NexusTrade SuperApp - Deployment Guide

Version: 1.0.0

Last Updated: October 14, 2025

Table of Contents

Pre-Deployment Checklist

Infrastructure Requirements

- [ ] Kubernetes cluster (1.29+) with 5 PoPs configured
- [ ] Load balancers configured for each PoP
- [ ] DNS configured with Geo-DNS routing
- [ ] SSL/TLS certificates issued
- [ ] Secrets management (Vault) configured
- [ ] Container registry accessible
- [ ] CI/CD pipeline configured

Access & Credentials

- [ ] AWS/Azure/GCP credentials configured
- [ ] Kubernetes cluster access verified
- [ ] Container registry credentials set up
- [ ] Database master passwords generated
- [ ] API keys for third-party services obtained
- [ ] Vault unsealed and accessible

Dependencies

- [ ] Docker images built and pushed
- [ ] Helm charts validated
- [ ] Database schemas prepared
- [ ] Configuration files reviewed
- [ ] Environment variables documented

Infrastructure Setup

1. Provision Kubernetes Clusters

Option A: Using Terraform (Recommended)

```bash
cd infrastructure/terraform/aws

# Initialize Terraform
terraform init

# Review plan
terraform plan -var-file=production.tfvars

# Apply configuration
terraform apply -var-file=production.tfvars

# Repeat for all 5 PoPs
```

Option B: Manual Provisioning

Lagos (Primary PoP):

```bash
# Create EKS cluster
eksctl create cluster \
  --name nexustrade-lagos \
  --version 1.29 \
  --region af-south-1 \
  --nodegroup-name standard-workers \
  --node-type t3.xlarge \
  --nodes 3 \
  --nodes-min 3 \
  --nodes-max 10 \
  --managed
```

Repeat for:

- Johannesburg: `af-south-1`
- London: `eu-west-2`
- Ashburn: `us-east-1`
- Singapore: `ap-southeast-1`

2. Configure kubectl Contexts

```bash
# Add contexts for all PoPs
kubectl config use-context nexustrade-lagos
kubectl config use-context nexustrade-joburg
kubectl config use-context nexustrade-london
kubectl config use-context nexustrade-ashburn
kubectl config use-context nexustrade-singapore

# Verify access
kubectl get nodes --all-namespaces
```

3. Install Core Components

```bash
# Install Helm
curl https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3 | bash

# Add Helm repositories
helm repo add prometheus-community https://prometheus-community.github.io/helm-charts
helm repo add grafana https://grafana.github.io/helm-charts
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo update

# Install cert-manager
kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.13.0/cert-manager.yaml

# Install ingress-nginx
helm install ingress-nginx ingress-nginx/ingress-nginx \
  --namespace ingress-nginx \
  --create-namespace \
  --set controller.service.type=LoadBalancer
```

Database Configuration

1. Deploy YugabyteDB Cluster

```bash
# Create namespace
kubectl create namespace nexustrade

# Add YugabyteDB Helm repo
helm repo add yugabytedb https://charts.yugabyte.com

# Install YugabyteDB
helm install yugabyte yugabytedb/yugabyte \
  --namespace nexustrade \
  --set resource.master.requests.cpu=2 \
  --set resource.master.requests.memory=4Gi \
  --set resource.tserver.requests.cpu=4 \
  --set resource.tserver.requests.memory=8Gi \
  --set replicas.master=3 \
  --set replicas.tserver=3 \
  --set gflags.tserver.ysql_enable_auth=true \
  --set gflags.master.ysql_enable_auth=true
```

2. Initialize YugabyteDB Schema

```bash
# Get YugabyteDB pod
YUGABYTE_POD=$(kubectl get pods -n nexustrade -l app=yb-tserver -o jsonpath='{.items[0].metadata.name}')

# Copy schema file
kubectl cp shared/prisma/schema.sql nexustrade/$YUGABYTE_POD:/tmp/

# Execute schema
kubectl exec -it -n nexustrade $YUGABYTE_POD -- ysqlsh -h yb-tserver-0.yb-tservers.nexustrade -U yugabyte -d yugab
```

3. Deploy ScyllaDB Cluster

bash

```
# Add ScyllaDB operator
kubectl apply -f https://github.com/scylladb/scylla-operator/releases/download/v1.10.0/install.yaml

# Deploy ScyllaDB cluster
kubectl apply -f - <<EOF
apiVersion: scylla.scylladb.com/v1
kind: ScyllaCluster
metadata:
  name: nexustrade-scylla
  namespace: nexustrade
spec:
  version: 5.4.0
  datacenter:
    name: lagos
    racks:
    - name: rack1
      members: 3
      storage:
        capacity: 500Gi
        storageClassName: fast-ssd
      resources:
        requests:
          cpu: 4
          memory: 16Gi
        limits:
          cpu: 8
          memory: 32Gi
EOF

# Wait for cluster to be ready
kubectl wait --for=condition=Ready scyllacluster/nexustrade-scylla -n nexustrade --timeout=10m
```

4. Initialize ScyllaDB Schema

```bash
# Get ScyllaDB pod
SCYLLA_POD=$(kubectl get pods -n nexustrade -l app=scylla -o jsonpath='{.items[0].metadata.name}')

# Copy schema
kubectl cp shared/scylladb/schema.cql nexustrade/$SCYLLA_POD:/tmp/

# Execute schema
kubectl exec -it -n nexustrade $SCYLLA_POD -- cqlsh -f /tmp/schema.cql
```

5. Deploy DragonflyDB

```bash
helm install dragonfly oci://ghcr.io/dragonflydb/dragonfly/helm/dragonfly \
  --namespace nexustrade \
  --set replicas=3 \
  --set resources.requests.memory=4Gi \
  --set resources.requests.cpu=2 \
  --set persistence.enabled=true \
  --set persistence.size=50Gi
```

6. Deploy MinIO

```bash
helm install minio bitnami/minio \
  --namespace nexustrade \
  --set mode=distributed \
  --set statefulset.replicaCount=4 \
  --set persistence.size=500Gi \
  --set auth.rootUser=nexustrade \
  --set auth.rootPassword=$(openssl rand -base64 32)
```

7. Deploy Kafka

```bash
helm install kafka bitnami/kafka \
  --namespace nexustrade \
  --set replicaCount=3 \
  --set persistence.size=100Gi \
  --set zookeeper.enabled=true \
  --set zookeeper.replicaCount=3
```

Service Deployment

1. Create Secrets

```bash
# Generate secrets
export JWT_SECRET=$(openssl rand -base64 64)
export JWT_REFRESH_SECRET=$(openssl rand -base64 64)
export DATABASE_PASSWORD=$(openssl rand -base64 32)

# Create Kubernetes secrets
kubectl create secret generic nexustrade-secrets \
  --from-literal=database-url="postgresql://nexustrade:${DATABASE_PASSWORD}@yugabytedb:5433/nexustrade" \
  --from-literal=jwt-secret="$JWT_SECRET" \
  --from-literal=jwt-refresh-secret="$JWT_REFRESH_SECRET" \
  --namespace nexustrade

# Store in Vault
vault kv put secret/nexustrade/production \
  database-url="postgresql://nexustrade:${DATABASE_PASSWORD}@yugabytedb:5433/nexustrade" \
  jwt-secret="$JWT_SECRET" \
  jwt-refresh-secret="$JWT_REFRESH_SECRET"
```

2. Deploy Keycloak

bash

```
helm install keycloak bitnami/keycloak \
  --namespace nexustrade \
  --set auth.adminUser=admin \
  --set auth.adminPassword=$(openssl rand -base64 32) \
  --set production=true \
  --set proxy=edge \
  --set postgresql.enabled=true
```

3. Deploy HashiCorp Vault

bash

```
helm install vault hashicorp/vault \
  --namespace nexustrade \
  --set server.ha.enabled=true \
  --set server.ha.replicas=3 \
  --set ui.enabled=true

# Initialize Vault
kubectl exec -it vault-0 -n nexustrade -- vault operator init

# Unseal Vault (repeat for all pods)
kubectl exec -it vault-0 -n nexustrade -- vault operator unseal <unseal-key-1>
kubectl exec -it vault-0 -n nexustrade -- vault operator unseal <unseal-key-2>
kubectl exec -it vault-0 -n nexustrade -- vault operator unseal <unseal-key-3>
```

4. Deploy Real-Time Services

bash

```
# MongooseIM
kubectl apply -f infrastructure/kubernetes/mongooseim-deployment.yaml

# Janus Gateway
kubectl apply -f infrastructure/kubernetes/janus-deployment.yaml

# coturn
kubectl apply -f infrastructure/kubernetes/coturn-deployment.yaml

# Kamailio
kubectl apply -f infrastructure/kubernetes/kamailio-deployment.yaml
```

5. Deploy Microservices

```bash
# API Gateway
kubectl apply -f infrastructure/kubernetes/api-gateway-deployment.yaml

# Trading Service
kubectl apply -f infrastructure/kubernetes/trading-deployment.yaml

# Bot Manager
kubectl apply -f infrastructure/kubernetes/bot-manager-deployment.yaml

# Bot Sidecar (Python)
kubectl apply -f infrastructure/kubernetes/bot-sidecar-deployment.yaml
```

6. Verify Deployments

```bash
# Check all pods
kubectl get pods -n nexustrade

# Check services
kubectl get svc -n nexustrade

# Check ingress
kubectl get ingress -n nexustrade

# View logs
kubectl logs -f deployment/api-gateway -n nexustrade
```

Network Configuration

1. Configure Ingress

```bash
kubectl apply -f - <<EOF
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nexustrade-ingress
  namespace: nexustrade
  annotations:
    cert-manager.io/cluster-issuer: letsencrypt-prod
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
    nginx.ingress.kubernetes.io/force-ssl-redirect: "true"
spec:
  ingressClassName: nginx
  tls:
  - hosts:
    - api.nexustrade.com
    secretName: nexustrade-tls
  rules:
  - host: api.nexustrade.com
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: api-gateway
            port:
              number: 80
EOF
```

2. Configure DNS

```bash
# Get LoadBalancer IP
LB_IP=$(kubectl get svc ingress-nginx-controller -n ingress-nginx -o jsonpath='{.status.loadBalancer.ingress[0].ip}')

# Update DNS records
# - api.nexustrade.com → $LB_IP
# - ws.nexustrade.com → $LB_IP
# - turn.nexustrade.com → TURN server IP
```

3. Configure Geo-DNS

Use your DNS provider's Geo-DNS features to route traffic:

- Africa → Lagos/Johannesburg
- Europe → London
- Americas → Ashburn
- Asia Pacific → Singapore

Security Hardening

1. Network Policies

```bash
# Apply network policies
kubectl apply -f infrastructure/kubernetes/network-policies/
```

2. Pod Security Policies

bash

```
kubectl apply -f - <<EOF
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: restricted
spec:
  privileged: false
  allowPrivilegeEscalation: false
  requiredDropCapabilities:
    - ALL
  volumes:
    - 'configMap'
    - 'emptyDir'
    - 'projected'
    - 'secret'
    - 'downwardAPI'
    - 'persistentVolumeClaim'
  runAsUser:
    rule: 'MustRunAsNonRoot'
  seLinux:
    rule: 'RunAsAny'
  fsGroup:
    rule: 'RunAsAny'
EOF
```

3. Enable mTLS

bash

```
# Install Istio service mesh (optional but recommended)
istioctl install --set profile=production

# Enable mTLS
kubectl apply -f - <<EOF
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
  name: default
  namespace: nexustrade
spec:
  mtls:
    mode: STRICT
EOF
```

Monitoring Setup

1. Deploy Prometheus

bash

```
helm install prometheus prometheus-community/kube-prometheus-stack \
  --namespace monitoring \
  --create-namespace \
  --set prometheus.prometheusSpec.retention=30d \
  --set prometheus.prometheusSpec.storageSpec.volumeClaimTemplate.spec.resources.requests.storage=100Gi
```

2. Deploy Grafana

bash

```
# Already included in kube-prometheus-stack
# Get Grafana password
kubectl get secret -n monitoring prometheus-grafana -o jsonpath="{.data.admin-password}" | base64 --decode
```

3. Deploy Loki

```bash
helm install loki grafana/loki-stack \
  --namespace monitoring \
  --set loki.persistence.enabled=true \
  --set loki.persistence.size=100Gi
```

4. Deploy Tempo

```bash
helm install tempo grafana/tempo \
  --namespace monitoring \
  --set tempo.persistence.enabled=true \
  --set tempo.persistence.size=50Gi
```

Post-Deployment Verification

1. Health Checks

```bash
# API Gateway health
curl https://api.nexustrade.com/health

# Database connectivity
kubectl exec -it -n nexustrade deployment/api-gateway -- npm run db:check

# Redis connectivity
kubectl exec -it -n nexustrade deployment/api-gateway -- npm run redis:check
```

2. Smoke Tests

```bash
# Run smoke tests
kubectl apply -f tests/smoke-tests.yaml

# View results
kubectl logs -f job/smoke-tests -n nexustrade
```

3. Load Testing

```bash
# Run load tests
k6 run tests/load-test.js
```

Rollback Procedures

Option 1: Helm Rollback

```bash
# List releases
helm list -n nexustrade

# Rollback to previous version
helm rollback api-gateway -n nexustrade

# Rollback to specific revision
helm rollback api-gateway 3 -n nexustrade
```

Option 2: Kubernetes Rollback

```bash
# View deployment history
kubectl rollout history deployment/api-gateway -n nexustrade

# Rollback to previous version
kubectl rollout undo deployment/api-gateway -n nexustrade

# Rollback to specific revision
kubectl rollout undo deployment/api-gateway --to-revision=2 -n nexustrade
```

Troubleshooting

Common Issues

1. Pods in CrashLoopBackOff

```bash
# View logs
kubectl logs -f <pod-name> -n nexustrade

# Describe pod
kubectl describe pod <pod-name> -n nexustrade

# Check events
kubectl get events -n nexustrade --sort-by='.lastTimestamp'
```

2. Service Not Accessible

```bash
# Check service
kubectl get svc <service-name> -n nexustrade

# Test from within cluster
kubectl run -it --rm debug --image=busybox --restart=Never -n nexustrade -- wget -O- http://<service-name>:port
```

3. Database Connection Issues

```bash
# Test connection
kubectl exec -it -n nexustrade <pod-name> -- psql -h yugabytedb -U nexustrade -d nexustrade -c "SELECT 1"
```

4. High Memory Usage

```bash
# Check resource usage
kubectl top pods -n nexustrade

# Increase resource limits
kubectl patch deployment api-gateway -n nexustrade -p '{"spec":{"template":{"spec":{"containers":[{"name":"api-gatew
```

Support & Escalation

Support Channels

- **Critical (P0):** Page on-call engineer
- **High (P1):** Email: ops@nexustrade.com
- **Medium (P2):** Slack: #nexustrade-ops
- **Low (P3):** Ticket: support.nexustrade.com

Runbooks

- Database failover: `docs/runbooks/database-failover.md`
- Service degradation: `docs/runbooks/service-degradation.md`
- Security incident: `docs/runbooks/security-incident.md`

Deployment completed successfully! 🎉

Monitor the system for 24 hours before declaring production-ready.