



# 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>)
  - 5.2 Testing
  - 5.3 Logger
  - 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
  - 5.9 WebHooks For Triggering the Pipeline
  - 5.10 ELK for continuous monitoring
6. Features and API

<a href="#">6.1 Features</a>
<a href="#">6.2 APIs</a>
<a href="#">6.3 Code Snapshots</a>
<a href="#">7. Key challenges</a>
<a href="#">8. Key learnings</a>
<a href="#">8.1 Technical</a>
<a href="#">8.2 Experience</a>
<a href="#">9. References</a>

## 1. Abstract

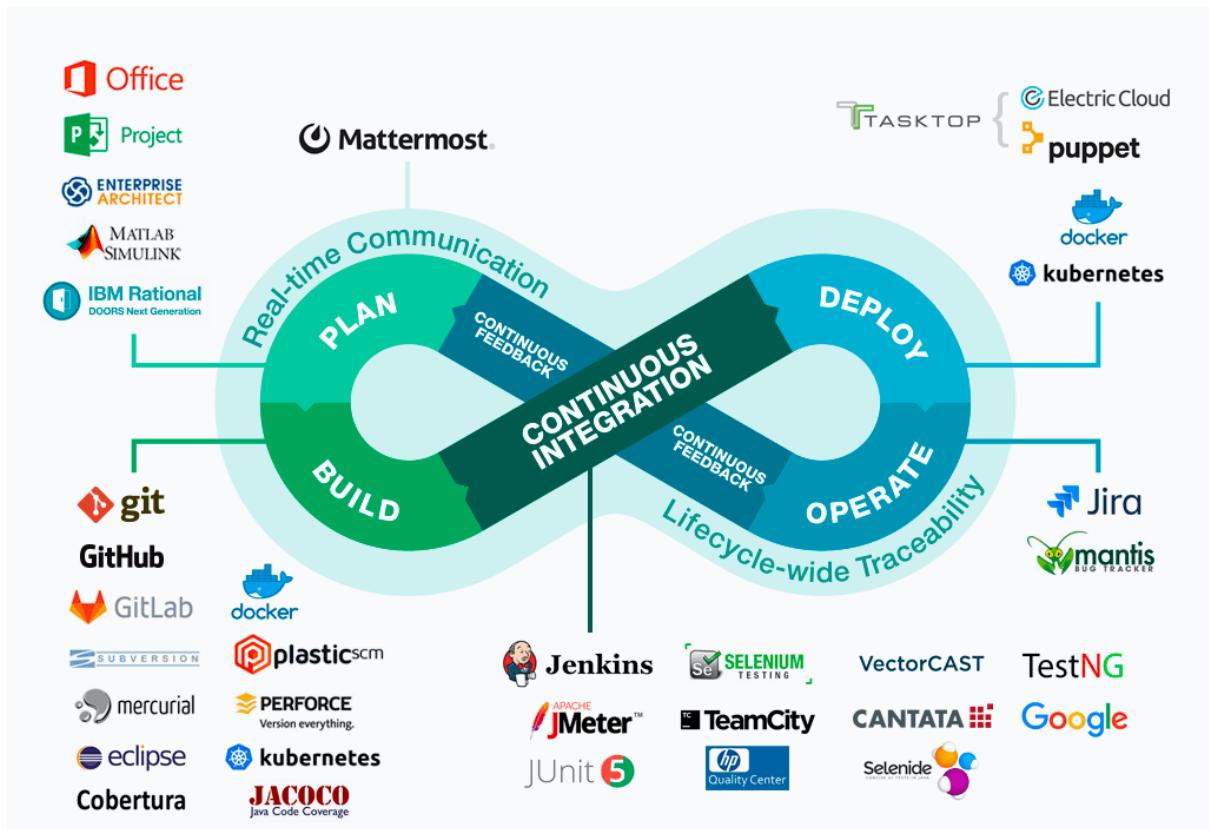
The online Pizza ordering system is a web-based application that enables customers to order their pizzas online for home delivery. Each country has its own kind of dishes to offer. But if we pick a food item that is loved by all the people on this planet, then pizza will be a clear winner in it. The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

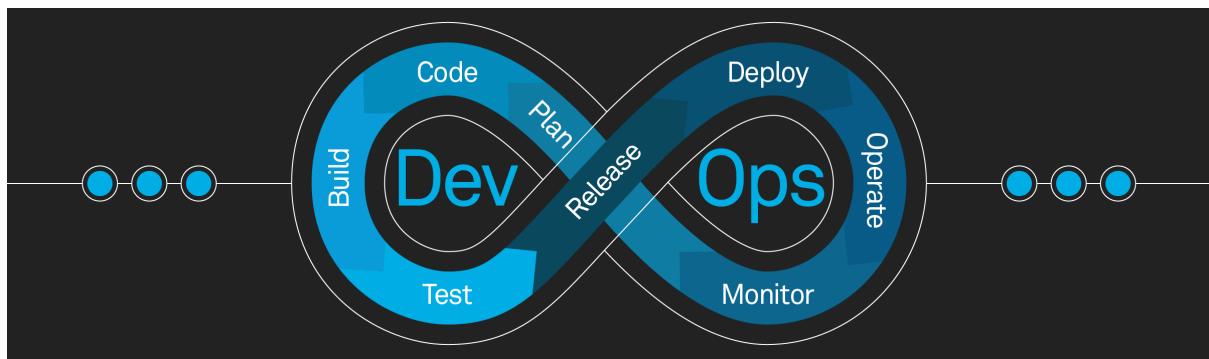


### 3. Why DevOps?

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

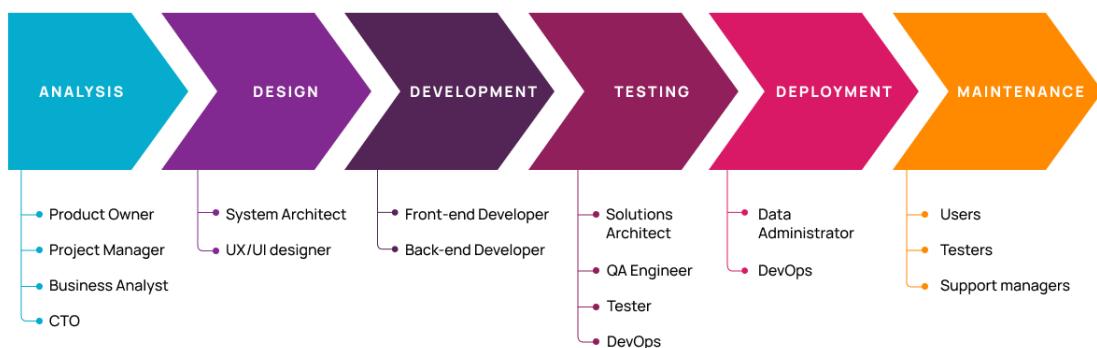
### 4.6 DevOps Tools

- 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](https://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 page for 'sethsamrat/Buttercrust-App'. The repository is public and has 27 commits. The commit history shows various file updates and initial commits. The 'About' section indicates no description, website, or topics provided. It shows 0 stars, 1 watching, and 0 forks. The 'Releases' section shows no releases published. The 'Packages' section shows no packages published. The 'Languages' section shows the code is primarily in JavaScript (92.2%), with smaller portions in CSS (3.1%), HTML (3.0%), Python (1.1%), and Dockerfile (0.6%).

## 5.2 Testing

- We used python for testing our APIs
- Below is the modular code to test our project's APIs
- We just need to pass the API in the function with the request method and it tests the API

```

inventory playbook.yml test.py logger.js pizzasRoute.js ordersRoute.js combined.log userRoute.js Jenkinsfile pack
* test.py > ...
1 import requests as re
2
3 host = "http://43.204.112.104:8000"
4
5 def test_get_api(api_url, params, code=200):
6     api_url = host + api_url
7     response = re.get(api_url, params)
8     assert response.status_code == code
9
10 def test_post_api(api_url, params, code = 200):
11     api_url = host + api_url
12     response = re.post(api_url, json=params)
13     assert response.status_code == code
14
15
16 get_urls = ["/api/pizzas/getallpizzas","/api/pizzas/getpizzabyid"]
17
18 def test(test_apis):
19     for item in test_apis:
20         api, method, params = item
21         if method == "post":
22             test_post_api(api, params)
23         else:
24             test_get_api(api, params)
25
26 test_apis = [("/api/pizzas/getallpizzas", "get", {}),("/api/pizzas/getpizzabyid", "post", {"pizzaid" : "6253b02c0f62ae566cd5a9a4"})]
27 test(test_apis)
28

```

## 5.3 Logger

- Winston Logger

`winston` is designed to be a simple and universal logging library with support for multiple transports. A transport is essentially a storage device for your logs. Each `winston` logger can have multiple transports configured at different levels. For example, one may want error logs to be stored in a persistent remote location (like a database), but all logs output to the console or a local file.

```

inventory playbook.yml test.py logger.js server.js pizzasRoute.js
utils > JS logger.js > ...
1 const winston = require("winston");
2 const { createLogger, format, transports } = require("winston");
3
4 const { combine, timestamp, label, printf } = format;
5
6 const myFormat = printf(({ level, message, label, timestamp }) => {
7   return `${timestamp} ${label} ${level}: ${message}`;
8 });
9
10 const logger = createLogger({
11   format: combine(label({ label: "" }), timestamp(), myFormat),
12   transports: [
13     new transports.Console(),
14     new winston.transports.File({
15       filename: "combined.log",
16       level: "info",
17     }),
18   ],
19 });
20
21 module.exports = logger;

```

- Logfile

```

combined.log
1 2022-05-11T15:09:34.161Z info: Get all pizzas called
2 2022-05-11T15:09:34.280Z info: Get all pizzas successful
3 2022-05-11T15:11:43.245Z info: Register user called
4 2022-05-11T15:11:43.255Z info: User registered successfully
5 2022-05-11T15:11:52.152Z info: Signin Called
6 2022-05-11T15:11:52.204Z error: Signin Failed
7 2022-05-11T15:11:59.193Z info: Signin Called
8 2022-05-11T15:11:59.248Z info: Signin Successfull
9 2022-05-11T15:11:59.961Z info: Get all pizzas called
10 2022-05-11T15:12:00.020Z info: Get all pizzas successful
11 2022-05-11T16:53:30.736Z info: Get all pizzas called
12 2022-05-11T16:53:30.980Z info: Get all pizzas successful
13 2022-05-11T16:55:50.808Z info: Get all pizzas called
14 2022-05-11T16:55:50.843Z info: Get all pizzas successful
15 2022-05-11T17:00:02.761Z info: Get all pizzas called
16 2022-05-11T17:00:02.792Z info: Get all pizzas successful
17 2022-05-11T17:02:57.580Z info: Signin Called
18 2022-05-11T17:02:57.614Z info: Signin Successfull
19 2022-05-11T17:02:58.781Z info: Get all pizzas called
20 2022-05-11T17:02:58.811Z info: Get all pizzas successful
21 2022-05-11T17:03:42.527Z info: Get all pizzas called
22 2022-05-11T17:03:42.559Z info: Get all pizzas successful
23 2022-05-11T17:04:21.813Z info: Get all pizzas called
24 2022-05-11T17:04:21.845Z info: Get all pizzas successful
25 2022-05-11T17:04:23.465Z info: Get all pizzas called
26 2022-05-11T17:04:23.496Z info: Get all pizzas successful
27 2022-05-11T17:07:24.009Z info: Order place request
28 2022-05-11T17:07:26.894Z info: Order placed successfully
29 2022-05-11T17:08:06.632Z info: Get user orders called
30 2022-05-11T17:08:06.681Z info: Get user orders successful
31 2022-05-11T17:12:36.548Z info: user info called
32 2022-05-11T17:12:36.606Z info: user info successfull
33 2022-05-11T17:12:38.009Z info: Get all pizzas called
34 2022-05-11T17:12:38.060Z info: Get all pizzas successful
35 2022-05-11T17:13:33.697Z info: user info called
36 2022-05-11T17:13:33.820Z info: user info successfull
37 2022-05-11T17:13:35.201Z info: Get all pizzas called
38 2022-05-11T17:13:35.246Z info: Get all pizzas successful
39 2022-05-11T17:13:38.707Z info: Get all pizzas called
40 2022-05-11T17:13:38.760Z info: Get all pizzas successful

```

## 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 x 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**

```
inventory 8 Dockerfile x playbook.yml t
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**

```

inventory 8 docker-compose.yml × playbo
+ docker-compose.yml
  version: '3'
  services:
    api-server:
      image: sethsamrat/server
      ports:
        - "8000:8000"
    networks:
      - mern-app
    container_name: server
    command: npm start
  react-app:
    image: sethsamrat/client
    stdin_open: true
    ports:
      - "3000:3000"
    networks:
      - mern-app
    command: npm start
  networks:
    mern-app:
      driver: bridge

```

- **Repositories in DockerHub**

Repository	Last pushed	Status	Stars	Downloads	Access
sethsamrat / server	18 hours ago	Not Scanned	0	3	Public
sethsamrat / client	18 hours ago	Not Scanned	0	3	Public
sethsamrat / buttercrust	Never	Not Scanned	0	0	Public
sethsamrat / calcimage	a month ago	Not Scanned	0	1	Public

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

- **Client Repository**

**Docker commands**

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

Tags and Scans		VULNERABILITY SCANNING - DISABLED	
This repository contains 1 tag(s).		<a href="#">Enable</a>	
TAG	OS	PULLED	PUSHED
latest	Ubuntu	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**

**Docker commands**

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

Tags and Scans		VULNERABILITY SCANNING - DISABLED	
This repository contains 1 tag(s).		<a href="#">Enable</a>	
TAG	OS	PULLED	PUSHED
latest	Ubuntu	---	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](#)

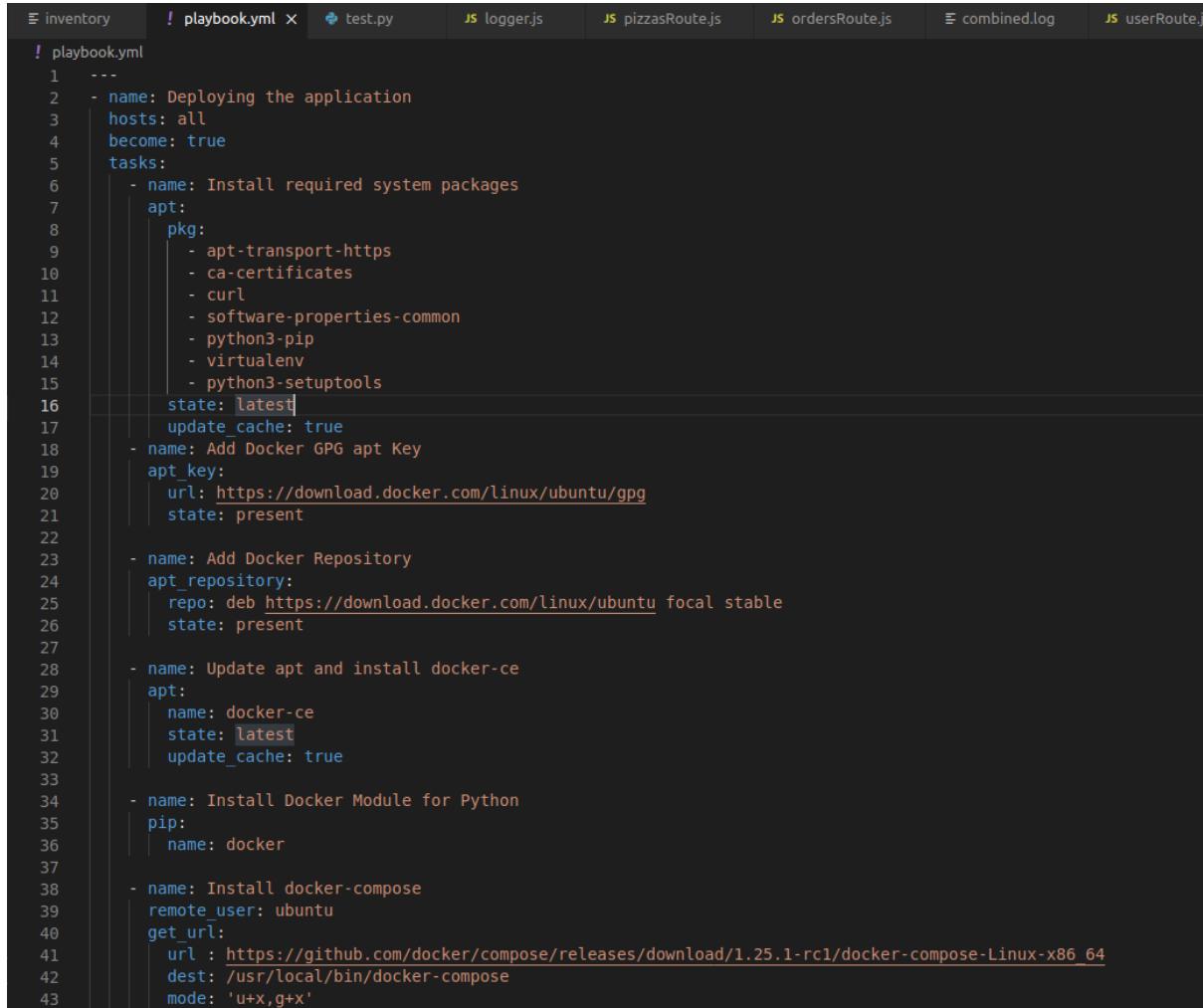
## 5.5 Deployment using Ansible (Deployed on AWS)

- 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
23      - name: Add Docker Repository
24        apt_repository:
25          repo: deb https://download.docker.com/linux/ubuntu focal stable
26          state: present
27
28      - name: Update apt and install docker-ce
29        apt:
30          name: docker-ce
31          state: latest
32          update_cache: true
33
34      - name: Install Docker Module for Python
35        pip:
36          name: docker
37
38      - name: Install docker-compose
39        remote_user: ubuntu
40        get_url:
41          url : https://github.com/docker/compose/releases/download/1.25.1-rc1/docker-compose-Linux-x86_64
42          dest: /usr/local/bin/docker-compose
43          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**

```

! playbook.yml • 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**

All +

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://ghp\_7DVgD5x9QCTjz1fpSl90T7cQEtwJGc2ZjMKZ@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**

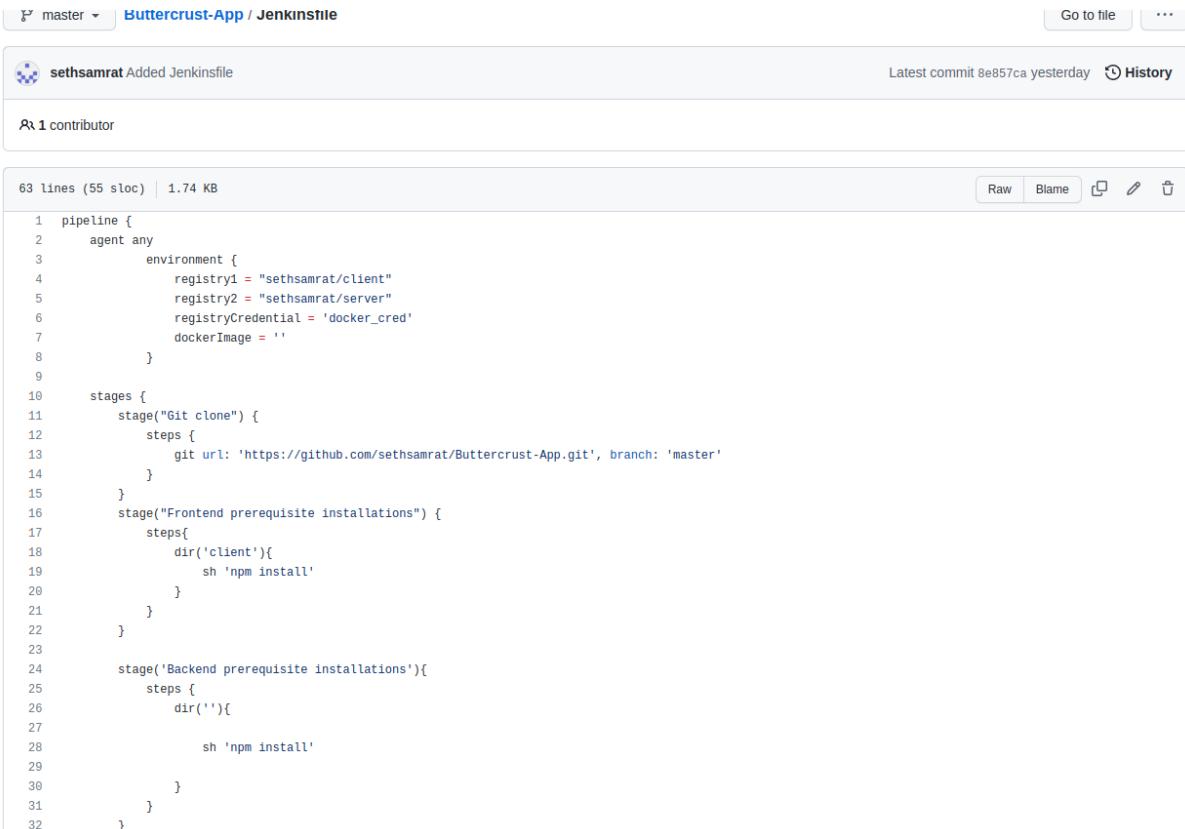
It is used to create images in our local system of the frontend and backend separately.

- o **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.

- o **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.



A screenshot of a GitHub commit page for a Jenkinsfile. The commit was made by user 'sethsamrat' and has been pushed to the 'master' branch. The commit message is 'Added Jenkinsfile'. The commit was made yesterday at 8e857ca. The Jenkinsfile contains 63 lines (55 sloc) and is 1.74 KB in size. The file content is as follows:

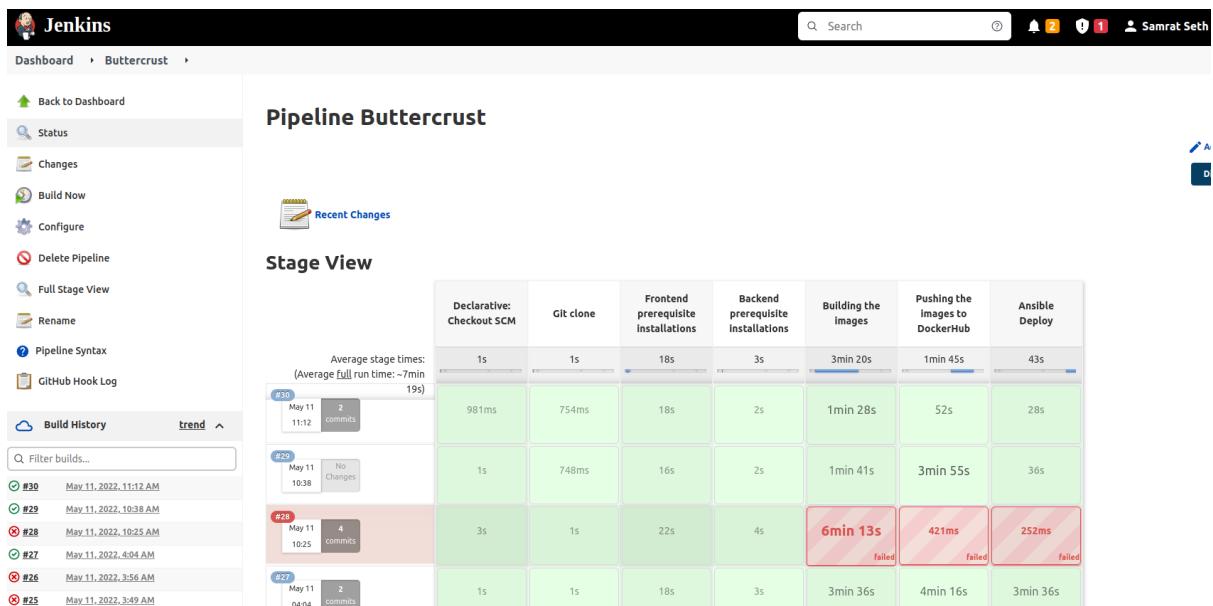
```
1 pipeline {
2     agent any
3         environment {
4             registry1 = "sethsamrat/client"
5             registry2 = "sethsamrat/server"
6             registryCredential = 'docker_cred'
7             dockerImage = ''
8         }
9
10    stages {
11        stage("Git clone") {
12            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                    sh 'npm install'
28                }
29            }
30        }
31    }
32 }
```

```

24      stage('Backend prerequisite installations'){
25          steps {
26              dir{}{
27                  sh 'npm install'
28              }
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 colored: true,credentialsId: "container_access_key", disableHostKeyChecking: true, installation: 'Ansible', inventory: 'inven
60     }
61 }
62 }
63 }

```

- Stage View



## 5.7 Amazon Web Services

- AWS (Amazon Web Services) is a comprehensive, evolving cloud computing 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**

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with various navigation options like EC2 Dashboard, Global View, Events, Tags, Limits, Instances, Images, AMIs, Elastic Block Store, Volumes, Snapshots, and Network & Security. The main area displays a table of instances. One instance is listed: 'Buttercr' (Instance ID: i-09bb5358fd9cf260c). The instance is running, t2.medium type, 2/2 checks passed, and located in ap-south-1b. It has a public IPv4 address of 43.204.112.104 and a private IP of 172.31.6.117. A detailed view of this instance is shown in a modal window.

- EC2 instance credentials in Jenkins**

The screenshot shows the Jenkins Credentials page. It lists four entries under the 'Jenkins' store, all categorized as '(global)'. The entries are: 'docker\_cred' (sethsamrat/\*\*\*\*\* credential for dockerhub), 'git\_ansible' (sethsamrat/\*\*\*\*\* access tokens for ansible jenkins), 'token-github' (sethsamrat/\*\*\*\*\*), and 'container\_access\_key' (ubuntu). Below the table, it says 'Stores scoped to Jenkins' and shows 'Domains' with 'Jenkins' selected and '(global)' as the domain.

## 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: ~
(Ctrl+C to quit)

ngrok

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**

## 5.10 ELK for continuous monitoring

It is a technology and process that IT organizations use to get near-immediate feedback and insight into performance and interactions across the network, which helps drive operational, security, and business performance. After doing deployment the next major thing which needs to be done is monitoring. Monitoring involves checking whether the software is behaving as it is meant to be or not.

ELK stack makes the monitoring tool for any deployed software, it analyzes the logs and the same analysis can then be viewed on the kibana dashboard. ELK stack comprises 3 independent components: Elasticsearch, Logstash, and Kibana.

- Elasticsearch is an open-source, full-text search and analysis engine.
- Logstash is a log aggregator that collects data from various input sources and then ships the data to various supported output destinations.

- Kibana is a visualization layer that works on top of Elasticsearch, providing users with the ability to analyze and visualize the data.
- **Uploading the log file**

The screenshot shows the Elastic Cloud interface with the 'Upload file' tab selected. A large central box titled 'Visualize data from a log file' contains instructions: 'Upload your file, analyze its data, and optionally import the data into an Elasticsearch index.' It lists supported file formats: Delimited text files (CSV and TSV), Newline-delimited JSON, and Log files with a common format for the timestamp. A note says you can upload files up to 100 MB. Below this is a file upload area with a 'Select or drag and drop a file' placeholder.

- **Importing the log file**

The screenshot shows the 'More ways to add data' section. A file named 'combined.log' is selected. The 'File contents' section displays the first 95 lines of the log file, which consists of timestamped log entries. At the bottom, there are 'Import' and 'Cancel' buttons.

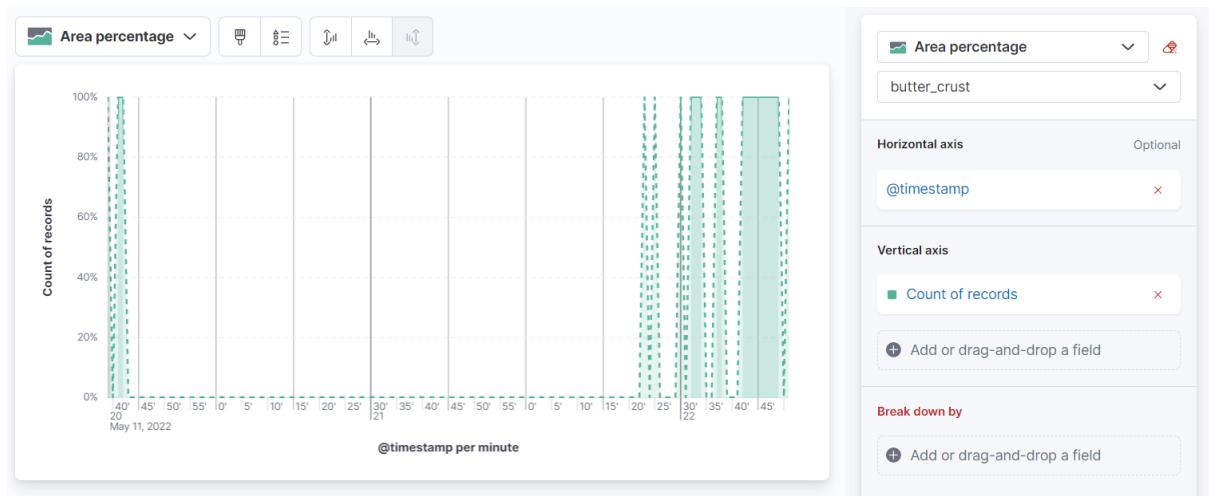
```

File contents
First 95 lines
1 2022-05-11T15:09:34.161Z info: Get all pizzas called
2 2022-05-11T15:09:34.280Z info: Get all pizzas successful
3 2022-05-11T15:11:43.245Z info: Register user called
4 2022-05-11T15:11:43.255Z info: User registered successfully
5 2022-05-11T15:11:52.152Z info: Signin Called
6 2022-05-11T15:11:59.153Z error: Signin Failed
7 2022-05-11T15:11:59.153Z info: Signin Called
8 2022-05-11T15:11:59.248Z info: Signin Successfull
9 2022-05-11T15:11:59.961Z info: Get all pizzas called
10 2022-05-11T15:12:00.020Z info: Get all pizzas successful
11 2022-05-11T16:55:59.736Z info: Get all pizzas called
12 2022-05-11T16:55:59.805Z info: Get all pizzas successful
13 2022-05-11T16:55:59.808Z info: Get all pizzas called
14 2022-05-11T16:55:59.843Z info: Get all pizzas successful

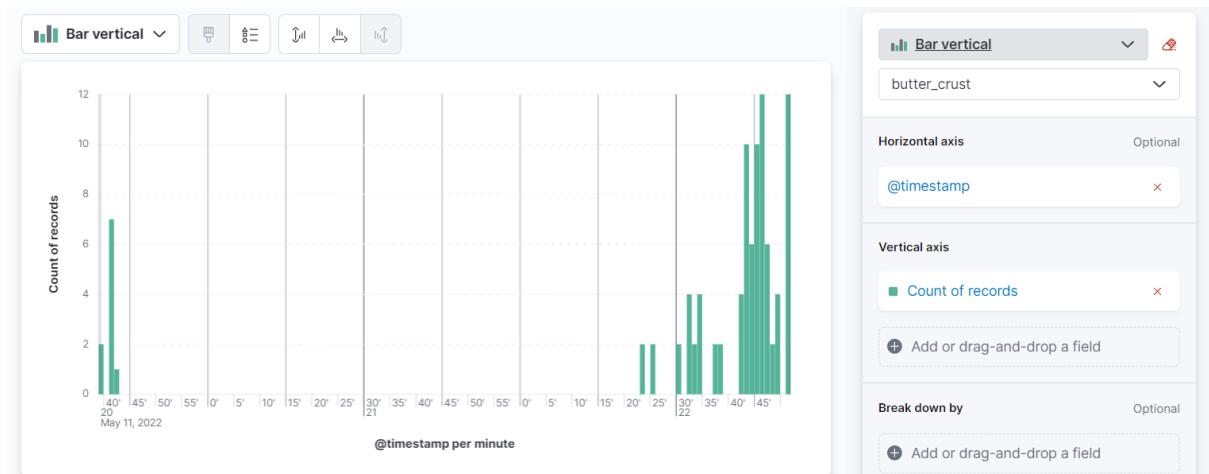
```

- **Giving an index name and adding the file**

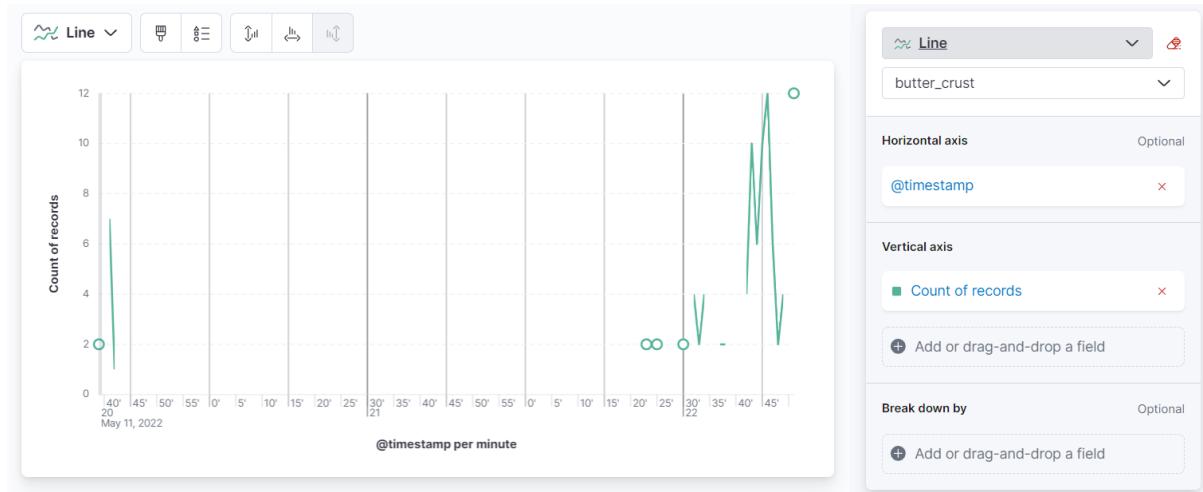
- **Area Percentage**



- **Bar Vertical**



- **Line Graph**

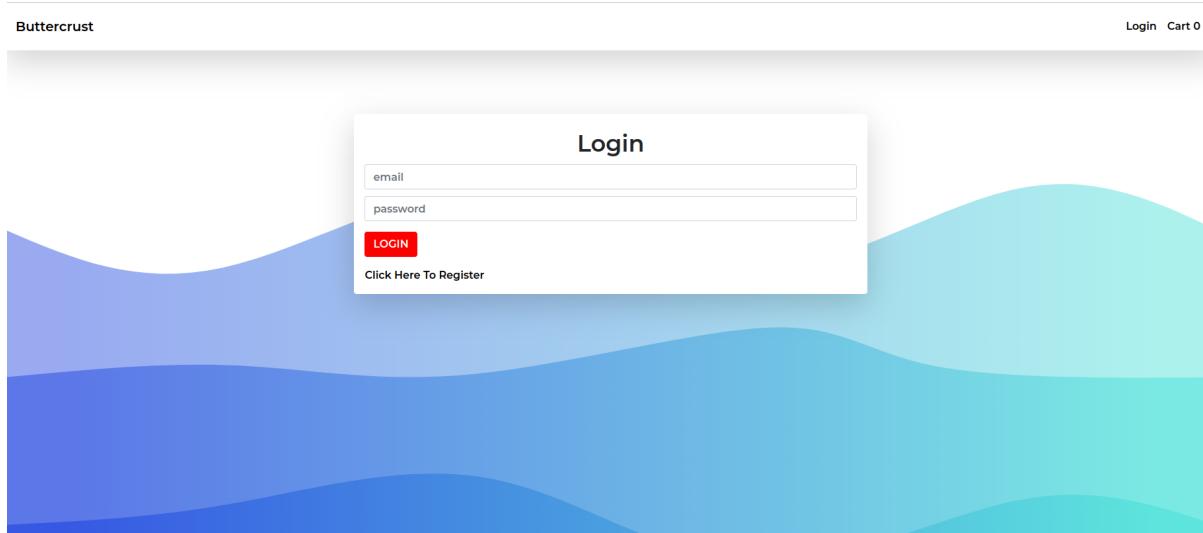


## 6. Features and API

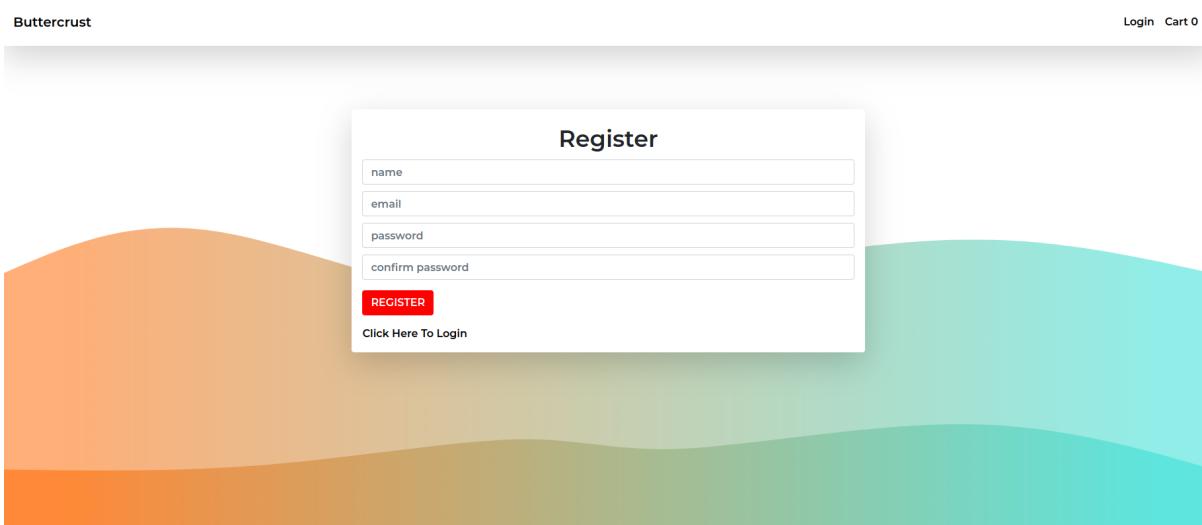
### 6.1 Features

This application has the following functionalities:

- **Login Page**



- **Update quantity in cart**



### • User Dashboard

A screenshot of the user dashboard displaying three pizza options: "PEPPER BARBECUE CHICKEN", "Non Veg Supreme", and "Golden Corn Pizza". Each listing includes a thumbnail image, a "variants" dropdown, a "Quantity" dropdown set to 1, and a red "ADD TO CART" button. The prices are listed as "Price : 200 Rs/-" for the first two and "Price : 180 Rs/-" for the third. A search bar at the top left and a filter button at the top right are also visible.

### • Pizza Filter

A screenshot of the user dashboard after applying a filter for "Veg" pizzas. It shows three new options: "Golden Corn Pizza", "Jalapeno &amp; Red Paprika Pizza", and "Margherita". Each listing includes a thumbnail image, a "variants" dropdown, a "Quantity" dropdown set to 1, and a red "ADD TO CART" button. The prices are listed as "Price : 180 Rs/-" for the first, "Price : 200 Rs/-" for the second, and "Price : 150 Rs/-" for the third.

- **Cart Page**

Buttercrust abc ▾ Cart 2

**My Cart**

**SubTotal : 800 /-**

**Pay Now**

PEPPER BARBECUE CHICKEN [small]  
Price : 3 \* 200 = 600  
Quantity : +3 -

Non Veg Supreme [small]  
Price : 1 \* 200 = 200  
Quantity : +1 -

- **Order Placed**

Buttercrust abc ▾ Cart 2

**My Cart**

**SubTotal : 800 /-**

**Your Order Placed Successfully**

**Pay Now**

PEPPER BARBECUE CHICKEN [small]  
Price : 3 \* 200 = 600  
Quantity : +3 -

Non Veg Supreme [small]  
Price : 1 \* 200 = 200  
Quantity : +1 -

- **Orders Page**

## My Orders

Items	Address	Order Info
PEPPER BARBECUE CHICKEN [small] * 3 = 600	Street : bangalore	Order Amount : 800
Non Veg Supreme [small] * 1 = 200	City : Bangalore	Date : 2022-05-11
	Country : India	Transaction Id : card_1KyJ00SIR2AbPxU0Nkq7hqdx
	Pincode : 560100	Order Id : 627bed4e6lc6831822e69508

- **Stripe Payment Gateway**

Buttercrust abc TEST MODE

### My Cart

PEPPER BARBECUE CHICKEN [small]  
Price : 3 \* 200 = 600  
Quantity : +3 -

Non Veg Supreme [small]  
Price : 1 \* 200 = 200  
Quantity : +1 -

SubTotal : 800 /-

Pay Now

Same billing & shipping info

City

Payment Info

Powered by stripe

- **Card Details**

Buttercrust abc TEST MODE

### My Cart

PEPPER BARBECUE CHICKEN [small]  
Price : 3 \* 200 = 600  
Quantity : +3 -

Non Veg Supreme [small]  
Price : 1 \* 200 = 200  
Quantity : +1 -

SubTotal : 800 /-

Pay Now

4242 4242 4242 4242

05 / 23

230

Pay INR ₹800.00

Powered by stripe

- Admin Panel - Users List

The screenshot shows the Admin Panel - Users List page. At the top, there is a navigation bar with links for 'Users List', 'Pizzas List', 'Add Pizza', and 'Orders List'. Below the navigation bar is a table titled 'Users list' with four columns: 'User Id', 'Name', 'Email', and 'Delete'. The table contains four rows of data, each with a unique User Id, a name (abc, bcd, def, fgh), an email address (abc@gmail.com, bcd@gmail.com, def@gmail.com, fgh@gmail.com), and a 'Delete' button.

User Id	Name	Email	Delete
627bd22f61c6831822e69507	abc	abc@gmail.com	
627bef55acfaf8279c9760a2	bcd	bcd@gmail.com	
627bef5facfaf8279c9760a3	def	def@gmail.com	
627bef70acfaf8279c9760a4	fgh	fgh@gmail.com	

- Admin Panel - Pizzas List

The screenshot shows the Admin Panel - Pizzas List page. At the top, there is a navigation bar with links for 'Users List', 'Pizzas List', 'Add Pizza', and 'Orders List'. Below the navigation bar is a table titled 'Pizzas List' with four columns: 'Name', 'Prices', 'Category', and 'Actions'. The table contains six rows of data, each with a pizza name, its prices (Small, Medium, Large), its category (nonveg or veg), and a 'Delete' button with a checkmark icon.

Name	Prices	Category	Actions
PEPPER BARBECUE CHICKEN	Small : 200 Medium : 350 Large : 400	nonveg	
Non Veg Supreme	Small : 200 Medium : 350 Large : 400	nonveg	
Golden Corn Pizza	Small : 180 Medium : 250 Large : 360	veg	
Jalapeno & Red Paprika Pizza	Small : 200 Medium : 300 Large : 420	veg	
Margherita	Small : 150 Medium : 220 Large : 300	veg	

- Admin Panel - Adding New Pizzas to the Menu

Buttercrust abc Cart 2

### Admin Panel

- [Users List](#)
- [Pizzas List](#)
- [Add Pizza](#)
- [Orders List](#)

**Add Pizza**








[Add Pizza](#)

- **Admin Panel - Orders List**

Buttercrust abc Cart 3

### Admin Panel

- [Users List](#)
- [Pizzas List](#)
- [Add Pizza](#)
- [Orders List](#)

Order Id	Email	User Id	Amount	Date	Status
627bed4e61c6831822e69508	abc@gmail.com	627bd22f61c6831822e69507	800	2022-05-11	<a href="#" style="background-color: red; color: white; padding: 2px 5px; border-radius: 5px;">Deliver</a>
627bf084acfaf8279c9760a5	abc@gmail.com	627bd22f61c6831822e69507	980	2022-05-11	Delivered

Show Applications

## 6.2 APIs

Serial no.	Use Case	API	Request Method	End-user
1.	User Registration	/api/users/register	POST	User
2.	User Login	/api/users/login	POST	User
3.	Get a list of all registered users	/api/users/getallusers	GET	Admin
4.	Delete a user	/api/users/deleteuser	POST	Admin
5.	Get a list of all pizzas	/api/pizzas/getallpizzas	GET	User
6.	Add new pizza to the menu	/api/pizzas/addpizza	POST	Admin

7.	Get a specific pizza	/api/pizzas/getpizzabyid	POST	User
8.	Update pizza description	/api/pizzas/editpizza	POST	Admin
9.	Remove a pizza from the menu	/api/pizzas/deletepizza	POST	Admin
10.	Place order	/api/orders/placeorder	POST	User
11.	Get the orders of a user	/api/orders/getuserorders	POST	User
12.	Get a list of all orders	/api/orders/getallorders	GET	Admin
13.	Update the delivery status	/api/orders/deliverorder	POST	Admin

### 6.3 Code Snapshots

- **Server.js**

```

JS server.js > ...
1 const express = require("express");
2
3 const Pizza = require('./models/pizzaModel')
4
5 const app = express();
6 const db = require("./db.js")
7 app.use(express.json());
8 const path = require('path')
9 const pizzasRoute = require('./routes/pizzasRoute')
10 const userRoute = require('./routes/userRoute')
11 const ordersRoute = require('./routes/ordersRoute')
12
13
14
15
16
17 app.use('/api/pizzas/' , pizzasRoute)
18 app.use('/api/users/' , userRoute)
19 app.use('/api/orders/' , ordersRoute)
20
21
22 if(process.env.NODE_ENV ==='production')
23 {
24     app.use('/' , express.static('client/build'))
25
26     app.get('*' , (req , res)=>{
27
28         res.sendFile(path.resolve(__dirname , 'client/build/index.html'))
29
30     })
31 }
32
33
34
35
36
37
38 const port = process.env.PORT || 8000;
39
40 app.listen(port, () => `Server running on port port 🔥`)

```

- **ordersRoute.js**

```

routes > js ordersRoutejs > ...
1 const express = require("express");
2 const router = express.Router();
3 const { v4: uuidv4 } = require('uuid');
4 const stripe = require("stripe")("sk_test_51IYnC05IR2AbPxU0ElMx1fTwzbZXLbka0cbc2cXx49528d9TGkQVjUINJfUDAnQMVaBFfBDP5xtcHCKZGln1V3E800U7qXFmGf");
5 const Order = require('../models/orderModel');
6 const logger = require("./utils/logger");
7
8 router.post("/placeorder", async (req, res) => {
9
10   logger.log({
11     level: "info",
12     message: "Order place request",
13   });
14   const { token, subtotal, currentUser, cartItems } = req.body
15
16   try {
17     const customer = await stripe.customers.create({
18       email: token.email,
19       source: token.id
20     })
21
22     const payment = await stripe.charges.create({
23       amount: subtotal * 100,
24       currency: 'inr',
25       customer: customer.id,
26       receipt_email: token.email
27     }, {
28       idempotencyKey: uuidv4()
29     })
30
31     if (payment) {
32
33       const neworder = new Order({
34         name: currentUser.name,
35         email: currentUser.email,
36         userid: currentUser._id,
37         orderItems: cartItems,
38         orderAmount: subtotal,
39         shippingAddress: {
40           street: token.card.address_line1,
41           city: token.card.address_city,
42           country: token.card.address_country,
43

```

## 7. Key challenges

- The challenges we faced initially were related to docker-compose. We were able to create the images properly and both the containers were running as well but the frontend and backend were unable to communicate with each other. The reason being we did not provide the container name in the docker-compose file. The fix was to put the container name in the docker-compose file and then use the same “*container name*” as URI in the API instead of “*localhost*”

## 8. Key learnings

### 8.1 Technical

- React.js
- Node.js
- Jenkins
- Docker
- Ansible
- Amazon Web Services

### 8.2 Experience

- Designing this application has taught us the power of collaborating and the importance of being a team player.

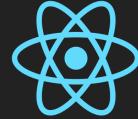
- Creating the application from scratch and aligning it with the team's vision and choosing the correct people for the appropriate task have taught us to believe and support each other.

## 9. References

### Getting Started - React

A JavaScript library for building user interfaces

 <https://reactjs.org/docs/getting-started.html>



<https://docs.docker.com/get-started/>

### User Guide - Ansible Documentation

Welcome to the Ansible User Guide! This guide covers how to work with Ansible, including using the command line, working with inventory, interacting with data, writing tasks, plays, and playbooks; executing

 [https://docs.ansible.com/ansible/latest/user\\_guide/index.html#getting-started](https://docs.ansible.com/ansible/latest/user_guide/index.html#getting-started)

Ansible Automation Platform

Extend the power  
of Ansible to your  
entire team

Try it free

### Tutorials overview

The following tutorials show how to use Jenkins to cover the basics of CI/CD concepts based on specific technology stacks. Choose the tutorial that's relevant to your technology stack or one that you're most

 <https://www.jenkins.io/doc/tutorials/#pipeline/>



# Jenkins

### Index | Node.js v18.1.0 Documentation

 <https://nodejs.org/docs/latest-v10.x/api/>

### What is Amazon EC2?

Amazon Elastic Compute Cloud (Amazon EC2) provides scalable computing capacity in the Amazon Web Services (AWS) Cloud. Using Amazon EC2 eliminates your need to invest in hardware up front, so you can develop and deploy applications faster. You can use Amazon EC2 to launch as many or as few virtual servers as you need,

 <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>