

DevSecOps Pipeline Project: Deploy Netflix Clone on Kubernetes

For this project, I will deploy a Netflix clone application on AWS using DevSecOps practices, integrating security from the start with tools like SonarQube, Trivy, and OWASP. The deployment pipeline will use CI/CD with Jenkins, monitoring with Prometheus and Grafana, and Kubernetes orchestration using ArgoCD and Helm.

DevSecOps Phase 1: DEV

The DEV part of DevSecOps deals with ensuring the application runs locally.

Steps

Initial Set-up and Deployment: Create an EC2 Server

To deploy this application, I used a **t2.large** EC2 instance for Jenkins, SonarQube, and Trivy. This instance type is necessary because multiple plugins will be installed for Jenkins, and I need to ensure the server can handle the resource requirements.

The screenshot shows the 'Instance type' section of the AWS CloudFormation 'Create New Stack' wizard. A dropdown menu is open, showing 't2.large' as the selected option. Below the dropdown, detailed information about the t2.large instance type is provided, including its family (t2), vCPUs (2), memory (8 GiB), current generation status, and base pricing for various operating systems. To the right of the dropdown, there are buttons for 'All generations' and 'Compare instance types'. A note at the bottom states 'Additional costs apply for AMIs with pre-installed software'.

For Firewall settings, I enabled HTTP and HTTPS traffic options since Jenkins will be needing to connect to Dockerhub via HTTPS.

Firewall (security groups) | [Info](#)
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group

Select existing security group

We'll create a new security group called '**launch-wizard-11**' with the following rules:

- Allow SSH traffic from
Helps you connect to your instance
- Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server
- Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

Anywhere

0.0.0.0/0

⚠️ Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only. **X**

Storage was also increased to accommodate the anticipated requirements of the project.

▼ Configure storage [Info](#)

[Advanced](#)

1x GiB Root volume, 3000 IOPS, Not encrypted

ⓘ Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage **X**

[Add new volume](#)

The selected AMI contains more instance store volumes than the instance allows. Only the first 0 instance store volumes from the AMI will be accessible from the instance

⌚ Click refresh to view backup information



The tags that you assign determine whether the instance will be backed up by any Data Lifecycle Manager policies.

0 x File systems

[Edit](#)

After the instance has been successfully created, I attached an ElasticIP to the instance. This will help if the project will be done in more than one or two sessions, so the same IP instance will still be used even if the instance is stopped.

Elastic IP addresses (1)



[Actions ▾](#)

Find resources by attribute or tag

Public IPv4 address : 54.79.85.218

[Clear filters](#)

<input type="checkbox"/> Name	Allocated IPv4 addr...	Type	Allocation ID	Rever:
<input type="checkbox"/> netflix-eip	54.79.85.218	Public IP	eipalloc-09bbebd87f4d6e53f	-

Connect to EC2 Instance

Next, I connected to my EC2 instance via EC2 Instance Connect, which supports Ubuntu (the AMI used to create the instance) and ran `sudo apt update -y` to update all relevant packages.

```
Get:28 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu noble-backports/main amd64 c-n-f Metadata [112 B]
Get:29 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Packages [14.2 kB]
Get:30 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu noble-backports/universe Translation-en [12.1 kB]
Get:31 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 Components [19.8 kB]
Get:32 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu noble-backports/universe amd64 c-n-f Metadata [1104 B]
Get:33 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 Components [216 B]
Get:34 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu noble-backports/restricted amd64 c-n-f Metadata [116 B]
Get:35 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 Components [212 B]
Get:36 http://ap-southeast-2.ec2.archive.ubuntu.com/ubuntu noble-backports/multiverse amd64 c-n-f Metadata [116 B]
Get:37 http://security.ubuntu.com/ubuntu noble-security/main amd64 Packages [634 kB]
Get:38 http://security.ubuntu.com/ubuntu noble-security/main Translation-en [121 kB]
Get:39 http://security.ubuntu.com/ubuntu noble-security/main amd64 Components [9000 B]
Get:40 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Packages [815 kB]
Get:41 http://security.ubuntu.com/ubuntu noble-security/universe Translation-en [174 kB]
Get:42 http://security.ubuntu.com/ubuntu noble-security/universe amd64 Components [51.9 kB]
Get:43 http://security.ubuntu.com/ubuntu noble-security/universe amd64 c-n-f Metadata [13.5 kB]
Get:44 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Packages [650 kB]
Get:45 http://security.ubuntu.com/ubuntu noble-security/restricted Translation-en [128 kB]
Get:46 http://security.ubuntu.com/ubuntu noble-security/restricted amd64 Components [212 B]
Get:47 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Packages [19.4 kB]
Get:48 http://security.ubuntu.com/ubuntu noble-security/multiverse Translation-en [4308 B]
Get:49 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 Components [212 B]
Get:50 http://security.ubuntu.com/ubuntu noble-security/multiverse amd64 c-n-f Metadata [356 B]
Fetched 32.4 MB in 18s (1761 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
120 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@ip-172-31-1-131:~$
```

Cloning The Application

I cloned the Netflix application source code from a GitHub repo:

<https://github.com/N4si/DevSecOps-Project> onto my EC2 instance by running the command `git clone <repo link>`. Running the `ls` command, we can see that the code has been successfully cloned into a *DevSecOps-Project* repository in the EC2 instance.

```
ubuntu@ip-172-31-1-131:~$ ls
DevSecOps-Project
```

```
ubuntu@ip-172-31-1-131:~/DevSecOps-Project$ ls
Dockerfile  Kubernetes  README.md  index.html  package.json  pipeline.txt  public  src  tsconfig.json  tsconfig.node.json  vercel.json  vite.config.ts  yarn.lock
```

Then I installed Docker, which is needed to create the Docker image.

```
sudo apt-get update
sudo apt-get install docker.io -y
sudo usermod -aG docker $USER  # Rep
newgrp docker
sudo chmod 777 /var/run/docker.sock
```

```
ubuntu@ip-172-31-1-131:~/DevSecOps-Project$ docker version
Client:
  Version:          26.1.3
  API version:      1.45
  Go version:       go1.22.2
  Git commit:       26.1.3-0ubuntu1~24.04.1
  Built:            Mon Oct 14 14:29:26 2024
  OS/Arch:          linux/amd64
  Context:          default

Server:
  Engine:
    Version:          26.1.3
    API version:      1.45 (minimum version 1.24)
    Go version:       go1.22.2
    Git commit:       26.1.3-0ubuntu1~24.04.1
    Built:            Mon Oct 14 14:29:26 2024
    OS/Arch:          linux/amd64
    Experimental:    false
  containerd:
    Version:          1.7.24
    GitCommit:
  runc:
    Version:          1.1.12-0ubuntu3.1
    GitCommit:
  docker-init:
```

Once docker has successfully been installed, the next step is to build and run the application using docker containers. Docker build was initiated using the command `docker build -t netflix .`

```
Successfully built 061aa0ad9b76
Successfully tagged netflix:latest
ubuntu@ip-172-31-1-131:~/DevSecOps-Project$ docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
netflix         latest       061aa0ad9b76   2 minutes ago  56.5MB
```

Next, I ran `docker run -d -p 8081:80 <image ID>` to start the Docker container from the Netflix image.

-d (detached mode)

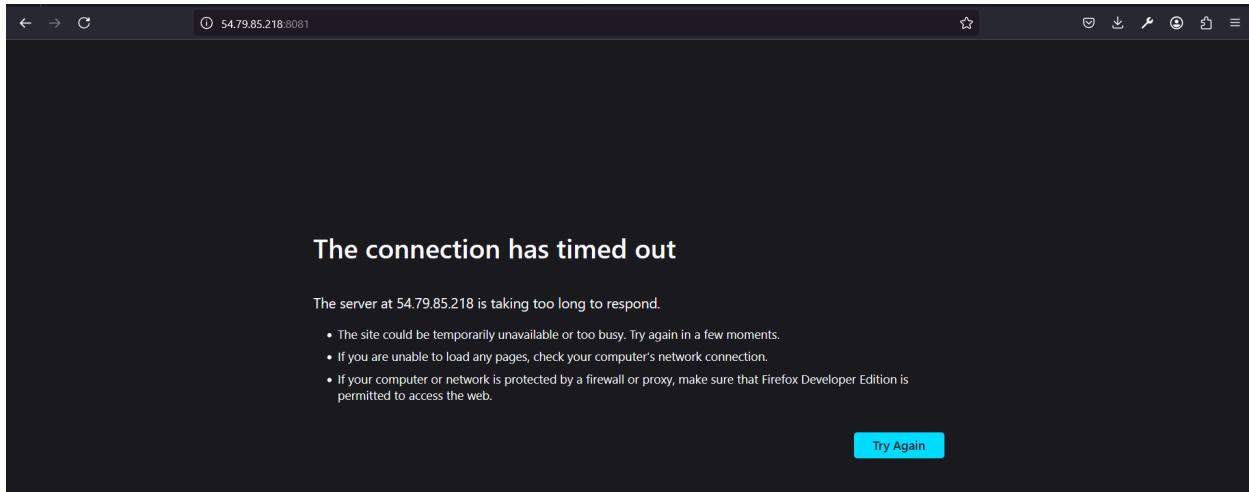
- Runs the container in the background (detached mode).
- Without -d, the container runs in the foreground, with the logs visible in the terminal.

-p 8081:80 (port mapping)

- Maps **port 8081** on the local machine (host) to **port 80** inside the container.

This will start the *netflix* image inside a container. The *netflix* server listens on port 80 inside the container. It can be accessed through <http://localhost:8081>

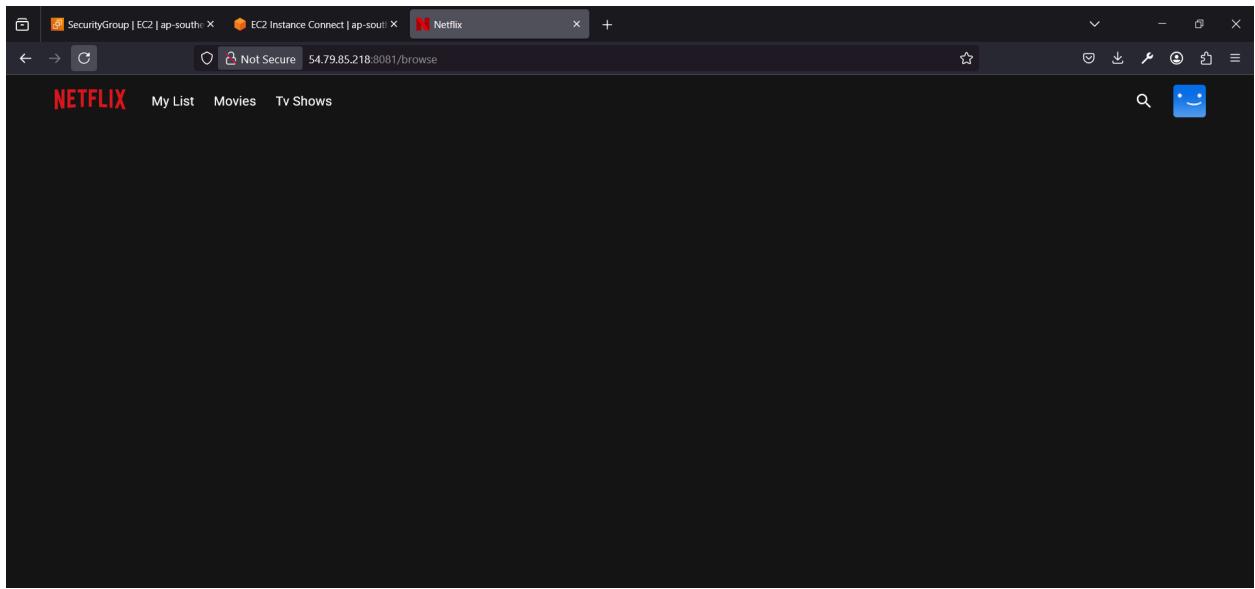
Accessing the localhost of the instance with port 8081:



This happens because port 8081 is not open in the instance Security Group. Ports 8081 for the application, 8080 for Jenkins, and 9000 for SonarQube were added to the inbound rules for the Security Group.

Port Range	Protocol	Source	Action
8081	TCP	Anywhere	app port
8080	TCP	Anywhere	Jenkins
9000	TCP	Anywhere	SonarQube

Now that we've allowed the ports on the Security Group, we can now access the Application on port 8081.



The application works; however, there is no movie selection or catalog of shows to choose from, which are expected features of the Netflix website. This issue is caused by the ARG statement in the Dockerfile as seen on the source code.

Endo replace nginx base image with alpine

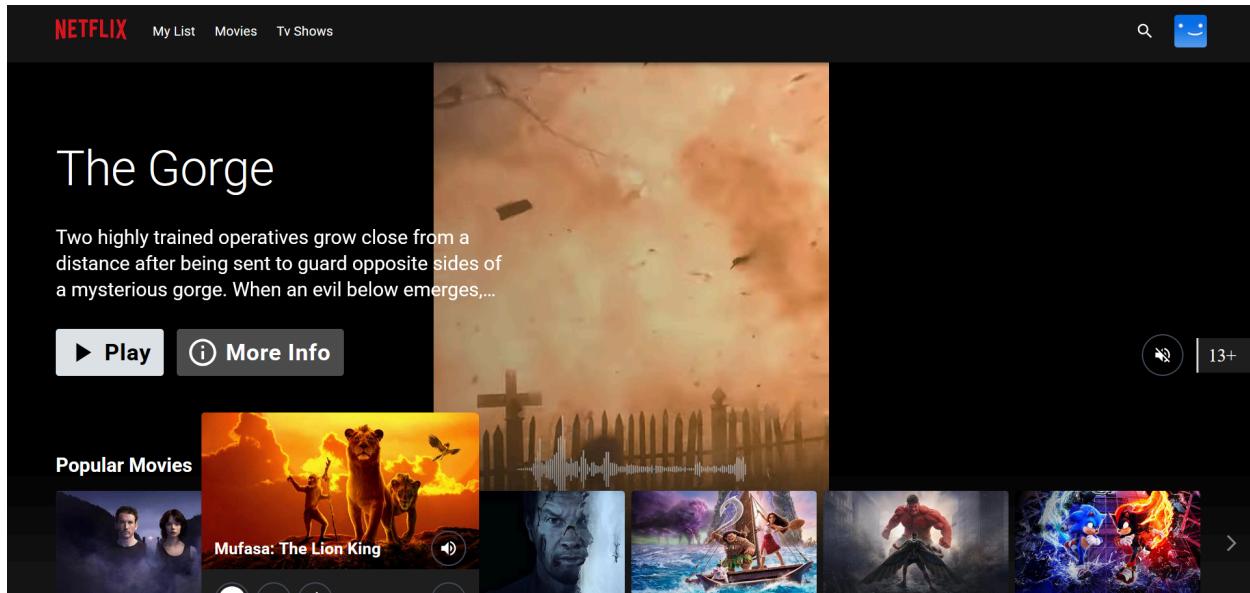
Code Blame 17 lines (16 loc) · 414 Bytes GitHub Copilot

```
1 FROM node:16.17.0-alpine as builder
2 WORKDIR /app
3 COPY ./package.json .
4 COPY ./yarn.lock .
5 RUN yarn install
6 COPY . .
7 ARG TMDB_V3_API_KEY
8 ENV VITE_APP_TMDB_V3_API_KEY=${TMDB_V3_API_KEY}
9 ENV VITE_APP_API_ENDPOINT_URL="https://api.themoviedb.org/3"
10 RUN yarn build
11
12 FROM nginx:stable-alpine
13 WORKDIR /usr/share/nginx/html
14 RUN rm -rf /*
15 COPY --from=builder /app/dist .
16 EXPOSE 80
17 ENTRYPOINT ["nginx", "-g", "daemon off;"]
```

The dockerfile expects an argument, and I haven't provided its requirements yet, an API key from an application called TMDB for the movie database. The API key is necessary so that the netflix application can fetch the data into the application.

I stopped and removed the container before starting it again with the needed API key with this command:

```
docker build --build-arg TMDB_V3_API_KEY=<api-key> -t netflix .
```



The site is now accessible from the local machine on port 8081 with the catalogue of films.

DevSecOps Phase 2: Security

The Security Phase deals with ensuring that the code, and all its dependencies and images are correct, and properly set-up. For security in this project, two tools will be utilized: SonarQube (analyzes code and gives reports on potential bugs and issues with code) and Trivy (scans file systems and gives reports on file system issues).

Install SonarQube and Trivy

To install SonarQube I ran `docker run -d --name sonar -p 9000:9000 sonarqube:lts-community`

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
58551aa9fe0	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
424ba7b155e6	netflix	"nginx -g 'daemon off;'"	12 minutes ago	Up 12 minutes	0.0.0.0:8081->80/tcp, :::8081->80/tcp	tender_faraday

Accessing SonarQube at port 9000

sonarqube Projects Issues Rules Quality Profiles Quality Gates Administration

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.

From Azure DevOps **From Bitbucket Server** **From Bitbucket Cloud** **From GitHub** **From GitLab**

Set up global configuration Set up global configuration Set up global configuration Set up global configuration Set up global configuration

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

< >

Next I installed Trivy by Running the command:

```
sudo apt-get install wget apt-transport-https gnupg lsb-release
wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | sudo apt-key add -
echo deb https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main | sudo tee -a /
sudo apt-get update
sudo apt-get install trivy
```

And scanned the netflix image

```
ubuntu@ip-172-31-1-131:~/DevSecOps-Project$ trivy image f6ca600c8b76
2025-02-27T08:43:22Z   INFO  [vuln] Vulnerability scanning is enabled
2025-02-27T08:43:22Z   INFO  [secret] Secret scanning is enabled
2025-02-27T08:43:22Z   INFO  [secret] If your scanning is slow, please try '--scanners vuln' to disable secret scanning
2025-02-27T08:43:22Z   INFO  [secret] Please see also https://aquasecurity.github.io/trivy/v0.59/docs/scanner/secret#recommendation for faster secret detection
2025-02-27T08:43:22Z   INFO  Detected OS      family="alpine" version="3.20.6"
2025-02-27T08:43:22Z   INFO  [alpine] Detecting vulnerabilities... os_version="3.20" repository="3.20" pkg_num=66
2025-02-27T08:43:22Z   INFO  Number of language-specific files      num=0
```

DevSecOps Phase 3: Operations

The operations segment includes Integrating a CI/CD Pipeline into the application using Jenkins.

Install Jenkins for Automation

Java installation is required for Jenkins to work. Using the following commands, Java and Jenkins were installed.

```

sudo apt update
sudo apt install fontconfig openjdk-17-jre
java -version
openjdk version "17.0.8" 2023-07-18
OpenJDK Runtime Environment (build 17.0.8+7-Debian-1deb12u1)
OpenJDK 64-Bit Server VM (build 17.0.8+7-Debian-1deb12u1, mixed mode, sharing)

#jenkins
sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
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
sudo apt-get install jenkins
sudo systemctl start jenkins
sudo systemctl enable jenkins

```

```

No VM guests are running outdated hypervisor (qemu) binaries on this host.
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
ubuntu@ip-172-31-1-131:~/DevSecOps-Project$ sudo jenkins status
● Jenkins.service - Jenkins Continuous Integration Server
    Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
    Active: active (running) since Thu 2025-02-27 08:53:49 UTC; 45s ago
      Main PID: 15154 (java)
        Tasks: 50 (limit: 9507)
       Memory: 815.3M (peak: 833.0M)
          CPU: 15.598s
         CGroup: /system.slice/jenkins.service
                 └─15154 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Feb 27 08:53:44 ip-172-31-1-131 jenkins[15154]: 98dace7ceaf148ecb06b9bcf7793dec
Feb 27 08:53:44 ip-172-31-1-131 jenkins[15154]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Feb 27 08:53:44 ip-172-31-1-131 jenkins[15154]: ****
Feb 27 08:53:49 ip-172-31-1-131 jenkins[15154]: 2025-02-27 08:53:49.410+0000 [id=30]     INFO  jenkins.InitReactorRunner$1#onAttained: Completed initialization
Feb 27 08:53:49 ip-172-31-1-131 jenkins[15154]: 2025-02-27 08:53:49.426+0000 [id=23]     INFO  hudson.lifecycle.Lifecycle$onReady: Jenkins is fully up and running
Feb 27 08:53:49 ip-172-31-1-131 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Feb 27 08:53:50 ip-172-31-1-131 jenkins[15154]: 2025-02-27 08:53:50.676+0000 [id=48]     INFO  h.m.DownloadService$Downloadable#load: Obtained the updated data
Feb 27 08:53:50 ip-172-31-1-131 jenkins[15154]: 2025-02-27 08:53:50.677+0000 [id=48]     INFO  hudson.util.Retriger#start: Performed the action check updates se

```

Accessing Jenkins on port 8080 of the EC2 Instance, I landed on a sign-in page.

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

I ran the command `sudo cat /var/lib/jenkins/secrets/initialAdminPassword` which outputs the administrator password, and allowed me to log in and start suggested downloads. After Jenkins has successfully completed its download into the machine, I downloaded some of the plugins needed such as NodeJS, Eclipse Temurin Installer, and SonarQube.

The screenshot shows the Jenkins Marketplace interface. At the top, there are tabs for 'Install' and 'Name ↓'. A search bar contains the text 'NodeJS'. On the right, a 'Released' filter is applied. Three plugins are listed:

- NodeJS 1.6.4**: An npm plugin that executes NodeJS scripts as build steps. It was released 6 days 20 hr ago.
- Eclipse Temurin installer 1.7**: Provides an installer for the JDK tool that downloads the Eclipse Temurin™ build based upon OpenJDK from the [Adoptium Working Group](#). It was released 1 day 15 hr ago.
- SonarQube Scanner 2.18**: An External Site/Tool Integrations plugin for Build Reports. It was released 1 mo 0 days ago. This plugin allows an easy integration of [SonarQube](#), the open source platform for Continuous Inspection of code quality.

Once these were downloaded, these plugins were set in the Tools section for NodeJS and JDK.

The screenshot shows the Jenkins 'Tools' configuration page. It is divided into two main sections: 'JDK' and 'NodeJS'.

JDK Section:

- Name: jdk17
- Install automatically: checked
- Install from adoptium.net:
 - Version: jdk-17.0.8.1+1
 - Add Installer

NodeJS Section:

- Name: node16
- Install automatically: checked
- Install from nodejs.org:
 - Version: NodeJS 16.2.0
 - For the underlying architecture, if available, force the installation of:
 - Force 32bit architecture
 - Global npm packages to install:
 - Specify list of packages to install globally -- see npm install -g. Note

For SonarQube, it has to be first connected to Jenkins, and then added to the system, and lastly to the tools like the other plugins. On SonarQube (Port 9000) I generated a token

The screenshot shows the Jenkins 'Tokens of Administrator' page.

Generate Tokens:

- Name: Enter Token Name
- Expires in: 30 days
- Generate

A message box indicates: "New token "jenkins" has been created. Make sure you copy it now, you won't be able to see it again!"

Copy button: squ_00207d25622e9f8642b707e32e356965aae2b69

Name	Type	Project	Last use	Created	Expiration
jenkins	User		Never	February 27, 2025	March 29, 2025

Back in Jenkins under 'credentials' I added credentials for the SonarQube token I just generated. These newly created credentials are important because it will be used for the SonarQube Tool that I need to create.

Global credentials (unrestricted)

+ Add Credentials

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

ID	Name	Kind	Description
 Sonar-token	Sonar-token	Secret text	Sonar-token

Name
sonar-server

Server URL
Default is <http://localhost:9000>
<http://54.79.85.218:9000>

Server authentication token
SonarQube authentication token. Mandatory when anonymous access is disabled.
Sonar-token

+ Add

Advanced ▾

Next, I created a ‘sonar-scanner’ tool since it will be used on the pipeline to be made.

≡ SonarQube Scanner

Name
sonar-scanner

Install automatically ?

≡ Install from Maven Central

Version
SonarQube Scanner 7.0.2.4839

Add Installer ▾

Now I was able to deploy and build an application on the pipeline using this script:

```

pipeline {
    agent any
    tools {
        jdk 'jdk17'
        nodejs 'node16'
    }
    environment {
        SCANNER_HOME = tool 'sonar-scanner'
    }
    stages {
        stage('clean workspace') {
            steps {
                cleanWs()
            }
        }
        stage('Checkout from Git') {
            steps {
                git branch: 'main', url: 'https://github.com/04si/DevSecOps-Project.git'
            }
        }
        stage("SonarQube Analysis") {
            steps {
                withSonarQubeEnv('sonar-server') {
                    sh """$SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=Netflix \
-Dsonar.projectKey=Netflix"""
                }
            }
        }
        stage("quality gate") {
            steps {
                script {
                    waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'
                }
            }
        }
    }
}

```

Back at the SonarQube, it started analyzing the code for any issues, vulnerabilities and security recommendations

The screenshot shows the SonarQube interface for a project named 'main'. The 'Issues' tab is selected. A specific issue is highlighted: 'Replace `as` with upper case format `AS`' (Code Smell, Major). The code snippet in the Dockerfile shows the line 'FROM node:16.17.0-alpine as builder' with the 'as' part highlighted in red. The SonarQube UI provides context for the issue, including the file path (/Dockerfile), line numbers (1-9), and the specific code line.

While it was building, I downloaded other plugins required: OWASP Dependency Check and Docker, and added credentials for Docker in Jenkins so that I can login using the provided DockerHub username and password.

The screenshot shows the Jenkins plugin store interface. A sidebar on the left lists categories: 'Recently Added', 'Security', 'DevOps', 'Build Tools', and 'Build Reports'. The 'Recently Added' section contains six items:

- OWASP Dependency-Check** 5.6.0: 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.
- Docker** 1.10.0: This plugin integrates Jenkins with Docker.
- Docker Commons** 445.v6b_646c962a_94: Provides the common shared functionality for various Docker-related plugins.
- Docker Pipeline** 601.v6c028908db_f: Build and use Docker containers from pipelines.
- Docker API** 3.4.1-96.v77147a_de67f8: This plugin provides `docker-JAVA` API for other plugins.
- docker-build-step** 2.12: This plugin allows to add various docker commands to your job as build steps.

Then, I added OWASP and Docker (recently added plugins) into the Tools section

Dependency-Check installations

Dependency-Check installations ^

Edited

Add Dependency-Check

Dependency-Check

Name

DP-Check

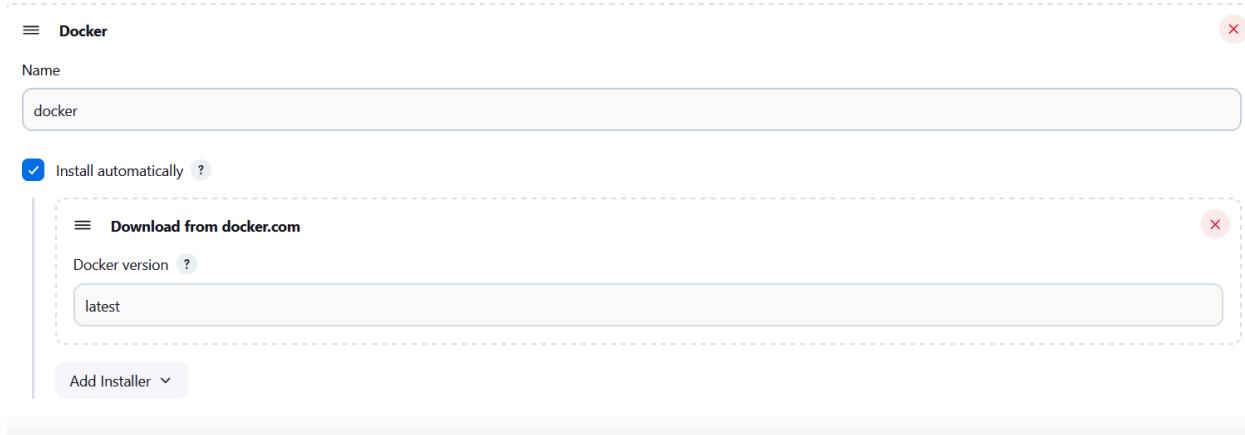
Install automatically ?

Install from github.com

Version

dependency-check 12.1.0

Add Installer ▾



Then I created a pipeline that will be scanning images through Trivy, check dependencies through OWASP, and create the image using commands, integrating all the new features downloaded, and the needed API key.

```
pipeline{
    agent any
    tools{
        jdk 'jdk17'
        nodejs 'node16'
    }
    environment {
        SCANNER_HOME=tool 'sonar-scanner'
    }
    stages {
        stage('clean workspace'){
            steps{
                cleanWs()
            }
        }
        stage('Checkout from Git'){
            steps{
                git branch: 'main', url: 'https://github.com/N4si/DevSecOps-Project.git'
            }
        }
        stage("Sonarqube Analysis"){
            steps{
                withSonarQubeEnv('sonar-server') {
                    sh ''' $SCANNER_HOME/bin/sonar-scanner -Dsonar.projectName=Netflix \
                    -Dsonar.projectKey=Netflix '''
                }
            }
        }
        stage("quality gate"){
            steps {
                script {
                    waitForQualityGate abortPipeline: false, credentialsId: 'Sonar-token'
                }
            }
        }
    }
}
```

The pipeline:

Cleanup workspace, get the code, sonarqube, quality gate check, install dependencies, run dependency scanner, scan file system, run command to build tag image and push image, once image created, scan thru trivy, and then deploy image as container on server.

Phase 4: Monitoring

Install Prometheus and Grafana:

Set up Prometheus and Grafana to monitor the application. Prometheus is used for monitoring alerting about the system's health and performance, optimizing troubleshooting issues. Grafana is then used to visualize the monitoring data that comes in.

To have monitoring enabled, I need to create a new t.2.medium instance due to Prometheus' system requirements (2 CPU Cores, 4GB RAM).

t2.medium

Family: t2 2 vCPU 4 GiB Memory Current generation: true
On-Demand Linux base pricing: 0.0584 USD per Hour On-Demand RHEL base pricing: 0.0872 USD per Hour
On-Demand Windows base pricing: 0.0764 USD per Hour On-Demand SUSE base pricing: 0.1584 USD per Hour
On-Demand Ubuntu Pro base pricing: 0.0619 USD per Hour

Once it was created, I attached an elasticIP to it, and then connected to the instance to install Grafana, Prometheus, Node Exporter, and everything else necessary for monitoring.

Installing Prometheus

```
sudo useradd --system --no-create-home --shell /bin/false prometheus
wget https://github.com/prometheus/prometheus/releases/download/v2.47.1/prometheus-2.47.1.linux-amd64.tar.gz
```

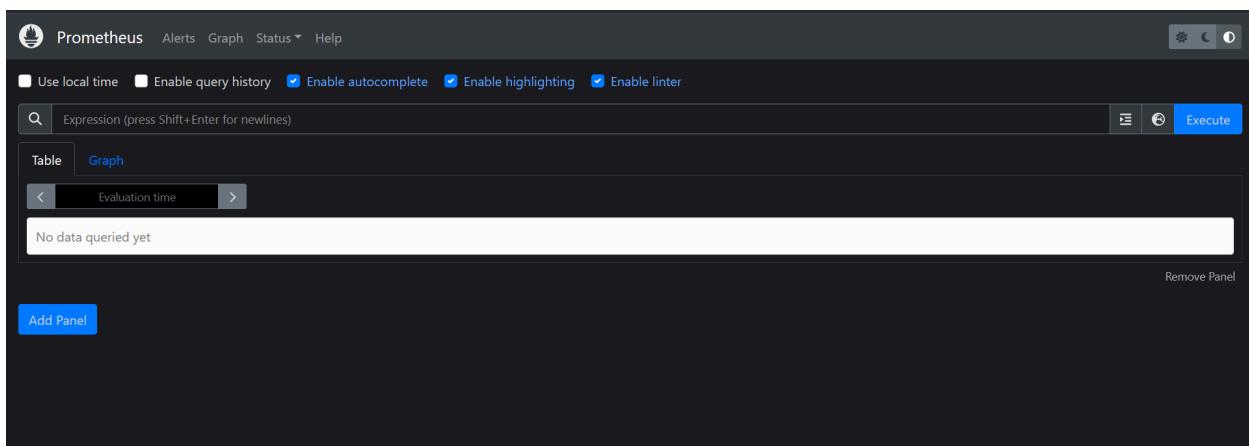
```
ubuntu@ip-172-31-12-9:~$ ls
prometheus-2.47.1.linux-amd64.tar.gz
```

Extract Prometheus files, move them, and create directories:

```
tar -xvf prometheus-2.47.1.linux-amd64.tar.gz
cd prometheus-2.47.1.linux-amd64/
sudo mkdir -p /data /etc/prometheus
sudo mv prometheus promtool /usr/local/bin/
sudo mv consoles/ console_libraries/ /etc/prometheus/
sudo mv prometheus.yml /etc/prometheus/prometheus.yml
```

Promtool is a query tool that uses queries to get data about monitoring

Prometheus.yml is where I can set the servers I want to monitor. To start monitoring I need a Node Exporter. Node Exporter is a lightweight agent developed by the Prometheus project that gathers and exposes system-level metrics from Linux systems, enabling Prometheus to monitor them. Node Exporter will extract the metric, and these metrics will be used by Prometheus to start monitoring.



Installing Node Exporter

```
sudo useradd --system --no-create-home --shell /bin/false node_exporter
wget https://github.com/prometheus/node_exporter/releases/download/v1.6.1/node_exporter-1.6.1.linux-x86_64.tar.gz
```

Back at the Pipeline build: Build Failed, Docker Failed to log in

```
[Pipeline] End of Pipeline
ERROR: docker login failed
Finished: FAILURE
```

I ran commands:

```
sudo su
sudo usermod -aG docker jenkins
sudo systemctl restart jenkins
```

To add to the Docker Group

Installing Node Exporter:

Create a system user for Node Exporter and download Node Exporter:

```
sudo useradd --system --no-create-home --shell /bin/false node_exporter
wget https://github.com/prometheus/node_exporter/releases/download/v1.6.1/node_exporter-1.6.1.linux-amd64.tar.gz
```

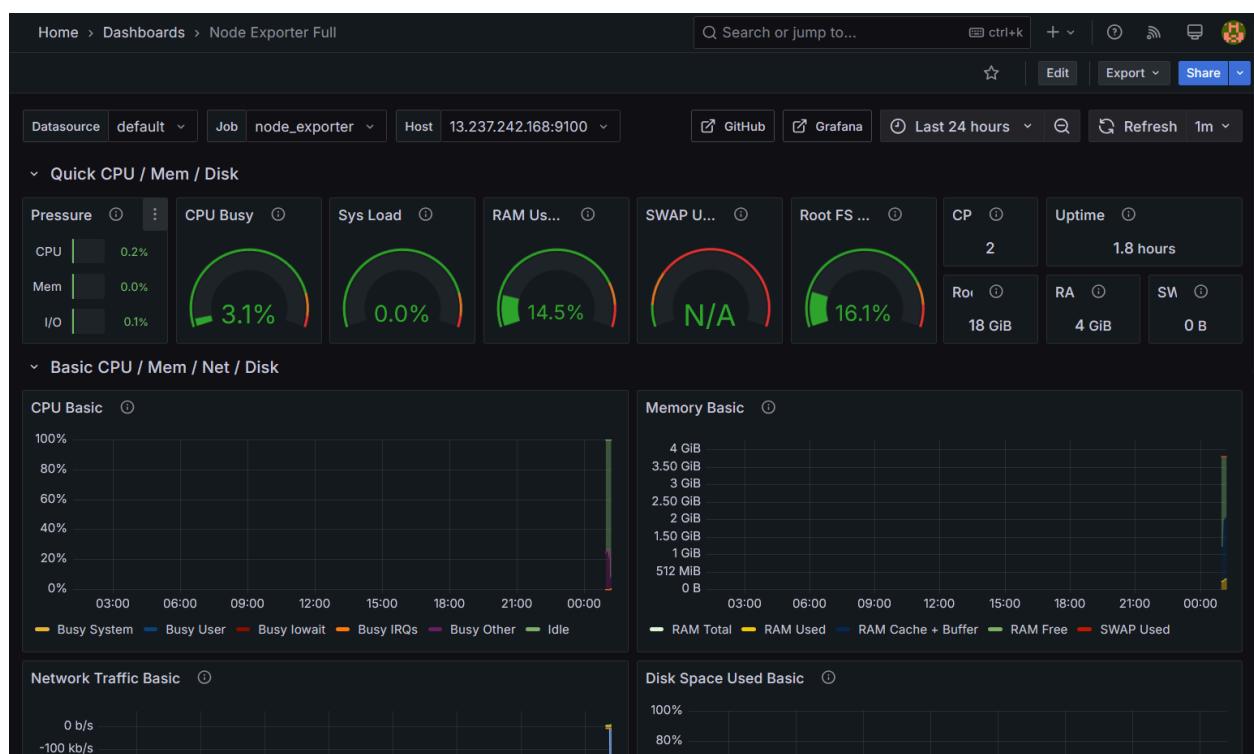
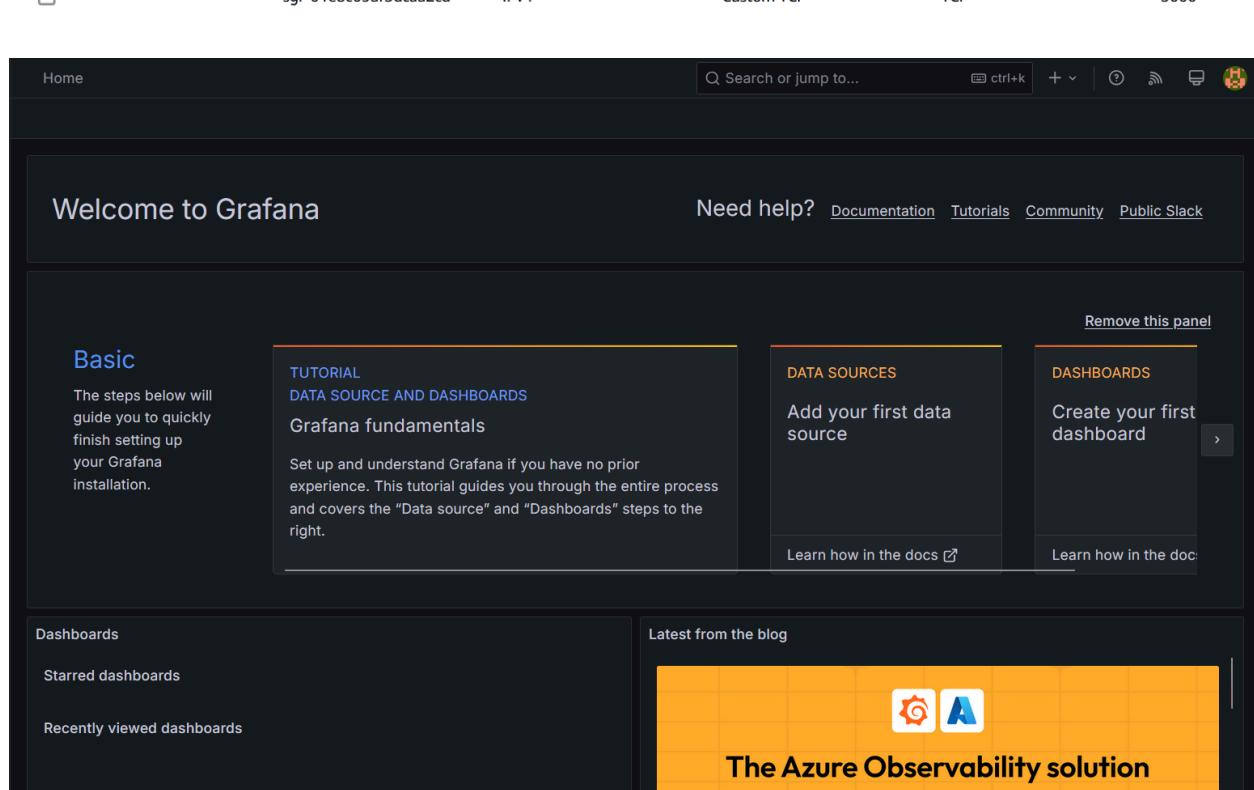
```
ubuntu@ip-172-31-12-9:~$ ls
node_exporter-1.6.1.linux-amd64.tar.gz  prometheus-2.47.1.linux-amd64  prometheus-2.47.1.linux-amd64.tar.gz
```

Targets						
All scrape pools ▾		All	Unhealthy	Collapse All	Filter by endpoint or labels	Scrape Duration
node_exporter (1/1 up) show less						
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error	
http://13.237.242.168:9100/metrics	UP	<code>instance="13.237.242.168:9100" job="node_exporter"</code>	9.693s ago	16.664ms		
prometheus (1/1 up) show less						
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error	
http://localhost:9090/metrics	UP	<code>instance="localhost:9090" job="prometheus"</code>	2.18s ago	3.788ms		

Install Grafana on Ubuntu 22.04 and Set it up to Work with Prometheus

Grafana is another tool that works with Prometheus that creates graphs out of provided metrics. I installed Grafana using the following commands.

```
sudo apt-get update
sudo apt-get install -y apt-transport-https software-properties-common
```



 **netflix**  Add description

SonarQube Quality Gate

Netflix Passed

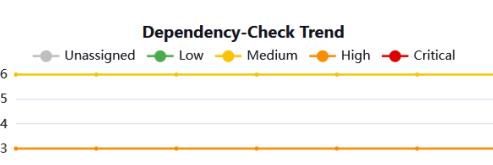
server-side processing: Success

 Latest Dependency-Check

Permalinks

- Last build (#9), 6 min 51 sec ago
- Last stable build (#9), 6 min 51 sec ago
- Last successful build (#9), 6 min 51 sec ago
- Last failed build (#9), 15 min ago

Dependency-Check Trend



The chart displays the trend of dependency checks over nine builds. The y-axis represents the severity level from 0 to 6. The x-axis shows build numbers from #3 to #9. A single orange line at level 3 indicates a constant High severity across all builds.

Build	Trend
#3	High
#4	High
#5	High
#6	High
#7	High
#8	High
#9	High

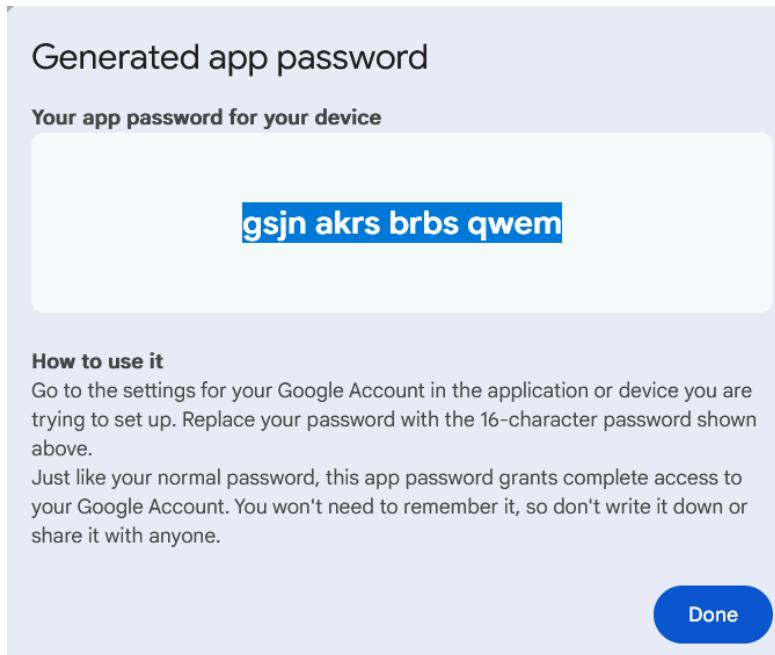
Targets		All scrape pools ▾	All	Unhealthy	Collapse All	<input type="text"/> Filter by endpoint or labels
jenkins (0/1 up)		show less				
Endpoint	State	Labels	Last Scrape	Scrape Duration	Error	
http://54.79.85.218:8080/prometheus	UNKNOWN	instance="54.79.85.218:8080" job="jenkins"	Never	0s		

The screenshot shows the Jenkins Performance and Health Overview dashboard. The top navigation bar includes links for Home, Dashboards, and Jenkins: Performance and Health Overview, along with search and filter options. The main content area is divided into several cards:

- Processing speed:** Shows a large white '0' on a dark background.
- Job queue duration:** A line chart showing duration over time (01:30 to 01:55). The y-axis ranges from 0 to 100. The chart shows a single data point at approximately 100 units.
- Queued rate:** Shows a large white '0' on a dark background.
- JVM free memory:** Shows a progress bar filled to 100% (67678370500.0%) with a green background.
- Memory Usage:** Shows 677 MB in blue text.
- Jenkins health:** Shows 1.0 in green text.
- JVM Uptime:** Shows N/A in red text.
- Jenkins nodes offline:** Shows None! in green text.
- CPU Usage:** Shows 0.00730% in green text.
- Total Jobs:** Shows None! in green text.
- Successful Jobs:** Shows None! in green text.
- Aborted Jobs:** Shows No data in green text.
- Unstable Jobs:** Shows No data in green text.
- Failed Jobs:** Shows None! in green text.
- Jenkins queue size:** Shows 0 in green text.

Implement Notification Services:

Email notifications were setup for Jenkins



E-mail Notification

SMTP server

smtp.gmail.com

Default user e-mail suffix ?

miguelhuerto@gmail.com

ⓘ This field should be '@' followed by a domain name.

Advanced ^ Edited

Use SMTP Authentication ?

User Name

miguelhuerto@gmail.com

⚠ For security when using authentication it is recommended to enable either TLS or SSL

Script ?

```

72     }
73   }
74 }
75 post {
76   success {
77     emailext to: "miguelhuerto@gmail.com",
78     subject: "Build Successful: ${env.JOB_NAME} #${env.BUILD_NUMBER}",
79     body: "Good news! Your build for ${env.JOB_NAME} was successful.\nCheck it here: ${env.BUILD_URL}"
80   }
81   failure {
82     emailext to: "miguelhuerto@gmail.com",
83     subject: "Build Failed: ${env.JOB_NAME} #${env.BUILD_NUMBER}",
84     body: "Oh no! Your build for ${env.JOB_NAME} failed.\nCheck the logs: ${env.BUILD_URL}"
85   }
86 }
87 }
88

```

Create Kubernetes Cluster with Nodegroups

The screenshot shows the AWS EKS console interface. On the left, there's a sidebar with navigation links for Amazon Elastic Kubernetes Service, Clusters, Amazon EKS Anywhere, Related services (Amazon ECR, AWS Batch), and links for Console settings, Documentation, and Submit feedback. The main area is titled 'Netflix' and shows a notification bar stating 'Add-on(s) metrics-server successfully added to cluster Netflix.' Below this, there's a message: 'The Amazon EKS console does not show Kubernetes resources until the cluster has an ACTIVE or UPDATING state. Please try again in a few minutes.' The 'Overview' tab is selected, displaying 'Cluster info' with status 'Creating', Kubernetes version '1.31', support period 'Standard support until November 26, 2025', and provider 'EKS'. It also shows 'Cluster health issues' (0), 'Upgrade insights' (0), and 'Node health issues' (0). Other tabs include Resources, Compute, Networking, Add-ons, Access, Observability, Update history, and Tags. A 'Details' section at the bottom contains API access credentials, OpenID Connect metadata URL, and a Created timestamp.

Monitor Kubernetes with Prometheus

Prometheus is then used to monitor the Kubernetes cluster. Additionally, node exporter was installed using Helm to collect metrics from the cluster nodes. Helm simplifies Kubernetes application deployment and management by acting as a package manager.

A Job was then added to prometheus.yml to Scrape Metrics on nodeip:9001/metrics i

```

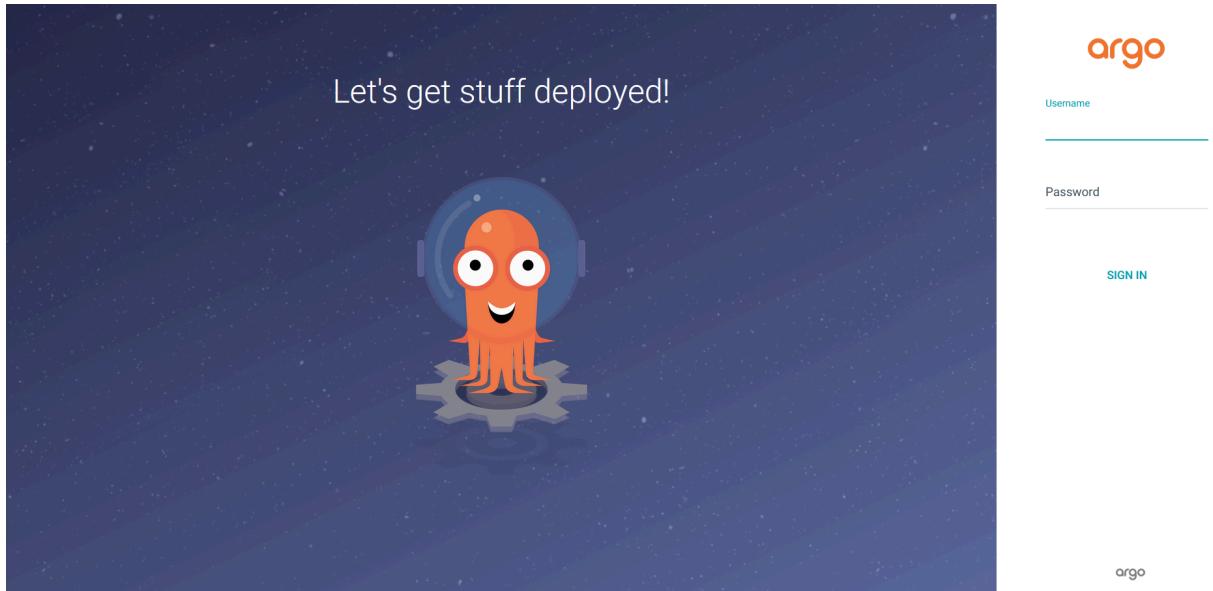
- job_name: 'Netflix'
  metrics_path: '/metrics'
  static_configs:
    - targets: ['node1Ip:9100']

```

Deploy Application with ArgoCD

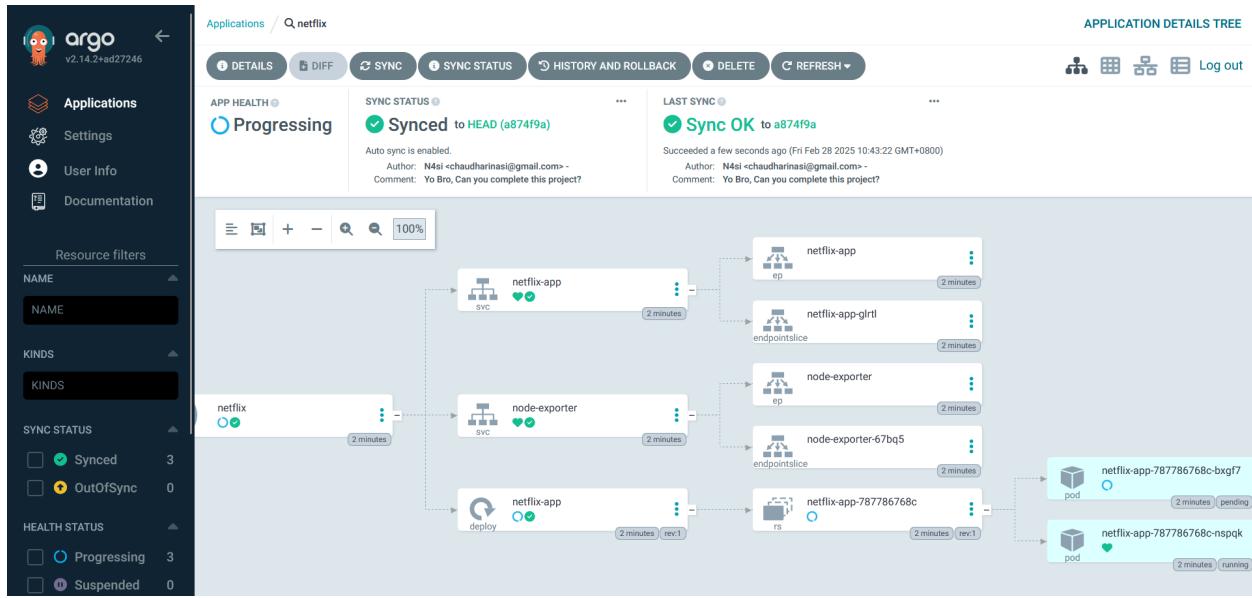
Argo CD is a declarative continuous delivery (CD) tool for Kubernetes that automates the deployment of applications by comparing the desired state defined in a Git repository with the current state of the cluster. Installing ArgoCD on the Kubernetes cluster:

```
kube-system      Active   12m
PS C:\Users\Miguel Huerto> kubectl create namespace argocd
namespace/argocd created
PS C:\Users\Miguel Huerto> kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/v2.1.0/install.yaml
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
serviceaccount/argocd-notifications-controller created
serviceaccount/argocd-redis created
serviceaccount/argocd-repo-server created
serviceaccount/argocd-server created
role.rbac.authorization.k8s.io/argocd-application-controller created
role.rbac.authorization.k8s.io/argocd-applicationset-controller created
role.rbac.authorization.k8s.io/argocd-dex-server created
role.rbac.authorization.k8s.io/argocd-notifications-controller created
role.rbac.authorization.k8s.io/argocd-redis created
role.rbac.authorization.k8s.io/argocd-server created
clusterrole.rbac.authorization.k8s.io/argocd-application-controller created
clusterrole.rbac.authorization.k8s.io/argocd-applicationset-controller created
clusterrole.rbac.authorization.k8s.io/argocd-server created
rolebinding.rbac.authorization.k8s.io/argocd-application-controller created
rolebinding.rbac.authorization.k8s.io/argocd-applicationset-controller created
rolebinding.rbac.authorization.k8s.io/argocd-dex-server created
rolebinding.rbac.authorization.k8s.io/argocd-notifications-controller created
rolebinding.rbac.authorization.k8s.io/argocd-redis created
```



The main application management screen. On the left is a sidebar with icons for "Applications" (selected), "Settings", "User Info", and "Documentation". The main area is titled "Applications" and contains buttons for "+ NEW APP", "SYNC APPS", and "REFRESH APPS". A search bar with placeholder text "Search applications..." is present. To the right of the search bar are icons for "APPLICATIONS TILES" (grid view), "LIST VIEW" (list view), and "Log out". The center of the screen shows a large circular icon with three stacked layers, indicating no applications are currently listed. Below this icon, the text "No applications available to you just yet" is displayed, followed by the instruction "Create new application to start managing resources in your cluster" and a "CREATE APPLICATION" button.

Create the ArgoCD Application



Accessing the working application on port 3007

