Document Clustering System with Docker

Technical Mathematics for Big Data

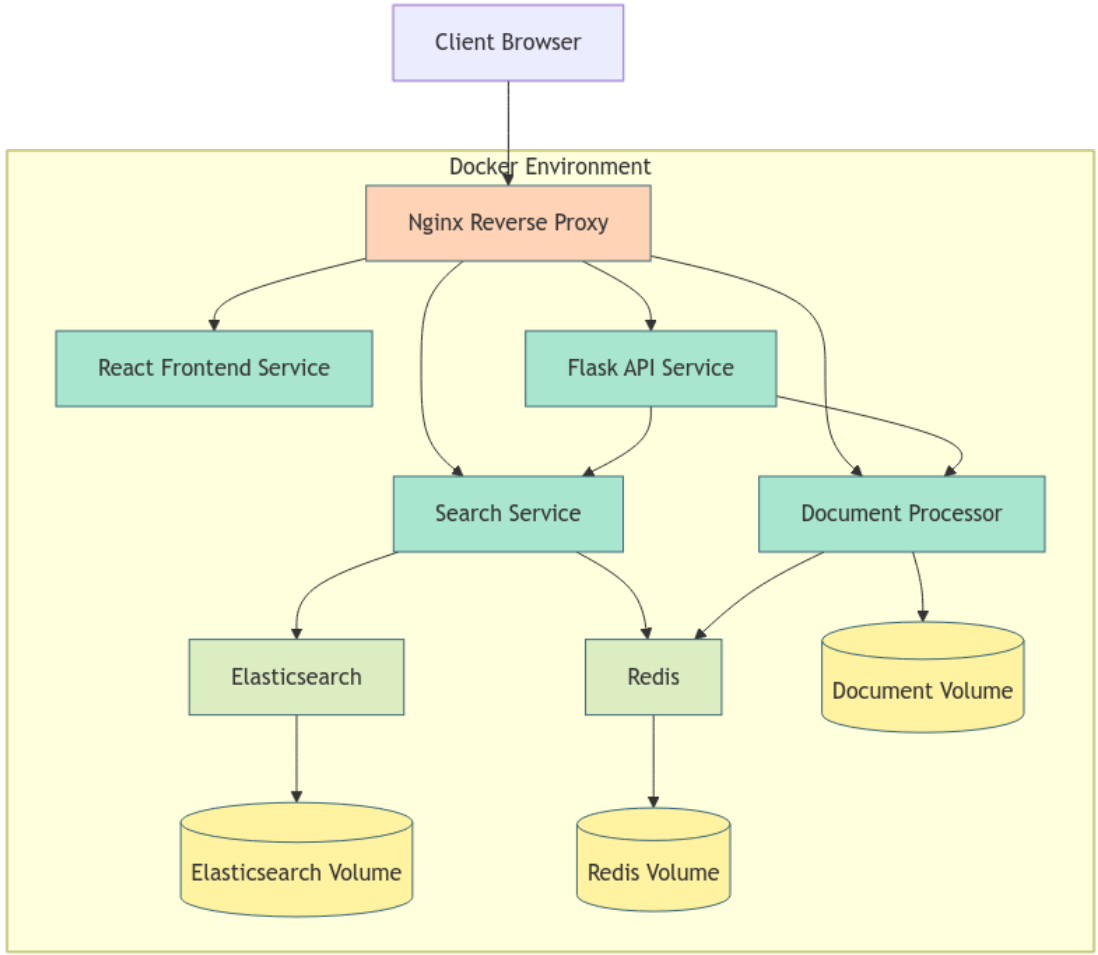
Oyedotun Oluwasegun Michael (#123168)
Silvia Mastracci (#123177)
Oleksandr Solovei (#126784)

January 16, 2025

Project Overview

- Document Clustering System built with microservices
- Containerized solution using Docker technology

- React-based UI
- Flask microservices
- Elasticsearch engine
- Document analysis system



Docker Setup and Operations

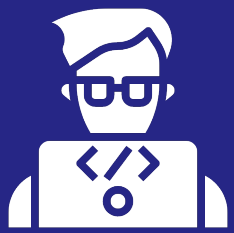
Docker Compose Config

```
services:
  nginx:
    image: nginx:alpine
    ports:
      - "4321:80"
  frontend:
    build: ./frontend
    expose:
      - "3000"
  api:
    build: ./api
    expose:
      - "8000"
  document-processor: (...)
```

Common Commands

```
# Build and start
docker-compose up --build

# Stop services
docker-compose down

# View logs
docker-compose logs -f

# Rebuild specific service
docker-compose build svc-name

# Show running containers
docker-compose ps
```

Docker Benefits



For Development

- **Consistent Environment**
  no "works on my machine"
- **Isolated Deps**
  no version conflicts
- **Rapid Dev Cycle**
  fast startup, easy rollbacks

For Performance

- **Resource Efficiency**
  lightweight architecture
- **Scalability**
  horizontal scaling
- **Maintenance**
  simple updates, min downtime

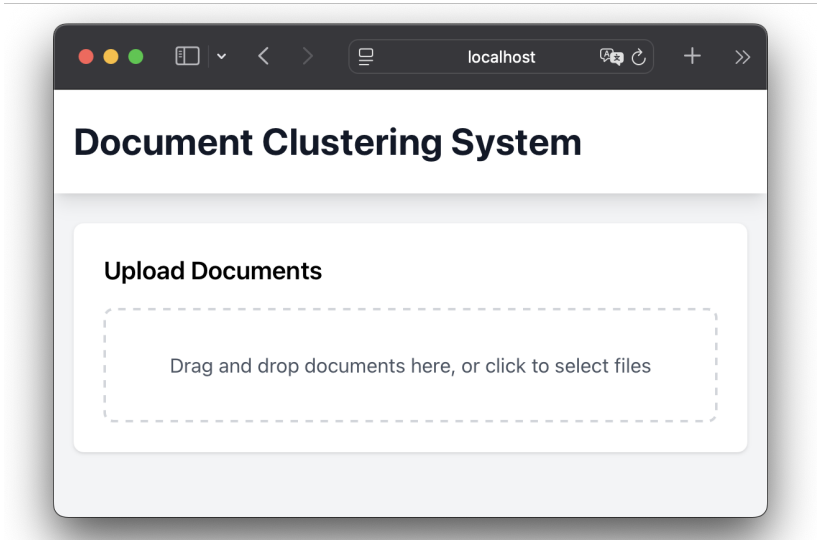
For Security

- **Vulnerability Management**
  image scanning, regular updates
- **Instance Isolation**
  for processes and networks
- **Security Features**
  access & resource limits

Project Implementation and System Features

- **Multi-stage builds**
  - Optimized image sizes
  - Reduced attack surface
- **Docker Compose**
  - Service orchestration
  - Environment configuration
  - Network management
- **Volume Management**
  - Persistent data storage
  - Efficient data sharing

	Name	Tag	Image ID
●	redis	alpine	1bf97f21f01b
●	docker.elastic.co/elasticsearch/elasticsearch	7.17.9	59b37f77bd8b
●	nginx	alpine	814a8e88df97
●	docker-project-search	latest	4f5fbb64e1de
●	docker-project-document-processor	latest	d8329a9c19bc
●	docker-project-api	latest	1806fbbc4c1a



- **Single Port Access**
  - All services through one port
  - Nginx reverse proxy
- **Monitoring**
  - Health checks
  - Automated recovery
- **Data Management**
  - Elasticsearch integration
  - Redis caching

Managing Local and Production Environments

Local Development: Docker Compose

- Use `docker-compose.yml` for local setup.
- Benefits: ease of testing and configuration.
- Example commands:
  - `docker-compose up --build`
  - `docker-compose down`
  - Add `docker-compose.override.yml` for environment-specific changes.

Production: Docker Swarm

- Convert `docker-compose.yml` to Swarm using `docker stack deploy`.
- Enable high availability and service replication.
- Commands:
  - `docker swarm init`
  - `docker stack deploy -c docker-compose.yml <stack_name>`

Parallelizable Text Processing Tasks

High-Load Document Processing Tasks

- **OCR (Optical Character Recognition)**
  - Divide documents into pages or segments.
  - Use Tesseract with multiprocessing.
- **Translations**
  - Batch translations across multiple documents.
  - Use APIs supporting parallel requests (e.g., Google Translate API, DeepL).
- **Natural Language Processing (NLP)**
  - Parallel topic modeling or sentiment analysis.
- **Parallelization Frameworks**
  - Celery with RabbitMQ or Redis.
  - Python's `multiprocessing`, `joblib` for batch processing.

Cloud Deployment with Blue-Green Strategy

Cloud Platforms

- **AWS:** Elastic Beanstalk, ECS/EKS for container orchestration.
- **Azure:** App Service, AKS for Kubernetes clusters.
- **DigitalOcean:** App Platform or Kubernetes droplets.

Blue-Green Deployment

- Keep two environments (blue = live, green = staging).
- Traffic shift via load balancer after validation.
- Benefits: zero-downtime upgrades and rollback safety.

Kubernetes with Docker Containers

Why Kubernetes?

- Advanced orchestration and scaling compared to Docker Swarm.
- Features: self-healing, automated rollouts.

Key Components

- Pods, Deployments, Services, ConfigMaps, and Ingress.

Migration Process

- Convert Docker Compose to Kubernetes manifests (`kubectl apply -f`).
- Use Helm charts for templating.
- Tools: Minikube for local testing, `kubectl` for CLI control.

Modern Docker-Based Approach vs Legacy System

Legacy Approach

- Monolithic deployments.
- Manual scaling and maintenance.
- Infrastructure tightly coupled.

Docker-Based Approach

- Microservices with independent scaling.
- Automated deployments and CI/CD.
- Isolated dependencies improve maintainability.
- Faster development cycles and simplified rollbacks.

Conclusion

Thank You

Document Clustering System
Docker-based Microservices Architecture

Questions?