



🇬🇧🇯🇵🇸🇬🇸🇬 CICD Group Activity (by using Jenkins Pipeline) (1)

▼ Group Activities 🇬🇧🇯🇵🇸🇬🇸🇬 (Netflix Clone Deployment on EKS with Jenkins CI/CD Pipeline)

Own testing Project (mini) netfilix-clone (by using Jenkins CI/CD PIPE Line)

Group Name	ROBOTs aka <i>(Super Active Guys)</i>
Project Name	Netflix Clone Deployment by using Jenkins CICD
participants	Ko Than Htike, Ma Ei Mi Mi Aung, Ma Chin Phoo, Ko Teddy

Summary

To access the movie sample web page deployment by using Jenkins CICD pipeline. by orchestrating cloud nature (AWS, Git, Kubernetes, Sonar, Jenkins, TMB, Docker and so on).

Requirements:

- AWS account with appropriate permissions
- Terraform installed
- AWS CLI configured
- eksctl installed
- Jenkins, SonarQube, and Trivy installed on an EC2 instance
- GitHub repository: <https://github.com/TEDDY201289/Netflix-test.git>
- TMDB API key from <https://www.themoviedb.org/settings/api>
- Google account for SMTP configuration (**Optional**)

- Docker installed on the Jenkins EC2 instance

1. First of All - needs to have and start with AWS account to create vpc/subent/IGW, EC2 and EKS , so on.

The screenshot shows the AWS IAM 'Users' page. On the left, there's a sidebar with 'Identity and Access Management (IAM)' and links to 'Dashboard', 'Access management' (which is expanded), 'User groups', 'Users' (which is selected and highlighted in blue), 'Roles', 'Policies', 'Identity providers', 'Account settings', and 'Root access management'. The main area is titled 'Users (3)'. It contains a table with columns: 'User name', 'Path', 'Group', 'Last activity', 'MFA', 'Password age', and 'Console last sign-in'. The users listed are 'master-console-admin', 'Master-Hello', and 'teddy-kube', each with their respective details like last activity time and password age.

I will use this iam role for this task

The screenshot shows the 'teddy-kube' user details page. The 'Summary' tab is active, displaying information such as ARN (arn:awsiam::774305596727:user/teddy-kube), Console access (Disabled), Last console sign-in (not shown), and two access keys. The 'Permissions' tab is also visible. In the 'Permissions' section, it shows 'Permissions policies (1)' attached to the user. A table lists the policy: 'AdministratorAccess' (AWS managed - job function). There are buttons for 'Remove' and 'Add permissions'.

Create the VPC with this below steps:

Then subnet mask, IGW, RTB all are created to allocate for Jenkins server automatically before deploying by terraform.

Create Keypair at AWS (pre-creation)

EC2 > Key pairs > Create key pair

Create key pair Info

Key pair
A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.

Name The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

Key pair type Info RSA ED25519

Private key file format .pem For use with OpenSSH
 .ppk For use with PuTTY

Tags - optional
No tags associated with the resource.
[Add new tag](#) You can add up to 50 more tags.

[Cancel](#) [Create key pair](#)

Install AWS cli for MAC users

```
teddyyekyawthu@mac Teddy-files % brew update
teddyyekyawthu@mac Teddy-files % brew install awscli
teddyyekyawthu@mac Teddy-files % aws --version
aws-cli/2.22.28 Python/3.12.6 Darwin/24.6.0 exe/x86_64
```

create aws configure in PC (to bind to create/use/deploy for Kubernetes cluster by using eksctl with created profile which is to authenticate with own AWS CLI)

```
teddyyekyawthu@mac Teddy-files % aws configure

AWS Access Key ID [None]: <your-access-key-id>
AWS Secret Access Key [None]: <your-secret-access-key>
Default region name [None]: ap-southeast-1 # Singapore example
Default output format [None]: json      # or table / text
```

```
teddyyekyawthu@mac .aws % cd ~/.aws
teddyyekyawthu@mac .aws % ls -al
total 16
drwxr-xr-x@ 4 teddyyekyawthu staff 128 Jul 10 22:31 .
drwxr-xr-x+ 71 teddyyekyawthu staff 2272 Aug 10 10:42 ..
-rw-----@ 1 teddyyekyawthu staff 188 Aug 8 11:03 config
-rw-----@ 1 teddyyekyawthu staff 375 Aug 8 11:03 credentials
```

```
teddyyekyawthu@mac .aws % cat config

[profile teddy-kube]
region = ap-southeast-1
output = json
```

```
teddyyekyawthu@mac .aws % cat credentials
```

```
[teddy-kube]
aws_access_key_id = AKIxxxxxxxxxBHP
aws_secret_access_key = PVGxxxxxxxxQAcxxxxx4zkyszt
```

install eksctl for Mac

```
teddyyekyawthu@mac Teddy-files % brew tap aws/tap
teddyyekyawthu@mac Teddy-files % brew install eksctl
teddyyekyawthu@mac Teddy-files % eksctl version
0.212.0-dev+db83da480.2025-07-29T20:05:26Z
```

terraform HCL version

```
teddyyekyawthu@mac Teddy-files % terraform version
Terraform v1.12.2
on darwin_arm64
```

```
# Earlier my one was plugged-in with asdf shell so that even I downloaded by using brew ,
but still showing 1.10.5
```

```
# teddyyekyawthu@mac Teddy-files % terraform version
# Terraform v1.10.5
# on darwin_arm64
```

```
# Your version of Terraform is out of date! The latest version
# is 1.12.2. You can update by downloading from https://www.terraform.io/downloads.html
# teddyyekyawthu@mac Teddy-files % which terraform
```

```
# /Users/teddyyekyawthu/.asdf/shims/terraform
# teddyyekyawthu@mac Teddy-files % /usr/local/bin/terraform # old version
```

```
# zsh: no such file or directory: /usr/local/bin/terraform
# teddyyekyawthu@mac Teddy-files % which terraform
```

```
# /Users/teddyyekyawthu/.asdf/shims/terraform
```

```
# teddyyekyawthu@mac Teddy-files % echo 'export PATH="/opt/homebrew/bin:$PATH"'>> ~/.zshrc
# source ~/.zshrc
# hash -r
# teddyyekyawthu@mac Teddy-files % asdf plugin remove terraform
# hash -r
# teddyyekyawthu@mac Teddy-files % which -a terraform
# /opt/homebrew/bin/terraform
# /opt/homebrew/bin/terraform
```

```
# /opt/homebrew/bin/terraform  
# /opt/homebrew/bin/terraform
```

Infrastructure readiness by using terraform (for Jenkins, Sonarcube, Docker, Trivy etc)

Write Terraform code ([provider.tf](#))

```
terraform {  
    required_providers {  
        aws = {  
            source  = "hashicorp/aws"  
            version = "~> 5.0"  
        }  
    }  
}  
  
# Configure the AWS Provider  
provider "aws" {  
    region  = "ap-southeast-1" #change region as per you requirement  
    profile = "teddy-kube" #add your cli profile *** important  
}
```

Write Terraform code ([main.tf](#))

```
resource "aws_instance" "web" {  
    ami           = "ami-06c4be2792f419b7b" #change ami id for different region → this is  
    ap-southeast-1 region  
    instance_type     = "t2.large" #recommended to use  
    key_name         = "cicd" #change keypair name as per your setup in AWS  
    vpc_security_group_ids = [aws_security_group.Jenkins-VM-SG.id]  
    subnet_id        = "subnet-0d01d8b3e2f5bf885" #change ur subnet id  
    user_data = templatefile("./install.sh", {})  
  
    tags = {  
        Name = "Jenkins-SonarQube" #any name you can given  
    }  
  
    root_block_device {  
        volume_size = 40  
    }  
}  
  
resource "aws_security_group" "Jenkins-VM-SG" {  
    name      = "Jenkins-VM-SG"  
    description = "Allow TLS inbound traffic"
```

```

vpc_id      = "vpc-00d7e4891080b872f" ###change vpc id

ingress = [
    for port in [22, 80, 443, 8080, 9000, 3000] : {
        description      = "inbound rules"
        from_port        = port
        to_port          = port
        protocol         = "tcp"
        cidr_blocks     = ["0.0.0.0/0"]
        ipv6_cidr_blocks = []
        prefix_list_ids = []
        security_groups = []
        self             = false
    }
]

egress {
    from_port  = 0
    to_port    = 0
    protocol   = "-1"
    cidr_blocks = ["0.0.0.0/0"]
}

tags = {
    Name = "Jenkins-VM-SG"
}

```

original file from KO Hlaing

```

resource "aws_instance" "web" {
    ami                  = "ami-06c4be2792f419b7b" # Update AMI ID for your region
    instance_type        = "t2.large"
    key_name            = "keypair" # Update key name
    vpc_security_group_ids = [aws_security_group.Jenkins-VM-SG.id]
    user_data           = templatefile("./install.sh", {})

    tags = {
        Name = "Jenkins-SonarQube"
    }

    root_block_device {
        volume_size = 25
    }
}

```

```

resource "aws_security_group" "Jenkins-VM-SG" {
  name      = "Jenkins-VM-SG"
  description = "Allow TLS inbound traffic"
  vpc_id    = "vpc-0d5b095006ca56d4f" # Update VPC ID

  ingress = [
    for port in [22, 80, 443, 8080, 9000, 3000] : {
      description      = "inbound rules"
      from_port        = port
      to_port          = port
      protocol         = "tcp"
      cidr_blocks     = ["0.0.0.0/0"]
      ipv6_cidr_blocks = []
      prefix_list_ids = []
      security_groups  = []
      self             = false
    }
  ]

  egress {
    from_port  = 0
    to_port    = 0
    protocol   = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }

  tags = {
    Name = "Jenkins-VM-SG"
  }
}

```

bash for installation ([./install.sh](#))

```

#!/bin/bash
sudo apt update -y
wget -O - https://packages.adoptium.net/artifactory/api/gpg/key/public | tee /etc/apt/keys/adoptium.asc
echo "deb [signed-by=/etc/apt/keys/adoptium.asc] https://packages.adoptium.net/artifactory/deb $(awk -F= '/^VERSION_CODENAME/{print$2}' /etc/os-release) main" | tee /etc/apt/sources.list.d/adoptium.list
sudo apt update -y
sudo apt install temurin-21-jdk -y
/usr/bin/java --version
curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key | sudo tee /usr/share/keys/jenkins-keyring.asc > /dev/null

```

```

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update -y
sudo apt-get install jenkins -y
sudo systemctl start jenkins
sudo systemctl status jenkins

# Install Docker and Run SonarQube as Container
sudo apt-get update
sudo apt-get install docker.io -y
sudo usermod -aG docker ubuntu
sudo usermod -aG docker jenkins
newgrp docker
sudo chmod 777 /var/run/docker.sock
docker run -d --name sonar -p 9000:9000 --restart unless-stopped sonarqube:its-community

# Install Trivy
sudo apt-get install wget apt-transport-https gnupg lsb-release -y
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | gpg --dearmor | sudo tee /usr/share/keyrings/trivy.gpg > /dev/null
echo "deb [signed-by=/usr/share/keyrings/trivy.gpg] https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main" | sudo tee -a /etc/apt/sources.list.d/trivy.list
sudo apt-get update
sudo apt-get install trivy -y

```

Optional if you want to install at Jenkins server for below: you can apply manually inside your server.

```

##install aws cli
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
sudo apt install unzip
unzip awscliv2.zip
sudo ./aws/install
aws --version

###eksctl
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
cd /tmp
sudo mv /tmp/eksctl /bin
eksctl version

##install kubectl
sudo apt update -y
sudo apt install curl -y

```

```
curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl  
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl  
kubectl version --client
```

after that , Go to the desire path (directory) to deploy with terraform (automation)

```
/Users/teddyyekyawthu/Teddy-files/Netflix-clone-bubududu/Deploy-Netflix-Clone-on-Kubernetes/terraform-jenkins
```

```
teddyyekyawthu@mac terraform-jenkins % terraform init  
teddyyekyawthu@mac terraform-jenkins % terraform fmt  
teddyyekyawthu@mac terraform-jenkins % terraform validate  
teddyyekyawthu@mac terraform-jenkins % terraform plan
```

```
teddyyekyawthu@mac terraform-jenkins % terraform plan  
aws_security_group.Jenkins-VM-SG: Refreshing state... [id=sg-0f1bfed39db942ac0]
```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:

+ create

Terraform will perform the following actions:

```
# aws_instance.web will be created  
+ resource "aws_instance" "web" {  
    + ami                  = "ami-06c4be2792f419b7b"  
    + arn                  = (known after apply)  
    + associate_public_ip_address = (known after apply)  
    + availability_zone      = (known after apply)  
    + cpu_core_count         = (known after apply)  
    + cpu_threads_per_core   = (known after apply)  
    + disable_api_stop       = (known after apply)  
    + disable_api_termination = (known after apply)  
    + ebs_optimized          = (known after apply)  
    + enable_primary_ipv6     = (known after apply)  
    + get_password_data      = false  
    + host_id                = (known after apply)  
    + host_resource_group_arn = (known after apply)  
    + iam_instance_profile    = (known after apply)  
    + id                     = (known after apply)  
    + instance_initiated_shutdown_behavior = (known after apply)  
    + instance_lifecycle      = (known after apply)  
    + instance_state          = (known after apply)  
    + instance_type            = "t2.large"  
    + ipv6_address_count      = (known after apply)  
    + ipv6_addresses          = (known after apply)
```

```

+ key_name          = "cicd"
+ monitoring        = (known after apply)
+ outpost_arn       = (known after apply)
+ password_data     = (known after apply)
+ placement_group   = (known after apply)
+ placement_partition_number = (known after apply)
+ primary_network_interface_id = (known after apply)
+ private_dns        = (known after apply)
+ private_ip         = (known after apply)
+ public_dns         = (known after apply)
+ public_ip          = (known after apply)
+ secondary_private_ips = (known after apply)
+ security_groups    = (known after apply)
+ source_dest_check  = true
+ spot_instance_request_id = (known after apply)
+ subnet_id          = "subnet-0d01d8b3e2f5bf885"
+ tags               = {
  + "Name" = "Jenkins-SonarQube"
}
+ tags_all           = {
  + "Name" = "Jenkins-SonarQube"
}
+ tenancy             = (known after apply)
+ user_data           = "42caed7d96911f34b2ce87368273c798f00234f2"
+ user_data_base64    = (known after apply)
+ user_data_replace_on_change = false
+ vpc_security_group_ids = [
  + "sg-0f1bfed39db942ac0",
]
+ capacity_reservation_specification (known after apply)

+ cpu_options (known after apply)

+ ebs_block_device (known after apply)

+ enclave_options (known after apply)

+ ephemeral_block_device (known after apply)

+ instance_market_options (known after apply)

+ maintenance_options (known after apply)

+ metadata_options (known after apply)

+ network_interface (known after apply)

```

```

+ private_dns_name_options (known after apply)

+ root_block_device {
    + delete_on_termination = true
    + device_name          = (known after apply)
    + encrypted             = (known after apply)
    + iops                  = (known after apply)
    + kms_key_id            = (known after apply)
    + tags_all               = (known after apply)
    + throughput              = (known after apply)
    + volume_id               = (known after apply)
    + volume_size              = 40
    + volume_type              = (known after apply)
}
}

```

Plan: 1 to add, 0 to change, 0 to destroy.

```
teddyyekyawthu@mac terraform-jenkins % terraform apply --auto-approve
aws_security_group.Jenkins-VM-SG: Refreshing state... [id=sg-0f1bfed39db942ac0]
```

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:

+ create

Terraform will perform the following actions:

```
# aws_instance.web will be created
+ resource "aws_instance" "web" {
    + ami                      = "ami-06c4be2792f419b7b"
    + arn                      = (known after apply)
    + associate_public_ip_address = (known after apply)
    + availability_zone          = (known after apply)
    + cpu_core_count             = (known after apply)
    + cpu_threads_per_core       = (known after apply)
    + disable_api_stop           = (known after apply)
    + disable_api_termination     = (known after apply)
    + ebs_optimized              = (known after apply)
    + enable_primary_ipv6         = (known after apply)
    + get_password_data          = false
    + host_id                    = (known after apply)
    + host_resource_group_arn      = (known after apply)
    + iam_instance_profile        = (known after apply)
    + id                         = (known after apply)
    + instance_initiated_shutdown_behavior = (known after apply)
```

```

+ instance.lifecycle          = (known after apply)
+ instance.state              = (known after apply)
+ instance.type               = "t2.large"
+ ipv6_address_count         = (known after apply)
+ ipv6_addresses              = (known after apply)
+ key_name                    = "cicd"
+ monitoring                  = (known after apply)
+ outpost_arn                 = (known after apply)
+ password_data                = (known after apply)
+ placement_group              = (known after apply)
+ placement_partition_number    = (known after apply)
+ primary_network_interface_id = (known after apply)
+ private_dns                  = (known after apply)
+ private_ip                   = (known after apply)
+ public_dns                   = (known after apply)
+ public_ip                    = (known after apply)
+ secondary_private_ips        = (known after apply)
+ security_groups              = (known after apply)
+ source_dest_check            = true
+ spot_instance_request_id     = (known after apply)
+ subnet_id                    = "subnet-0d01d8b3e2f5bf885"
+ tags                         = {
  + "Name" = "Jenkins-SonarQube"
}
+ tags_all                     = {
  + "Name" = "Jenkins-SonarQube"
}
+ tenancy                       = (known after apply)
+ user_data                     = "42caed7d96911f34b2ce87368273c798f00234f2"
+ user_data_base64              = (known after apply)
+ user_data_replace_on_change   = false
+ vpc_security_group_ids        = [
  + "sg-0f1bfed39db942ac0",
]
+ capacity_reservation_specification (known after apply)
+ cpu_options (known after apply)
+ ebs_block_device (known after apply)
+ enclave_options (known after apply)
+ ephemeral_block_device (known after apply)
+ instance_market_options (known after apply)

```

```

+ maintenance_options (known after apply)

+ metadata_options (known after apply)

+ network_interface (known after apply)

+ private_dns_name_options (known after apply)

+ root_block_device {
    + delete_on_termination = true
    + device_name      = (known after apply)
    + encrypted        = (known after apply)
    + iops              = (known after apply)
    + kms_key_id       = (known after apply)
    + tags_all         = (known after apply)
    + throughput        = (known after apply)
    + volume_id        = (known after apply)
    + volume_size      = 40
    + volume_type      = (known after apply)
}
}

```

Plan: 1 to add, 0 to change, 0 to destroy.

aws_instance.web: Creating...

aws_instance.web: Still creating... [10s elapsed]

aws_instance.web: Creation complete after 13s [id=i-0409e8c0f63924319]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

then, we can go to check and see in our AWS → EC2 instance

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
Jenkins-SonarQube	i-0409e8c0f63924319	Running	t2.large	2/2 checks passed		ap-southeast-1a	ec2-47-129-7-56.ap-so...	47.129.7.56

you can login two ways as below: Option 1 (from aws) , Option 2 (by using .pem key-pair to access)

Option 1:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...
Jenkins-SonarQube	i-0409e8c0f63924319	Running	t2.large	2/2 checks passed		ap-southeast-1a	ec2-47-129-7-56.ap-so...	47.129.7.56
netflix-cluster1-1-node	i-0b4089ca0c90b6d40	Running	t3.medium	3/3 checks passed		ap-southeast-1a	ec2-13-250-61-217.ap-...	12.76.61.7.11
netflix-cluster1-1-node	i-0c6ed1b07abab386f	Running	t3.medium	3/3 checks passed		ap-southeast-1b	ec2-3-0-95-236.ap-sout...	3.0.95.236

```
systemctl status jenkins
ubuntu@ip-10-0-0-5:~$ aws --version
trivy version
eksctl version
docker version
docker ps
```

Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1014-aws x86_64)

- * Documentation: <https://help.ubuntu.com>
- * Management: <https://landscape.canonical.com>
- * Support: <https://ubuntu.com/pro>

System information as of Mon Aug 11 00:59:05 UTC 2025

System load:	0.0	Processes:	119
Usage of /:	27.3% of 38.58GB	Users logged in:	0
Memory usage:	46%	IPv4 address for docker0:	172.17.0.1
Swap usage:	0%	IPv4 address for eth0:	10.0.0.5

* Ubuntu Pro delivers the most comprehensive open source security and compliance features.

<https://ubuntu.com/aws/pro>

Expanded Security Maintenance for Applications is not enabled.

69 updates can be applied immediately.

To see these additional updates run: apt list --upgradable

1 additional security update can be applied with ESM Apps.

Learn more about enabling ESM Apps service at <https://ubuntu.com/esm>

New release '24.04.3 LTS' available.

Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***

Last login: Sun Aug 10 15:31:26 2025 from 3.0.5.37

ubuntu@ip-10-0-0-5:~\$ systemctl status jenkins

● jenkins.service - Jenkins Continuous Integration Server

 Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enable
d)

 Active: active (running) since Sat 2025-08-09 05:35:41 UTC; 1 day 19h ago

 Main PID: 6067 (java)

 Tasks: 67 (limit: 9498)

```
Memory: 1.2G
CPU: 2h 10min 46.139s
CGroup: /system.slice/jenkins.service
    └─ 6067 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war -
        -webroot=/var/cache/jenkins/war --httpPort=8080
            ├─ 117871 /usr/sbin/ "" "" "" "" "" "" "
            └─ 117872 /usr/sbin/ "" "" "" "" "" "
```

```
Aug 11 00:55:55 ip-10-0-0-5 jenkins[6067]:      at PluginClassLoader for kubernetes-cre
dentials-provider//com.cloudbees.jenkins.plugins.kubernetes_credentials_provider.Kuber
netesCredentialProvider$1.doRun(Ku>
Aug 11 00:55:55 ip-10-0-0-5 jenkins[6067]:      at hudson.triggers.SafeTimerTask.run(Sa
feTimerTask.java:92)
Aug 11 00:55:55 ip-10-0-0-5 jenkins[6067]:      at jenkins.security.ImpersonatingSchedul
edExecutorService$1.run(ImpersonatingScheduledExecutorService.java:67)
Aug 11 00:55:55 ip-10-0-0-5 jenkins[6067]:      at java.base/java.util.concurrent.Executor
s$RunnableAdapter.call(Executors.java:572)
Aug 11 00:55:55 ip-10-0-0-5 jenkins[6067]:      at java.base/java.util.concurrent.FutureTa
sk.run(FutureTask.java:317)
Aug 11 00:55:55 ip-10-0-0-5 jenkins[6067]:      at java.base/java.util.concurrent.Schedul
edThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:30
4)
Aug 11 00:55:55 ip-10-0-0-5 jenkins[6067]:      at java.base/java.util.concurrent.ThreadP
oolExecutor.runWorker(ThreadPoolExecutor.java:1144)
Aug 11 00:55:55 ip-10-0-0-5 jenkins[6067]:      at java.base/java.util.concurrent.ThreadP
oolExecutor$Worker.run(ThreadPoolExecutor.java:642)
Aug 11 00:55:55 ip-10-0-0-5 jenkins[6067]:      at java.base/java.lang.Thread.run(Threa
d.java:1583)
Aug 11 00:55:55 ip-10-0-0-5 jenkins[6067]: 2025-08-11 00:55:55.340+0000 [id=50]     I
NFO c.c.j.p.k.KubernetesCredentialProvider#reconnectLater: Attempting to reco
```

```
ubuntu@ip-10-0-0-5:~$ aws --version
aws-cli/2.28.6 Python/3.13.4 Linux/6.5.0-1014-aws exe/x86_64.ubuntu.22
ubuntu@ip-10-0-0-5:~$ trivy version
Version: 0.65.0
ubuntu@ip-10-0-0-5:~$ eksctl version
0.212.0nnect Kubernetes client in 5 mins
```

```
ubuntu@ip-10-0-0-5:~$ docker version
Client:
Version: 27.5.1
API version: 1.47
Go version: go1.22.2
Git commit: 27.5.1-0ubuntu3~22.04.2
Built: Mon Jun 2 12:18:38 2025
OS/Arch: linux/amd64
Context: default
```

```

Server:
Engine:
Version: 27.5.1
API version: 1.47 (minimum version 1.24)
Go version: go1.22.2
Git commit: 27.5.1-0ubuntu3~22.04.2
Built: Mon Jun 2 12:18:38 2025
OS/Arch: linux/amd64
Experimental: false
containerd:
Version: 1.7.27
GitCommit:
runc:
Version: 1.2.5-0ubuntu1~22.04.1
GitCommit:
docker-init:
Version: 0.19.0
GitCommit:

ubuntu@ip-10-0-0-5:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
TS
8c50e65fcd85 sonarqube:its-community "/opt/sonarqube/dock..." 43 hours ago Up 43 hours 0.0.0.0:9000→9000/tcp, :::9000→9000/tcp sonar

```

Option 2: (From our PC by using ssh with key-pairs)

my key-pair ".pem" is situated in Downloads file, so needs to go that path

```
teddyyekyawthu@mac Downloads % ssh -i cicd.pem ubuntu@47.129.7.56
```

```

ubuntu@ip-10-0-0-5:~$ Read from remote host 47.129.7.56: Connection reset by peer
Connection to 47.129.7.56 closed.
client_loop: send disconnect: Broken pipe
teddyyekyawthu@mac Downloads % ssh -i cicd.pem ubuntu@47.129.7.56
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-1014-aws x86_64)


```

- * Documentation: <https://help.ubuntu.com>
- * Management: <https://landscape.canonical.com>
- * Support: <https://ubuntu.com/pro>

System information as of Mon Aug 11 01:10:16 UTC 2025

```

System load: 0.00390625 Processes: 121
Usage of /: 27.3% of 38.58GB Users logged in: 1
Memory usage: 46% IPv4 address for docker0: 172.17.0.1

```

Swap usage: 0% IPv4 address for eth0: 10.0.0.5

* Ubuntu Pro delivers the most comprehensive open source security and compliance features.

<https://ubuntu.com/aws/pro>

Expanded Security Maintenance for Applications is not enabled.

69 updates can be applied immediately.

To see these additional updates run: apt list --upgradable

1 additional security update can be applied with ESM Apps.

Learn more about enabling ESM Apps service at <https://ubuntu.com/esm>

New release '24.04.3 LTS' available.

Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***

Last login: Mon Aug 11 00:59:06 2025 from 3.0.5.36

ubuntu@ip-10-0-0-5:~\$

Access test to Jenkins & SonarQube from web-page

<http://47.129.7.56:8080/> (IP will be changed according to our public IP)

<http://47.129.7.56:9000/> (IP will be changed according to our public IP)

<http://47.129.7.56:8080/> (Jenkins web)

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

Continue

then - default page is there - need to copy this path and find the password from our Jenkins EC2 server

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

```
ubuntu@ip-10-0-0-5:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
e3723da9d8cb49ddac260181b46620d7
```

Jenkins Setup

1. Access Jenkins at <http://<public-ip>:8080> and install suggested plugins.
2. Install the following Jenkins plugins:

- [Eclipse Temurin installer](#)
- [SonarQube Scanner](#)
- [Sonar Quality Gates](#)
- [NodeJS](#)
- [Docker, Docker Commons, Docker Pipeline, Docker API](#)
- [Kubernetes, Kubernetes Credentials, Kubernetes Client API, Kubernetes CLI](#)
- [Pipeline: Stage View](#)
- [Blue Ocean](#)
- [AWS Credentials](#)
- [Prometheus metrics](#)

- OWASP Dependency-Check

install plug-in sample

Install	Name	Released	Health
<input type="checkbox"/>	PAM Authentication 1.12 Security	5 mo 9 days ago	97
<input type="checkbox"/>	JSch dependency 0.2.16-95.v34e6cb_55fa_b_78 Library plugins (for use by other plugins) Miscellaneous	5 mo 16 days ago	100
<input type="checkbox"/>	Javadoc 354-veela_6600_4990 Built Tools	9 days 3 hr ago	100
<input type="checkbox"/>	Maven Integration 3.26 Build Tools	3 mo 16 days ago	100

Obtain an NVD API Key

- **Request an API Key:**
 - Go to the [NVD API Key Request Page](#).
 - Fill out the form with your details (e.g., name, email, organization).
 - You'll receive an API key via email (usually within minutes).
- **Add API Key to Jenkins:**
 - In Jenkins, go to **Manage Jenkins > Manage Plugins > Credentials > System > Global Credentials**

T	P	Store	Domain	ID	Name
<input type="checkbox"/>		System	(global)	SonarQube-Token	SonarQube-Token
<input type="checkbox"/>		System	(global)	dockerhub	teddy2012*****
<input type="checkbox"/>		System	(global)	tmdb-api-key	tmdb-api-key
<input type="checkbox"/>		System	(global)	kubernetes	kubernetes
<input type="checkbox"/>		System	(global)	aws-secret	AKIA3IS8VYU33ASAHBHP

Stores scoped to Jenkins

P	Store	Domains	
<input type="checkbox"/>		System	(global)
<input type="checkbox"/>		Kubernetes	(global)

- Click **Add Credentials**.
- Set:
 - **Kind:** Secret text
 - **Secret:** Paste the NVD API key
 - **ID:** nvd-api-key
 - **Description:** NVD API Key for Dependency-Check
- Save the credential.

before configuring tools , need to check with below Jenkins (Can be referred the video CICD Pipeline with Jenkins)

```
pipeline {  
    agent any  
  
    parameters {  
        string(name: 'MAJOR_VERSION', defaultValue: '1', description: 'Major version  
for the Docker image tag (e.g., 1 for v1.x)')  
    }  
  
    tools {  
        jdk 'jdk21'  
        nodejs 'node20'  
    }  
  
    environment {  
        SCANNER_HOME = tool 'sonarqube-scanner'  
        TRIVY_HOME = '/usr/bin'  
        REPO_URL = 'https://github.com/TEDDY201289/Netflix-test.git'  
        REPO_BRANCH = 'main'  
        DOCKER_IMAGE_NAME = 'teddy2012/netflix-bubudu'  
        SONAR_PROJECT_NAME = 'Netflix'  
        SONAR_PROJECT_KEY = 'Netflix'  
        DOCKER_CREDENTIALS_ID = 'dockerhub'  
        SONAR_CREDENTIALS_ID = 'SonarQube-Token'  
        KUBERNETES_CREDENTIALS_ID = 'kubernetes'  
        SONAR_SERVER = 'SonarQube-Server'  
        DOCKER_TOOL_NAME = 'docker'  
        TRIVY_FS_REPORT = 'trivyfs.txt'  
        TRIVY_IMAGE_REPORT = 'trivyimage.txt'  
        K8S_NAMESPACE = 'default'  
        APP_NAME = 'netflix'  
    }  
  
    stages {  
        stage('Clean Workspace') {  
            steps {  
                cleanWs()  
            }  
            post {  
                always {  
                    echo "Workspace cleaned successfully"  
                }  
            }  
        }  
        stage('Checkout from Git') {  
    }
```

```

steps {
    git branch: "${REPO_BRANCH}", url: "${REPO_URL}", credentialsId: 'gith
ub-credentials'
}
}
stage('SonarQube Analysis') {
    steps {
        withSonarQubeEnv(credentialsId: "${SONAR_CREDENTIALS_ID}", installa
tionName: "${SONAR_SERVER}") {
            sh """
                ${SCANNER_HOME}/bin/sonar-scanner \
                    -Dsonar.projectName=${SONAR_PROJECT_NAME} \
                    -Dsonar.projectKey=${SONAR_PROJECT_KEY}
            """
        }
    }
}
stage('Quality Gate') {
    steps {
        script {
            waitForQualityGate abortPipeline: false, credentialsId: "${SONAR_CRE
DENTIALS_ID}"
        }
    }
}
stage('Install Dependencies') {
    steps {
        script {
            def nodeHome = tool name: 'node20', type: 'nodejs'
            env.PATH = "${nodeHome}/bin:${env.PATH}"
            sh "node --version"
            sh "npm --version"
            sh "npm install"
        }
    }
    post {
        failure {
            echo "Failed to install dependencies. Check Node.js setup or package.
json."
        }
    }
}
// stage('OWASP Dependency Check') {
//     steps {
//         withCredentials([string(credentialsId: 'nvd-api-key', variable: 'NVD_API
_KEY')])
//             dependencyCheck additionalArguments: "--scan ./ --disableYarnAu

```

```

dit --disableNodeAudit --nvdApiKey ${NVD_API_KEY}", odcInstallation: 'DP-Chec
k'
        //      dependencyCheckPublisher pattern: '**/dependency-check-report.
xml'
        //
        //
        //
    stage('Trivy FS Scan') {
        steps {
            script {
                sh "${TRIVY_HOME}/trivy fs . > ${TRIVY_FS_REPORT}"
            }
        }
        post {
            always {
                archiveArtifacts artifacts: "${TRIVY_FS_REPORT}", allowEmptyArchive:
true
            }
        }
    }
    stage('Set Version') {
        steps {
            script {
                def buildNumber = env.BUILD_NUMBER ?: '0'
                env.IMAGE_TAG = "v${params.MAJOR_VERSION}.${buildNumber}"
                echo "Docker image will be tagged as: ${DOCKER_IMAGE_NAME}:${e
nv.IMAGE_TAG}"
            }
        }
    }
    stage('Docker Build & Push') {
        steps {
            withCredentials([string(credentialsId: 'tmdb-api-key', variable: 'TMDB_A
PI_KEY')]) {
                script {
                    withDockerRegistry(credentialsId: "${DOCKER_CREDENTIALS_ID}",
toolName: "${DOCKER_TOOL_NAME}") {
                        sh "docker build -t ${APP_NAME} --build-arg TMDB_V3_API_KEY
=${TMDB_API_KEY} ."
                        sh "docker tag ${APP_NAME} ${DOCKER_IMAGE_NAME}:${env.I
MAGE_TAG}"
                        sh "docker push ${DOCKER_IMAGE_NAME}:${env.IMAGE_TAG}"
                    }
                }
            }
        }
        post {
    
```

```

        always {
            sh "docker rmi ${APP_NAME} ${DOCKER_IMAGE_NAME}:${env.IMAGE_TAG} || true"
        }
    }
}

stage('Trivy Image Scan') {
    steps {
        script {
            sh "${TRIVY_HOME}/trivy image ${DOCKER_IMAGE_NAME}:${env.IMAGE_TAG} > ${TRIVY_IMAGE_REPORT}"
        }
    }
    post {
        always {
            archiveArtifacts artifacts: "${TRIVY_IMAGE_REPORT}", allowEmptyArchive: true
        }
    }
}
stage('Deploy to Kubernetes') {
    steps {
        withCredentials([
            $class: 'AmazonWebServicesCredentialsBinding',
            credentialsId: 'aws-secret'
        ]) {
            script {
                dir('Kubernetes') {
                    withKubeConfig(
                        credentialsId: "${KUBERNETES_CREDENTIALS_ID}",
                        serverUrl: 'https://ABCD027AC4657D748C746D37AC3C7DFE.gcr7.ap-southeast-1.eks.amazonaws.com', // Replace with actual EKS endpoint
                        namespace: "${K8S_NAMESPACE}"
                    ) {
                        sh 'kubectl version'
                        sh "sed -i 's|image: ${DOCKER_IMAGE_NAME}:.*|image: ${DOCKER_IMAGE_NAME}:${env.IMAGE_TAG}|' deployment.yml"
                        sh 'kubectl apply -f deployment.yml'
                        sh 'kubectl apply -f service.yml'
                    }
                }
            }
        }
    }
}
post {

```

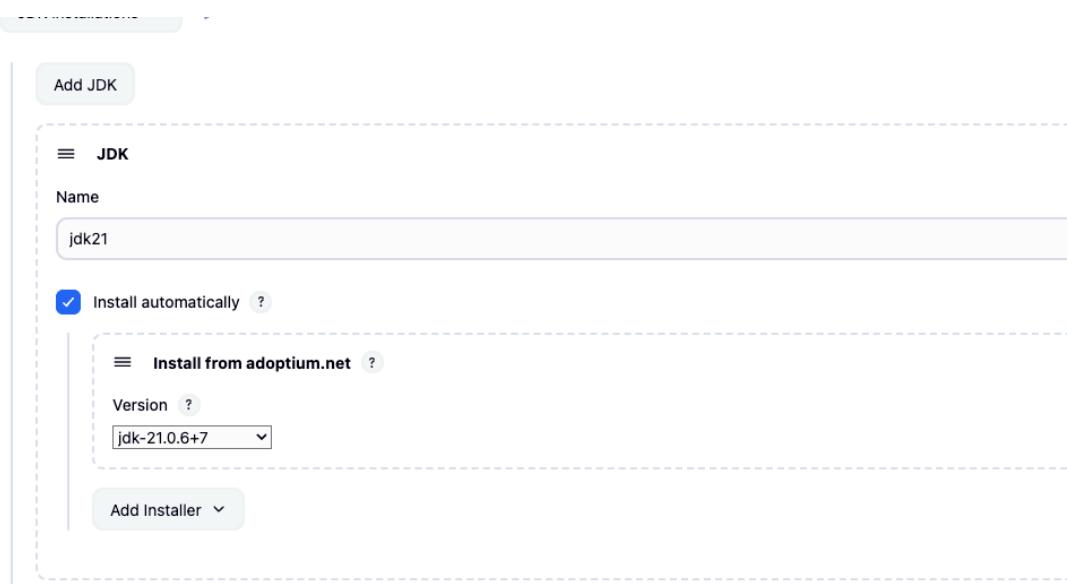
```

        always {
            emailext attachLog: true,
                subject: "${currentBuild.result}",
                body: "Project: ${env.JOB_NAME}<br/>" +
                    "Build Number: ${env.BUILD_NUMBER}<br/>" +
                    "URL: ${env.BUILD_URL}<br/>",
                to: 'hlaingminpaing.ygn@gmail.com', // Update with your email
                attachmentsPattern: 'trivyfs.txt,trivyimage.txt'
            }
        }
    }
}

```

3. Configure tools in Jenkins:

sample screenshots:



- **NodeJS:** Add NodeJS 20 installation (`node20`).
- **JDK:** Add JDK 21 installation (`jdk21`).
- **Docker:** Configure Docker installation (`docker`).
- **SonarQube Scanner:** Add SonarQube Scanner installation (`sonarqube-scanner`).
- **Dependency-Check:** Add Dependency-Check installation (`DP-Check`), install automatically from GitHub.

4. Generate a SonarQube token:

- Log in to SonarQube at <http://<public-ip>:9000>.
- Navigate to **User > My Account > Security > Generate Tokens**.
- Copy the token and add it as a secret in Jenkins (Credentials ID: `SonarQube-Token`).

5. Add credentials in Jenkins:

- **Docker Hub:** Add Docker Hub credentials (Credentials ID: `dockerhub`).
- **GitHub:** Add GitHub credentials (Credentials ID: `github-credentials`).
- **AWS:** Add AWS credentials (Credentials ID: `aws-secret`).
- **Kubernetes:** Copy `/home/ubuntu/.kube/config` and save as a secret (Credentials ID: `kubernetes`).
- **TMDB API Key:** Add the TMDB API key as a secret text (Credentials ID: `tmdb-api-key`).

6. Configure environment variables in Jenkins for SonarQube and Docker.

- **SonarQube servers:** SonarQube server configuration (URL: `http://<public-ip>:9000`, Name: `SonarQube-Server`)

7. Set up a webhook in SonarQube:

- Navigate to **Administration > Configuration > Webhooks**.
- Add a webhook with the URL: `http://<jenkins-public-ip>:8080/sonarqube-webhook/`.

8. Create a SonarQube project and quality gate:

- In SonarQube, create a new project (Name: `Netflix`, Key: `Netflix`).
- Configure a quality gate with desired metrics (e.g., code coverage, vulnerabilities).

`SonarQube scanner & Sonar Qube`

The screenshot shows the Jenkins 'Tools' configuration page. At the top, there are tabs for 'Jenkins', 'Manage Jenkins', and 'Tools'. Below these tabs, there are sections for different build tools:

- SonarQube Scanner installations:** A dropdown menu labeled 'SonarQube Scanner installations' with a 'Edited' status indicator.
- Ant installations:** A section with a 'Add Ant' button.
- Maven installations:** A section with a 'Add Maven' button.
- NodeJS installations:** A dropdown menu labeled 'NodeJS installations' with a 'Edited' status indicator.
- Docker installations:** A dropdown menu labeled 'Docker installations' with a 'Edited' status indicator.

 Jenkins / Manage Jenkins / Tools

SonarQube Scanner installations ^  Edited

Add SonarQube Scanner

SonarQube Scanner

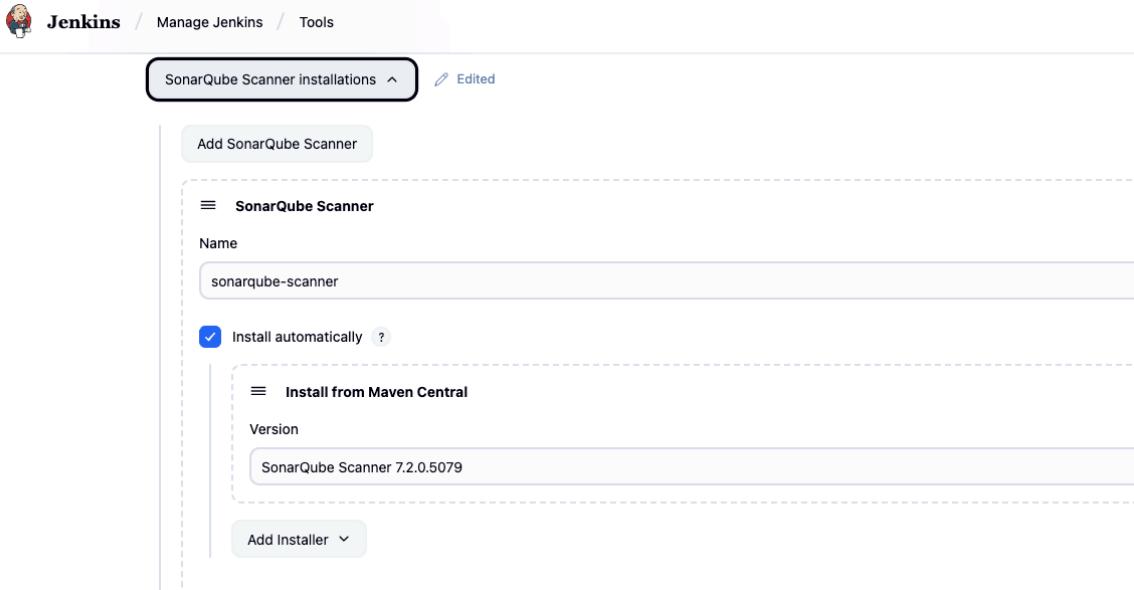
Name: sonarqube-scanner

Install automatically ?

Install from Maven Central

Version: SonarQube Scanner 7.2.0.5079

Add Installer ▾



NodeJS

 Jenkins / Manage Jenkins / Tools

NodeJS installations ^  Edited

Add NodeJS

NodeJS

Name: node20

Install automatically ?

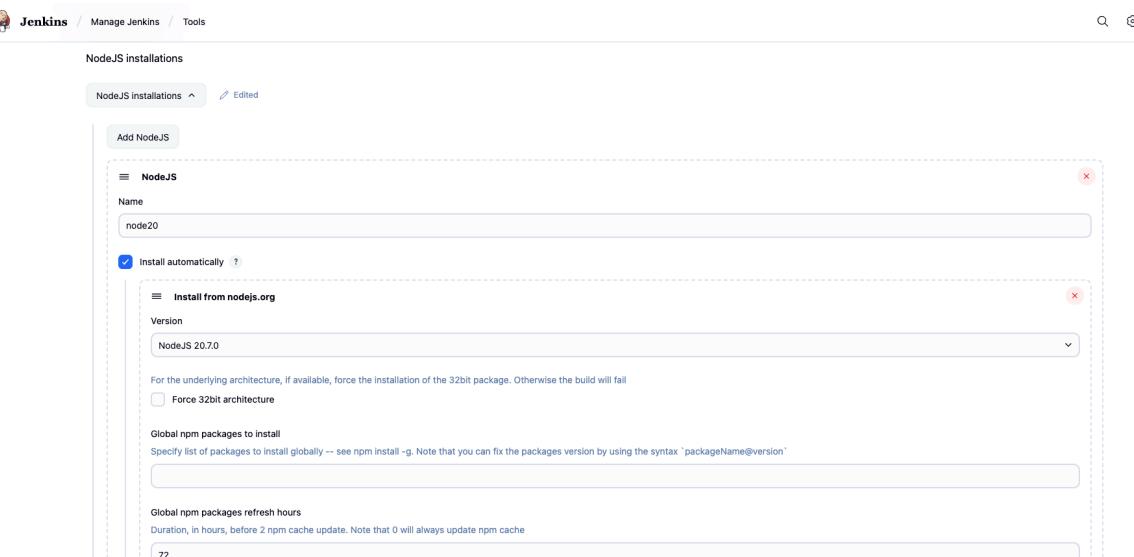
Install from nodejs.org

Version: NodeJS 20.7.0

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



Docker

Docker installations

Docker installations ^ Edited

Add Docker

Docker

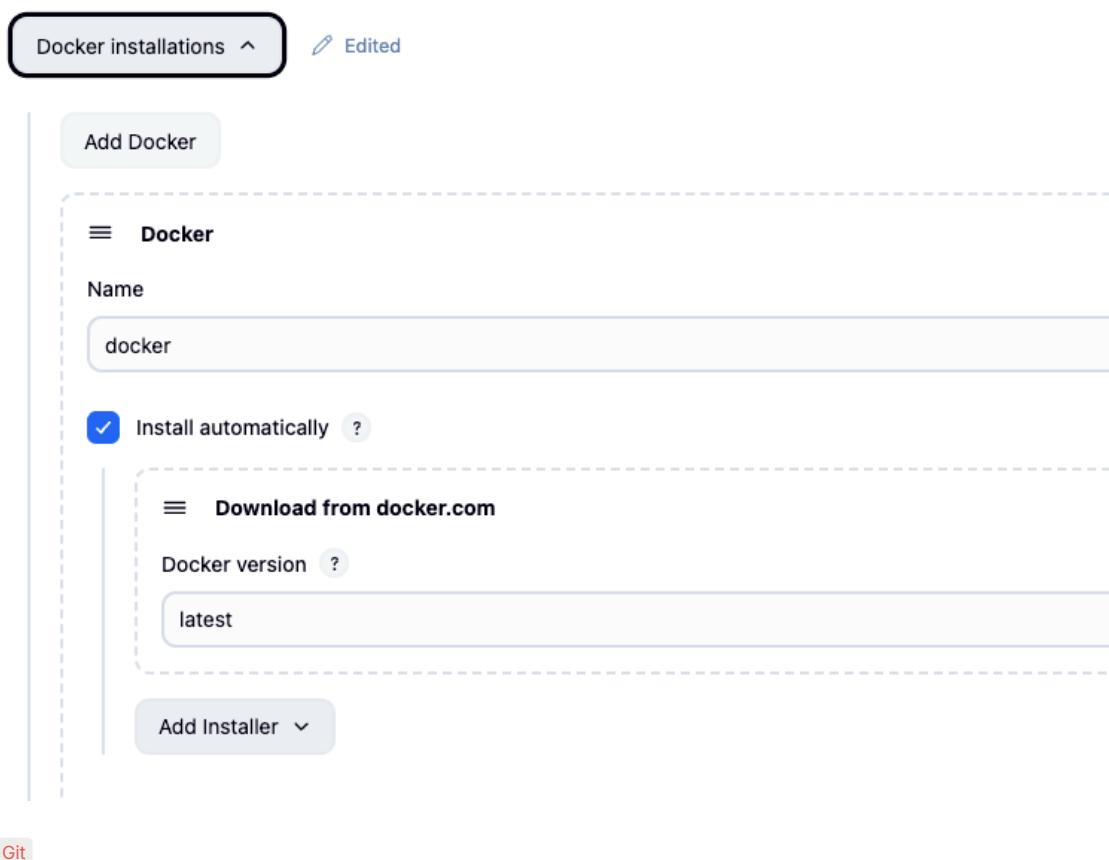
Name: docker

Install automatically ?

Download from docker.com

Docker version ?
latest

Add Installer ▾



Git

Docker installations

Docker installations ^ Edited

Add Docker

Docker

Name: docker

Install automatically ?

Download from docker.com

Docker version ?
latest

Add Installer ▾



Git installations

Git

Name
Default

Path to Git executable ?
git

Install automatically ?

Add Git ▾

Add Credentials according to the Jenkins file (Docker, Github, SonarQube-token, Tmdb-api-key, Kubernetes and AWS)

Credentials

T	P	Store	Domain	ID	Name
		System	(global)	SonarQube-Token	SonarQube-Token
		System	(global)	dockerhub	teddy2012/*****
		System	(global)	tmdb-api-key	tmdb-api-key
		System	(global)	kubernetes	kubernetes
		System	(global)	aws-secret	AKIA3ISBVYU33ASAHBHP

sample for docker

Update credentials

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

Username ?
teddy2012

Treat username as secret ?

Password ?
 Concealed Change Password

ID ?
dockerhub

Description ?

Save

 Jenkins / Manage Jenkins / Credentials / System / Global credentials (unrestricted) / tmdb-api-key

[Update](#) [Delete](#) [Move](#)

Update credentials

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

Secret [Concealed](#) [Change Password](#)

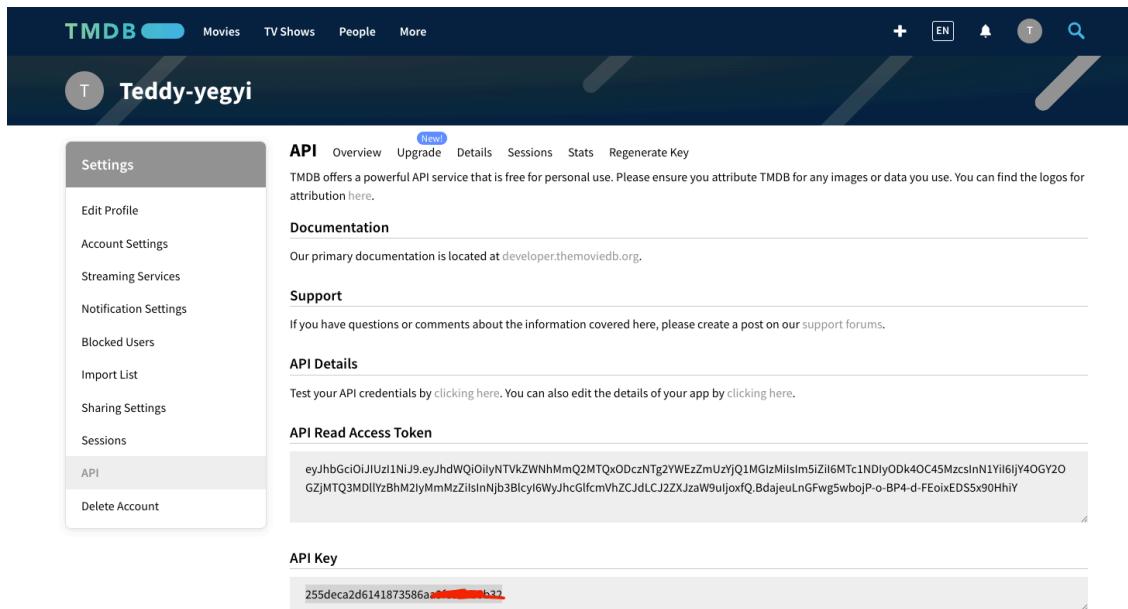
ID ? tmdb-api-key

Description ?

[Save](#)

TMTDB (account should be created and go to this setting)

<https://www.themoviedb.org/settings/api>



The screenshot shows the TMDB API settings page. On the left, there's a sidebar with options like Edit Profile, Account Settings, Streaming Services, Notification Settings, Blocked Users, Import List, Sharing Settings, Sessions, and API. The API option is currently selected. The main content area has tabs for API Overview, Upgrade, Details, Sessions, Stats, and Regenerate Key. Below the tabs, it says "TMDB offers a powerful API service that is free for personal use. Please ensure you attribute TMDB for any images or data you use. You can find the logos for attribution [here](#)". There are sections for Documentation (link to developer.themoviedb.org), Support (link to support forums), and API Details (link to test credentials). Under API Read Access Token, there's a long API key displayed: eyJhbGciOiJIUzI1NiJ9eyJhdWQiOiIyNTVkJzWnhMmQ2MTQxODczNTg2YWEzZmUzYjQ1MGlxMlsIm5iZi6MtC1NDlyODk4OC45MzsInN1Yi6IjY4OGY2OGZjMTQ3MDIyBhM2lyMmMzZlslNjb3Bicy6WyJhcGlfcmVhZCldCJ2ZXJzaW9uIjoxQ.BdajeulnGFwg5wbojP-o-BP4-d-FeoiEDS5x90HhiY. Below this is an API Key field containing the value 255deca2d6141873586a... (redacted).

Go to Jenkins >> Managed Jenkins >> System

Jenkins Location

Jenkins URL ? <http://47.129.756:8080/>

System Admin e-mail address ? address not configured yet <nobody@nowhere>

SonarQube installations

List of SonarQube installations

Name	SonarQube-Server	X
Server URL	Default is http://localhost:9000 http://47.129.756:9000/	
Server authentication token	SonarQube authentication token. Mandatory when anonymous access is disabled. - none -	

Jenkins / Manage Jenkins / System

GitHub Pull Request Builder

GitHub Auth

GitHub Server API URL ?
https://api.github.com

Jenkins URL override ?

Shared secret ?
Concealed

Credentials ?
SonarQube-Token

+ Add

Then go to access <http://47.129.7.56:9000/admin/webhooks> (IP will be changed)

Default username and password - admin / admin

Not Secure 47.129.7.56:9000/admin/users

Service Now Daily routine AWS Jira Staging DOMO accessories Dashboard and m... Copilot | Microsoft... ChatGPT Google Gemini DeepSeek - Into th... self learning Staging DOMO study

sonarcube Projects Issues Rules Quality Profiles Quality Gates Administration

Search for projects... A

Administration Configuration Security Projects System Marketplace

Users Create and administer individual users. Create User

Search by login or name...

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

1 of 1 shown

Tokens of Administrator

Generate Tokens

Name	Expires in
Enter Token Name	30 days
<input type="button" value="Generate"/>	

Name	Type	Project	Last use	Created	Expiration
jenkins	User		1 day ago	August 9, 2025	-

Create webhook to authenticate with Jenkins

The screenshot shows the SonarQube administration interface under the 'Webhooks' section. A single webhook named 'jenkin' is listed with the URL 'http://47.129.7.56:8080/sonarqube-webhook/'. It has no secret and was last delivered on August 9, 2025 at 9:01 PM. There is a 'Create' button in the top right corner.

Name	URL	Has secret?	Last delivery	Actions
jenkin	http://47.129.7.56:8080/sonarqube-webhook/	No	✓ August 9, 2025 at 9:01 PM	

Create Webhook

All fields marked with * are required

Name *

jenkins-webhook

URL *

http://htt18.143.170.247:8080/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.

Create **Cancel**

To do Quality gate - Go to Manual code >> Create Project

Create a project

```
netflix-test > Jenkinsfile
1   pipeline {
2     /
3     tools {
4       jdk 'jdk21'
5       nodejs 'node20'
6     }
7
8     environment {
9       SCANNER_HOME = tool 'sonarqube-scanner'
10      TRIVY_HOME = '/usr/bin'
11      REPO_URL = 'https://github.com/TEDDY201289/Netflix-test.git'
12      REPO_BRANCH = 'main'
13      DOCKER_IMAGE_NAME = 'teddy2012/netflix-bubudu'
14      SONAR_PROJECT_NAME = 'Netflix'
15      SONAR_PROJECT_KEY = 'Netflix'
16      DOCKER_CREDENTIALS_ID = 'dockerhub'
17      SONAR_CREDENTIALS_ID = 'SonarQube-Token'
18      KUBERNETES_CREDENTIALS_ID = 'kubernetes'
19      SONAR_SERVER = 'SonarQube-Server'
20      DOCKER_TOOL_NAME = 'docker'
21      TRIVY_FS_REPORT = 'trivyfs.txt'
22      TRIVY_IMAGE_REPORT = 'trivyimage.txt'
23      K8S_NAMESPACE = 'default'
24      APP_NAME = 'netflix'
```

Then we go to GitHub account:

Create Repo from UI

New repository

Create a new repository [Preview](#) [Switch back to classic experience](#)

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#). Required fields are marked with an asterisk (*).

1 General

Owner * Repository name *

TEDDY201289 /

Great repository names are short and memorable. How about [vigilant-bassoon](#)?

Description

0 / 350 characters

2 Configuration

Choose visibility * Public

Add README Off READMEs can be used as longer descriptions. [About READMEs](#)

Add .gitignore No .gitignore .gitignore tells git which files not to track. [About ignoring files](#)

Add license No license Licenses explain how others can use your code. [About licenses](#)

Create repository

TEDDY201289 / Netflix-test

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Netflix-test Public

main 1 Branch 0 Tags Go to file Add file Code

TEDDY201289 addurlaws 645b7e3 · 2 days ago 6 Commits

Kubernetes	dockernamechange	2 days ago
grafana	move	2 days ago
k8s-sa	move	2 days ago
public	move	2 days ago
src	move	2 days ago
terraform-jenkins	move	2 days ago
terraform-monitoring	move	2 days ago
.DS_Store	move	2 days ago

About

test

- Readme
- Activity
- 0 stars
- 0 watching
- 0 forks

Releases

No releases published [Create a new release](#)

Packages

No packages published [Publish your first package](#)

<https://github.com/TEDDY201289/Netflix-test/settings/hooks/562924452?tab=settings>

Add webhook with payload URL (jenkins) to get automated triggering for CI between GitHub and Jenkins

The screenshot shows the GitHub Settings interface for a repository named "Netflix-test". The left sidebar has a "Webhooks" section selected. The main area is titled "Webhooks / Manage webhook". It includes fields for "Payload URL" (set to "http://47.129.7.56:8080/github-webhook/"), "Content type" (set to "application/json"), and a "Secret" field. Under "SSL verification", the "Enable SSL verification" option is selected. In the "Which events would you like to trigger this webhook?" section, the "Just the push event" option is selected. A "Active" checkbox is checked. At the bottom are "Update webhook" and "Delete webhook" buttons.

Then , Go to Docker hub, and create new repo

The screenshot shows the Docker Hub "My Hub" page for the user "teddy2012". The left sidebar has a "Repositories" section selected. The main area shows a table of repositories. One repository, "teddy2012/netflix-bubudu", is listed with details: Last Pushed (1 day ago), Contains (IMAGE), Visibility (Public), and Scout (Inactive). A "Create a repository" button is visible at the top right.

Create Pipeline

 Jenkins / All / New Item

New Item

Enter an item name

Select an item type

-  **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.
-  **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**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
-  **Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
-  **Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.
-  **Organization Folder**
Creates a set of multibranch project subfolders by scanning for repositories.

If you want to create a new item from other existing, you can use this option:

 Jenkins / Test / Configuration

Configure

-  General
-  **Triggers**
-  Pipeline
-  Advanced

Build after other projects are built ?
 Build periodically ?
 GitHub Branches
 GitHub Pull Request Builder
 GitHub Pull Requests ?
 GitHub hook trigger for GITScm polling ?
 Poll SCM ?
 Schedule ?

No schedules so will only run due to SCM changes if triggered by a post-commit hook

Ignore post-commit hooks ?
 Trigger builds remotely (e.g., from scripts) ?

 Jenkins / Test / Configuration

Configure

-  General
-  Triggers
-  **Pipeline**
-  Advanced

Pipeline script from SCM

SCM ?

Repositories ?

Repository URL ?

Credentials ? - none -

+ Add

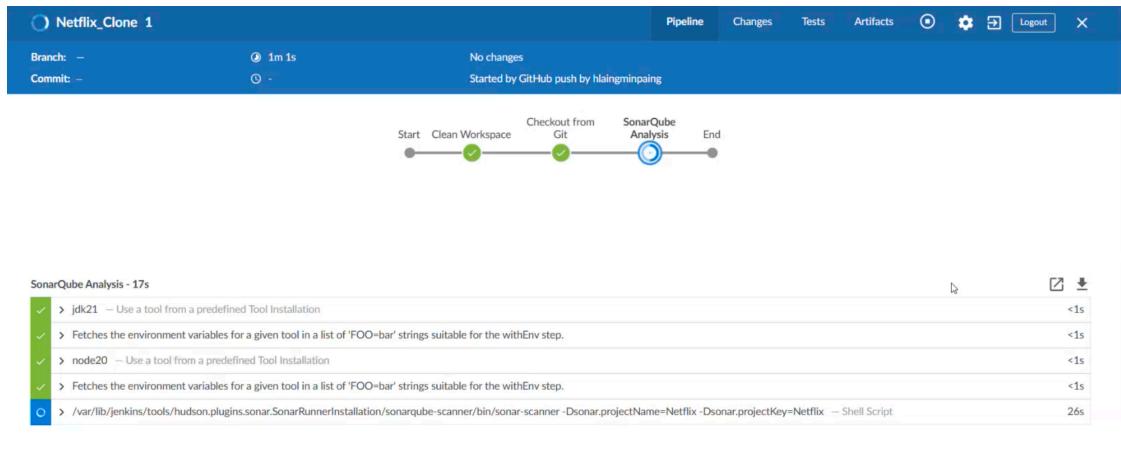
Advanced

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

Add Branch



Now, we gonna to set up for EKS cluster as below:

Install eksctl

```
curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp
cd /tmp
sudo mv /tmp/eksctl /bin
eksctl version
```

Create Cluster

```
eksctl create cluster --name netflix-cluster \
--region us-east-1 \
--node-type t3.large \
--nodes 3
--profile teddy-kube
```

Wait for 30 mins minimum then Cluster is up

Verify Cluster

```
kubectl get nodes
kubectl get svc
```

EKS

EC2

The screenshot shows the AWS EC2 Instances page. The left sidebar has 'EC2' selected under 'Instances'. The main table lists three instances:

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elast.
Jenkins-SonarCube	i-0409e8c0f63924319	Running	t2.large	2/2 checks passed	View alarms	ap-southeast-1a	ec2-47-129-7-5.ap...so...	47.129.7.56	-
eksctl-cluster1-n1-Node	i-0b4089ca0c90b6d40	Running	t3.medium	3/3 checks passed	View alarms	ap-southeast-1a	ec2-13-250-61-217.ap...	13.250.61.217	-
netflix-cluster1-nginx-1-Node	i-06ec6d5b07aba386f	Running	t3.medium	3/3 checks passed	View alarms	ap-southeast-1b	ec2-3-0-95-236.ap.sou...	3.0.95.236	-

VPC

The screenshot shows the AWS VPC Your VPCs page. The left sidebar has 'VPC dashboard' selected under 'Virtual private cloud'. The main table lists three VPCs:

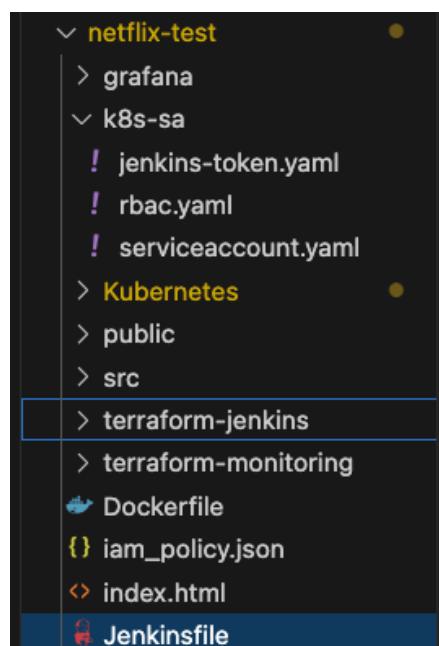
Name	VPC ID	State	Block Public Access	IPv4 CIDR	IPv6 ...	DHCP option set	Main route tabl...
cldc-vpc	vpc-00d7e4891080b872f	Available	Off	10.0.0.0/24	-	dopt-0a038608db4050...	rtb-0cc0f0961ee
-	vpc-0198c7df3c9af437e	Available	Off	172.31.0.0/16	-	dopt-0a038608db4050...	rtb-047e94b90
eksctl-netflix-cluster1-cluster/VPC	vpc-01a9795acee1e0d39	Available	Off	192.168.0.0/16	-	dopt-0a038608db4050...	rtb-032c68406

Subnet Mask

The screenshot shows the AWS VPC Subnets page. The left sidebar has 'VPC dashboard' selected under 'Virtual private cloud'. The main table lists subnets across various VPCs:

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
cldc-subnet-private-1-ap-southeast-1a	subnet-00361ee8cd0fed330	Available	vpc-00d7e4891080b872f cldc-vpc	Off	10.0.0.128/
cldc-subnet-public-1-ap-southeast-1a	subnet-0d1db3e2f5bf885	Available	vpc-00d7e4891080b872f cldc-vpc	Off	10.0.0.0/28
eksctl-netflix-cluster1-cluster/SubnetPrivateAPSOUTHEAST1A	subnet-0b6733f91bb230676	Available	vpc-01a9795acee1e0d39 eksctl-netflix-cluster1-cluster...	Off	192.168.96.
eksctl-netflix-cluster1-cluster/SubnetPrivateAPSOUTHEAST1B	subnet-03abe052631aca6f	Available	vpc-01a9795acee1e0d39 eksctl-netflix-cluster1-cluster...	Off	192.168.161
eksctl-netflix-cluster1-cluster/SubnetPrivateAPSOUTHEAST1C	subnet-0167eff55469352zd	Available	vpc-01a9795acee1e0d39 eksctl-netflix-cluster1-cluster...	Off	192.168.121
eksctl-netflix-cluster1-cluster/SubnetPublicAPSOUTHEAST1A	subnet-0c28a2a23ad05679	Available	vpc-01a9795acee1e0d39 eksctl-netflix-cluster1-cluster...	Off	192.168.0.0
eksctl-netflix-cluster1-cluster/SubnetPublicAPSOUTHEAST1B	subnet-0b9b5e6090724494e	Available	vpc-01a9795acee1e0d39 eksctl-netflix-cluster1-cluster...	Off	192.168.64.
eksctl-netflix-cluster1-cluster/SubnetPublicAPSOUTHEAST1C	subnet-092c161b277f662b7	Available	vpc-01a9795acee1e0d39 eksctl-netflix-cluster1-cluster...	Off	192.168.32.

Then Go to check at Kubernetes >> k8s-sa



jenkins-sa.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: jenkins
  namespace: default
```

Jenkins-Token.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: jenkins-token
  namespace: default
  annotations:
    kubernetes.io/service-account.name: jenkins
type: kubernetes.io/service-account-token
```

```
teddyyekyawthu@mac .aws % kubectl get secret
NAME      TYPE          DATA   AGE
jenkins-token  kubernetes.io/service-account-token  3     37h
```

Retrieve token

```
kubectl apply -f jenkins-token.yaml
kubectl get secret jenkins-token -n default -o jsonpath='{.data.token}' | base64 --decode

teddyyekyawthu@mac .aws % kubectl get secret jenkins-token -n default -o jsonpath='.
data.token' | base64 --decode
eyJhbGciOiJSUzI1NiIiLmtpZCI6IjNpZW1hUEYyUkpqUII1MTVTbDhHSVRhSE5VQ2NIU1VFR0
I1UU8zdjVHQ1UiFQ.eyJpc3MiOiJrdWJlcmt5IdGVzL3NlcnPZpY2VhY2NvdW50liwia3ViZXJuZ
XRlc5pb9zZXJ2aWNiYWNjb3VudC9uYW1lc3BhY2UiOjKZWZhdWx0liwia3ViZXJuZXRLc
y5pb9zZXJ2aWNiYWNjb3VudC9zZWNyZXQubmFtZSI6ImplbmtbnnMtdG9rZW4iLCJrdW
Jlc5IdGVzLmlvL3NlcnPZpY2VhY2NvdW50L3NlcnPZpY2UtYWNjb3VudC5uYW1ljoiamVua2I
ucylslmt1YmVybmV0ZXMuaw8vc2VydmljZWfjY291bnQvc2VydmljZS1hY2NvdW50LnVpZ
CI6ImQwYTQwNTJmLTImOGQtNDAwMS1iMjBkLTrkNTI4M2JmN2U3ZilsInN1Yi6lnN5c3RI
bTpzZXJ2aWNiYWNjb3VudDpkZWZhdWx0OmplbmtbnnMifQ.N02ZFtIcdBcdKzW3D7i3kVf
8sTu1TMF8HJqSxo0K7DgkP-96hs2unAfxkFfuXPvneHjSTV3Qsv34mkbbBeLJ7MSKdQred
FCfe-v53yYG3jMN96u_h2nPwf9jvp4pAqUanLng0hxzRofTwU3aJzQ0_gDcPlksckchazB7
Mi3IMdwVqYIGjRqMUVbBl6JomcoJviZ6ui_3RTiu2tbXmLi36KP6VGi7cJ24x0oBHeArgiB3D
XoyAKJb8wXOb33iL8uklcTEbGdc9RLZnRIVgd9OJrAuYcSSuVBFADkI0U-PY6BmRlY0If_KH
bXIGwICsod7nIGNxC1X07jTqn3WfFGOYQ
```

1. Store the token in Jenkins (Credentials ID: kubernetes).

2. Get the EKS API server URL:

```
aws eks describe-cluster --name netflix-cluster --query 'cluster.endpoint' --output text
```

```
teddyyekyawthu@mac .aws % aws eks describe-cluster --name netflix-cluster1 --query 'cluster.endpoint' --output text --profile teddy-kube
https://ABCD027AC4657D748C746D37AC3C7DFE.gr7.ap-southeast-1.eks.amazonaws.com
```

The screenshot shows the Jenkins Global credentials (unrestricted) page. A new credential is being created with the following details:

- ID:** aws-secret
- Access Key ID:** AKIA3ISBVYU33ASAHBHP
- Secret Access Key:** Concealed
- Description:** (empty)
- IAM Role Support:** Advanced

```
aws eks describe-cluster --name netflix-cluster1 --query 'cluster.endpoint' --output text --profile teddy-kube
https://ABCD027AC4657D748C746D37AC3C7DFE.gr7.ap-southeast-1.eks.amazonaws.com
```

edit in jenkins file

```
netflix-test > Jenkinsfile
  1 pipeline {
  2     stages {
  32         stage('Deploy to Kubernetes') {
  42             steps {
  43                 withCredentials([
  44                     $class: 'AmazonWebServicesCredentialsBinding',
  45                     credentialsId: 'aws-secret'
  46                 ]) {
  47                     script {
  48                         dir('Kubernetes') {
  49                             withKubeConfig(
  50                                 credentialsId: "${KUBERNETES_CREDENTIALS_ID}",
  51                                 serverUrl: 'https://ABCD027AC4657D748C746D37AC3C7DFE.gr7.ap-southeast-1.eks.amazonaws.com', // Replace with actual EKS endpoint
  52                                 namespace: "${K8S_NAMESPACE}"
  53                             ) {
  54                             sh 'kubectl version'
  55                             sh "sed -i 's/image: ${DOCKER_IMAGE_NAME}:.*|image: ${DOCKER_IMAGE_NAME}:${env.IMAGE_TAG}|' deployment.yml"
  56                             sh 'kubectl apply -f deployment.yml'
  57                             sh 'kubectl apply -f service.yml'
  58                         }
  59                     }
  60                 }
  61             }
  62         }
  63     }
  64 }
```

RBAC.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: jenkins-deployer
  namespace: default
```

```

rules:
- apiGroups: [ "", "apps" ]
  resources: [ "deployments", "services", "pods" ]
  verbs: [ "get", "list", "create", "update", "delete", "patch" ]
- apiGroups: [ "" ]
  resources: [ "deployments" ]
  verbs: [ "watch" ]

---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: jenkins-deployer-binding
  namespace: default
subjects:
- kind: ServiceAccount
  name: jenkins
  namespace: default
roleRef:
  kind: Role
  name: jenkins-deployer
  apiGroup: rbac.authorization.k8s.io

```

Jenkins File Setting up

```

pipeline {
  agent any

  parameters {
    string(name: 'MAJOR_VERSION', defaultValue: '1', description: 'Major version for the Docker image tag (e.g., 1 for v1.x)')
  }

  tools {
    jdk 'jdk21'
    nodejs 'node20'
  }

  environment {
    SCANNER_HOME = tool 'sonarqube-scanner'
    TRIVY_HOME = '/usr/bin'
    REPO_URL = 'https://github.com/TEDDY201289/Netflix-test.git'
    REPO_BRANCH = 'main'
    DOCKER_IMAGE_NAME = 'teddy2012/netflix-bubudu'
    SONAR_PROJECT_NAME = 'Netflix'
    SONAR_PROJECT_KEY = 'Netflix'
    DOCKER_CREDENTIALS_ID = 'dockerhub'
  }
}

```

```

SONAR_CREDENTIALS_ID = 'SonarQube-Token'
KUBERNETES_CREDENTIALS_ID = 'kubernetes'
SONAR_SERVER = 'SonarQube-Server'
DOCKER_TOOL_NAME = 'docker'
TRIVY_FS_REPORT = 'trivyfs.txt'
TRIVY_IMAGE_REPORT = 'trivyimage.txt'
K8S_NAMESPACE = 'default'
APP_NAME = 'netflix'

}

stages {
    stage('Clean Workspace') {
        steps {
            cleanWs()
        }
        post {
            always {
                echo "Workspace cleaned successfully"
            }
        }
    }
    stage('Checkout from Git') {
        steps {
            git branch: "${REPO_BRANCH}", url: "${REPO_URL}", credentialsId: 'github-credentials'
        }
    }
    stage('SonarQube Analysis') {
        steps {
            withSonarQubeEnv(credentialsId: "${SONAR_CREDENTIALS_ID}", installationName: "${SONAR_SERVER}") {
                sh """
${SCANNER_HOME}/bin/sonar-scanner \
-Dsonar.projectName=${SONAR_PROJECT_NAME} \
-Dsonar.projectKey=${SONAR_PROJECT_KEY}
"""
                }
            }
        }
    stage('Quality Gate') {
        steps {
            script {
                waitForQualityGate abortPipeline: false, credentialsId: "${SONAR_CREDENTIALS_ID}"
            }
        }
    }
}

```

```

stage('Install Dependencies') {
    steps {
        script {
            def nodeHome = tool name: 'node20', type: 'nodejs'
            env.PATH = "${nodeHome}/bin:${env.PATH}"
            sh "node --version"
            sh "npm --version"
            sh "npm install"
        }
    }
    post {
        failure {
            echo "Failed to install dependencies. Check Node.js setup or package.json."
        }
    }
}

// stage('OWASP Dependency Check') {
//     steps {
//         withCredentials([string(credentialsId: 'nvd-api-key', variable: 'NVD_API_KEY')])
//             dependencyCheck additionalArguments: "--scan ./ --disableYarnAudit --disableNodeAudit --nvdApiKey ${NVD_API_KEY}", odciInstallation: 'DP-Check'
//             dependencyCheckPublisher pattern: '**/dependency-check-report.xml'
//     }
// }

stage('Trivy FS Scan') {
    steps {
        script {
            sh "${TRIVY_HOME}/trivy fs . > ${TRIVY_FS_REPORT}"
        }
    }
    post {
        always {
            archiveArtifacts artifacts: "${TRIVY_FS_REPORT}", allowEmptyArchive: true
        }
    }
}

stage('Set Version') {
    steps {
        script {
            def buildNumber = env.BUILD_NUMBER ?: '0'
            env.IMAGE_TAG = "v${params.MAJOR_VERSION}.${buildNumber}"
            echo "Docker image will be tagged as: ${DOCKER_IMAGE_NAME}:${env.IMAGE_TAG}"
        }
    }
}

```

```

        }
    stage('Docker Build & Push') {
        steps {
            withCredentials([string(credentialsId: 'tmdb-api-key', variable: 'TMDB_API_KEY')])
            {
                script {
                    withDockerRegistry(credentialsId: "${DOCKER_CREDENTIALS_ID}", toolName: "${DOCKER_TOOL_NAME}")
                    sh "docker build -t ${APP_NAME} --build-arg TMDB_V3_API_KEY=${TMDB_API_KEY}"
                    sh "docker tag ${APP_NAME} ${DOCKER_IMAGE_NAME}:${env.IMAGE_TAG}"
                    sh "docker push ${DOCKER_IMAGE_NAME}:${env.IMAGE_TAG}"
                }
            }
        }
    }
    post {
        always {
            sh "docker rmi ${APP_NAME} ${DOCKER_IMAGE_NAME}:${env.IMAGE_TAG} | true"
        }
    }
}
stage('Trivy Image Scan') {
    steps {
        script {
            sh "${TRIVY_HOME}/trivy image ${DOCKER_IMAGE_NAME}:${env.IMAGE_TAG} > ${TRIVY_IMAGE_REPORT}"
        }
    }
    post {
        always {
            archiveArtifacts artifacts: "${TRIVY_IMAGE_REPORT}", allowEmptyArchive: true
        }
    }
}
stage('Deploy to Kubernetes') {
    steps {
        withCredentials([
            $class: 'AmazonWebServicesCredentialsBinding',
            credentialsId: 'aws-secret'
        ])
        {
            script {
                dir('Kubernetes') {
                    withKubeConfig(

```

ALB installation on EKS

```
# Download an IAM policy for the AWS Load Balancer Controller that allows it to make calls to AWS APIs on your behalf.  
curl -O https://raw.githubusercontent.com/kubernetes-sigs/aws-load-balancer-controller/v2.13.0/docs/install/iam_policy.json  
# Create an IAM policy using the policy downloaded in the previous step.
```

`export AWS_PROFILE=profilename` (Export profile name)

```
export AWS_PROFILE=teddy-kube
```

```
# Create an IAM policy using the policy downloaded in the previous step.  
aws iam create-policy \  
  --policy-name AWSLoadBalancerControllerIAMPolicy \  
  --policy-document file://iam_policy.json \  
  --profile teddy-kube
```

Policies (1385)

A policy is an object in AWS that defines permissions.

Policy name	Type	Used as	Description
AWSLoadBalancerControllerIAMPolicy	Customer managed	Permissions policy (1)	-

Policy details

Type: Customer managed | Creation time: August 09, 2025, 21:10 (UTC+08:00) | Edited time: August 09, 2025, 21:10 (UTC+08:00) | ARN: arn:aws:iam::774305596727:policy/AWSLoadBalancerControllerIAMPolicy

Permissions Entities attached Tags Policy versions Last Accessed

Attached as a permissions policy (1)

To grant permissions to an entity, attach a permissions policy to it.

Entity name	Entity type
-	-

No entities attached , so OIDC is required to run.

```
# Create OIDC provider (eksctl)
cluster_name=<my-cluster>
oidc_id=$(aws eks describe-cluster --name $cluster_name --query "cluster.identity.oidc.issuer" --output text | cut -d '/' -f 5)
echo $oidc_id
```

check iam policy

```
eksctl utils associate-iam-oidc-provider --cluster netflix-cluster1 --approve
eksctl create iamserviceaccount \
--cluster=netflix-cluster1 \
--namespace=kube-system \
--name=aws-load-balancer-controller \
--attach-policy-arn=arn:aws:iam::774305596727:policy/AWSLoadBalancerControllerIAMPolicy \
--override-existing-serviceaccounts \
--region ap-southeast-1 \
--approve
--profile teddy-kube
```

Install AWS Load Balancer Controller (ALB)

Add the eks-charts Helm chart repository. AWS maintains this repository on GitHub.

```
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 kube-system \
--set clusterName=my-cluster \
--set serviceAccount.create=false \
--set serviceAccount.name=aws-load-balancer-controller \
--version 1.13.0
kubectl get deployment -n kube-system aws-load-balancer-controller

teddyyekyawthu@mac .aws % 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           37h
```

If not working with eksctl Create OIDC provider (AWS Console)

- Open the Amazon EKS console.
- In the left pane, select Clusters, and then select the name of your cluster on the Clusters page.
- In the Details section on the Overview tab, note the value of the OpenID Connect provider URL.
- Open the IAM console at <https://console.aws.amazon.com/iam/>.
- In the left navigation pane, choose Identity Providers under Access management. If a Provider is listed that matches the URL for your cluster, then you already have a provider for your cluster. If a provider isn't listed that matches the URL for your cluster, then you must create one.
- To create a provider, choose Add provider.
- For Provider type, select OpenID Connect.

- For Provider URL, enter the OIDC provider URL for your cluster.
- For Audience, enter sts.amazonaws.com.
- (Optional) Add any tags, for example a tag to identify which cluster is for this provider.
- Choose Add provider.

ingress.yaml

```

apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  namespace: default
  name: ingress
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
spec:
  ingressClassName: alb
  rules:
  - host: netflix.bubududu.life
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: netflix-app
            port:
              number: 80
# apiVersion: networking.k8s.io/v1
# kind: Ingress
# metadata:
#   namespace: default
#   name: ingress
#   annotations:
#     alb.ingress.kubernetes.io/scheme: internet-facing
#     alb.ingress.kubernetes.io/target-type: ip
#     # alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}]'
#     # alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}, {"HTTPS": 443}]'
#     # alb.ingress.kubernetes.io/certificate-arn: arn:aws:acm:ap-southeast-1:53746792616
#     # certificate/aa669601-04ad-4e78-96a2-077ac68623b6
#     # alb.ingress.kubernetes.io/ssl-redirect: '443'
#     # kubernetes.io/ingress.class: alb
#   spec:
#     # # ingressClassName: alb
#     # rules:
#     #   - host: netflix.myanmartechsolutions.site
#     #     http:

```

```

#   paths:
#     - path: /
#       pathType: Prefix
#       backend:
#         service:
#           name: netflix-app
#           port:
#             number: 80
# - host: youtube.myanmartechnologies.site
#   http:
#     paths:
#       - path: /
#         pathType: Prefix
#         backend:
#           service:
#             name: youtube-clone
#             port:
#               number: 80

```

```

teddyyekyawthu@mac Kubernetes % kubectl get ingress
NAME    CLASS   HOSTS          ADDRESS          P
ORTS   AGE
ingress  alb    netflix.bubududu.life  k8s-default-ingress-d940d6d82e-1249514683.ap-s
outheast-1.elb.amazonaws.com  80    37h

```

#running svc type would be Loadbalancer , so needs to change back ClusterIP else ALB will not work.

```
teddyyekyawthu@mac Kubernetes % kubectl edit svc netflix-app
```

```

# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: Service
metadata:
  annotations:
    kubectl.kubernetes.io/last-applied-configuration: |
      creationTimestamp: "2025-08-09T13:03:27Z"
  labels:
    app: netflix-app
    name: netflix-app
    namespace: default
    resourceVersion: "10055"
    uid: a3a39e64-34d1-4dbb-95c0-5f37bf84ae64

```

```

spec:
  clusterIP: 10.100.220.54
  clusterIPs:
  - 10.100.220.54
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: netflix-app
  sessionAffinity: None
  type: ClusterIP #change the IP type from LoadBalancer to ClusterIP
status:
  loadBalancer: {}

```

running svc type would be Loadbalancer , so needs to change back ClusterIP else ALB will not work.

```

teddyyekyawthu@mac .aws % kubectl get svc
NAME      TYPE      CLUSTER-IP      EXTERNAL-IP      PORT(S)      AGE
kubernetes  ClusterIP  10.100.0.1    <none>        443/TCP      38h
netflix-app  ClusterIP  10.100.220.54  <none>        80/TCP      38h

```

```

k8s-default-ingress-d940d6d82e-1249514683.ap-southeast-1.elb.amazonaws.com.
52.220.67.87
18.139.120.226
13.214.251.191

```

back to Push (CICD)

Go to path what we want to push via Git

```

teddyyekyawthu@mac netflix-test % pwd
/Users/teddyyekyawthu/Teddy-files/netflix-test

```

```

ssh -T git@github.com (#authentication)

rm -rf .git (optional)

git init

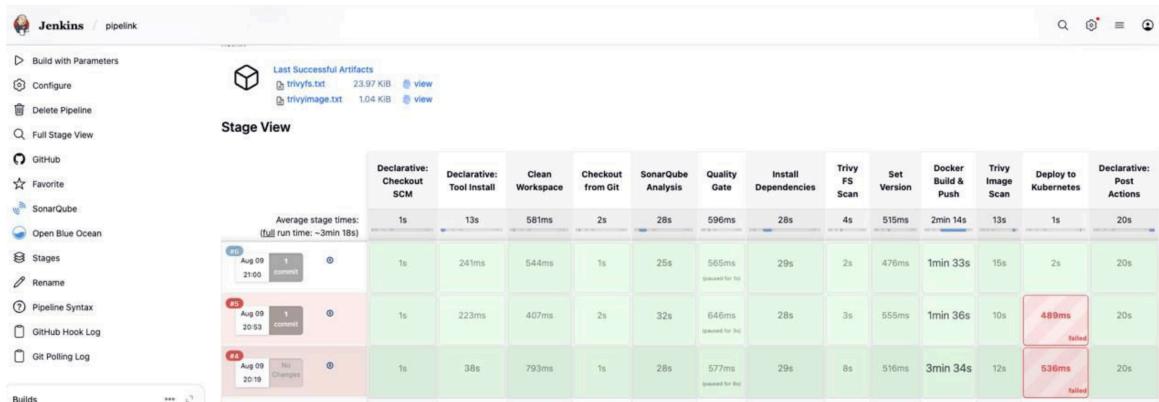
git add .

git commit -m "added output.tf"

git branch -M main

git push -u origin main

```



Go to Godaddy

<https://dcc.godaddy.com/control/dnsmanagement?domainName=bubududu.life&subtab=nameservers>

add Nameserver based on Cloudflare DNS information:

The screenshot shows the GoDaddy DNS Management interface for the domain "bubududu.life". The left sidebar includes options for Domains, Portfolio, DNS, Transfers, Services, and Settings. The main area is titled "DNS Management" and shows the "Nameservers" tab selected. It lists two nameservers: "chole.ns.cloudflare.com" and "ray.ns.cloudflare.com". There are buttons for "Domain Settings" and "Select a different domain". Below the nameservers, there's a note about what they determine and a "Change Nameservers" button.

Cloudflare

<https://dash.cloudflare.com/2d979a5d2c1c7389c0f62a7968ec1140/bubududu.life/dns/records>

CLOUDFLARE 2ybfn8dzn@privaterelay.appleid.com's Account

bubududu.life Active Starred Free plan

Records

Configure DNS records and review proxy status of your hostnames. [DNS documentation](#)

Recommended steps to complete zone set-up

- ✓ Add an A, AAAA, or CNAME record for www so that www.bubududu.life will resolve.
- ✓ Add an A, AAAA, or CNAME record for your root domain so that bubududu.life will resolve.
- ✓ Add an MX record for your root domain so that mail can reach @bubududu.life addresses or [set up restrictive SPF, DKIM, and DMARC records](#) to prevent email spoofing. [New Alert](#)

DNS management for bubududu.life

Review, add, and edit DNS records. Edits will go into effect once saved. [DNS Setup: Full](#) [Import and Export](#) [Dashboard Display Settings](#)

Type	Name	Content	Proxy status	TTL	Actions
A	bubududu.life	13.248.243.5	DNS only	Auto	Edit
A	bubududu.life	76.223.105.230	DNS only	Auto	Edit
A	bubududu.life	192.0.2.1	DNS only	Auto	Edit
CNAME	netflix	k8s-default-ingress-d940d...	DNS only	Auto	Edit

Cloudflare Nameservers	
Every DNS zone on Cloudflare is assigned a set of Cloudflare-branded nameservers.	
Type	Value
NS	chloe.ns.cloudflare.com
NS	ray.ns.cloudflare.com

