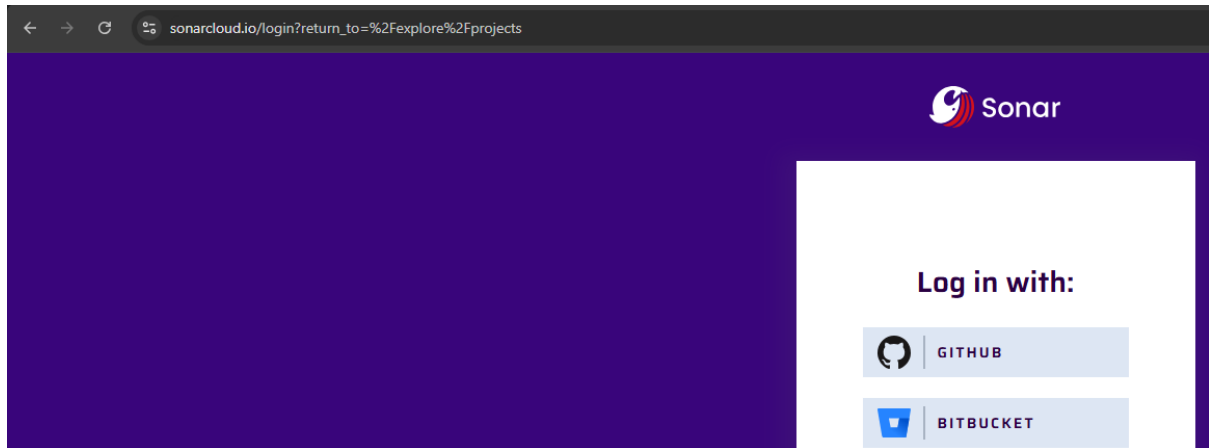
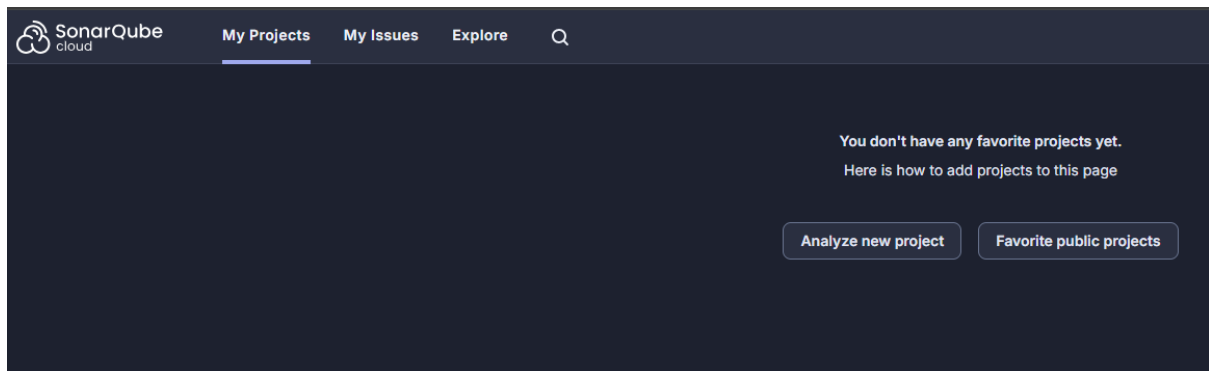


Sonar Cloud

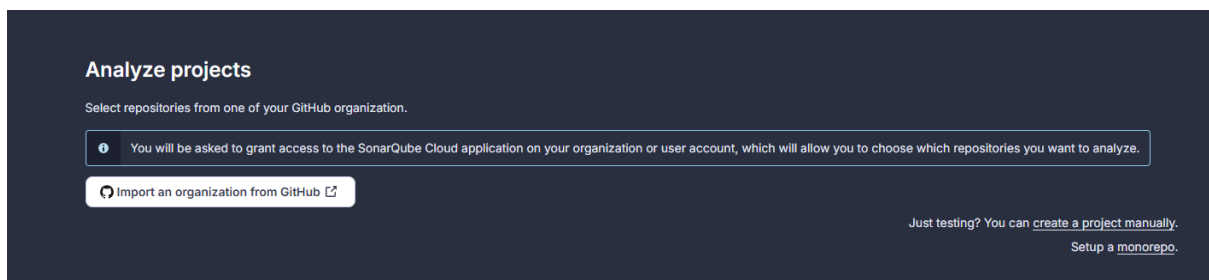
Login Into Sonar Cloud using GITHUB



Analyse New Project





Import from your GIT HUB



Give Organization Name and Select Free Version


1

Import organization details

Import  Aayush Doshi into a SonarQube Cloud organization 

Name *


AD_1507



Up to 255 characters

Key * ?

ad-1507



Organization key must start with a lowercase letter or number, followed by lowercase letters, numbers or hyphens, and must end with a letter or number. Maximum length: 255 characters.

[Add additional info](#)

2

Choose a plan

Free

For individual developers, students and open-source projects

\$0

Select Free

Team

Free trial available

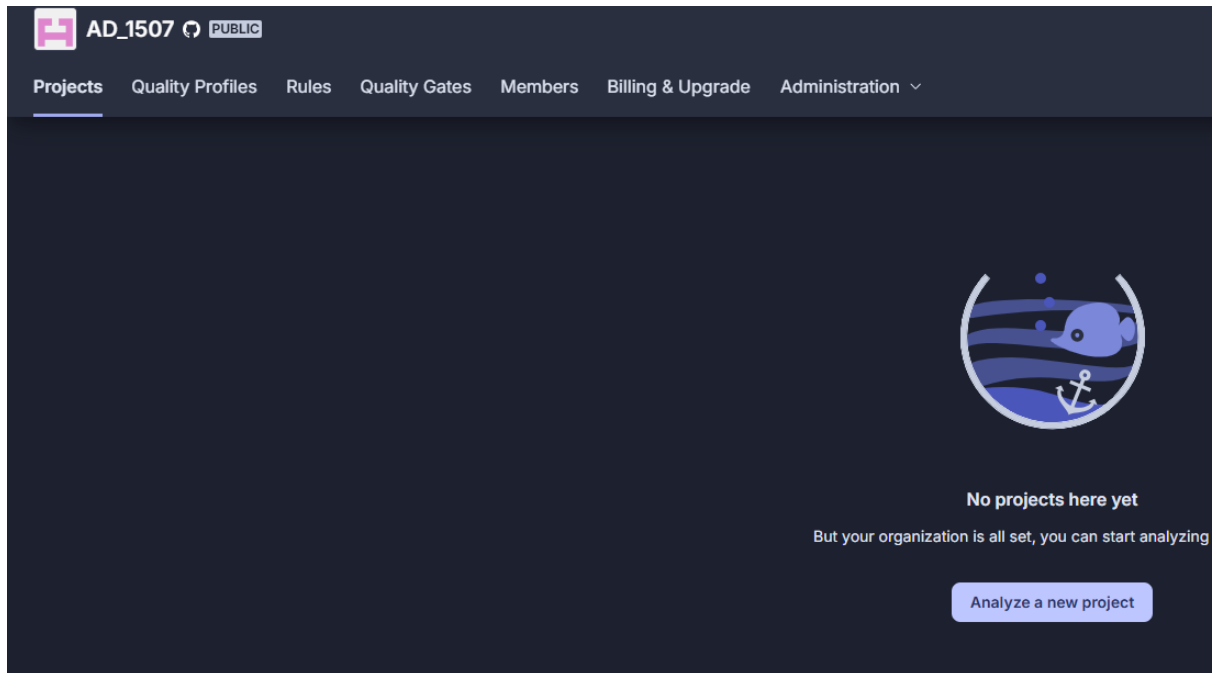
Essential capabilities for small business

Starting at

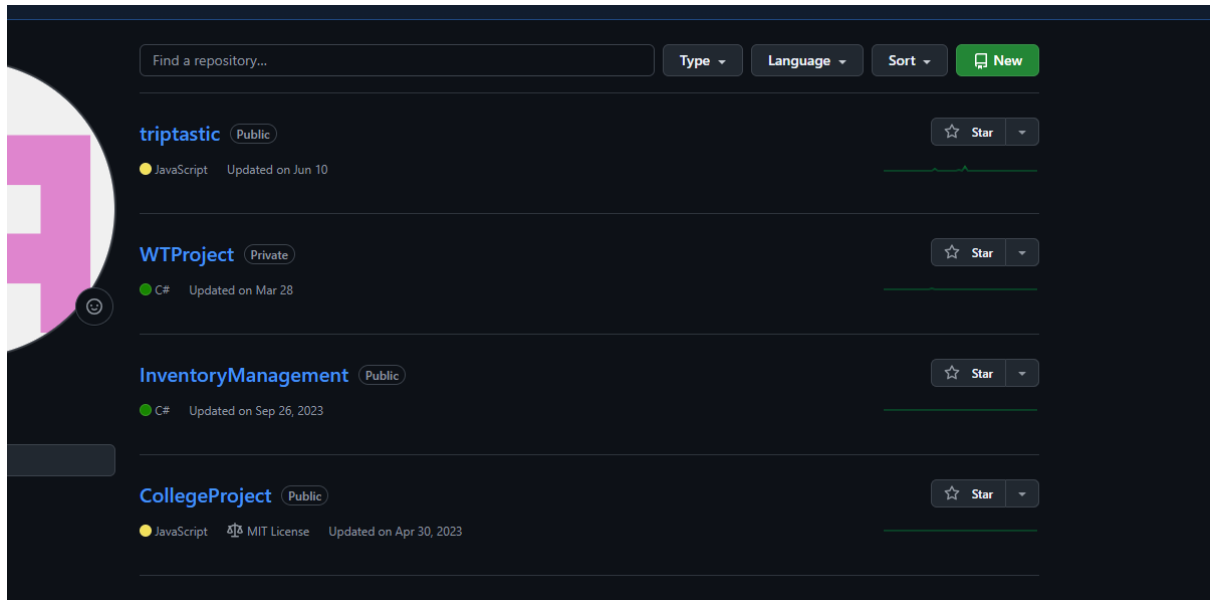
~~\$64~~ \$32/month • 50% off

Start Free Trial

Analyze New Project



Go to Git Hub and Click on New to Create a New repository




Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere?

[Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *

 AayushD15

Repository name *

DevOps

✔ DevOps is available.

Great repository names are short and memorable. Need inspiration? How about [literate-chainsaw](#) ?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: **None**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **None**

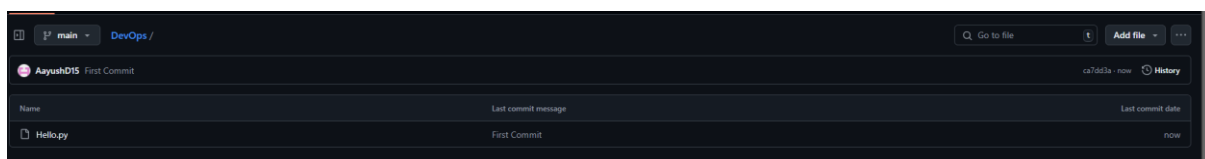
A license tells others what they can and can't do with your code. [Learn more about licenses.](#)



You are creating a public repository in your personal account.

Create repository

Add a new File from Add File Option on Top Right Cor



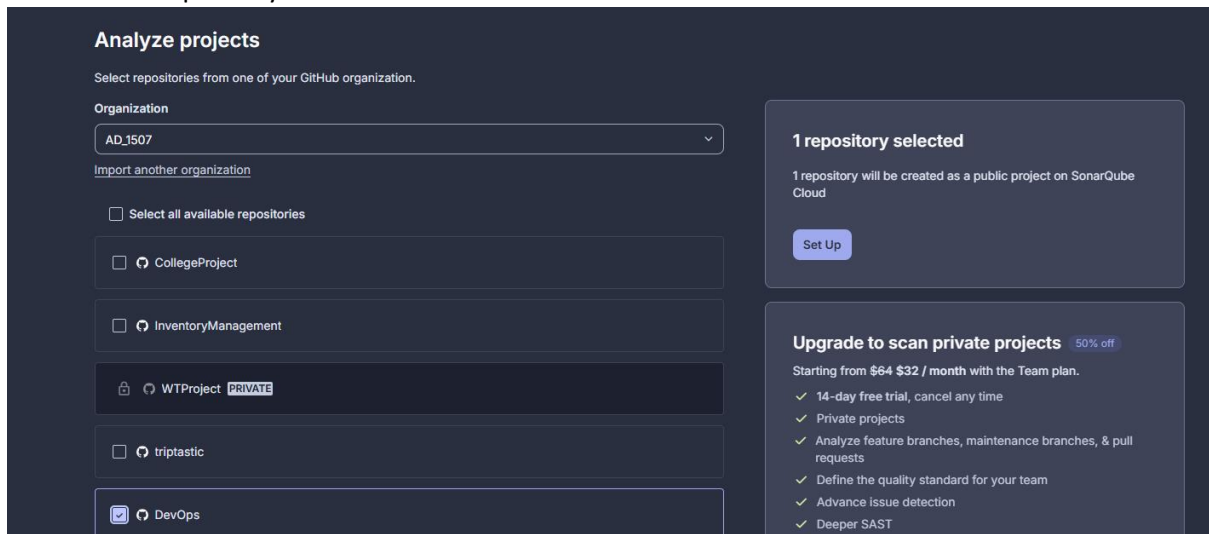
In the Root Directory create a normal python file



The screenshot shows a GitHub repository interface. At the top, there's a navigation bar with a file icon, a dropdown menu set to 'main', and the repository name 'DevOps / Hello.py'. Below this, a commit header shows the user 'AayushD15' and the message 'First Commit'. The main content area has tabs for 'Code' and 'Blame'. The 'Code' tab is active, displaying a Python file with 6 lines of code. The code defines a function 'say_hello()' that prints 'Hello, World!' and then calls this function. A badge on the right indicates 'Code 55% faster with GitHub'.

```
1 # Function to print "Hello, World!"
2 def say_hello():
3     print("Hello, World!")
4
5 # Call the function
6 say_hello()
```

Click on the Repository in Sonar Cloud



The screenshot displays the 'Analyze projects' page in SonarCloud. It prompts the user to 'Select repositories from one of your GitHub organization.' The 'Organization' dropdown is set to 'AD_1507'. Below this, there's a list of repositories with checkboxes for selection. The 'DevOps' repository is checked. To the right, a sidebar indicates '1 repository selected' and provides a 'Set Up' button. Another section promotes an 'Upgrade to scan private projects' with a 50% off discount, listing benefits like a 14-day free trial, private projects support, and deeper SAST.

Analyze projects

Select repositories from one of your GitHub organization.

Organization: AD_1507

[Import another organization](#)

☐ Select all available repositories

- ☐ CollegeProject
- ☐ InventoryManagement
- ☒ WTPProject **PRIVATE**
- ☐ triptastic
- ☒ DevOps

1 repository selected

1 repository will be created as a public project on SonarQube Cloud

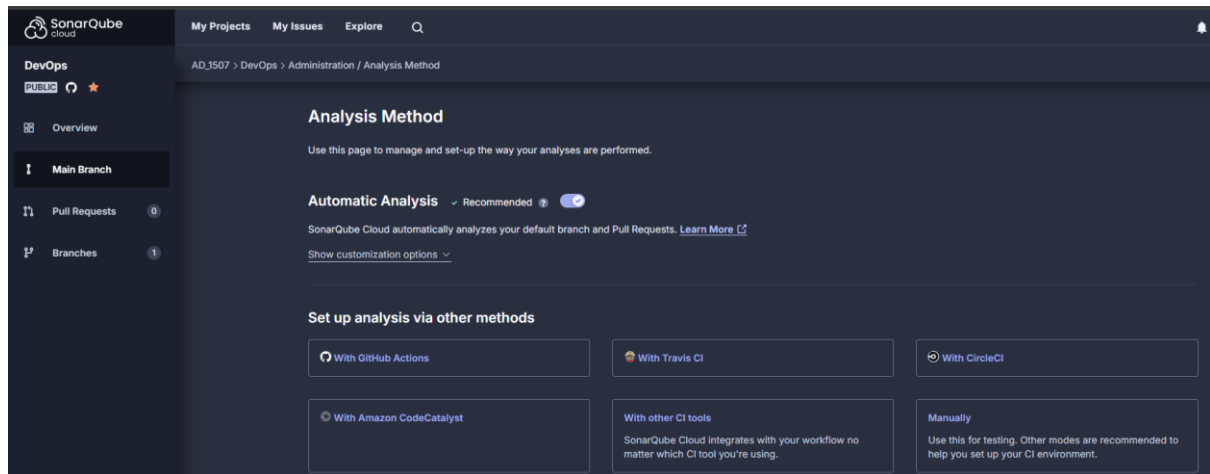
[Set Up](#)

Upgrade to scan private projects 50% off

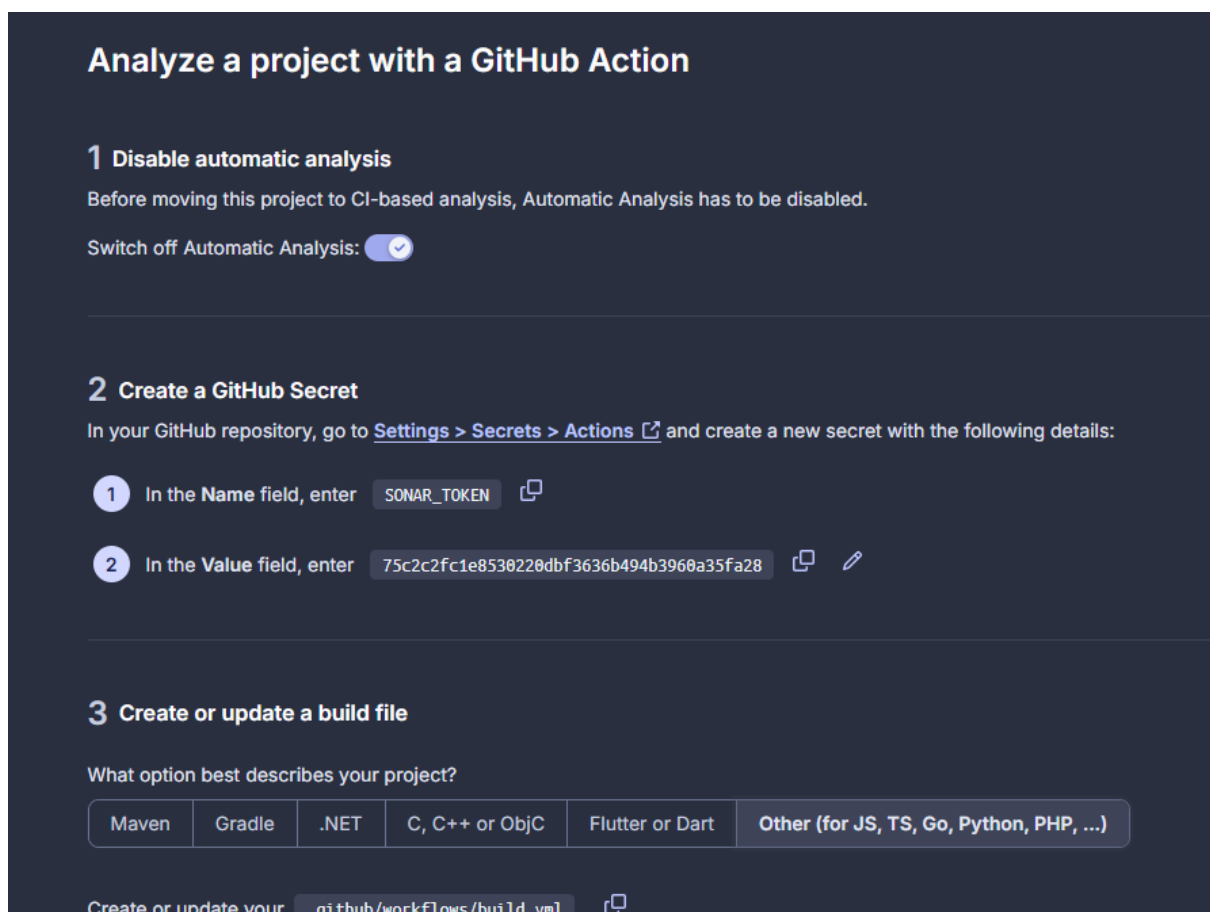
Starting from \$64 \$32 / month with the Team plan.

- ✓ 14-day free trial, cancel any time
- ✓ Private projects
- ✓ Analyze feature branches, maintenance branches, & pull requests
- ✓ Define the quality standard for your team
- ✓ Advance issue detection
- ✓ Deeper SAST


In the Left Hand Panel click on Administration and click on Analysis Method and Click on



Copy the Name: **SONAR_TOKEN** and Token and click on Python on 3rd Step




Create a file in Github Repository and name should be → **.github/workflows/build.yml**

Create or update your `.github/workflows/build.yml` 


Here is a base configuration to run a SonarQube Cloud analysis on your master branch and Pull Requests. If you already have some GitHub Actions, you might want to just add some of these new steps to an existing one.

```
name: Build
on:
  push:
    branches:
      - main
  pull_request:
    types: [opened, synchronize, reopened]
jobs:
  sonarcloud:
    name: SonarQube Cloud
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v4
        with:
          fetch-depth: 0 # Shallow clones should be disabled for a better relevancy of analysis
      - name: SonarQube Cloud Scan
        uses: SonarSource/sonarcloud-github-action@master
        env:
          GITHUB_TOKEN: ${ secrets.GITHUB_TOKEN } # Needed to get PR information, if any
          SONAR_TOKEN: ${ secrets.SONAR_TOKEN }
```

 Copy

Create Another File with name → **sonar-project.properties**

4 Create a `sonar-project.properties` file


Create a configuration file in the root directory of the project and name it `sonar-project.properties` 

```
sonar.projectKey=Aayush015_DevOps
sonar.organization=ad-1507

# This is the name and version displayed in the SonarCloud UI.
#sonar.projectName=DevOps
#sonar.projectVersion=1.0

# Path is relative to the sonar-project.properties file. Replace "\" by "/" on Windows.
#sonar.sources=.

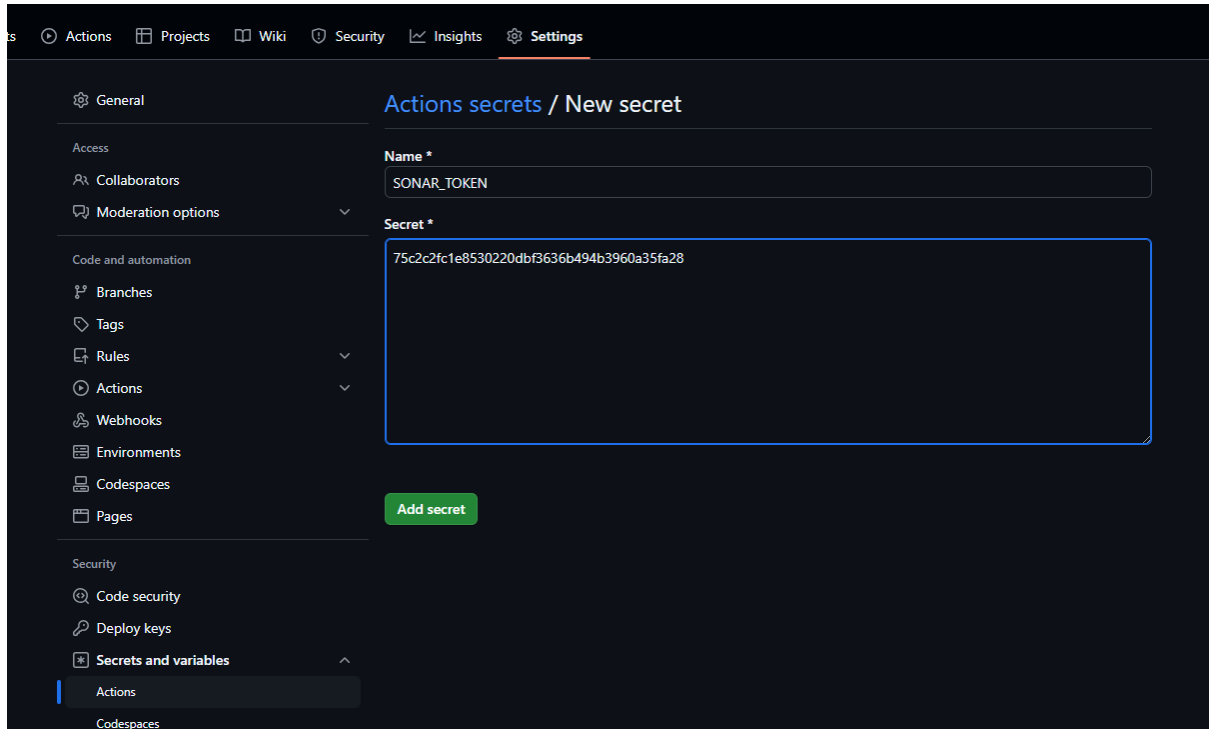
# Encoding of the source code. Default is default system encoding
#sonar.sourceEncoding=UTF-8
```

 Copy

In Git Hub Repository

Go to Settings → Secrets and Variables → Actions

Copy the Name and Value here



The screenshot shows the GitHub repository settings page, specifically the 'Actions secrets / New secret' section. The left sidebar contains a navigation menu with categories: General, Access, Code and automation, and Security. Under 'Access', there are links for Collaborators, Moderation options, and a dropdown for Code and automation (which includes Branches, Tags, Rules, Actions, Webhooks, Environments, Codespaces, and Pages). Under 'Security', there are links for Code security, Deploy keys, and a dropdown for Secrets and variables (which includes Actions and Codespaces). The main content area is titled 'Actions secrets / New secret' and contains a form with two fields: 'Name *' with the value 'SONAR_TOKEN' and 'Secret *' with the value '75c2c2fc1e8530220dbf3636b494b3960a35fa28'. A green 'Add secret' button is located below the form.

ts Actions Projects Wiki Security Insights Settings

General

Access

Collaborators

Moderation options

Code and automation

Branches

Tags

Rules

Actions

Webhooks

Environments

Codespaces

Pages

Security

Code security

Deploy keys

Secrets and variables

Actions

Codespaces

Actions secrets / New secret

Name *

SONAR_TOKEN

Secret *

75c2c2fc1e8530220dbf3636b494b3960a35fa28

Add secret

Terraform

- **Command**

docker-compose run terraform init

docker-compose run terraform plan

docker-compose run terraform apply -auto-approve

- **Create Main.TF file in a Directory**

Configure the Docker provider

```
terraform {  
  required_providers {  
    docker = {  
      source = "kreuzwerker/docker"  
      version = "~> 2.0"  
    }  
  }  
}  
  
provider "docker" {  
  host = "unix:///var/run/docker.sock"  
}  
  
# Pull Nginx image from DockerHub  
resource "docker_image" "nginx" {  
  name      = "nginx:latest"  
  keep_locally = false  
}
```

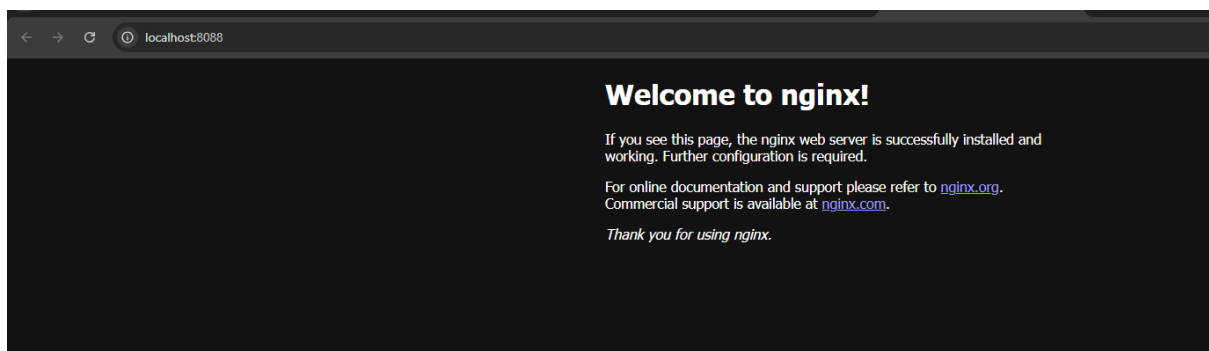
Create an Nginx container

```
resource "docker_container" "nginx" {  
  image = docker_image.nginx.image_id  
  name = "Devops_Final_Exam"  
  
  ports {  
    internal = 80  
    external = 8088  
  }  
}
```

- **docker-compose.yml**

```
version: '3.8'  
  
services:  
  terraform:  
    image: hashicorp/terraform:latest  
    container_name: terraform  
    volumes:  
      - E:/Aayush/DevOps/Terraform:/workspace # Adjust this path based on  
your local directory  
      - /var/run/docker.sock:/var/run/docker.sock # This allows Terraform to  
interact with Docker  
    working_dir: /workspace  
    entrypoint: ["terraform"]  
    command: ["init"] # Default command (can be overridden)
```

- **Output**



Docker

Index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Final Examination</title>
</head>
<body>
  <h1>Docker Question</h1>
  <p>MCA Final Examnation </p>
</body>
</html>
```

dockerfile

```
# Use the official Nginx image
FROM nginx:alpine

# Copy the index.html to the appropriate directory in the container
COPY index.html /usr/share/nginx/html/index.html

# Expose port 80 to access the web page
EXPOSE 8080
```

Commands

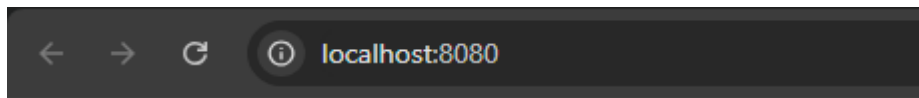
Create Image

- docker build -t firstimage .

Create Container and Run Image

- docker run -d -p 8080:80 firstimage

Output



Docker Question

MCA Final Examination

Kubernetes

For Nginx Image

Deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: public-deployment
spec:
  replicas: 2 # Start with 2 replicas
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest # Public Nginx image
          ports:
            - containerPort: 80
```

Service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: public-service
spec:
  selector:
```

```
  app: nginx
  ports:
  - protocol: TCP
    port: 80 # Exposing port 80 externally
    targetPort: 80 # Port inside the container
  type: NodePort # Use NodePort for local access
```

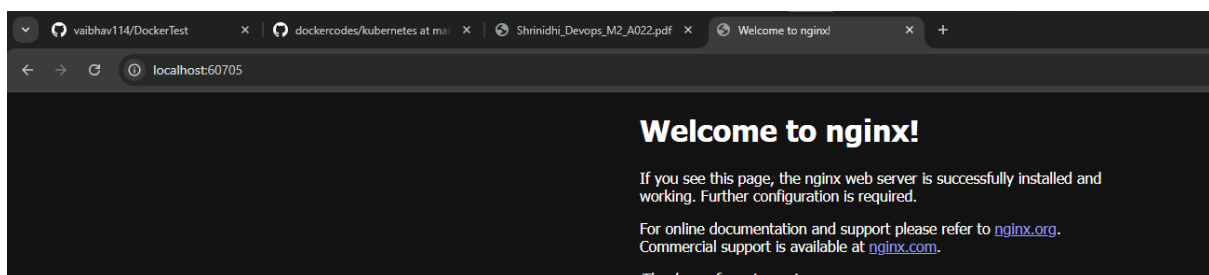
Commands

- `kubectl apply -f Deployment.yaml`
- `kubectl apply -f Service.yaml`

- `kubectl get deployments`
- `kubectl get services`

Forward the Port in Rancher Desktop

Output



For Custom Image

dockerfile

```
# Use the official Nginx image
FROM nginx:alpine

# Copy the index.html to the appropriate directory in the container
COPY index.html /usr/share/nginx/html/index.html

# Expose port 80 to access the web page
EXPOSE 8080
```

Index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Final Examination</title>
</head>
<body>
  <h1>Docker Question</h1>
  <p>MCA Final Examination </p>
</body>
</html>
```

Deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: custom-deployment
spec:
  replicas: 2 # Initial number of replicas
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: aayush1507/secondimage:latest
          ports:
            - containerPort: 80
```

Service.yaml

```
apiVersion: v1
```

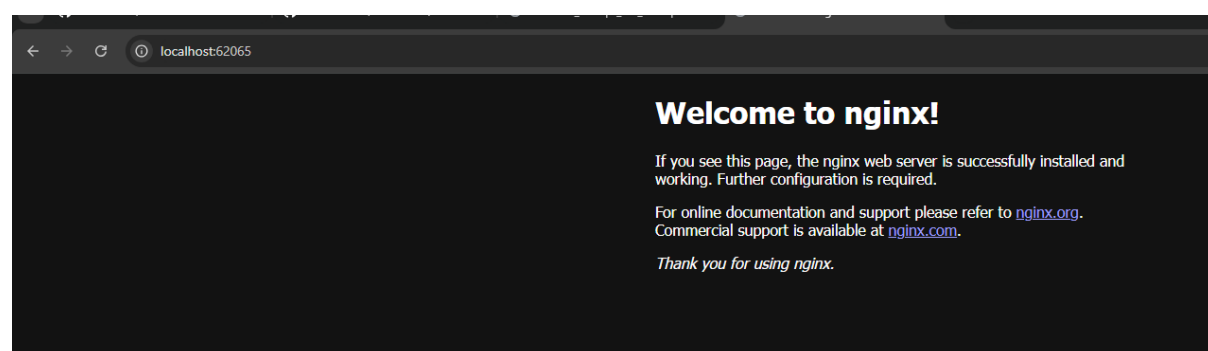
```
kind: Service
metadata:
  name: custom-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80 # Exposing port 80
      targetPort: 80 # Inside the container
  type: NodePort # Or you can use NodePort if LoadBalancer isn't available
```

Commands

- docker login
- docker build -t aayush1507/secondimage:latest .
- docker push aayush1507/secondimage:latest
- kubectl apply -f Deployment.yaml
- kubectl apply -f Service.yaml
- kubectl get services
- kubectl get deployments

Output

```
PS E:\Aayush\DevOps\Kubernetes\Custom> kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
custom-deployment   2/2     2             2           5m59s
public-deployment   2/2     2             2           34m
PS E:\Aayush\DevOps\Kubernetes\Custom> kubectl get services
NAME                TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)          AGE
custom-service      NodePort    10.43.70.162 <none>        80:30790/TCP     5m55s
kubernetes          ClusterIP   10.43.0.1    <none>        443/TCP          36m
public-service      NodePort    10.43.167.230 <none>        80:30610/TCP     34m
PS E:\Aayush\DevOps\Kubernetes\Custom>
```



Jenkins

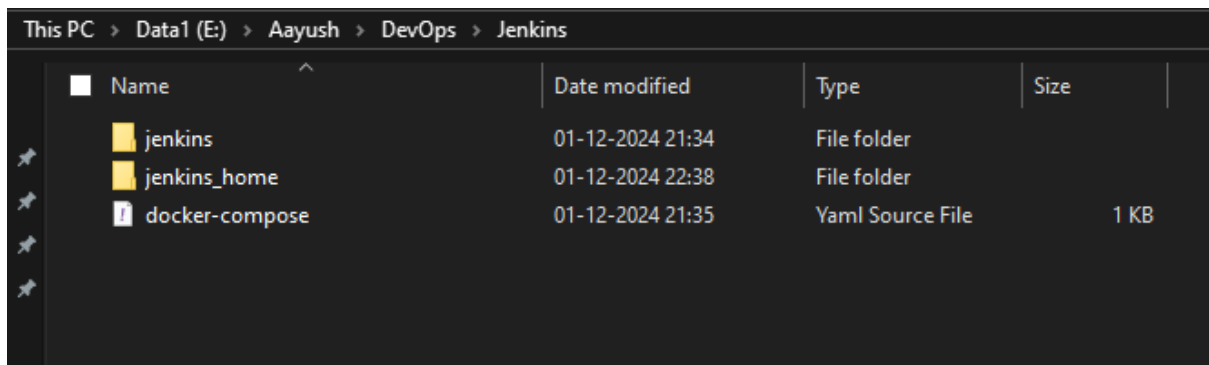
Create a Project Directory

Jenkins

Inside Project Directory

Create a Folder **jenkins**

Just like below



The screenshot shows a Windows File Explorer window with the address bar displaying the path: This PC > Data1 (E:) > Aayush > DevOps > Jenkins. The main area shows a list of files and folders with columns for Name, Date modified, Type, and Size.

Name	Date modified	Type	Size
jenkins	01-12-2024 21:34	File folder	
jenkins_home	01-12-2024 22:38	File folder	
docker-compose	01-12-2024 21:35	Yaml Source File	1 KB

Create docker-compose.yml

```
version: '3.8'
services:
  jenkins:
    build:
      context: ./jenkins # This is where your Jenkins-specific Dockerfile is
                        located
      dockerfile: Dockerfile # This references the Dockerfile inside the
                        jenkins folder
    privileged: true
    user: root
    ports:
      - "8083:8080"
      - "50000:50000"
    container_name: jenkins
```



```

volumes:
  - E:/Aayush/DevOps/Jenkins/jenkins_home:/var/jenkins_home
  - /var/run/docker.sock:/var/run/docker.sock
networks:
  - jenkins-net

networks:
  jenkins-net:
    driver: bridge

volumes:
  jenkins_home:

```

inside the jenkins folder create a dockerfile

```

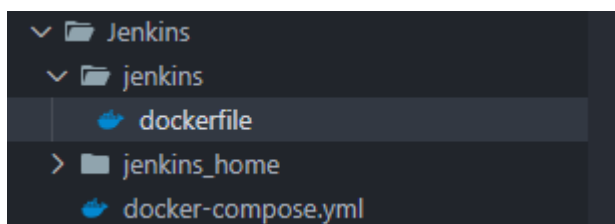
FROM jenkins/jenkins:lts
USER root

# Install Docker CLI and dependencies
RUN apt-get update && \
    apt-get -y install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg2 \
    software-properties-common && \
    curl -fsSL https://download.docker.com/linux/debian/gpg | apt-key add - && \
    add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/debian \
    $(lsb_release -cs) \
    stable" && \
    apt-get update && \
    apt-get -y install docker.io

# Switch back to jenkins user
USER jenkins

```

Folder Structure



Command

Run the below command in both the Root Directory and jenkins directory

`docker-compose up --build -d`

Once Executed the above command

The below URL will be Jenkins URL

<http://localhost:8083/>

To Login it will ask for initialpassword it will be in

Jenkins_home → secrets → initialAdminPassword

Once logged in click on Manage Jenkins on Left Hand Side

Click on Plugin

Search Docker and select below and install



Go to Dashboard → New Items

Give a Name and Select Pipeline

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:

Copy from

OK

In the Script Section paste the below script and click on Save

```
pipeline {
  agent any

  environment {
    DOCKER_IMAGE = 'nginx'
    CONTAINER_NAME = 'nginx-container'
  }

  stages {
    stage('Pull Nginx Docker Image') {
      steps {
```

```

    script {
        // Pull the Nginx image from Docker Hub
        sh 'docker pull $DOCKER_IMAGE'
    }
}

stage('Run Docker Container') {
    steps {
        script {
            // Run the Nginx container on port 8080
            sh 'docker run -d --name $CONTAINER_NAME -p 8080:80 $DOCKER_IMAGE'
        }
    }
}

post {
    always {
        echo 'Pipeline finished.'
    }
    success {
        echo 'Deployment succeeded.'
    }
    failure {
        echo 'Deployment failed.'
    }
}
}

```

Pipeline

Definition

Pipeline script

Script ?

```

1 pipeline {
2     agent any
3
4     environment {
5         DOCKER_IMAGE = 'nginx'
6         CONTAINER_NAME = 'nginx-container'
7     }
8
9     stages {
10        stage('Pull Nginx Docker Image') {
11            steps {
12                script {
13                    // Pull the Nginx image from Docker Hub
14                    sh 'docker pull $DOCKER_IMAGE'
15                }
16            }
17        }
18    }
19 }

```

☒ Use Groovy Sandbox ?

Save

Apply

Click on Build Now

Dashboard > All > test1 >

Status

</> Changes

▶ Build Now

⚙️ Configure

🗑️ Delete Pipeline

📁 Stages

✎ Rename

❓ Pipeline Syntax

✓ test1

Permalinks

- [Last build \(#3\), 15 min ago](#)
- [Last stable build \(#3\), 15 min ago](#)
- [Last successful build \(#3\), 15 min ago](#)
- [Last completed build \(#3\), 15 min ago](#)

Builds

⋮

🔍 Filter

Today

✓ #3 5:08 PM

In the Docker Desktop

The below container will be created

name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
nginx-container	13f55e4eecb6	nginx:<none>	8080:80	0%	16 minutes ago	

Output

< > ↻ localhost:8080

Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

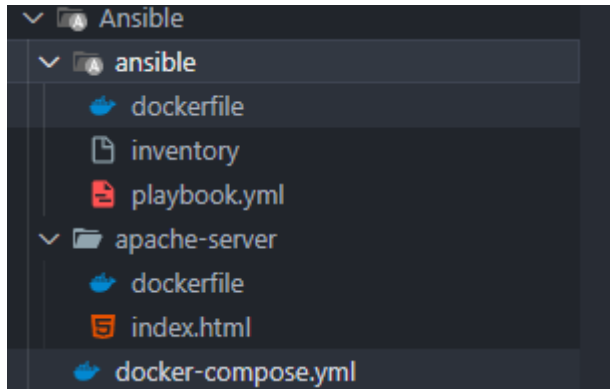
For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Ansible

Commands

docker-compose up --build -d



Ansible → dockerfile

```
# ansible/Dockerfile
FROM ubuntu:20.04

# Install Ansible and SSH client for remote communication
RUN apt-get update && \
    apt-get install -y ansible sshpass python3-pip && \
    apt-get clean

# Install necessary Ansible dependencies
RUN pip3 install ansible

# Set working directory
WORKDIR /ansible

# Copy the playbook and inventory into the container
COPY playbook.yml /ansible/playbook.yml

# Entry point for Ansible to run the playbook
CMD ["ansible-playbook", "/ansible/playbook.yml", "-i", "inventory"]
```

Playbook.yml

```
# ansible/playbook.yml
---
- name: Install Apache and ensure it is running
  hosts: apache-server
  become: true
  tasks:
    - name: Install Apache
      apt:
```

```
    name: apache2
    state: present
    update_cache: yes

- name: Start Apache service
  service:
    name: apache2
    state: started
    enabled: yes
```

inventory

[apache-server]

apache-server ansible_host=apache-server ansible_port=22

[apache-server:vars]

ansible_ssh_user=root

ansible_ssh_pass=password

apache-server → dockerfile

```
FROM ubuntu:20.04

# Minimize layer creation and reduce build time
RUN set -xe \
    && apt-get update \
    && DEBIAN_FRONTEND=noninteractive apt-get install -y --no-install-
recommends \
    apache2 \
    openssh-server \
    && rm -rf /var/lib/apt/lists/* \
    && mkdir /var/run/sshd \
    && echo 'root:password' | chpasswd \
    && sed -i 's/^PermitRootLogin prohibit-password/PermitRootLogin yes/'
/etc/ssh/sshd_config

# Expose Apache and SSH ports
EXPOSE 80 22

# Create a basic index.html for Apache
COPY index.html /var/www/html/index.html

# Start SSH and Apache in the background and keep the container running
CMD service ssh start && apache2ctl -D FOREGROUND
```

Index.html

```
<!-- apache-server/index.html -->
<html>
  <head>
    <title>Apache Server</title>
  </head>
  <body>
    <h1>Welcome to Apache!</h1>
  </body>
</html>
```

docker-compose.yml











```
# docker-compose.yml
version: '3.8'

services:
  apache-server:
    build: ./apache-server
    container_name: apache-server
    ports:
      - "8085:80"
    networks:
      - app-network

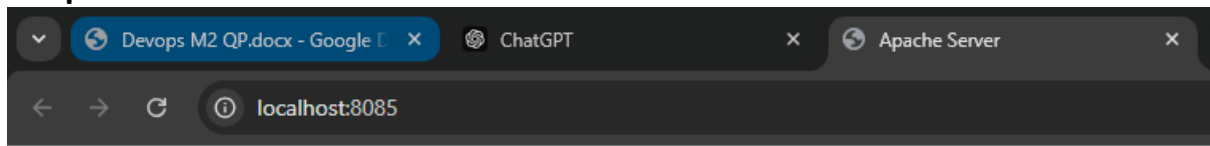
  ansible:
    build: ./ansible
    container_name: ansible-container
    depends_on:
      - apache-server
    networks:
      - app-network

networks:
  app-network:
    driver: bridge
```

Once ran the command

<input type="checkbox"/>	Name	Container ID	Image	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	ansible	-	-	-	0.01%	4 minutes ago	  
<input type="checkbox"/>	apache-server	867c82ba1bb3	ansible-apache-server:<none>	8085:80 	0.01%	4 minutes ago	  
<input type="checkbox"/>	container	a8cfff4e452ea	ansible-ansible:<none>	-	0%	4 minutes ago	  

Output



Welcome to Apache!