AVGC Tender Management System - Kubernetes Deployment

Overview

This directory contains all the Kubernetes manifests required to deploy the AVGC Tender Management Platform on a Kubernetes cluster.

Prerequisites

- Kubernetes cluster (v1.21+)
- kubectl configured to access your cluster
- Helm (for cert-manager and ingress-nginx)
- Docker registry with built images
- SSL certificates (or cert-manager for automatic certificate management)

Architecture Components

- Frontend: React-based web application
- API Gateway: Kong/Nginx-based API gateway
- Microservices:
 - Auth Service
 - Tender Service
 - Document Service
 - EMD Management Service
 - Security Management Service
 - Reporting Service
 - Notification Service

Databases:

- PostgreSQL (for relational data)
- MongoDB (for document storage)
- Redis (for caching and sessions)
- Elasticsearch (for search and analytics)
- Message Queue: RabbitMQ
- Monitoring: Prometheus + Grafana

Deployment Steps

1. Install Prerequisites

bash

Install NGINX Ingress Controller

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.8.0/deploy/static/provider/

Install cert-manager for SSL certificates

kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.12.0/cert-manager.yaml

2. Create Namespace

bash

kubectl apply -f namespace.yaml

3. Update Secrets

Before deploying, update the base64 encoded values in (secrets.yaml):

bash

Encode your secrets

echo -n "your-actual-secret" | base64

Edit secrets.yaml with your encoded values

4. Deploy Using Kustomize

bash

Deploy all resources

kubectl apply -k.

Or deploy individual components

kubectl apply -f postgres-deployment.yaml

kubectl apply -f mongodb-deployment.yaml

... continue for other services

5. Verify Deployment

```
bash

# Check all pods are running
kubectl get pods -n avgc-tender-system

# Check services
kubectl get svc -n avgc-tender-system

# Check ingress
kubectl get ingress -n avgc-tender-system
```

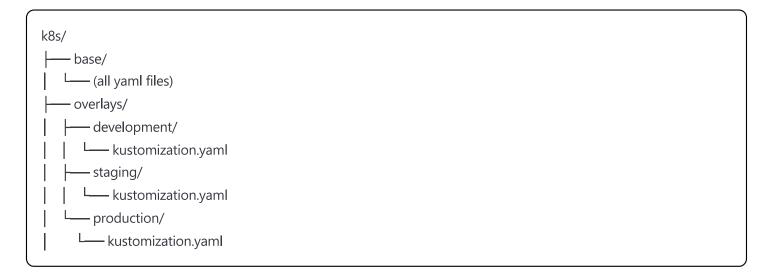
6. Access the Application

- Frontend: https://avgc-tenders.com
- API: https://api.avgc-tenders.com
- WebSocket: wss://ws.avgc-tenders.com
- Grafana: https://grafana.avgc-tenders.com (internal)

Configuration

Environment-Specific Configurations

Create overlays for different environments:



Resource Limits

Adjust resource limits based on your cluster capacity in each deployment file.

Scaling

To scale services:

bash

kubectl scale deployment/frontend --replicas=5 -n avgc-tender-system kubectl scale deployment/api-gateway --replicas=5 -n avgc-tender-system

Database Migrations

Run database migrations before starting services:

bash

kubectl run migration --image=avgc/migration-job:latest --restart=Never -n avgc-tender-system

Monitoring

Access Prometheus

bash

kubectl port-forward svc/prometheus-service 9090:9090 -n avgc-tender-system

Access Grafana

bash

kubectl port-forward svc/grafana-service 3000:3000 -n avgc-tender-system

Default credentials: admin/adminpassword

Troubleshooting

Check Logs

```
bash

# Pod logs
kubectl logs <pod-name> -n avgc-tender-system

# Previous pod logs
kubectl logs <pod-name> -n avgc-tender-system --previous

# Follow logs
kubectl logs -f <pod-name> -n avgc-tender-system
```

Debug Pods

```
bash

# Describe pod

kubectl describe pod <pod-name> -n avgc-tender-system

# Execute into pod

kubectl exec -it <pod-name> -n avgc-tender-system -- /bin/sh
```

Common Issues

- 1. **ImagePullBackOff**: Check image names and registry credentials
- 2. **CrashLoopBackOff**: Check pod logs for startup errors
- 3. **Pending PVCs**: Ensure storage class is available
- 4. **Service Discovery**: Verify service names and ports

Backup and Recovery

Database Backups

```
# PostgreSQL backup
kubectl exec -it postgres-0 -n avgc-tender-system -- pg_dump -U postgres > backup.sql
# MongoDB backup
kubectl exec -it mongodb-0 -n avgc-tender-system -- mongodump --out=/tmp/backup
```

Restore Procedures

Documented in the disaster recovery plan (separate document).

Security Considerations

- All secrets are stored in Kubernetes secrets (consider using sealed-secrets or external secret managers)
- Network policies restrict inter-pod communication
- RBAC limits service account permissions
- SSL/TLS enabled for all external communications
- Regular security updates for base images

Maintenance

- Regular updates of base images
- Monitor resource usage and adjust limits
- Review and rotate secrets periodically
- Keep Kubernetes cluster updated