**1. Objective**

This document details the migration of **Nexus Repository Manager (NexusRM)** from an **AWS EC2-based environment with EFS and PostgreSQL** to a **Highly Available (HA) AWS EKS** environment. This migration aims to improve scalability, availability, and operational efficiency.

**2. Migration Goals**

* **High Availability:** Multi-node NexusRM setup with auto-scaling.
* **Data Integrity:** Maintain consistent data between EC2 and EKS.
* **Operational Efficiency:** Leverage Kubernetes for automated management.
* **Performance:** Improved resilience and scalability through EKS.
* **Seamless Transition:** Minimal downtime and comprehensive rollback plan.

**3. Current Architecture (EC2-based)**

| **Component** | **Details** |
| --- | --- |
| Compute | Single/multi-node EC2 instances |
| Storage | Amazon EFS for persistent data |
| Database | Amazon RDS PostgreSQL (Multi-AZ) |
| Backup | EFS snapshots + RDS automated backups |
| Load Balancer | ELB with Route 53 |
| Monitoring | CloudWatch |

**4. Target Architecture (EKS-based)**

| **Component** | **Details** |
| --- | --- |
| Compute | AWS EKS with Managed Node Groups |
| Storage | Amazon EFS (existing, mounted on EKS) |
| Database | Amazon RDS PostgreSQL (existing) |
| Ingress | AWS ALB via Ingress Controller |
| Auto-scaling | HPA for NexusRM Pods, Cluster Autoscaler |
| Load Balancer | ALB integrated with Route 53 |
| Monitoring | Prometheus, Grafana, CloudWatch |

**5. Architecture Diagram**

lua

CopyEdit

+----------------------+

| Route 53 |

+----------+------------+

|

+------+------+

| AWS ALB |

+------+------+

|

+---------------------+---------------------+

| |

+-------v-------+ +-------v-------+

| NexusRM Pod 1 | <--> Shared EFS <--> | NexusRM Pod 2 |

+---------------+ +---------------+

| |

+--------+--------+ +-----------+---+

| RDS PostgreSQL | | EFS (HA) |

+-----------------+ +---------------+

**6. Migration Strategy**

**6.1 Approach**

* **Blue-Green Deployment:** Run both EC2 and EKS simultaneously during migration.
* **Data Synchronization:** EFS remains the same, ensuring no data loss.
* **Minimal Downtime:** DNS cutover during non-peak hours.
* **Rollback Ready:** Retain EC2 instances until full validation.

**7. Pre-Migration Checklist**

* Backup NexusRM data (EFS and PostgreSQL).
* Validate EFS mount availability from EKS nodes.
* Test Helm charts in a staging environment.
* Verify access from EKS to RDS PostgreSQL.
* Document current configurations and dependencies.

**8. Infrastructure Setup (Terraform)**

**8.1 EKS Cluster Configuration**

hcl

CopyEdit

module "eks" {

source = "terraform-aws-modules/eks/aws"

cluster\_name = "nexusrm-cluster"

cluster\_version = "1.29"

vpc\_id = module.vpc.vpc\_id

subnet\_ids = module.vpc.private\_subnets

node\_groups = {

default = {

desired\_capacity = 3

max\_capacity = 6

instance\_type = "m6i.xlarge"

}

}

}

**8.2 RDS PostgreSQL**

Reuse existing RDS instance:

hcl

CopyEdit

resource "aws\_db\_instance" "nexusrm" {

engine = "postgres"

instance\_class = "db.t3.medium"

multi\_az = true

allocated\_storage = 50

username = "nexus\_user"

password = var.db\_password

}

**8.3 EFS Mount on EKS**

hcl

CopyEdit

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

encrypted = true

}

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

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

subnet\_id = module.vpc.private\_subnets[0]

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

}

**9. Helm Configuration for NexusRM**

**9.1 Helm Values (values.yaml)**

yaml

CopyEdit

replicaCount: 3

persistence:

enabled: true

storageClass: "efs-sc"

accessMode: ReadWriteMany

size: 100Gi

env:

- name: NEXUS\_DATA

value: /nexus-data

- name: DATABASE\_URL

value: jdbc:postgresql://nexusrm-db.xxxxxx.rds.amazonaws.com:5432/nexusrm

ingress:

enabled: true

annotations:

alb.ingress.kubernetes.io/scheme: internet-facing

hosts:

- host: nexus.example.com

paths:

- path: /

pathType: Prefix

**9.2 Deploy NexusRM with Helm**

bash

CopyEdit

helm repo add sonatype https://sonatype.github.io/helm3-charts/

helm upgrade --install nexusrm sonatype/nexus-repository-manager \

--namespace nexusrm --create-namespace \

-f values.yaml

**10. Data Synchronization**

**10.1 Sync Data from EC2 to EKS**

bash

CopyEdit

rsync -avz /nexus-data/ efs-mount:/nexus-data/

**10.2 Validate Data**

* Check file consistency on both mounts.
* Validate database connectivity from the EKS pods.

**11. Testing & Validation**

* Validate repository functions (upload/download).
* Test failover by stopping a pod.
* Check DNS cutover and ALB routing.
* Simulate high load to test autoscaling.
* Monitor EFS performance metrics.

**12. DNS Cutover**

* Update **Route 53** to point to the new ALB endpoint.
* Monitor access logs to confirm traffic redirection.

**13. Monitoring and Alerts**

| **Metric** | **Tool** | **Action** |
| --- | --- | --- |
| Pod CPU/Memory | Prometheus | Scale Up/Down |
| DB Connection Errors | Grafana Alerts | Investigate RDS |
| EFS Performance | CloudWatch | Scale IOPS |
| Pod Restarts | Grafana Alerts | Debug Issues |

**14. Rollback Plan**

* Revert DNS to EC2-based ELB.
* Synchronize recent data from EKS to EC2.
* Restore any configuration changes made during the migration.
* Investigate root cause before retrying the migration.

**15. Success Criteria**

* Zero data loss and minimal downtime.
* Seamless user access after DNS cutover.
* Automatic scaling under workload changes.
* Effective monitoring with proactive alerting.

**📑 NexusRM HA Deployment on AWS EKS with Autoscaling**

**1. Objective**

This document outlines the deployment of **Nexus Repository Manager (NexusRM)** on **AWS EKS** with a focus on **High Availability (HA)** and **Autoscaling**. The architecture ensures resilience, scalability, and minimal downtime.

**2. Deployment Objectives**

* **High Availability:** Deploy NexusRM in a multi-node setup.
* **Scalability:** Leverage Kubernetes Horizontal Pod Autoscaler (HPA).
* **Performance:** Optimize response time during high traffic.
* **Resilience:** Automatically recover from failures.
* **Operational Efficiency:** Manage infrastructure via Helm and Terraform.

**3. Architecture Overview**

| **Component** | **Description** |
| --- | --- |
| Compute | AWS EKS with Managed Node Groups |
| Persistent Storage | Amazon EFS (for shared repository data) |
| Database | Amazon RDS (PostgreSQL) for metadata storage |
| Ingress | AWS ALB via ALB Ingress Controller |
| Autoscaling | Kubernetes HPA and Cluster Autoscaler |
| Monitoring & Logging | Prometheus, Grafana, CloudWatch |

**4. Architecture Diagram**

lua

CopyEdit

+----------------------+

| Route 53 |

+----------+------------+

|

+------+------+

| AWS ALB |

+------+------+

|

+----------------------+----------------------+

| |

+------+-------+ +----------+------+

| NexusRM Pod1 | <-- Shared EFS --> | NexusRM Pod2 |

+--------------+ +-----------------+

| |

+------v-----+ +-------v-----+

| RDS (Postgres) | | Amazon EFS |

+---------------+ +--------------+

**5. Prerequisites**

* **AWS CLI** and **kubectl** configured.
* **Helm** v3 installed.
* **Terraform** setup for infrastructure management.
* **AWS IAM Role** for EKS with necessary policies.

**6. Infrastructure Provisioning (Terraform)**

**6.1 AWS EKS Cluster**

hcl

CopyEdit

module "eks" {

source = "terraform-aws-modules/eks/aws"

cluster\_name = "nexusrm-cluster"

cluster\_version = "1.29"

vpc\_id = module.vpc.vpc\_id

subnet\_ids = module.vpc.private\_subnets

node\_groups = {

default = {

desired\_capacity = 3

max\_capacity = 6

instance\_type = "m6i.xlarge"

}

}

}

**6.2 EFS for Nexus Data**

hcl

CopyEdit

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

encrypted = true

}

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

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

subnet\_id = module.vpc.private\_subnets[0]

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

}

**6.3 RDS PostgreSQL Database**

hcl

CopyEdit

resource "aws\_db\_instance" "nexusrm" {

engine = "postgres"

instance\_class = "db.t3.medium"

allocated\_storage = 20

multi\_az = true

username = "nexus\_user"

password = var.db\_password

storage\_encrypted = true

}

**7. Helm Chart Configuration**

**7.1 Helm Chart Values (values.yaml)**

yaml

CopyEdit

replicaCount: 3

persistence:

enabled: true

storageClass: "efs-sc"

accessMode: ReadWriteMany

size: 100Gi

resources:

requests:

memory: "2Gi"

cpu: "500m"

limits:

memory: "4Gi"

cpu: "1"

env:

- name: DATABASE\_URL

value: jdbc:postgresql://nexusrm-db.xxxxxx.rds.amazonaws.com:5432/nexusrm

- name: NEXUS\_DATA

value: /nexus-data

autoscaling:

enabled: true

minReplicas: 3

maxReplicas: 10

metrics:

- type: Resource

resource:

name: cpu

target:

type: Utilization

averageUtilization: 70

ingress:

enabled: true

annotations:

alb.ingress.kubernetes.io/scheme: internet-facing

hosts:

- host: nexus.example.com

paths:

- path: /

pathType: Prefix

**7.2 Install NexusRM on EKS**

bash

CopyEdit

helm repo add sonatype https://sonatype.github.io/helm3-charts/

helm upgrade --install nexusrm sonatype/nexus-repository-manager \

--namespace nexusrm --create-namespace \

-f values.yaml

**8. Autoscaling Configuration**

**8.1 Enable Cluster Autoscaler**

bash

CopyEdit

kubectl apply -f cluster-autoscaler-autodiscover.yaml

**8.2 Horizontal Pod Autoscaler (HPA)**

bash

CopyEdit

kubectl autoscale deployment nexusrm \

--cpu-percent=70 --min=3 --max=10

**9. Monitoring and Alerts**

* **Metrics Collection:** Prometheus deployed in the EKS cluster.
* **Dashboarding:** Grafana integrated with Prometheus.
* **Alerting:** Alerts configured for high CPU, memory, and pod failures.
* **Log Management:** AWS CloudWatch for logs from NexusRM pods.

**10. Testing & Validation**

* **Deployment Verification:**

bash

CopyEdit

kubectl get pods -n nexusrm

kubectl get svc -n nexusrm

* **Autoscaling Test:**  
  Increase workload to observe HPA behavior.

bash

CopyEdit

kubectl run load-generator --image=busybox \

-- /bin/sh -c "while true; do wget -q -O- http://nexus.example.com; done"

* **Load Balancer Test:**  
  Access the application via DNS and check round-robin distribution.

**11. Monitoring Metrics**

| **Metric** | **Tool** | **Action** |
| --- | --- | --- |
| CPU/Memory Utilization | Prometheus | HPA Adjustment |
| Pod Restart Count | Grafana | Error Investigation |
| HTTP Latency/Errors | CloudWatch | Scaling Up |
| Disk I/O on EFS | AWS CloudWatch | Increase IOPS |

**12. Failure Scenarios and Recovery**

| **Scenario** | **Recovery Action** |
| --- | --- |
| Pod Failure | Autoscaler deploys new pods |
| Node Failure | Cluster Autoscaler adds new nodes |
| High CPU Usage | HPA scales out NexusRM pods |
| Database Connectivity | Check RDS endpoint and Kubernetes secrets |

**13. Success Criteria**

* Zero downtime during scaling events.
* Automated pod scaling under load.
* Seamless ALB routing with DNS failover.
* Consistent data availability via EFS.