DevOps Expert Kubernetes Deployment Checklist

Interactive Deployment Questionnaire

Application Architecture & Requirements

1. Application Type & Scale

- [ ] **Web Application** (Frontend + Backend)
- [ ] **API Service** (Backend only)
- [ ] **Full-Stack Monolith**
- [ ] **Microservices Architecture**

Auto-generates: Service mesh requirements, ingress configuration

2. Expected Traffic & Scaling

- [ ] **Low Traffic** (<1K users/day) → t3.micro instances
- [ ] **Medium Traffic** (1K-100K users/day) → t3.small/medium + HPA
- [ ] **High Traffic** (>100K users/day) → Multi-AZ + Cluster Autoscaler
- [ ] **Variable/Spiky Traffic** → KEDA + Spot instances

Auto-generates: Resource limits, HPA configs, node groups

3. Database Requirements

- [ ] **No Database** → Skip DB setup
- [ ] **SQL Database** → RDS PostgreSQL/MySQL + connection pooling
- [ ] **NoSQL Database** → DocumentDB/DynamoDB
- [ ] **Cache Layer** → ElastiCache Redis
- [ ] **In-Memory Database** → Redis on K8s

Auto-generates: Database terraform, secrets management, backup strategies

Security & Compliance

4. Security Posture

- [ ] **Basic Security** → Network policies, basic RBAC
- [ ] **Enhanced Security** → Pod Security Standards, OPA Gatekeeper
- [ ] **Enterprise Security** → Falco, Admission controllers, Image scanning
- [ ] **Compliance Required** (SOC2/HIPAA) → Full audit logging, encryption

Auto-generates: Security policies, network policies, admission controllers

5. Secrets Management

- [ ] **Kubernetes Secrets** → Basic secret management
- [ ] **External Secrets Operator** → AWS Secrets Manager integration
- [ ] **HashiCorp Vault** → Full secrets lifecycle management

Auto-generates: Secret management setup, rotation policies

6. SSL/TLS Configuration

- [ ] **Let's Encrypt** → Cert-manager with ACME
- [ ] **AWS Certificate Manager** → ALB integration
- [ ] **Custom Certificates** → Manual cert management

Auto-generates: Certificate management, ingress TLS configuration

Infrastructure & Networking

7. Cluster Architecture

- [ ] **Single Region** → Standard EKS cluster
- [ ] **Multi-AZ** → Cross-AZ deployment
- [ ] **Multi-Region** → Regional clusters + cross-region replication
- [ ] **Hybrid Cloud** → On-premises + cloud integration

Auto-generates: Cluster terraform, networking setup, disaster recovery

8. Networking Requirements

- [ ] **Public Internet Access** → ALB + public subnets
- [ ] **Private with VPN** → Private subnets + VPN gateway
- [ ] **CDN Required** → CloudFront integration
- [ ] **API Gateway** → AWS API Gateway + service mesh

Auto-generates: VPC setup, subnets, security groups, ingress controllers

9. Storage Requirements

- [ ] **No Persistent Storage** → EmptyDir volumes only
- [ ] **File Storage** → EFS integration
- [ ] **Block Storage** → EBS CSI driver
- [ ] **Object Storage** → S3 integration

Auto-generates: Storage classes, PV provisioning, backup strategies

Monitoring & Operations

10. Observability Level

- [ ] **Basic Monitoring** → CloudWatch + basic dashboards
- [ ] **Standard Observability** → Prometheus + Grafana + AlertManager
- [ ] **Full Observability** → ELK/EFK stack + distributed tracing
- [ ] **Enterprise Monitoring** → DataDog/New Relic integration

Auto-generates: Monitoring stack, dashboards, alerting rules

11. Logging Strategy

- [ ] **Container Logs Only** → kubectl logs access
- [ ] **Centralized Logging** → FluentBit + CloudWatch/S3
- [ ] **Log Analytics** → ELK stack with Kibana
- [ ] **Structured Logging** → JSON logs + log parsing

Auto-generates: Logging infrastructure, log retention policies

12. Backup & Disaster Recovery

- [ ] **No Backups** → Stateless application
- [ ] **Basic Backups** → EBS snapshots + DB backups
- [ ] **Application-Level Backups** → Velero + cross-region replication
- [ ] **Full DR Strategy** → Multi-region setup + automated failover

Auto-generates: Backup schedules, DR procedures, RTO/RPO configs

Development & Deployment

13. CI/CD Integration

- [ ] **GitHub Actions** → GHA workflows + OIDC
- [ ] **GitLab CI** → GitLab runners on K8s
- [ ] **Jenkins** → Jenkins on K8s + pipeline templates
- [ ] **AWS CodePipeline** → Native AWS CI/CD

Auto-generates: CI/CD pipelines, runner setup, deployment workflows

14. GitOps Configuration

- [ ] **ArgoCD** → GitOps with ArgoCD + app-of-apps pattern
- [ ] **FluxCD** → GitOps with Flux v2
- [ ] **Manual Deployment** → kubectl-based deployment

- [ ] **Helm-based** → Helm charts + Helmfile

Auto-generates: GitOps setup, application definitions, sync policies

15. Environment Strategy

- [ ] **Single Environment** → Production only
- [ ] **Dev/Prod Split** → Separate namespaces
- [ ] **Multi-Environment** → Dev/Staging/Prod clusters
- [ ] **Preview Environments** → Dynamic environments per PR

Auto-generates: Environment configs, promotion workflows, resource quotas

Cost & Resource Management

16. Cost Optimization

- [ ] **Cost-Aware** → Resource limits + Spot instances
- [ ] **Budget-Constrained** → Aggressive scaling + smaller instances
- [ ] **Performance-First** → Larger instances + dedicated nodes
- [ ] **Balanced Approach** → Mixed instance types + intelligent scaling

Auto-generates: Resource requests/limits, node selectors, cost monitoring

17. Resource Management

- [ ] **Basic Resources** → Simple requests/limits
- [ ] **Resource Quotas** → Namespace-level quotas
- [ ] **Priority Classes** → Workload prioritization
- [ ] **Vertical Pod Autoscaler** → Automatic resource optimization

Auto-generates: Resource policies, LimitRanges, VPA configurations

Generated Infrastructure Components

Based on checklist selections, the system auto-generates:

1. Terraform Modules

```
├── terraform/
│   ├── aws/
│   │   ├── eks-cluster/
│   │   ├── vpc/
│   │   ├── rds/
│   │   ├── s3/
│   │   └── iam/
│   ├── modules/
│   └── environments/
│       ├── dev/
│       ├── staging/
│       └── prod/
```

2. Kubernetes Manifests

```
├── k8s/
│   ├── base/
│   │   ├── deployment.yaml
│   │   ├── service.yaml
│   │   ├── ingress.yaml
│   │   └── configmap.yaml
│   ├── overlays/
│   │   ├── dev/
│   │   ├── staging/
│   │   └── prod/
│   └── helm-charts/
```

3. GitOps Repository Structure

```
├── gitops/
│   ├── applications/
│   │   ├── app-of-apps.yaml
│   │   └── environments/
│   ├── infrastructure/
│   │   ├── monitoring/
│   │   ├── ingress/
│   │   └── security/
│   └── configs/
```

4. CI/CD Pipelines

```
├── .github/workflows/
│   ├── build-and-test.yml
│   ├── deploy-to-staging.yml
│   ├── deploy-to-prod.yml
│   └── infrastructure.yml
```

5. Monitoring & Observability

```
├── monitoring/
│   ├── prometheus/
│   │   ├── values.yaml
│   │   └── alerts/
│   ├── grafana/
│   │   └── dashboards/
│   └── logging/
│       └── fluentbit/
```

One-Click Deployment Process

1. **Generate Infrastructure Code** → All Terraform, K8s manifests, CI/CD
2. **Create GitOps Repository** → Initialize with generated configs
3. **Setup GitHub Actions** → Configure OIDC, secrets, workflows
4. **Deploy Infrastructure** → Terraform apply for AWS resources
5. **Bootstrap ArgoCD** → Install and configure GitOps
6. **Deploy Application** → ArgoCD syncs from GitOps repo
7. **Configure Monitoring** → Deploy observability stack
8. **Run Health Checks** → Verify all components

Result: Production-ready Kubernetes deployment in ~15 minutes