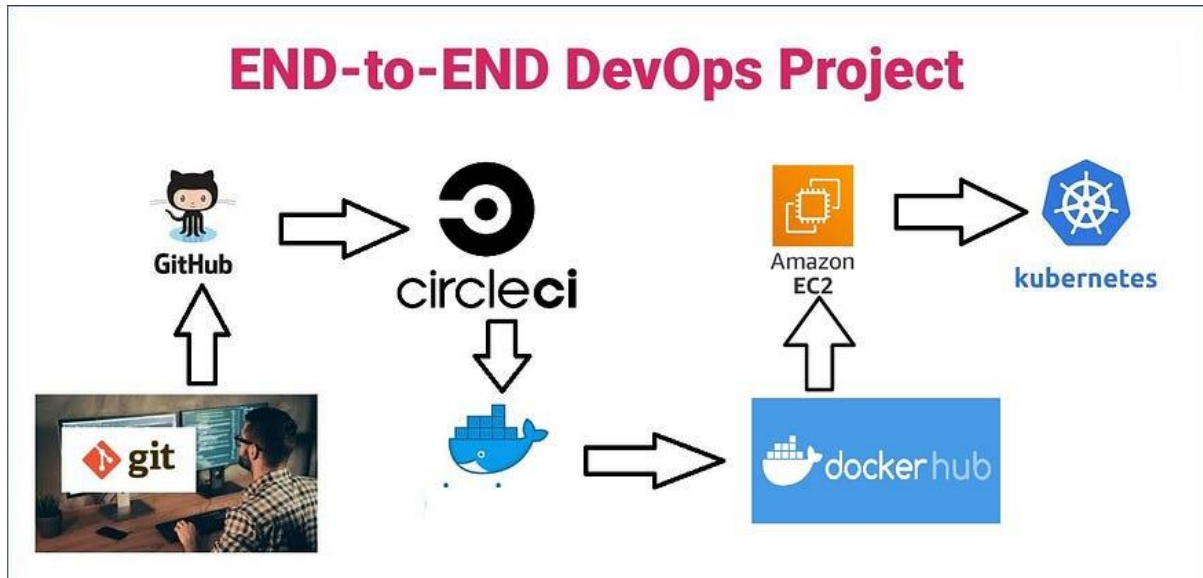


Deployment using Circle Ci and Kubernetes



Completion Steps →

1. upload the source code to GitHub
2. setup circle ci and connect the GitHub repository
3. upload the .circleci/config.yml file to build pipeline
4. This ci -cd pipeline build your docker-image and sent it to DockerHub
5. setup ec2 on AWS and take ssh of it
6. setup minikube and create a sample pod.yml file
7. start your kubernetes cluster

Step 1 → Upload code to GitHub

1. first you need to clone the following repository

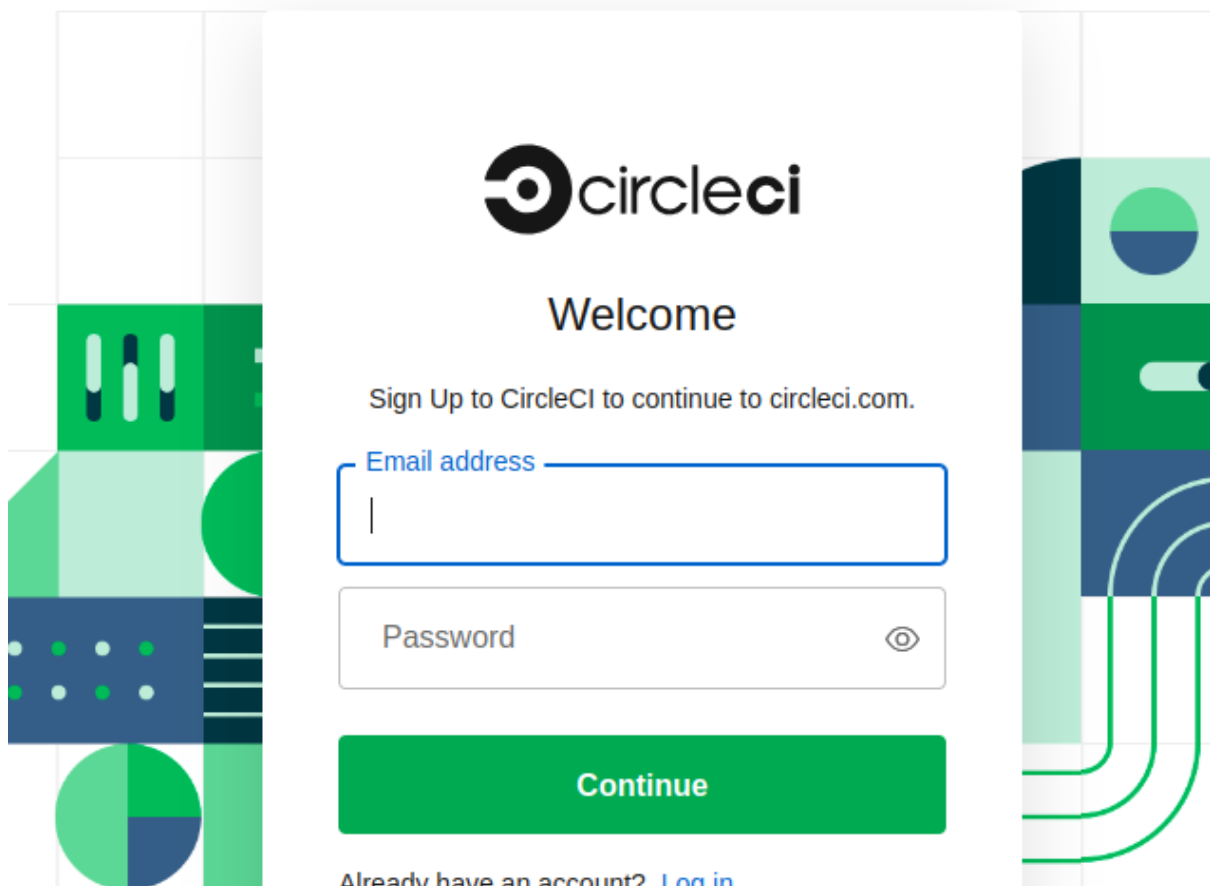
git clone <https://github.com/AdeshNavale98/Devops-end-to-end-deployment-using-circle-ci-and-kubernetes.git>

2. Now push the exact to your GitHub account and move forward to Circle ci step

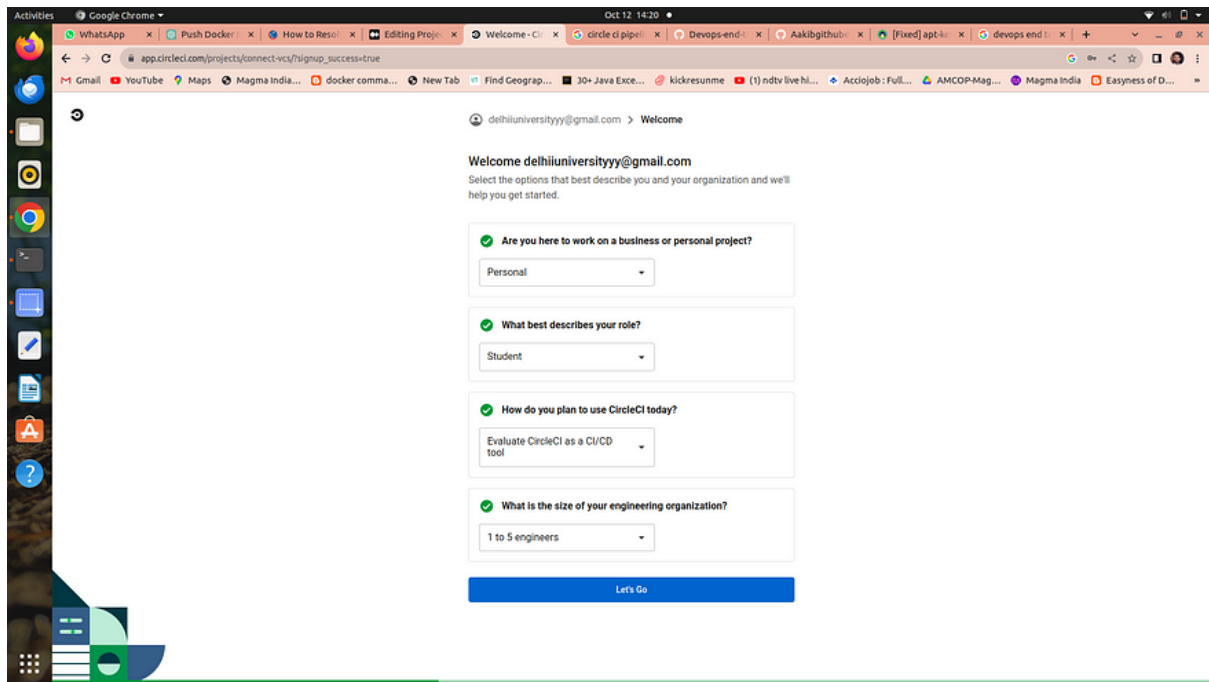
Step 2 → Setup circle ci and connect with GitHub repository

Sign into your circle ci account and if you don't have an account just sign up by the following steps

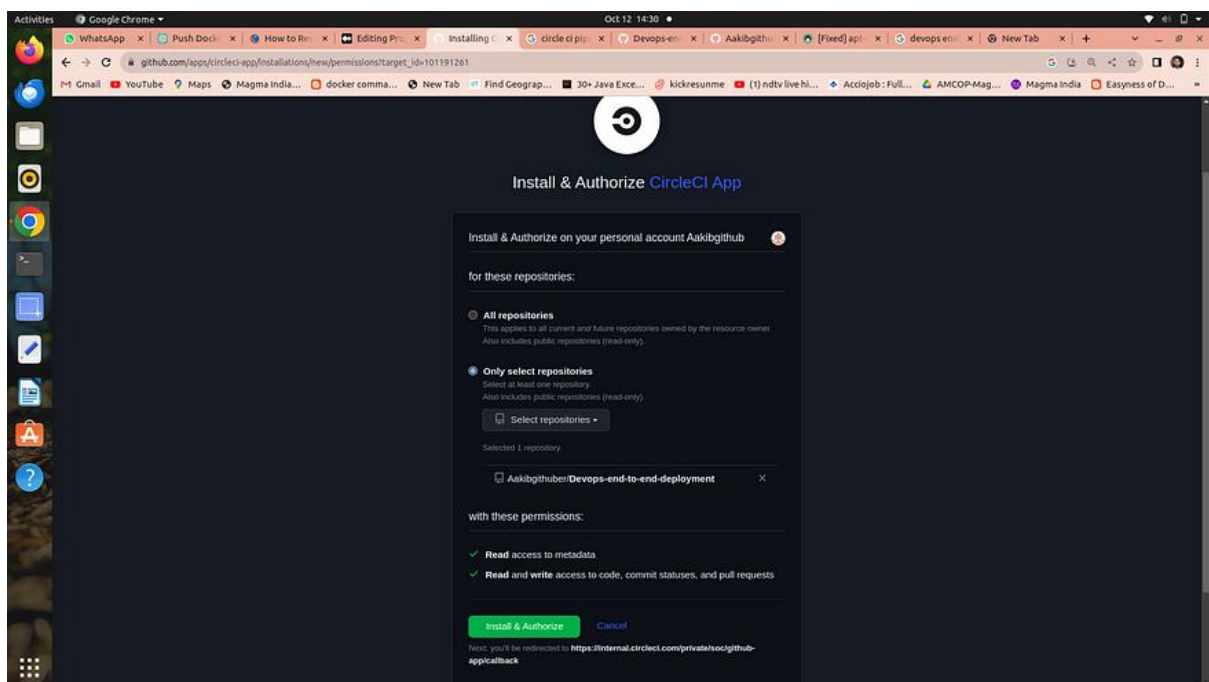
1. go to circle ci sign up page
2. enter your email and create your password

The image shows the CircleCI sign-up page. At the top center is the CircleCI logo, which consists of a black circle with a white dot inside, followed by the text "circleci". Below the logo is the word "Welcome" in a large, black, sans-serif font. Underneath "Welcome" is the text "Sign Up to CircleCI to continue to circleci.com." in a smaller, black, sans-serif font. There are two input fields: the first is labeled "Email address" in blue text and has a blue border; the second is labeled "Password" in gray text and has a gray border. To the right of the password field is a small eye icon. Below the input fields is a large green button with the word "Continue" in white, bold, sans-serif font. At the bottom of the form is the text "Already have an account? [Log in](#)" in a small, black, sans-serif font, where "Log in" is a blue link.

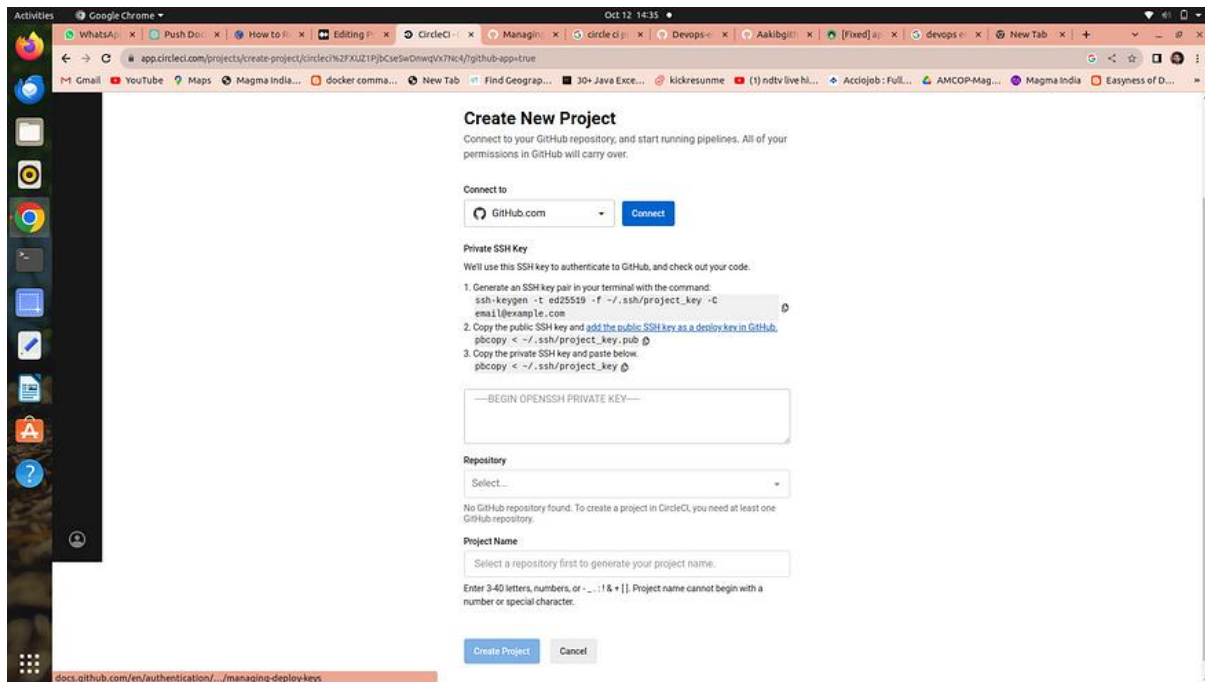
3. Choose the following options



4. Now you have to choose GitHub and you redirect to your GitHub account select the repository where your code present and then click on authorize



5. now you have to create a project for which you have to require a key pairs



1. go to your terminal and generate a key pairs in which public key is paste to deploy key options in GitHub and private key on your create project page
2. run → `ssh-keygen`
3. `cd /root/.ssh`
4. you will find `id_rsa`(private key) and `id_rsa.pub`(public key) just copy paste the on the location mentioned above

Now you are connected to your GitHub repository

Step 3 →upload the .circleci/config.yml file to build pipeline

1. copy the following code and make a folder `.circleci/config.yml` and paste into it and push it to your GitHub repo

version: 2.1

executors:

docker-publisher:

environment:

IMAGE_NAME: adeshnavvale/building-on-ci

docker:

- image: circleci/buildpack-deps:stretch

jobs:

build:

executor: docker-publisher

steps:

- checkout
- setup_remote_docker
- run:
 - name: Build Docker image
 - command: |
docker build -t \$IMAGE_NAME:latest .
- run:
 - name: Archive Docker image
 - command: docker save -o image.tar \$IMAGE_NAME
- persist_to_workspace:
 - root: .
 - paths:
 - ./image.tar

publish-latest:

executor: docker-publisher

steps:

- attach_workspace:
 - at: /tmp/workspace
- setup_remote_docker
- run:
 - name: Load archived Docker image
 - command: docker load -i /tmp/workspace/image.tar
- run:
 - name: Publish Docker Image to Docker Hub
 - command: |
echo "\$DOCKERHUB_PASSWORD" | docker login -u
"\$DOCKERHUB_USERNAME" --password-stdin
docker push \$IMAGE_NAME:latest

workflows:

version: 2

build-master:

jobs:

- build:
 - filters:
 - branches:
 - only: master
- publish-latest:
 - requires:
 - build

filters:

branches:

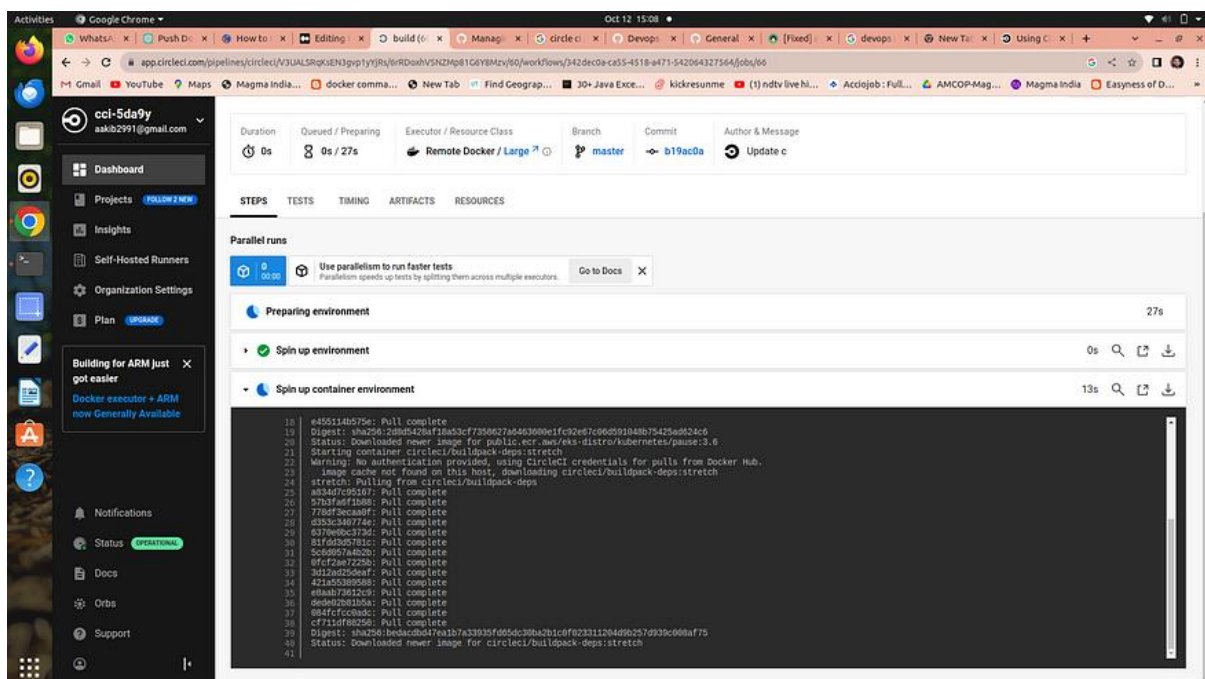
only: master

2. Add the environment variable by click on project setting option you will find Environment variable option

\$DOCKERHUB_USERNAME → your dockerhub username

\$DOCKERHUB_PASSWORD → your dockerhub password

now it will automatically starts building your docker image and push it to your dockerhub account



Step 4 → Setup EC2 on AWS and take ssh of it

1. go to your AWS account and launch ubuntu machine and take ssh it into your terminal

commands to run on your terminal

a. `ssh -i <your key pair name> ec2-user@<public ip>`

b. `apt update`

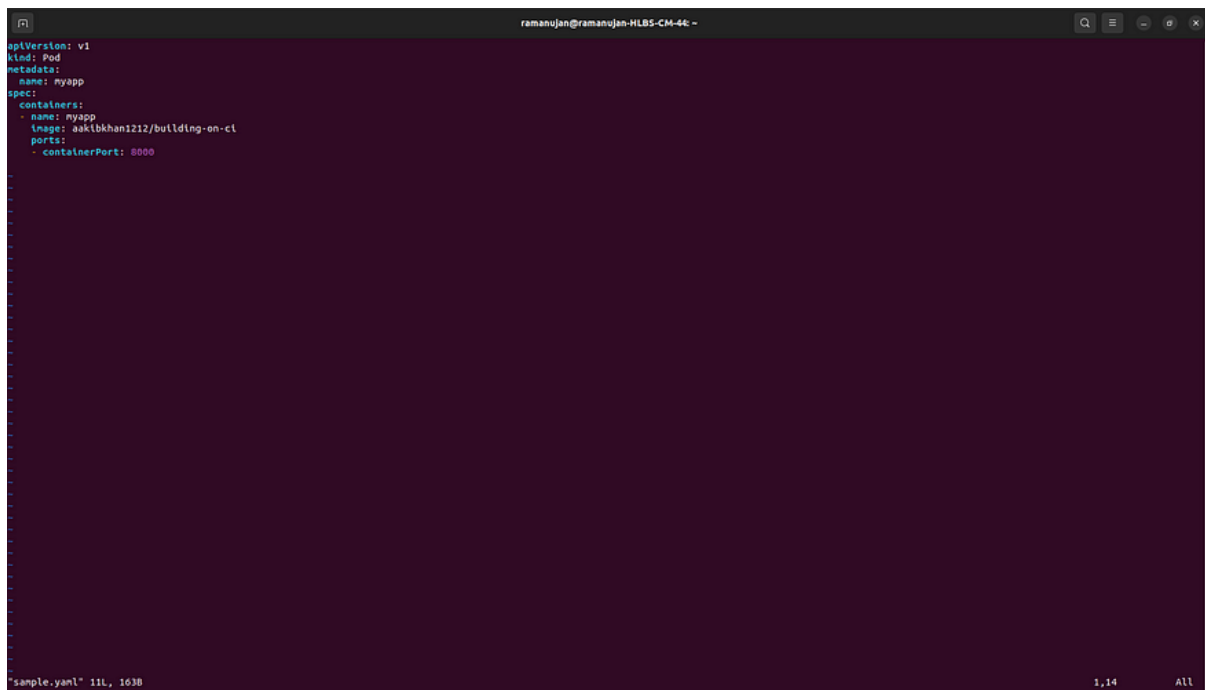
Step 5 → Setup Minikube for kubernetes

command to run on your terminal

1. curl -
LO <https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64>
2. sudo install minikube-linux-amd64 /usr/local/bin/minikube
3. minikube start — driver docker
4. kubectl get po -A

Now we have to write the sample.yaml file for pod deployment →

Below is the example file just replace the image name that is present in dockerhub repo



```
apiVersion: v1
kind: Pod
metadata:
  name: myapp
spec:
  containers:
    - name: myapp
      image: aakibkhan1212/building-on-cl
      ports:
        - containerPort: 8000
```

The screenshot shows a terminal window with a dark background. The terminal displays the content of a file named 'sample.yaml'. The file defines a Pod with a single container named 'myapp' using the image 'aakibkhan1212/building-on-cl' and exposing port 8000. The terminal window title is 'ramanujan@ramanujan-HL85-CM-46 -'. At the bottom of the terminal, it shows 'sample.yaml' 11L, 163B and a status bar with '1,14 All'.

once you created a file then you have to run a command to create a pod from file

1. kubectl apply -f pod sample.yaml
2. kubectl get po -o wide

you will see your pod is created and running having a private ip which you could use in checking

1. minikube ssh
2. **curl -L http://<your pod ip>:<your port no.>**

you will see your application is running inside a minikube cluster change now you are free to make changes in yaml file such as creating replicas etc.