Event Booking Application Report

Repository:
https://github.com/AyaanKhan1576/Event-Booking-Microservices-DevOps-Project.git

By:

Mishal Ali 22i-1291

Ayaan Khan 22i-0832

Minahil Ali 22i-0849

1. Introduction

This report documents the architecture, toolchain, workflows, and DevOps practices implemented in the development of a scalable, containerized, microservices-based Event Booking platform.

The system integrates a complete CI/CD and GitOps pipeline using Terraform, Ansible, Docker, Kubernetes, Argo CD, GitHub Actions, and observability tools like Prometheus and Grafana.

The platform demonstrates robust infrastructure-as-code, continuous delivery, monitoring, and service orchestration in both cloud and local Kubernetes environments.

2. Architecture Overview

The application follows a microservices architecture with the following components:

Component	Description
API Gateway	Routes requests to backend services.
Event Service	Manages event creation, updates, and queries.
Booking Service	Handles ticket reservations and payment processing.
User Service	Manages user authentication and profiles.
Databases	MongoDB and MySQL .

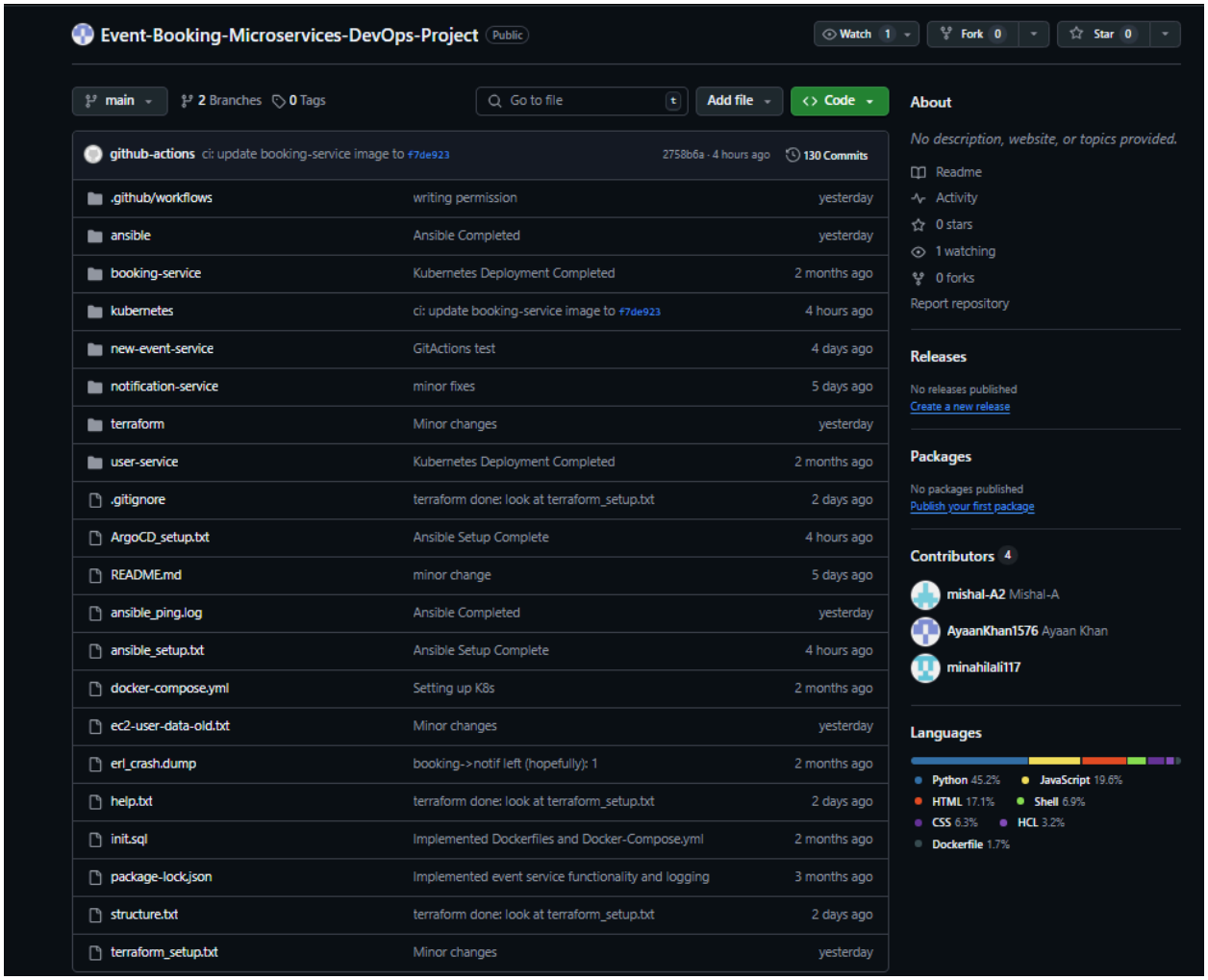
3. DevOps Tools

3.1 GitHub

Role: Source code management, collaboration and version control

Implementation:

Repository structure includes application code, Dockerfiles, Kubernetes manifests, and Argo CD configurations.



3.2 GitHub Actions

Role: CI pipeline automation.

All microservices (Booking, Event, Notification, User) follow a consistent GitHub Actions pattern with slight customizations per service. The Gitactions use *DOCKERHUB_USERNAME* and *DOCKERHUB_TOKEN* that are set in repo secrets.

Workflow:

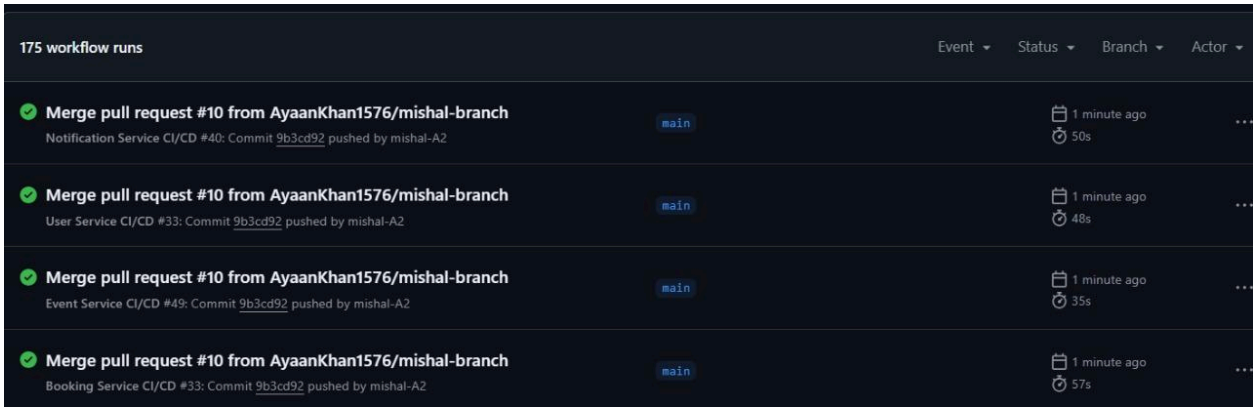
on:

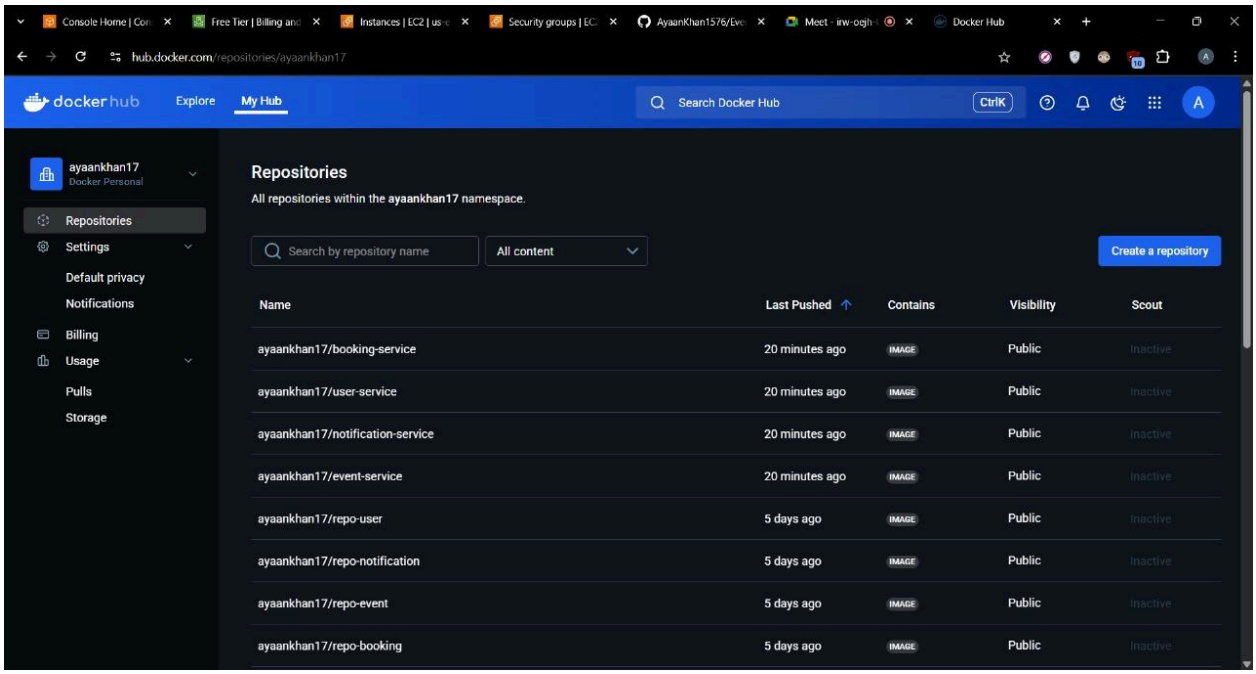
```
  push:
    branches: ["main"]
  pull_request:
    branches: ["main"]
permissions:
  contents: write
```

- push to main: Runs CI/CD on new commits.
- pull_request on main: Validates builds for incoming PRs
- Allows committing updated Kubernetes manifests back to the repo.

Each workflow defines two primary jobs:

1. **build-and-push**: Builds a Docker image and pushes to Docker Hub.
2. **update-k8s-manifest**: Updates the service's Kubernetes Deployment YAML with the new image tag and commits it in the repo to be used for argoCD in the future.





3.3 Terraform

Role: Infrastructure-as-Code (IaC) provisioning.

Implementation:

Provisions cloud resources (VMs, networks, Kubernetes clusters).

Code location: terraform/ directory.

Instances (1/1) Info

Last updated 3 minutes ago | Connect | Instance state | Actions | Launch instances

Find Instance by attribute or tag (case-sensitive) | All states | 1

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
EventAppServer	i-0ffe61c2a270ceb87	Running	t2.micro	2/2 checks passed	View alarms +	us-east-1d	ec2-44-208-164-168.

i-0ffe61c2a270ceb87 (EventAppServer)

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

Instance summary Info

Instance ID
i-0ffe61c2a270ceb87

IPv6 address
–

Hostname type
IP name: ip-172-31-85-120.ec2.internal

Public IPv4 address
44.208.164.168 | open address

Instance state
Running

Private IP DNS name (IPv4 only)
ip-172-31-85-120.ec2.internal

Private IPv4 addresses
172.31.85.120

Public IPv4 DNS
ec2-44-208-164-168.compute-1.amazonaws.com | open address

Inbound security group rules successfully modified on security group (sg-0f0826b599d22bcdd | allow_ssh_web)

Details

Security Groups (2) Info

Actions | Export security groups to CSV | Create security group

Find security groups by attribute or tag | 1

Name	Security group ID	Security group name	VPC ID	Description
–	sg-0e973de27785ce94e	default	vpc-0ab53c57e7cb67611	default VPC security group
–	sg-0f0826b599d22bcdd	allow_ssh_web	vpc-0ab53c57e7cb67611	Allow SSH and HTTP/HTTP

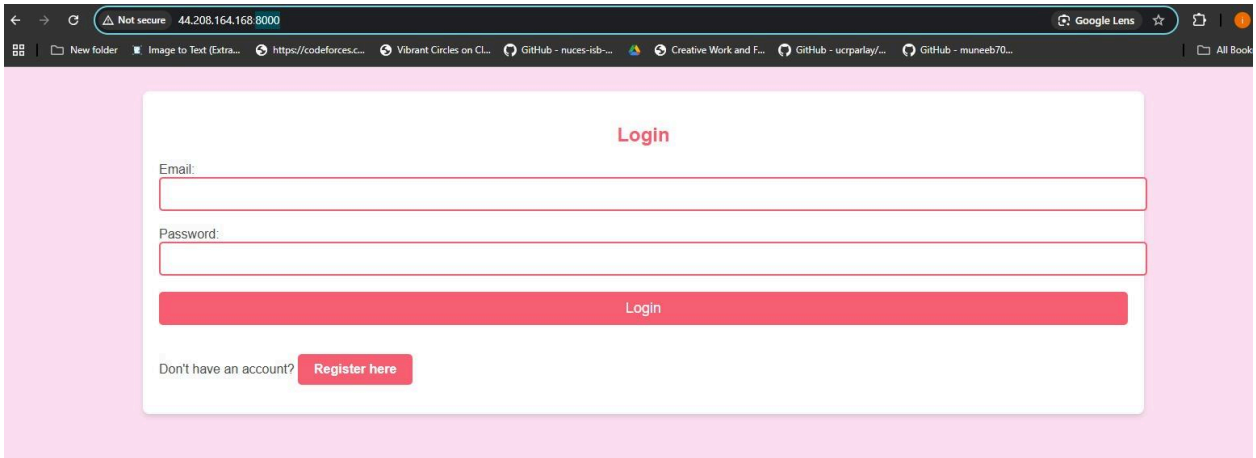
sg-0f0826b599d22bcdd - allow_ssh_web

Inbound rules (5)

Manage tags | Edit inbound rules

Search | 1

Security group rule ID	IP version	Type	Protocol	Port range	Source	D
sgr-087b8ef3dc663a35e	IPv4	HTTP	TCP	80	0.0.0.0/0	H
sgr-066b92e9329a1360b	IPv4	Custom TCP	TCP	8000	0.0.0.0/0	–
sgr-0f544a797fc4e123a	IPv4	Custom TCP	TCP	5001	0.0.0.0/0	–
sgr-085c0aec4a0f5c6e0	IPv4	Custom TCP	TCP	5000	0.0.0.0/0	–
sgr-051d8fc2769f9b736	IPv4	SSH	TCP	22	0.0.0.0/0	S



3.4 Ansible

Role: Configuration management.

Implementation:

Configures servers (installs dependencies, sets up dockers and docker-compose**).**

Playbooks: ansible/playbooks/.

```
[ec2-user@ip-172-31-21-5 ~]$ ls
Event-Booking-Microservices-DevOps-Project
[ec2-user@ip-172-31-21-5 ~]$ docker ps
CONTAINER ID   IMAGE                       COMMAND                  CREATED        STATUS                  PORTS
                                                                                                                    NAMES
5564087f4343   user-service:latest         "sh ./entrypoint.sh"     8 minutes ago  Up 8 minutes            0.0.0.0:8000->8000/tcp, :::800
0->8000/tcp                                                                                                         user-service
a4e7e059350a   booking-service:latest      "sh ./entrypoint.sh"     8 minutes ago  Up 8 minutes            0.0.0.0:5001->5001/tcp, :::500
1->5001/tcp                                                                                                         booking-service
f646b9a52a20   notification-service:latest "docker-entrypoint.s…"   8 minutes ago  Up 8 minutes            0.0.0.0:5002->5002/tcp, :::500
2->5002/tcp                                                                                                         notification-service
8467319e7c88   new-event-service:latest    "docker-entrypoint.s…"   8 minutes ago  Up 8 minutes            0.0.0.0:5000->5000/tcp, :::500
0->5000/tcp                                                                                                         new-event-service
0390fe7acfc6   postgres:latest             "docker-entrypoint.s…"   8 minutes ago  Up 8 minutes (healthy)  0.0.0.0:5432->5432/tcp, :::543
2->5432/tcp                                                                                                         postgres
4a67d59deb1a   mongo:latest                "docker-entrypoint.s…"   8 minutes ago  Up 8 minutes            0.0.0.0:27017->27017/tcp, :::2
7017->27017/tcp                                                                                                     mongodb
792c558410de   rabbitmq:3-management       "docker-entrypoint.s…"   8 minutes ago  Up 8 minutes            4369/tcp, 5671/tcp, 0.0.0.0:56
72->5672/tcp, :::5672->5672/tcp, 15671/tcp, 15691-15692/tcp, 25672/tcp, 0.0.0.0:15672->15672/tcp, :::15672->15672/tcp   rabbitmq
[ec2-user@ip-172-31-21-5 ~]$
```

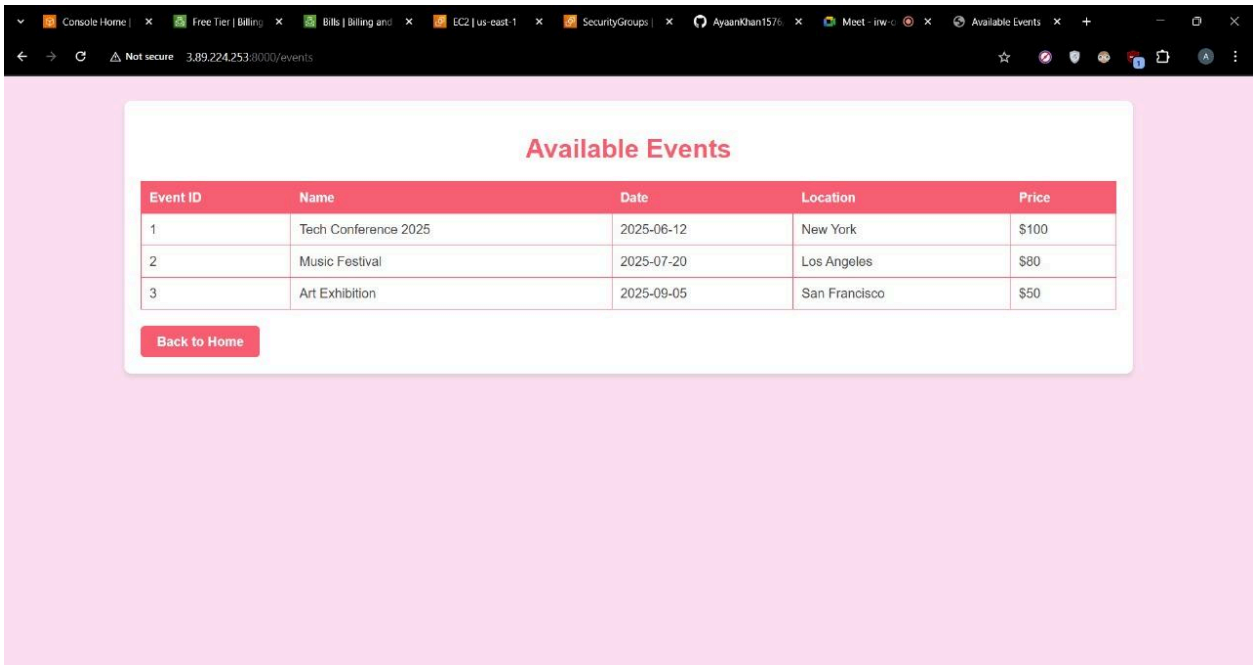
3.5 Docker

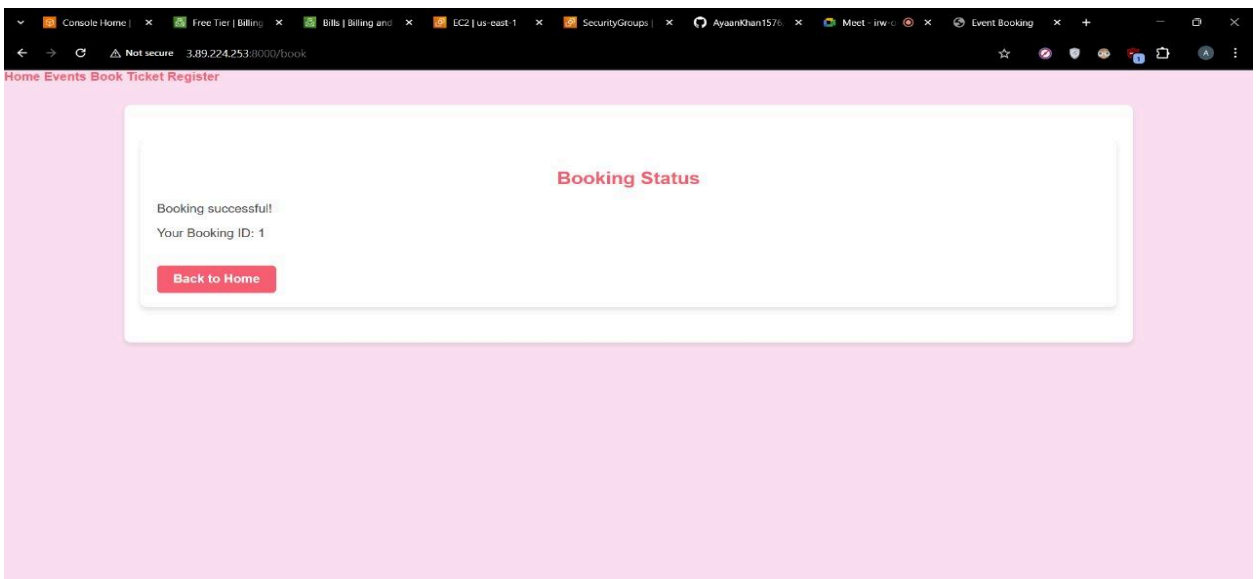
Role: Containerization of microservices.

Implementation:

Each service has a dedicated Dockerfile (e.g., event-service/Dockerfile)**.**

The application also has a docker-compose.yml file that runs all the microservices.





3.6 Kubernetes

Role: Container orchestration.

Implementation:

Deployment manifests in kubernetes folders (including the manifest files such as config.yaml, secrets.yaml etc)

3.7 Argo CD

Role: GitOps-based continuous deployment.

Implementation:

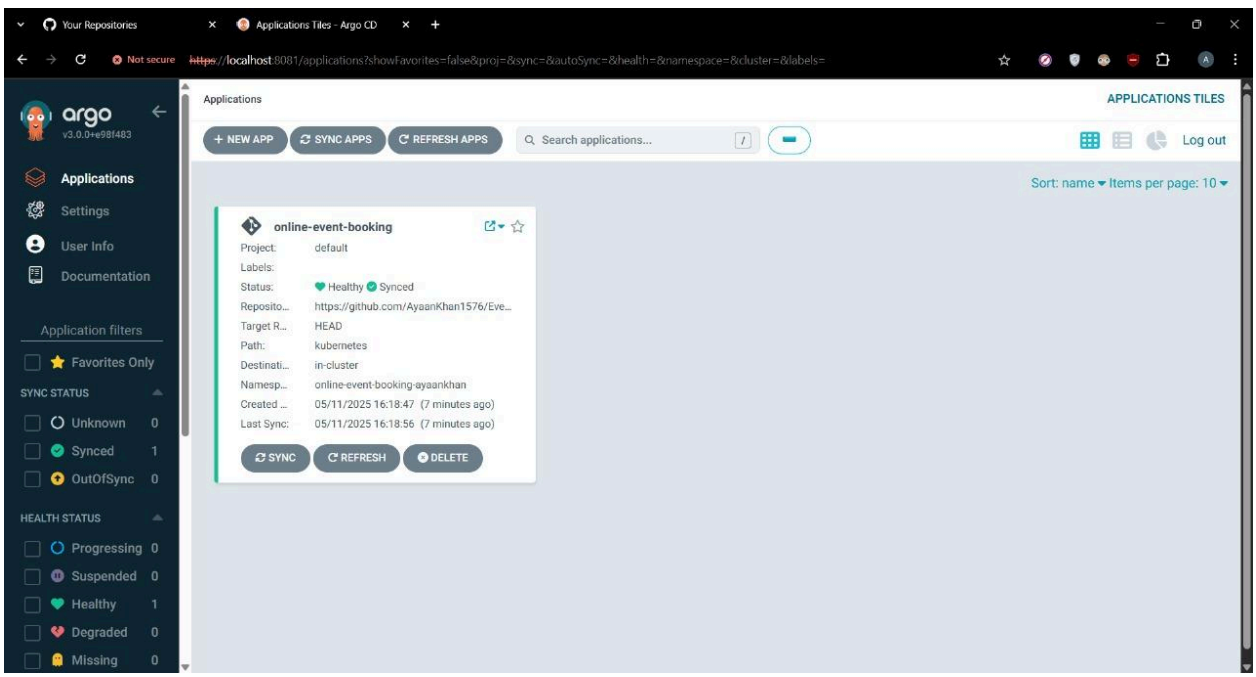
Syncs Kubernetes manifests from Git to the cluster.

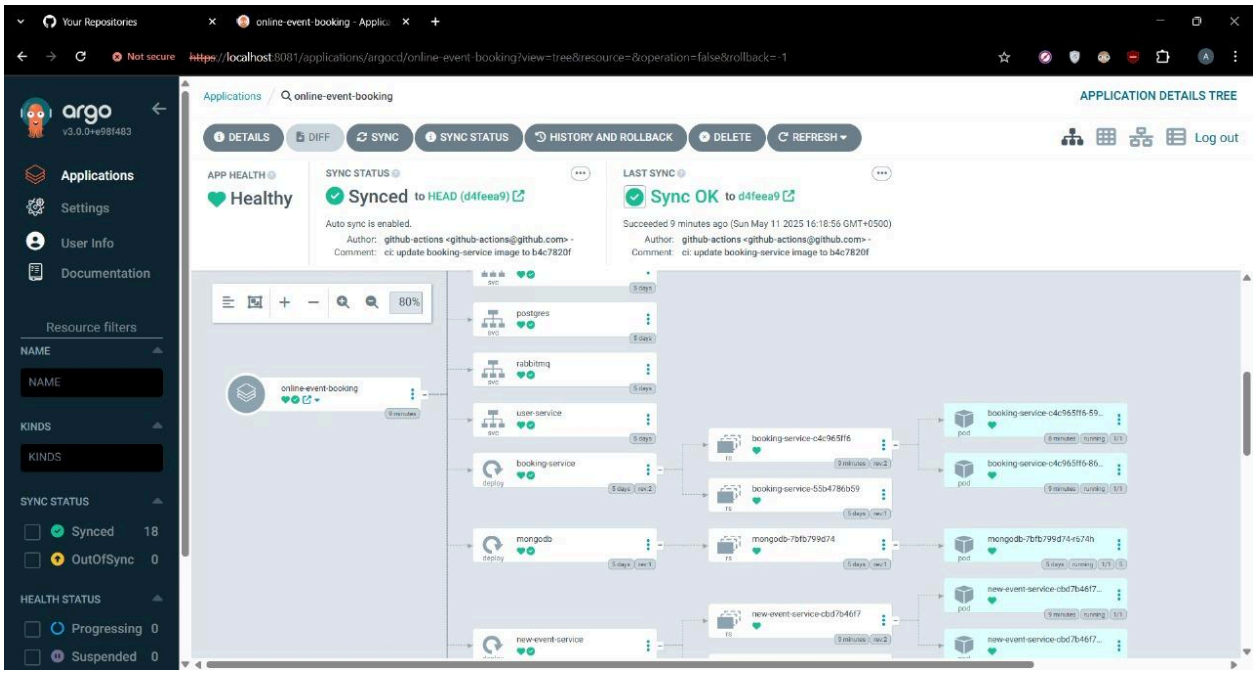
Configuration: argocd/applications/event-booking-app.yaml.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\ayaan\Documents\University\Semester 6\DevOps\Project\Project_Repo\kubernetes> argocd app list
NAME                        CLUSTER                         NAMESPACE                    PROJECT  STATUS  HEALTH   SYNCPOLICY  CONDITIONS
  REPO                                                                                   PATH        TARGET
argocd/online-event-booking  https://kubernetes.default.svc  online-event-booking-ayaankhan  default  Synced  Healthy  Auto-Prune  <none>
  https://github.com/AyaanKhan1576/Event-Booking-Microservices-DevOps-Project.git  kubernetes  HEAD
PS C:\Users\ayaan\Documents\University\Semester 6\DevOps\Project\Project_Repo\kubernetes> argocd app get online-event-booking
Name:               argocd/online-event-booking
Project:            default
Server:             https://kubernetes.default.svc
Namespace:          online-event-booking-ayaankhan
URL:                https://localhost:8081/applications/online-event-booking
Source:
- Repo:             https://github.com/AyaanKhan1576/Event-Booking-Microservices-DevOps-Project.git
  Target:           HEAD
  Path:             kubernetes
SyncWindow:         Sync Allowed
Sync Policy:        Automated (Prune)
Sync Status:        Synced to HEAD (d4feea9)
Health Status:      Healthy

GROUP              KIND        NAMESPACE                    NAME                         STATUS   HEALTH   HOOK  MESSAGE
                   Namespace   online-event-booking-ayaankhan  online-event-booking-ayaankhan  Running  Synced         namespace/online-event
-booking-ayaankhan configured
                   Secret      online-event-booking-ayaankhan  app-secrets                  Synced                  secret/app-secrets con
figured
                   ConfigMap   online-event-booking-ayaankhan  app-config                   Synced                  configmap/app-config c
onfigured
                   Service     online-event-booking-ayaankhan  notification-service         Synced   Healthy        service/notification-s
ervice configured
                   Service     online-event-booking-ayaankhan  rabbitmq                     Synced   Healthy        service/rabbitmq confi
gured
                   Service     online-event-booking-ayaankhan  postgres                     Synced   Healthy        service/postgres confi
gured
                   Service     online-event-booking-ayaankhan  mongodb                      Synced   Healthy        service/mongodb config
ured
                   Service     online-event-booking-ayaankhan  new-event-service            Synced   Healthy        service/new-event-serv
```



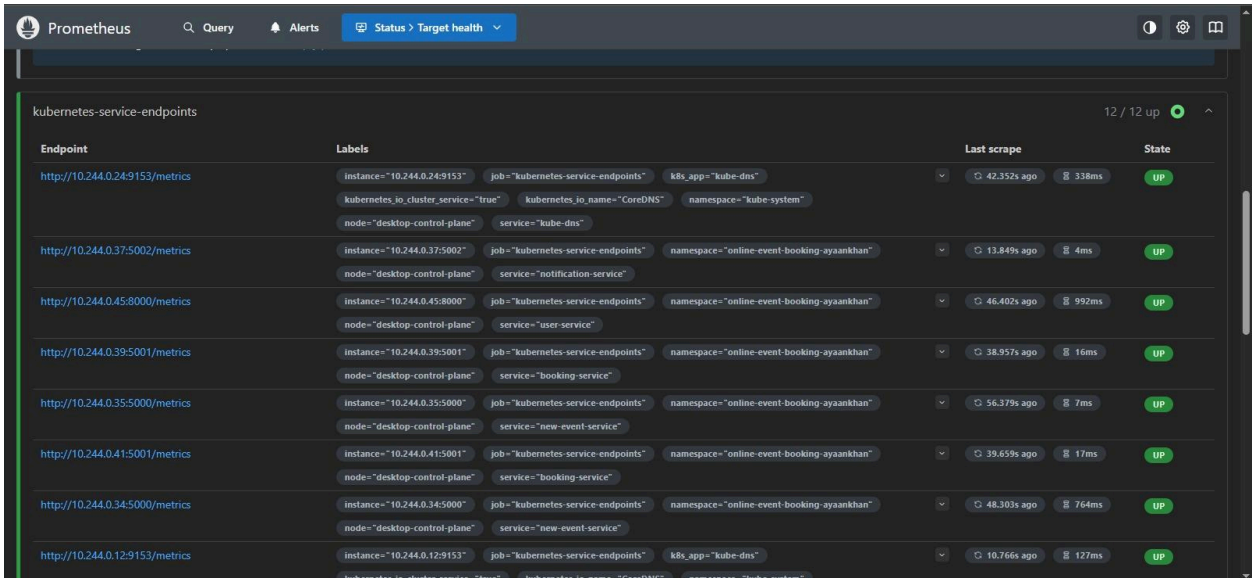


3.8 Prometheus

Role: Monitoring and alerting for Kubernetes and microservices.

Implementation:

- Deployed as part of the monitoring stack via Helm.
- Collects metrics from microservices, Kubernetes nodes, and pods.

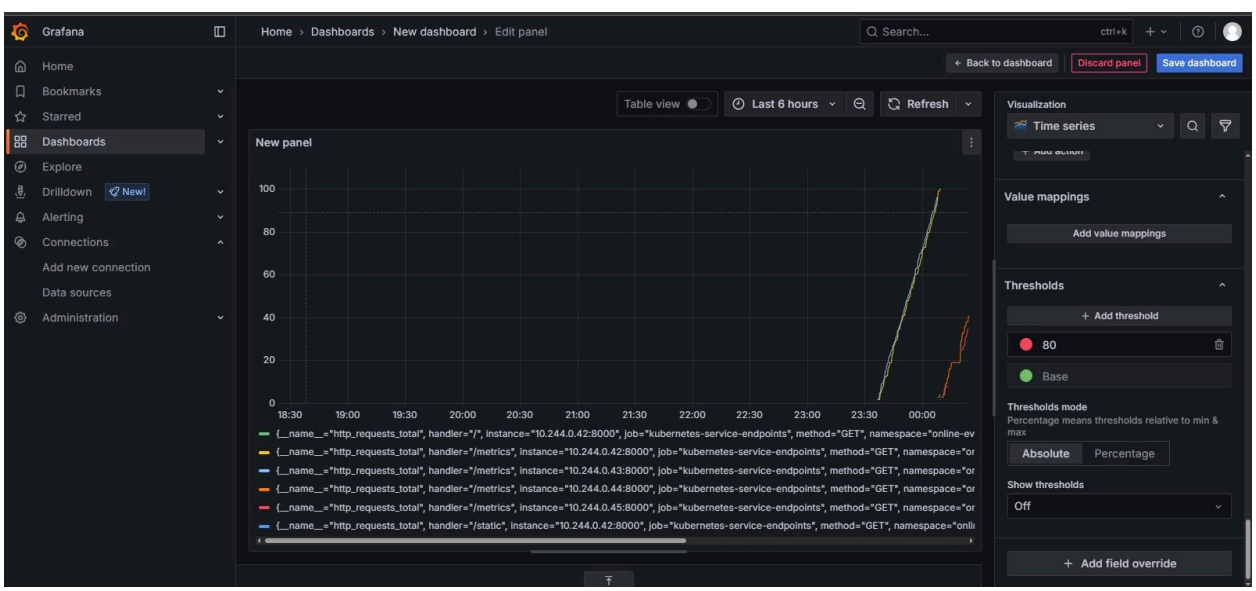


3.9 Grafana

Role: Visualization of system metrics.

Implementation:

- Integrated with Prometheus as a data source.
- Dashboards created for tracking data



4. Workflow Overview

Code Commit:

Developers push changes to the GitHub repository.

CI Pipeline (GitHub Actions):

Test code.

Builds and pushes Docker images to a registry.

Infrastructure Setup (Terraform):

Provisions cloud resources (e.g., Kubernetes cluster).

Configuration (Ansible):

Configures nodes and installs dependencies.

CD Pipeline (Argo CD):

Automatically deploys Kubernetes manifests to the cluster.

Monitoring Setup (Prometheus + Grafana):

- Prometheus and Grafana scrapes metrics from services and cluster components.
- Grafana visualizes these metrics with real-time dashboards.

5. Challenges and Solutions

Challenge	Solution
Complex orchestration	Kubernetes + Argo CD GitOps model streamlined deployments.

Environment drift	Terraform ensured consistent, reproducible infrastructure.
CI/CD debugging	GitHub Actions logs improved troubleshooting and traceability.
Secret management	Kubernetes Secrets and GitHub repo secrets were integrated securely.
Monitoring and observability	Prometheus and Grafana provided deep visibility into system performance and uptime.

6. Conclusion

The platform demonstrates a fully automated DevOps pipeline for microservices. By integrating tools like Terraform, Ansible, Docker, Kubernetes, Argo CD, Prometheus, and Grafana, the solution achieves scalability, reliability, observability, and rapid deployment.

The GitOps approach ensures declarative, version-controlled infrastructure, while the monitoring stack helps proactively detect and resolve issues.