

Implementation of Docker and DevOps Principles in Django-React Web Project

Reference Paper - <https://ieeexplore.ieee.org/document/10117715>

Team Members

- Sachchida Nand Tiwari (M23CSA527)
- Balakrishna Kariveda (M23CSA511)
- Ritesh Lamba (M23CSA544)



Introduction

- Implements principles from the IEEE paper on Docker and DevOps.
- Focus: Scalable and maintainable deployment for a Django-React web app.
- Tools: Docker, Kubernetes, Prometheus/Grafana.



System Overview

Backend: Django + PostgreSQL.

Frontend: React.

Containerization: Docker for all components.

Orchestration: Kubernetes for scalability and high availability.



Alignment with Paper

Aspect	Paper Recommendations	Our Implementation
Containerization	Use Docker	Docker for Django, React, PostgreSQL
Microservices	Modularize services	Separate containers



Alignment with Paper

Aspect	Paper Recommendations	Our Implementation
Orchestration	Use Kubernetes for cluster management	Kubernetes for scalability
High Availability	Redundancy, self-healing	Kubernetes manages failovers



Key Features

Containerized Deployment:

- Django, React, PostgreSQL in isolated Docker environments.

Kubernetes Orchestration:

- Ensures availability and resource optimization.

Load Balancing:

- Kubernetes ingress for routing and load handling.



Benefits Achieved

- **Scalability:** Adjusts to workload changes dynamically.
- **Reliability:** Redundancy and Kubernetes self-healing.
- **Efficiency:** Optimized resource usage.
- **Observability:** Actionable insights via monitoring tools.



Kubernetes Configurations Overview

File	Purpose
namespace.yaml	Isolates CRS resources.
django-config.yaml	Configures Django settings.
django-deployment.yaml	Deploys Django backend.
postgres-pv.yaml	Defines PostgreSQL storage.
ingress.yaml	Routes frontend/backend traffic.



Django Configuration

django-config.yaml: ConfigMap for environment variables.

django-secret.yaml: Secret for database credentials.

django-deployment.yaml: Deploys backend container on port 8000.



PostgreSQL Configuration

postgres-pv.yaml: PersistentVolume for database storage.

postgres-secret.yaml: Secret for PostgreSQL credentials.

postgres-deployment.yaml: Deploys PostgreSQL container on port 5432.



React Configuration

react-deployment.yaml: Deploys frontend container.

react-service.yaml: Exposes React app on port 3000.

ingress.yaml: Manages traffic routing for React and Django.



Screenshots- Containers

Containers [Give feedback](#)

Container CPU usage ⓘ

0.00% / 1200% (12 CPUs available)





Container memory usage ⓘ

0B / 3.57GB

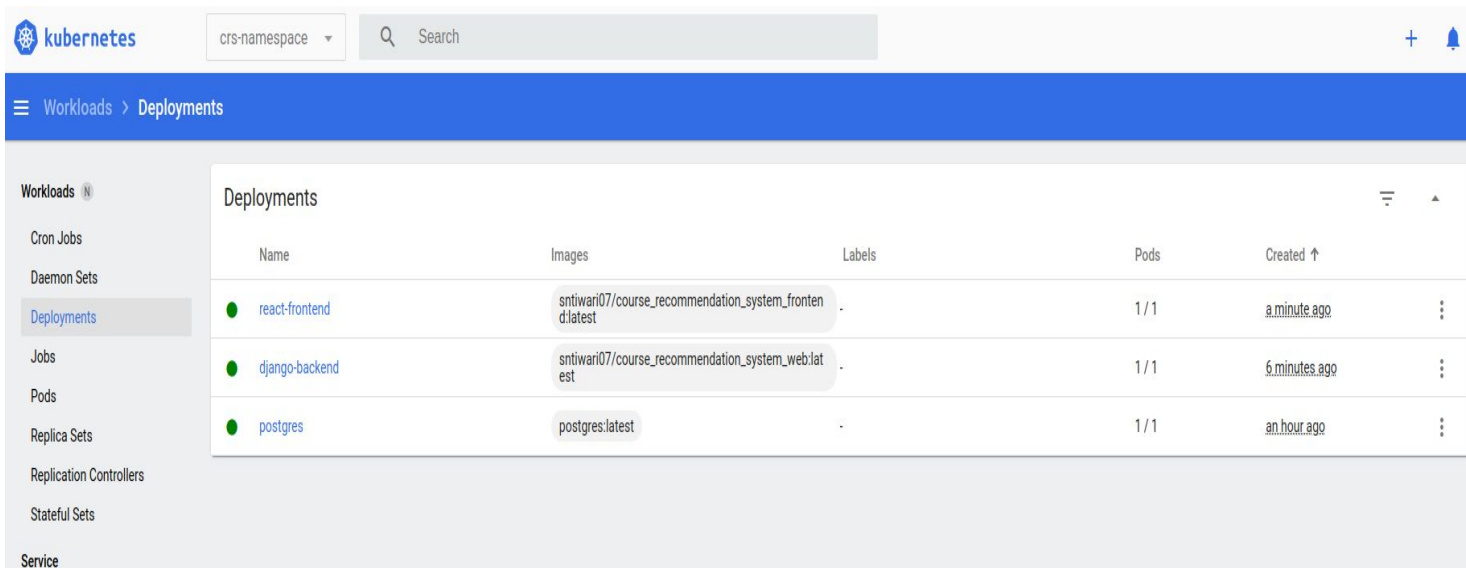
[Show charts](#)



☒ Only show running containers

<input type="checkbox"/>	Name	Image	Status	Port(s)	CPU (%)	Last started	Actions
<input type="checkbox"/>	 minikube f5ae23a30f7d 	kicbase/stable:v0.0.45	Running	43563:22  Show all ports (5)	0%	27 seconds ago	  

Screenshots- Deployments



The screenshot displays the Kubernetes dashboard interface. At the top, the 'kubernetes' logo is on the left, followed by a dropdown menu set to 'crs-namespace' and a search bar. On the right of the top bar are a plus icon and a bell icon. Below this is a blue header bar with a hamburger menu icon and the text 'Workloads > Deployments'. A left-hand sidebar lists various workload types: 'Workloads' (with a count of 11), 'Cron Jobs', 'Daemon Sets', 'Deployments' (highlighted in blue), 'Jobs', 'Pods', 'Replica Sets', 'Replication Controllers', 'Stateful Sets', and 'Service'. The main content area is titled 'Deployments' and contains a table with the following data:

Name	Images	Labels	Pods	Created ↑
● react-frontend	sntiwari07/course_recommendation_system_frontend:latest	-	1 / 1	a minute ago
● django-backend	sntiwari07/course_recommendation_system_web:latest	-	1 / 1	6 minutes ago
● postgres	postgres:latest	-	1 / 1	an hour ago


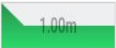



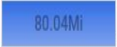





Screenshots- Ingresses

Ingresses					
Name	Labels	Endpoints ↗	Hosts ↗	Created ↑	
crs-ingress	-	192.168.58.2	localhost	2 hours ago	⋮






Screenshots- Pods

Pods									
Name	Images	Labels	Node	Status	Restarts	CPU Usage (cores)	Memory Usage (bytes)	Created ↑	
 react-frontend-667f564cd6-q79hn	sntiwari07/course_recommen dation_system_frontend:latest	app: react-frontend pod-template-hash: 667f564cd6	minikube	Running	1	 1.00m	 513.90Mi	9 minutes ago	⋮
 django-backend-6ccd46bcb-lskmq	sntiwari07/course_recommen dation_system_web:latest	app: django-backend pod-template-hash: 6ccd46bcb	minikube	Running	1	 19.00m	 80.04Mi	15 minutes ago	⋮
 postgres-5b57587dbf-cjd8g	postgres:latest	app: postgres pod-template-hash: 5b57587dbf	minikube	Running	6	 1.00m	 34.78Mi	2 hours ago	⋮



Screenshots- Replica Sets

Replica Sets					
Name	Images	Labels	Pods	Created ↑	
 react-frontend-667f564cd6	sntiwari07/course_recommendation_system_frontend:latest	app: react-frontend pod-template-hash: 667f564cd6	1 / 1	10 minutes ago	⋮
 django-backend-6ccd46bcbb	sntiwari07/course_recommendation_system_web:latest	app: django-backend pod-template-hash: 6ccd46bcbb	1 / 1	16 minutes ago	⋮
 postgres-5b57587dbf	postgres:latest	app: postgres pod-template-hash: 5b57587dbf	1 / 1	2 hours ago	⋮



Screenshots - Scale

Scale a resource

deployment react-frontend will be updated to reflect the desired replicas count.

Desired replicas *

Actual replicas

1|



1



This action is equivalent to: `kubectl scale -n crs-namespace deployment react-frontend --replicas=1`

Scale

Cancel



Conclusion

- Integrated Docker, Kubernetes, and DevOps principles.
- Achieved a scalable, efficient, and maintainable architecture.
- Monitoring ensures proactive issue resolution.



Thank You!