Description of the system

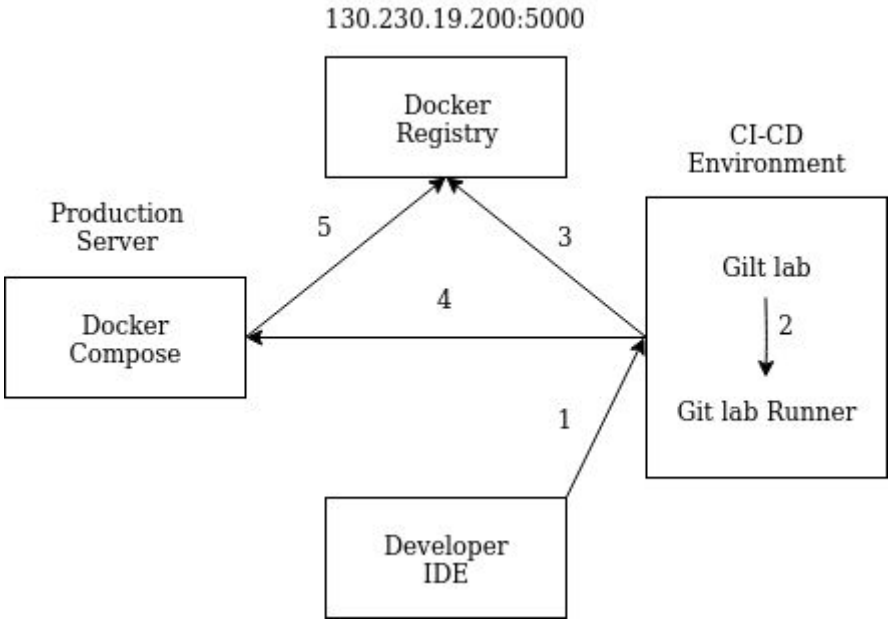
This system consist of 4 components which automate the CI/CD pipeline from code commit to deploy the feature of a service. In our scenario all 4 components are on same VM having IP address 130.230.19.200 which is accessible in University premises.

- **Developer IDE**
  We just emulate developer IDE in same VM by doing any change in code using Linux editor like nano or vi.

- **Gitlab Server**
  This components consists of further two parts.
  First is Gitlab server and second is gitlab-runner both run in docker containers. Both are ubuntu based images. We formatted docker-compose file (To run multi-containers app) with all its images, network settings, mapped volume and exposed ports.
  We registered gitlab-runner as shell. So we have to install docker and python in the gitlab-runner.

- **Docker Registry**
  When image of our service is build it is pushed to docker registry on 130.230.19.200:5000 to use it on remote machine, in our case out host VM. There is no authentication on docker registry. Anyone can pull or push the image in University premises knowing image URL/imagename:tag

- **Deployment Server**
  This is the server where our build services1 “130.230.19.200:8001” and service2 “130.230.19.200:8002” run to server.
  To enable authentication without password, We generated a private/public key pair on gitlab-runner. Copy public key on deployments server in “/home/tanveer/.ssh/authorized_keys” and set permission as 400. We will use private key in .gitlab-ci.yml file to access deployment server.



Instructions for examiner to test the system

Examiner can clone the repository using credentials because it is private repository, made changes and push the code. Pipeline will run and deploy updated services.

Credetials:

Username: engr.tanveerkn@gmail.com
Password: pakistan

System Flow

Developer will add required code to add feature in its code, add changed files, commit code and push on the gitlab repository. It will ask for authentication. After authentication a pipeline will be trigger.

Git Steps

$git add –all
$git commit -m “Any change message”
$git push

All pipeline steps are mentioned in .gitlab-ci.yml file.

Pipeline Steps

Step 1

Gitlab server will start gitlab-runner. Updated code will be cloned in gitlab runner. Images of both services will be build using respective Dockerfile.

Step 2

It will be tested either docker registry is working or not.

Step 3

Create images will be tagged and pushed to docker registry.

Step 4

Gitlab-runner will access deployment server using private key and down the docker-compose file if it is running and then start the docker-compose file as below. Go into the respective directory and run.
$docker-compose up -d
Both services with new images having latest change will be started in containers.

- **Example:**

  We applied check status for docker registry in test stage but it is not working properly right now.

Main learnings and worst difficulties

1. We installed gitlab server on port 80 which mapped with 8000 port of host machine. Everything was fine but when gitlab-runner try to clone the code, it use URL
   http://130.230.19.200/tanveer/docker-project.git instead of following
   http://130.230.19.200:8000/tanveer/docker-project.git.

   So we have to mapped host port 80 with container port 80.
2. It was difficult to access deployment server from gitlab-runner container even we were using right private key. Finally we added a parameter (StrictHostKeyChecking=no)which fix the issue.

Amount effort (hours) used

64 hours

Application and its Features:

The programing language we used is python. We used Flask, a micro web framework written in Python. The application consists of the following features:

1. HTTP info about service1 and service2. Just clik on the links once u execute the python code.
2. POST /fibo : The fibonacci output is accessible through the postman by entering the fibonacci number at the end of URL such as http://127.0.0.1:8001/fib/9 or you can enter it at the body in JSON format such as {“number”,”9”} and it will return the output.
3. POST /stop
4. GET /run-log: It will give you the time information.
5. POST /shutdown