Software Production Engineering (CS816) (1)

Major Project
Buttercrust

Under the guidance of Professor B. Thangaraju and Vaibhav Tandon

Group Members

- Harshit Agrawal (MT2021051)
- Samrat Seth (MT2021113)

1. Abstract

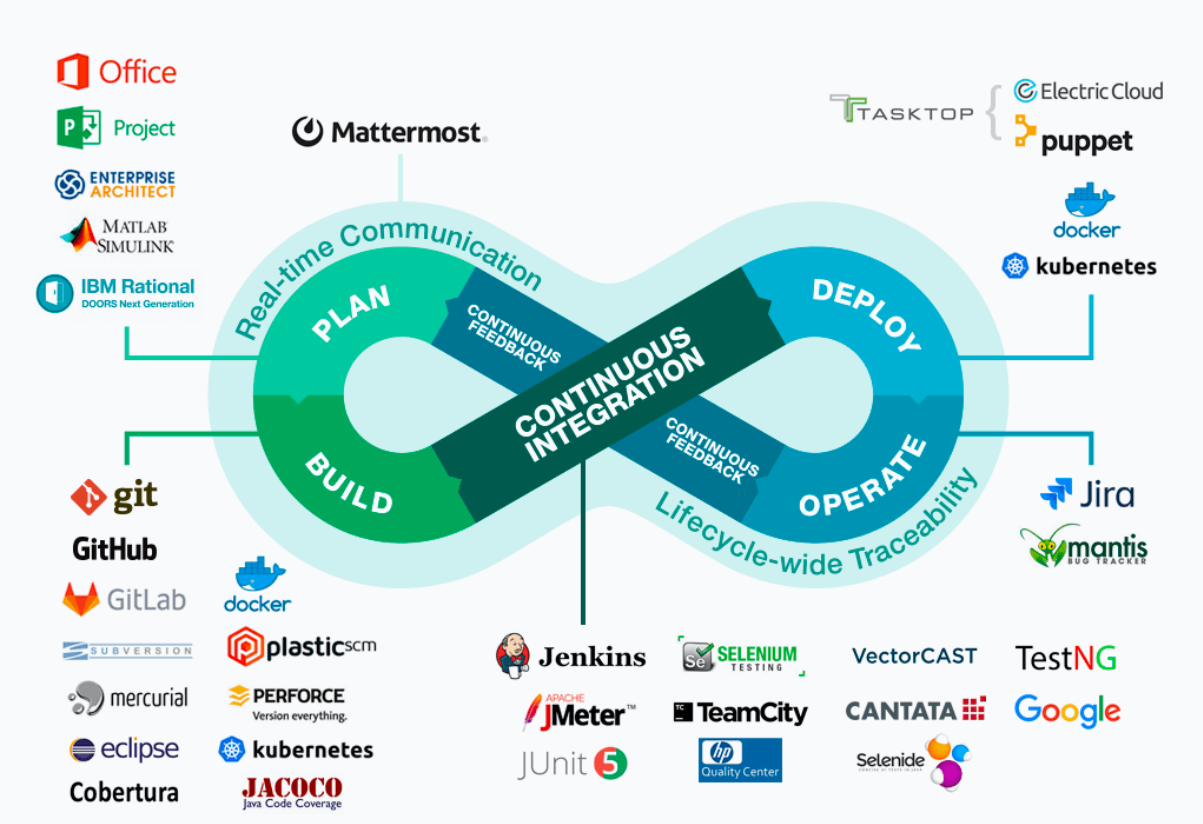
The online Pizza ordering system is a web-based application that enables customers to order their pizzas online for home delivery. Each country has its own kind of dishes to offer. But if we pick a food item that is loved by all the people on this planet, then pizza will be a clear winner in it. The whole world is in love with pizzas. The billions of dollars earned by different pizzerias across the globe just prove this. The love of pizzas has enabled the rise of large pizza companies like Pizza Hut, Domino's, Papa John's, and much more.

As the internet users are increasing exponentially, these companies have introduced an Online Pizza ordering system for taking orders from customers. This system not only improves the customer experience but also eases the workload on the staff of pizzerias.

This is a Full Stack (MERN) Pizza Delivery Application developed using React for Front End, Redux-Thunk for Asynchronous operations, Node JS for Runtime environment, Express JS for Backend Routing, and Mongo DB for Database.

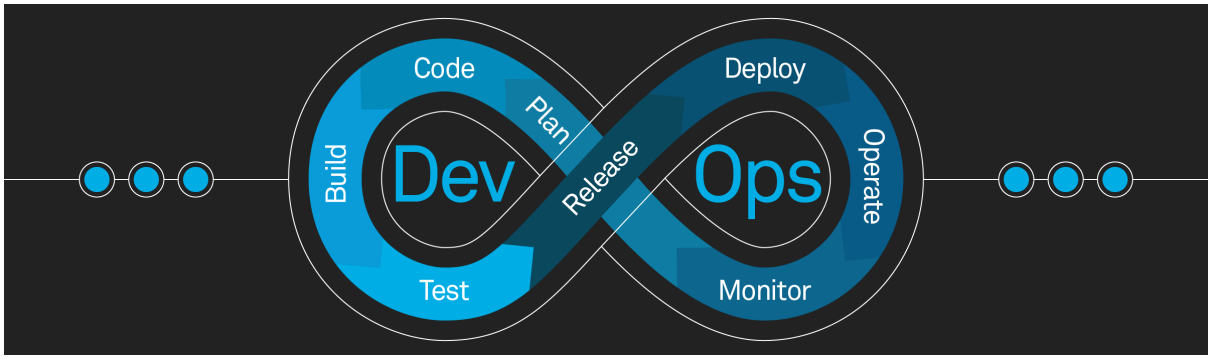
2. What is DevOps?

1. DevOps is the practice of operations and development engineers participating together in
   the entire service lifecycle, from design through the development process to production
   support.
2. DevOps is also characterized by operations staff making use of many of the same
   techniques/tools as developers for their systems work.
3. DevOps is the combination of cultural philosophies, practices, and tools that increases an
   organization's ability to deliver applications and services at high velocity: evolving and
   improving products at a faster pace than organizations using traditional software development and infrastructure management processes. This speed enables organizations
   to better serve their customers and compete more effectively in the market.



3. Why DevOps?

We plan to build this project in a growing way. The design has ideas and services that work
independently. Given the complexity of the project, it is impossible for any of us to create and
test the entire code manually every time we make a small change. And since the three of us
work from different locations, the automatic pipeline will not only make our job easier, and
make it more efficient. The amount of communication that must take place between us will
decrease. DevOps helps us focus on key aspects of the project, improving efficiency, stability, and security. There is a small range of manual errors as well. And since we plan to build this
product at some point, continuous delivery makes it easier. Also, monitoring allows us to better
understand usage and help us improve the application.



4. System Configuration

4.1 Operation System

- Ubuntu 20.04.4 LTS (Focal Fossa)

4.2 CPU and RAM

- Ryzen 9 CPU and 16 GB Ram

4.3 Frameworks

- React JS
- Node JS

4.4 Database

- MongoDB

4.5 Building Tools

- npm (npm is a package manager for the JavaScript programming language)

4.6 AWS EC2 Instance

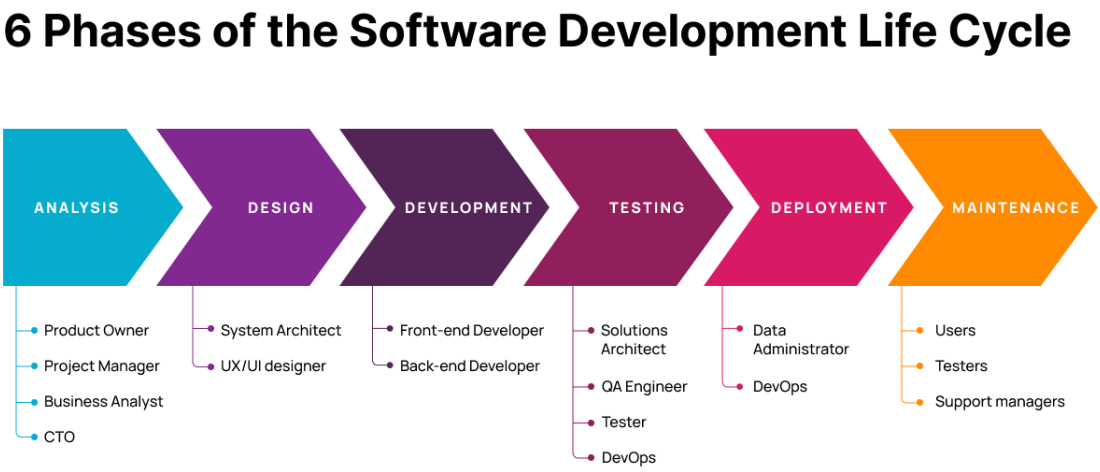
- Type - T2.medium
- OS - Ubuntu 20.04
- Ram - 2 GiB
- Storage - 15 GiB

4.6 DevOps Tools

- GitHub: Version control system

- Jenkins: CI/CD pipeline
- Ansible: Configuration management and infrastructure as code
- ELK: Monitoring
- Docker: Deployment/Containerization

5. Software Development Life Cycle (SDLC)



5.1. Source Code Management (SCM): (Link to the repository: https://github.com/sethsamrat/Buttercrust-App)

- SCMs are used to give versions/revisions to the program. Each type is given a timestamp and
  includes the person responsible for the change. Even different versions can be compared and
  integrated with other types. That is why SCM is also called Version Control, Revision Control or
  Source Control.
- In order to achieve SCM, we need to create a GitHub repository on github.com by specifying
  the name and description of the project. This creates an empty repository on GitHub.

- We can also add a readme file in the repository that contains some information about the
  project. After creating an empty repository on GitHub we need to clone an empty project to the local system. This would create a directory in the name of the project in which we can add the files of our project. In this directory, add all the files of the project.
- Now add these files to the staging area.

  ```
  $ git add *
  ```

  Then commit those changes so that the files would get added to the local repo.

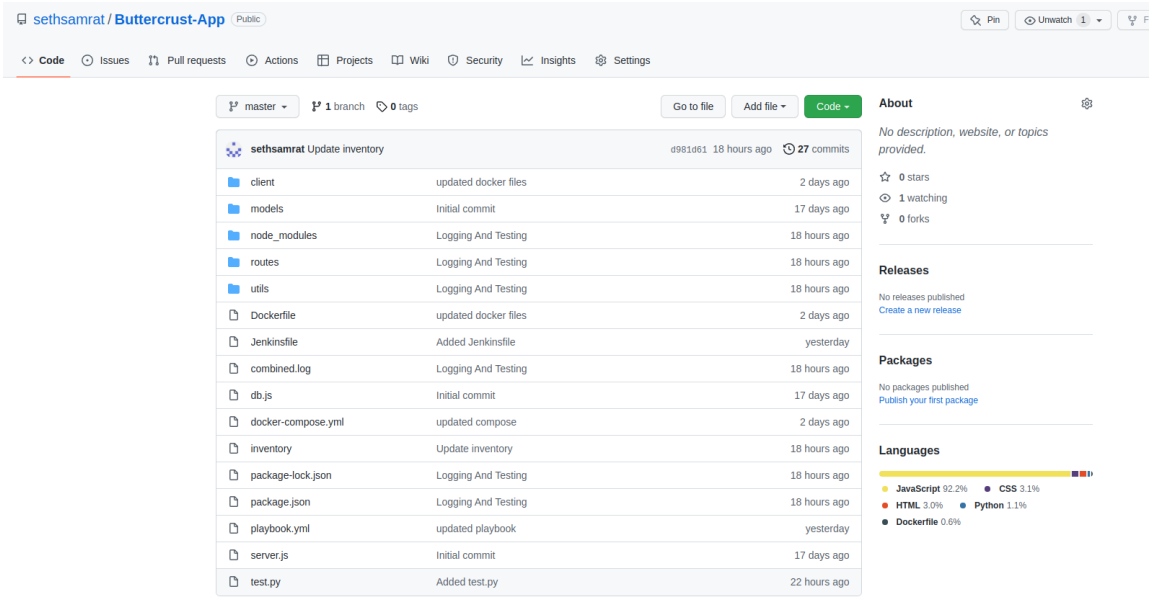
  ```
  $ git commit -m “message”
  ```

  Now in order to add these files to the GitHub repo, we need to push the files to the repo.

  ```
  $ git push origin master
  ```

  We have now successfully added our project to the GitHub which enables the other users to use
  the same project and made required modifications to the project by git pull.
  After pushing the files to GitHub the GitHub would look as below.

  Repository



5.2 Build

5.3 Testing

5.4 Docker

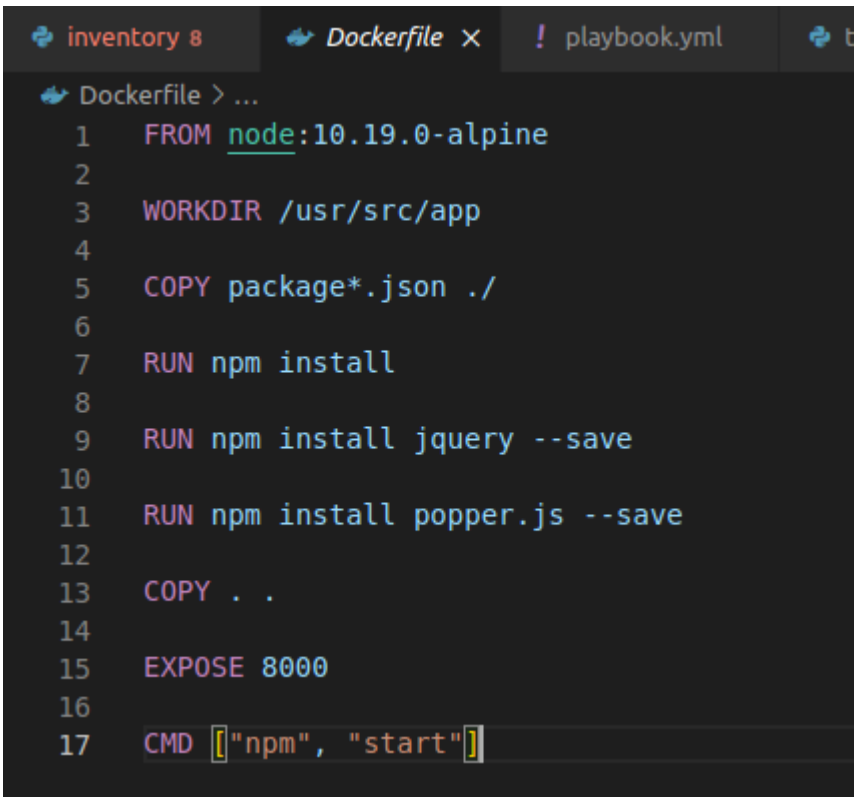
- Docker enables developers to easily pack, ship, and run any application as a lightweight, portable, self-sufficient container, which can run virtually anywhere. As Bottomley told me, containers give you instant application portability.
- Containers do this by enabling developers to isolate code into a single container. This makes it
  easier to modify and update the program. It also lends itself, as Docker points out, for enterprises to break up big development projects among multiple smaller, Agile teams using Jenkins, an open-source CI/CD program, to automate the delivery of new software in containers.
- **Dockerfile for Client Image**

inventory 8 | Dockerfile | playbook.yml

client > Dockerfile > ...

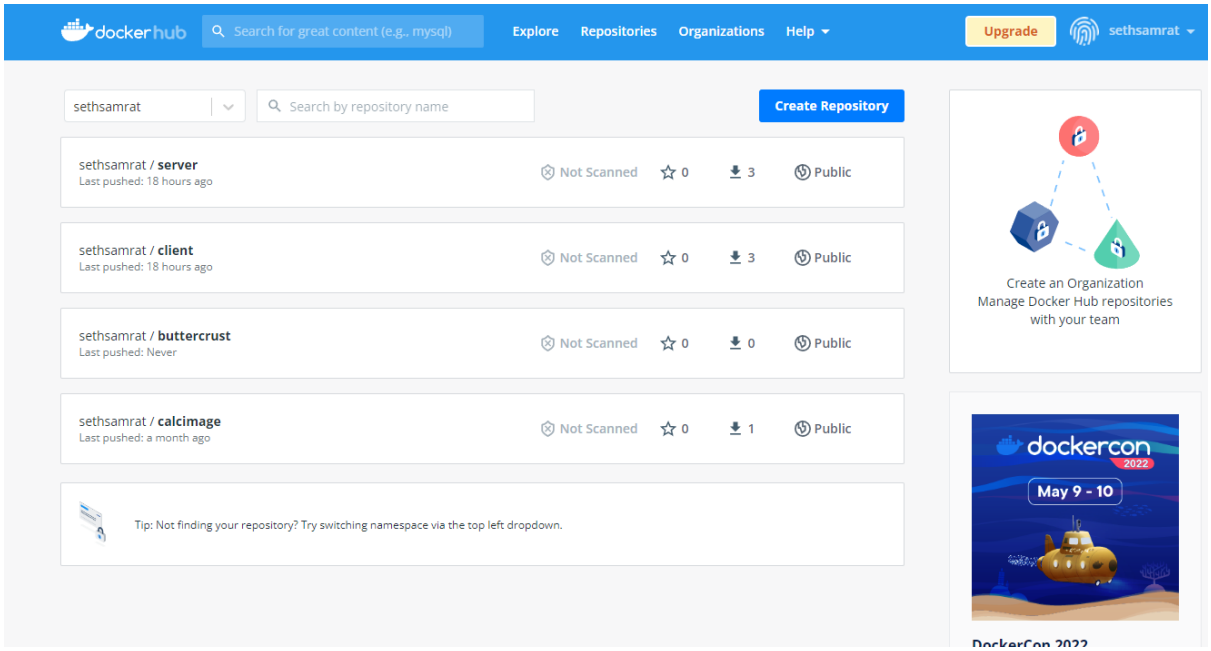
```
1   FROM node:10.19.0-alpine
2
3   WORKDIR /usr/src/app
4
5   COPY package*.json ./
6
7   RUN npm install --verbose
8
9   RUN npm install jquery --save
10
11  RUN npm install popper.js --save
12
13  COPY . .
14
15  EXPOSE 3000
16
17  CMD ["npm", "start"]
```

- **Dockerfile for Server Image**

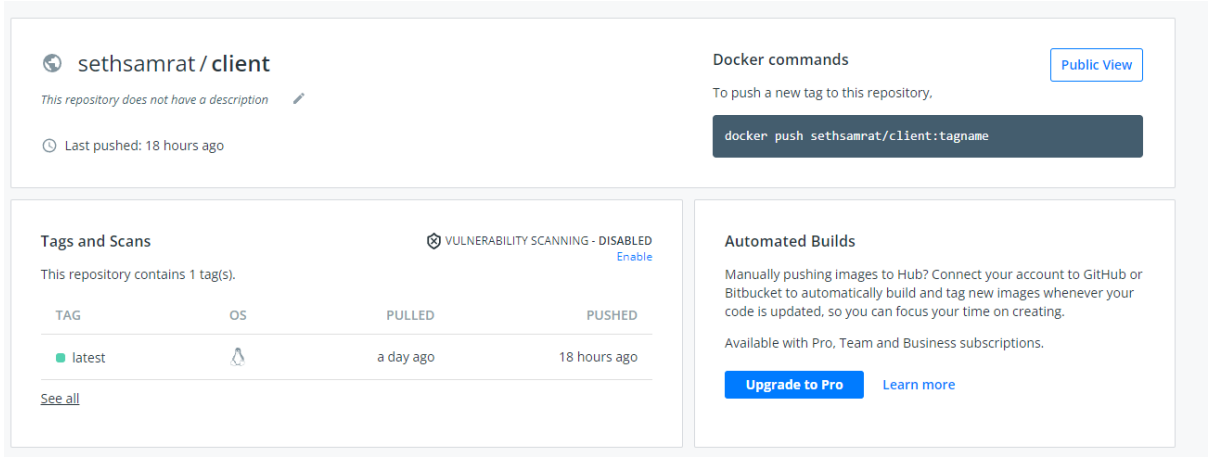


- **Docker-Compose**

- **Repositories in DockerHub**



- **Client Repository**



- **Server Repository**