Deployment Guide

Overview

This deployment guide provides detailed steps for deploying the AutoSync Cloud Service across multiple environments, including development, staging, and production. The system is containerized using Docker and orchestrated via Kubernetes for scalability and resilience.

---

1. Deployment Environments

1.1 Local Development Environment

- Used for development and debugging by individual developers.
- Runs all components locally using Docker Compose.

1.2 Staging Environment

- Mimics the production environment for testing.
- Deployed on a cloud provider (AWS/GCP/Azure) using Kubernetes.

1.3 Production Environment

- Fully scalable and monitored deployment hosted in the cloud with high availability and disaster recovery enabled.

---

2. Prerequisites

2.1 Hardware Requirements

Environment	CPU	RAM	Storage
Development	4 cores	16 GB	50 GB SSD
Staging	8 cores	32 GB	200 GB SSD
Production	16 cores	64 GB	1 TB SSD

2.2 Software Requirements

- **Operating System:** Linux (Ubuntu 22.04 recommended).
- **Docker:** Version 20.10 or later.
- **Kubernetes:** Version 1.24 or later.
- **Helm:** Version 3.10 or later.
- **kubectl:** Kubernetes command-line tool.

---

3. Local Deployment Using Docker Compose

3.1 Steps to Set Up Local Environment

1. **Clone the Repository:**

```
1 bash
```

Copy code

`git clone https://github.com/example/autosync.git cd autosync`

2. **Configure Environment Variables:**
   - Create a `.env` file in the root directory.
   - Add the following entries:

     ```
     1 env
     ```

     Copy code

     `DATABASE_URL=postgresql://user:password@localhost:5432/autosync REDIS_URL=redis://localhost:6379 KAFKA_BROKER=kafka://localhost:9092`

3. **Run Docker Compose:**

```
1 bash
```

Copy code

`docker-compose up --build`

4. **Verify Services:**
   - Visit `http://localhost:3000` for the frontend.
   - Check logs to ensure services are running.

---

4. Kubernetes Deployment

4.1 Create a Kubernetes Cluster

- For **AWS**, use **EKS (Elastic Kubernetes Service):**

```
1 bash
```

Copy code

`eksctl create cluster --name autosync-cluster --region us-west-2`

- For **Google Cloud**, use **GKE (Google Kubernetes Engine):**

```
1 bash
```

Copy code

`gcloud container clusters create autosync-cluster --zone us-central1-a`

4.2 Set Up Namespace and Secrets

1. **Create a Namespace:**

```
1 bash
```

Copy code

`kubectl create namespace autosync`

2. **Add Secrets for Environment Variables:**

```
1 bash
```

Copy code

```
kubectl create secret generic autosync-secrets \ --from-literal=DATABASE_URL=postgresql://user:password@db:5432/autosync \ --from-literal=REDIS_URL=redis://redis-service:6379
```

4.3 Deploy with Helm

1. **Install Helm Chart:**
   - Navigate to the Helm directory in the repository.
   - Run the following command:

```bash
bash
```

Copy code

```
helm install autosync ./helm/autosync --namespace autosync
```

2. **Verify Deployment:**

```bash
bash
```

Copy code

```
kubectl get pods -n autosync kubectl get services -n autosync
```

---

5. Monitoring and Scaling

5.1 Enable Horizontal Pod Autoscaling (HPA):

- Scale services automatically based on CPU or memory utilization:

```bash
bash
```

Copy code

```
kubectl autoscale deployment autosync-backend --cpu-percent=50 --min=2 --max=10
```

5.2 Monitoring with Prometheus and Grafana:

- **Prometheus** for metrics collection.
- **Grafana** for visualization.

Deploy the monitoring stack using Helm:

```bash
bash
```

Copy code

```
helm install monitoring prometheus-community/kube-prometheus-stack --namespace monitoring
```

---

6. Backup and Disaster Recovery

6.1 Backup Strategies:

1. **Database:**
   - Schedule daily backups with tools like pgBackRest for PostgreSQL.
   - Example command:

```
1 bash
```

Copy code

```
pg_dump -U user -h db autosync > backup.sql
```

2. **Storage:**
   - Enable versioning on AWS S3 buckets.
   - Automate backups with AWS Backup service.

6.2 Disaster Recovery:

- Set up multi-region deployment with active-passive failover.
- Use AWS Route 53 for automatic failover routing.

---

7. Deployment Flowchart

(Below is a flowchart visualizing the deployment process.)