A
Summer Internship Report
On
Web Design with React JS

(CE346 – Summer Internship - I)

Prepared by

MEET CHHATRALA (22CE019)

Under the Supervision of

Dr. DHAVAL BHOI

Submitted to

Charotar University of Science & Technology (CHARUSAT)
for the Partial Fulfillment of the Requirements for the
Degree of Bachelor of Technology (B.Tech.)
for Semester 5

Submitted at

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY
Accredited with Grade A by NAAC
Accredited with Grade A by KCG

U & P U. PATEL DEPARTMENT OF COMPUTER ENGINEERING
Chandubhai S. Patel Institute of Technology (CSPIT)
Faculty of Technology & Engineering (FTE), CHARUSAT
At: Changa, Dist: Anand, Pin: 388421.

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

Accredited with Grade A by NAAC
Accredited with Grade A by KCG

CERTIFICATE

This is to certify that the report entitled "**DevOps Engineer**" is a bonafied work carried out by **Meet Chhatrala(22CE019)** under the guidance and supervision of **Mr. Keyur Trivedi** for the subject **Summer Internship – I (CE346)** of 5th Semester of Bachelor of Technology in **Computer Engineering** at Chandubhai S. Patel Institute of Technology (CSPIT), Faculty of Technology & Engineering (FTE) – CHARUSAT, Gujarat.

To the best of my knowledge and belief, this work embodies the work of candidate himself, has duly been completed, and fulfills the requirement of the ordinance relating to the B.Tech. Degree of the University and is up to the standard in respect of content, presentation and language for being referred by the examiner(s).

Under the supervision of, Dr. DHAVAL BHOI U & P U. Patel Dept. of Computer Engineering CSPIT, FTE, CHARUSAT, Changa, Gujarat	Mr. Keyur Trivedi Computer Department Codderzone Solution

Dr. Nikita Bhatt
Head - U & P U. Patel Department of Computer Engineering,
CSPIT, FTE, CHARUSAT, Changa, Gujarat.

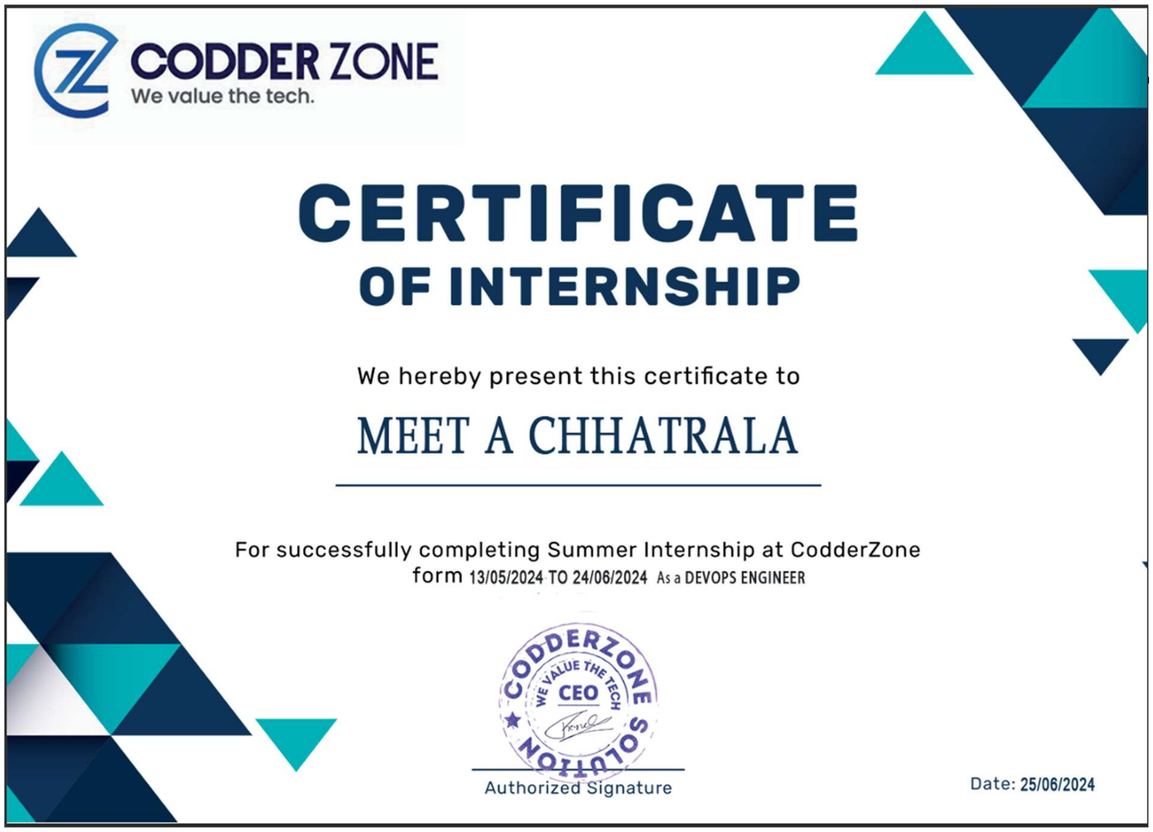
Chandubhai S. Patel Institute of Technology (CSPIT)

Faculty of Technology & Engineering (FTE), CHARUSAT

At: Changa, Ta. Petlad, Dist. Anand, Pin: 388421. Gujarat.

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

Accredited with Grade A by NAAC
Accredited with Grade A by KCG



CODDER ZONE
We value the tech.

CERTIFICATE
OF INTERNSHIP

We hereby present this certificate to

MEET A CHHATRALA

For successfully completing Summer Internship at CodderZone
form 13/05/2024 TO 24/06/2024 As a DEVOPS ENGINEER

CODDERZONE SOLUTION
WE VALUE THE TECH
CEO

[illegible]

Authorized Signature

Date: 25/06/2024

Abstract

This report provides an overview of my DevOps training, focusing on Docker, Kubernetes, and Python with Django. Throughout this training, I gained crucial insights into containerization, orchestration, and modern application deployment practices. My learning journey began with Docker, where I mastered the creation and management of containerized applications, enabling consistent and efficient software environments. I then explored Kubernetes, gaining expertise in automating deployment, scaling, and managing containerized applications across clusters. Additionally, I delved into Python with Django, learning to develop robust and scalable web applications. This comprehensive training has equipped me with a solid foundation in DevOps principles and tools, enhancing my ability to deliver reliable and scalable software solutions.

Acknowledgement

I hereby acknowledge that, I have successfully completed my summer internship on Devops at Codderzone Solution for 30 days. During my internship I had learned many new things for this Devops Tools , also I got the opportunity to enhance my skills in Devops and Cloud Computing.

I am also thankful to Mr. Keyur Trivedi, my supervisor, whose guidance, support, and insightful feedback were invaluable throughout the internship. His expertise and mentorship have significantly contributed to my professional development.

I am grateful to the entire team at Codderzone Solution for their encouragement and cooperation during this internship journey. Their collaborative spirit and shared knowledge have enhanced my understanding and skills.

I am deeply grateful to Dr. Sneha Padhiar for suggesting this internship opportunity.

Description of company

CodderZone is a leading internet products company that builds innovative software, web applications and mobile apps that are used by millions of end-users worldwide through our products and custom software development services for over 1,000+ clients across 100+ countries.
Top web & mobile app development company offering innovative solutions for diverse industry verticals. We act as an invaluable catalyst for building a culture of innovation. Our breakthrough web, mobile and software solutions have the capability to challenge the limits and give business a competitive edge.

Table of Contents

List of Figures

List of Tables

CHAPTER 1: INTRODUCTION

1.1 INTERNSHIP OBJECTIVES

Objectives of an Internship in DevOps with Docker, Kubernetes, AWS, and Python with Django.

- Gain hands-on experience and practical skills in DevOps practices and tools, including Docker, Kubernetes, AWS, and Python with Django.
- Develop expertise in containerization using Docker, including creating, managing, and deploying containerized applications.
- Learn to automate deployment, scaling, and management of containerized applications across clusters using Kubernetes.
- Contribute to real-world projects or prototypes that apply DevOps principles to enhance the deployment pipeline and application performance.
- Improve problem-solving skills by debugging and troubleshooting issues in containerized environments, orchestration systems, cloud infrastructure, and web applications.

1.2 OVERVIEW OF INTERNSHIP ACTIVITIES

	Date	Day	Name of Topic/Module
Week 1	15/05/2024	Wednesday	Introduction to Docker
	16/05/2024	Thursday	Introduction to Docker and Installation and Setup
	17/05/2024	Friday	Docker CLI
	20/05/2024	Monday	Creating Docker Images frontend
	21/05/2024	Tuesday	Creating Docker Images for backend
	22/05/2024	Wednesday	Docker Compose
Week 2	23/05/2024	Thursday	Container Management
	24/05/2024	Friday	Volumes and Networking
	27/05/2024	Monday	Dockerizing a full-stack website for Development
	28/05/2024	Tuesday	Dockerizing a full-stack website for Development
	29/05/2024	Wednesday	Dockerizing a full-stack website for Deployment
Week 3	30/05/2024	Thursday	Dockerizing a full-stack website for Deployment
	31/05/2024	Friday	Introduction to AWS and Account Setup
	03/06/2024	Monday	IAM Roles and Policies
	04/06/2024	Tuesday	EC2 Instances
	05/06/2024	Wednesday	S3 Storage
Week 4	06/06/2024	Thursday	Deploying Docker images in EC2 instance
	07/06/2024	Friday	Introduction to Kubernetes
	10/06/2024	Monday	Introduction to Kubernetes
	11/06/2024	Tuesday	Introduction to Django and Environment Setup and Creating a Django Project
	12/06/2024	Wednesday	Django Models
Week 5	13/06/2024	Thursday	Django Admin
	14/06/2024	Friday	Views and Templates
	17/06/2024	Monday	Django crud operation
	18/06/2024	Tuesday	User Authentication
	19/06/2024	Wednesday	Signup and login page using django

chapter 2 tools and technologies

2.1 INTRODUCTION TO DOCKER

Docker is a powerful platform designed for developing, shipping, and running applications inside containers. Containers allow developers to package an application with all its dependencies and run it consistently across different environments. Docker uses a system of images and containers, where an image is a lightweight, standalone, executable package that includes everything needed to run a piece of software, including the code, runtime, libraries, and system tools. Containers, in turn, are instances of these images running in isolated environments.

A typical Docker workflow starts with writing a Dockerfile, a text file containing a series of instructions on how to build a Docker image. This file specifies the base image, application dependencies, and commands to run. Once the Dockerfile is created, developers use the docker build command to create the image and the docker run command to start the container.

Docker is essential for modern software development and deployment. It simplifies the process of creating, testing, and deploying applications by providing a consistent environment across development, testing, and production stages. This consistency reduces bugs and deployment issues, making development more efficient.

During a technology training internship, learning Docker is crucial. Trainees acquire skills to containerize applications, manage containerized environments, and deploy containers efficiently. Mastery of Docker enables developers to build scalable, portable, and reliable software solutions. Understanding Docker is the first step toward leveraging container orchestration tools like Kubernetes and advanced cloud services, making it an indispensable part of the training. This foundational knowledge is critical for any aspiring DevOps engineer or software developer and serves as a stepping stone to more advanced topics in containerization and cloud computing.

2.2 INTRODUCTION TO CLOUD COMPUTING

Cloud computing is the delivery of computing services over the internet, providing faster innovation, flexible resources, and economies of scale. It allows businesses to access and pay for computing resources such as servers, storage, databases, and networking on an as-needed basis instead of maintaining physical data centers. Cloud computing enables organizations to scale resources up or down dynamically, ensuring optimal performance and cost-efficiency.

Key benefits of cloud computing include:

- **Cost Efficiency**: Reduces capital expenditure on hardware and software.
- **Scalability**: Easily scale resources to meet demand.
- **Flexibility**: Access resources from anywhere, anytime.
- **Innovation**: Quickly deploy and experiment with new technologies.

2.2.1 Introduction To AWS

Amazon Web Services (AWS) is a comprehensive and widely adopted cloud platform offering a vast array of services, including computing power, storage, and databases, as well as advanced technologies such as machine learning, IoT, and serverless computing. AWS helps businesses innovate faster, reduce costs, and scale applications seamlessly.

Key services of AWS include:

- **Compute**: EC2, Lambda
- **Storage**: S3, EBS
- **Database**: RDS, DynamoDB
- **Networking**: VPC, CloudFront

EC2 Service (Elastic Compute Cloud)

Amazon EC2 provides scalable computing capacity in the cloud, allowing developers to launch virtual servers, known as instances, on-demand. EC2 supports various instance types tailored to different use cases, including general-purpose, compute-optimized, and memory-optimized instances.

Key features of EC2 include:

- **Elasticity**: Easily scale instances up or down.
- **Variety**: Choose from a wide range of instance types.
- **Security**: Integrated security features like VPC and IAM.
- **Cost Management**: Pay only for the compute time used.

S3 Bucket (Simple Storage Service)

Amazon S3 is a scalable object storage service used for storing and retrieving any amount of data from anywhere on the web. It offers industry-leading durability, availability, and security.

Key features of S3 include:

- **Scalability**: Store and retrieve unlimited data.
- **Durability**: 99.999999999% durability for stored objects.

- **Security**: Fine-grained access control using IAM policies.
- **Cost-Effective**: Multiple storage classes to optimize cost.

IAM Role (Identity and Access Management)

AWS IAM roles allow you to manage permissions and control access to AWS resources. An IAM role is an AWS identity with specific permissions that can be assumed by users, applications, or services needing temporary access to AWS resources.

Key features of IAM roles include:

- **Fine-Grained Permissions**: Define specific permissions for resources.
- **Security**: Enhance security with roles and policies.
- **Temporary Access**: Provide temporary access to resources without sharing long-term credentials.

Serverless Service

Serverless computing enables developers to build and run applications without managing servers. AWS Lambda, the core serverless service, allows you to run code in response to events and automatically manages the underlying compute resources.

Key features of serverless services include:

- **Automatic Scaling**: Scale automatically in response to demand.
- **Cost Efficiency**: Pay only for the compute time consumed.
- **Increased Agility**: Focus on writing code without worrying about infrastructure.
- **Event-Driven**: Trigger functions based on events from other AWS services or external sources.

Benefits of Learning Cloud Computing, AWS, EC2, S3, IAM, and Serverless

During a technology training internship, learning about cloud computing and AWS is crucial. Trainees acquire skills to deploy, manage, and scale applications in the cloud, ensuring high availability and fault tolerance. Understanding EC2 allows trainees to leverage scalable computing resources, while S3 provides efficient and durable storage solutions. IAM roles ensure secure access management, and serverless services like Lambda enable the development of highly scalable and cost-effective applications. This foundational knowledge is critical for any aspiring cloud engineer or software developer and serves as a stepping stone to more advanced topics in cloud architecture and DevOps.

2.3 INTRODUCTION TO KUBERNETES

Kubernetes is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. Developed by Google and now maintained by the Cloud Native Computing Foundation

(CNCF), Kubernetes has become the industry standard for container orchestration due to its powerful and flexible architecture.

Key features of Kubernetes include:

Container Orchestration: Kubernetes automates the deployment, scaling, and operations of application containers across clusters of hosts, providing a container-centric infrastructure.

Declarative Configuration: Users define their desired state for applications and Kubernetes maintains that state, automatically adjusting as needed.

Scalability: Kubernetes can scale applications up and down based on demand, ensuring optimal use of resources.

Self-Healing: Kubernetes automatically restarts, replaces, or reschedules containers that fail, become unresponsive, or need to be moved.

Service Discovery and Load Balancing: Kubernetes can automatically expose a container using the DNS name or their own IP address and distribute network traffic to balance load.

Automated Rollouts and Rollbacks: Kubernetes can update applications progressively, monitoring their health to ensure updates do not disrupt service. If something goes wrong, it can rollback changes automatically.

.

During a technology training internship, learning Kubernetes is invaluable for modern DevOps and cloud-native development.

2.4 INTRODUCTION TO PYTHON

Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Developed by the Django Software Foundation, Django is designed to help developers build robust and scalable web applications quickly.

Key features of Django include:

MVC Architecture: Django follows the Model-View-Controller (MVC) architectural pattern, which helps separate the business logic from the user interface.

Built-in Admin Interface: Django comes with an automatic admin interface, which allows for easy management of application data.

ORM (Object-Relational Mapping): Django's ORM allows developers to interact with the database using Python code instead of SQL, making database operations more intuitive and less error-prone.

Scalability and Security: Django is designed to handle high-traffic sites and includes built-in protection against common security threats such as SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).

Comprehensive Documentation: Django has extensive and well-maintained documentation, making it easier for developers to learn and use the framework effectively.

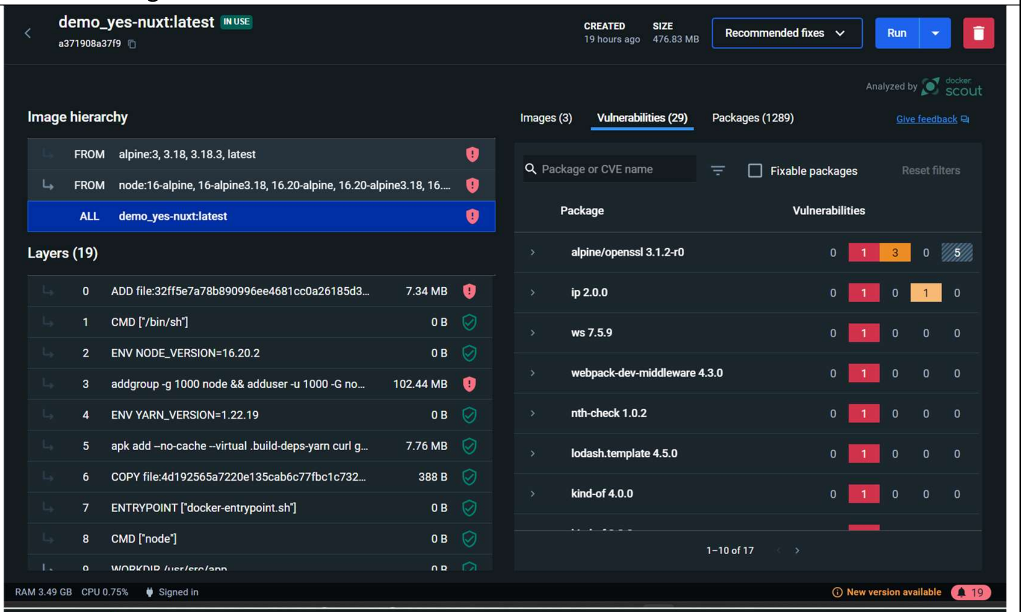
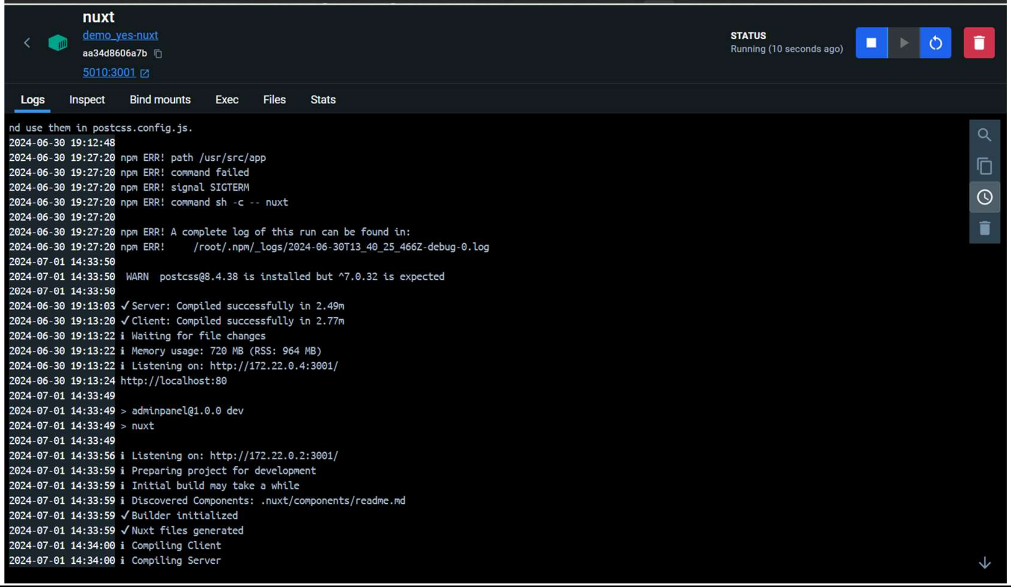
During a technology training internship, learning Django is essential for building modern web applications. Trainees gain experience with setting up and configuring Django projects, defining models and views, creating templates, and managing URLs. They also learn about Django's powerful admin interface, authentication system, and deployment techniques. Mastery of Django enables developers to create scalable, secure, and maintainable web applications, making it a valuable skill in the web development industry.

.

CHAPTER 3 TASK DESCRIPTION

3. TASK DESCRIPTION

Table 3.1 : Task 1

Task 1	
Task Description	Gain a foundational understanding of what is Docker and Docker image and Docker Container. Learn How to Create Docker Image and Container And Understand Difference Between Docker Image and Container. My First task is Create Docker image for Frontend and that image run as Container.
Output Screenshots	 

demo_yes-nuxt:latest IN USE
a371908a37f9

CREATED 19 hours ago SIZE 476.83 MB

Analyzed by docker scout

Image hierarchy

FROM	alpine:3, 3.18, 3.18.3, latest
FROM	node:16-alpine, 16-alpine3.18, 16.20-alpine, 16.20-alpine3.18, 16....
ALL	demo_yes-nuxt:latest

Layers (19)

0	ADD file:32ff5e7a78b890996ee4681cc0a26185d3...	7.34 MB
1	CMD ["/bin/sh"]	0 B
2	ENV NODE_VERSION=16.20.2	0 B
3	addgroup -g 1000 node && adduser -u 1000 -G no...	102.44 MB
4	ENV YARN_VERSION=1.22.19	0 B
5	apk add --no-cache --virtual .build-deps-yarn curl g...	7.76 MB
6	COPY file:4d192565a7220e135cab6c77fbc1c732...	388 B
7	ENTRYPOINT ["docker-entrypoint.sh"]	0 B
8	CMD ["node"]	0 B
9	WORKDIR /usr/src/app	0 B

Images (3) Vulnerabilities (29) Packages (1289)

Package	Vulnerabilities				
alpine/openssl 3.1.2-r0	0	1	3	0	5
ip 2.0.0	0	1	0	1	0
ws 7.5.9	0	1	0	0	0
webpack-dev-middleware 4.3.0	0	1	0	0	0
nth-check 1.0.2	0	1	0	0	0
lodash.template 4.5.0	0	1	0	0	0
kind-of 4.0.0	0	1	0	0	0

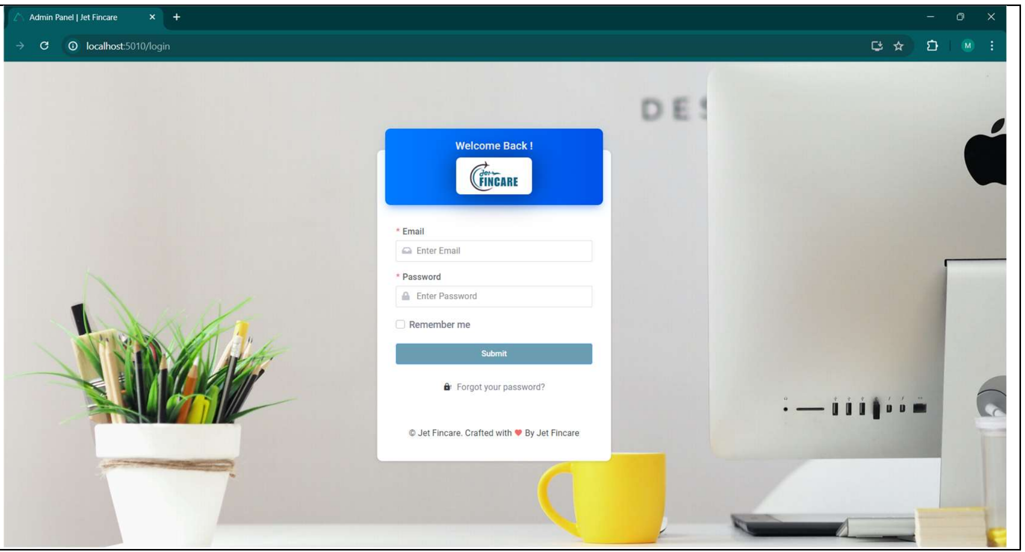
1–10 of 17

RAM 3.49 GB CPU 0.75% Signed in New version available 19

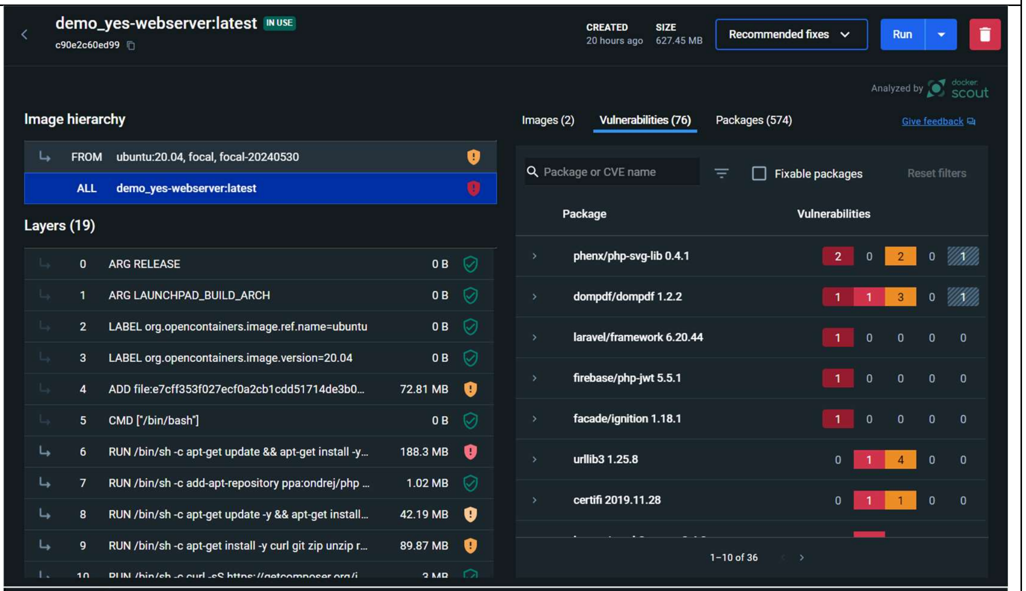
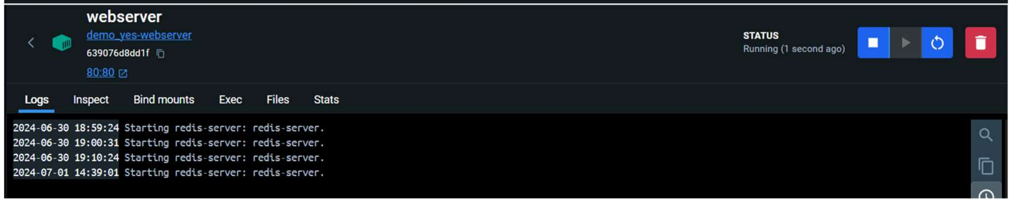
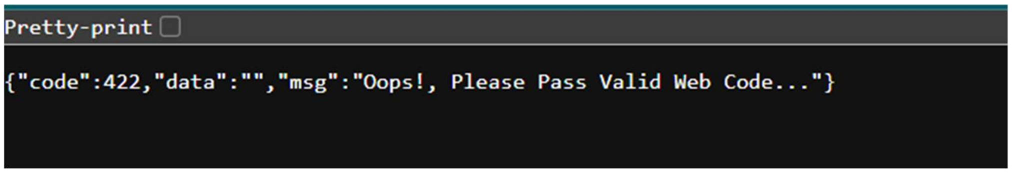
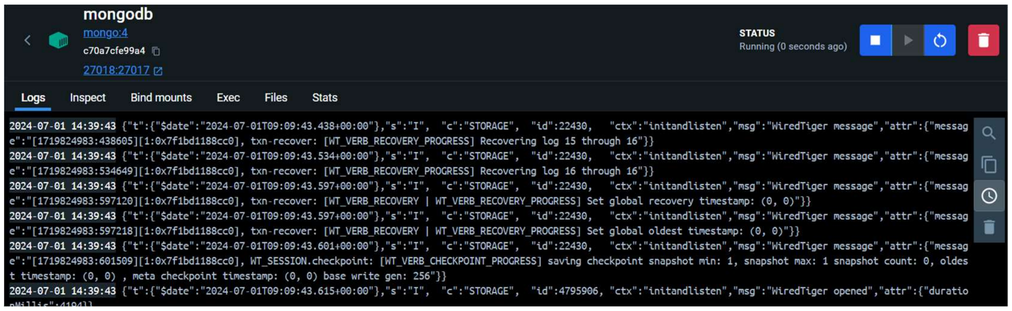
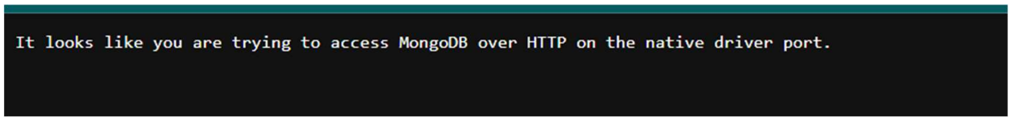
nuxt
demo_yes-nuxt
aa34d8606a7b
5010:3001

STATUS Running (10 seconds ago)

```
nd use them in postcss.config.js.
2024-06-30 19:12:48
2024-06-30 19:27:20 npm ERR! path /usr/src/app
2024-06-30 19:27:20 npm ERR! command failed
2024-06-30 19:27:20 npm ERR! signal SIGTERM
2024-06-30 19:27:20 npm ERR! command sh -c -- nuxt
2024-06-30 19:27:20
2024-06-30 19:27:20 npm ERR! A complete log of this run can be found in:
2024-06-30 19:27:20 npm ERR!     /root/.npm/_logs/2024-06-30T13_40_25_466Z-debug-0.log
2024-07-01 14:33:50
2024-07-01 14:33:50  WARN  postcss@8.4.38 is installed but ^7.0.32 is expected
2024-07-01 14:33:50
2024-06-30 19:13:03 ✓ Server: Compiled successfully in 2.49m
2024-06-30 19:13:20 ✓ Client: Compiled successfully in 2.77m
2024-06-30 19:13:22 ℹ Waiting for file changes
2024-06-30 19:13:22 ℹ Memory usage: 720 MB (RSS: 964 MB)
2024-06-30 19:13:22 ℹ Listening on: http://172.22.0.4:3001/
2024-06-30 19:13:24 http://localhost:80
2024-07-01 14:33:49
2024-07-01 14:33:49 > adminpanel@1.0.0 dev
2024-07-01 14:33:49 > nuxt
2024-07-01 14:33:49
2024-07-01 14:33:56 ℹ Listening on: http://172.22.0.2:3001/
2024-07-01 14:33:59 ℹ Preparing project for development
2024-07-01 14:33:59 ℹ Initial build may take a while
2024-07-01 14:33:59 ℹ Discovered Components: .nuxt/components/readme.md
2024-07-01 14:33:59 ✓ Builder initialized
2024-07-01 14:33:59 ✓ Nuxt files generated
2024-07-01 14:34:00 ℹ Compiling Client
2024-07-01 14:34:00 ℹ Compiling Server
```

Admin Panel | Jet Fincare
localhost:5010/login
Welcome Back !
Email
Enter Email
Password
Enter Password
Remember me
Submit
Forgot your password?
© Jet Fincare. Crafted with ♥ By Jet Fincare

Task 2	
Task Description	Similarly my Task 2 Create Image for Backend and Database. And that image as Container.
Output Screenshots	    

demo_yes-webserver:latest IN USE
c90e2c60ed99

CREATED 20 hours ago SIZE 627.45 MB

Image hierarchy

FROM	ubuntu:20.04, focal, focal-20240530
ALL	demo_yes-webserver:latest

Layers (19)

0	ARG RELEASE	0 B
1	ARG LAUNCHPAD_BUILD_ARCH	0 B
2	LABEL org.opencontainers.image.ref.name=ubuntu	0 B
3	LABEL org.opencontainers.image.version=20.04	0 B
4	ADD file:e7cff353f027ecf0a2cb1cdd51714de3b0…	72.81 MB
5	CMD ["/bin/bash"]	0 B
6	RUN /bin/sh -c apt-get update && apt-get install -y…	188.3 MB
7	RUN /bin/sh -c add-apt-repository ppa:ondrej/php …	1.02 MB
8	RUN /bin/sh -c apt-get update -y && apt-get install…	42.19 MB
9	RUN /bin/sh -c apt-get install -y curl git zip unzip r…	89.87 MB

Images (2) Vulnerabilities (76) Packages (574)

Package	Vulnerabilities				
phenx/php-svg-lib 0.4.1	2	0	2	0	1
dompdf/dompdf 1.2.2	1	1	3	0	1
laravel/framework 6.20.44	1	0	0	0	0
firebase/php-jwt 5.5.1	1	0	0	0	0
facade/ignition 1.18.1	1	0	0	0	0
urllib3 1.25.8	0	1	4	0	0
certifi 2019.11.28	0	1	1	0	0

1–10 of 36

webserver
demo_yes-webserver
639076d8dd1f
80:80

STATUS Running (1 second ago)

```
2024-06-30 18:59:24 Starting redis-server: redis-server.
2024-06-30 19:00:31 Starting redis-server: redis-server.
2024-06-30 19:10:24 Starting redis-server: redis-server.
2024-07-01 14:39:01 Starting redis-server: redis-server.
```

Pretty-print

```
{"code":422,"data":"","msg":"Oops!, Please Pass Valid Web Code..."}
```

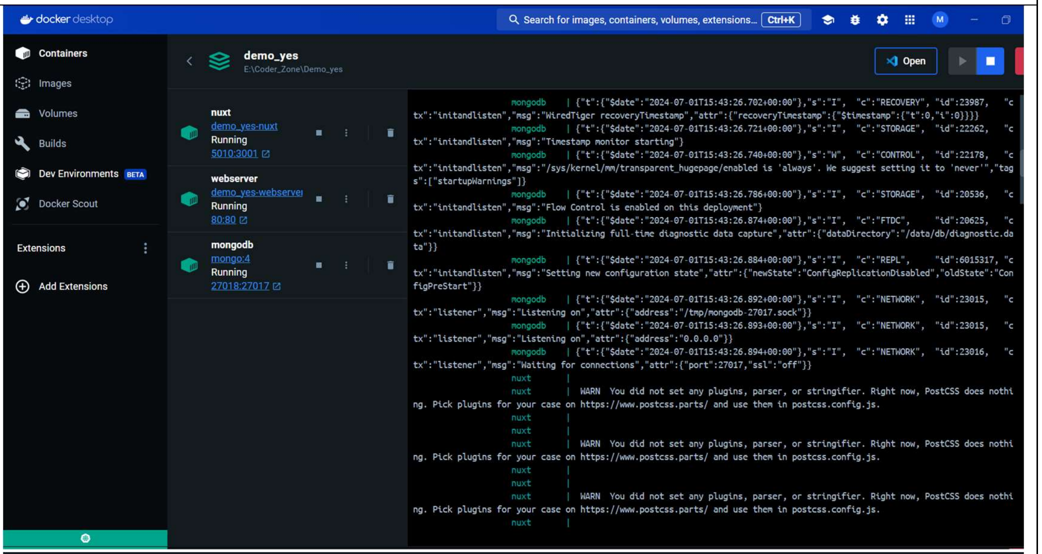
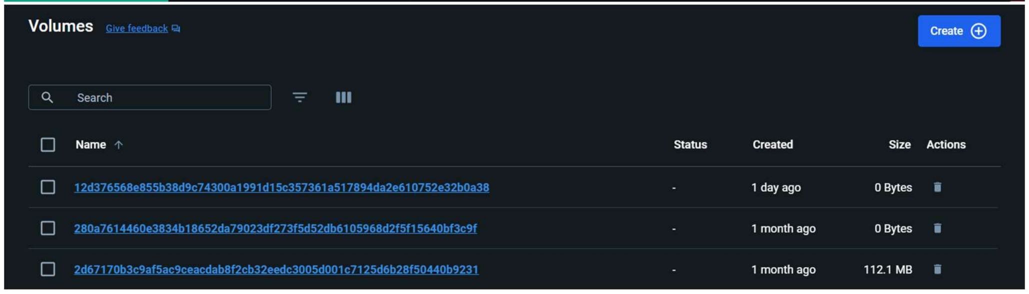
mongodb
mongo:4
c70a7cfe99a4
27018:27017

STATUS Running (0 seconds ago)

```
2024-07-01 14:39:43 {"t":{"$date":"2024-07-01T09:09:43.438+00:00"},"s":"I",  "c":"STORAGE",  "id":22430,   "ctx":"initandlisten","msg":"WiredTiger message","attr":{"messag
e":"[1719824983:438605][1:0x7f1bd1188cc0], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 15 through 16"}}
2024-07-01 14:39:43 {"t":{"$date":"2024-07-01T09:09:43.534+00:00"},"s":"I",  "c":"STORAGE",  "id":22430,   "ctx":"initandlisten","msg":"WiredTiger message","attr":{"messag
e":"[1719824983:534649][1:0x7f1bd1188cc0], txn-recover: [WT_VERB_RECOVERY_PROGRESS] Recovering log 16 through 16"}}
2024-07-01 14:39:43 {"t":{"$date":"2024-07-01T09:09:43.597+00:00"},"s":"I",  "c":"STORAGE",  "id":22430,   "ctx":"initandlisten","msg":"WiredTiger message","attr":{"messag
e":"[1719824983:597120][1:0x7f1bd1188cc0], txn-recover: [WT_VERB_RECOVERY | WT_VERB_RECOVERY_PROGRESS] Set global recovery timestamp: (0, 0)"}}
2024-07-01 14:39:43 {"t":{"$date":"2024-07-01T09:09:43.597+00:00"},"s":"I",  "c":"STORAGE",  "id":22430,   "ctx":"initandlisten","msg":"WiredTiger message","attr":{"messag
e":"[1719824983:597218][1:0x7f1bd1188cc0], txn-recover: [WT_VERB_RECOVERY | WT_VERB_RECOVERY_PROGRESS] Set global oldest timestamp: (0, 0)"}}
2024-07-01 14:39:43 {"t":{"$date":"2024-07-01T09:09:43.601+00:00"},"s":"I",  "c":"STORAGE",  "id":22430,   "ctx":"initandlisten","msg":"WiredTiger message","attr":{"messag
e":"[1719824983:601509][1:0x7f1bd1188cc0], WT_SESSION.checkpoint: [WT_VERB_CHECKPOINT_PROGRESS] saving checkpoint snapshot min: 1, snapshot max: 1 snapshot count: 0, oldes
t timestamp: (0, 0) , meta checkpoint timestamp: (0, 0) base write gen: 256"}}
2024-07-01 14:39:43 {"t":{"$date":"2024-07-01T09:09:43.615+00:00"},"s":"I",  "c":"STORAGE",  "id":4795906, "ctx":"initandlisten","msg":"WiredTiger opened","attr":{"duratio
```

```
It looks like you are trying to access MongoDB over HTTP on the native driver port.
```

Table 3.2 : Task 2

Task 3	
Task Description	3rd Task is Create Network and Connect all this three Container in One Network and Also Create Volume For Database. Run Full Stack Web Site in Docker together.
Output Screenshots	 

demo_yes
E:\Coder_Zone\Demo_yes

nuxt
demo_yes-nuxt
Running
5010:3001

webserver
demo_yes-webserver
Running
80:80

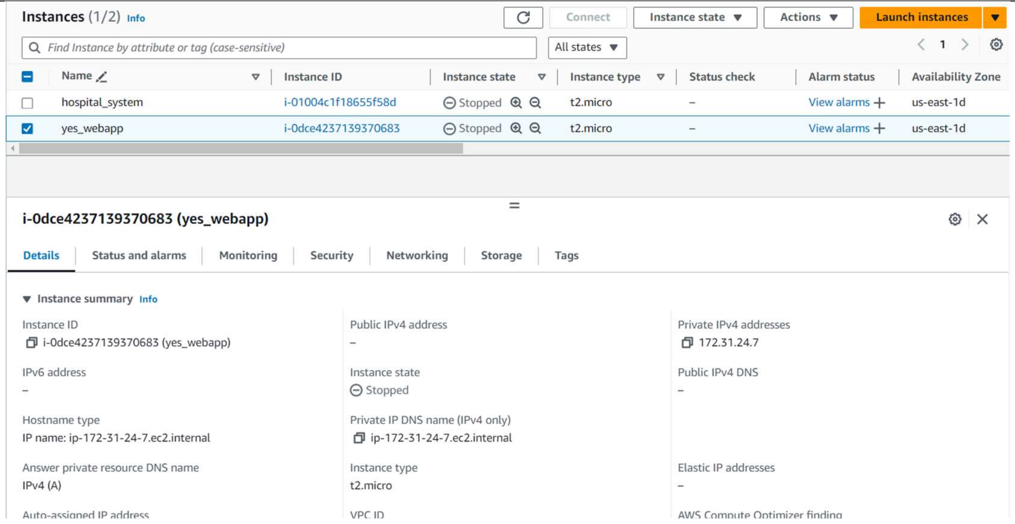
mongodb
mongo:4
Running
27018:27017

```
               mongodb    | {"t":{"$date":"2024-07-01T15:43:26.702+00:00"},"s":"I",  "c":"RECOVERY", "id":23987,   "c
tx":"initandlisten","msg":"WiredTiger recoveryTimestamp","attr":{"recoveryTimestamp":{"$timestamp":{"t":0,"i":0}}}}
               mongodb    | {"t":{"$date":"2024-07-01T15:43:26.721+00:00"},"s":"I",  "c":"STORAGE",  "id":22262,   "c
tx":"initandlisten","msg":"Timestamp monitor starting"}
               mongodb    | {"t":{"$date":"2024-07-01T15:43:26.740+00:00"},"s":"W",  "c":"CONTROL",  "id":22178,   "c
tx":"initandlisten","msg":"/sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'never'","tag
s":["startupWarnings"]}
               mongodb    | {"t":{"$date":"2024-07-01T15:43:26.786+00:00"},"s":"I",  "c":"STORAGE",  "id":20536,   "c
tx":"initandlisten","msg":"Flow Control is enabled on this deployment"}
               mongodb    | {"t":{"$date":"2024-07-01T15:43:26.874+00:00"},"s":"I",  "c":"FTDC",     "id":20625,   "c
tx":"initandlisten","msg":"Initializing full-time diagnostic data capture","attr":{"dataDirectory":"/data/db/diagnostic.da
ta"}}
               mongodb    | {"t":{"$date":"2024-07-01T15:43:26.884+00:00"},"s":"I",  "c":"REPL",     "id":6015317, "c
tx":"initandlisten","msg":"Setting new configuration state","attr":{"newState":"ConfigReplicationDisabled","oldState":"Con
figPreStart"}}
               mongodb    | {"t":{"$date":"2024-07-01T15:43:26.892+00:00"},"s":"I",  "c":"NETWORK",  "id":23015,   "c
tx":"listener","msg":"Listening on","attr":{"address":"/tmp/mongodb-27017.sock"}}
               mongodb    | {"t":{"$date":"2024-07-01T15:43:26.893+00:00"},"s":"I",  "c":"NETWORK",  "id":23015,   "c
tx":"listener","msg":"Listening on","attr":{"address":"0.0.0.0"}}
               mongodb    | {"t":{"$date":"2024-07-01T15:43:26.894+00:00"},"s":"I",  "c":"NETWORK",  "id":23016,   "c
tx":"listener","msg":"Waiting for connections","attr":{"port":27017,"ssl":"off"}}
               nuxt       |
               nuxt       |  WARN  You did not set any plugins, parser, or stringifier. Right now, PostCSS does nothi
ng. Pick plugins for your case on https://www.postcss.parts/ and use them in postcss.config.js.
               nuxt       |
               nuxt       |
               nuxt       |  WARN  You did not set any plugins, parser, or stringifier. Right now, PostCSS does nothi
ng. Pick plugins for your case on https://www.postcss.parts/ and use them in postcss.config.js.
               nuxt       |
               nuxt       |
               nuxt       |  WARN  You did not set any plugins, parser, or stringifier. Right now, PostCSS does nothi
ng. Pick plugins for your case on https://www.postcss.parts/ and use them in postcss.config.js.
               nuxt       |
```

Volumes Give feedback

Name	Status	Created	Size	Actions
12d376568e855b38d9c74300a1991d15c357361a517894da2e610752e32b0a38	-	1 day ago	0 Bytes	
280a7614460e3834b18652da79023df273f5d52db6105968d2f5f15640bf3c9f	-	1 month ago	0 Bytes	
2d67170b3c9af5ac9ceacdab8f2cb32eedc3005d001c7125d6b28f50440b9231	-	1 month ago	112.1 MB	

Table 3.3 : Task 3

Task 4	
Task Description	Set up Aws Account and Launch EC2 Instance For deployment and Setup Security Group and Other Configuration.
Output Screenshots	

Instances (1/2) Info

Find Instance by attribute or tag (case-sensitive) — All states

	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
☐	hospital_system	i-01004c1f18655f58d	⊖ Stopped	t2.micro	–	View alarms +	us-east-1d
☑	yes_webapp	i-0dce4237139370683	⊖ Stopped	t2.micro	–	View alarms +	us-east-1d

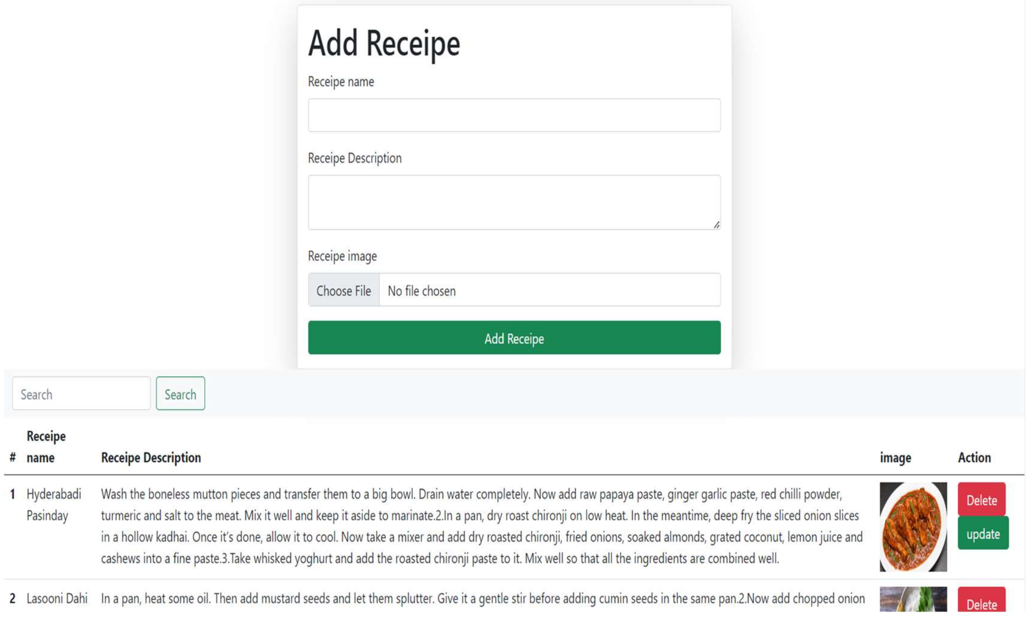
i-0dce4237139370683 (yes_webapp)

Details | Status and alarms | Monitoring | Security | Networking | Storage | Tags

Instance summary Info

Instance ID i-0dce4237139370683 (yes_webapp)	Public IPv4 address –	Private IPv4 addresses 172.31.24.7
IPv6 address –	Instance state Stopped	Public IPv4 DNS –
Hostname type IP name: ip-172-31-24-7.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-24-7.ec2.internal	
Answer private resource DNS name IPv4 (A)	Instance type t2.micro	Elastic IP addresses –
Auto-assigned IP address	VPC ID	AWS Compute Optimizer finding

Table 3.4 : Task 4

Task 5	
Task Description	Set Django Project. Perform CRUD Operation in Django and make Recipe Web Site.
Output Screenshots	

Add Receipe

Receipe name

Receipe Description

Receipe image

Choose File No file chosen

Add Receipe

Search

#	Receipe name	Receipe Description	image	Action
1	Hyderabadi Pasinday	Wash the boneless mutton pieces and transfer them to a big bowl. Drain water completely. Now add raw papaya paste, ginger garlic paste, red chilli powder, turmeric and salt to the meat. Mix it well and keep it aside to marinate.2.In a pan, dry roast chironji on low heat. In the meantime, deep fry the sliced onion slices in a hollow kadhai. Once it's done, allow it to cool. Now take a mixer and add dry roasted chironji, fried onions, soaked almonds, grated coconut, lemon juice and cashews into a fine paste.3.Take whisked yoghurt and add the roasted chironji paste to it. Mix well so that all the ingredients are combined well.		Delete update
2	Lasooni Dahi	In a pan, heat some oil. Then add mustard seeds and let them splutter. Give it a gentle stir before adding cumin seeds in the same pan.2.Now add chopped onion		Delete

Table 3.5 : Task 5

CHAPTER 4 LEARNING EXPERIENCES

4.1 KNOWLEDGE ACQUIRED/SKILLS LEARNT

During my internship, I gained extensive knowledge and skills in Docker, AWS, and Django. Key learnings include:

Technical Skills:

Docker:

- **Containerization Fundamentals**: Developed a deep understanding of containerization concepts and the benefits of using containers for application deployment. Learned to create and manage containers using Docker.
- **Docker Images and Dockerfile**: Gained proficiency in writing Dockerfiles to create custom Docker images, incorporating all necessary dependencies and configurations.
- **Container Orchestration**: Explored orchestration tools such as Docker Compose to define and run multi-container Docker applications. Managed container lifecycles and automated deployments.
- **Networking and Volume Management**: Configured container networks to enable communication between containers. Used Docker volumes to persist data and manage stateful applications.
- **Best Practices**: Implemented best practices for building efficient and secure Docker images, including minimizing image size and managing secrets.

AWS:

- **Cloud Computing Fundamentals**: Acquired a comprehensive understanding of cloud computing concepts and the advantages of using AWS for scalable and cost-effective solutions.
- **EC2 (Elastic Compute Cloud)**: Learned to launch and configure EC2 instances, manage security groups, and deploy applications on virtual servers. Gained skills in monitoring and scaling instances.
- **S3 (Simple Storage Service)**: Used S3 for object storage, learning to manage buckets, set permissions, and implement lifecycle policies for cost-effective storage management.
- **IAM (Identity and Access Management)**: Managed access to AWS resources by creating and assigning IAM roles, policies, and users. Ensured secure access control and implemented best practices for AWS security.
- **Serverless Computing**: Explored AWS Lambda to create serverless functions that automatically scale and integrate with other AWS services. Developed and deployed serverless applications, reducing operational overhead.

Django:

- **Web Development with Django**: Gained proficiency in building web applications using the Django framework, following the Model-View-Controller (MVC) architectural pattern.

- **Django ORM (Object-Relational Mapping)**: Used Django's ORM to interact with databases using Python code. Defined models, executed queries, and managed database migrations.
- **Template Engine**: Created dynamic web pages using Django's template engine, incorporating HTML, CSS, and JavaScript. Implemented reusable templates for consistent UI design.
- **Form Handling and Validation**: Built forms to handle user input and validated data using Django's form handling capabilities. Ensured robust user input management and error handling.
- **Authentication and Authorization**: Implemented user authentication and authorization using Django's built-in authentication system. Managed user roles and permissions to secure application access.

These experiences have significantly enhanced my technical abilities and professional skills, preparing me for a successful career in development and cloud computing.

4.2 INDUSTRY PRACTICES ADAPTED

- **Taking Screenshots at Each Step:** I took screenshots during each step of my tasks. This helped me keep track of my progress, solve problems more easily, and communicate clearly with my team.
- **Writing Clean Code**: I made sure to write code that was easy to read and maintain. I followed good coding practices, respected intellectual property, and prioritized security and privacy.
- **Regular Testing:** I regularly tested my code by performing unit and integration tests. This helped me find and fix bugs early, making the application more stable and reliable.
- **Participating in Code Reviews**: I took part in code reviews, which provided me with valuable feedback. This practice improved the quality of my code and allowed me to learn from my peers, enhancing my coding skills and understanding.

4.3 REALTIME APPLICABILITY OF TECHNOLOGIES LEARNT

1. DOCKER:

- **Consistent Development Environment:** Docker ensures that developers work in consistent environments, reducing the "it works on my machine" problem. This consistency extends across different operating systems and deployment environments, ensuring seamless collaboration and fewer integration issues.
- **Efficient Application Deployment**: Docker containers package applications and their dependencies into isolated units, which can be easily deployed across various environments, from development to production. This simplifies the deployment process, accelerates time-to-market, and improves deployment reliability**.**
- **Microservices Architecture:** Docker facilitates the adoption of microservices architecture by allowing each service to run in its own container. This modular

approach enhances scalability, flexibility, and fault isolation, enabling organizations to develop and deploy complex applications more efficiently.
- **Continuous Integration and Deployment (CI/CD):** Docker integrates seamlessly with CI/CD pipelines, automating the build, test, and deployment processes. By using Docker images as artifacts, teams can ensure consistency between development, testing, and production environments, leading to faster release cycles and improved software quality.
- **Resource Optimization:** Docker containers share the host system's kernel and only include necessary dependencies, making them lightweight and efficient in resource utilization. This efficiency enables organizations to optimize infrastructure costs and maximize the utilization of computing resources.
- **Cloud-Native Development:** Docker is a foundational technology in cloud-native development, enabling applications to be developed, deployed, and managed in cloud environments like AWS, Azure, and Google Cloud Platform. This aligns with modern practices of scalability, resilience, and agility in software development.

2. AWS:

- **Computing Power:** AWS offers virtual servers known as EC2 instances. These instances can be used to run applications, host websites, or perform data processing tasks without needing physical hardware.
- **Storage:** AWS provides various storage options, including S3 (Simple Storage Service), which allows you to store and retrieve large amounts of data securely and flexibly.
- **Database Management:** AWS offers managed database services like RDS (Relational Database Service) and DynamoDB, which simplify database setup, management, and scaling.
- **Security and Identity:** AWS IAM (Identity and Access Management) lets you control who can access your AWS resources. It helps manage users, groups, and permissions to ensure secure access to your applications and data.
- **Scaling and Elasticity:** AWS allows you to easily scale your resources up or down based on demand. This elasticity ensures that your applications can handle varying levels of traffic without manual intervention.
- **Deployment and Management Tools:** AWS provides tools like CloudFormation for automating infrastructure deployment, and AWS Elastic Beanstalk for deploying and managing applications without worrying about the underlying infrastructure.

3. DJANGO:

- **Web Development Made Easier:** Django simplifies the process of building web applications by providing ready-to-use components and tools. It follows the "batteries-included" philosophy, meaning it includes everything you need to get started.
- **Model-View-Controller (MVC) Design:** Django uses a design pattern called Model-View-Controller (MVC). This separates different aspects of the application:
  - Models define the data structure, usually related to your database.

    - Views render the data to the user, usually in HTML templates.
    - Controllers handle the logic and user inputs.
- **Admin Interface:** Django comes with a built-in admin interface. This allows you to manage and update your application's data through a user-friendly interface without writing a lot of code.
- **Database Management:** Django provides an Object-Relational Mapping (ORM) layer, which allows you to interact with your database using Python code. You can define your data models as Python classes, and Django handles the database queries**.**
- **URL Routing:** Django uses a URL routing mechanism that maps URLs to views. This makes it easy to organize your application's URLs and direct users to the appropriate views or pages**.**
- **Security Features**: Django includes built-in security features, such as protection against common web vulnerabilities like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF). This helps developers build secure web applications by default.

4. KUBERNETES:

- **Container Orchestration:** Kubernetes automates the deployment, scaling, and management of containerized applications. It allows you to run containers (like those created with Docker) across a cluster of nodes (machines), ensuring that your applications are always available and running smoothly.
- **Cluster Management:** Kubernetes organizes your containers into groups called pods. Pods are the smallest unit in Kubernetes and can contain one or more containers that work together. Kubernetes manages these pods and ensures they run in the desired state, restarting or replacing them if necessary.
- **Scaling:** Kubernetes makes it easy to scale your application horizontally by adding or removing pods based on traffic or resource utilization. This ensures that your application can handle varying levels of demand without manual intervention.
- **Service Discovery and Load Balancing:** Kubernetes includes built-in mechanisms for service discovery and load balancing. It automatically assigns network addresses to pods and distributes incoming traffic across them, ensuring efficient use of resources and optimal performance.

CHAPTER 5 CONCLUSION

"In conclusion, my internship experience at Codderzone Solution. lt has been invaluable in shaping my skills and understanding of Devops. Over the 1- month , I had the opportunity to work on specific tasks, where I applied Docker ,Cloud Computing ,AWS ,Django. I am proud to have achieved certification of internship which demonstrates my growth in Devops. This internship has not only enhanced my technical proficiency but also improved my problem-solving abilities and collaboration skills within a professional team setting. Looking ahead, I am eager to continue exploring Devops Life cycle and leveraging my knowledge of the Devops in future projects. I am grateful Codderzone Solution for this enriching experience and to my mentors for their guidance and support throughout."