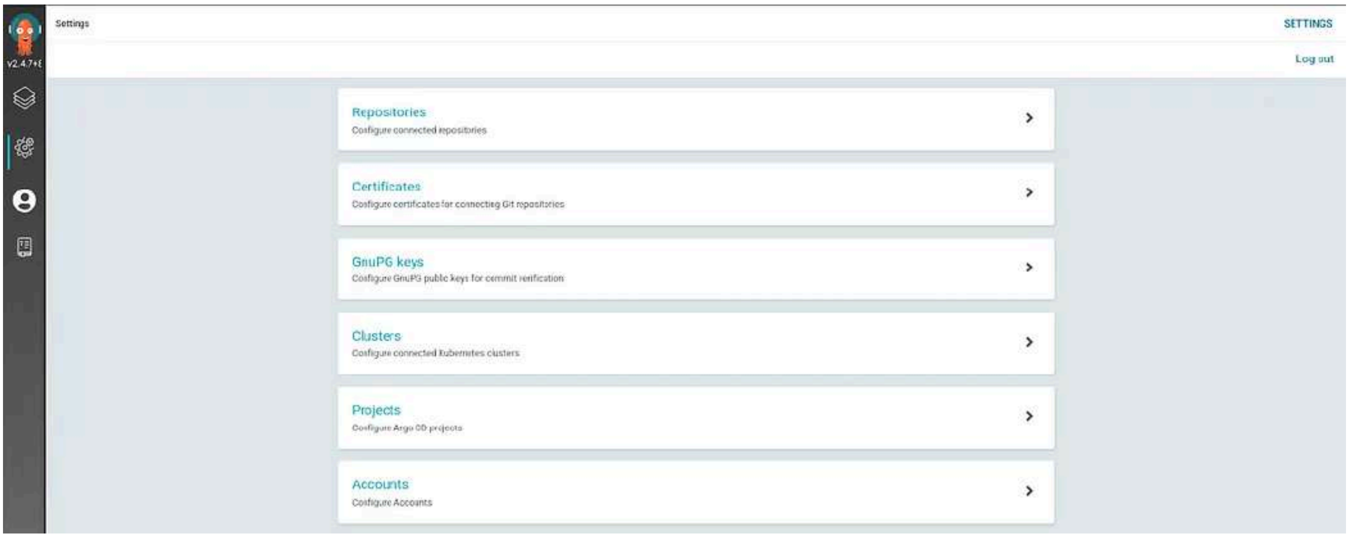


Time	Metric	Value
2024-01-18 02:53:41	stable-kube-state-metrics	1
2024-01-18 02:53:41	stable-kube-prometheus-sta-operator	1

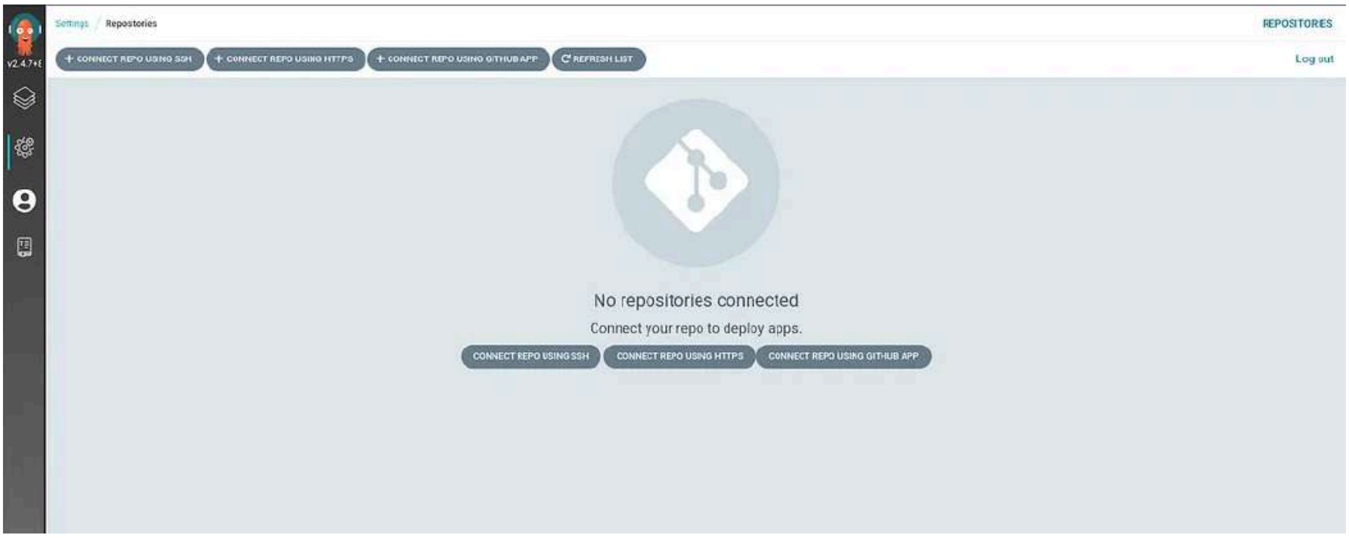
Step 11: We will deploy our Three-Tier Application using ArgoCD.

As our repository is private. So, we need to configure the Private Repository in ArgoCD.

Click on **Settings** and select **Repositories**


Settings
SETTINGS
Log out
v2.4.7+
Repositories
Configure connected repositories
Certificates
Configure certificates for connecting Git repositories
GnuPG keys
Configure GnuPG public keys for commit verification
Clusters
Configure connected Kubernetes clusters
Projects
Configure Argo CD projects
Accounts
Configure Accounts

Click on **CONNECT REPO USING HTTPS**

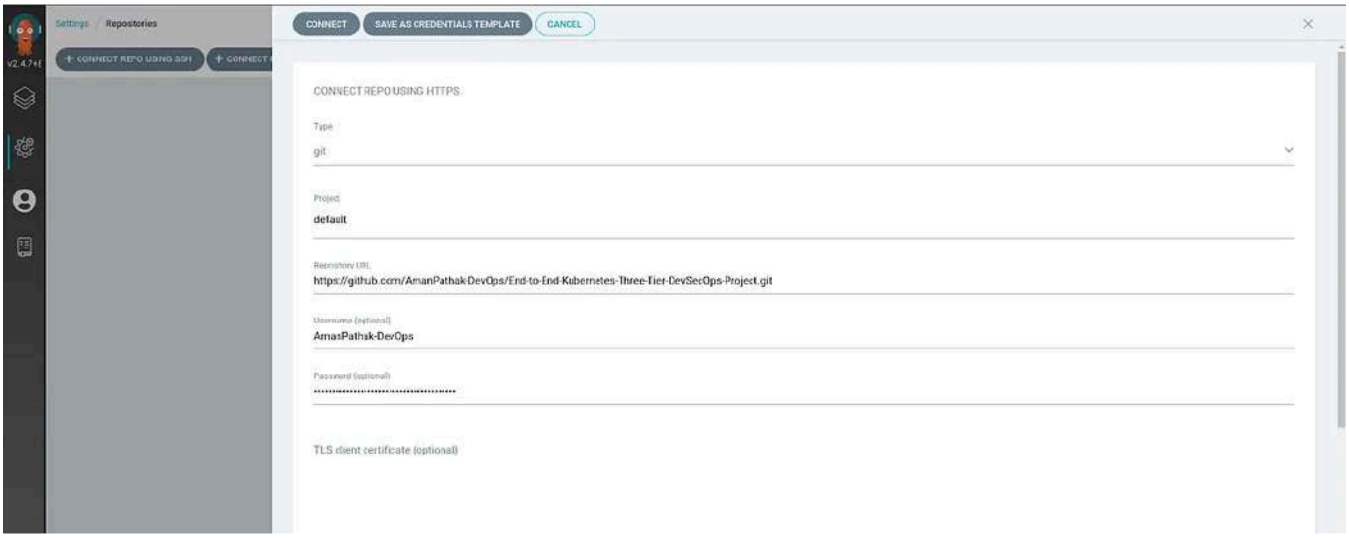


No repositories connected

Connect your repo to deploy apps.

Now, provide the repository name where your Manifests files are present.

Provide the username and GitHub Personal Access token and click on **CONNECT.**



CONNECT REPO USING HTTPS

Type
git

Project
default

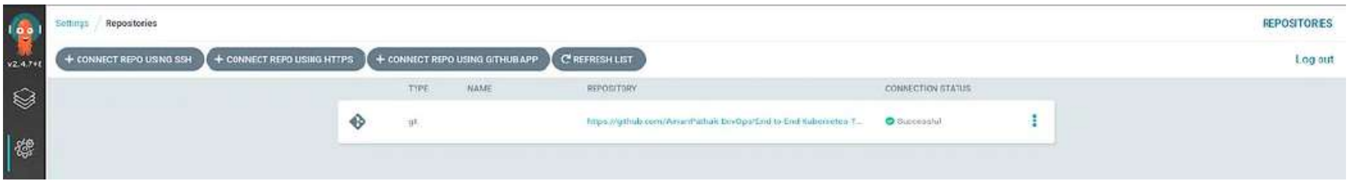
Repository URL
https://github.com/AmanPathak-DevOps/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project.git

Username (optional)
AmasPathak-DevOps

Password (optional)
••

TLS client certificate (optional)

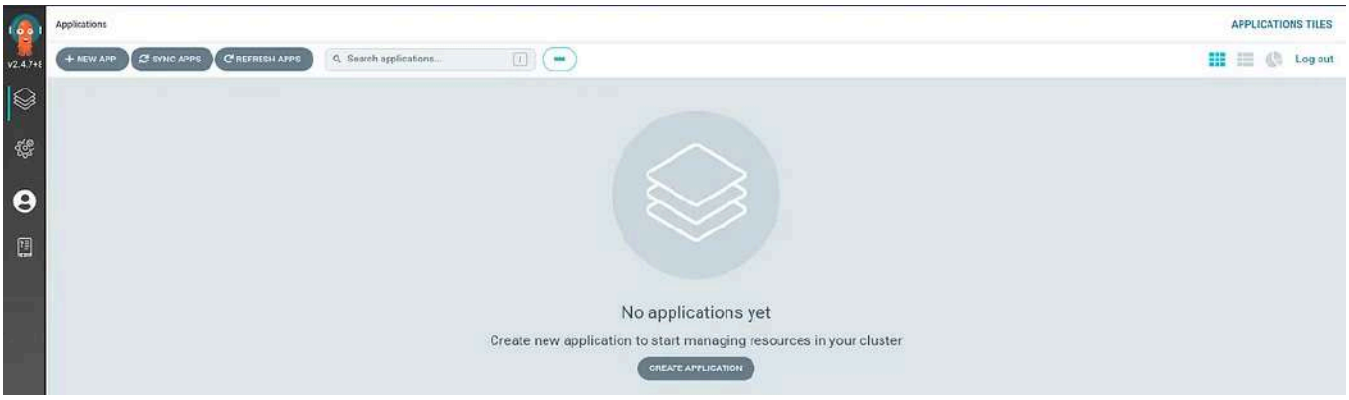
If your **Connection Status** is **Successful** it means repository connected successfully.



TYPE	NAME	REPOSITORY	CONNECTION STATUS
git		https://github.com/AmanPathak-DevOps/End-to-End-Kubernetes-T...	Successful

Now, we will create our first application which will be a database.

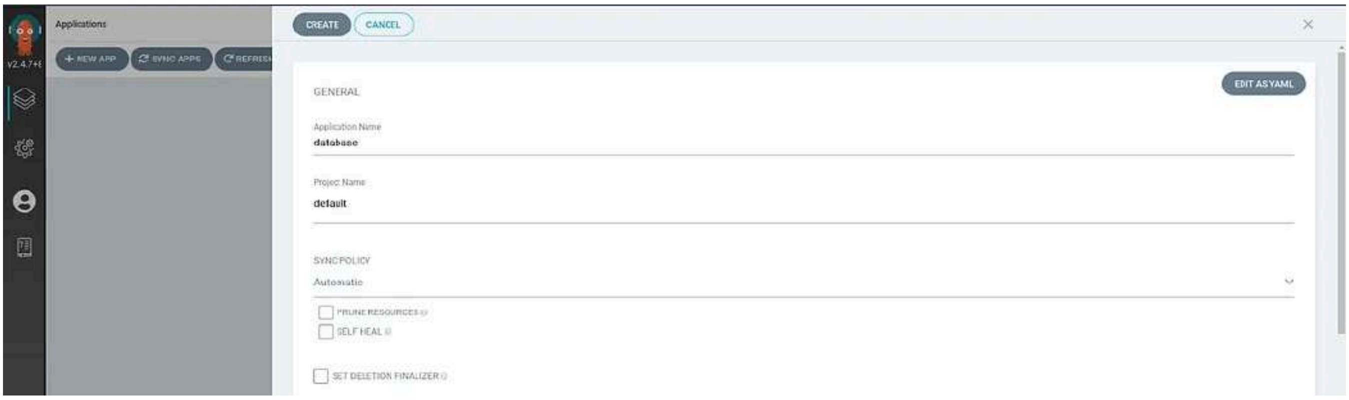
Click on **CREATE APPLICATION.**



No applications yet

Create new application to start managing resources in your cluster

Provide the details as it is provided in the below snippet and scroll down.



GENERAL

Application Name
database

Project Name
default

SYNC POLICY
Automatic

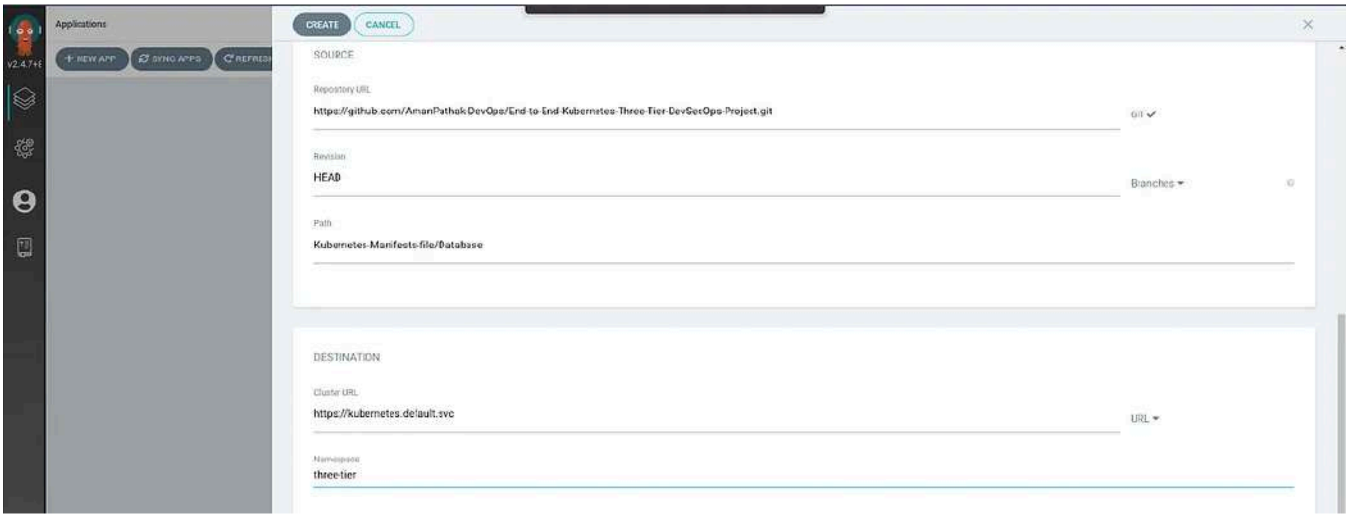
□ PRUNE RESOURCES
□ SELF HEAL

□ SET DELETION FINALIZER

Select the same repository that you configured in the earlier step.

In the **Path**, provide the location where your Manifest files are presented and provide other things as shown in the below screenshot.

Click on **CREATE.**



SOURCE

Repository URL
https://github.com/AmanPathak-DevOps/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project.git

Revision
HEAD

Path
Kubernetes-Manifests-file/Database

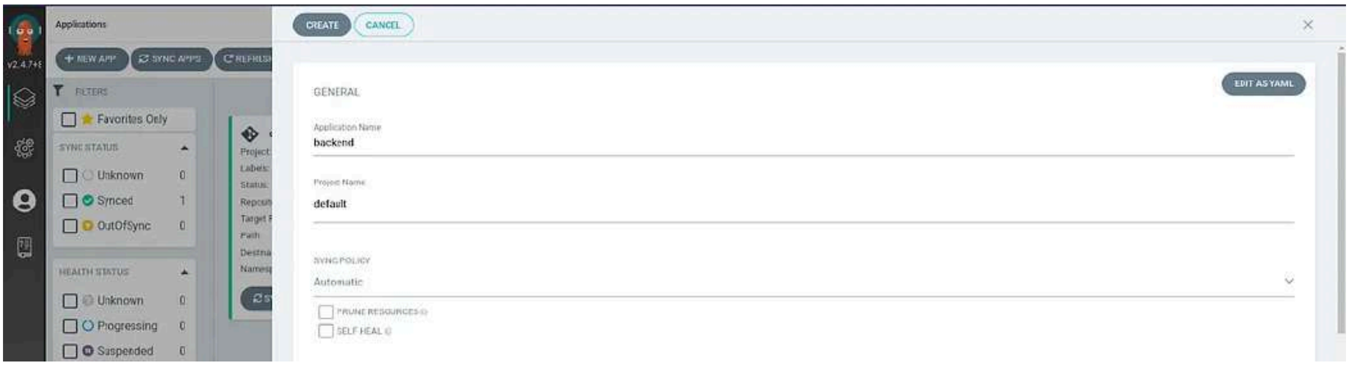
DESTINATION

Cluster URL
https://kubernetes.default.svc

Namespace
three-tier

While your database Application is starting to deploy, We will create an application for the backend.

Provide the details as it is provided in the below snippet and scroll down.



GENERAL

Application Name
backend

Project Name
default

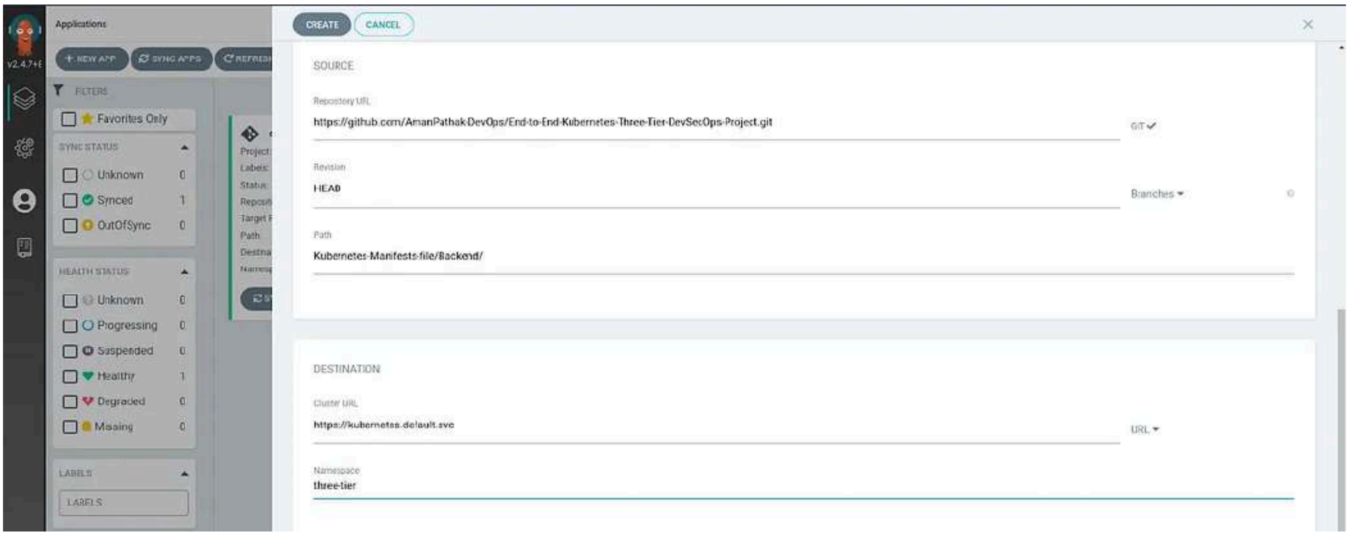
SYNC POLICY
Automatic

□ PRUNE RESOURCES
□ SELF HEAL

Select the same repository that you configured in the earlier step.

In the **Path**, provide the location where your Manifest files are presented and provide other things as shown in the below screenshot.

Click on **CREATE.**



SOURCE

Repository URL
https://github.com/AmanPathak-DevOps/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project.git

Revision
HEAD

Path
Kubernetes-Manifests-file/Backend/

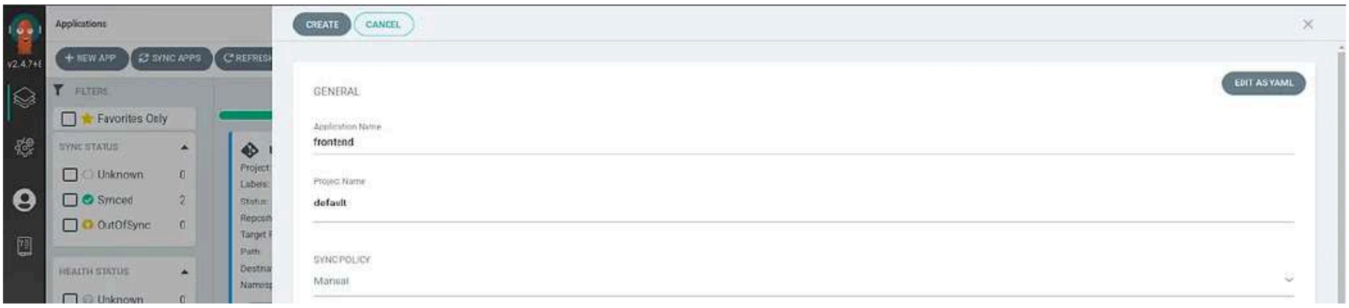
DESTINATION

Cluster URL
https://kubernetes.default.svc

Namespace
three-tier

While your backend Application is starting to deploy, We will create an application for the frontend.

Provide the details as it is provided in the below snippet and scroll down.



GENERAL

Application Name
frontend

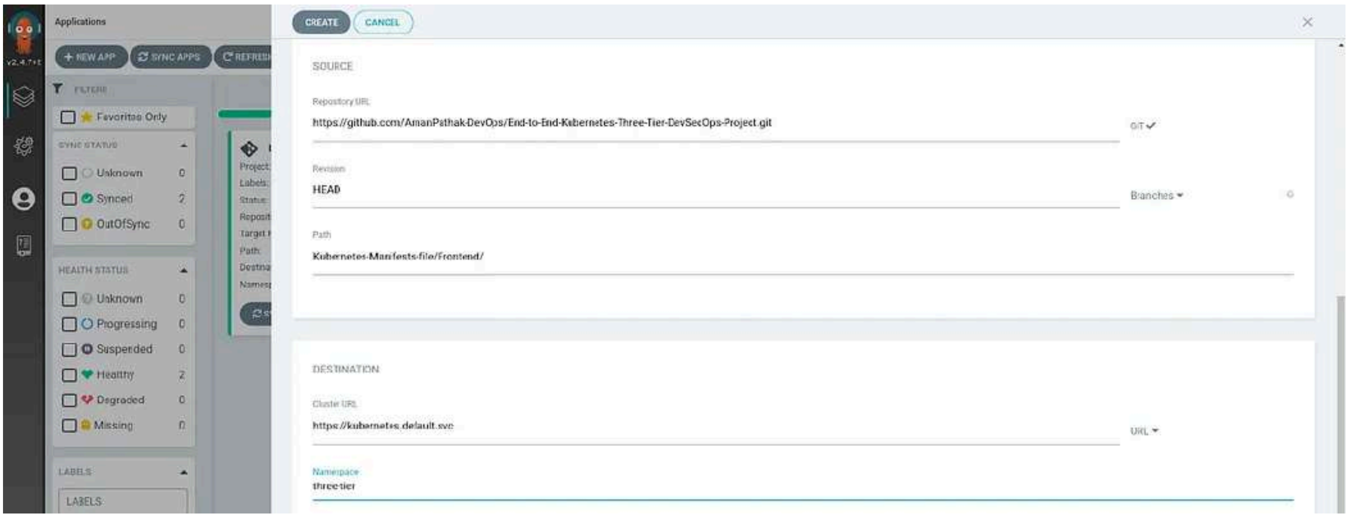
Project Name
default

SYNC POLICY
Manual

Select the same repository that you configured in the earlier step.

In the **Path**, provide the location where your Manifest files are presented and provide other things as shown in the below screenshot.

Click on **CREATE.**



SOURCE

Repository URL: https://github.com/AmanPathak-DevOps/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project.git

Revision: HEAD

Path: Kubernetes-Manifests-file/Frontend/

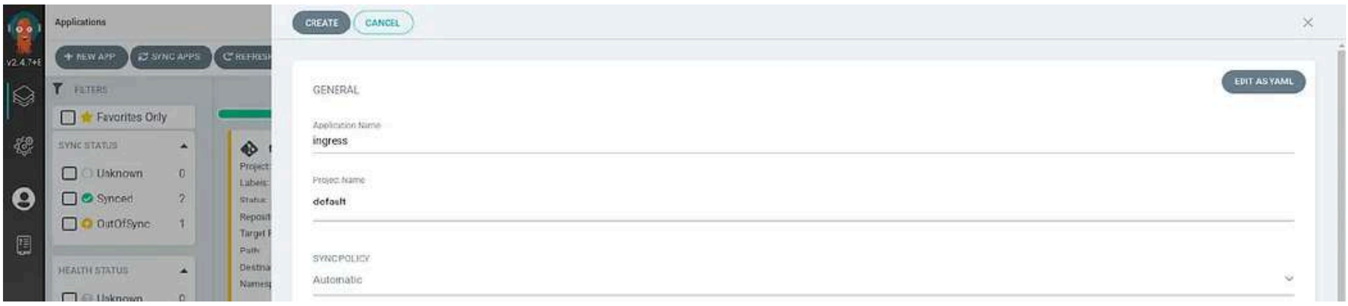
DESTINATION

Cluster URL: https://kubernetes.default.svc

Namespace: three-tier

While your frontend Application is starting to deploy, We will create an application for the ingress.

Provide the details as it is provided in the below snippet and scroll down.



GENERAL

Application Name: ingress

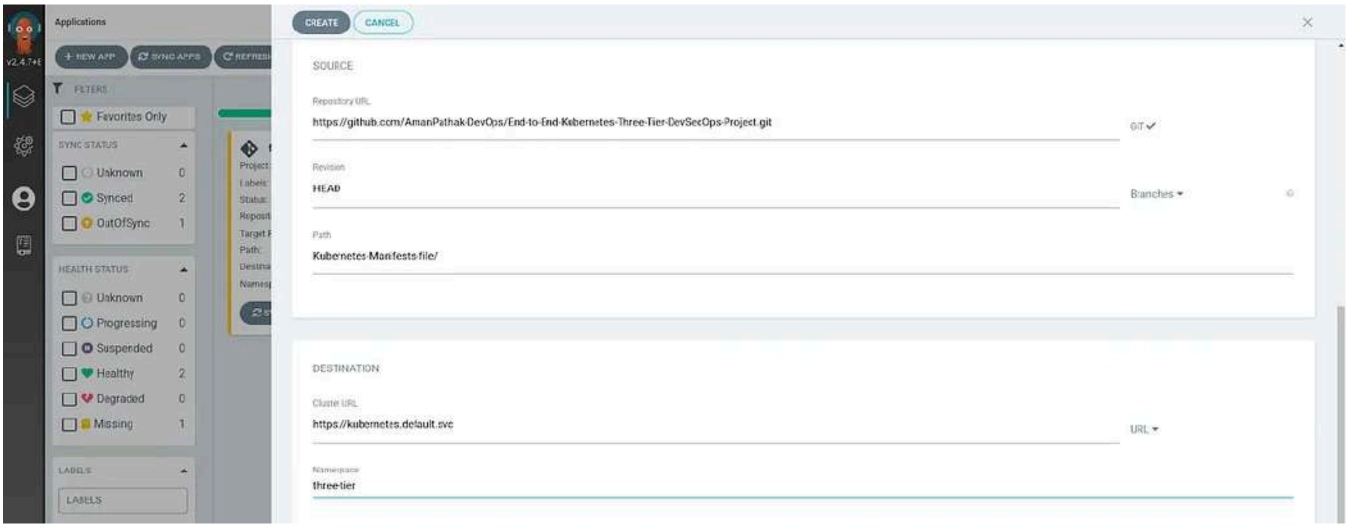
Project Name: default

SYNC POLICY: Automatic

Select the same repository that you configured in the earlier step.

In the **Path**, provide the location where your Manifest files are presented and provide other things as shown in the below screenshot.

Click on **CREATE.**



SOURCE

Repository URL
https://github.com/AmanPathak-DevOps/End-to-End-Kubernetes-Three-Tier-DevSecOps-Project.git

Revision
HEAD

Path
Kubernetes-Manifests-file/

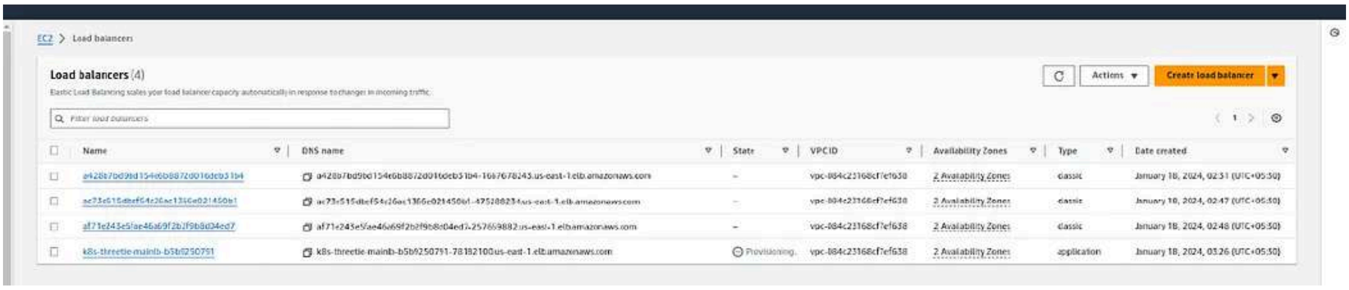
DESTINATION

Cluster URL
https://kubernetes.default.svc

Namespace
three-tier

Once your Ingress application is deployed. It will create an **Application Load Balancer**

You can check out the load balancer named with k8s-three.



Load balancers (4)

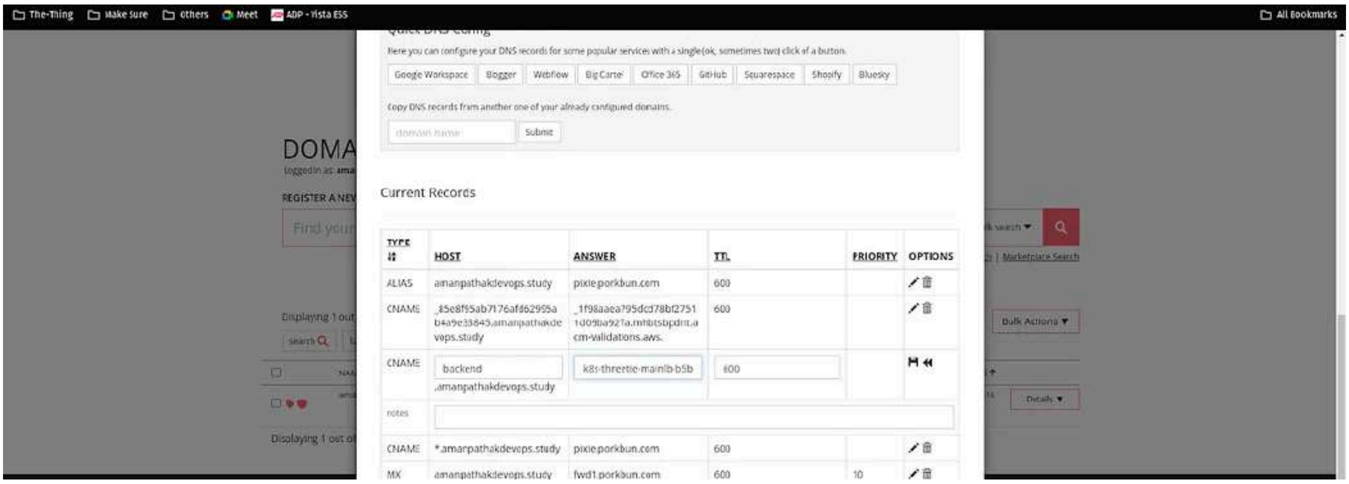
Elastic Load Balancing scales your load balancer capacity automatically in response to changes in incoming traffic.

Name	DNS name	State	VPC ID	Availability Zones	Type	Date created
a428b7bd9bd154e6b8872d016deb31b4	a428b7bd9bd154e6b8872d016deb31b4-1667678243.us-east-1.elb.amazonaws.com	–	vpc-084c23168cf7ef638	2 Availability Zones	classic	January 18, 2024, 02:31 (UTC+05:30)
ac73e515dbef54c26ac1366e021450b1	ac73e515dbef54c26ac1366e021450b1-475288234.us-east-1.elb.amazonaws.com	–	vpc-084c23168cf7ef638	2 Availability Zones	classic	January 18, 2024, 02:47 (UTC+05:30)
af71e243e5fae46a69f2b2f9b8d04ed7	af71e243e5fae46a69f2b2f9b8d04ed7-257659882.us-east-1.elb.amazonaws.com	–	vpc-084c23168cf7ef638	2 Availability Zones	classic	January 18, 2024, 02:48 (UTC+05:30)
k8s-threetie-mainlb-b5b9250791	k8s-threetie-mainlb-b5b9250791-78182100.us-east-1.elb.amazonaws.com	Provisioning	vpc-084c23168cf7ef638	2 Availability Zones	application	January 18, 2024, 03:26 (UTC+05:30)

Now, Copy the ALB-DNS and go to your Domain Provider in my case porkbun is the domain provider.

Go to **DNS** and add a **CNAME** type with hostname **backend** then add your **ALB** in the Answer and click on **Save**

Note: I have created a subdomain backend.amanpathakdevops.study



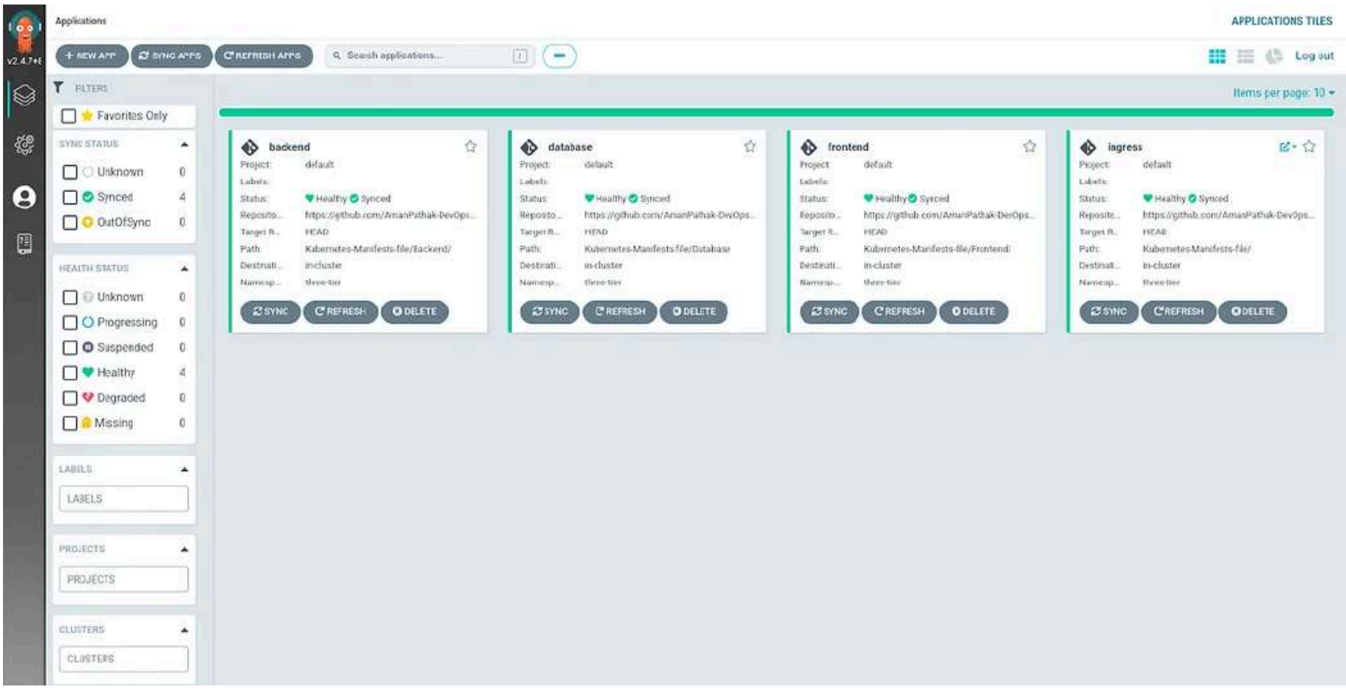
Here you can configure your DNS records for some popular services with a single(ok, sometimes two) click of a button.

Copy DNS records from another one of your already configured domains.

Current Records

TYPE	HOST	ANSWER	TTL	PRIORITY	OPTIONS
ALIAS	amanpathakdevops.study	pixie.porkbun.com	600		
CNAME	_85e8f95ab7176afd62995ab4a9e33845.amanpathakdevops.study	_1f98aaea795dcd78bf27511d09ba921a.mhbtsbpdnt.acm-validations.aws.	600		
CNAME	backend .amanpathakdevops.study	k8s-threetie-mainlb-b5b	600		
notes					
CNAME	*.amanpathakdevops.study	pixie.porkbun.com	600		
MX	amanpathakdevops.study	fwd1.porkbun.com	600	10	

You can see all 4 application deployments in the below snippet.



☐ Favorites Only

SYNC STATUS

☐ Unknown 0
☐ Synced 4
☐ OutOfSync 0

HEALTH STATUS

☐ Unknown 0
☐ Progressing 0
☐ Suspended 0
☐ Healthy 4
☐ Degraded 0
☐ Missing 0

LABELS: LABELS

PROJECTS: PROJECTS

CLUSTERS: CLUSTERS

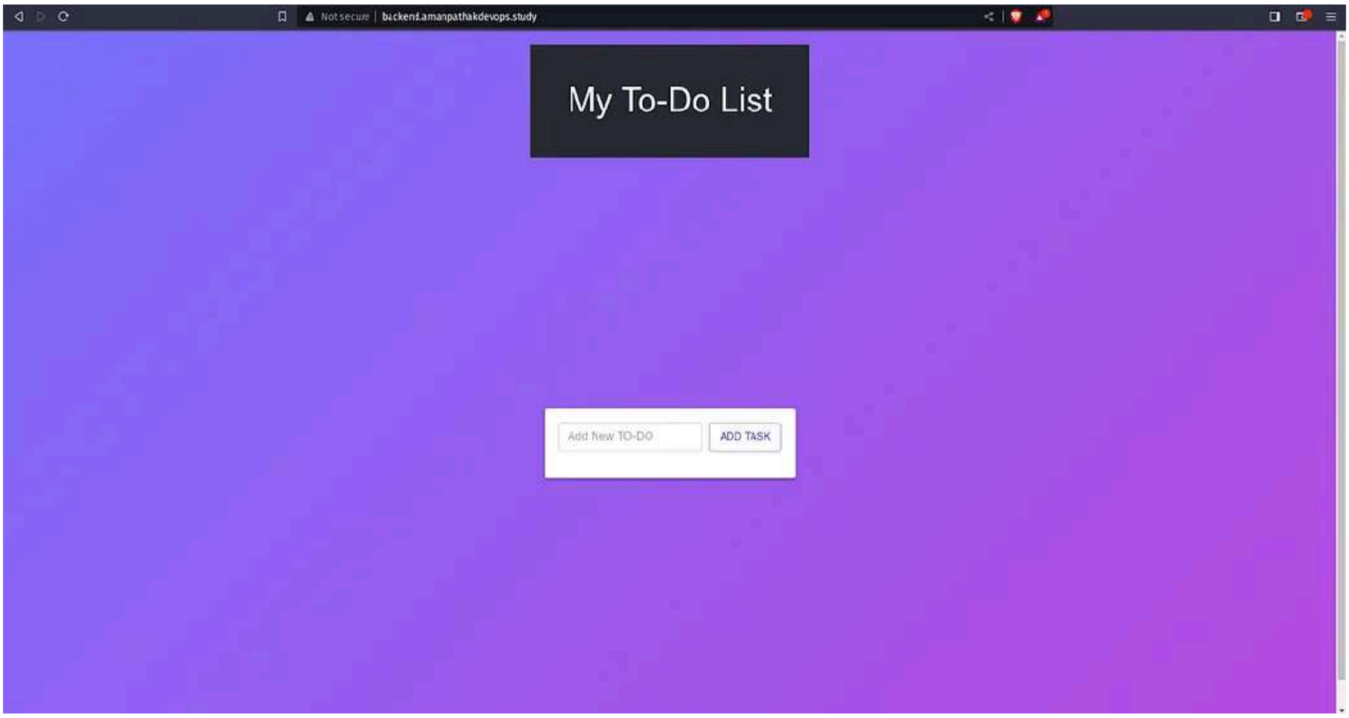
backend
Project: default
Labels:
Status: Healthy Synced
Reposito...: https://github.com/AmanPathak-DevOps...
Target R...: HEAD
Path: Kubernetes-Manifests-file/Backend/
Destinati...: in-cluste
Namesp...: three-tier

database
Project: default
Labels:
Status: Healthy Synced
Reposito...: https://github.com/AmanPathak-DevOps...
Target R...: HEAD
Path: Kubernetes-Manifests-file/Database
Destinati...: in-cluster
Namesp...: three-tier

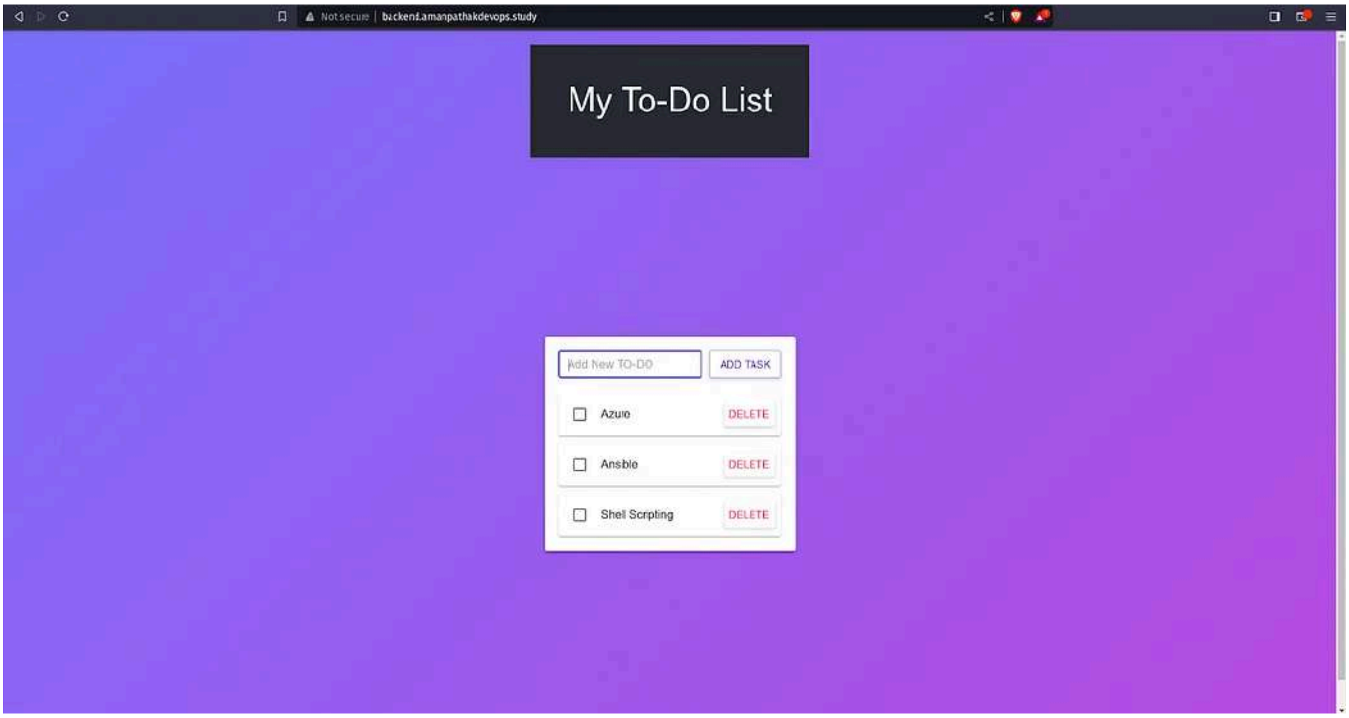
frontend
Project: default
Labels:
Status: Healthy Synced
Reposito...: https://github.com/AmanPathak-DevOps...
Target R...: HEAD
Path: Kubernetes-Manifests-file/Frontend/
Destinati...: in-cluster
Namesp...: three-tier

ingress
Project: default
Labels:
Status: Healthy Synced
Reposito...: https://github.com/AmanPathak-DevOps...
Target R...: HEAD
Path: Kubernetes-Manifests-file/
Destinat...: in-cluster
Namesp...: three-tier

Now, hit your subdomain after 2 to 3 minutes in your browser to see the magic.



You can play with the application by adding the records.



My To-Do List

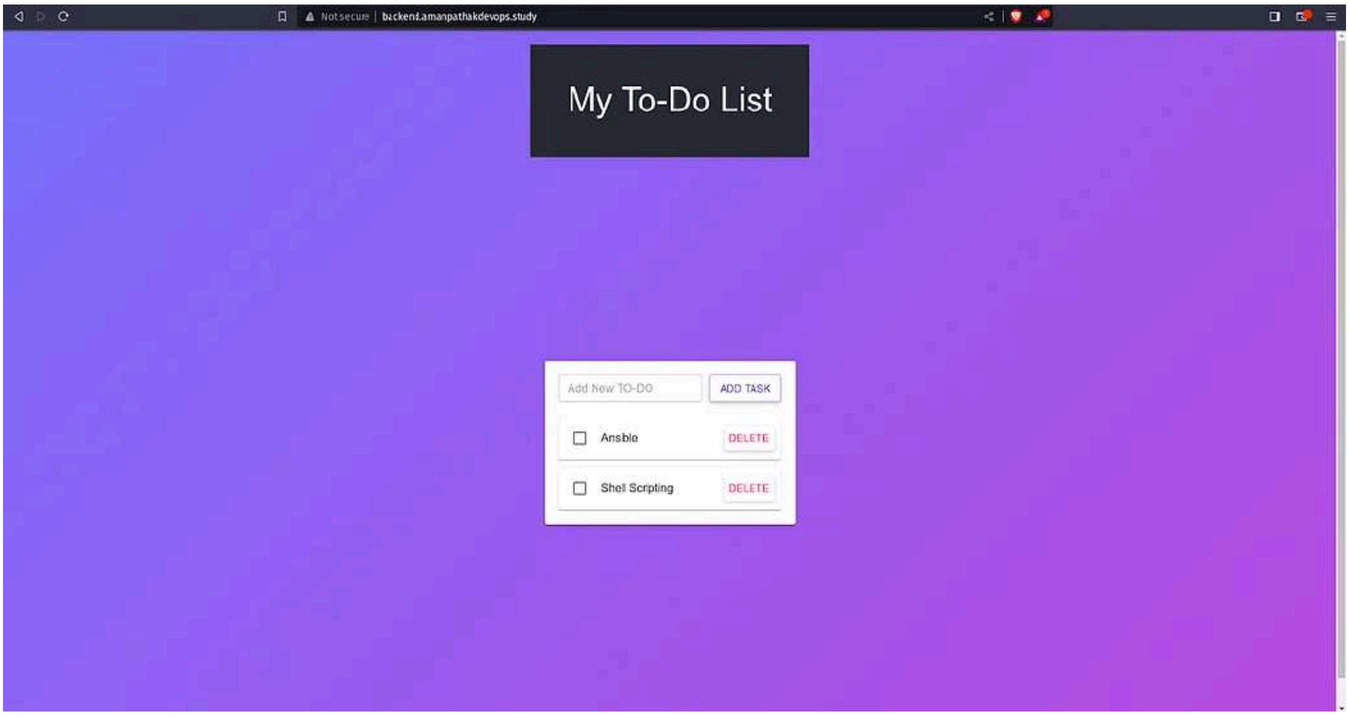
Add New TO-DO ADD TASK

□ Azure DELETE

□ Ansble DELETE

□ Shell Scrpting DELETE

You can play with the application by deleting the records.



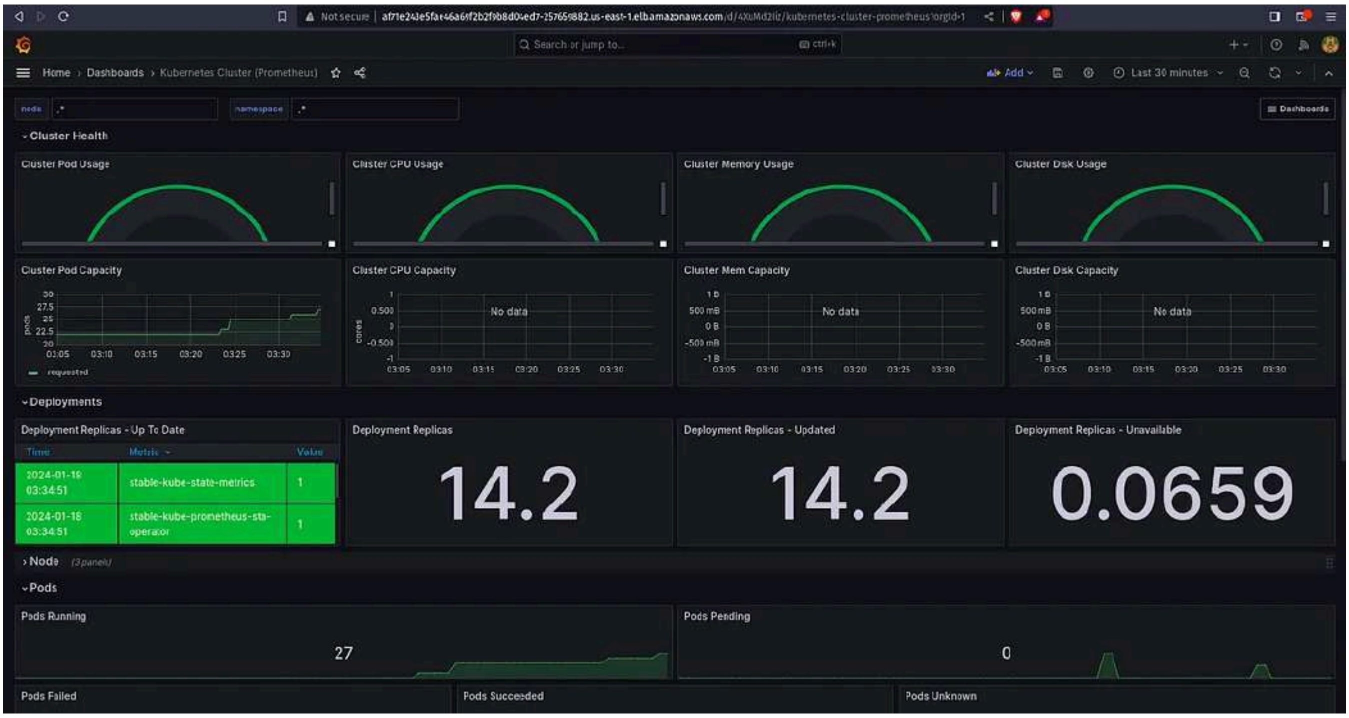
My To-Do List

Add New TO-DO ADD TASK

□ Ansble DELETE

□ Shell Scrpting DELETE

Now, you can see your Grafana Dashboard to view the EKS data such as pods, namespace, deployments, etc.



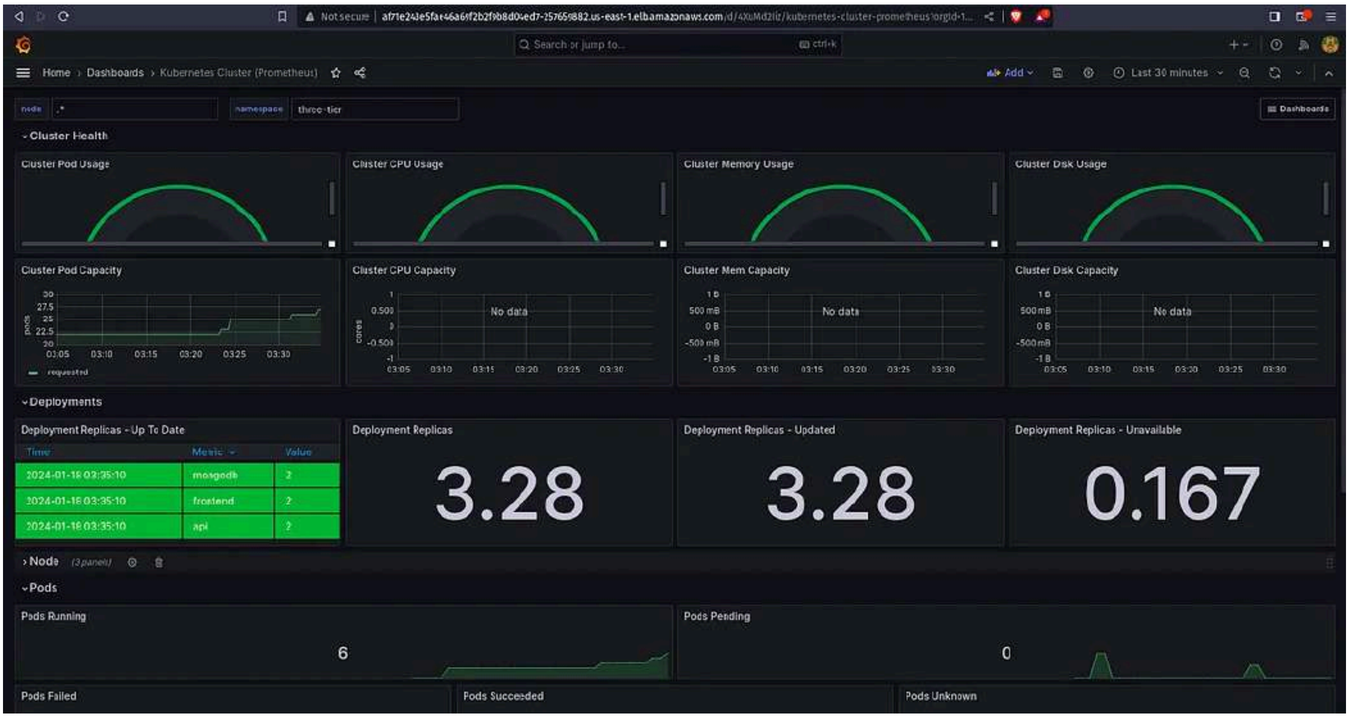
Deployment Replicas - Up To Date

Time	Metric	Value
2024-01-19 03:34:51	stable-kube-state-metrics	1
2024-01-18 03:34:51	stable-kube-prometheus-sta-operator	1

If you want to monitor the three-tier namespace.

In the namespace, replace three-tier with another namespace.

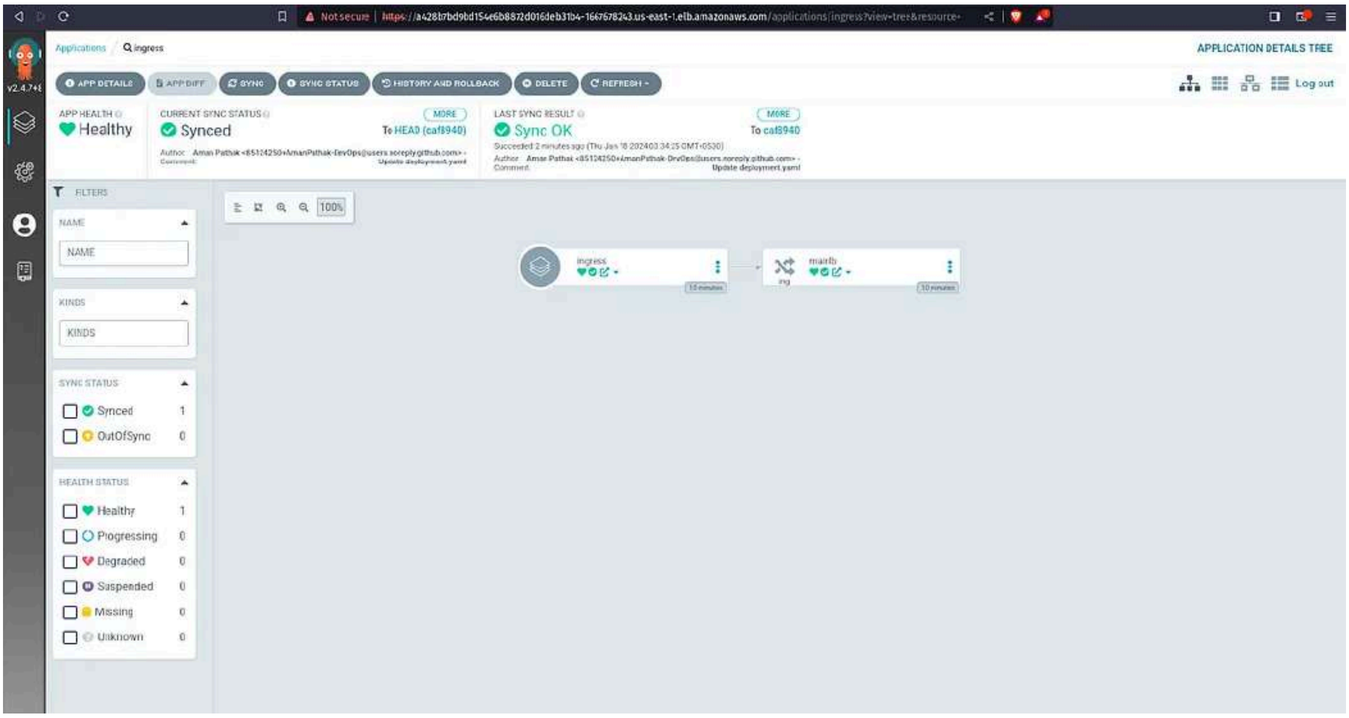
You will see the deployments that are done by ArgoCD



Deployment Replicas - Up To Date

Time	Metric	Value
2024-01-18 03:35:10	mongodb	2
2024-01-18 03:35:10	frontend	2
2024-01-18 03:35:10	api	2

This is the **Ingress** Application Deployment in ArgoCD



APP HEALTH: Healthy

CURRENT SYNC STATUS: Synced To HEAD (caf8940)

Author: Aman Pathak <85124250+AmanPathak-DevOps@users.noreply.github.com> -
Comment: Update deployment.yaml

LAST SYNC RESULT: Sync OK To caf8940

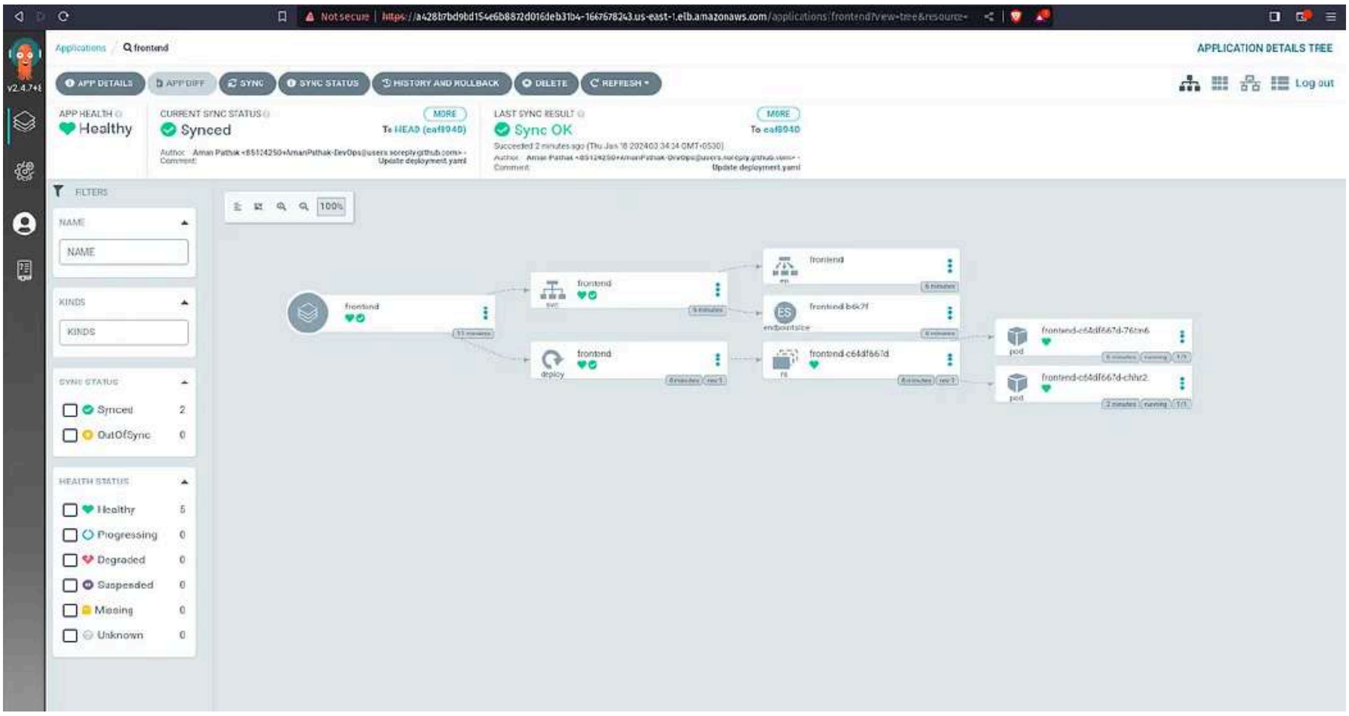
Succeeded 2 minutes ago (Thu Jan 18 2024 03:34:25 GMT+0530)

Author: Aman Pathak <85124250+AmanPathak-DevOps@users.noreply.github.com> -
Comment: Update deployment.yaml

SYNC STATUS	
□ Synced	1
□ OutOfSync	0

HEALTH STATUS	
□ Healthy	1
□ Progressing	0
□ Degraded	0
□ Suspended	0
□ Missing	0
□ Unknown	0

This is the **Frontend** Application Deployment in ArgoCD



APP HEALTH: Healthy

CURRENT SYNC STATUS: Synced To HEAD (caf8940)

Author: Aman Pathak <85124250+AmanPathak-DevOps@users.noreply.github.com> -
Comment: Update deployment.yaml

LAST SYNC RESULT: Sync OK To caf8940

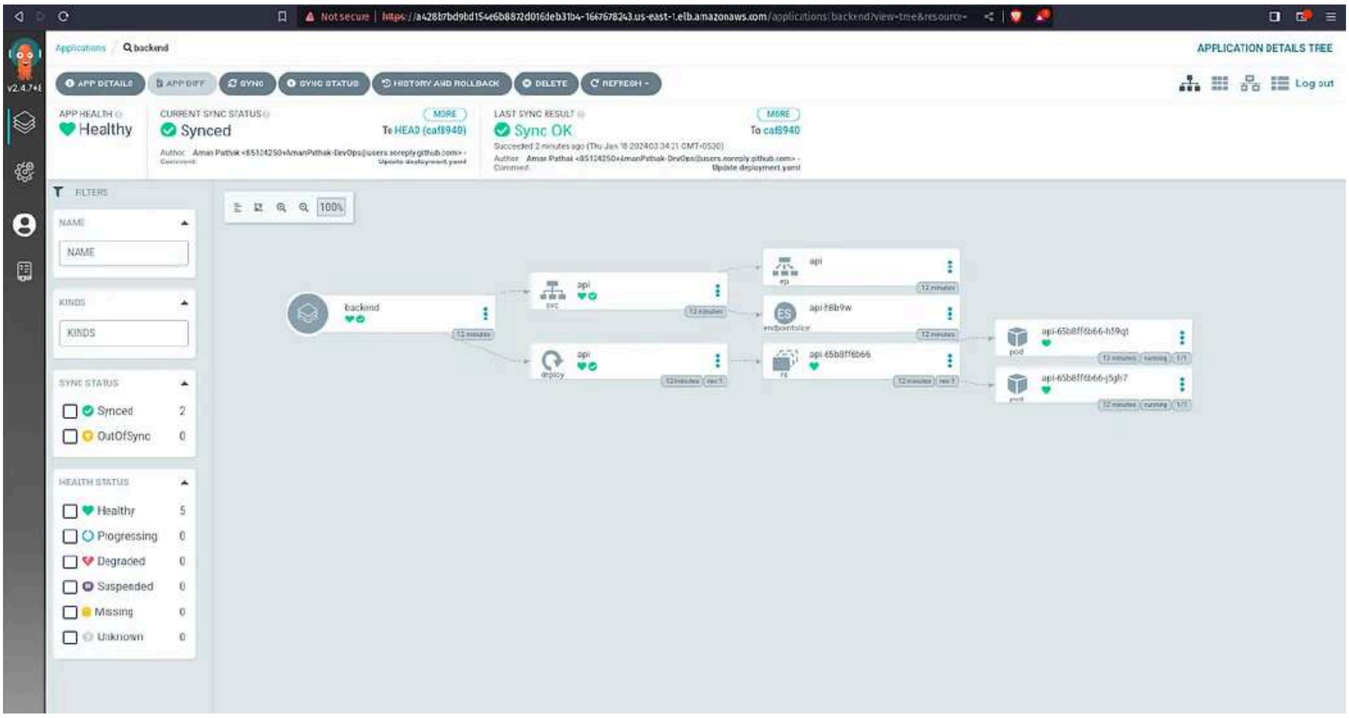
Succeeded 2 minutes ago (Thu Jan 18 2024 03:34:34 GMT+0530)

Author: Aman Pathak <85124250+AmanPathak-DevOps@users.noreply.github.com> -
Comment: Update deployment.yaml

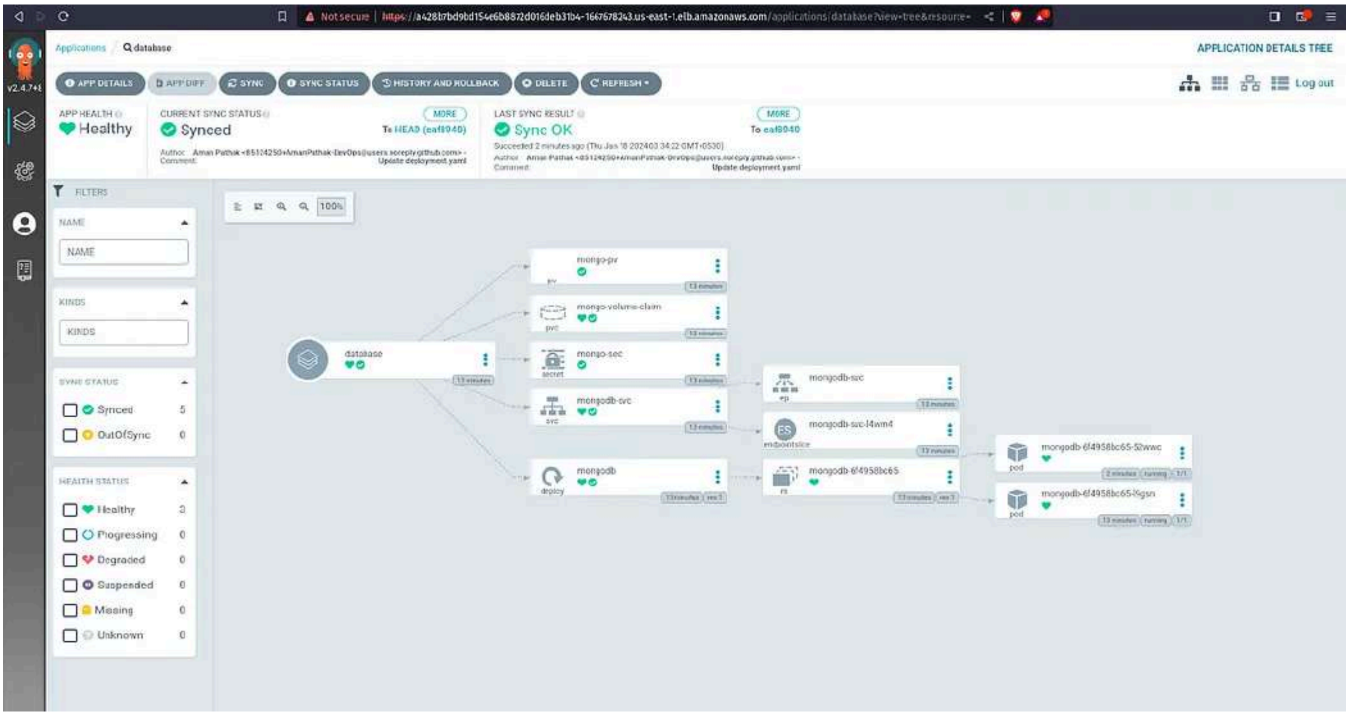
SYNC STATUS	
□ Synced	2
□ OutOfSync	0

HEALTH STATUS	
□ Healthy	5
□ Progressing	0
□ Degraded	0
□ Suspended	0
□ Missing	0
□ Unknown	0

This is the **Backend** Application Deployment in ArgoCD

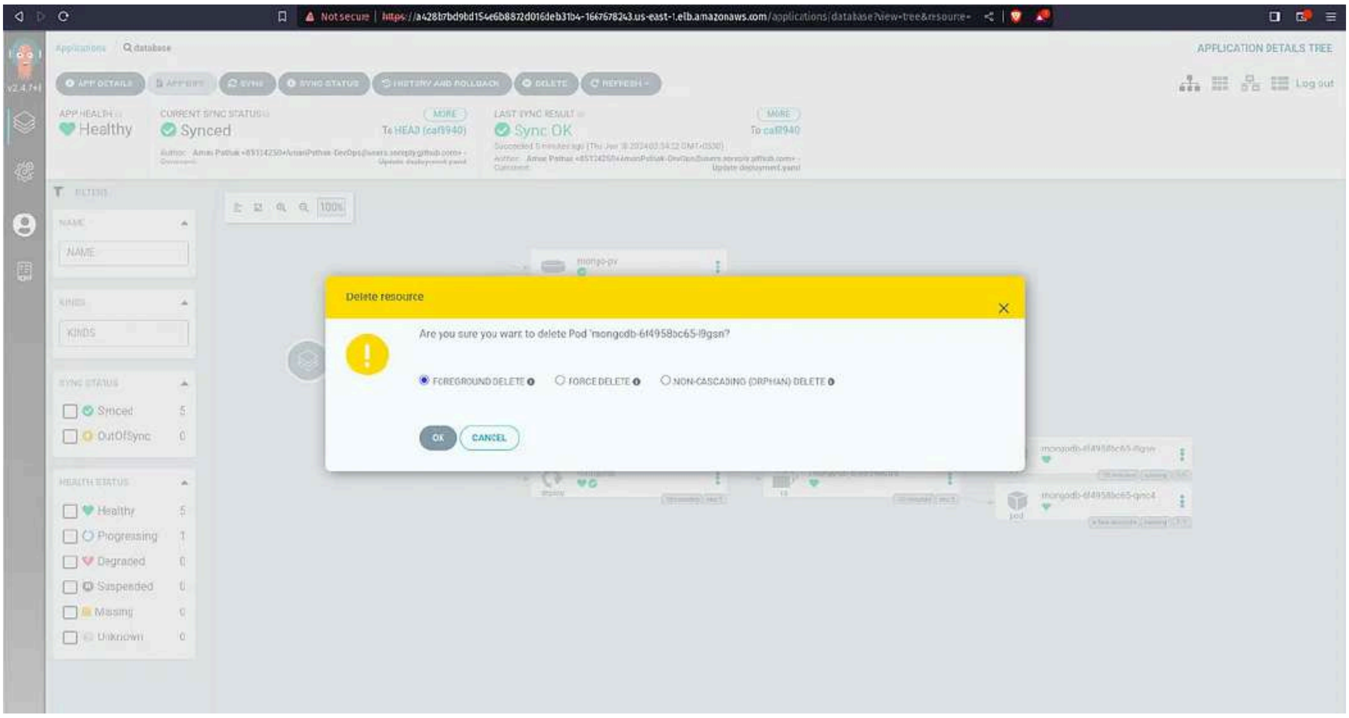


This is the **Database** Application Deployment in ArgoCD



If you observe, we have configured the Persistent Volume & Persistent Volume Claim. So, if the pods get deleted then, the data won't be lost. The Data will be stored on the host machine.

To validate it, delete both Database pods.

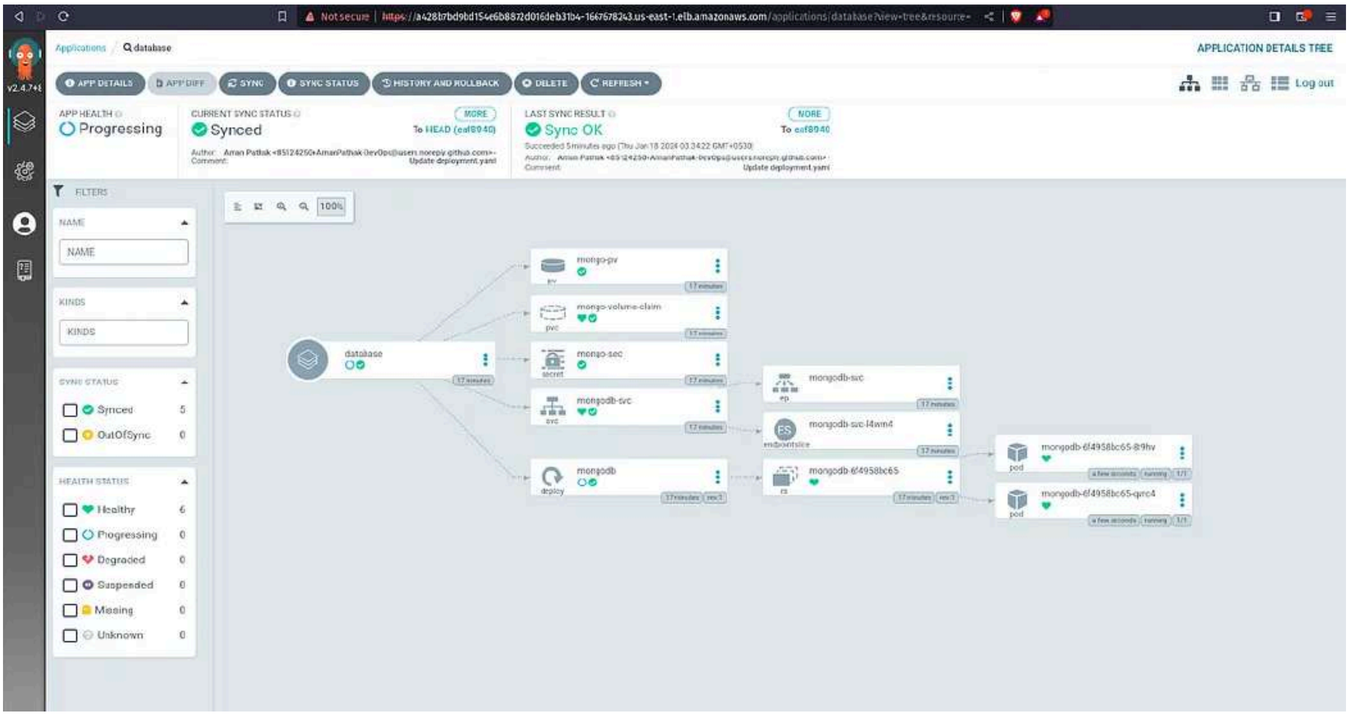


Delete resource

Are you sure you want to delete Pod 'mongodb-6f4958bc65-l9gsn'?

● FOREGROUND DELETE ○ FORCE DELETE ○ NON-CASCADING (ORPHAN) DELETE

Now, the new pods will be started.

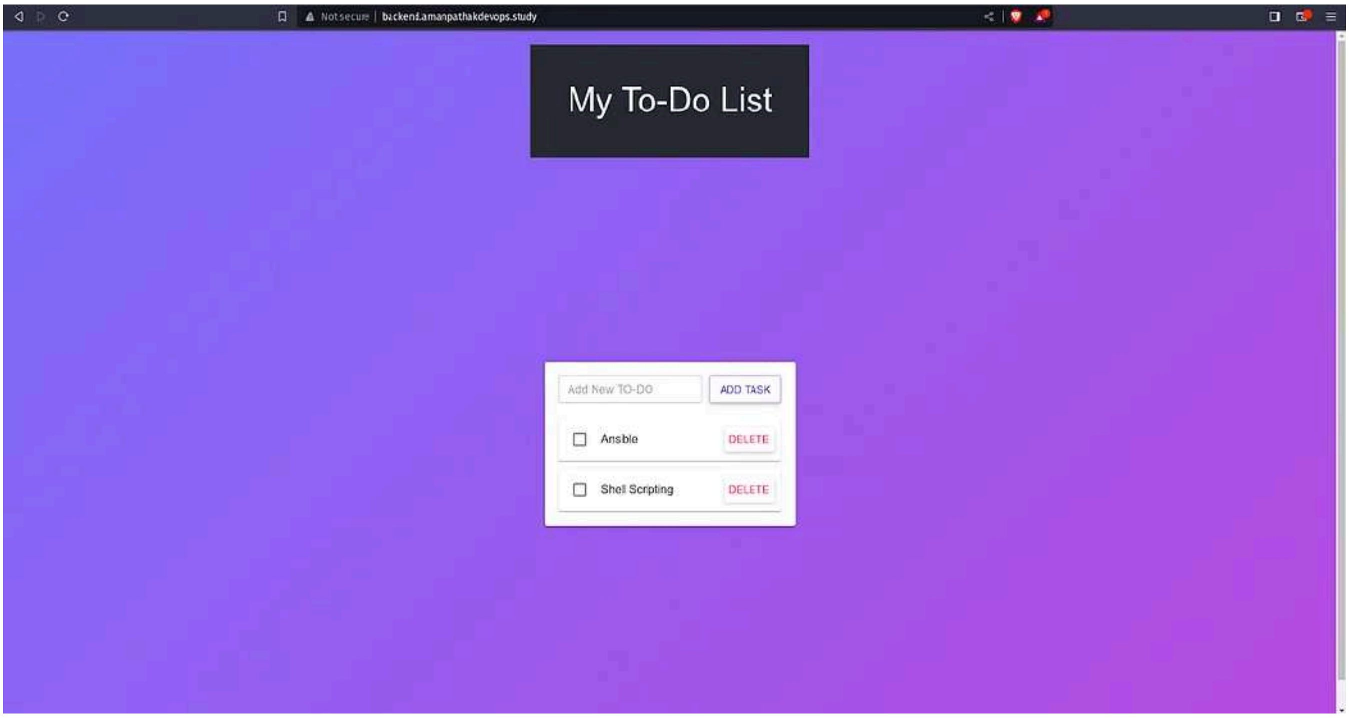


APP HEALTH: Progressing

CURRENT SYNC STATUS: Synced To HEAD (caf8940)

LAST SYNC RESULT: Sync OK To caf8940

And Your Application won’t lose a single piece of data.



☐ Ansble

☐ Shell Scrpting

Conclusion:

In this comprehensive DevSecOps Kubernetes project, we successfully:

- Established IAM user and Terraform for AWS setup.
- Deployed Jenkins on AWS, configured tools, and integrated it with Sonarqube.
- Set up an EKS cluster, configured a Load Balancer, and established private ECR repositories.
- Implemented monitoring with Helm, Prometheus, and Grafana.
- Installed and configured ArgoCD for GitOps practices.
- Created Jenkins pipelines for CI/CD, deploying a Three-Tier application.
- Ensured data persistence with persistent volumes and claims.

Support my work-
https://www.buymeacoffee.com/aman.pathak

Stay connected on **LinkedIn**: LinkedIn Profile

Stay up-to-date with **GitHub**: GitHub Profile

Want to discuss trending technologies in DevOps & Cloud
Join the **Discord Server**- https://discord.gg/jdzF8kTtw2

Feel free to reach out to me, if you have any other queries.

Happy Learning!

Stackademic

Thank you for reading until the end. Before you go:

- Please consider **clapping** and **following** the writer! 👏
- Follow us **X** | **LinkedIn** | **YouTube** | **Discord**
- Visit our other platforms: **In Plain English** | **CoFeed** | **Venture**

Kubernetes DevOps Devsecops Jenkins AWS

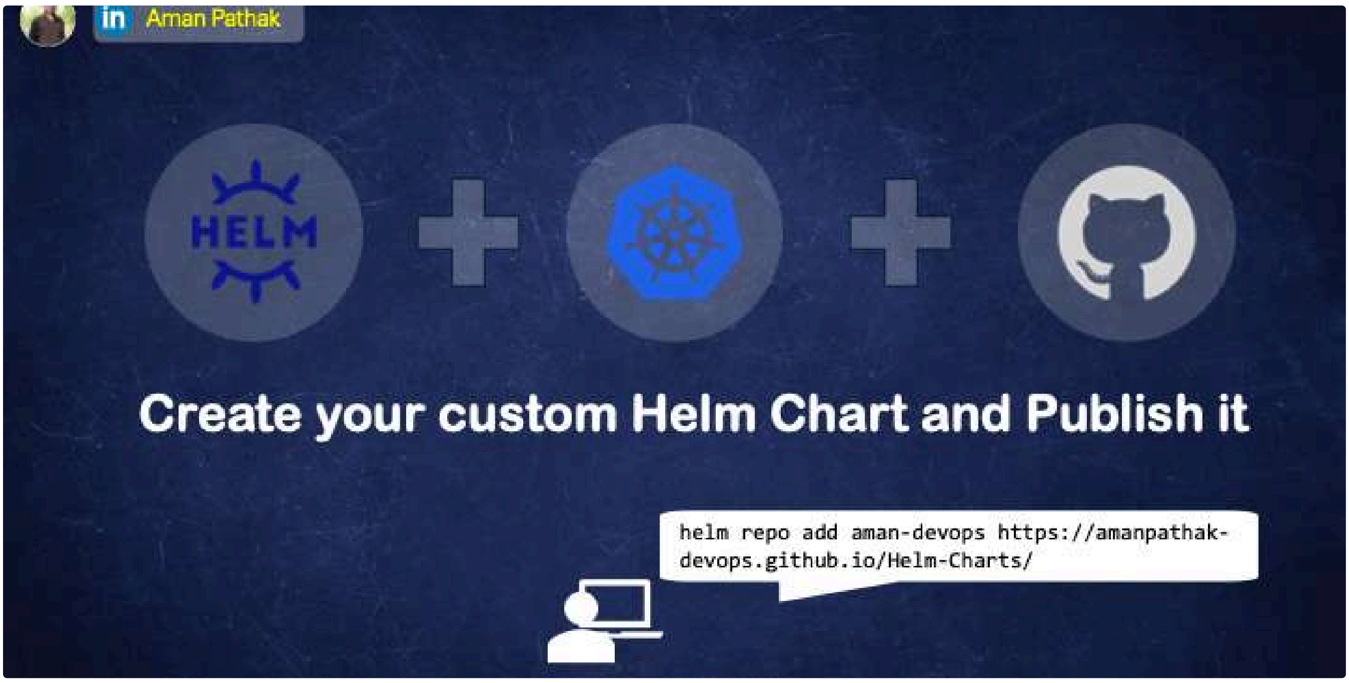
Follow

Written by Aman Pathak

9.4K Followers · Writer for Stackademic

DevOps & Cloud Engineer | AWS Community Builder | AWS Certified | Azure | Terraform | Docker | Ansible | CI/CD Jenkins | Oracle Certified

More from Aman Pathak and Stackademic


Aman Pathak
HELM
Create your custom Helm Chart and Publish it
helm repo add aman-devops https://amanpathak-devops.github.io/Helm-Charts/

Aman Pathak in Towards Dev

How to Package and Publish Your Custom Helm Charts- A Hands-On Tutorial

Introduction

Oct 13 24

Abdur Rahman in Stackademic

Python is No More The King of Data Science

5 Reasons Why Python is Losing Its Crown

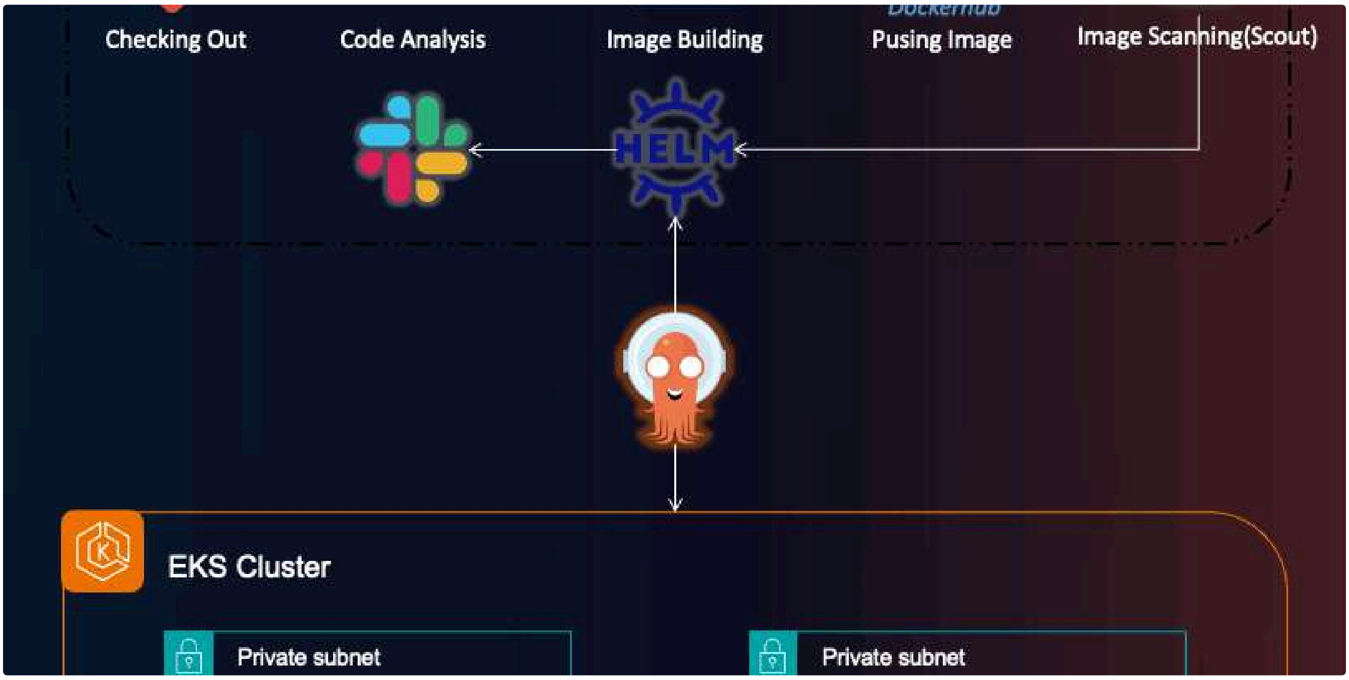
Oct 23 4.4K 23

Abdur Rahman in Stackademic

20 Python Scripts To Automate Your Daily Tasks

A must-have collection for every developer

Oct 7 2.4K 24



Aman Pathak in Towards Dev

Deploy Go Application on EKS Cluster using GitHub Actions, Terraform, Helm, and ArgoCD

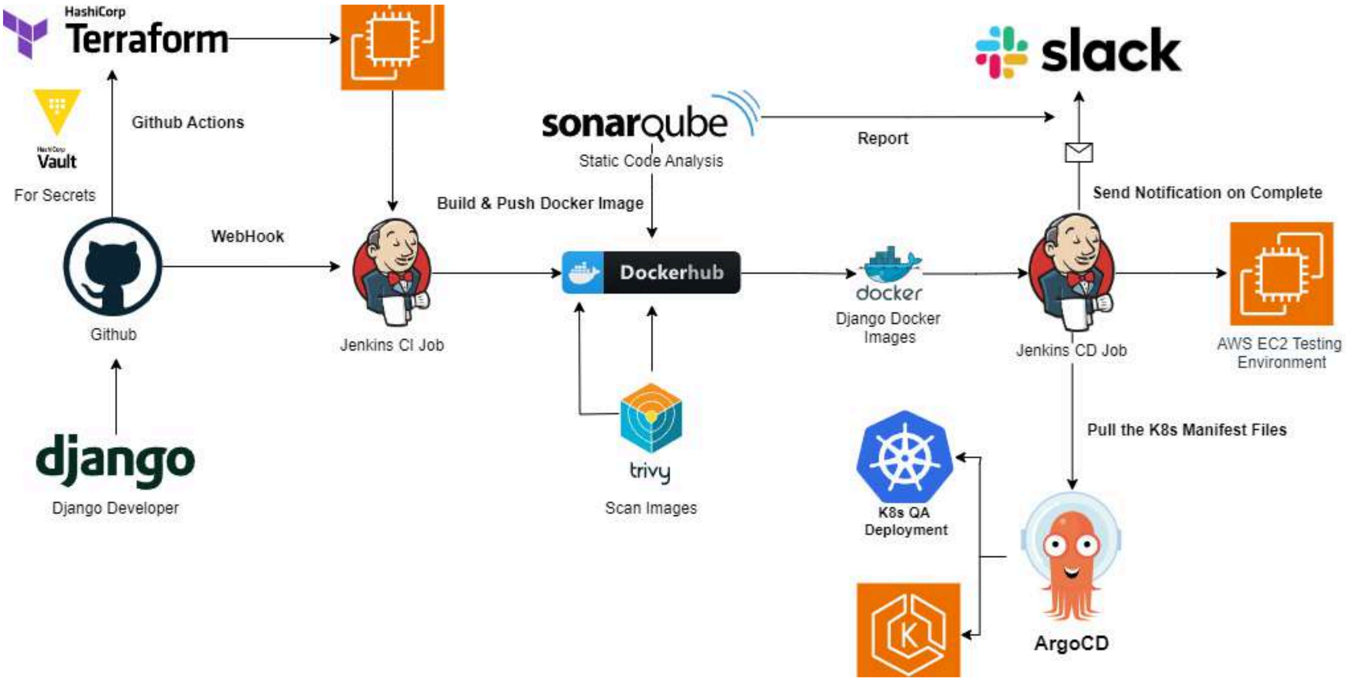
Introduction

Oct 20 36 3

See all from Aman Pathak

See all from Stackademic

Recommended from Medium

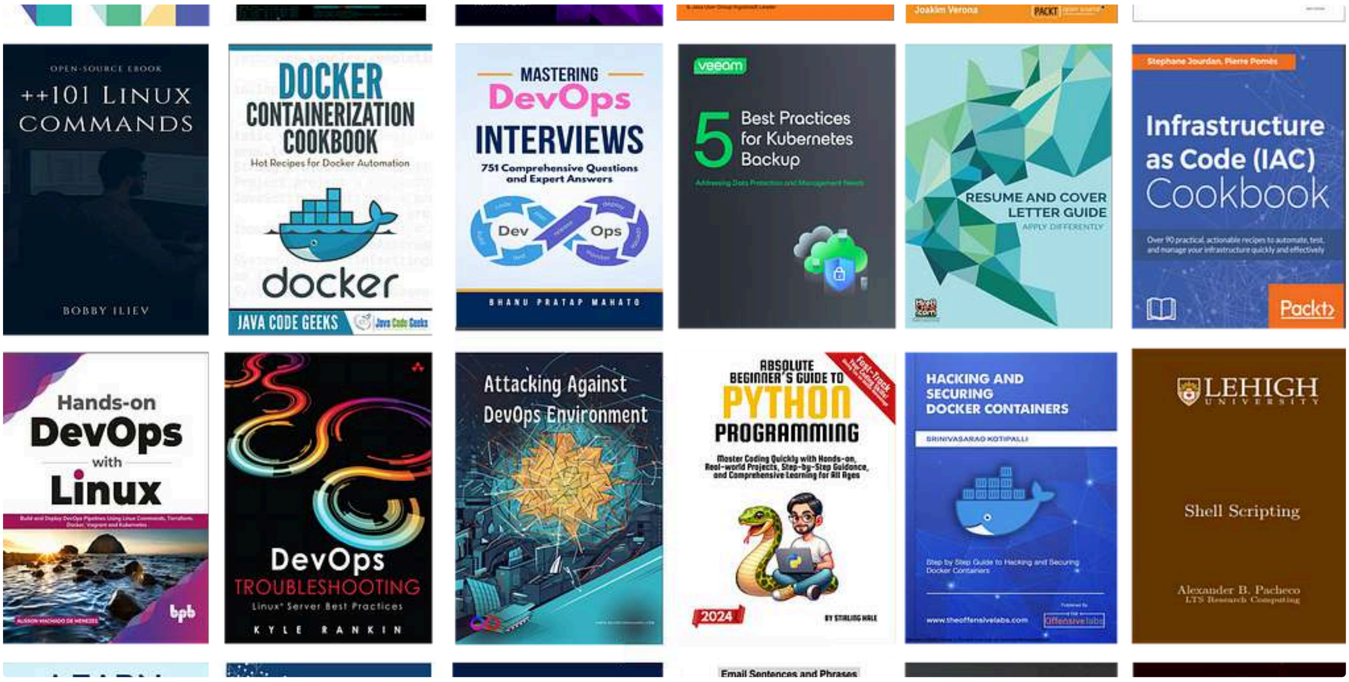


Joel Wembo in Django Unleashed

Technical Guide: End-to-End CI/CD DevOps with Jenkins, Docker, Kubernetes, ArgoCD, Github Actions …

Building an end-to-end CI/CD pipeline for Django applications using Jenkins, Docker, Kubernetes, ArgoCD, AWS EKS, AWS EC2

Apr 12 908 20



Cumhur Akkaya

DevOps and Cloud Resources 2024 (PDF Books)

I updated DevOps and Cloud resources. I added 51 new documents. In this article;🔎 You may find the pdf of the resources about DevOps Tools...

✦ Sep 17 133 2

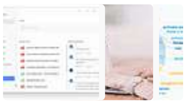
Lists

General Coding Knowledge

20 stories · 1714 saves

Natural Language Processing

1801 stories · 1415 saves



Productivity

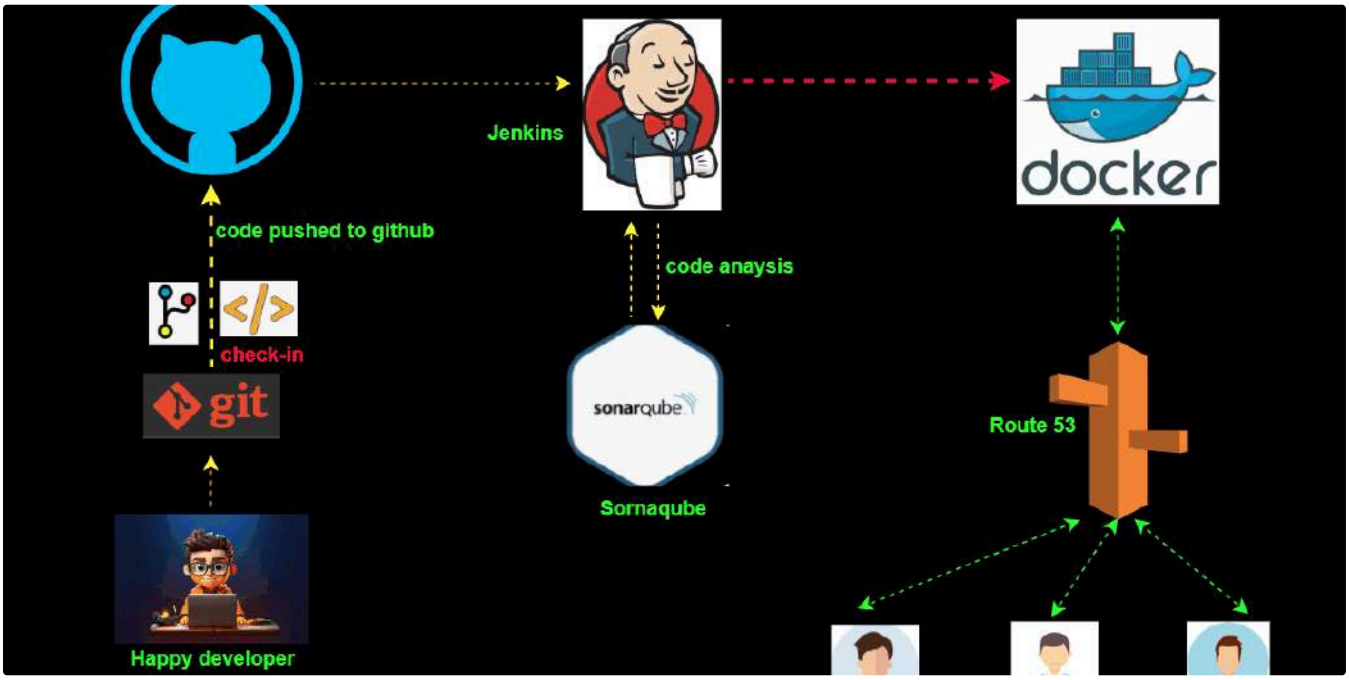
242 stories · 604 saves

Obafemi in DevOps.dev

7 DevOps weekend projects

sharpen specific DevOps skills.

Oct 26 123 4



Victor wasonga onyango

Building a Robust CI/CD Pipeline with Jenkins, SonarQube, Docker, and GitHub on AWS.

Introduction

Jul 27 74

Harendra

How I Am Using a Lifetime 100% Free Server

Get a server with 24 GB RAM + 4 CPU + 200 GB Storage + Always Free

Oct 26 3.7K 44

techwithpatil

Top 10 Common DevOps/SRE Interview Questions and Answers on Kubernetes Best Practices

How Do You Set Resource Requests and Limits in Kubernetes?

✦ Sep 23 👏 33

See more recommendations