



Software Production Engineering (CS816)

Major Project Buttercrust

Under the guidance of Professor B. Thangaraju and Vaibhav Tandon

Group Members

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

Major Project

Buttercrust

Group Members

1. Abstract

2. What is DevOps?

3. Why DevOps?

4. System Configuration

4.1 Operation System

4.2 CPU and RAM

4.3 Frameworks

4.4 Database

4.5 Building Tools

4.6 AWS EC2 Instance

4.6 DevOps Tools

5. Software Development Life Cycle (SDLC)

5.1. Source Code Management (SCM):

(Link to the repository: <https://github.com/sethsamrat/Buttercrust-App>)

5.3 Testing

5.4 Docker

5.5 Deployment using Ansible (Deployed on AWS)

5.6 Continuous Integration using Jenkins

5.7 Amazon Web Services

5.8 Logs and Monitoring

1. Abstract

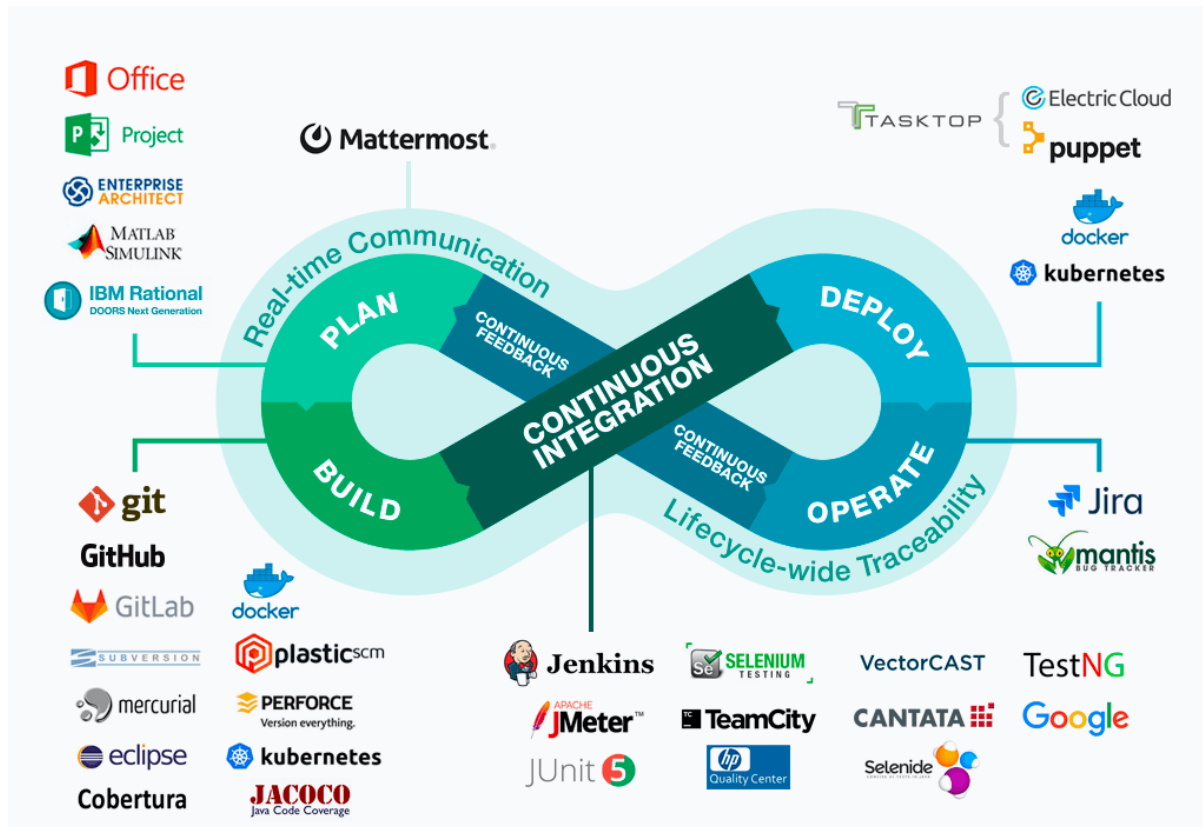
The online Pizza ordering system is a web-based application that enables customers to order their pizzas online for home delivery. Each country has its own kind of dishes to offer. But if we pick a food item that is loved by all the people on this planet, then pizza will be a clear winner in it. The 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.

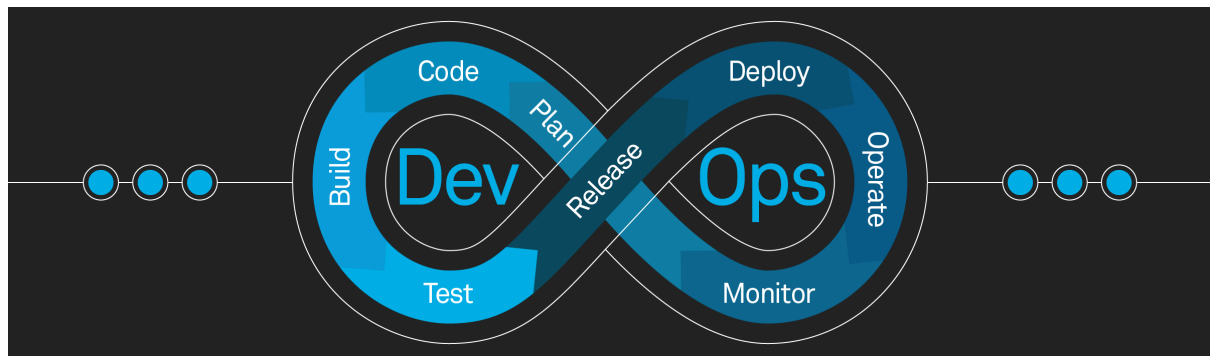
2. What is DevOps?

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

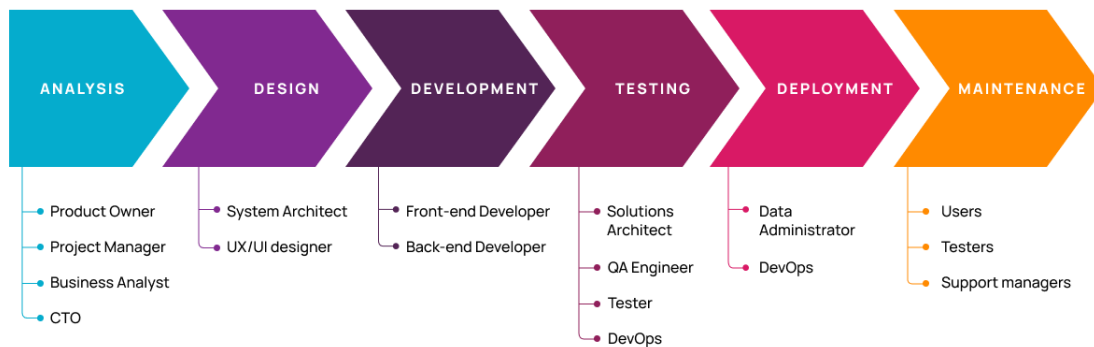
4.6 DevOps Tools

- 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)

6 Phases of the Software Development Life Cycle



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

The screenshot shows a GitHub repository for 'sethsamrat / Buttercrust-App'. The main content is a commit history table. The right sidebar contains sections for 'About', 'Releases', 'Packages', and 'Languages'.

File	Commit Message	Time Ago
client	updated docker files	2 days ago
models	Initial commit	17 days ago
node_modules	Logging And Testing	18 hours ago
routes	Logging And Testing	18 hours ago
utils	Logging And Testing	18 hours ago
Dockerfile	updated docker files	2 days ago
Jenkinsfile	Added Jenkinsfile	yesterday
combined.log	Logging And Testing	18 hours ago
db.js	Initial commit	17 days ago
docker-compose.yml	updated compose	2 days ago
inventory	Update inventory	18 hours ago
package-lock.json	Logging And Testing	18 hours ago
package.json	Logging And Testing	18 hours ago
playbook.yml	updated playbook	yesterday
server.js	Initial commit	17 days ago
test.py	Added test.py	22 hours ago

About
No description, website, or topics provided.
0 stars
1 watching
0 forks

Releases
No releases published
[Create a new release](#)

Packages
No packages published
[Publish your first package](#)

Languages

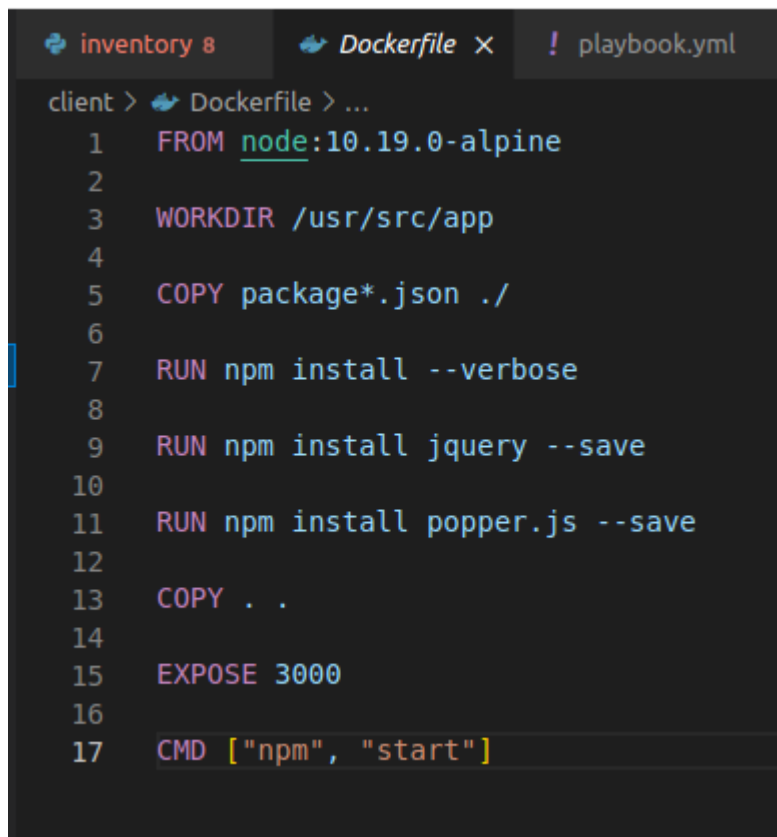
Language	Percentage
JavaScript	92.2%
CSS	3.1%
HTML	3.0%
Python	1.1%
Dockerfile	0.6%

5.3 Testing

/

5.4 Docker

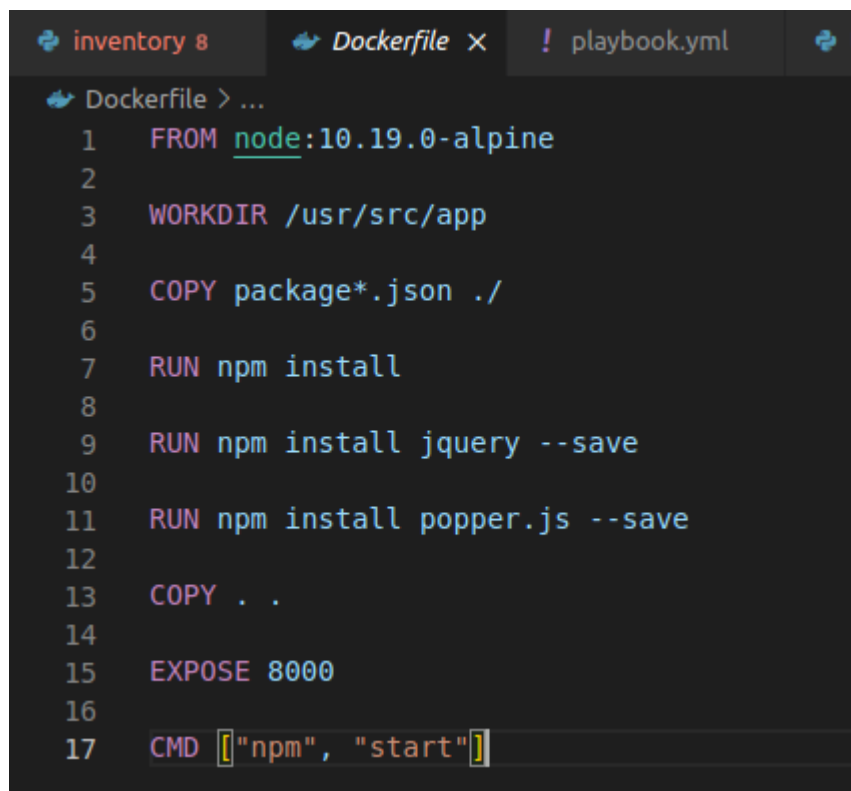
- Docker enables developers to easily pack, ship, and run any application as a lightweight, portable, self-sufficient container, which can run virtually anywhere. As Bottomley told me, containers give you instant application portability.
- Containers do this by enabling developers to isolate code into 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**



The screenshot shows a code editor with three tabs: 'inventory 8', 'Dockerfile x', and 'playbook.yml'. The 'Dockerfile' tab is active, displaying a Dockerfile with 17 lines of code. The code starts with 'FROM node:10.19.0-alpine', sets 'WORKDIR /usr/src/app', copies 'package*.json' to the current directory, runs 'npm install --verbose', 'npm install jquery --save', and 'npm install popper.js --save'. It then copies the current directory to the container, exposes port 3000, and sets the command to 'npm start'.

```
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"]
```

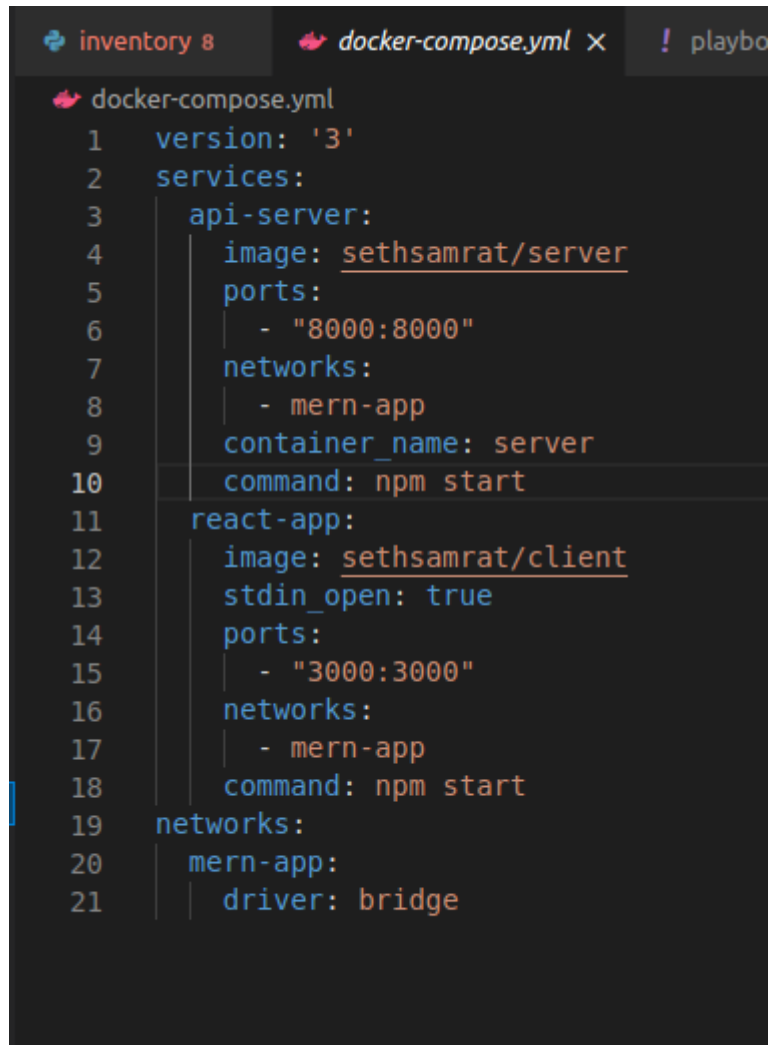
- Dockerfile for Server Image



The screenshot shows a code editor with four tabs: 'inventory 8', 'Dockerfile x', 'playbook.yml', and 't'. The 'Dockerfile' tab is active, displaying a Dockerfile with 17 lines of code. The code starts with 'FROM node:10.19.0-alpine', sets 'WORKDIR /usr/src/app', copies 'package*.json' to the current directory, runs 'npm install', 'npm install jquery --save', and 'npm install popper.js --save'. It then copies the current directory to the container, exposes port 8000, and sets the command to 'npm start'.

```
Dockerfile > ...
1 FROM node:10.19.0-alpine
2
3 WORKDIR /usr/src/app
4
5 COPY package*.json ./
6
7 RUN npm install
8
9 RUN npm install jquery --save
10
11 RUN npm install popper.js --save
12
13 COPY . .
14
15 EXPOSE 8000
16
17 CMD ["npm", "start"]
```


- Docker-Compose



```
1  version: '3'
2  services:
3    api-server:
4      image: sethsamrat/server
5      ports:
6        - "8000:8000"
7      networks:
8        - mern-app
9      container_name: server
10     command: npm start
11   react-app:
12     image: sethsamrat/client
13     stdin_open: true
14     ports:
15       - "3000:3000"
16     networks:
17       - mern-app
18     command: npm start
19   networks:
20     mern-app:
21       driver: bridge
```

- Repositories in DockerHub

[Explore](#)
[Repositories](#)
[Organizations](#)
[Help](#)
[Upgrade](#)

[Create Repository](#)

sethsamrat / server

Not Scanned
0
3
Public

sethsamrat / client

Not Scanned
0
3
Public

sethsamrat / buttermcrust

Not Scanned
0
0
Public

sethsamrat / calcimage

Not Scanned
0
1
Public

Tip: Not finding your repository? Try switching namespace via the top left dropdown.

Create an Organization
Manage Docker Hub repositories
with your team

May 9 - 10

DockerCon 2022

• Client Repository

sethsamrat / client

This repository does not have a description

Last pushed: 18 hours ago

Docker commands

To push a new tag to this repository,

```
docker push sethsamrat/client:tagname
```

Tags and Scans

VULNERABILITY SCANNING - DISABLED
Enable

This repository contains 1 tag(s).

TAG	OS	PULLED	PUSHED
latest		a day ago	18 hours ago

[See all](#)

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions.

[Upgrade to Pro](#)
[Learn more](#)

• Server Repository

sethsamrat / server

This repository does not have a description

Last pushed: 18 hours ago

Docker commands

To push a new tag to this repository,

```
docker push sethsamrat/server:tagname
```

Tags and Scans

VULNERABILITY SCANNING - DISABLED
Enable

This repository contains 1 tag(s).

TAG	OS	PULLED	PUSHED
latest		---	18 hours ago

[See all](#)

Automated Builds

Manually pushing images to Hub? Connect your account to GitHub or Bitbucket to automatically build and tag new images whenever your code is updated, so you can focus your time on creating.

Available with Pro, Team and Business subscriptions.

[Upgrade to Pro](#)
[Learn more](#)

5.5 Deployment using Ansible (Deployed on AWS)

- Ansible is a radically simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs
- Designed for multi-tier deployments since day one, Ansible models your IT infrastructure by describing how all of your systems interrelate, rather than just managing one system at a time.
- It uses no agents and no additional custom security infrastructure, so it's easy to deploy - and most importantly, it uses a very simple language (YAML, in the form of Ansible Playbooks) that allow you to describe your automation jobs in a way that approaches plain English.
- Ansible works by connecting to your nodes and pushing out small programs, called "Ansible modules" to them. These programs are written to be resource models of the desired state of the system. Ansible then executes these modules (over SSH by default) and removes them when finished. Your library of modules can reside on any machine, and there are no servers, daemons, or databases required
- Typically you'll work with your favorite terminal program, a text editor, and probably a version control system to keep track of changes to your content.
- Passwords are supported, but SSH keys with ssh-agent are one of the best ways to use Ansible. Though if you want to use Kerberos, that's good too.
- **playbook.yml**

```
! playbook.yml
1 ---
2 - name: Deploying the application
3   hosts: all
4   become: true
5   tasks:
6     - name: Install required system packages
7       apt:
8         pkg:
9           - apt-transport-https
10            - ca-certificates
11            - curl
12            - software-properties-common
13            - python3-pip
14            - virtualenv
15            - python3-setuptools
16         state: latest
17         update_cache: true
18     - name: Add Docker GPG apt Key
19       apt_key:
20         url: https://download.docker.com/linux/ubuntu/gpg
21         state: present
22     - name: Add Docker Repository
23       apt_repository:
24         repo: deb https://download.docker.com/linux/ubuntu focal stable
25         state: present
26     - name: Update apt and install docker-ce
27       apt:
28         name: docker-ce
29         state: latest
30         update_cache: true
31     - name: Install Docker Module for Python
32       pip:
33         name: docker
34     - name: Install docker-compose
35       remote_user: ubuntu
36       get_url:
37         url: https://github.com/docker/compose/releases/download/1.25.1-rc1/docker-compose-Linux-x86_64
38         dest: /usr/local/bin/docker-compose
39         mode: 'u+x,g+x'
```

```
- name: Starting the docker service
  service:
    name: docker
    state: started

- name: Copying the docker compose file
  copy:
    src: ./docker-compose.yml
    dest: ./

- name: Starting the application
  shell: docker-compose up -d
```

- Inventory

```

inventory 8 x  !  playbook.yml  test.py  JS  logger.js  JS  pizza
inventory > ...
1  Buttercrust App ansible host=43.204.112.104
2
3  [Buttercrust App_group]
4  Buttercrust App
5
6  [Buttercrust App_group:vars]
7  ansible_ssh_common_args='-o StrictHostKeyChecking=no'

```

5.6 Continuous Integration using Jenkins

- Jenkins is an open-source automation tool written in Java with plugins built for Continuous Integration purposes. Jenkins is used to building and testing your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.
- The following steps were followed to install Jenkins in our localhost

```

$ wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
$ sudo apt-get update
$ sudo apt-get install jenkins

```

• Summary of builds

S	W	Name	Last Success	Last Failure	Last Duration
✓	☁	Buttercrust	17 hr #30	17 hr #28	3 min 14 sec

Icon: S M L Icon legend Atom feed for all Atom feed for failures Atom feed for just latest builds

• Pipeline Script from Git SCM

General Build Triggers Advanced Project Options **Pipeline**

Pipeline

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://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@github.com/sethsamrat/Buttercrust-App.git

Credentials ?

sethsamrat/***** Add

Advanced...

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?

*/master

Save Apply

- **Pipeline Code**

- **Stage 1 : Declarative: Checkout SCM**

It pulls code from the GitHub repo for the Jenkins pipeline.

- **Stage 2: Git Clone**

It pulls the remote repository from GitHub using Jenkins.

- **Stage 3: Frontend prerequisite installations**

This step is for building our react app.

- **Stage 4: Backend prerequisite installations**

This step is for building our server.

- **Stage 5: Building the images**

It is used to create images in our local system of the frontend and backend separately.

- **Stage 6: Pushing the images to DockerHub**

The build images are pushed into the public DockerHub repository so that they can be pulled by anyone later on or during docker-compose by us.

- **Stage 7: Ansible Deploy**

This is the deployment stage were using the already build images and the

concept

of containerization we can now execute our app on any platform using the Ansible inventory file and the playbook files.

master Buttercrust-App / Jenkinsfile

Go to file ...

sethsamrat Added Jenkinsfile

Latest commit 8e857ca yesterday History

1 contributor

63 lines (55 sloc) 1.74 KB

Raw Blame

```
1 pipeline {
2   agent any
3   environment {
4     registry1 = "sethsamrat/client"
5     registry2 = "sethsamrat/server"
6     registryCredential = 'docker_cred'
7     dockerImage = ''
8   }
9
10  stages {
11    stage("Git clone") {
12      steps {
13        git url: 'https://github.com/sethsamrat/Buttercrust-App.git', branch: 'master'
14      }
15    }
16    stage("Frontend prerequisite installations") {
17      steps {
18        dir('client'){
19          sh 'npm install'
20        }
21      }
22    }
23
24    stage('Backend prerequisite installations'){
25      steps {
26        dir(''){
27
28          sh 'npm install'
29
30        }
31      }
32    }
33
34    stage('Building the images'){
35      steps {
36        dir('client'){
37          sh 'docker build -t sethsamrat/client .'
38        }
39        dir(''){
40          sh 'docker build -t sethsamrat/server .'
41        }
42      }
43    }
44
45    stage('Pushing the images to DockerHub'){
46      steps{
47        script {
48          withDockerRegistry([ credentialsId: registryCredential, url: "" ])
49          {sh 'docker push $registry1'}
50
51          withDockerRegistry([ credentialsId: registryCredential, url: "" ])
52          {sh 'docker push $registry2'}
53        }
54      }
55    }
56
57    stage('Ansible Deploy') {
58      steps {
59        ansiblePlaybook colorized: true,credentialsId: "container_access_key", disableHostKeyChecking: true, installation: 'Ansible', inventory: 'inve
60      }
61    }
62  }
63 }
```

24

stage('Backend prerequisite installations'){

25

steps {

26

dir(''){

27

sh 'npm install'

28

}

29

}

30

}

31

}

32

}

33

stage('Building the images'){

34

steps {

35

dir('client'){

36

sh 'docker build -t sethsamrat/client .'

37

}

38

dir(''){

39

sh 'docker build -t sethsamrat/server .'

40

}

41

}

42

}

43

}

44

stage('Pushing the images to DockerHub'){

45

steps{

46

script {

47

withDockerRegistry([credentialsId: registryCredential, url: ""])

48

{sh 'docker push \$registry1'}

49

50

withDockerRegistry([credentialsId: registryCredential, url: ""])

51

{sh 'docker push \$registry2'}

52

}

53

}

54

}

55

}

56

stage('Ansible Deploy') {

57

steps {

58

ansiblePlaybook colorized: true,credentialsId: "container_access_key", disableHostKeyChecking: true, installation: 'Ansible', inventory: 'inve

59

}

60

}

61

}

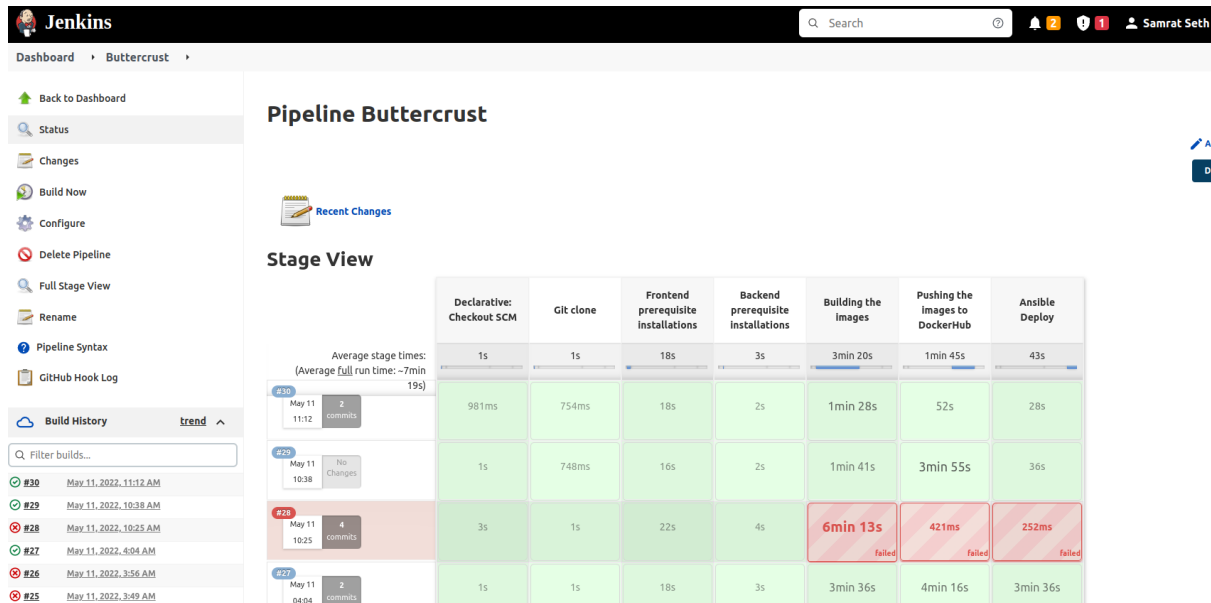
62

}

63

}

- **Stage View**



5.7 Amazon Web Services

- AWS (Amazon Web Services) is a comprehensive, evolving cloud computing the platform provided by Amazon that includes a mixture of infrastructure as a service (IaaS), platform as a service (PaaS) and packaged software as a service (SaaS) offerings. AWS services can offer an organization tools such as computing power, database storage and content delivery services
- AWS launched in 2006 from the internal infrastructure that Amazon.com built to handle its online retail operations. AWS was one of the first companies to introduce a pay-as-you-go cloud computing model that scales to provide users with computing, storage, or throughput as needed.

5.7.1 EC2

- Amazon EC2 (Elastic Compute Cloud) is a web service interface that provides resizable compute capacity in the AWS cloud. It is designed for developers to have complete control over web-scaling and computing resources.

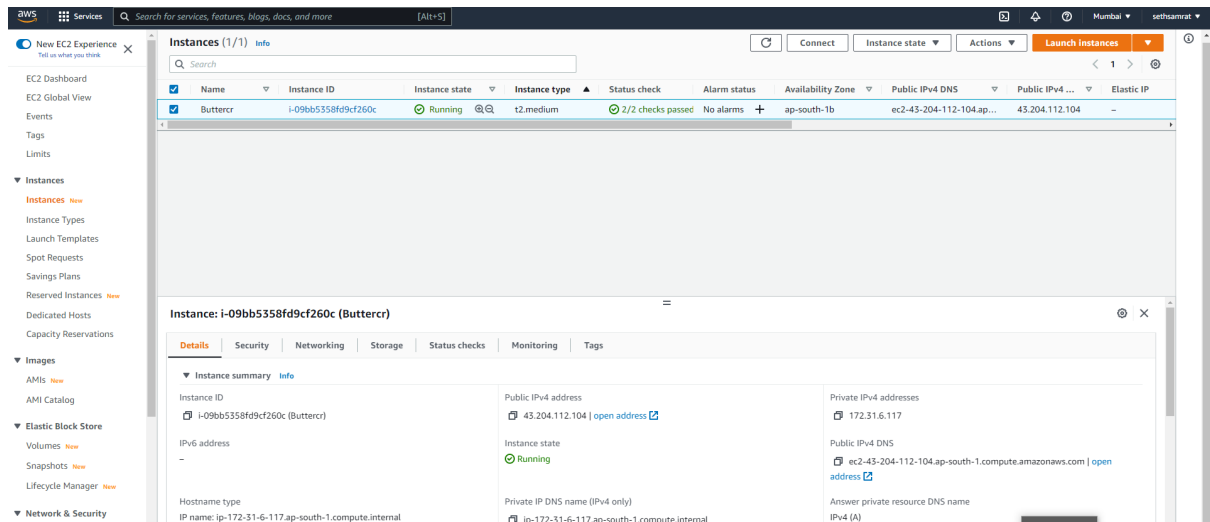
- EC2 instances can be resized and the number of instances scaled up or down as per our requirement. These instances can be launched in one or more geographical locations or regions, and Availability Zones (AZs). Each region comprises several AZs at distinct locations, connected by low latency networks in the same region.

5.7.2 Features of EC2

- **Reliable** – Amazon EC2 offers a highly reliable environment where the replacement of instances is rapidly possible. Service Level Agreement commitment is 99.9% availability for each Amazon EC2 region.
- **Designed for Amazon Web Services** – Amazon EC2 works fine with Amazon services like Amazon S3, Amazon RDS, Amazon DynamoDB, and Amazon SQS. It provides a complete solution for computing, query processing, and storage across a wide range of applications.
- **Secure** – Amazon EC2 works in Amazon Virtual Private Cloud to provide a secure and robust network of resources.
- **Flexible Tools** – Amazon EC2 provides the tools for developers and system administrators to build failure applications and isolate themselves from common failure situations.
- **Inexpensive** – Amazon EC2 wants us to pay only for the resources that we use. It includes multiple purchase plans such as On-Demand Instances, Reserved Instances, Spot Instances, etc. which we can choose as per our requirement.
- The instance of the AWS can be accessed from a local computer with the key provided by the AWS using the command:

`"sudo ssh -i "sethsamrat.pem" ubuntu@ec2-43-204-112-104.ap-south-1.compute.amazonaws.com"`

- **AWS Instance**



- **EC2 instance credentials in Jenkins**

Credentials

T	P	Store	Domain	ID	Name
		Jenkins	(global)	docker_cred	sethsamrat/***** (credential for dockerhub)
		Jenkins	(global)	git_ansible	sethsamrat/***** (access tokens for ansible Jenkins)
		Jenkins	(global)	token-github	sethsamrat/*****
		Jenkins	(global)	container_access_key	ubuntu

Icon: S M L

Stores scoped to Jenkins

P	Store	Domains
	Jenkins	(global)

5.8 Logs and Monitoring

When the application is deployed and running properly on the managed node, we also want to check whether there are any problems in the run time or not. To do that we can implement a monitoring system using the ELK stack.

5.9 WebHooks For Triggering the Pipeline

Webhooks are automated messages that are sent whenever any changes are made. In our case when we make any changes to the GitHub repo, the webhook will

automatically

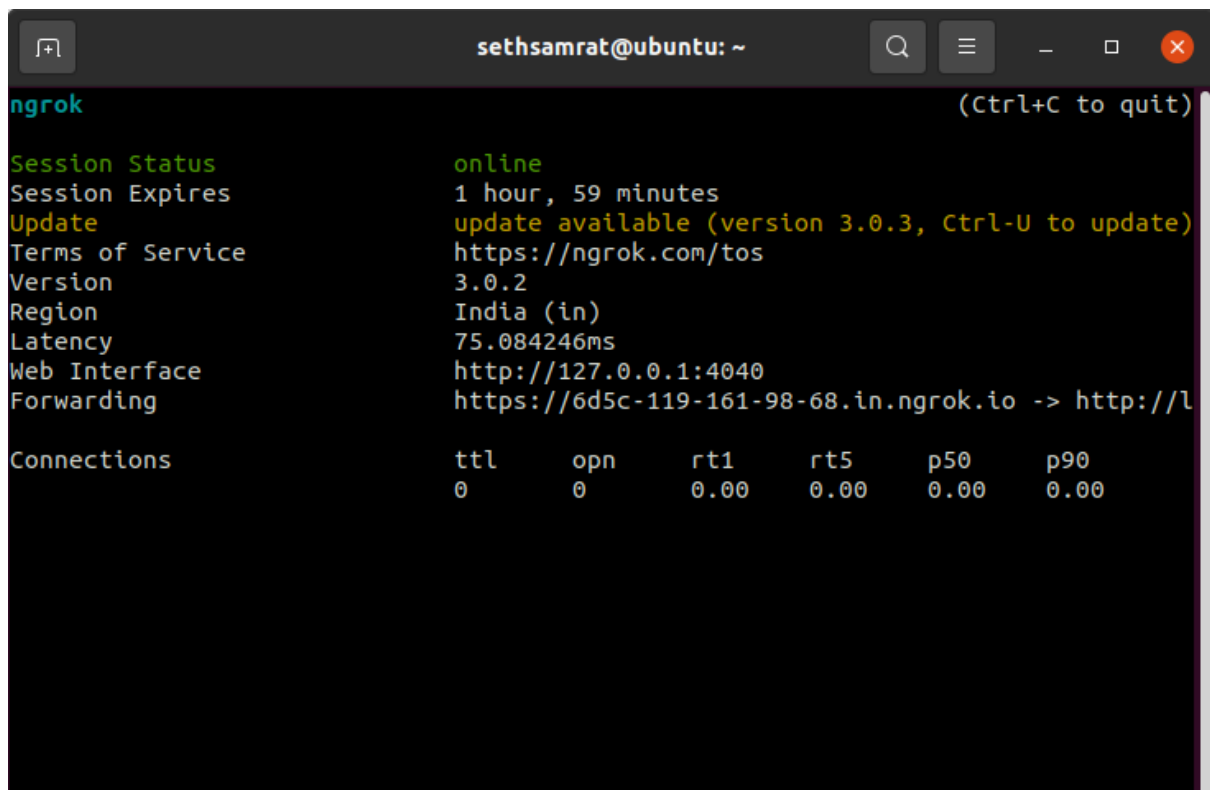
start the Jenkins pipeline.

Ngrok exposes local servers behind NATs (Network Address Translation) and firewalls to

the public internet over secure tunnels. It provides a real-time web UI where you can introspect all HTTP traffic running over your tunnels. It allows you to expose a web server running on your local machine to the internet. Just tell ngrok what port your web

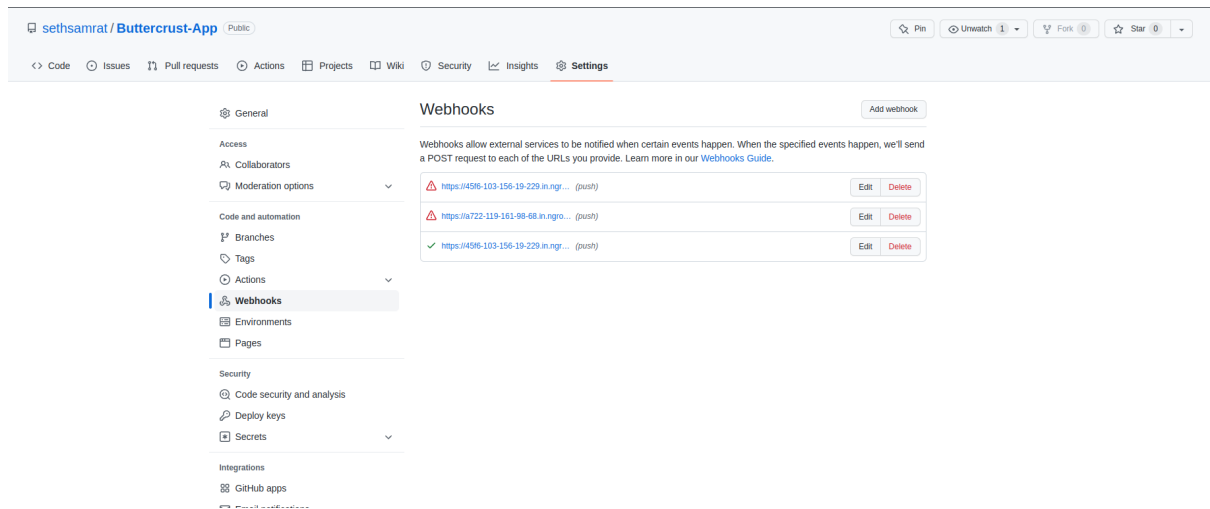
server is listening on.

- **NGROK**



```
sethsamrat@ubuntu: ~  
ngrok (Ctrl+C to quit)  
Session Status      online  
Session Expires     1 hour, 59 minutes  
Update              update available (version 3.0.3, Ctrl-U to update)  
Terms of Service     https://ngrok.com/tos  
Version             3.0.2  
Region              India (in)  
Latency              75.084246ms  
Web Interface        http://127.0.0.1:4040  
Forwarding           https://6d5c-119-161-98-68.in.ngrok.io -> http://l  
  
Connections          ttl      opn      rt1      rt5      p50      p90  
0                   0        0.00     0.00     0.00     0.00
```

- **Setting up webhooks for Buttercrust repository**



- Adding webhooks to Jenkins Location

