Day 12

Jenkin

Build a Docker Jenkins Pipeline to Implement CI/CD Workflow.

Course-end Project 2

Description

Demonstrate the continuous integration and delivery by building a Docker Jenkins Pipeline.

Problem Statement Scenario:

You are a DevOps consultant in AchiStar Technologies. The company decided to implement DevOps to develop and deliver their products. Since it is an Agile organization, it follows Scrum methodology to develop the projects incrementally. You are working with multiple DevOps Engineers to build a Docker Jenkins Pipeline. During the sprint planning, you agreed to take the lead on this project and plan on the requirements, system configurations, and track the efficiency. The tasks you are responsible for:

- Availability of the application and its versions in the GitHub
    - Track their versions every time a code is committed to the repository
- Create a Docker Jenkins Pipeline that will create a Docker image from the Dockerfile and host it on Docker Hub
- It should also pull the Docker image and run it as a Docker container
- Build the Docker Jenkins Pipeline to demonstrate the continuous integration and continuous delivery workflow

Company goal is to deliver the product frequently to the production with high-end quality.

You must use the following tools:

- Docker: To build the application from a Dockerfile and push it to Docker Hub
- Docker Hub: To store the Docker image
- GitHub: To store the application code and track its revisions
- Git: To connect and push files from the local system to GitHub
- Linux (Ubuntu): As a base operating system to start and execute the project
- Jenkins: To automate the deployment process during continuous integration

Following requirements should be met:

- Document the step-by-step process from the initial installation to the final stage
- Track the versions of the code in the GitHub repository
- Availability of the application in the Docker Hub
- Track the build status of Jenkins for every increment of the project

Index.html or java or python

```
<html>
<head>
</head>
<body>
<p>Welcome to HTML Web Page</p>
</body>
</html>
```

With simple message

Dockerfile for index.html app with help of nginx server

Push this code to git hub

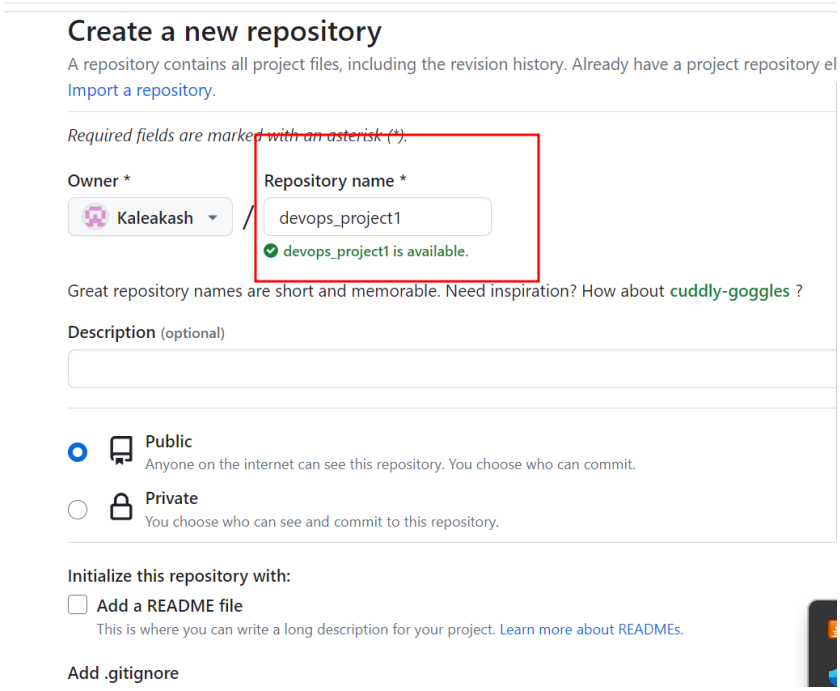
Git can be in your local machine or VM machine etc.

Dockerfile

```
FROM nginx
COPY index.html /usr/share/nginx/html
```

```
MINGW64:/c/Users/akash/Desktop
akash@Hp MINGW64 ~/Desktop/D
$ git init
Initialized empty Git repository in C:/Users/akash/Desktop/DevOps App/.git/

akash@Hp MINGW64 ~/Desktop/DevOps App (master)
$ ls -a
./  ../  .git/  Dockerfile  index.html

akash@Hp MINGW64 ~/Desktop/DevOps App (master)
$ git add .

akash@Hp MINGW64 ~/Desktop/DevOps App (master)
$ git commit -m "initial commit"
[master (root-commit) ca8d9fc] initial commit
 2 files changed, 14 insertions(+)
 create mode 100644 Dockerfile
 create mode 100644 index.html

akash@Hp MINGW64 ~/Desktop/DevOps App (master)
$ git config --global user.email "akash300383@gmail.com"

akash@Hp MINGW64 ~/Desktop/DevOps App (master)
$ git config --global user.name "akash"

akash@Hp MINGW64 ~/Desktop/DevOps App (master)
$ git commit -m "Next commit"
On branch master
nothing to commit, working tree clean

akash@Hp MINGW64 ~/Desktop/DevOps App (master)
$ git status
On branch master
nothing to commit, working tree clean

akash@Hp MINGW64 ~/Desktop/DevOps App (master)
$
```

Now we need to create remote repository in github account.



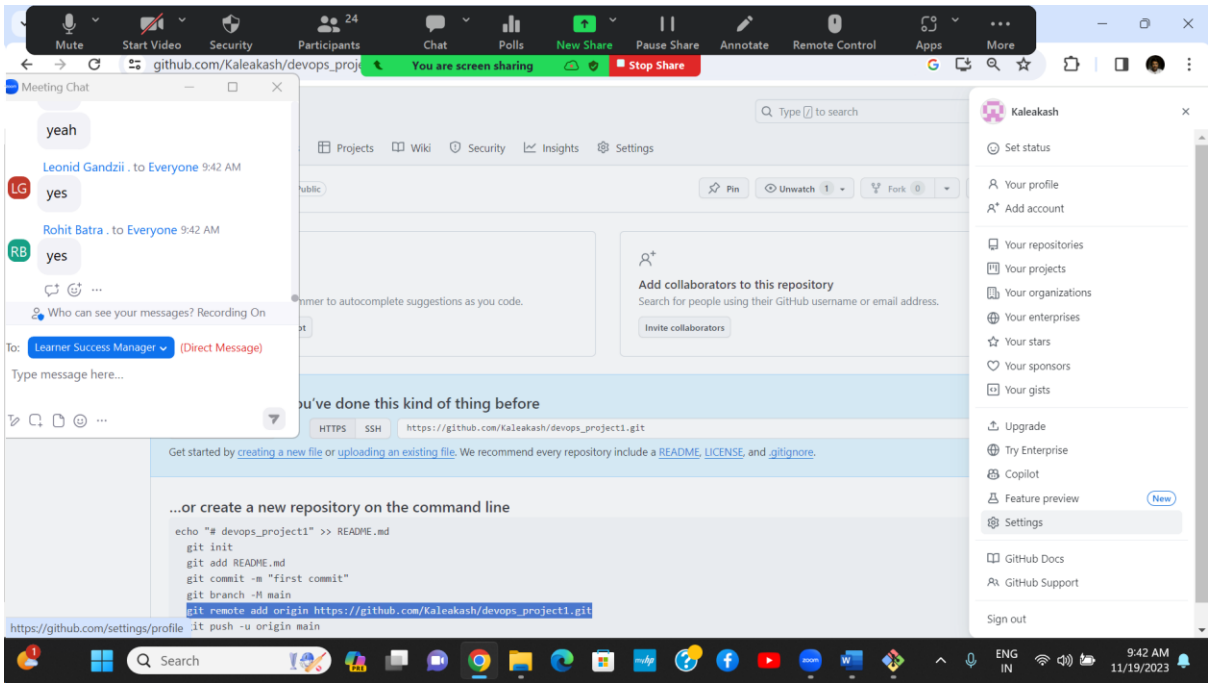
```
git remote add origin https://github.com/Kaleakash/devops_project1.git
```

```
git remote add origin https://token@github.com/Kaleakash/devops_project1.git
```

```
git remote add origin
https://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@github.com/Kaleakash/devops_project1.git
```

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

we will generate token in git hub account.



Moderation

Code, planning, and automation

- Repositories
- Codespaces
- Packages
- Copilot
- Pages
- Saved replies

Security

- Code security and analysis

Integrations

- Applications
- Scheduled reminders

Archives

- Security log
- Sponsorship log

- Developer settings

Don't specify

URL

Social accounts

Link to social profile

Link to social profile

Link to social profile

Link to social profile

Company

You can **@mention** your company's GitHub organization to link it.

Location

Display current local time

Other users will see the time difference from their local time.

All of the fields on this page are optional and can be deleted at any time, and by filling them out, you're giving us consent to share this data wherever your user profile appears. Please see our privacy statement to learn more about how we use this information.

Update profile

ettings / **Developer Settings**

Type / to search

- GitHub Apps
- OAuth Apps
- Personal access tokens
  - Fine-grained tokens Beta
  - Tokens (classic)

GitHub Apps

Ne

Want to build something that integrates with and extends GitHub? Register a new GitHub App to get started c the GitHub API. You can also read more about building GitHub Apps in our developer documentation.

/ Developer Settings

Type / to search

GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens Beta
Tokens (classic)

Personal access tokens (classic)

Generate new token | Revoke all

Tokens you have generated that can be used to access the Gi

Generate new token Beta
Fine-grained, repo-scoped

Generate new token (classic)
For general use

my_token — *admin:org, delete:packages, repo, write:packages*
Expired *on Tue, Nov 14 2023.*

Delete

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

© 2023 GitHub, Inc. Terms Privacy Security Status Docs Contact GitHub Pricing API Training Blog About

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

dev_ops_token

What's this token for?

Expiration *

30 days — The token will expire on Tue, Dec 19 2023

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

☑ **repo**	Full control of private repositories
☑ repo:status	Access commit status
☑ repo_deployment	Access deployment status
☑ public_repo	Access public repositories
☑ repo:invite	Access repository invitations
☑ security_events	Read and write security events
☐ **workflow**	Update GitHub Action workflows
☑ **write:packages**	Upload packages to GitHub Package Registry
☑ read:packages	Download packages from GitHub Package Registry
☑ **delete:packages**	Delete packages from GitHub Package Registry
☑ **admin:org**	Full control of orgs and teams, read and write org project
☑ write:org	Read and write org and team membership, read and write
☑ read:org	Read org and team membership, read org projects
☑ manage_runners:org	Manage org runners and runner groups

Meeting

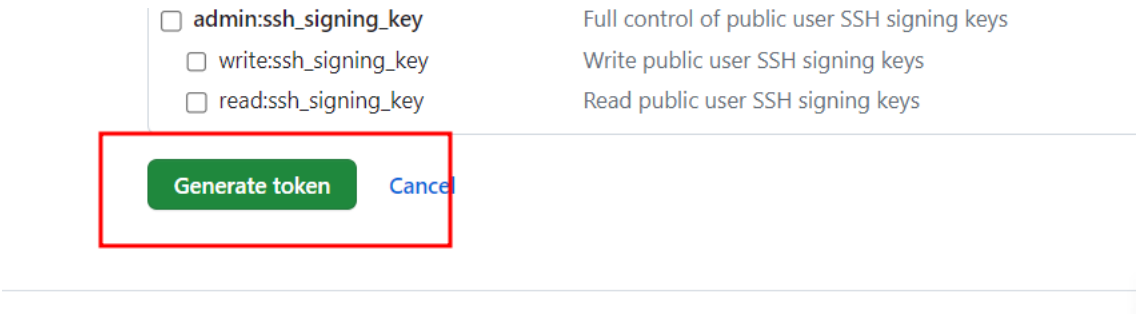
A. ye

BN ye

RB ye

To: Learn

Type mes

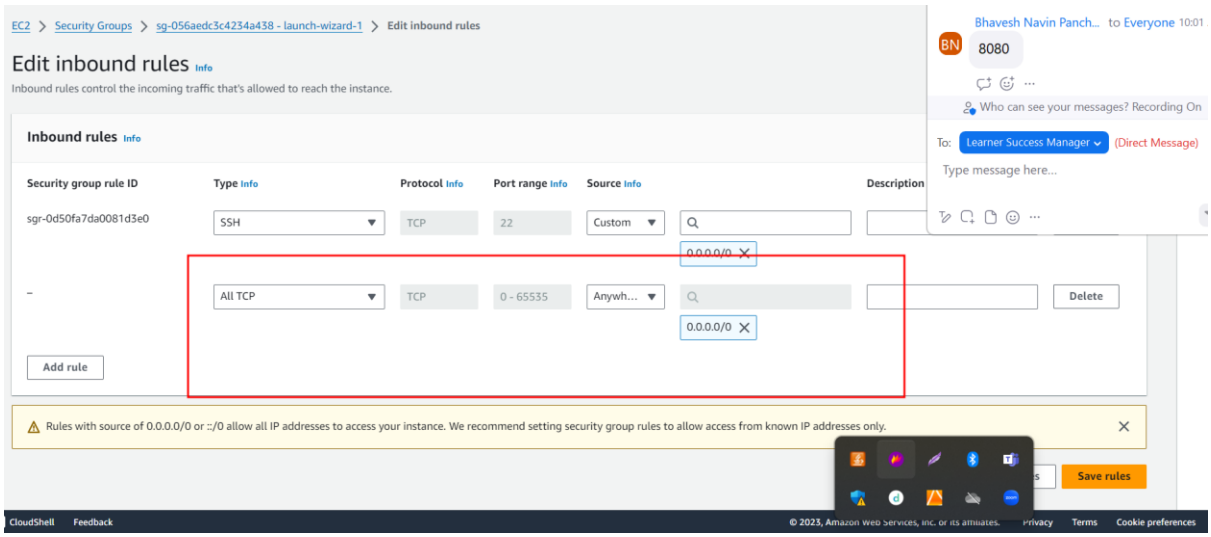



```
git remote add origin https://token@github.com/Kaleakash/devops_project1.git
```

```
git push -u origin HEAD
```

now you need to create EC2 instance

After created EC2 instance now you need to open port number



Connect EC2 instance

Then install required software.

Install the git

sudo yum install git -y

Now you need to clone this project in EC2 instance.

```
aws  Services  Search  [Alt+S]
[ec2-user@ip-172-31-47-227 ~]$ git --version
git version 2.40.1
[ec2-user@ip-172-31-47-227 ~]$ git clone https://github.com/Kaleakash/devops_project1.git
Cloning into 'devops_project1'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 4 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
[ec2-user@ip-172-31-47-227 ~]$ ls
devops_project1
[ec2-user@ip-172-31-47-227 ~]$
```

sudo docker build -t my-web-devops . -f Dockerfile

```
aws  Services  Search  [Alt+S]
[ec2-user@ip-172-31-47-227 devops_project1]$ docker build -t my-web-devops . -f Dockerfile
ERROR: permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%
nnect: permission denied
[ec2-user@ip-172-31-47-227 devops_project1]$ sudo docker build -t my-web-devops . -f Dockerfile
[+] Building 5.7s (7/7) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 144B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/nginx:latest
 => [internal] load build context
 => => transferring context: 177B
 => [1/2] FROM docker.io/library/nginx@sha256:86e53c4c16a6a276b204b0fd3a8143d86547c967dc8258b3d47c3a21bb68d3c6
 => => resolve docker.io/library/nginx@sha256:86e53c4c16a6a276b204b0fd3a8143d86547c967dc8258b3d47c3a21bb68d3c6
 => => sha256:c20060033e06f882b0fbe2db7d974d72e0887a3be5e554efdb0dcf8d53512647 8.15kB / 8.15kB
 => => sha256:578acb154839e9d0034432e8f53756d6f53ba62cf8c7ea5218a2476bf5b58fc9 29.15MB / 29.15MB
 => => sha256:e398db710407fbc310b4bc0b0db1c94161480ac9b44638c6655939f426529780 41.38MB / 41.38MB
 => => sha256:85c41ebe6d660b75d8e2e985314ebce75e602330cd325bc5cfbf9d9723c329a1 627B / 627B
 => => sha256:86e53c4c16a6a276b204b0fd3a8143d86547c967dc8258b3d47c3a21bb68d3c6 1.86kB / 1.86kB
 => => sha256:7170a263b582e6a7b5f650b7f1c146267f694961f837ffefc2505bb9148dd4b0 958B / 958B
 => => sha256:d2e65182b5fd330470eca9b8e23e8a1a0d87cc9b820eb1fb3f034bf8248d37ee 1.78kB / 1.78kB
 => => sha256:8f28d06e2e2ec58753e1acf21d96619aafeab87e63e06fb0590f56091db38014 367B / 367B
 => => sha256:6f837de2f88742f4e8083bff54bd1c64c1df04e6679c343d1a1c9a650078fd48 1.21kB / 1.21kB
 => => sha256:c1dfc7e1671e8340003503af03d067bae6846c12c30cbc1af3e589cb124fd45d 1.40kB / 1.40kB
 => => extracting sha256:578acb154839e9d0034432e8f53756d6f53ba62cf8c7ea5218a2476bf5b58fc9
 => => extracting sha256:e398db710407fbc310b4bc0b0db1c94161480ac9b44638c6655939f426529780
 => => extracting sha256:85c41ebe6d660b75d8e2e985314ebce75e602330cd325bc5cfbf9d9723c329a1
```

After image created successfully please run the image

```
[ec2-user@ip-172-31-47-227 devops_project1]$ ls
Dockerfile  index.html
[ec2-user@ip-172-31-47-227 devops_project1]$ cat Dockerfile
FROM nginx
COPY index.html /usr/share/nginx/html[ec2-user@ip-172-31-47-227 devops_project1]$ sudo docker images
REPOSITORY      TAG       IMAGE ID       CREATED         SIZE
my-web-devops   latest    5b47160b7964   5 minutes ago   187MB
[ec2-user@ip-172-31-47-227 devops_project1]$ sudo docker images
REPOSITORY      TAG       IMAGE ID       CREATED          SIZE
my-web-devops   latest    5b47160b7964   22 minutes ago   187MB
[ec2-user@ip-172-31-47-227 devops_project1]$ sudo docker run -d -p 80:80 my-web-devops
6ac8d43340a8d80511489e434079a9ac4a037e5c978b02e48716d926b98bc3b8
[ec2-user@ip-172-31-47-227 devops_project1]$ sudo docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED         STATUS         PORTS                                   NAMES
6ac8d43340a8   my-web-devops   "/docker-entrypoint.…"   6 seconds ago   Up 4 seconds   0.0.0.0:80->80/tcp, :::80->80/tcp   quirky_ptolemy
[ec2-user@ip-172-31-47-227 devops_project1]$
```

Please check your application running or not

Using EC2 instance public ip address

http://publicIpAddress:80

Now we run docker container using Jenkin

First install java software.

EC2 instance running two service

1. Docker
2. Jenkin

Inside Jenkin Pipeline or jobs we want to run Docker.

We do the changes in local machine in the application ie

Index.html

Then we check status, add, commit and push

In Jenkin you need to create job or pipeline job which pull the project

Whenever we push new changes in remote repository

Jenkin job pull the project from git hub account

And build the docker and run docker container.

Create web page and docker file in local machine.

Then push this code to remove repository using login details ie token base.

Then create EC2 instance

Install required software ie git, docker,java, jenkin refere e2-instance-plugin file.

Then using public ip address open jenkin dashboard

Then create jenkin job or pipeline

Use trigger concept to pull the project whenever we push the code in remote repository

And build the project means create docker image and start new container.

<h2>First Messge</h2>

<h2>Welcome once again </h2>

3 documentation

1st documentation : Git URL which contains your project. And steps to do the task.

2nd documentation : screen shot.

3rd documentation : copy and paste your application code

index.html

Dockerfile