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)

1. Abstract

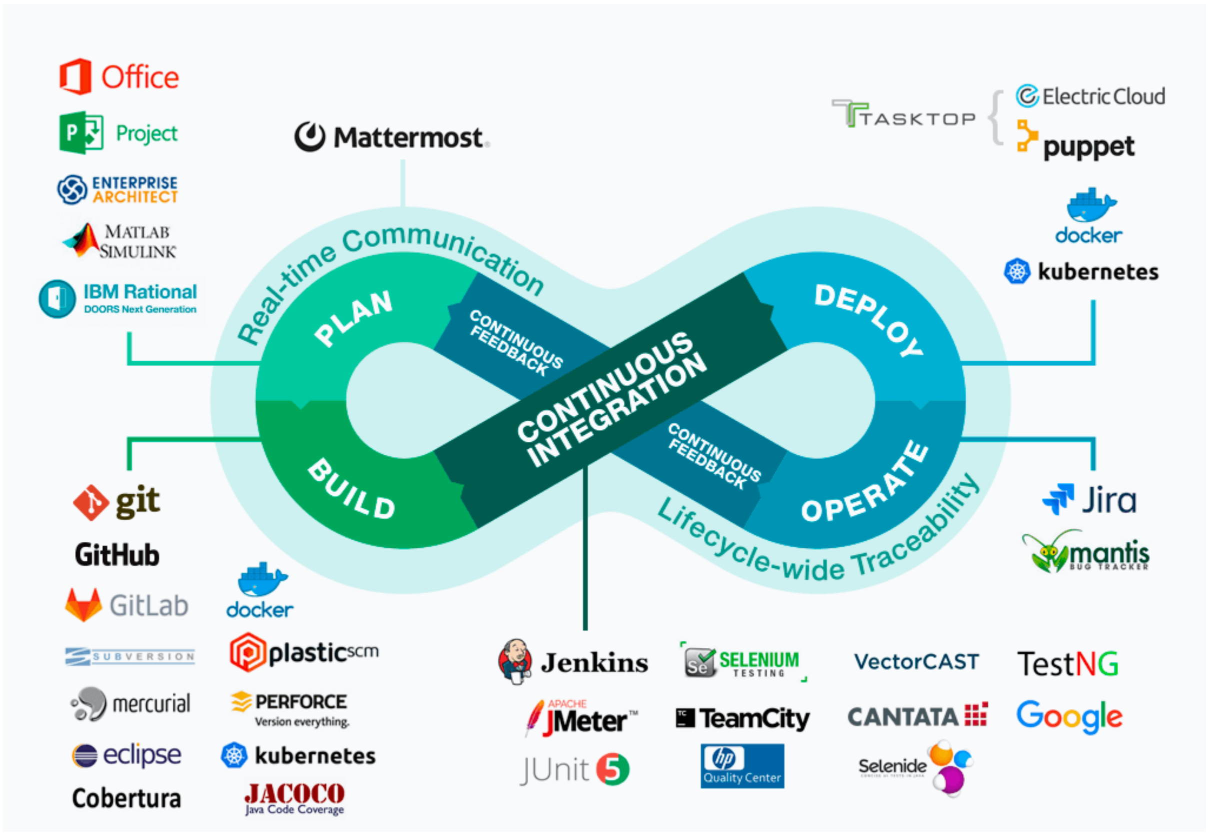
The online Pizza ordering system is a web-based application that enables customers to order their pizzas online for home delivery. Each country has its own kind of dishes to offer. But if we pick a food item that is loved by all the people on this planet, then pizza will be a clear winner in it. The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

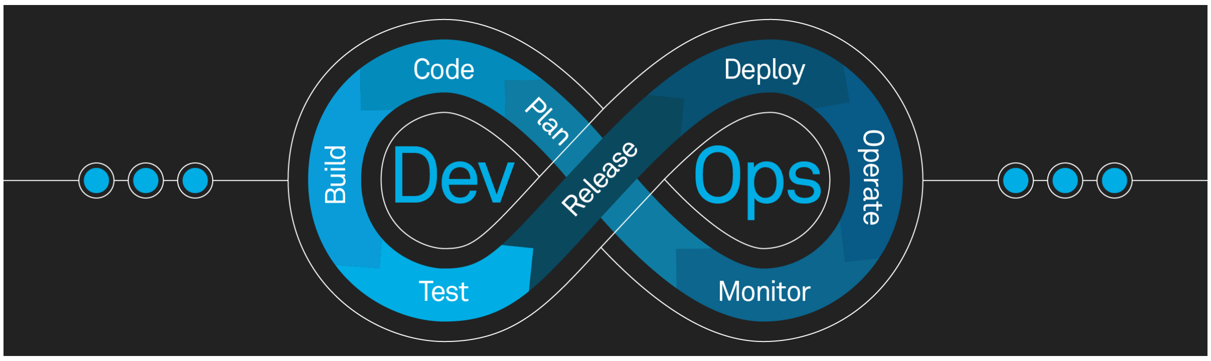
2. What is DevOps?

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

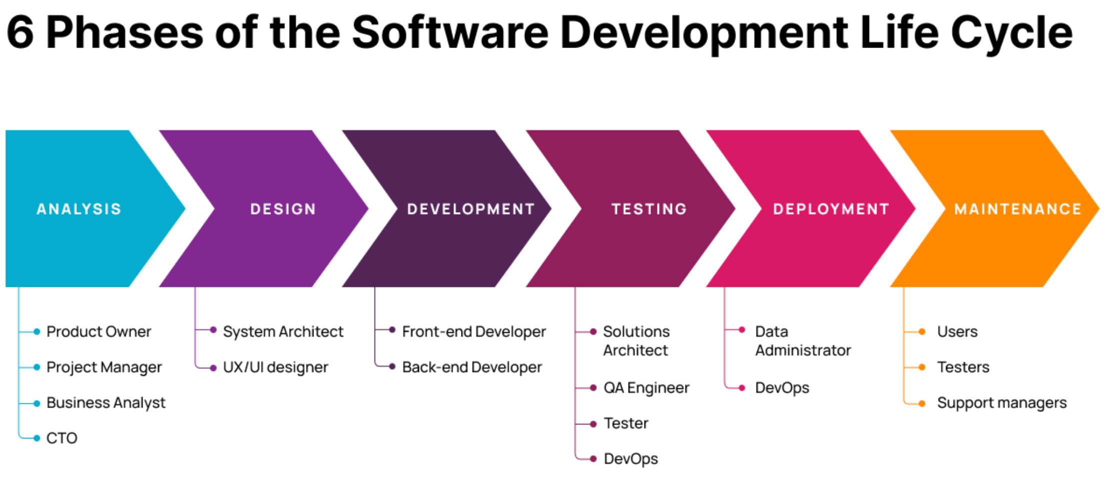
- Type - T2.medium
- OS - Ubuntu 20.04
- Ram - 2 GiB
- Storage - 15 GiB

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.

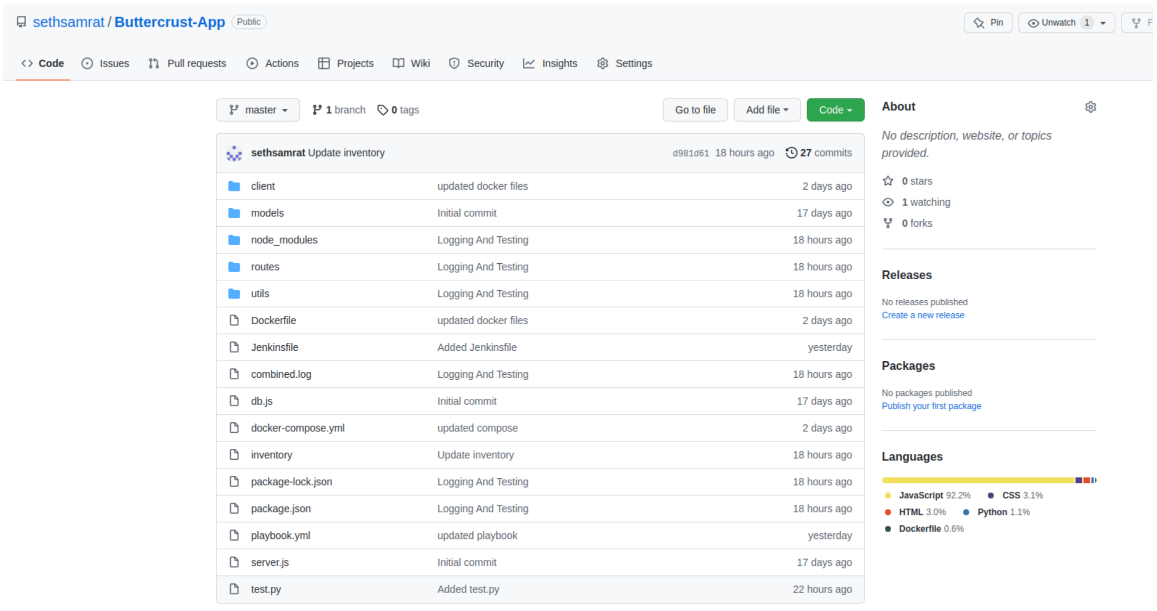
```
$ git commit -m “message”
```

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


sethsamrat / Buttercrust-App Public

Pin | Unwatch 1

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

master | 1 branch | 0 tags | Go to file | Add file | Code

sethsamrat Update inventory d981d61 18 hours ago 27 commits

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

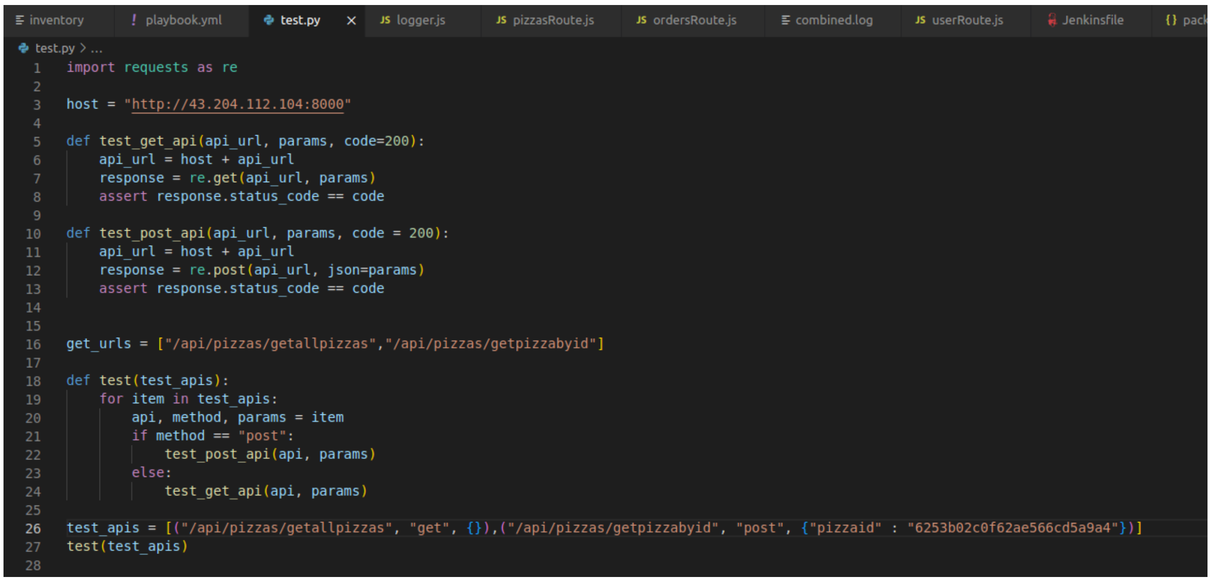
No packages published
Publish your first package

Languages

● JavaScript 92.2% ● CSS 3.1%
● HTML 3.0% ● Python 1.1%
● Dockerfile 0.6%

5.3 Testing

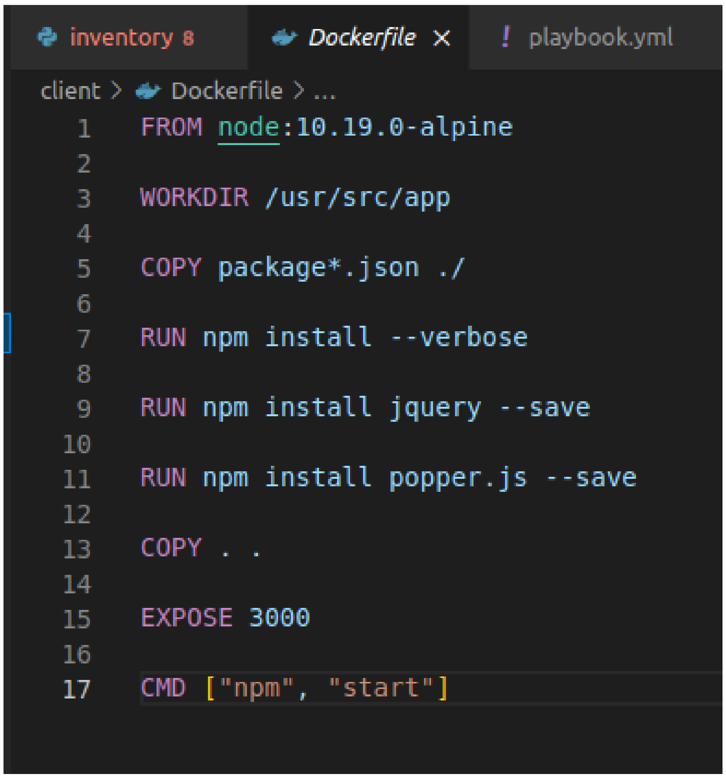
- We used python for testing our APIs
- Below is the modular code to test our project’s APIs
- We just need to pass the API in the function with the request method and it tests the API



```python
import requests as re

host = "http://43.204.112.104:8000"

def test_get_api(api_url, params, code=200):
    api_url = host + api_url
    response = re.get(api_url, params)
    assert response.status_code == code

def test_post_api(api_url, params, code = 200):
    api_url = host + api_url
    response = re.post(api_url, json=params)
    assert response.status_code == code


get_urls = ["/api/pizzas/getallpizzas","/api/pizzas/getpizzabyid"]

def test(test_apis):
    for item in test_apis:
        api, method, params = item
        if method == "post":
            test_post_api(api, params)
        else:
            test_get_api(api, params)

test_apis = [("/api/pizzas/getallpizzas", "get", {}),("/api/pizzas/getpizzabyid", "post", {"pizzaid" : "6253b02c0f62ae566cd5a9a4"})]
test(test_apis)

```

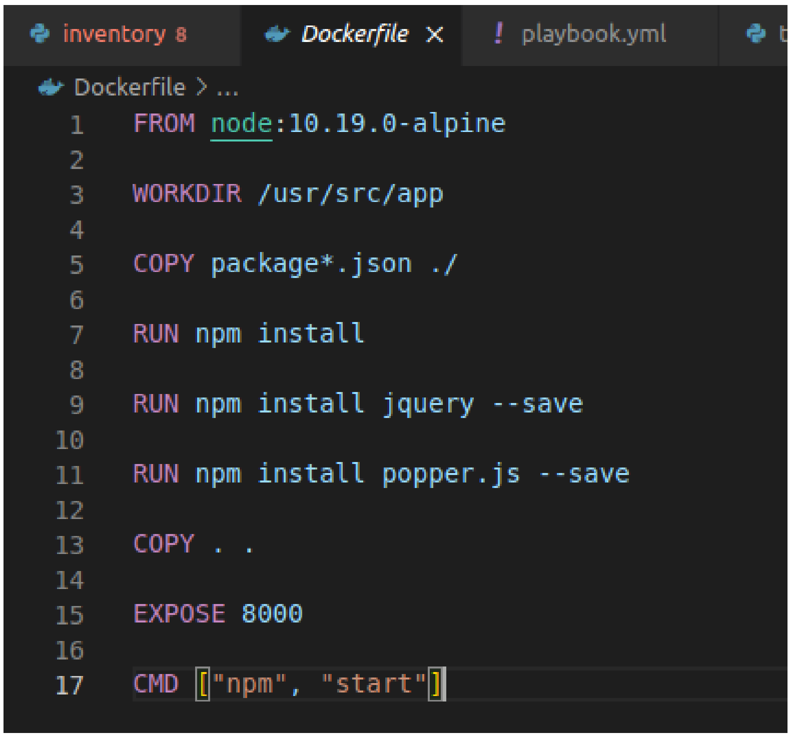
5.4 Docker

- 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**



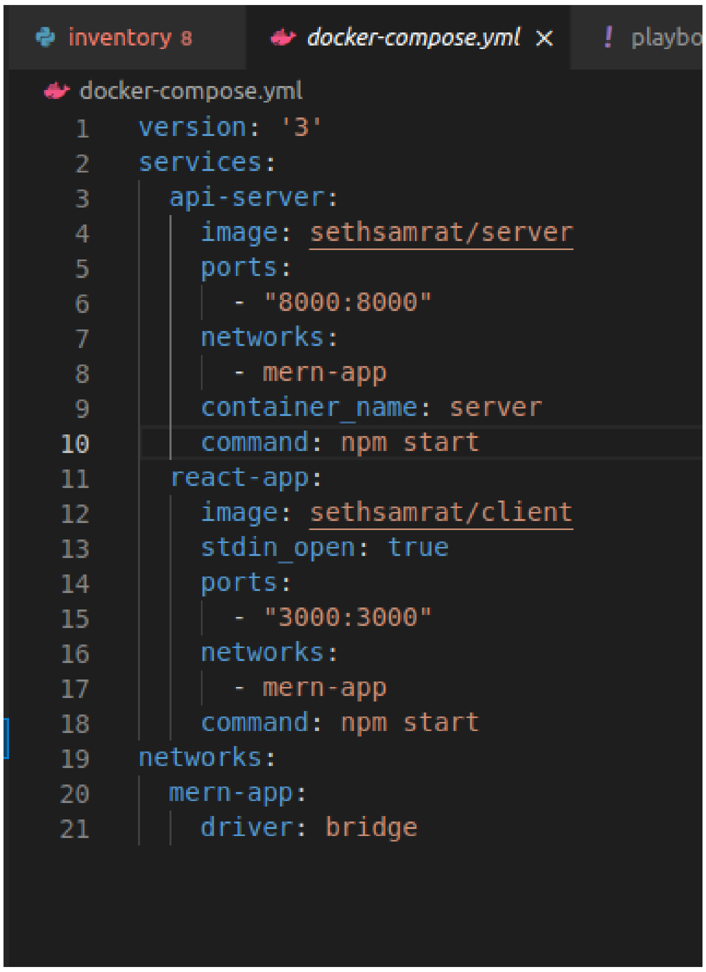
```
FROM node:10.19.0-alpine

WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install --verbose

RUN npm install jquery --save

RUN npm install popper.js --save

COPY . .

EXPOSE 3000

CMD ["npm", "start"]
```

- **Dockerfile for Server Image**



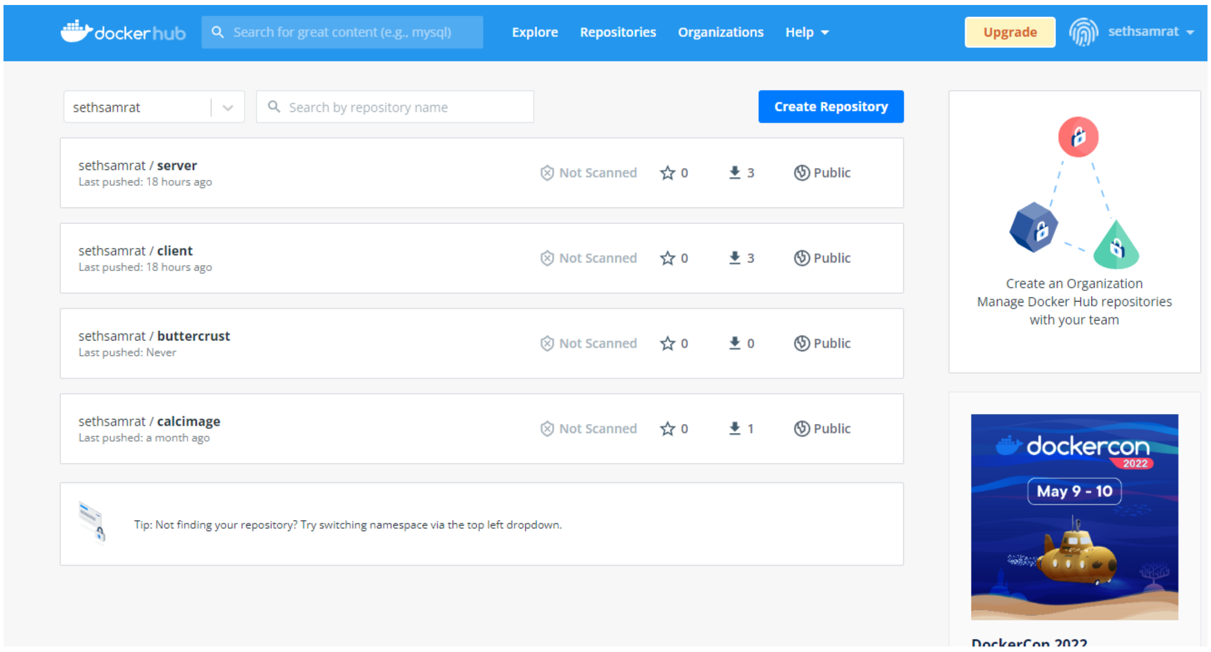
```
FROM node:10.19.0-alpine

WORKDIR /usr/src/app

COPY package*.json ./

RUN npm install

RUN npm install jquery --save

RUN npm install popper.js --save

COPY . .

EXPOSE 8000

CMD ["npm", "start"]
```

- **Docker-Compose**



```yaml
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**



sethsamrat / **server**
Last pushed: 18 hours ago — Not Scanned — ☆ 0 — ⬇ 3 — Public

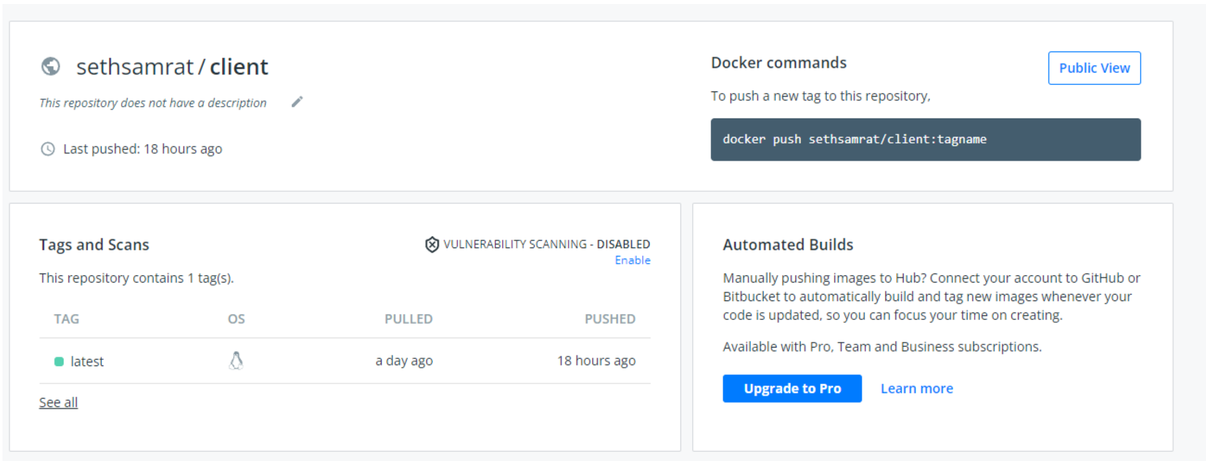
sethsamrat / **client**
Last pushed: 18 hours ago — Not Scanned — ☆ 0 — ⬇ 3 — Public

sethsamrat / **buttercrust**
Last pushed: Never — Not Scanned — ☆ 0 — ⬇ 0 — Public

sethsamrat / **calcimage**
Last pushed: a month ago — Not Scanned — ☆ 0 — ⬇ 1 — Public

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

- **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

This repository contains 1 tag(s).

TAG	OS	PULLED	PUSHED
latest		a day ago	18 hours ago

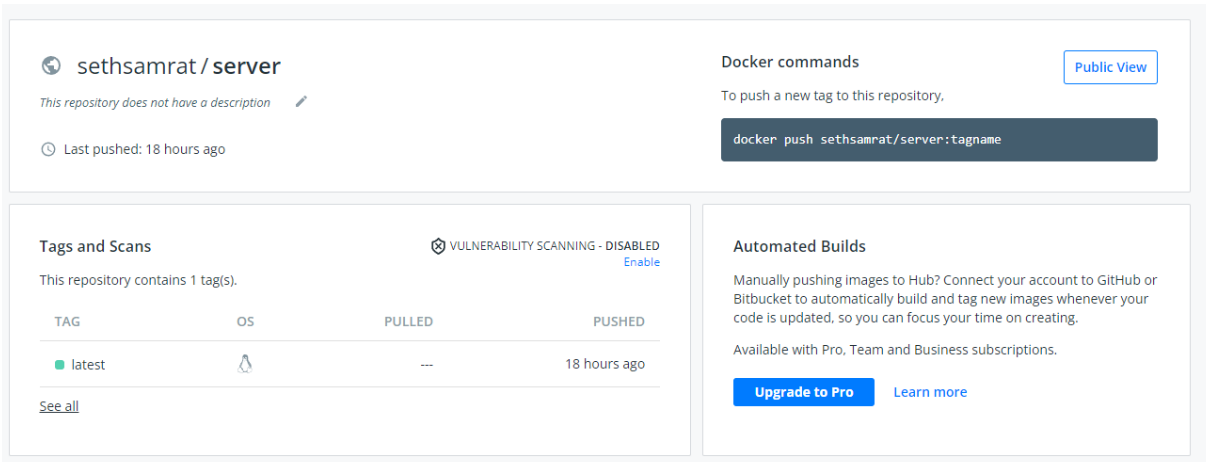
See all

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.

- **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

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.

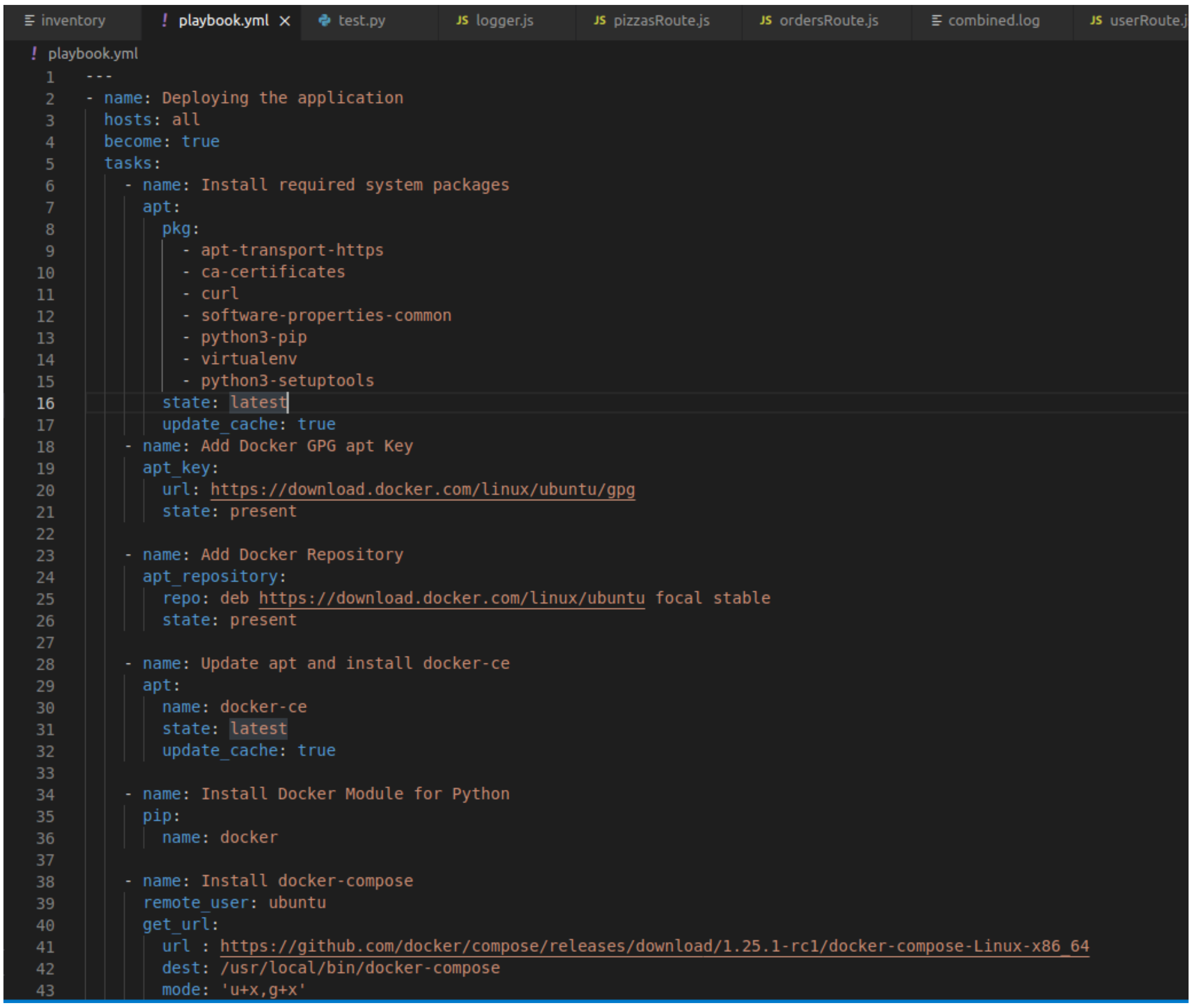
Available with Pro, Team and Business subscriptions.

5.5 Deployment using Ansible (Deployed on AWS)

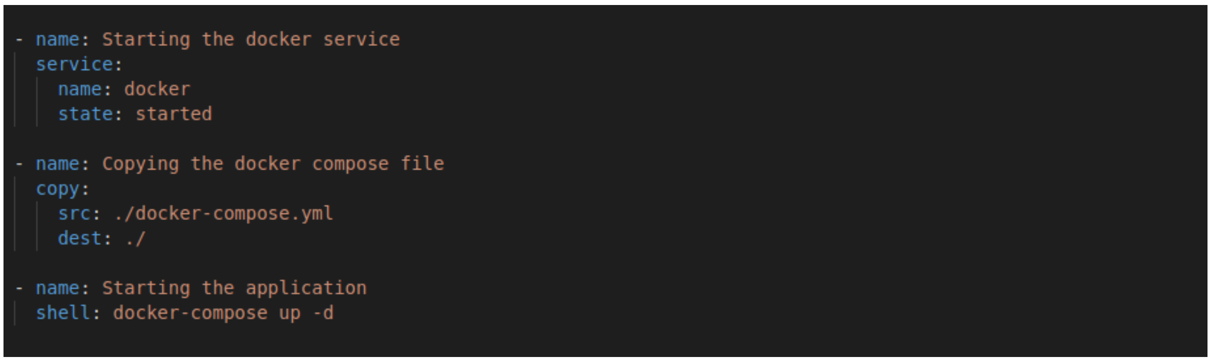
- Ansible is a radically simple IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs
- Designed for multi-tier deployments since day one, Ansible models your IT infrastructure
  by describing how 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

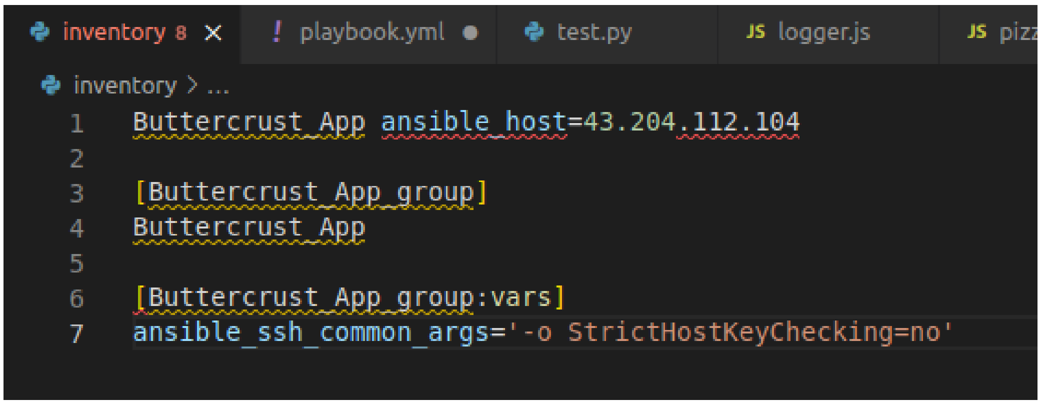
- Typically you'll work with your favorite terminal program, a text editor, and probably a version control system to keep track of changes to your content.
- Passwords are supported, but SSH keys with ssh-agent are one of the best ways to use Ansible. Though if you want to use Kerberos, that's good too.
- p**laybook.yml**



```yaml
---
- name: Deploying the application
  hosts: all
  become: true
  tasks:
    - name: Install required system packages
      apt:
        pkg:
          - apt-transport-https
          - ca-certificates
          - curl
          - software-properties-common
          - python3-pip
          - virtualenv
          - python3-setuptools
        state: latest
        update_cache: true
    - name: Add Docker GPG apt Key
      apt_key:
        url: https://download.docker.com/linux/ubuntu/gpg
        state: present

    - name: Add Docker Repository
      apt_repository:
        repo: deb https://download.docker.com/linux/ubuntu focal stable
        state: present

    - name: Update apt and install docker-ce
      apt:
        name: docker-ce
        state: latest
        update_cache: true

    - name: Install Docker Module for Python
      pip:
        name: docker

    - name: Install docker-compose
      remote_user: ubuntu
      get_url:
        url : https://github.com/docker/compose/releases/download/1.25.1-rc1/docker-compose-Linux-x86_64
        dest: /usr/local/bin/docker-compose
        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**



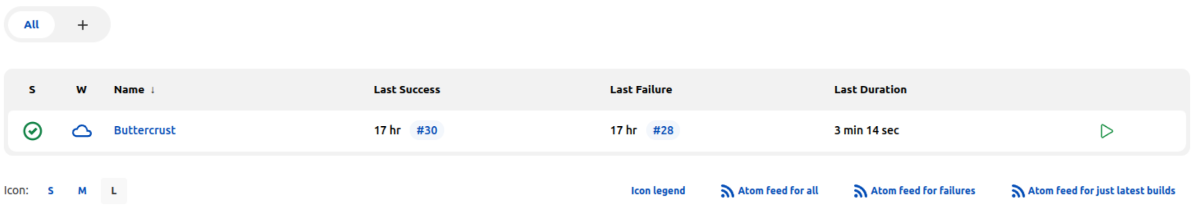
```
Buttercrust_App ansible_host=43.204.112.104

[Buttercrust_App_group]
Buttercrust_App

[Buttercrust_App_group:vars]
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

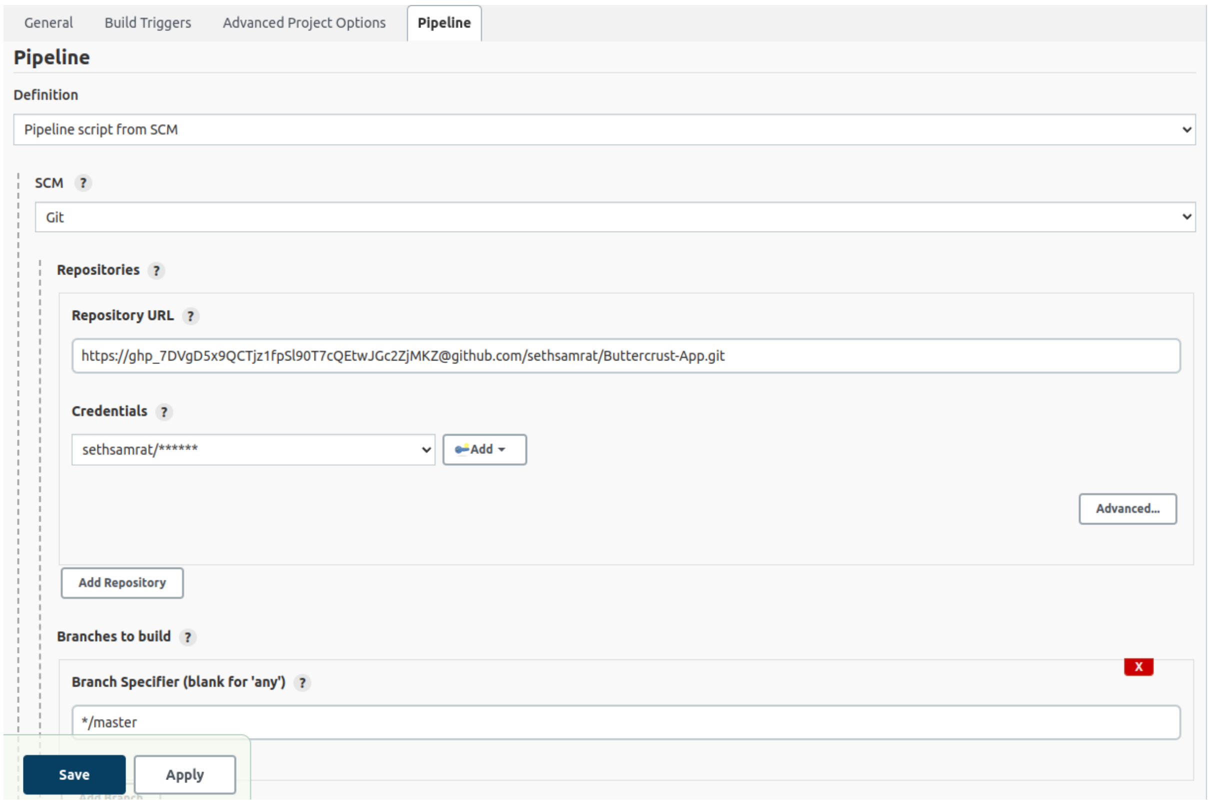
```
$ wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -
$ sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
$ sudo apt-get update
$ sudo apt-get install jenkins
```

- **Summary of builds**



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

- **Pipeline Script from Git SCM**



Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories

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

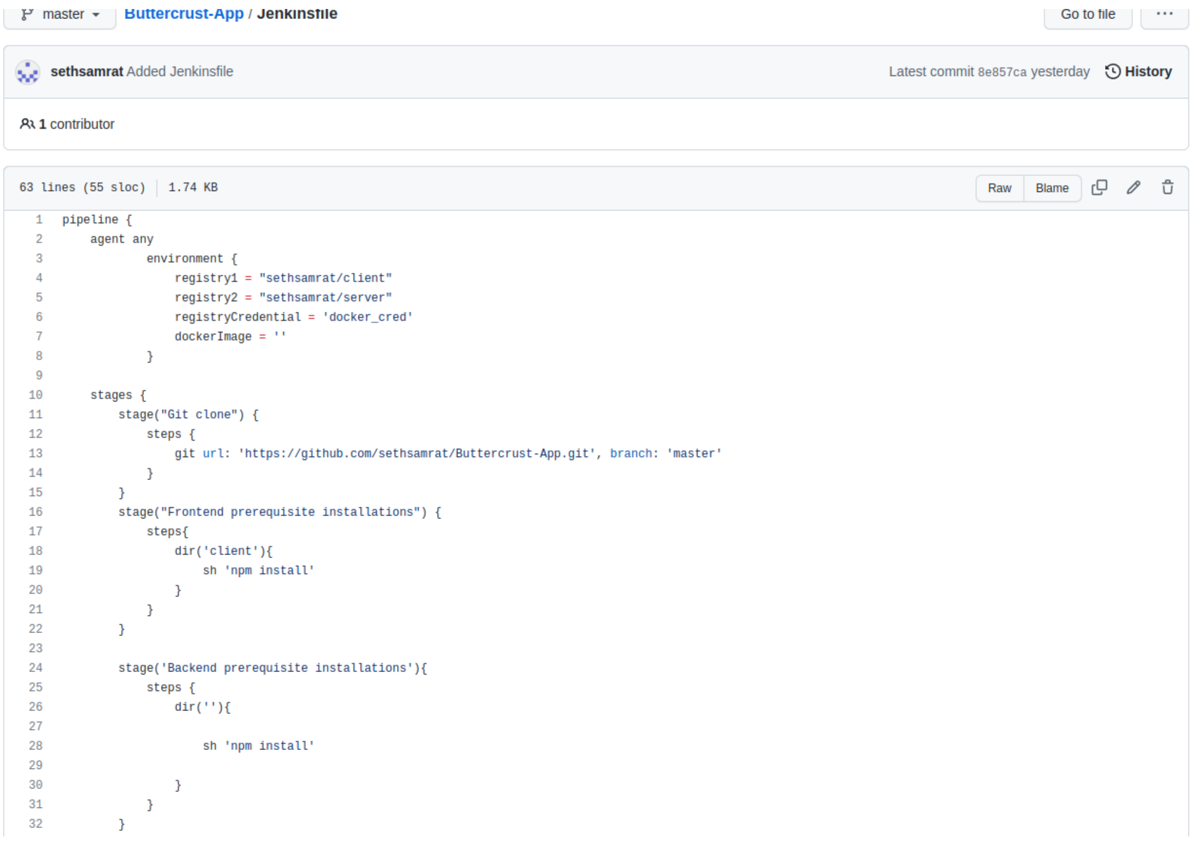
Credentials: sethsamrat/******

Branches to build

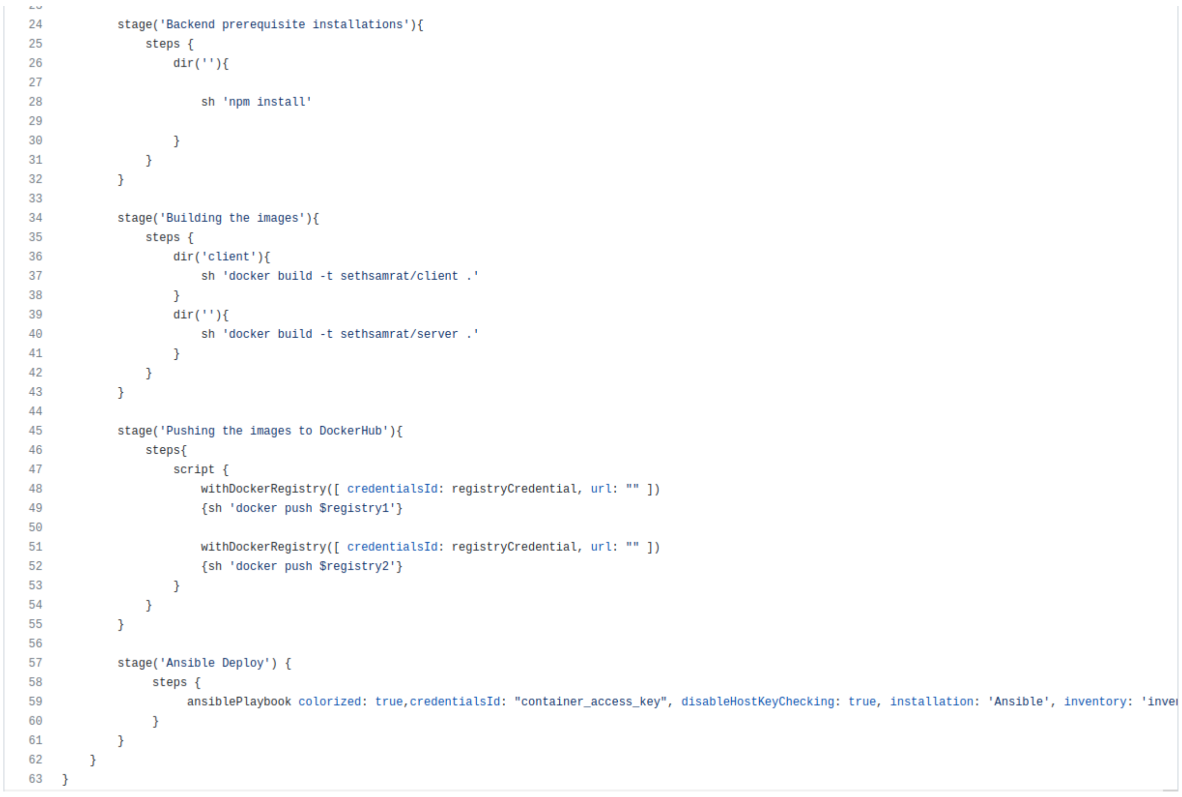
Branch Specifier (blank for 'any'): */master

- **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.

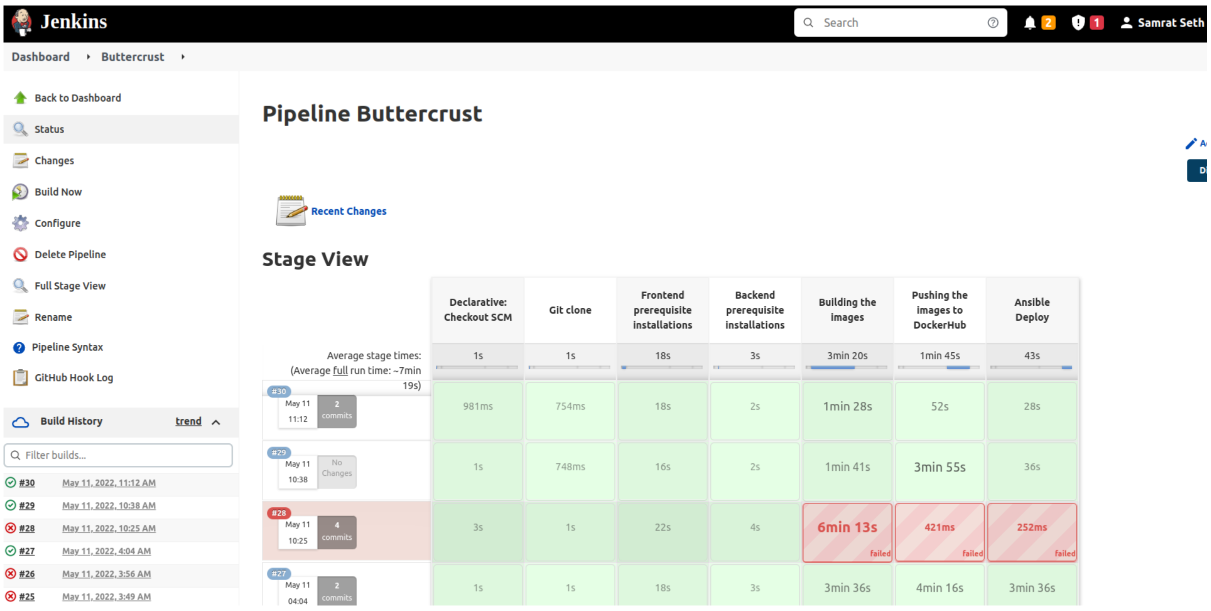

master
Buttercrust-App / Jenkinsfile
Go to file
sethsamrat Added Jenkinsfile
Latest commit 8e857ca yesterday
History
1 contributor
63 lines (55 sloc) | 1.74 KB
Raw
Blame

```
pipeline {
    agent any
            environment {
                registry1 = "sethsamrat/client"
                registry2 = "sethsamrat/server"
                registryCredential = 'docker_cred'
                dockerImage = ''
            }

    stages {
        stage("Git clone") {
            steps {
                git url: 'https://github.com/sethsamrat/Buttercrust-App.git', branch: 'master'
            }
        }
        stage("Frontend prerequisite installations") {
            steps{
                dir('client'){
                    sh 'npm install'
                }
            }
        }

        stage('Backend prerequisite installations'){
            steps {
                dir(''){

                    sh 'npm install'

                }
            }
        }
```



```
        stage('Backend prerequisite installations'){
            steps {
                dir(''){

                    sh 'npm install'

                }
            }
        }

        stage('Building the images'){
            steps {
                dir('client'){
                    sh 'docker build -t sethsamrat/client .'
                }
                dir(''){
                    sh 'docker build -t sethsamrat/server .'
                }
            }
        }

        stage('Pushing the images to DockerHub'){
            steps{
                script {
                    withDockerRegistry([ credentialsId: registryCredential, url: "" ])
                    {sh 'docker push $registry1'}

                    withDockerRegistry([ credentialsId: registryCredential, url: "" ])
                    {sh 'docker push $registry2'}
                }
            }
        }

        stage('Ansible Deploy') {
             steps {
                  ansiblePlaybook colorized: true,credentialsId: "container_access_key", disableHostKeyChecking: true, installation: 'Ansible', inventory: 'inve
             }
        }
    }
}
```

- **Stage View**



Jenkins

Dashboard › Buttercrust ›

Back to Dashboard
Status
Changes
Build Now
Configure
Delete Pipeline
Full Stage View
Rename
Pipeline Syntax
GitHub Hook Log

Build History trend

#30 May 11, 2022, 11:12 AM
#29 May 11, 2022, 10:38 AM
#28 May 11, 2022, 10:25 AM
#27 May 11, 2022, 4:04 AM
#26 May 11, 2022, 3:56 AM
#25 May 11, 2022, 3:49 AM

Pipeline Buttercrust

Recent Changes

Stage View

	Declarative: Checkout SCM	Git clone	Frontend prerequisite installations	Backend prerequisite installations	Building the images	Pushing the images to DockerHub	Ansible Deploy
Average stage times: (Average full run time: ~7min 19s)	1s	1s	18s	3s	3min 20s	1min 45s	43s
#30 May 11 11:12 2 commits	981ms	754ms	18s	2s	1min 28s	52s	28s
#29 May 11 10:38 No Changes	1s	748ms	16s	2s	1min 41s	3min 55s	36s
#28 May 11 10:25 4 commits	3s	1s	22s	4s	6min 13s failed	421ms failed	252ms failed
#27 May 11 04:04 2 commits	1s	1s	18s	3s	3min 36s	4min 16s	3min 36s

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.

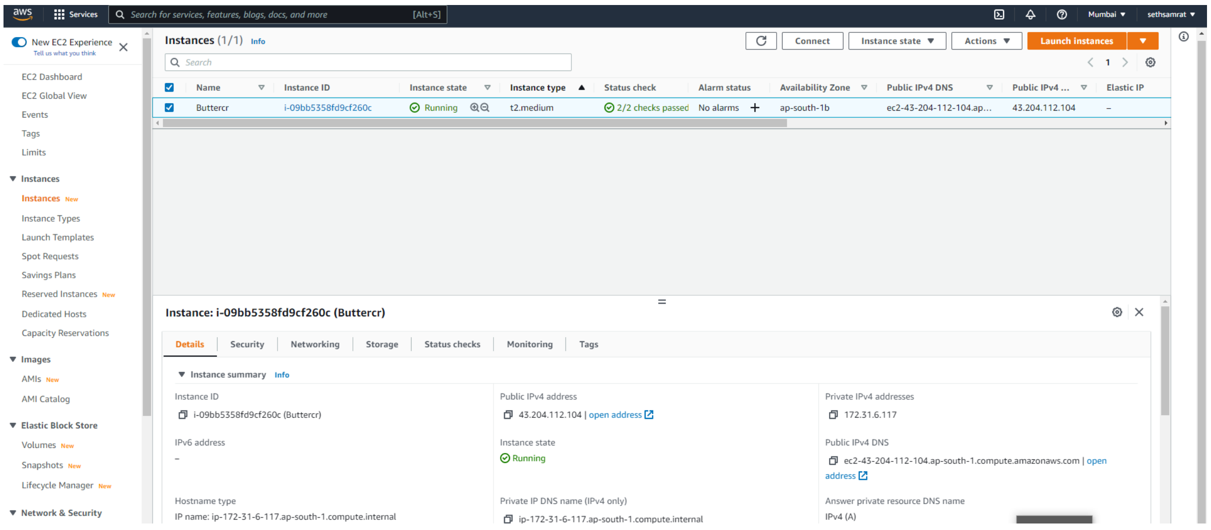
5.7.2 Features of EC2

- **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**



Instances (1/1) Info

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS	Public IPv4 ...	Elastic IP
Buttercr	i-09bb5358fd9cf260c	Running	t2.medium	2/2 checks passed	No alarms	ap-south-1b	ec2-43-204-112-104.ap...	43.204.112.104	–

Instance: i-09bb5358fd9cf260c (Buttercr)

Instance summary Info

Instance ID
i-09bb5358fd9cf260c (Buttercr)

Public IPv4 address
43.204.112.104 | open address

Private IPv4 addresses
172.31.6.117

IPv6 address
–

Instance state
Running

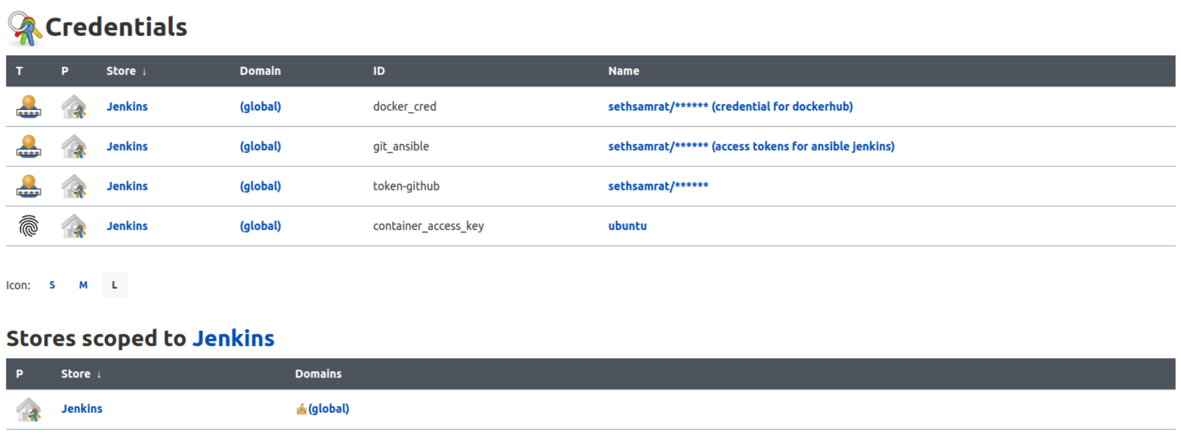
Public IPv4 DNS
ec2-43-204-112-104.ap-south-1.compute.amazonaws.com | open address

Hostname type
IP name: ip-172-31-6-117.ap-south-1.compute.internal

Private IP DNS name (IPv4 only)
ip-172-31-6-117.ap-south-1.compute.internal

Answer private resource DNS name
IPv4 (A)

- **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)

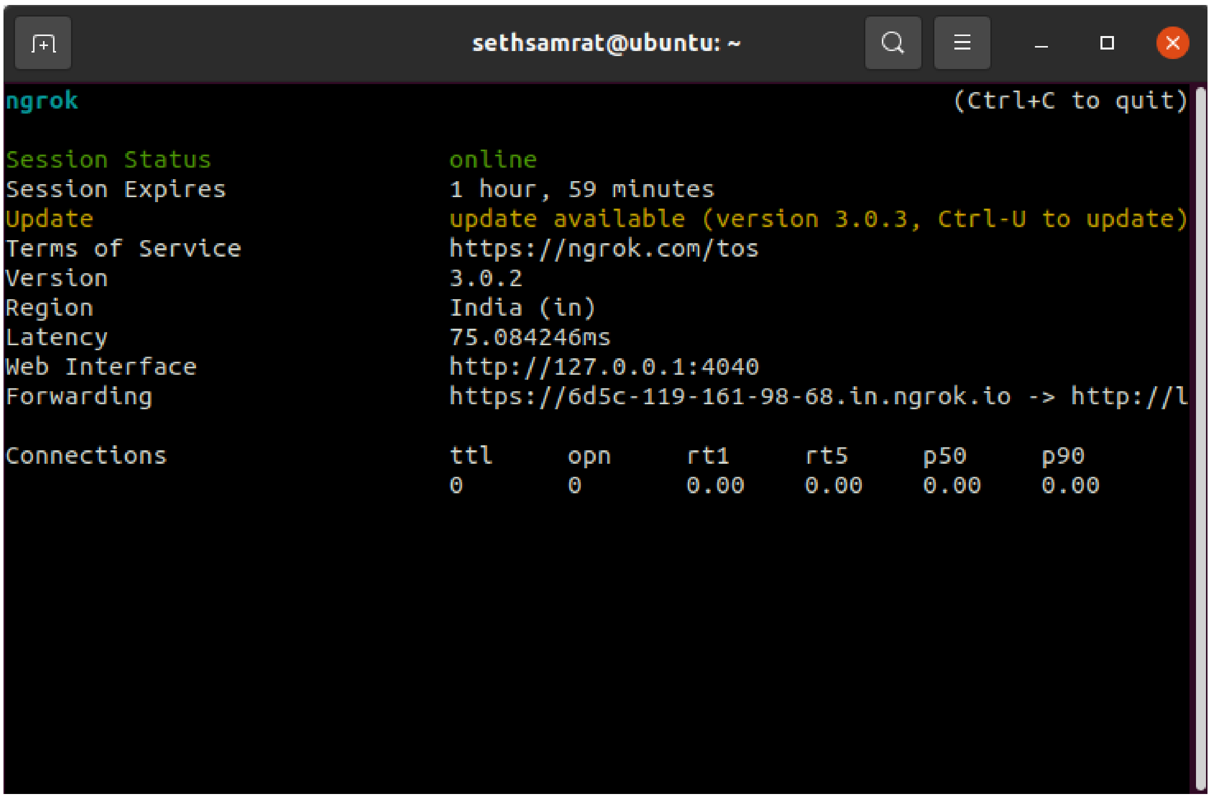
5.8 Logs and Monitoring

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

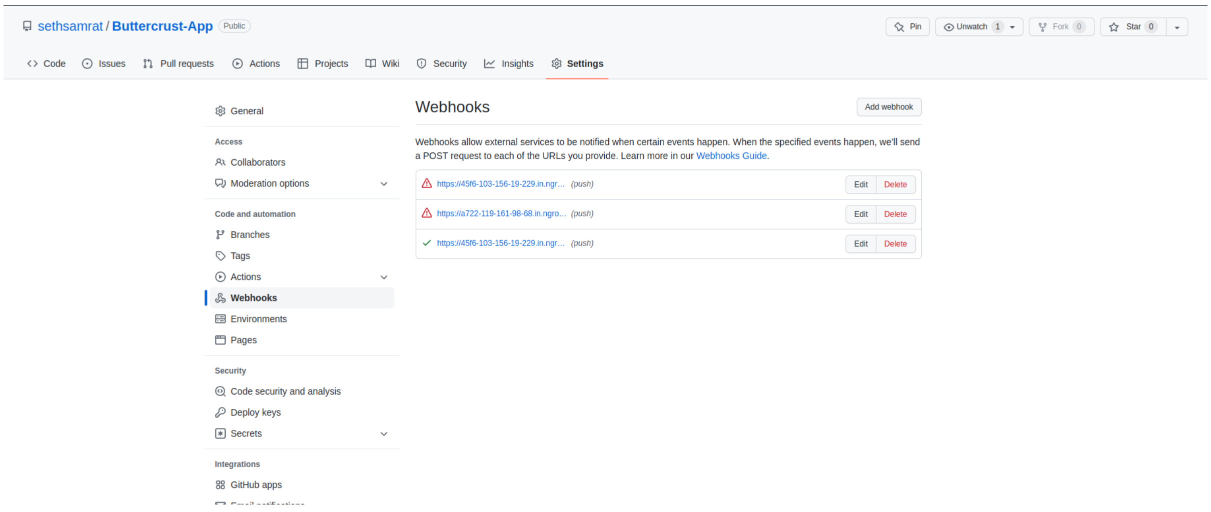
Webhooks are automated messages that are sent whenever any changes are made. In our case when we make any changes to the GitHub repo, the webhook will automatically start the Jenkins pipeline.
Ngrok exposes local servers behind NATs (Network Address Translation) and firewalls to the public internet over 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**



```
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**

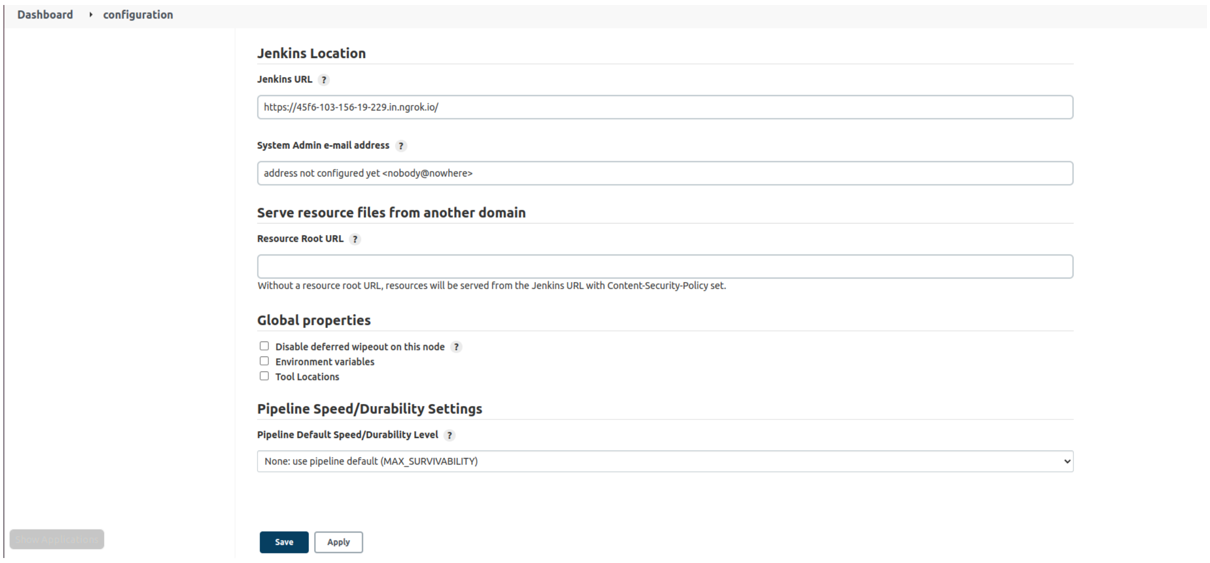

sethsamrat / Buttercrust-App Public

Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

- https://45f6-103-156-19-229.in.ngr… (push)
- https://a722-119-161-98-68.in.ngro… (push)
- https://45f6-103-156-19-229.in.ngr… (push)

- **Adding webhooks to Jenkins Location**


Dashboard › configuration

Jenkins Location

Jenkins URL: https://45f6-103-156-19-229.in.ngrok.io/

System Admin e-mail address: address not configured yet <nobody@nowhere>

Serve resource files from another domain

Resource Root URL:

Without a resource root URL, resources will be served from the Jenkins URL with Content-Security-Policy set.

Global properties

- [ ] Disable deferred wipeout on this node
- [ ] Environment variables
- [ ] Tool Locations

Pipeline Speed/Durability Settings

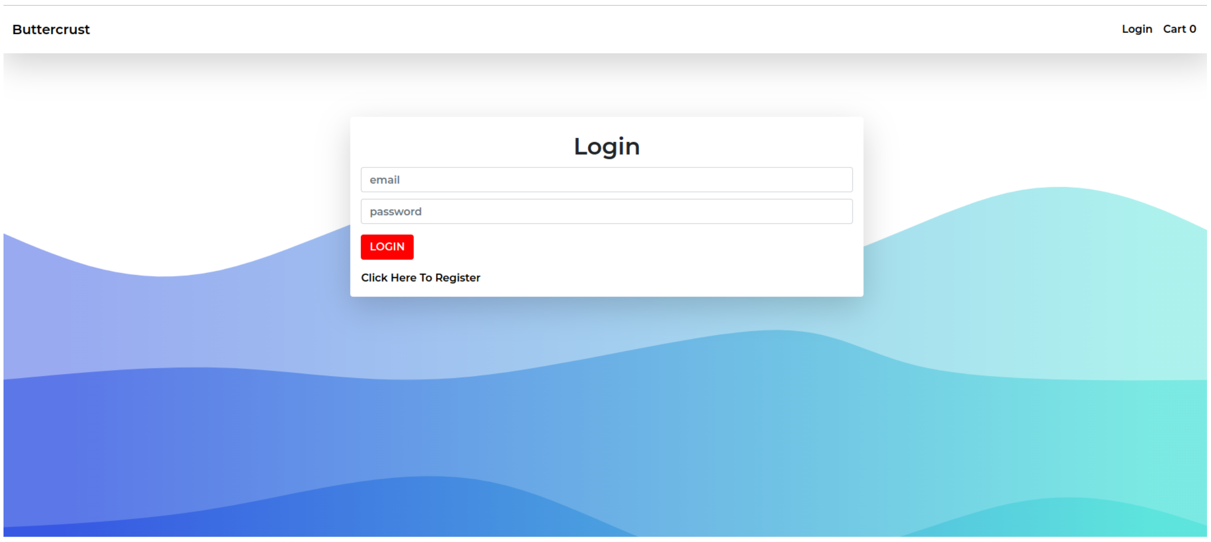
Pipeline Default Speed/Durability Level: None: use pipeline default (MAX_SURVIVABILITY)

6. Features and API

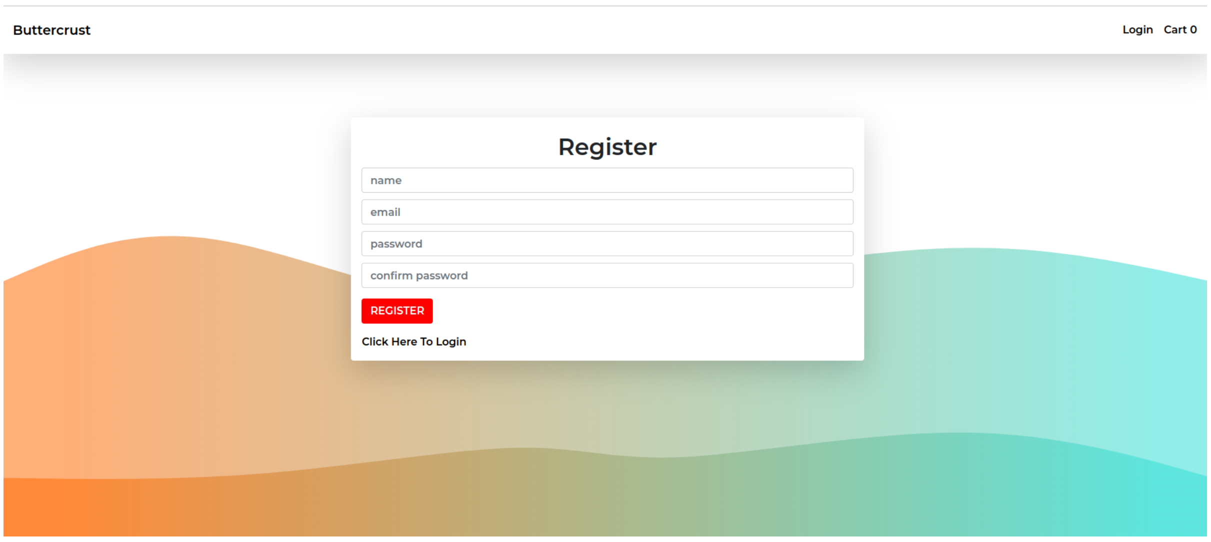
6.1 Features

This application has the following functionalities:

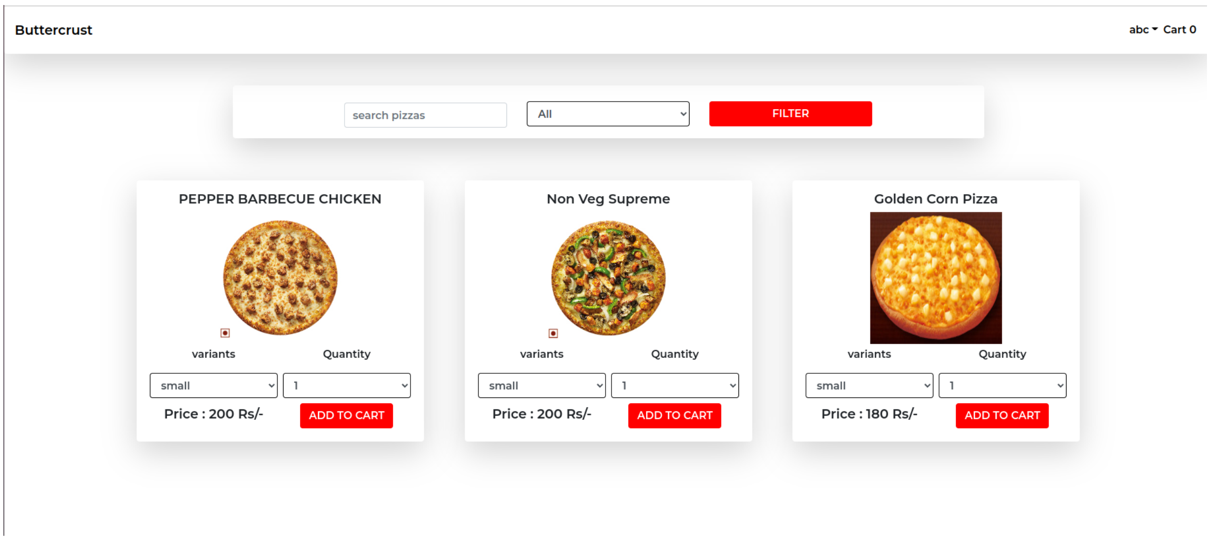
- **Login Page**



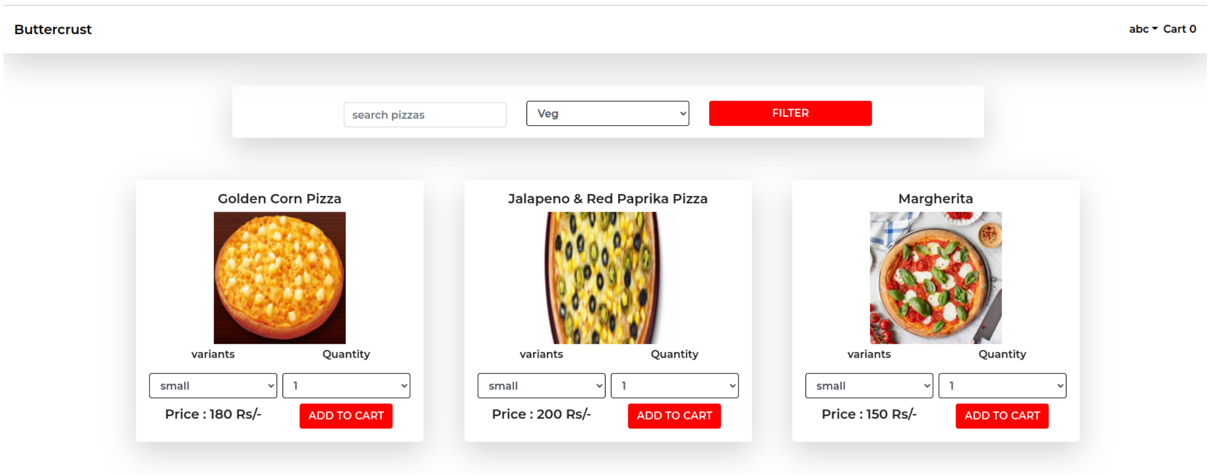
- **Update quantity in cart**



- **User Dashboard**



- ## Pizza Filter



Buttercrust

abc ▾ Cart 0

search pizzas

Veg

FILTER

Golden Corn Pizza

variants small Quantity 1

Price : 180 Rs/-

ADD TO CART

Jalapeno & Red Paprika Pizza

variants small Quantity 1

Price : 200 Rs/-

ADD TO CART

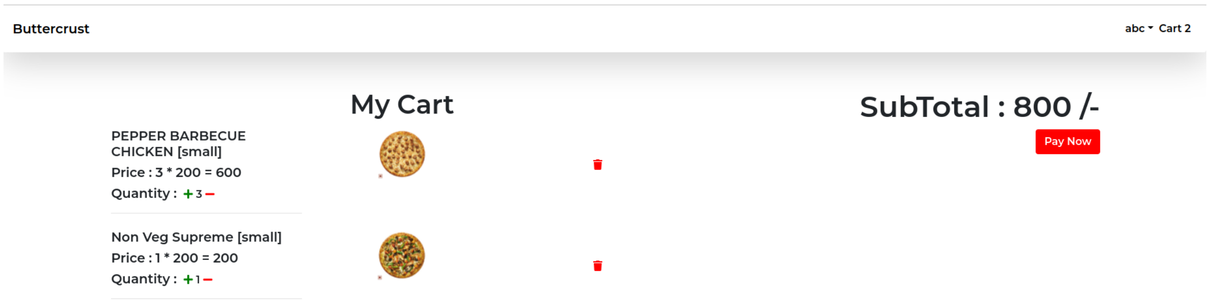
Margherita

variants small Quantity 1

Price : 150 Rs/-

ADD TO CART

- ## Cart Page



Buttercrust

abc ▾ Cart 2

My Cart

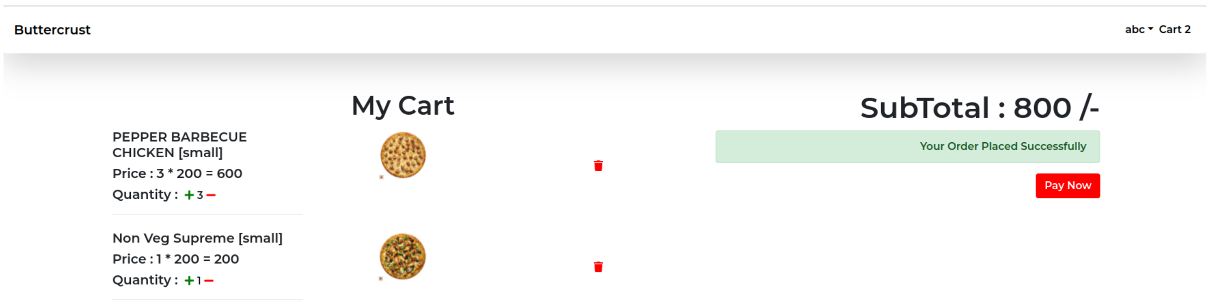
SubTotal : 800 /-

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

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

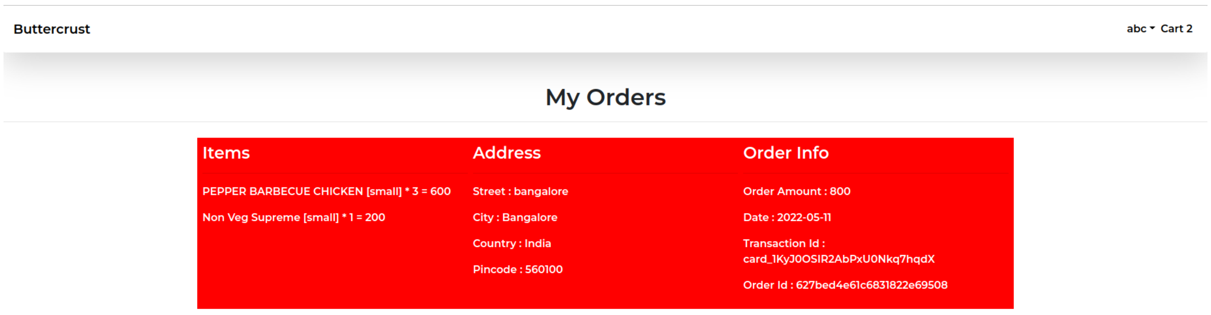
Non Veg Supreme [small]
Price : 1 * 200 = 200
Quantity : + 1 –

SubTotal : 800 /-

Your Order Placed Successfully

Pay Now

- **Orders Page**



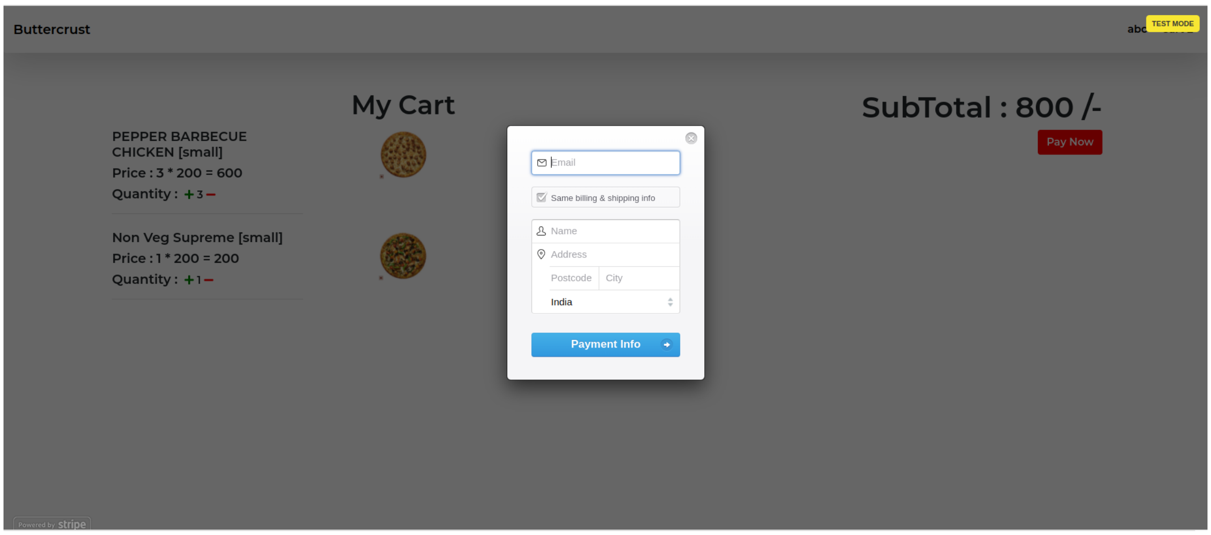
Buttercrust

abc Cart 2

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_1KyJ0OSIR2AbPxU0Nkq7hqdX
	Pincode : 560100	Order Id : 627bed4e61c6831822e69508

- **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

Email

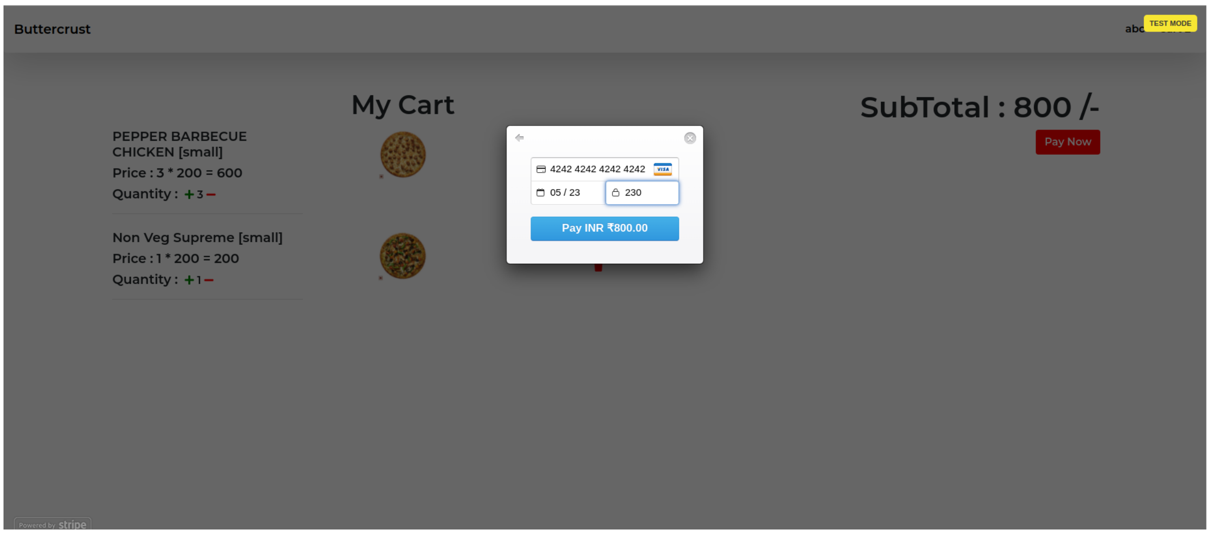
Same billing & shipping info

Name
Address
Postcode City
India

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 −

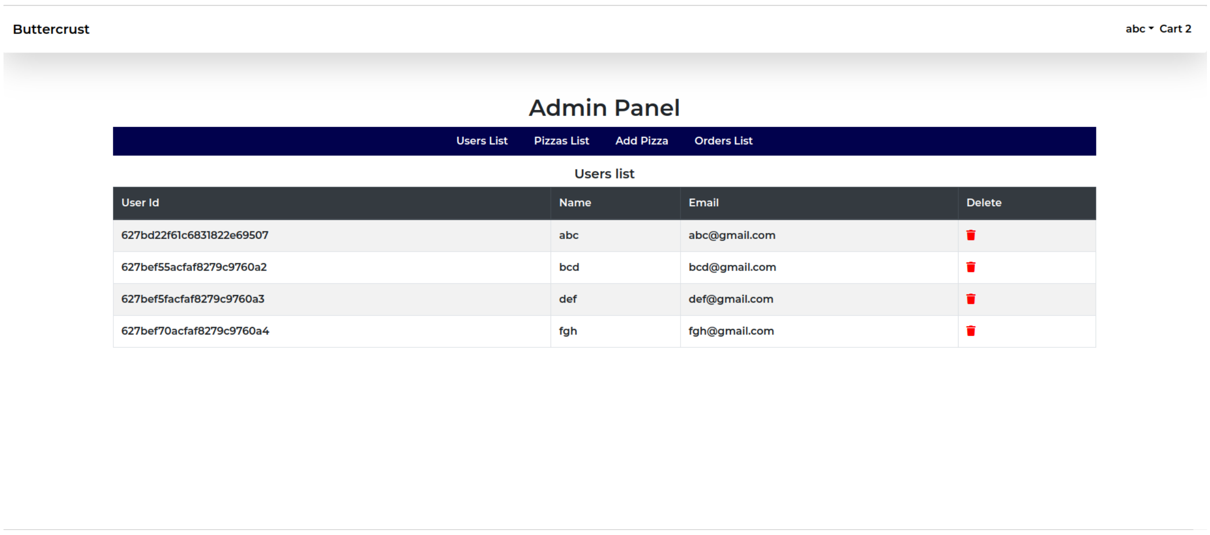
SubTotal : 800 /-

Pay Now

4242 4242 4242 4242
05 / 23
230
Pay INR ₹800.00

Powered by stripe

- **Admin Panel - Users List**


Buttercrust

abc Cart 2

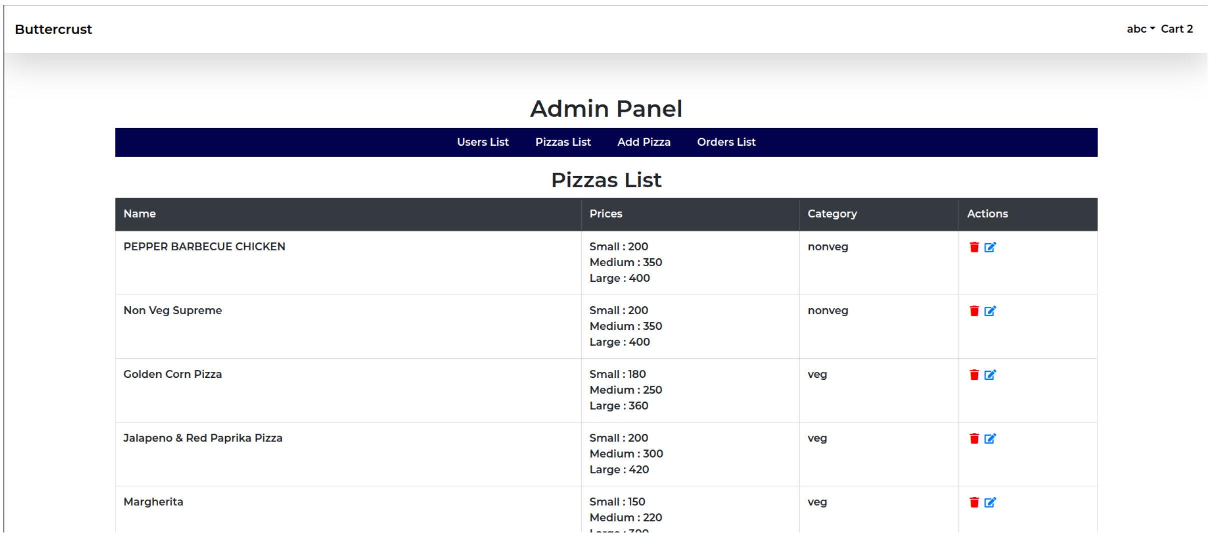
Admin Panel

Users List Pizzas List Add Pizza Orders List

Users list

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**


Buttercrust

abc Cart 2

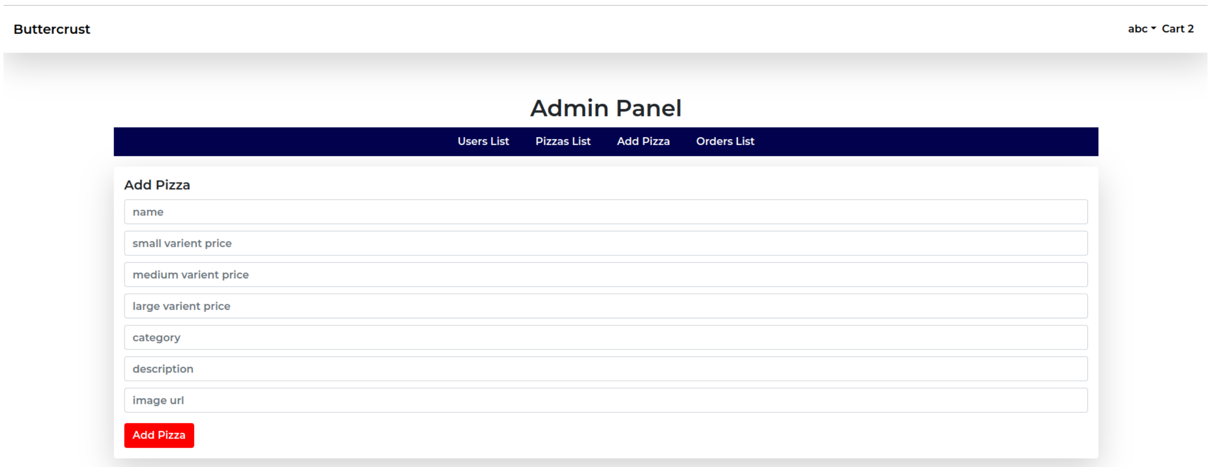
Admin Panel

Users List Pizzas List Add Pizza Orders List

Pizzas List

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	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

name

small varient price

medium varient price

large varient price

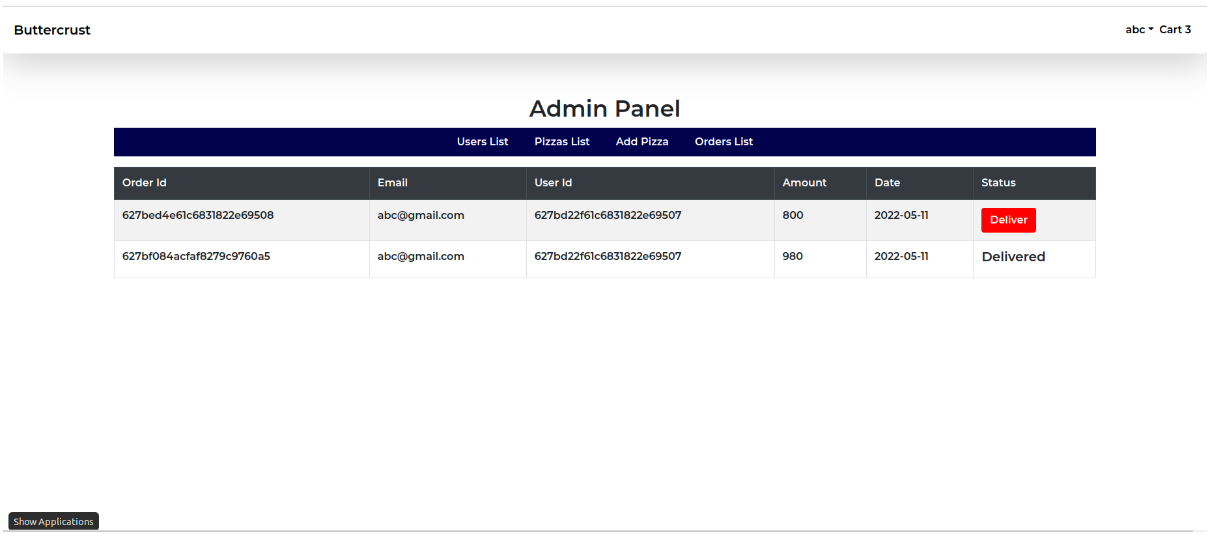
category

description

image url

Add Pizza

- **Admin Panel - Orders List**



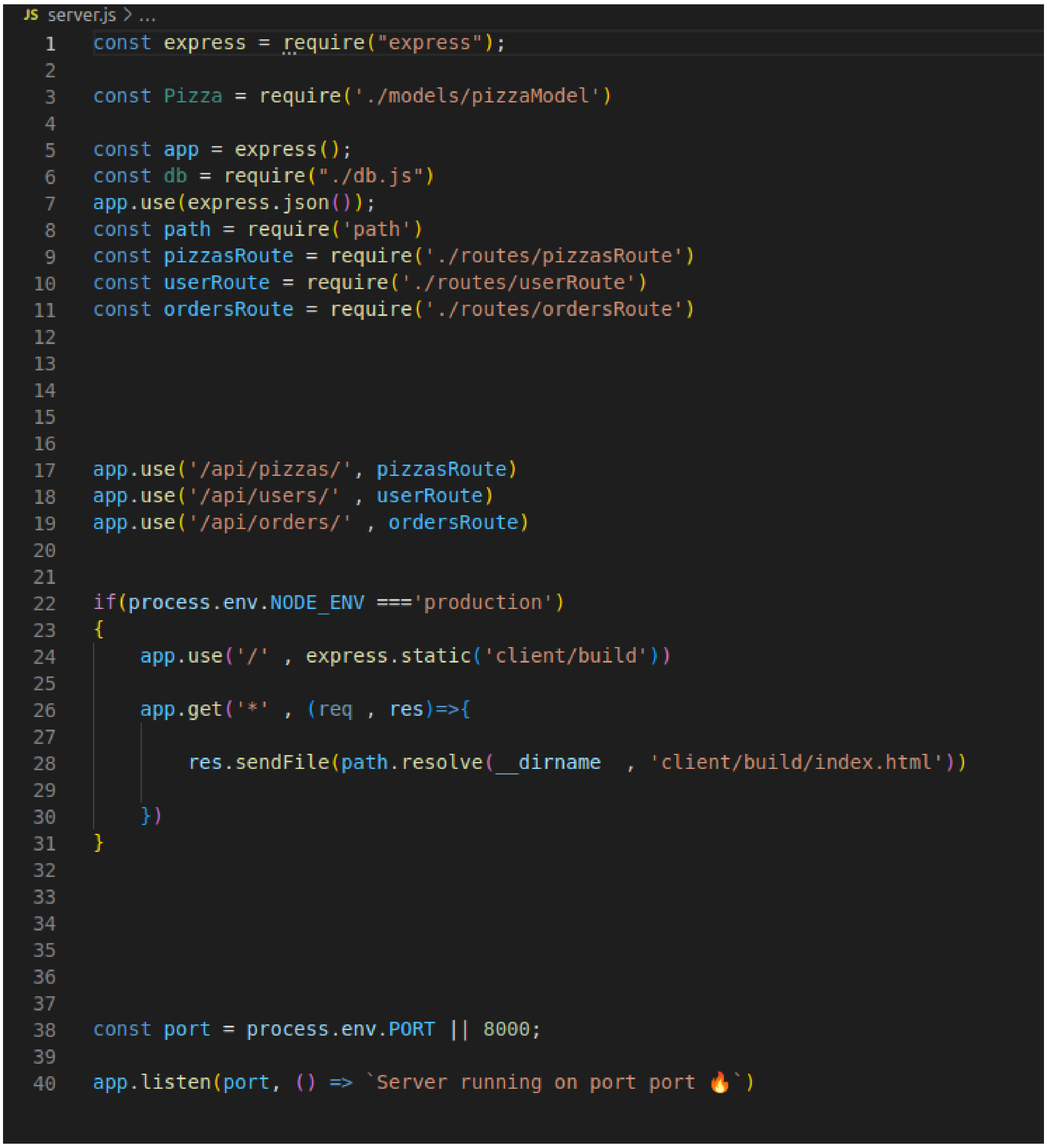
Order Id	Email	User Id	Amount	Date	Status
627bed4e61c6831822e69508	abc@gmail.com	627bd22f61c6831822e69507	800	2022-05-11	Deliver
627bf084acfaf8279c9760a5	abc@gmail.com	627bd22f61c6831822e69507	980	2022-05-11	Delivered

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

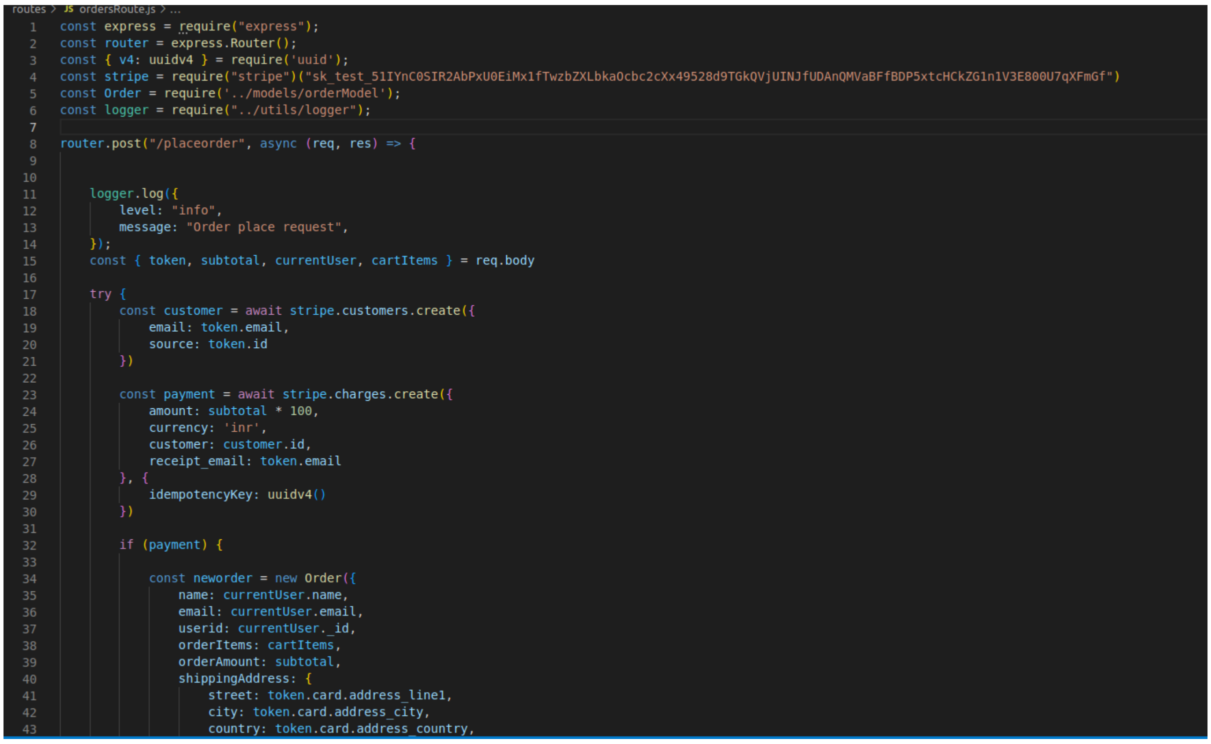
- **Server.js**



server.js > ...

```
const express = require("express");

const Pizza = require('./models/pizzaModel')

const app = express();
const db = require("./db.js")
app.use(express.json());
const path = require('path')
const pizzasRoute = require('./routes/pizzasRoute')
const userRoute = require('./routes/userRoute')
const ordersRoute = require('./routes/ordersRoute')





app.use('/api/pizzas/', pizzasRoute)
app.use('/api/users/' , userRoute)
app.use('/api/orders/' , ordersRoute)


if(process.env.NODE_ENV ==='production')
{
    app.use('/' , express.static('client/build'))

    app.get('*' , (req , res)=>{

        res.sendFile(path.resolve(__dirname  , 'client/build/index.html'))

    })
}






const port = process.env.PORT || 8000;

app.listen(port, () => `Server running on port port 🔥`)
```

- **ordersRoute.js**


routes > JS ordersRoute.js > ...

```
const express = require("express");
const router = express.Router();
const { v4: uuidv4 } = require('uuid');
const stripe = require("stripe")("sk_test_51IYnC0SIR2AbPxU0EiMx1fTwzbZXLbkaOcbc2cXx49528d9TGkQVjUINJfUDAnQMVaBFfBDP5xtcHCkZG1n1V3E800U7qXFmGf")
const Order = require('../models/orderModel');
const logger = require("../utils/logger");

router.post("/placeorder", async (req, res) => {


    logger.log({
        level: "info",
        message: "Order place request",
    });
    const { token, subtotal, currentUser, cartItems } = req.body

    try {
        const customer = await stripe.customers.create({
            email: token.email,
            source: token.id
        })

        const payment = await stripe.charges.create({
            amount: subtotal * 100,
            currency: 'inr',
            customer: customer.id,
            receipt_email: token.email
        }, {
            idempotencyKey: uuidv4()
        })

        if (payment) {

            const neworder = new Order({
                name: currentUser.name,
                email: currentUser.email,
                userid: currentUser._id,
                orderItems: cartItems,
                orderAmount: subtotal,
                shippingAddress: {
                    street: token.card.address_line1,
                    city: token.card.address_city,
                    country: token.card.address_country,
```

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

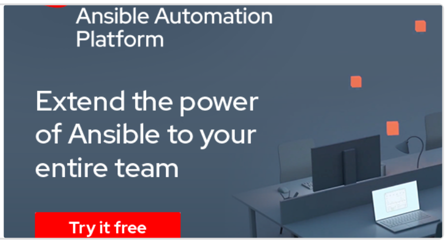
https://reactjs.org/docs/getting-started.html

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


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/

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