eliatra

The OpenSearch Experts

OpenSearch in Kubernetes: Helm Charts and Operator

Simplified OpenSearch Cluster Deployment on Kubernetes:
Navigating Helm Charts and the OpenSearch Operator

Introduction

Anton Rubin

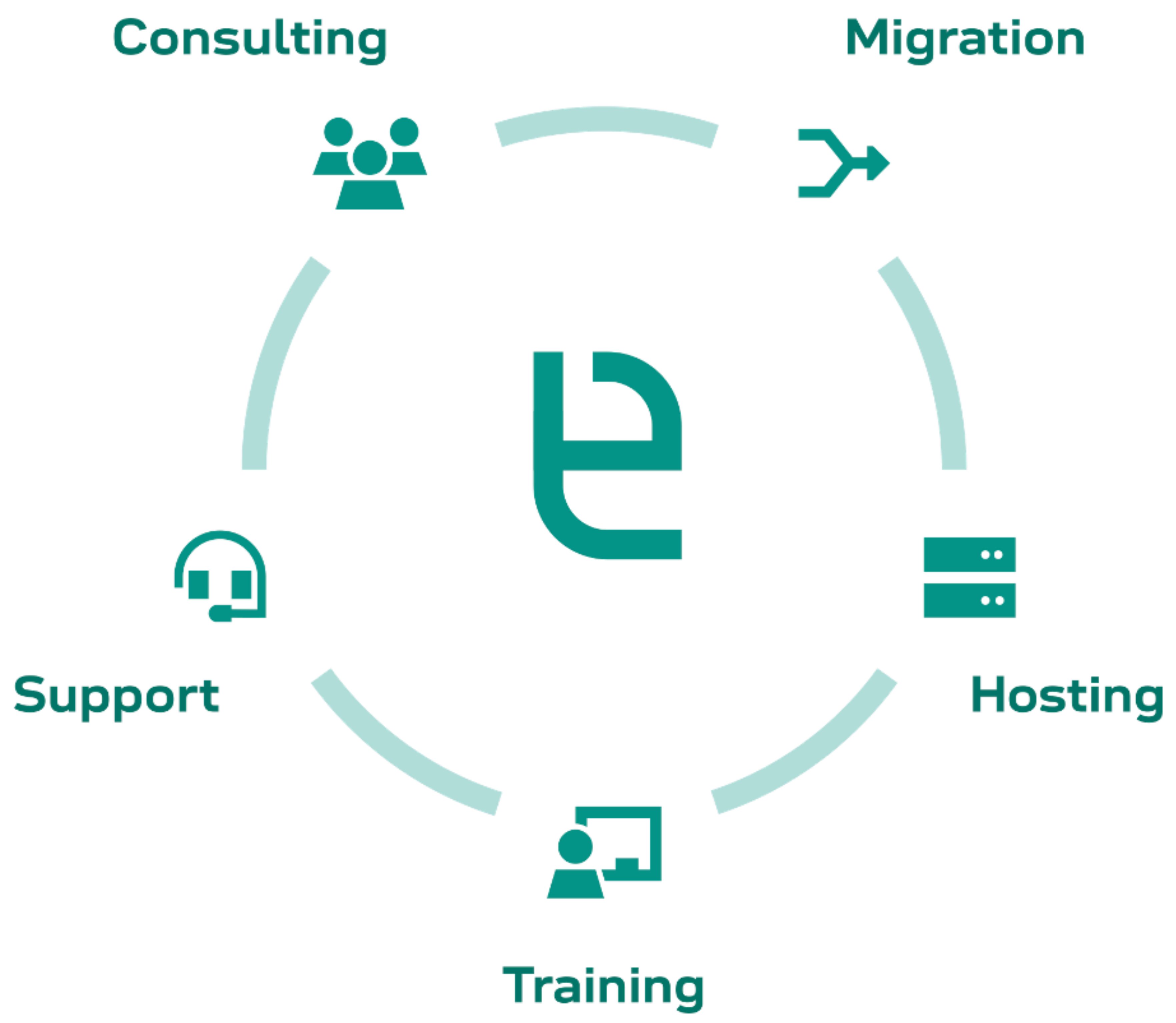
Technical Support Engineer at Eliatra

Company dedicated to one thing - to support & improve OpenSearch

linkedin.com/in/anton-rubin-89a186150

Eliatra OpenSearch Services

- 27/4 Enterprise grade support
- Professional Services
- Training/Workshops



What we are going to cover

- Creating self signed CA and certificates using Cert Manager
- Creating production ready certificates using Let’s Encrypt
- Configuring and deploying OpenSearch Cluster
    - Using helm charts
    - Using Operator
- Caveats to watch out for
- Rolling upgrade using both methods

Before we begin

Create EKS cluster

NOTE:

- No permission to create EBS volumes
- No Container Storage Interface (CSI) driver for EBS volumes

```
eksctl create cluster \
  --name os-cluster \
  --region eu-west-1 \
  --nodegroup-name linux-nodes \
  --node-type t2.medium \
  --nodes 5
```

Let’s fix that…

Configuring EKS cluster

Get role name

```
kubectl -n kube-system describe configmap aws-auth | awk '/rolearn:/ {print
$2}' | awk -F'role/' '{print $2}'
```

Create policy

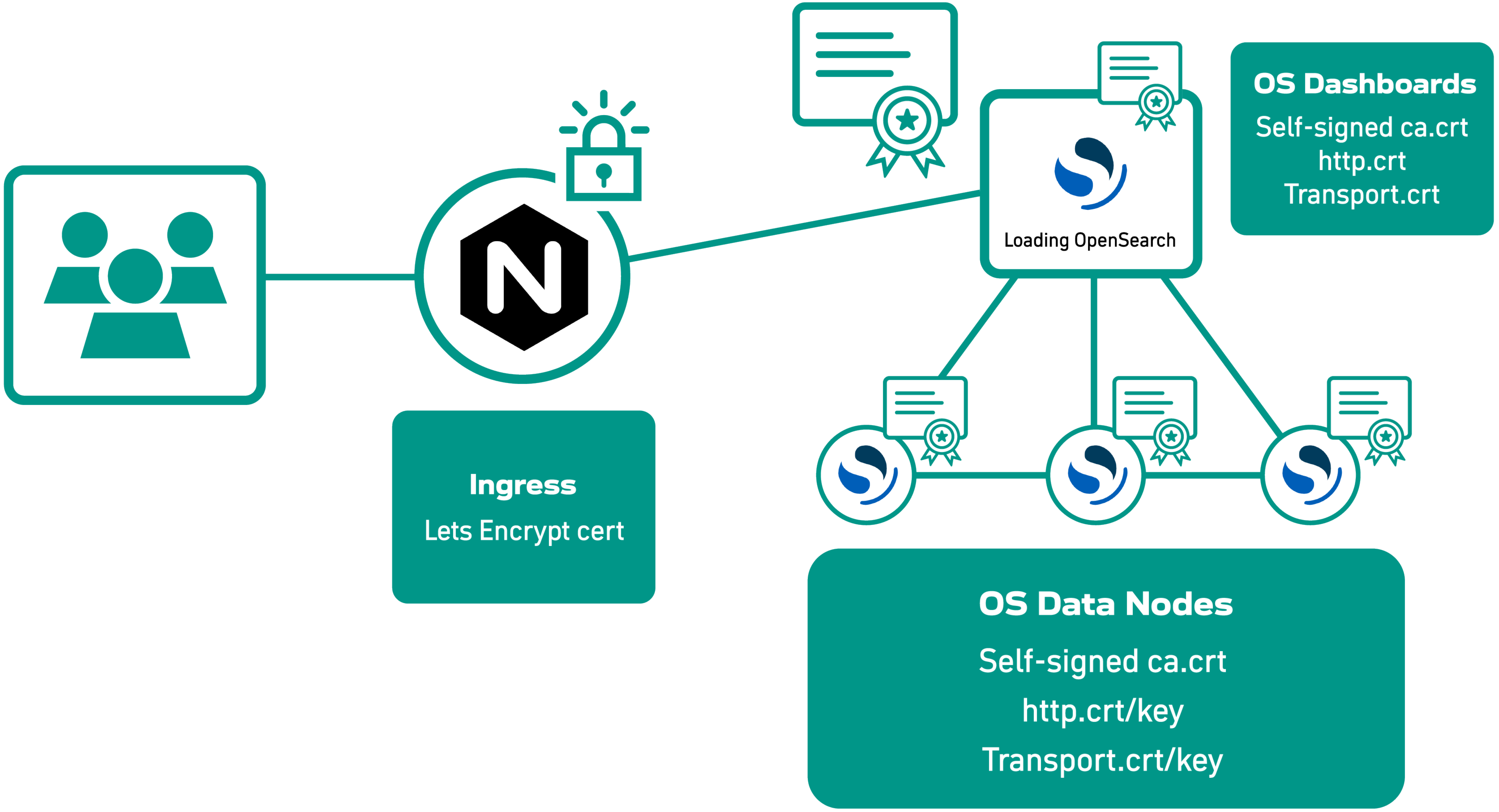
```
{
    „Version“: „2012-10-17“,
    „Statement“: [
        {
            „Sid“: „VisualEditor0“,
            „Effect“: „Allow“,
            „Action“: „ec2:*“,
            „Resource“: „*“
        }
    ]
}
```

Attach policy

```
aws iam attach-role-policy --role-name <role_name> --policy-arn <policy-arn>
```

Install EBS CSI driver

```
eksctl create addon --name aws-ebs-csi-driver --cluster os-cluster --force
```


Loading OpenSearch
OS Dashboards
Self-signed ca.crt
http.crt
Transport.crt
Ingress
Lets Encrypt cert
OS Data Nodes
Self-signed ca.crt
http.crt/key
Transport.crt/key

Installing Cert Manager

```
kubectl create namespace cert-manager
helm repo add jetstack https://charts.jetstack.io
helm repo update
helm install cert-manager jetstack/cert-manager --namespace cert-manager --version v1.7.0 --set
installCRDs=true
```

```
kubectl get pods --namespace cert-manager

NAME                                       READY   STATUS    RESTARTS   AGE
cert-manager-85bf57b9cf-d9wfk              1/1     Running   0          18m
cert-manager-cainjector-fc78d5686-76877    1/1     Running   0          18m
cert-manager-webhook-67f796c6d5-cqt88      1/1     Running   0          18m
```

Create self signed Certificate Authority (CA)

ClusterIssuer.yaml

```
---
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: selfsigned
spec:
  selfSigned: {}
```

Ca-cert.yaml

```
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: ca-cert
  namespace: cert-manager
spec:
  isCA: true
  duration: 43800h # 5 years
  commonName: not-important.io
  secretName: ca-key-pair
  privateKey:
    algorithm: RSA
    encoding: PKCS8
    size: 4096
  issuerRef:
    name: selfsigned
    kind: ClusterIssuer
    group: cert-manager.io
```

```
kubectl apply -f <file.yaml>
```

Create self signed Certificate Authority (CA) – Continued

View the ca certificate

```
kubectl get secret -n cert-manager

NAME                                                TYPE                DATA   AGE
ca-key-pair                                         kubernetes.io/tls   3      18m
```

```
kubectl describe secret ca-key-pair -n cert-manager
…
Data
====
ca.crt:   1797 bytes
tls.crt:  1797 bytes
tls.key:  3272 bytes
```

ca.crt and tls.crt is the same i.e. self signed

Create self signed certificates

Certificates we need to create

- OpenSearch transport and http layer
- OpenSearch Dashboards https
- OpenSearch Admin certificate for use with securityadmin.sh and special APIs

Create self signed certificates – Continued

Create ClusterIssuer from ca

```
---
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: cluster-issuer-from-ca
spec:
  ca:
    secretName: ca-key-pair
```

Transport and http

```
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: tls-for-opensearch
  namespace: default
spec:
  isCA: false
  duration: 2160h # 90d
  renewBefore: 360h # 15d
  commonName: node-eliatra-support.co.uk
  dnsNames:
  - node-eliatra-support.co.uk
  secretName: tls-for-opensearch-key-pair
  privateKey:
    algorithm: RSA
    encoding: PKCS8
    size: 2048
  usages:
    - server auth
    - client auth
  issuerRef:
    name: cluster-issuer-from-ca
    kind: ClusterIssuer
    group: cert-manager.io
```

Dashboards

```
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: tls-for-dashboards
  namespace: default
spec:
  isCA: false
  secretName: tls-for-dashboards-key-pair
  commonName: client-eliatra-support.co.uk
  dnsNames:
  - eliatra-support.co.uk
  privateKey:
    algorithm: RSA
    encoding: PKCS8
    size: 2048
  usages:
    - client auth
  issuerRef:
    name: cluster-issuer-from-ca
    kind: ClusterIssuer
    group: cert-manager.io
```

Admin

```
---
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: admin-for-opensearch
  namespace: default
spec:
  isCA: false
  duration: 2160h # 90d
  renewBefore: 360h # 15d
  commonName: admin
  dnsNames:
  - admin
  secretName: admin-for-opensearch-key-pair
  privateKey:
    algorithm: RSA
    encoding: PKCS8
    size: 2048
  usages:
    - client auth
  issuerRef:
    name: cluster-issuer-from-ca
    kind: ClusterIssuer
    group: cert-manager.io
```

Create self signed certificates – Continued

```
kubectl get certificate

NAME                      READY   SECRET                          AGE
admin-for-opensearch   True    admin-for-opensearch-key-pair   5s
tls-for-dashboards.     True    tls-for-dashboards-key-pair      5s
tls-for-opensearch      True    tls-for-opensearch-key-pair      5s
```

```
kubectl get secret

NAME                                 TYPE                 DATA  AGE
admin-for-opensearch-key-pair        kubernetes.io/tls    3     1s
tls-for-dashboards-key-pair           kubernetes.io/tls    3     1s
tls-for-opensearch-key-pair           kubernetes.io/tls    3     1s
```

```
kubectl describe secret tls-for-dashboards-key-pair

Data
====
ca.crt:   1797 bytes
tls.crt:  1582 bytes
tls.key:  1704 bytes
```

```
kubectl describe secret ca-key-pair -n cert-manager

Data
====
ca.crt:   1797 bytes
tls.crt:  1797 bytes
tls.key:  3272 bytes
```

Let’s Encrypt

Let's Encrypt is a nonprofit Certificate Authority (CA) that provides free SSL/ TLS certificates to enable HTTPS (secure HTTP) for websites.

2 ways to prove domain ownership:

http01 challenge vs DNS challenge

- Challenge file in .well-known/acme-challenge/ directory of the domain's web server
- uses port 80 to solve the challenge

- adds a specific DNS TXT record to the domain's DNS configuration

euatra

Installing Ingress controller

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update
helm install nginx-ingress ingress-nginx/ingress-nginx
```

```
helm list
nginx-ingress       default    1           2024-05-03 13:01:54.747017 +0100 IST  deployedingress-nginx-4.10.1

kubectl get pods
default       nginx-ingress-ingress-nginx-controller-f6bbd8ff9-5v2j8    1/1     Running     0          2m
```

Let’s Encrypt - HTTP01 Challenge

ClusterIssuer.yml with http challenge

```
---
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
  name: letsencrypt-prod
spec:
  acme:
    #server: https://acme-staging-v02.api.letsencrypt.org/directory
    server: https://acme-v02.api.letsencrypt.org/directory
    email: info@eliatra.com
    privateKeySecretRef:
      name: letsencrypt-prod-key
    solvers:
      - http01:
          ingress:
            class: nginx
```

Let’s Encrypt - DNS Challenge

ClusterIssuer
with DNS challenge

```
apiVersion: cert-manager.io/v1
kind: ClusterIssuer
metadata:
 name: letsencrypt-prod
spec:
 acme:
   #server: https://acme-staging-v02.api.letsencrypt.org/directory
   server: https://acme-v02.api.letsencrypt.org/directory
   email: info@eliatra.com
   privateKeySecretRef:
     name: letsencrypt-prod
   solvers:
     - selector:
         dnsZones:
           - "eliatra-support.co.uk"
       dns01:
         route53:
           region: eu-west-1
           accessKeyID: AKIA...
           secretAccessKeySecretRef:
             name: my-aws-secret
             key: token
```

Save aws token as secret

```
apiVersion: v1
kind: Secret
metadata:
  name: my-aws-secret
  namespace: cert-manager
type: Opaque
data:
  token: ....
```

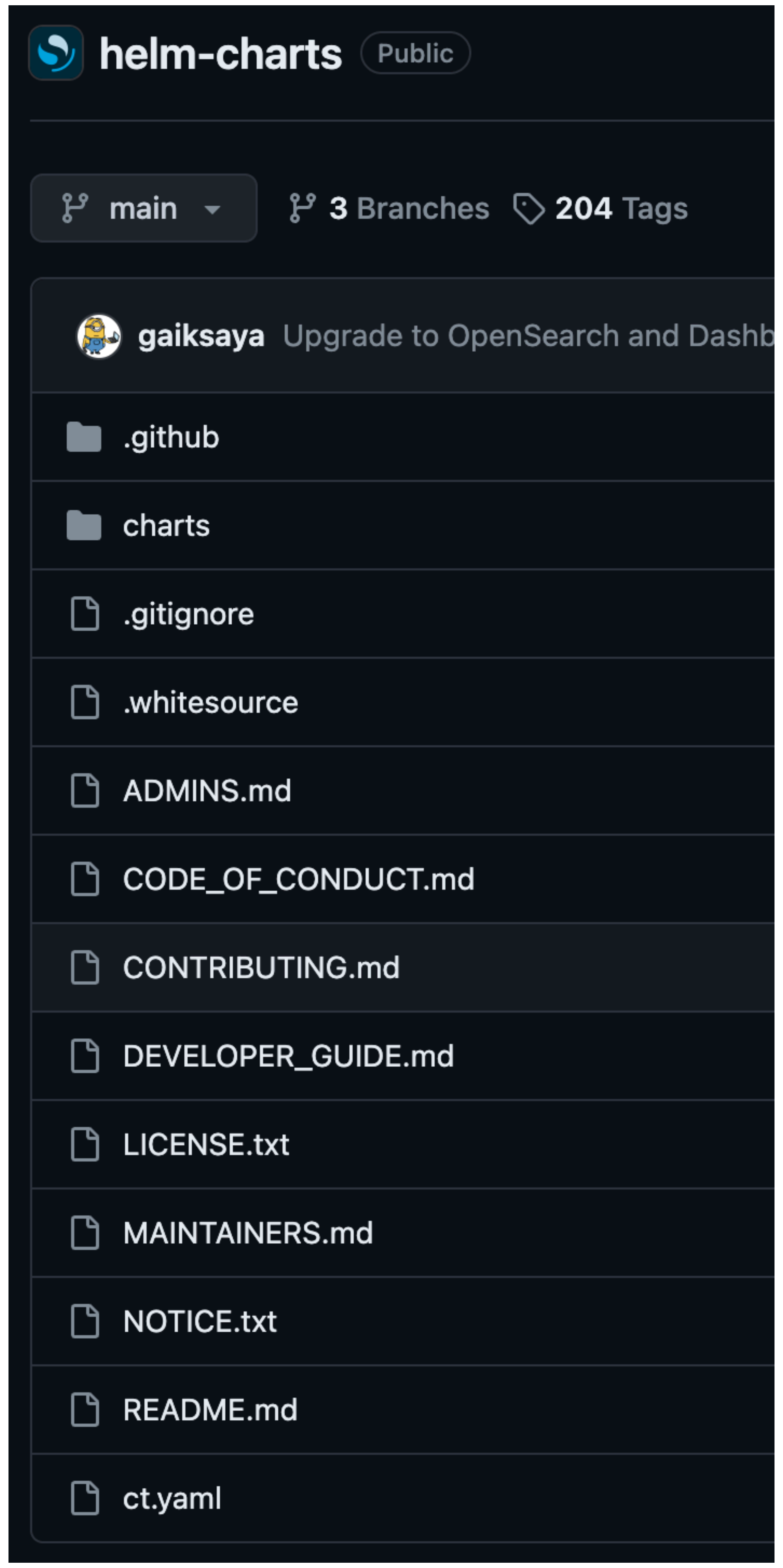
Let’s Encrypt - Certificates

Creating node, client and admin certificates

```
apiVersion: cert-manager.io/v1
kind: Certificate
metadata:
  name: tls-for-opensearch
  namespace: default
  annotations:
    cert-manager.io/secret-name: tls-for-opensearch-key
spec:
  secretName: tls-for-opensearch-key
  issuerRef:
    name: letsencrypt-prod
    kind: ClusterIssuer
    group: cert-manager.io
  commonName: eliatra-support.co.uk
  dnsNames:
  - eliatra-support.co.uk
  privateKey:
    algorithm: RSA
    encoding: PKCS8
    size: 2048
  usages:
    - server auth
    - client auth
```

Configuring OpenSearch helm files

https://github.com/opensearch-project/helm-charts

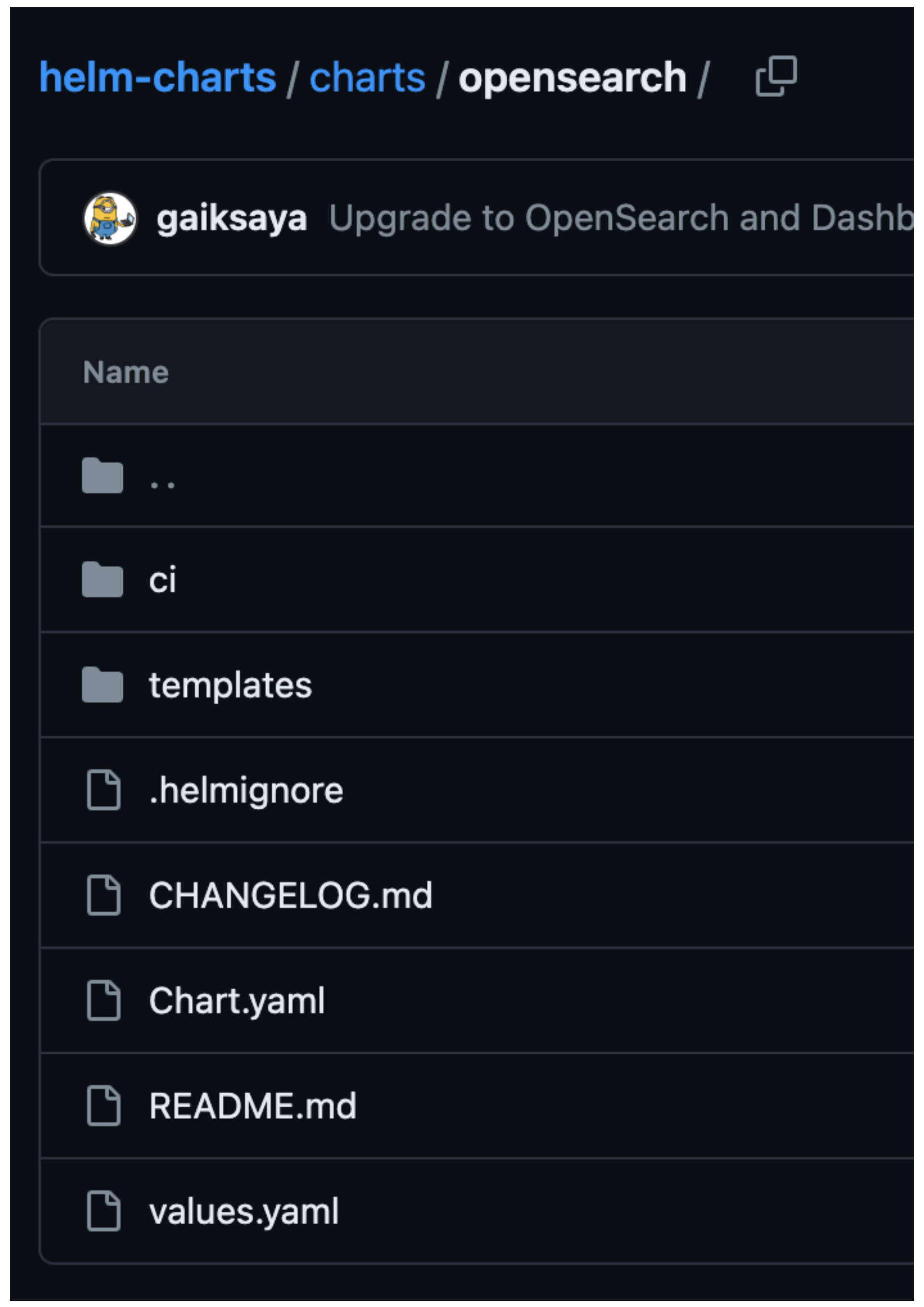


helm-charts Public

main 3 Branches 204 Tags

gaiksaya Upgrade to OpenSearch and Dashb

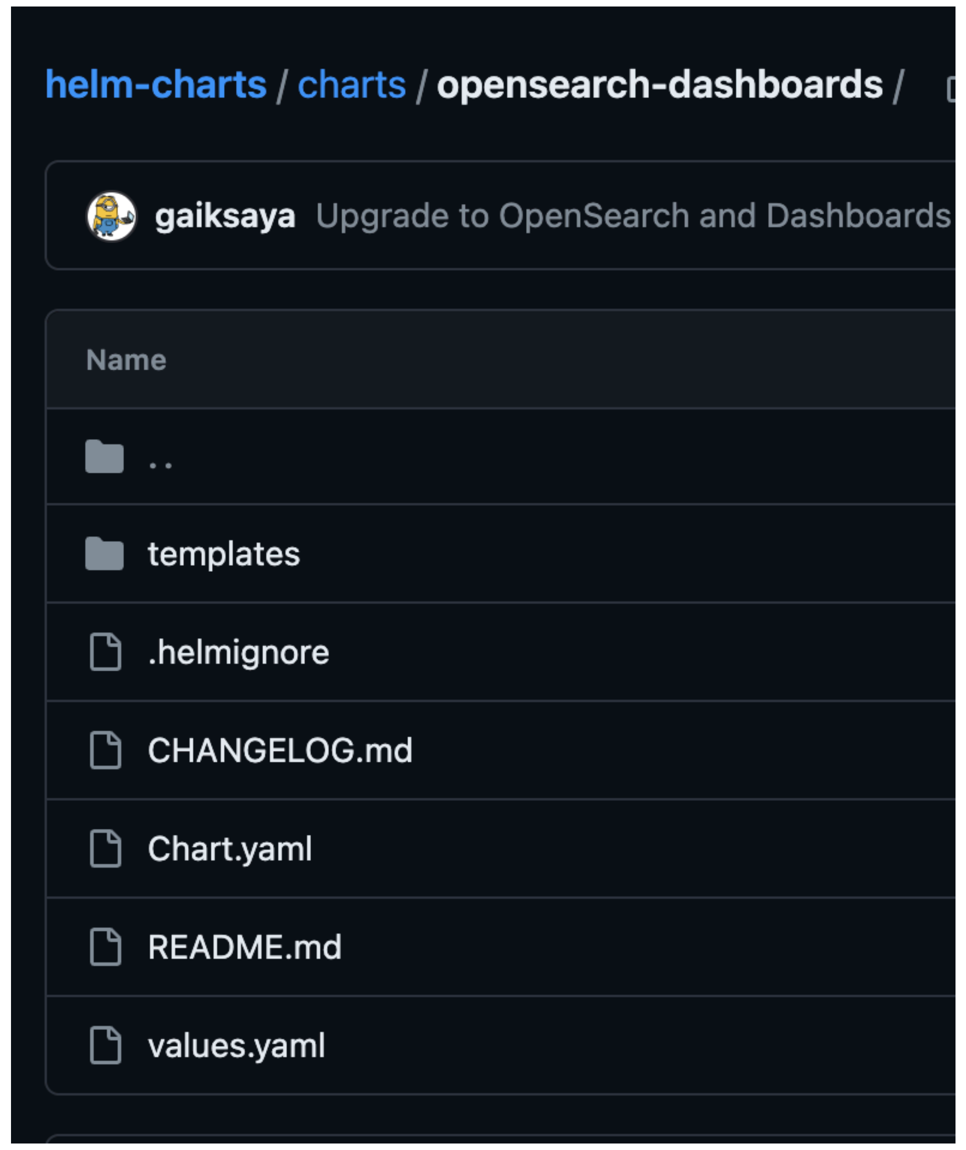
.github
charts
.gitignore
.whitesource
ADMINS.md
CODE_OF_CONDUCT.md
CONTRIBUTING.md
DEVELOPER_GUIDE.md
LICENSE.txt
MAINTAINERS.md
NOTICE.txt
README.md
ct.yaml



helm-charts / charts / opensearch /

gaiksaya Upgrade to OpenSearch and Dashb

Name
..
ci
templates
.helmignore
CHANGELOG.md
Chart.yaml
README.md
values.yaml



helm-charts / charts / opensearch-dashboards /

gaiksaya Upgrade to OpenSearch and Dashboards

Name
..
templates
.helmignore
CHANGELOG.md
Chart.yaml
README.md
values.yaml

Configuring OpenSearch helm files

Charts/opensearch/values.yaml

```
securityConfig.config:
 data:
   action_groups.yml: |-
    _meta:
     type: "actiongroups"
     config_version: 2
   config.yml: |-
    _meta:
     type: "config"
     config_version: 2
    config:
     dynamic:
      do_not_fail_on_forbidden: true
      http:
       anonymous_auth_enabled: false
       xff:
        enabled: false
        internalProxies: "192\\.168\\.0\\.10|192\\.168\\.0\\.11"
      authc:
…
```

```
secretMounts:
 - name: opensearch-certs
  secretName: tls-for-opensearch-key
  path: /usr/share/opensearch/config/certs
 - name: opensearch-certs-admin
  secretName: admin-for-opensearch-key
  path: /usr/share/opensearch/config/certs/admin
```

```
extraEnvs:
 - name: DISABLE_INSTALL_DEMO_CONFIG
   value: "true"
```

```
opensearch.yml: |
  cluster.name: opensearch-cluster
  network.host: 0.0.0.0
  plugins:
   security:
    nodes_dn:
      - 'CN=node-eliatra-support.co.uk'
    ssl:
     transport:
      pemcert_filepath: certs/tls.crt
      pemkey_filepath: certs/tls.key
…
```

Configuring OpenSearch Dashboards helm files

Charts/opensearch-dashboards/values.yaml

```
config:
  opensearch_dashboards.yml:
    server.ssl.enabled: true
    server.ssl.certificate: /usr/share/opensearch-dashboards/certs/
tls.crt
    server.ssl.key: /usr/share/opensearch-dashboards/certs/tls.key
    opensearch.hosts: ["https://opensearch-cluster-master:9200"]
    opensearch.ssl.verificationMode: none
    opensearch.username: ${OPENSEARCH_USERNAME}
    opensearch.password: ${OPENSEARCH_PASSWORD}
    opensearch.requestHeadersAllowlist:
[ authorization,securitytenant ]
    opensearch_security.multitenancy.enabled: true
    opensearch_security.multitenancy.tenants.preferred: ["Private",
"Global"]
    opensearch_security.readonly_mode.roles: ["kibana_read_only"]
    opensearch_security.cookie.secure: false
```

```
secretMounts:
  - name: opensearch-certs
    secretName: tls-for-dashboards-key-pair
    path: /usr/share/opensearch-dashboards/certs
```

```
opensearchAccount:
  secret: "opensearch-dashboards-account"
  keyPassphrase:
    enabled: false
```

Create secrets for Dashboards

opensearch_dashboards.yml

```
opensearch.username: <…>
opensearch.password: <…>
opensearch_security.cookie.password: <…>
```

dashboards_secret.yml

```
apiVersion: v1
kind: Secret
metadata:
  name: opensearch-dashboards-account
data:
  username: a2liYW5hc2VydmVy
  password: a2liYW5hc2VydmVy
  cookie: b3BlbnNlYXJjaC1kYXNoYm9hcmRzLWNvb2tpZXBhc3N3b3Jk
```

Deploying OpenSearch and Dashboards with helm

```
cd charts/opensearch
helm install --values=values.yaml opensearch .
```

```
cd charts/opensearch-dashboards
helm install --values=values.yaml dashboards .
```

```
kubectl get pods

NAME                                            READY   STATUS    RESTARTS   AGE
dashboards-opensearch-dashboards-56b9b9bdb8-f7gpb   1/1     Running   0          15m
opensearch-cluster-master-0                     1/1     Running   0          26m
opensearch-cluster-master-1                     1/1     Running   0          26m
opensearch-cluster-master-2                     1/1     Running   0          26m
```

Helm deployment caveats

Separating node roles:

- multiple helm deployments are needed with separate values.yaml files to manage.
- therefore if you want 1 master, 1 data and 1 ingest node, you would need three separate helm deployments.

Helm deployment caveats

Rolling updates:

- rolling updates are performed on pods in sequence but only on an infrastructure level, therefore for example disabling shard allocation, which is highly recommended during upgrades/restarts, is not performed and there is no check for cluster state prior to moving to next node.

Possible workarounds: pre and post hooks

Deploying Operator

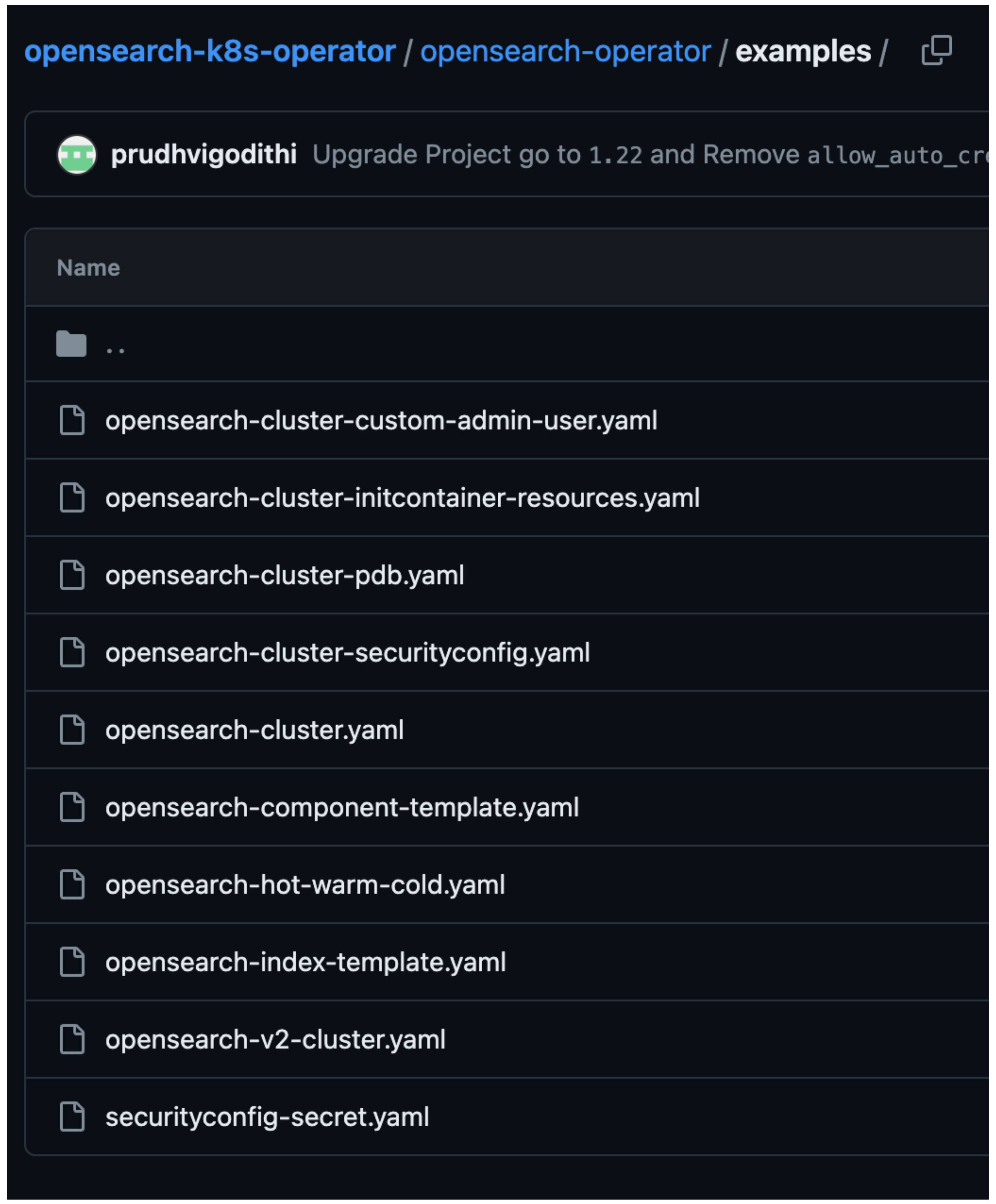
https://github.com/opensearch-project/opensearch-k8s-operator

```
helm repo add opensearch-operator https://opensearch-project.github.io/opensearch-k8s-operator/
helm install opensearch-operator opensearch-operator/opensearch-operator # (-f values.yaml)

kubectl get pods
opensearch-operator-controller-manager-7f4ffb4cb8-xr56v   2/2     Running   0          52s

1. kube-rbac-proxy
2. operator-controller-manager
```

*Single node clusters are not currently supported

Configuring OpenSearchCluster

One yaml file for all configuration using new CRD



opensearch-k8s-operator / opensearch-operator / examples /

prudhvigodithi Upgrade Project go to `1.22` and Remove `allow_auto_cr`

Name
..
opensearch-cluster-custom-admin-user.yaml
opensearch-cluster-initcontainer-resources.yaml
opensearch-cluster-pdb.yaml
opensearch-cluster-securityconfig.yaml
opensearch-cluster.yaml
opensearch-component-template.yaml
opensearch-hot-warm-cold.yaml
opensearch-index-template.yaml
opensearch-v2-cluster.yaml
securityconfig-secret.yaml

```
apiVersion: opensearch.opster.io/v1
kind: OpenSearchCluster
metadata:
  name: my-first-cluster
  namespace: default
spec:
  general:
    serviceName: my-first-cluster
    version: 2.3.0
  dashboards:
    enable: true
    version: 2.3.0
    replicas: 1
    resources:
      requests:
         memory: "512Mi"
         cpu: “200m"
...
```

Configuring OpenSearchCluster

Configuring node pools

Separate StatefulSet
for each node pool

```
spec:
  nodePools:
    - component: masters
      replicas: 3  # The number of replicas
      diskSize: "30Gi" # The disk size to use
      resources: # The resource requests and limits for that nodepool
        requests:
          memory: "2Gi"
          cpu: "500m"
        limits:
          memory: "2Gi"
          cpu: "500m"
      roles: # The roles the nodes should have
        - "cluster_manager"
        - "data"
    - component: nodes
      replicas: 3
      diskSize: "10Gi"
      nodeSelector:
      resources:
        requests:
          memory: "2Gi"
          cpu: "500m"
        limits:
          memory: "2Gi"
          cpu: "500m"
      roles:
        - "data"
```

Configuring OpenSearchCluster

Configuring TLS on transport level

Two ways:

1. Let operator generate certificates
    - generate: true
    - perNode: true/false

2. Import your own certificates
    - tls.transport.generate: false
    - tls.transport.perNode: true/false
    - tls.transport.caSecret.name
    - tls.transport.nodesDn
    - tls.transport.adminDn
    - tls.transport.secret.name
    - config.adminSecret.name

```
spec:
  security:
    tls:  # Everything related to TLS configuration
      transport:  # Configuration of the transport endpoint
        generate: true  # Have the operator generate and sign certificates
        perNode: true  # Separate certificate per node
        secret:
          name:  # Name of the secret that contains the provided certificate
        caSecret:
          name:  # Name of the secret that contains a CA the operator should use
        nodesDn: [ ]  # List of certificate DNs allowed to connect
        adminDn: [ ]  # List of certificate DNs that should get admin access
```

One certificate per node:
- all in one secret: ca.crt, <hostname>.key, <hostname>.crt

(hostname: <cluster-name>-<nodepool-component>-<index>)

Configuring OpenSearchCluster

Configuring TLS on http level

Similar to transport level
- no “perNode” configuration available

If importing your own certificate add to SubjectAltNames (SAN):
 - <cluster-name>
 - <cluster-name>.<namespace>
 - <cluster-name>.<namespace>.svc
 - <cluster-name>.<namespace>.svc.cluster.local

```
spec:
  security:
    tls:  # Everything related to TLS configuration
      http:  # Configuration of the HTTP endpoint
        generate: true  # Have the Operator generate and sign certificates
        secret:
          name:  # Name of the secret that contains the provided certificate
        caSecret:
          name:  # Name of the secret that contains a CA the Operator should use
```

euatra

Configuring OpenSearchCluster

Additional opensearch.yml configuration

general vs nodePools

```
spec:
  general:
    # ...
    additionalConfig:
      some.config.option: somevalue
  # ...
  nodePools:
  - component: masters
    # ...
    additionalConfig:
      some.other.config: foobar
```

*These configs will be added as environment variable
And will take precedence over the values in opensearch.yml file inside the container

Configuring OpenSearchCluster

Adding security configuration

There are 2 ways to configure users, roles, tenants etc

1. Creating and mounting a secret with all security configuration
2. Using operator resources

*Currently both ways cannot be used at the same time

Configuring OpenSearchCluster

Creating a secret with all security configuration

secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: securityconfig-secret
type: Opaque
stringData:
      action_groups.yml: |-
         _meta:
           type: "actiongroups"
           config_version: 2
      internal_users.yml: |-
        _meta:
          type: "internalusers"
          config_version: 2
        admin:
...
```

opensearchcluster.yaml

```
spec:
  security:
    config:  # Everything related to the securityconfig
      securityConfigSecret:
        name:  securityconfig-secret # Name of the secret that contains the securityconfig files
      adminSecret:
        name:  # Name of a secret that contains the admin client certificate
      adminCredentialsSecret:
        name:  # Name of a secret that contains username/password for admin access
```

Configuring OpenSearchCluster

Using operator resources to configure security

user

```
apiVersion: opensearch.opster.io/v1
kind: OpensearchUser
metadata:
  name: sample-user
  namespace: default
spec:
  opensearchCluster:
    name: my-first-cluster
  passwordFrom:
    name: sample-user-password
    key: password
  backendRoles:
  - kibanauser
```

role

```
apiVersion: opensearch.opster.io/v1
kind: OpensearchRole
metadata:
  name: sample-role
  namespace: default
spec:
  opensearchCluster:
    name: my-first-cluster
  clusterPermissions:
  - cluster_composite_ops
  - cluster_monitor
  indexPermissions:
  - indexPatterns:
    - logs*
    allowedActions:
    - index
    - read
```

role mapping

```
apiVersion: opensearch.opster.io/v1
kind: OpensearchUserRoleBinding
metadata:
  name: sample-urb
  namespace: default
spec:
  opensearchCluster:
    name: my-first-cluster
  users:
  - sample-user
  backendRoles:
  - sample-backend-role
  roles:
  - sample-role
```

*Secret need to exist in the `default` namespace with the base64 encoded password in the `password` key

Configuring OpenSearchCluster

vm.max_map_count

Needs to be set to at least 262144

- creates initContainer on each pod using docker.io/busybox:latest
- executes sysctl command

initContainers used as helpers throughout
OpenSearch deployment can be disabled using config:

```
spec:
  general:
    setVMMaxMapCount: true
```

```
manager:
  extraEnv:
    - name: SKIP_INIT_CONTAINER
      value: "true"
```

Configuring OpenSearchCluster

Configuring Dashboards

```
spec:
  dashboards:
    enable: true  # Set this to true to enable the Dashboards deployment
    version: 2.3.0  # The Dashboards version to deploy. This should match the configured opensearch
version
    replicas: 1  # The number of replicas to deploy

    additionalConfig:
      opensearch_security.auth.type: "proxy"
      opensearch.requestHeadersWhitelist: |
        ["securitytenant","Authorization","x-forwarded-for","x-auth-request-access-token"]
      opensearch_security.multitenancy.enabled: "true"
```

Variable substitution

```
spec:
  dashboards:
    env:
      - name: OPENID_CLIENT_SECRET
        valueFrom:
          secretKeyRef:
            name: dashboards-oidc-config
            key: client_secret
    additionalConfig:
      opensearch_security.openid.client_secret: "${OPENID_CLIENT_SECRET}"
```

Configuring OpenSearchCluster

Configuring Dashboards HTTPS

Similar to transport and http layer on OpenSearch configuration

```
spec:
  dashboards:
    enable: true  # Deploy Dashboards component
    tls:
      enable: true  # Configure TLS
      generate: true  # Have the Operator generate and sign a certificate
      secret:
        name:  # Name of the secret that contains the provided certificate
      caSecret:
       name:  # Name of the secret that contains a CA the Operator should use
```

*To expose Dashboards externally use of valid certificate from an accredited CA (LetsEncrypt) with ingress is recommended

Configuring OpenSearchCluster

Additional configurations options:

- Adding plugins
- Add secrets to keystore
- Set Java heap size
- Configuring Snapshot Repositories
- Data Persistence
- Security Context for pods and containers
- Labels or Annotations on OpenSearch nodes
- Priority class on OpenSearch nodes
- Additional Volumes
- Adding environment variables to pods
- Custom cluster domain name
- Custom init helper
- Edit init container resources
- PodDisruptionBudget
- Exposing OpenSearch Dashboards
- Configuring the Dashboards K8s Service
- Customizing probe timeouts and thresholds
- Configuring Resource Limits/Requests
- Rolling Upgrades
- Volume Expansion
- User and role management
- SmartScaler

Rolling update using operator

It may seem that the rolling updates are performed similar to helm charts at first glance,
however there are several key differences.

- Shard allocation is disabled on each node prior to restarting it.
- Cluster health is checked prior to moving to the next node

Changing opensearch.yml configuration on an already installed cluster will be detected by the operator and it trigger a rolling restart of all cluster nodes to apply the new configuration. Similar mechanism is set up for nodepool-specific configuration like `resources`, `annotation` or `labels`.

Downgrades and upgrades that span more than one major version are not supported

Certificate expiration

Certificates generated by operator are valid for one year.

Currently there is no certificate renewal implemented, therefore make sure you have tested your implementation of certificate renewals before going to production.

When certificates are updated in secrets, they are automatically loaded into containers.
There is an option to perform hot reloading by adding line
`plugins.security.ssl_cert_reload_enabled: true` to opensearch.yml file
and using admin certificate to call the reload APIs for transport and http layers.

Exposing Dashboards Externally with Ingress

Ingress.yml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-opensearch
  namespace: default
  annotations:
    nginx.ingress.kubernetes.io/backend-protocol: "HTTPS"
spec:
  ingressClassName: nginx
  tls:
  - hosts:
    - eliatra-support.co.uk
    secretName: tls-for-dashboards-key-pair
  rules:
  - host: eliatra-support.co.uk
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: dashboards-opensearch-dashboards
            port:
              number: 5601
```

Summary

What we went through

Creating and configuring EKS cluster

Deploying Cert Manager

Creating Self Signed CA and individual certificates

Using Let’s Encrypt to generate all certificates

Configuring and deploying OpenSearch using helm

Configuring and deploying OpenSearch using operator

Rolling update

Get in Touch

info@eliatra.com

eliatra

eliatra_ire

Thank You!