Configuring an OpenSearch cluster with K8s Operator and GitOps Patterns

Marcel Yeonghyeon Ko

@SK hynix | AI/Data Engineer

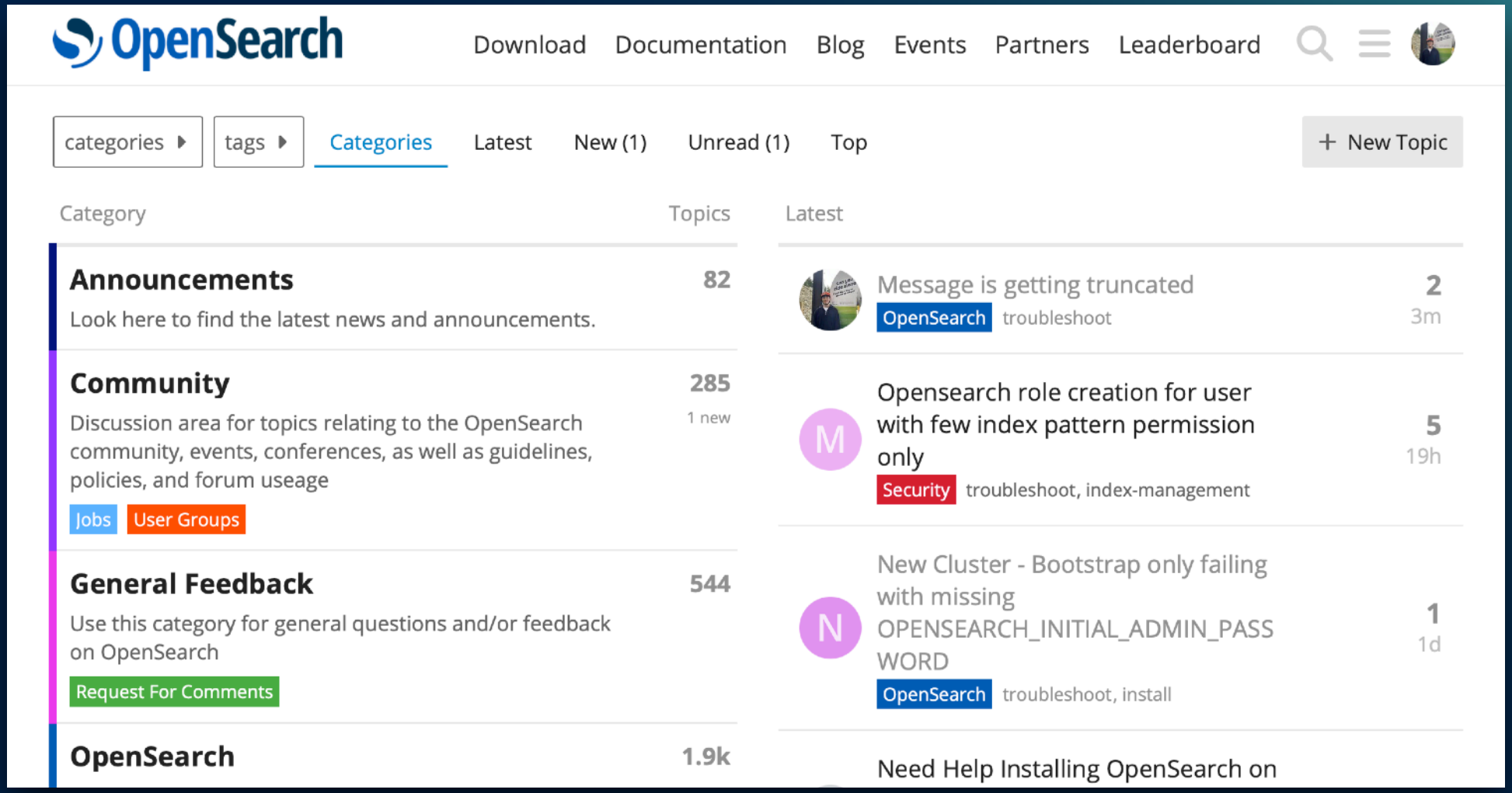
Introduction

Marcel Yeonghyeon Ko

- AI/Data Engineer @SK hynix
  - Search Engine & Performance Tuning (Elasticsearch, Spring Kafka)
  - RAG Pipeline (Crawling, KafkaConnect, OpenSearch)
  - Hybrid Cloud Platform Operation (Linux, Kubernetes)
- OpenSource Contribution
  - Lucene, KafkaConnect, Elasticsearch, OpenSearch
- Community
  - AWSKRUG (AIEngineering, PlatformEngineering)
  - OpenSearch Forum/Community

OpenSearch

Introduction



OpenSearch

- OpenSearch Forum - https://forum.opensearch.com

Index

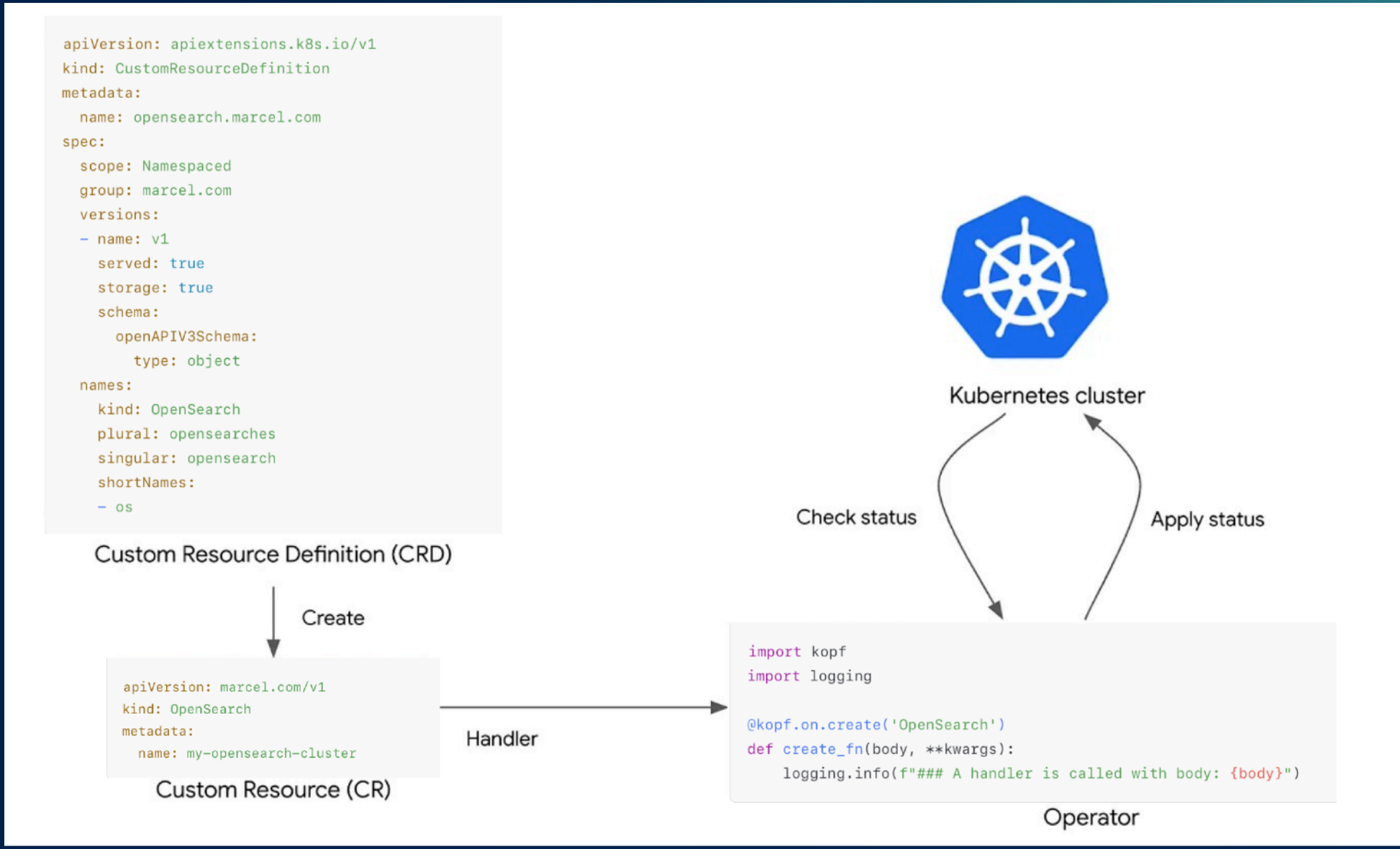
- Kubernetes Operator
- GitOps (feat. ArgoCD)
- Architecture in OpenSearch cluster
- TLS Authentication for Transport/HTTP Layer (feat. cert-manager)
- External Access to Cluster/Dashboards Endpoint (feat. Ingress)

1. Kubernetes Operator

Configuring an OpenSearch cluster with K8s Operator and GitOps Patterns

OpenSearch

1. Kubernetes Operator - **Flow**



1. Kubernetes Operator - **5W1H**

- **Who** - Managed Service Provider (ex. AWS OpenSearch Service staffs)
- **What** - CustomResourceDefinition(CRD) & Operator
- **Where** - Kubernetes Cluster (compatible version with CRD)
- **When** - Management for complex resource ecosystem (Monitor, DR)
- **Why** - Declarative Setting/Deployment, Scalability, Overview
- **How** - Operator's reconciler detects CR changes

1. Kubernetes Operator - **Reconciliation**

- K8s Operator continuously monitors the cluster and reconciles resource objects in response to changes in CustomResources (CRs)

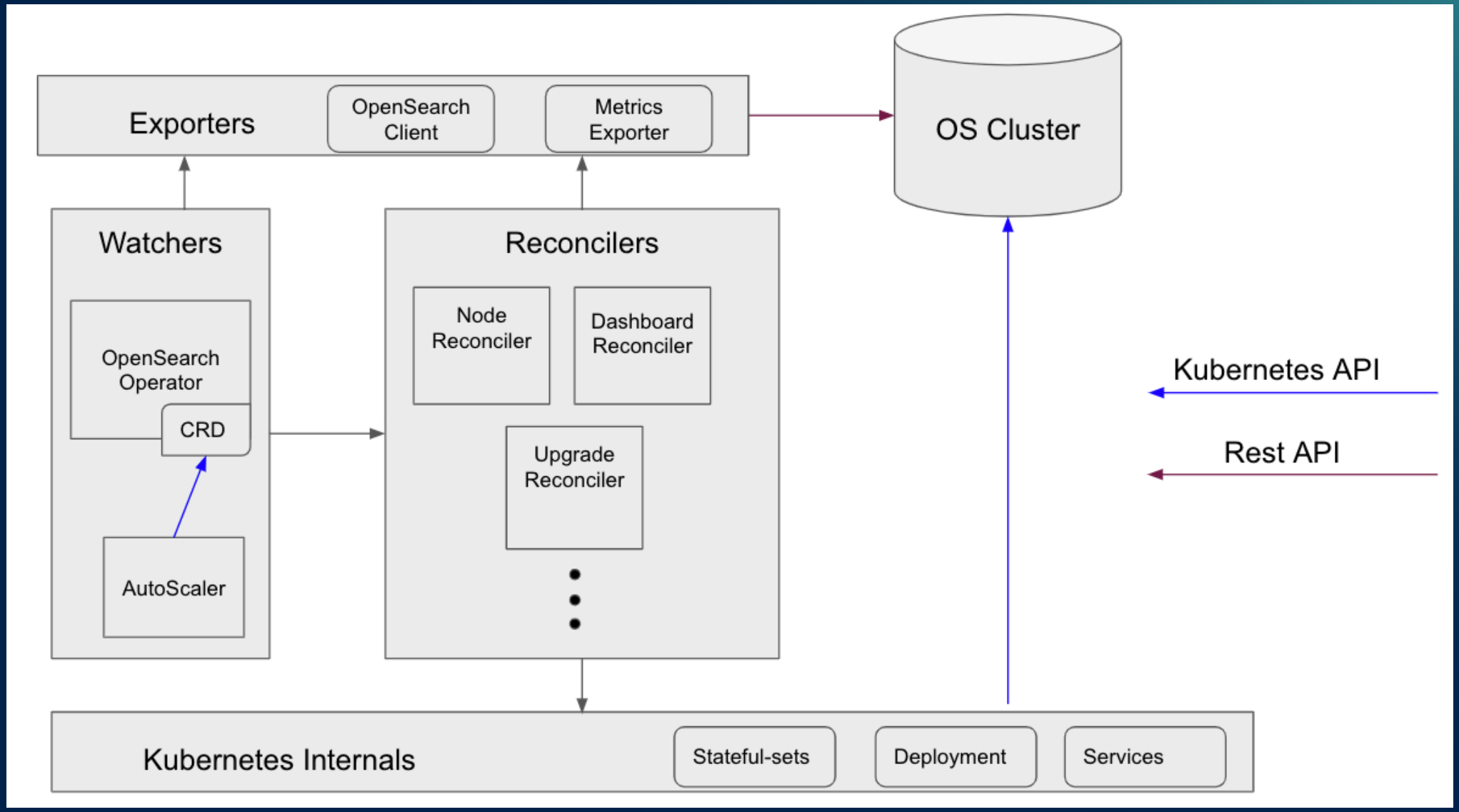
Reconciler	Description
Cluster	mapped to **nodePools**
Dashboards	mapped to **dashboards**
Configuration	In charge of OpenSearch **configuration**
TLS	mapped to security.**tls** (Security Plugin)
Securityconfig	mapped to security.**config** with users/roles (executes securityadmin.sh)

1. Kubernetes Operator - **CRD**

- OpenSearchCluster CR’s spec

Name	Description
general	OpenSearch general configuration
bootstrap	Bootstrap pod configuration
dashboards	OpenSearch Dashboards configuration
security	Defined security reconciler configuration
nodePools[ ]	Each nodePool represents a group of nodes with the same roles/resources. Deployed as a Kubernetes StatefulSet.

1. Kubernetes Operator - **Architecture**



2. GitOps (feat. ArgoCD)

Configuring an OpenSearch cluster with K8s Operator and GitOps Patterns

OpenSearch

2. GitOps (feat. ArgoCD) : **Paradigm**

Manual Deployment

- BC (**B**efore **C**I/CD)

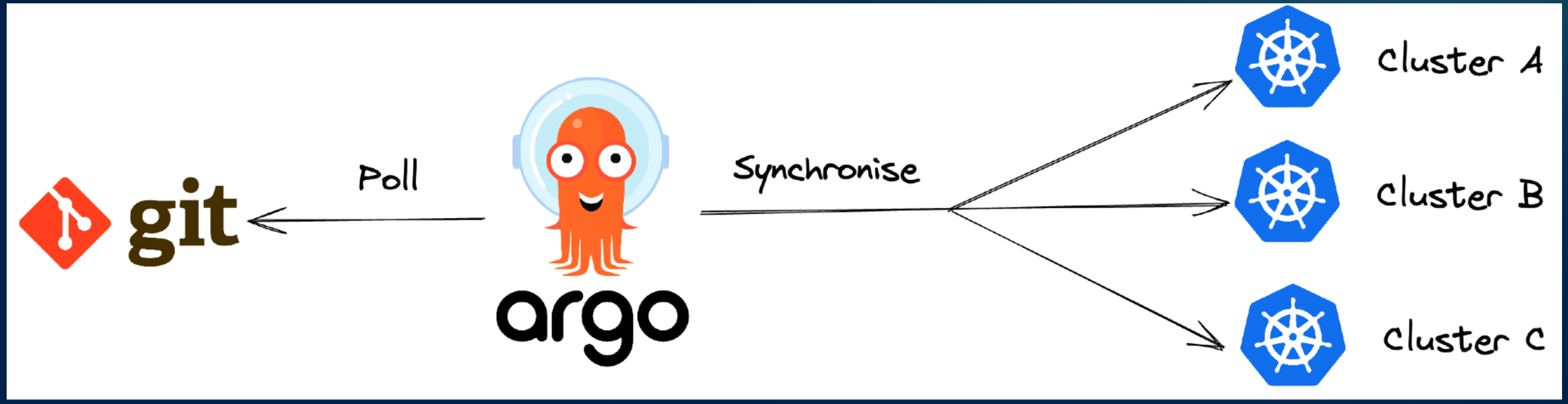
A-1	After local testing, push the merged branch(prd) to the remote Git Repo
A-2	Access the target instance, run git pull and build. For a Java application: gradlew build && java -jar {path}
B	Change local env and build, then deploy to the instance via FTP/SCP

* In Scenario B, Git doesn’t even exist!

2. GitOps (feat. ArgoCD) : **Paradigm**

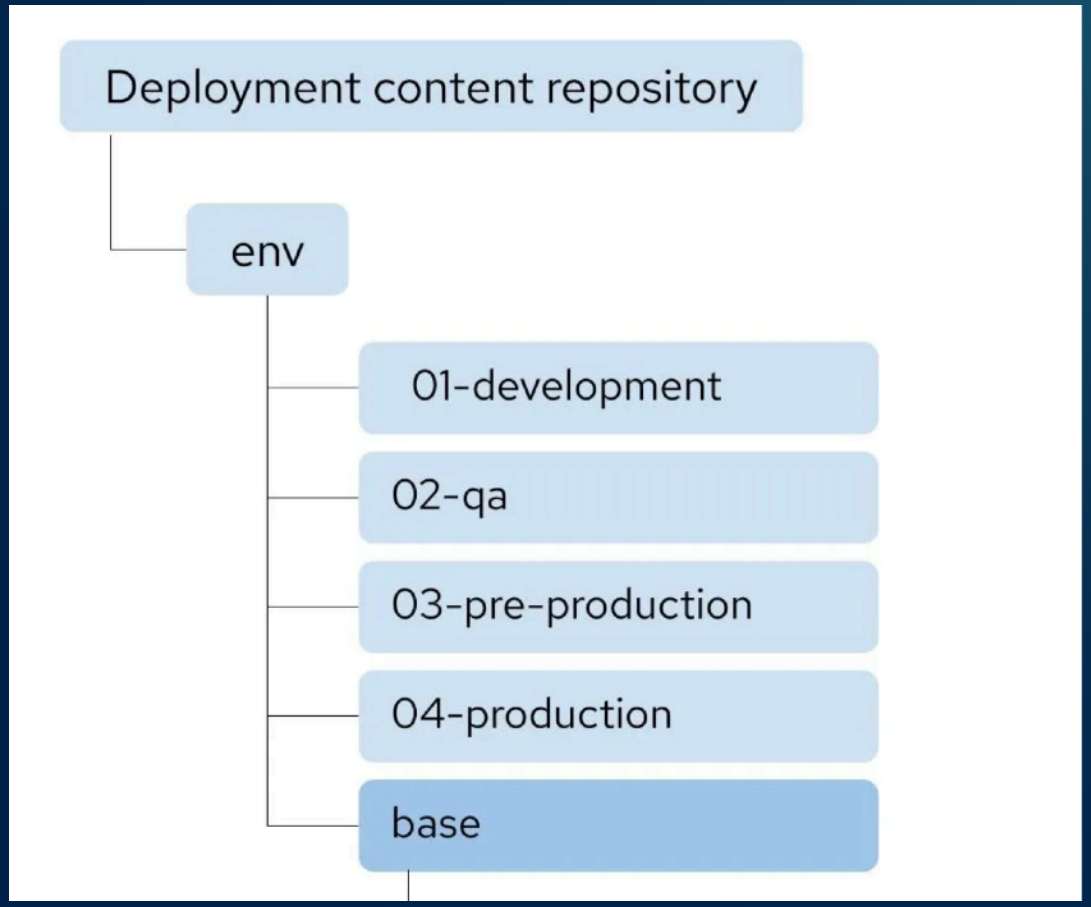
Declarative CI/CD와 Git Workflow

- AD (**A**fter ArgoC**D**)



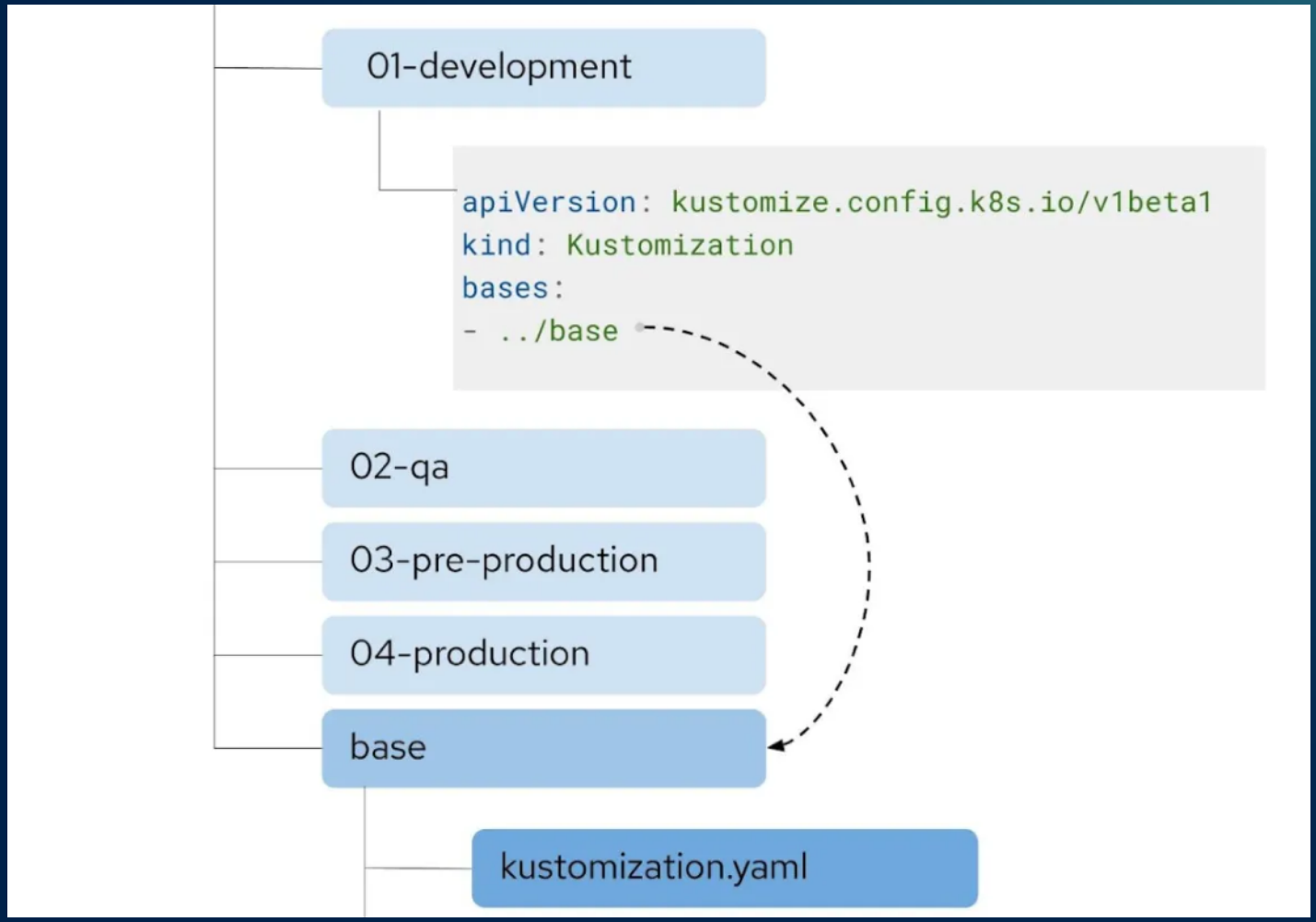
2. GitOps (feat. ArgoCD) : **Repo Structure**

- Common Git Repository structure:
  - **Base** (official helm charts) & **Environments** (dev/qa/stg/prd)



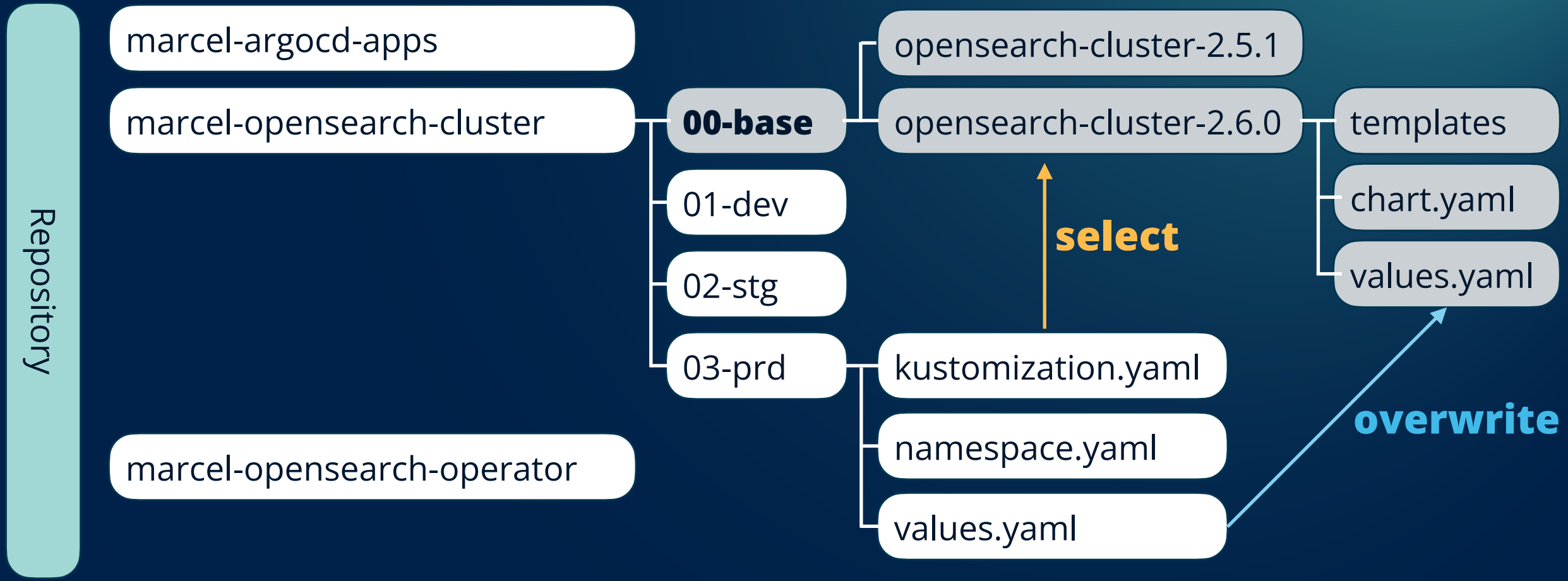
2. GitOps (feat. ArgoCD) : **Repo Structure**

- Overwrite YAMLs in **base** for each deployment env



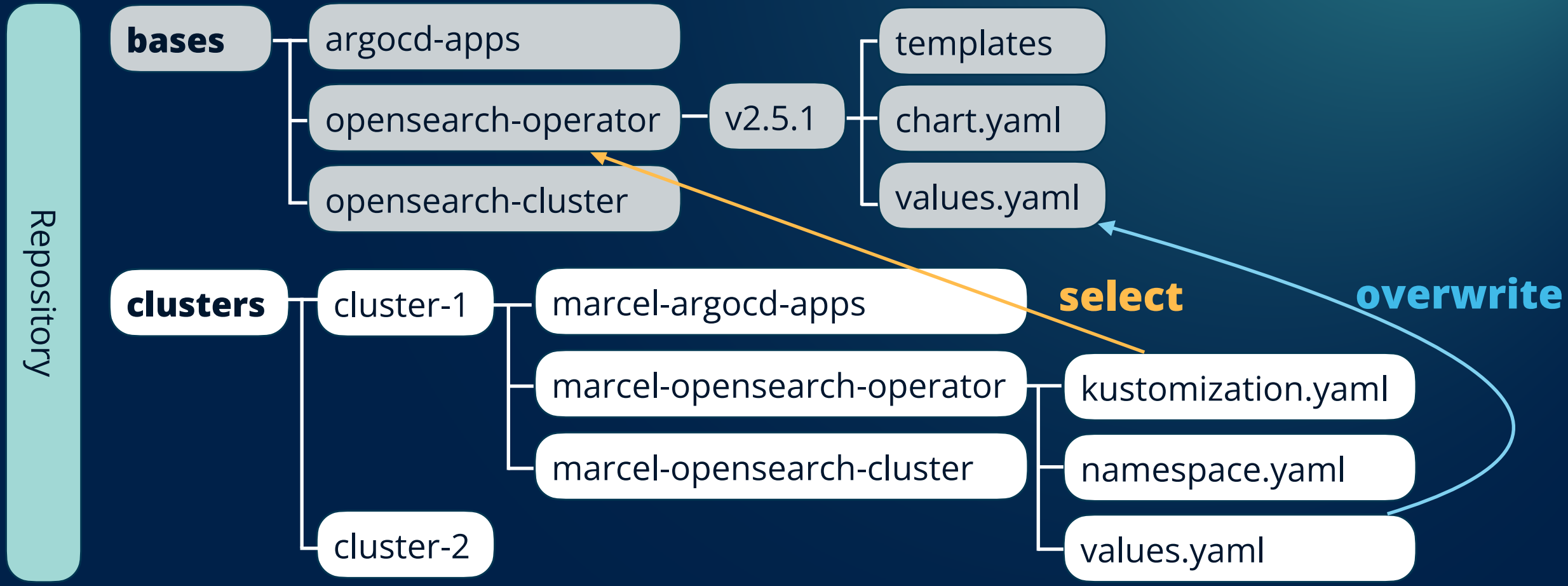
2. GitOps (feat. ArgoCD) : **Repo Structure**

Best Practice (1) - Single Cluster



2. GitOps (feat. ArgoCD) : **Repo Structure**

Best Practice (2) - Multi Clusters



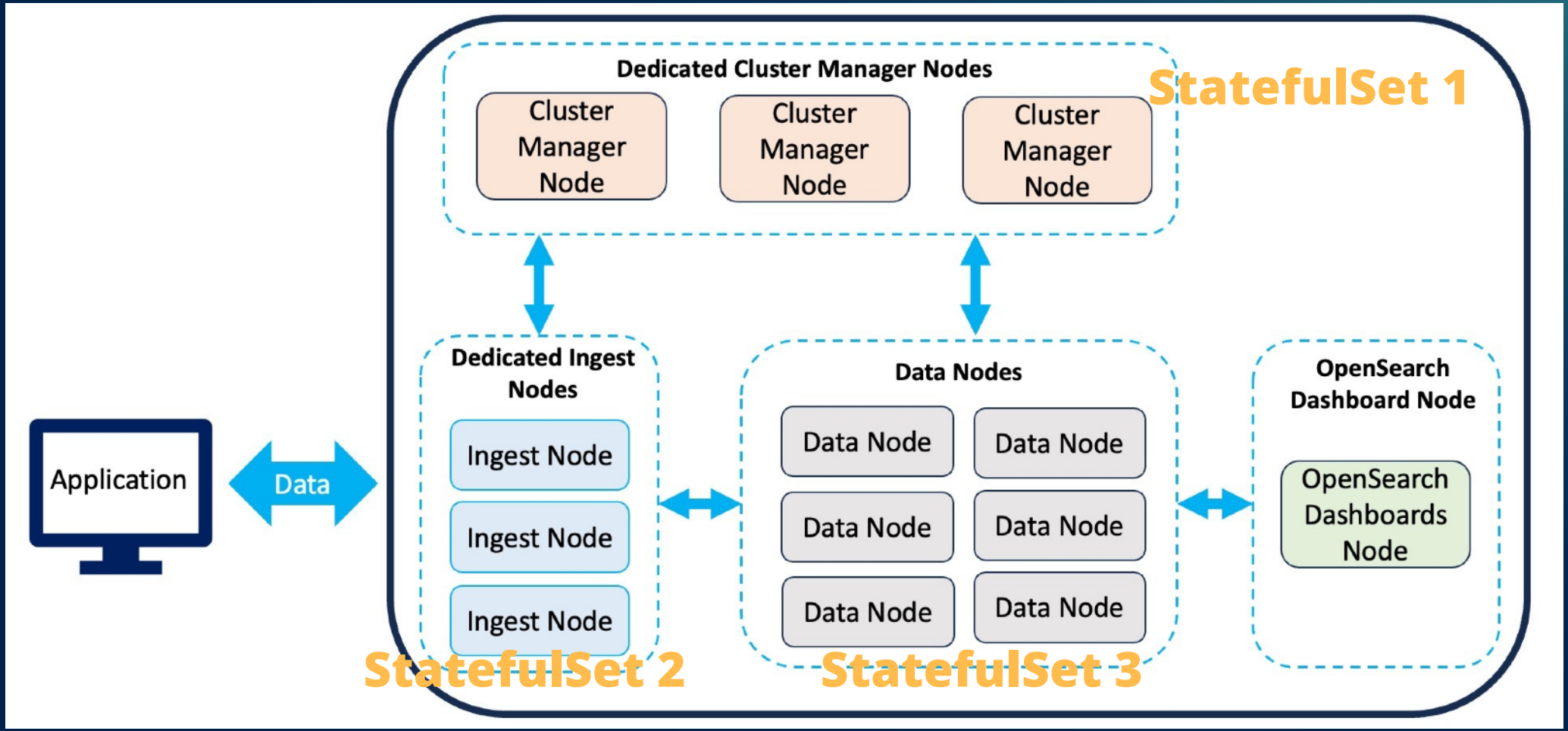
3. Architecture in OpenSearch cluster

Configuring an OpenSearch cluster with K8s Operator and GitOps Patterns

OpenSearch

3. Architecture in OpenSearch cluster - **Node**

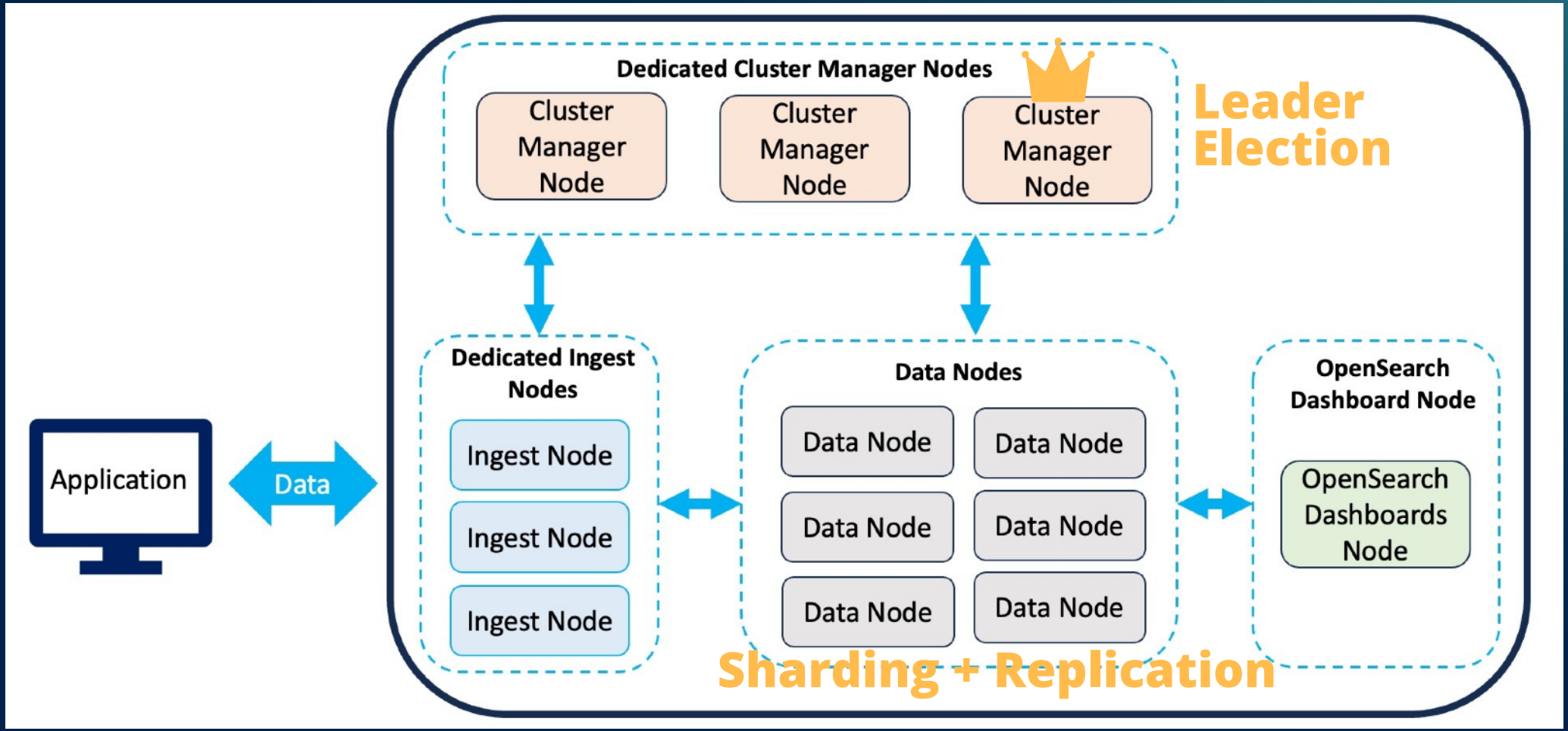
- OpenSearch : Kubernetes = **Nodes** : **StatefulSet**



https://www.instaclustr.com/blog/dedicated-ingest-nodes-public-preview-release-for-opensearch-on-instaclustr-by-netapp/

3. Architecture in OpenSearch cluster - **Node**

- Guarantee HA with **Quorum** & **Shard Replication**



https://www.instaclustr.com/blog/dedicated-ingest-nodes-public-preview-release-for-opensearch-on-instaclustr-by-netapp/

3. Architecture in OpenSearch cluster - **Node**

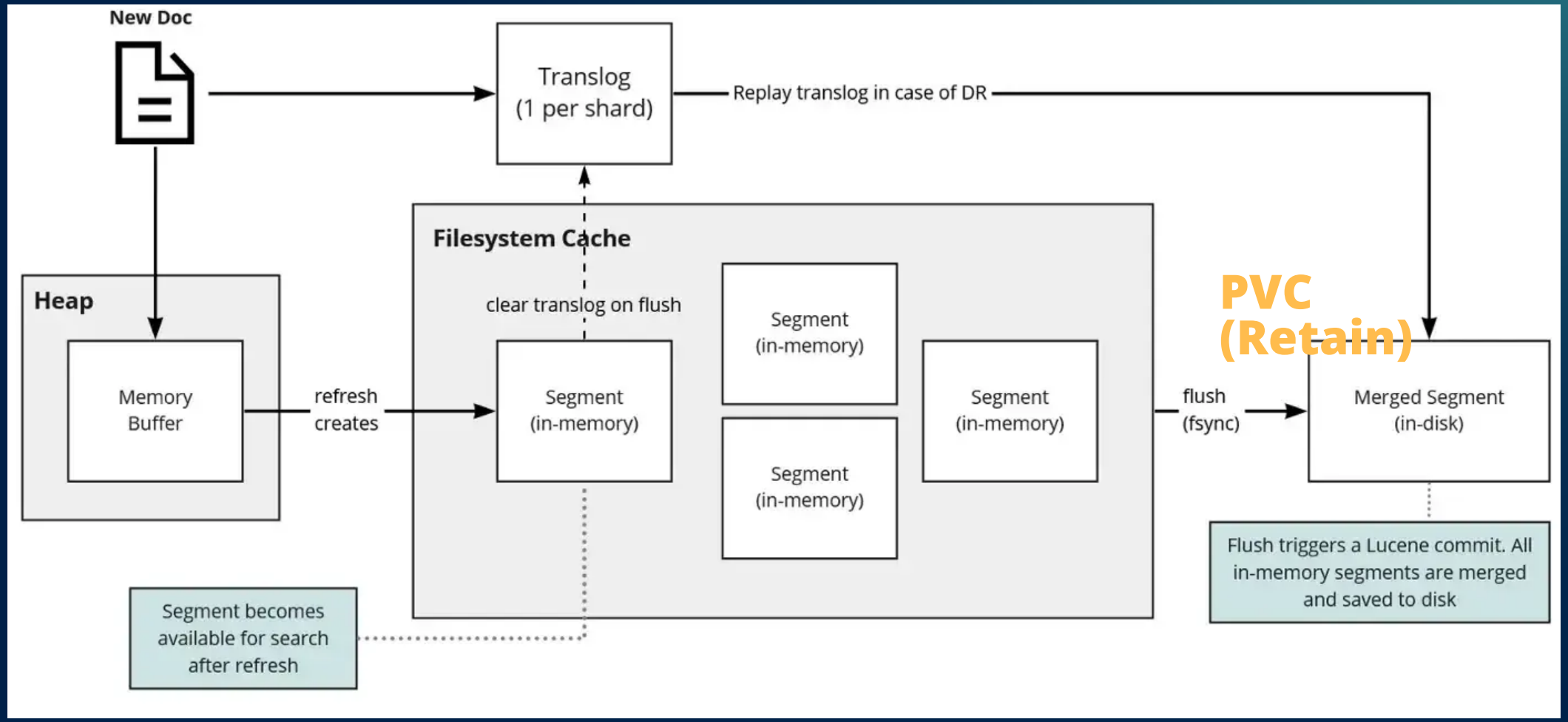
```yaml
spec:
  nodePools:
  - component: master
    replicas: 3
    diskSize: "10Gi"
    resources:
      requests:
        memory: "2Gi"
        cpu: "100m"
      limits:
        memory: "2Gi"
        cpu: "1"
    roles:
      - "cluster_manager"
  - component: data
    replicas: 2
    diskSize: "500Gi"
    resources:
      requests:
        memory: "8Gi"
        cpu: "100m"
      limits:
        memory: "8Gi"
        cpu: "4"
    roles:
      - "data"
```

StatefulSet - master

StatefulSet - data

3. Architecture in OpenSearch cluster - **Segment**

- OpenSearch : Kubernetes = **Segments** : **PVCs**



https://blog.devgenius.io/elasticsearch-solution-to-searching-71116220c82f

3. Architecture in OpenSearch cluster - **Segment**

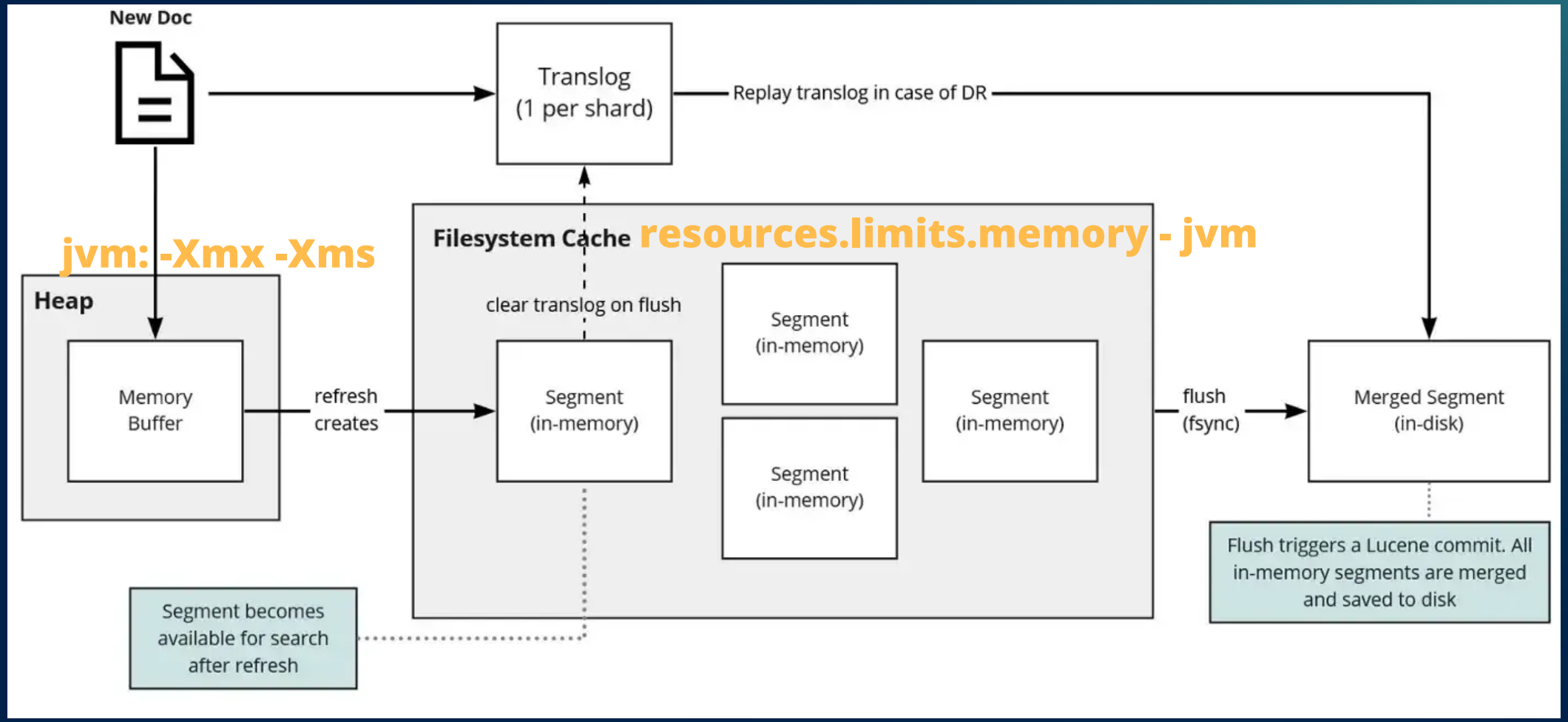
- OpenSearch : Kubernetes = **Segments** : **PVCs**

```
nodePools:
- component: standalone
  replicas: 3
  diskSize: 300
  roles:
    - "data"
    - "cluster_manager"
  persistence:
    pvc:
      storageClass: marcel-sc-vmware-vsphere
      accessModes:
      - ReadWriteOnce
```

PVC Volume Expansion is allowed with compatible CSI-driver & K8s version

3. Architecture in OpenSearch cluster - **Segment**

- JVM Heap & Page Cache in K8s/Linux



https://blog.devgenius.io/elasticsearch-solution-to-searching-71116220c82f

3. Architecture in OpenSearch cluster - **Segment**

- JVM Heap & Page Cache in K8s/Linux

```
spec:
    nodePools:
      - component: nodes
        replicas: 3
        diskSize: "10Gi"
        jvm: -Xmx1024M -Xms1024M
        resources:
          requests:
            memory: "2Gi"
            cpu: "500m"
          limits:
            memory: "2Gi"
            cpu: "500m"
        roles:
          - "data"
```

Kernel utilizes memory as system cache (2Gi - 1024M ~= 1123M)

*** memory-mapped I/O**

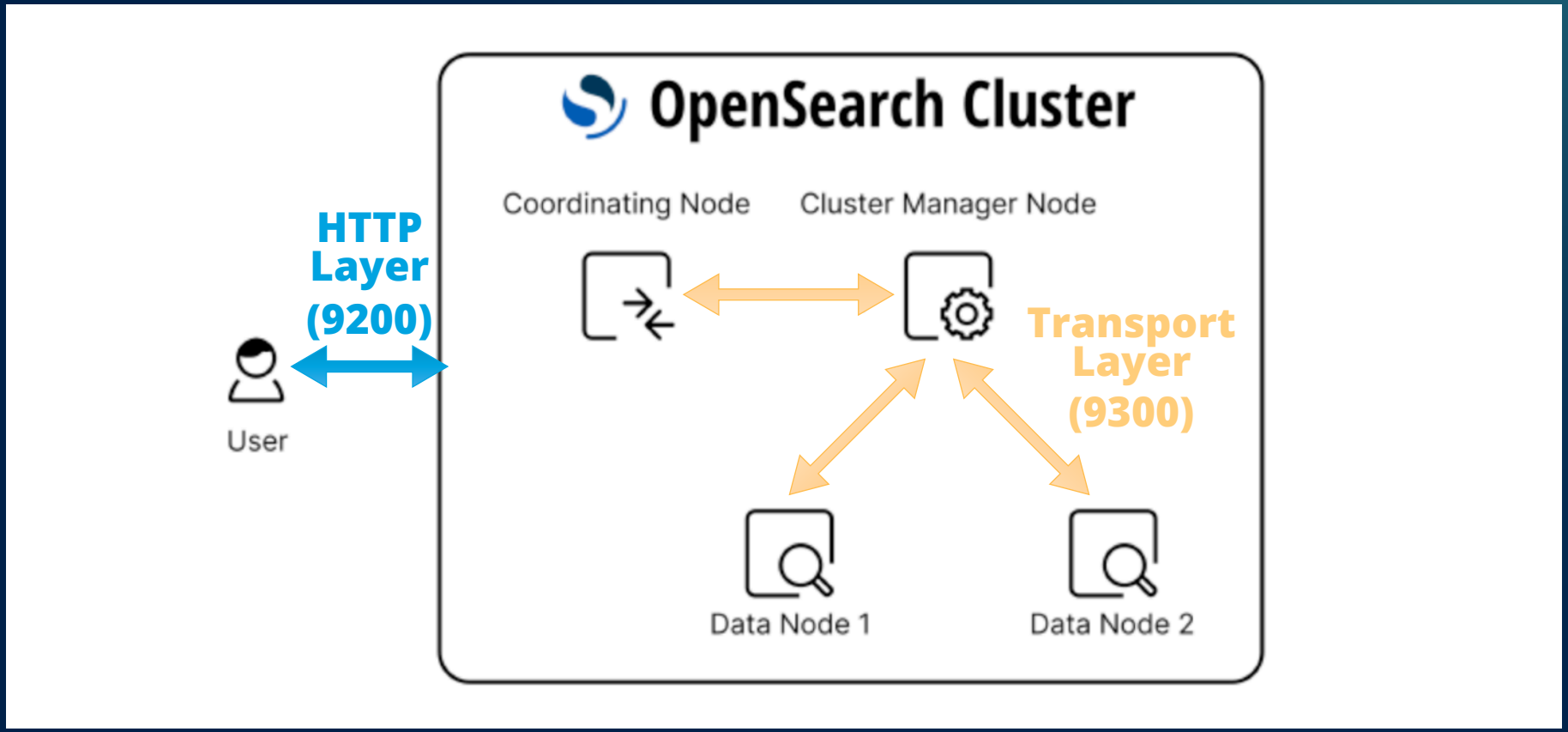
4. TLS Authentication for Transport/HTTP Layer (feat. cert-manager)

Configuring an OpenSearch cluster with K8s Operator and GitOps Patterns

OpenSearch

4. TLS Authentication for Transport/HTTP Layer

- OpenSearch uses HTTP Layer for Rest API and Transport Layer for internal node-node communication



4. TLS Authentication for Transport/HTTP Layer

- OpenSearch K8s Operator provides TWO ways of TLS:
    A. Operator-generated CA/Certificates
    B. Self-signed CA/Certificates

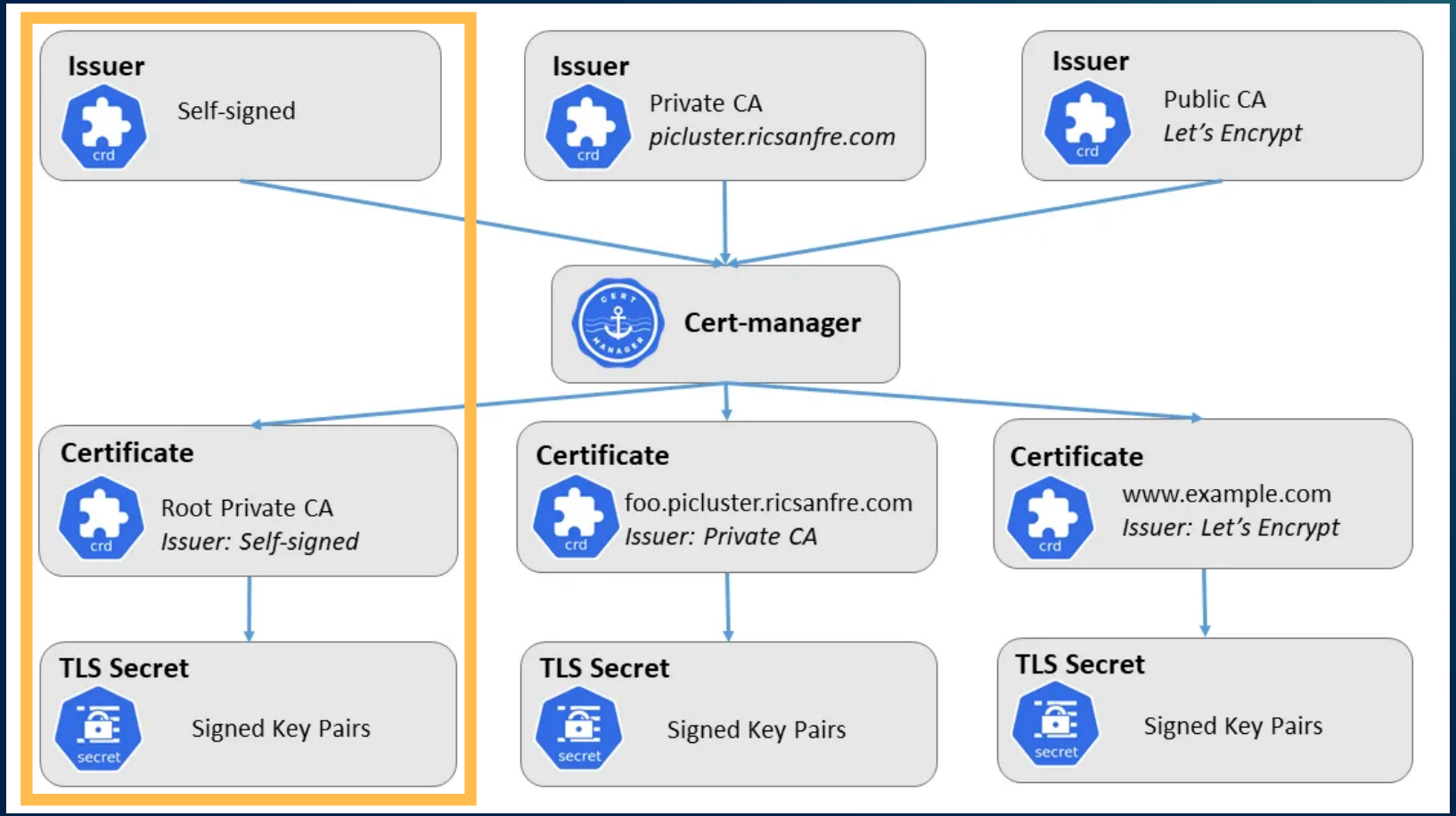
Depending on your requirements, the Operator offers two ways of managing TLS certificates. You can either supply your own certificates, or the Operator will generate its own CA and sign certificates for all nodes using that CA. The second option is recommended, unless you want to directly expose your OpenSearch cluster outside your Kubernetes cluster, or your organization has rules about using self-signed certificates for internal communication.

> ⚠️ **Clusters with operator-generated certificates will stop working after 1 year**: Make sure you have tested certificate renewals in your cluster before putting it in production!

https://github.com/opensearch-project/opensearch-k8s-operator/edit/main/docs/userguide/main.md#tls

4. TLS Authentication for Transport/HTTP Layer

- Let's Create Self-signed CA/Certificates with **cert-manager**



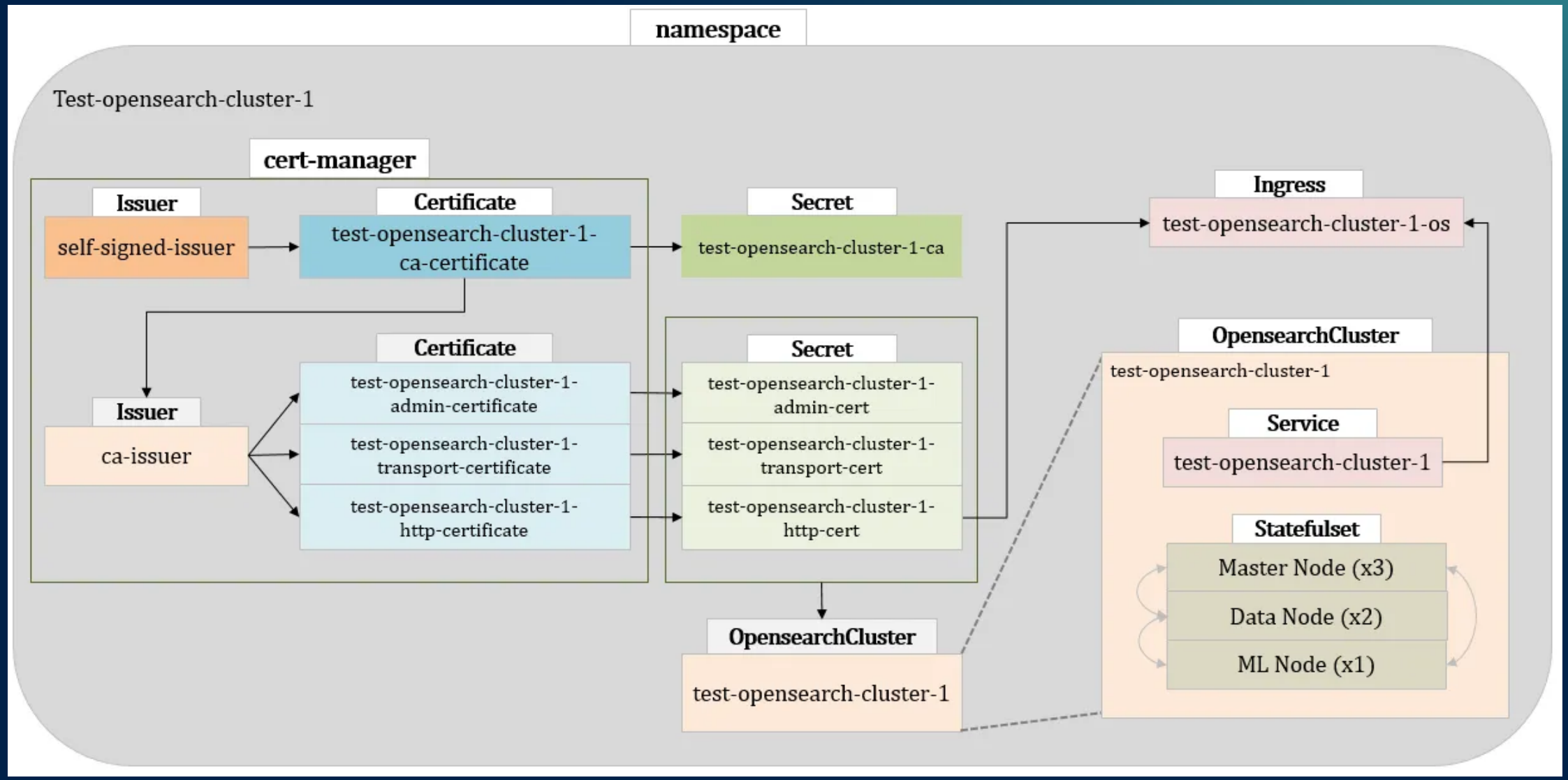
4. TLS Authentication for Transport/HTTP Layer

- **Issuer** : defined Certificate Authority (ex. ACME, Self-signed, etc)
- **Certificate** : generated Certificates by an Issuer
- **Secret** : stored resources including TLS certificates and key files

```
# transport/http 계층의 tls auto-generate 방식도 admin 인증서는 별도로 생성함.
test-opensearch-cluster-1-admin-cert       kubernetes.io/tls   3      26d
test-opensearch-cluster-1-admin-password   Opaque              2      26d
test-opensearch-cluster-1-ca               Opaque              2      26d
test-opensearch-cluster-1-http-cert        kubernetes.io/tls   3      26d
test-opensearch-cluster-1-transport-cert   kubernetes.io/tls   3      26d
```

https://medium.com/@dk02315/customizing-tls-for-opensearch-cluster-using-cert-manager-fb7c212e8ded

4. TLS Authentication for Transport/HTTP Layer



https://medium.com/@dk02315/customizing-tls-for-opensearch-cluster-using-cert-manager-fb7c212e8ded

5. External Access to Cluster/ Dashboards Endpoints (feat. Ingress)

Configuring an OpenSearch cluster with K8s Operator and GitOps Patterns

OpenSearch

5. External Access to Cluster/Dashboards Endpoints

- Why are Endpoints important for accessing OpenSearch?

✓ REST API : _index, _search (for Client)
✓ Cluster HealthCheck
✓ Index Management
✓ Security
✓ DevTools
✓ Discover / Visualize

Cluster Endpoint can

Dashboards Endpoint can

https://playground.opensearch.org/auth/anonymous?nextUrl=/app/home#/

5. External Access to Cluster/Dashboards Endpoints

```yaml
security:
  tls:
    transport:
      generate: false
      perNode: false
      secret:
        name: test-opensearch-cluster-1-transport-cert
      caSecret:
        name: test-opensearch-cluster-1-ca
      nodesDn: ["CN=test-opensearch-cluster-1,OU=test-opensearch-cluster-1"]
      adminDn: ["CN=admin,OU=test-opensearch-cluster-1"]
    http:
      generate: false
      secret:
        name: test-opensearch-cluster-1-http-cert
  config:
    adminSecret:
      name: test-opensearch-cluster-1-admin-cert
    adminCredentialsSecret:
      name: admin-credentials-secret
    securityConfigSecret:
      name: securityconfig-secret
```

5. External Access to Cluster/Dashboards Endpoints

[Cluster]

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    ingress.kubernetes.io/backend-protocol: HTTPS
  name: test-opensearch-cluster-1-os
  namespace: test-opensearch-cluster-1
spec:
  ingressClassName: nginx
  rules:
  - host: test-opensearch-cluster-1-os.marcel.com
    http:
      paths:
      - backend:
          service:
            name: test-opensearch-cluster-1
            port:
              number: 9200
        path: /
        pathType: ImplementationSpecific
  tls:
  - hosts:
    - test-opensearch-cluster-1-os.marcel.com
    secretName: test-opensearch-cluster-1-http-cert
```

[Dashboards]

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: test-opensearch-cluster-1-osd
  namespace: test-opensearch-cluster-1
spec:
  ingressClassName: nginx
  rules:
  - host: test-opensearch-cluster-1-osd.marcel.com
    http:
      paths:
      - backend:
          service:
            name: test-opensearch-cluster-1-dashboards
            port:
              number: 5601
        path: /
        pathType: ImplementationSpecific
```

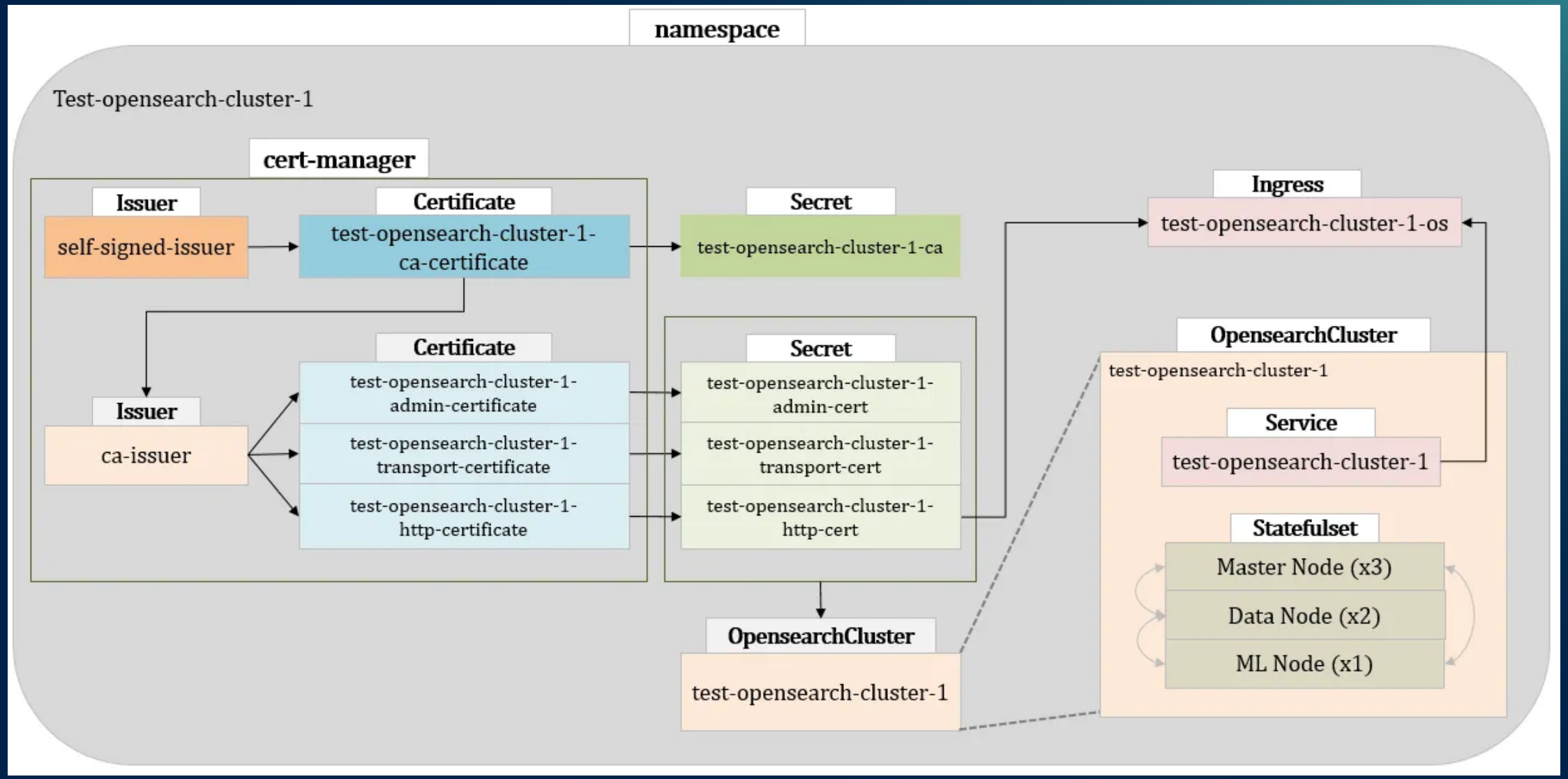
5. External Access to Cluster/Dashboards Endpoints

```
curl -ku admin:test -v https://test-opensearch-cluster-1-os.marcel.com/

*   Trying 10.100.200.32:443...
* TCP_NODELAY set
* Connected to test-opensearch-cluster-1-os.marcel.com (10.100.200.32) port 443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: /etc/ssl/certs/ca-certificates.crt
  CApath: /etc/ssl/certs
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384
...
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* old SSL session ID is stale, removing
* Connection state changed (MAX_CONCURRENT_STREAMS == 128)!
< HTTP/2 200
< date: Sun, 05 Jan 2025 07:37:26 GMT
< content-type: application/json; charset=UTF-8
< content-length: 599
< strict-transport-security: max-age=15724800; includeSubDomains
<
```


```
{
  "name" : "test-opensearch-cluster-1-data-0",
  "cluster_name" : "test-opensearch-cluster-1",
  "cluster_uuid" : "cDYu-y5bSfGWD-pQPP1VQQ",
  "version" : {
    "distribution" : "opensearch",
    "number" : "2.17.1",
    "build_type" : "tar",
    "build_hash" : "61dbcd0795c9bfe9b81e5762175414bc38bbcadf",
    "build_date" : "2024-06-20T03:26:49.193630411Z",
    "build_snapshot" : false,
    "lucene_version" : "9.11.0",
    "minimum_wire_compatibility_version" : "7.10.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "The OpenSearch Project: https://opensearch.org/"
}
```

5. External Access to Cluster/Dashboards Endpoints



Know-how for operating Managed OpenSearch

✓ How much should you trust Managed Service users

- The less you trust them, the less freedom the service provides.

  -> overcome to some extent with Good guidelines/references and UI/UX.

1) Enable drainDataNodes (cluster.routing.allocation.exclude) ?
2) Auto-allocate JVM Heap as 50% of available memory ?
3) How granular should we make internal_user & roles ?

Know-how for operating Managed OpenSearch

✓ Tips in Air-Gapped environments

- OpenSearch Cluster CR auto-installs prometheus-exporter starting from v3.0.0.
- But in Air-Gapped, only firewall errors are logged...
- Use [COPY] command in Docker Image to add plugins
- Likewise, ML Models can be installed/registered

Wrap-up

- Now you can understand :
  - ✓ OpenSearch K8s Operator
  - ✓ CI/CD via GitOps patterns
  - ✓ OpenSearch Architecture in Kubernetes Ecosystems
  - ✓ OpenSearch Network - TLS in HTTP/Transport Layer
  - ✓ External Endpoints for Cluster/Dashboards

Q&A

OpenSearch