To achieve a **High Availability (HA), minimal downtime Nexus IQ Server deployment** on **AWS EKS**, using **RDS Aurora PostgreSQL** and **EFS** for persistent storage, while provisioning and managing everything through **Terraform** and **Helm**, here’s a comprehensive strategy.

This approach emphasizes **minimal downtime** (instead of zero-downtime) by balancing complexity with high reliability and recoverability.

**🧱 Core Infrastructure Components**

| **Layer** | **Technology** |
| --- | --- |
| **Platform** | AWS EKS (Multi-AZ worker nodes) |
| **App Deploy** | Helm via Terraform |
| **Storage** | EFS for IQ data (optional) |
| **Database** | RDS Aurora PostgreSQL (Multi-AZ, encrypted) |
| **State Mgmt** | Terraform (EKS, EFS, RDS, Helm) |
| **Secrets** | AWS Secrets Manager + IRSA |
| **Backup** | Automated RDS Snapshots + EFS AWS Backup |

**✅ Key Goals**

* Minimize downtime during upgrades, scale events, and recoveries
* Avoid complex blue-green duplication (no dual Helm releases)
* Enable fast rollbacks using automated EFS + RDS backups
* Ensure state consistency and high resilience

**⚙️ Deployment Strategy (Minimal Downtime)**

**1. Use Rolling Updates in Helm Deployment**

Nexus IQ supports a rolling update pattern with zero-pod-downtime if you configure readiness/liveness probes correctly.

yaml

CopyEdit

strategy:

type: RollingUpdate

readinessProbe:

httpGet:

path: /healthcheck

port: 8071

livenessProbe:

httpGet:

path: /healthcheck

port: 8071

**2. Use Helm for App Deployment (via Terraform)**

hcl

CopyEdit

resource "helm\_release" "nexus\_iq" {

name = "nexus-iq"

chart = "nexus-iq-server"

repository = "https://sonatype.github.io/helm-nexus-iq-server"

namespace = "nexus"

values = [

file("${path.module}/values.yaml")

]

}

**3. RDS Aurora PostgreSQL with Backups and PITR**

h

CopyEdit

module "aurora" {

source = "terraform-aws-modules/rds-aurora/aws"

name = "nexus-iq-db"

engine = "aurora-postgresql"

backup\_retention\_period = 7

preferred\_backup\_window = "03:00-04:00"

storage\_encrypted = true

enable\_http\_endpoint = true

}

**🔁 PITR and Snapshot Recovery**

* **Pre-deployment**: snapshot creation
* **Failure recovery**: restore from snapshot or PITR to new cluster and re-point app

**4. EFS for Shared Storage (if needed)**

Provision EFS via Terraform:

hcl

CopyEdit

resource "aws\_efs\_file\_system" "nexus\_efs" {

encrypted = true

lifecycle\_policy {

transition\_to\_ia = "AFTER\_30\_DAYS"

}

}

resource "aws\_efs\_mount\_target" "nexus\_mount" {

for\_each = toset(data.aws\_subnets.eks.ids)

file\_system\_id = aws\_efs\_file\_system.nexus\_efs.id

subnet\_id = each.value

security\_groups = [aws\_security\_group.efs.id]

}

Mount in values.yaml:

yaml

CopyEdit

volumeClaimTemplates:

- metadata:

name: nexus-data

spec:

accessModes: [ "ReadWriteMany" ]

storageClassName: "efs-sc"

resources:

requests:

storage: 50Gi

**5. Backups**

**✅ RDS Aurora**

* Enabled via Terraform
* Daily automated snapshots
* PITR (Point-in-Time Recovery) support
* Optional: manual snapshots before upgrades

**✅ EFS Backup**

Enable **AWS Backup** for EFS:

hcl

CopyEdit

resource "aws\_backup\_plan" "efs\_plan" {

name = "nexus-efs-backup"

rule {

rule\_name = "daily-backup"

target\_vault\_name = aws\_backup\_vault.nexus.name

schedule = "cron(0 5 \* \* ? \*)" # Daily at 5AM

lifecycle {

delete\_after = 30

}

}

}

resource "aws\_backup\_selection" "efs\_selection" {

iam\_role\_arn = aws\_iam\_role.backup\_role.arn

name = "nexus-efs-selection"

plan\_id = aws\_backup\_plan.efs\_plan.id

resources = [

aws\_efs\_file\_system.nexus\_efs.arn

]

}

**🔐 Secrets Management (via AWS Secrets Manager + IRSA)**

* Store DB and admin credentials in Secrets Manager
* Use **IRSA** (IAM Role for Service Account) to inject into the app

**🧪 Recovery and Rollback Plan**

| **Component** | **Failure** | **Recovery Strategy** |
| --- | --- | --- |
| App | Bad deployment | helm rollback nexus-iq <REVISION> |
| RDS | Corruption | Restore from snapshot → update DB host in Helm values |
| EFS | Data loss | Restore volume via AWS Backup |
| Full | Infra loss | Re-deploy from Terraform → restore RDS/EFS from backup |

**📝 values.yaml Highlights**

yaml

CopyEdit

replicaCount: 2

externalDatabase:

host: nexus-iq-db.cluster-xxxxx.us-east-1.rds.amazonaws.com

name: iq

user: iqadmin

password: ${DB\_PASSWORD}

persistence:

enabled: true

storageClass: "efs-sc"

accessMode: ReadWriteMany

size: 50Gi

**✅ Summary**

| **Capability** | **Status** |
| --- | --- |
| HA EKS with Nexus IQ | ✅ |
| Minimal downtime rolling upgrades | ✅ |
| Helm + Terraform automation | ✅ |
| RDS Aurora with PITR and backups | ✅ |
| EFS persistent volume with backups | ✅ |
| Rollback strategy for app and DB | ✅ |
| Secrets managed via AWS Secrets/IAM | ✅ |

**➕ Optional Next Steps**

Would you like generated:

* Terraform module for RDS + EFS backup
* values.yaml for Nexus IQ with EFS + RDS
* helm rollback and restore-db scripts
* GitHub Actions for backup + restore testing

Let me know what part you want to automate next.

4o

**You said:**

Nexus IQ HA minimal downtime deployment strategy AWS EKS, RDS Aurora, EFS, Helm, terraform, EFS and RDS backups blue-green

**ChatGPT said:**

To implement a **High Availability (HA), minimal downtime Nexus IQ Server deployment** on **AWS EKS** using **RDS Aurora PostgreSQL**, **EFS**, **Terraform**, **Helm**, and **EFS/RDS backups**, **combined with a blue-green strategy**, here is a complete deployment and rollback plan optimized for **resilience, safety, and automation**.

**🚀 Objective**

| **Goal** | **Approach** |
| --- | --- |
| HA & Scalability | EKS (Multi-AZ), replica sets |
| Persistent, shared storage | EFS with ReadWriteMany access |
| Scalable relational database | Aurora PostgreSQL with Multi-AZ & PITR |
| Zero/Mimimal Downtime Deployments | Blue-Green deployment using Helm and Ingress |
| Safe rollback | Helm rollback + restore EFS/RDS from snapshot |
| Automation | Terraform-managed infra, pre-deploy backups, IRSA + Secrets Manager |

**⚙️ Architecture Overview**

java

CopyEdit

┌───────────────────────────┐

│ AWS Route 53 │

└────────────┬──────────────┘

│

┌──────────────▼─────────────┐

│ AWS ALB Ingress │

└────┬────────┬────────┬─────┘

│ │ │

┌─────────────▼──┐ ┌───▼────────▼────┐

│ nexus-iq-blue │ │ nexus-iq-green │ ← new deployment

└─────▲──────────┘ └─────▲───────────┘

│ │

EFS PVC EFS PVC

│ │

┌───▼────────────┐ ┌────▼─────────────┐

│ AWS EFS (RWX) │ │ AWS EFS (clone\*) │ ← if testing new vol

└────────────────┘ └──────────────────┘

│

┌──────▼────────┐

│ RDS Aurora │ ← shared or cloned DB (blue vs green optional)

└───────────────┘

**🧩 Components**

**✅ EKS Setup (via Terraform)**

* Multi-AZ NodeGroups
* IRSA for Secrets + EFS

**✅ RDS Aurora PostgreSQL (Terraform)**

* Multi-AZ cluster
* Automated backups (7+ days)
* PITR enabled

hcl

CopyEdit

backup\_retention\_period = 7

enable\_http\_endpoint = true

**✅ EFS (Terraform)**

* Mounted with efs-sc StorageClass
* Used by both blue and green for shared or cloned data

**✅ Nexus IQ Helm Values**

**Blue:**

yaml

CopyEdit

nameOverride: nexus-iq-blue

externalDatabase:

host: nexus-iq-db.cluster-xxxx.rds.amazonaws.com

persistence:

storageClass: efs-sc

accessMode: ReadWriteMany

**Green:**

yaml

CopyEdit

nameOverride: nexus-iq-green

externalDatabase:

host: nexus-iq-db.cluster-xxxx.rds.amazonaws.com # or new cluster for test

persistence:

storageClass: efs-sc

accessMode: ReadWriteMany

**🔁 Blue-Green Deployment Strategy (Minimal Downtime)**

**1. Pre-deployment Steps**

* Take **RDS snapshot** (Terraform or CLI)
* Optionally clone EFS for staging (efs-utils, or AWS Backup restore)
* Deploy nexus-iq-green using Helm with same (or cloned) RDS/EFS
* Health check green via preview subdomain or path

**2. Cutover Traffic**

Use AWS ALB Ingress Controller annotations or update Route 53 DNS to point to green service.

bash

CopyEdit

kubectl patch ingress nexus-iq \

--type='json' \

-p='[{"op": "replace", "path": "/spec/rules/0/http/paths/0/backend/service/name", "value":"nexus-iq-green"}]'

**3. Rollback (if needed)**

* Revert Ingress to nexus-iq-blue
* Restore RDS from snapshot and re-point green to old DB
* Restore EFS via AWS Backup if corruption/data loss

**🔁 RDS & EFS Backup Strategy**

**✅ RDS**

* **Daily snapshots**
* **Manual snapshot before deployment**
* **PITR enabled** for sub-hour rollback

bash

CopyEdit

aws rds create-db-cluster-snapshot \

--db-cluster-identifier nexus-iq-db \

--db-cluster-snapshot-identifier nexus-iq-predeploy-$(date +%s)

To restore:

bash

CopyEdit

aws rds restore-db-cluster-from-snapshot \

--snapshot-identifier nexus-iq-predeploy-<timestamp> \

--db-cluster-identifier nexus-iq-db-rollback

**✅ EFS**

Use **AWS Backup** for scheduled and on-demand backups.

hcl

CopyEdit

resource "aws\_backup\_plan" "efs\_plan" {

name = "nexus-iq-efs-backup"

rule {

rule\_name = "daily"

schedule = "cron(0 3 \* \* ? \*)"

lifecycle { delete\_after = 30 }

}

}

To restore:

bash

CopyEdit

aws backup start-restore-job \

--recovery-point-arn arn:aws:backup:... \

--resource-type EFS \

--iam-role-arn arn:aws:iam::... \

--metadata file-system-id=fs-xxxx

**🔐 Secrets and Configuration**

Use **AWS Secrets Manager** + IRSA to inject secrets securely into the Helm release.

Example:

yaml

CopyEdit

externalDatabase:

user: {{ .Values.secrets.db.user }}

password: {{ .Values.secrets.db.password }}

**🧪 CI/CD Steps (Optional)**

| **Stage** | **Action** |
| --- | --- |
| pre-deploy | Snapshot RDS + backup EFS |
| deploy | Helm upgrade nexus-iq-green |
| verify | Health checks on /healthcheck |
| cutover | Switch Ingress or DNS to green |
| rollback | Revert Ingress + restore RDS/EFS if needed |

Use GitHub Actions or ArgoCD hooks to automate this flow.

**✅ Summary**

| **Feature** | **Enabled** |
| --- | --- |
| HA Nexus IQ on EKS | ✅ |
| Minimal downtime via blue-green | ✅ |
| Shared persistent storage via EFS | ✅ |
| Resilient RDS with PITR & snapshots | ✅ |
| Terraform automation for all resources | ✅ |
| Helm deploy with rollback | ✅ |
| AWS Backup support for EFS & RDS | ✅ |
| Secure secrets via IRSA & SecretsManager | ✅ |

**📦 Want Terraform Modules or Helm Templates?**

I can generate:

* ✅ Terraform code for RDS, EFS, Backup, EKS, IAM
* ✅ Helm values for blue/green
* ✅ CI/CD scripts for backup + cutover
* ✅ Ingress switch scripts

Let me know which you'd like first.