

RadonDB DMP 数据库管理平台 v2.0

OpenSearch

文档版本：01

发布日期：2023-11-21

【版权声明】

版权所有 © 北京青云科技股份有限公司 2023。保留所有权利。

未经本公司书面许可，任何单位和个人不得摘抄、复制或以其他方式传播本文档的部分或全部内容。

【商标声明】

 和其他青云商标均由北京青云科技股份有限公司拥有。

本文档提及的其他商标或注册商标，由各自的所有人拥有。

【产品或服务声明】

本文档描述的部分产品、服务和特性可能不在您的购买或使用范围之内，您购买的产品、服务或特性应受青云公司商业合同和条款的约束。

本文档内容会不定期进行更新。本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

北京青云科技股份有限公司

网址：<https://www.qingcloud.com>

前言

本节介绍如何管理 RadonDB OpenSearch 数据库。

RadonDB OpenSearch 是一款灵活且可扩展的搜索和分析引擎。提供实时日志处理、全文搜索和数据分析等能力。

在 KubeSphere 企业版 Web 控制台安装“数据库管理平台”扩展组件后，[进入数据库管理平台](#)，即可以容器化的方式部署 OpenSearch 数据库，并通过图形界面对 OpenSearch 数据库进行管理。

目录

1. 创建 OpenSearch 实例.....	1
2. 查看运行状态.....	3
3. 访问 OpenSearch 数据库.....	5
4. 修改数据库配置参数.....	7
5. 扩展数据库节点.....	9
6. 调整 CPU 和内存资源.....	10
7. 扩展卷.....	11
8. 删除 OpenSearch 实例.....	12
9. 配置 OpenSearch 集群证书.....	13
9.1. 配置 OpenSearch 集群客户端证书.....	13
9.2. 配置 OpenSearch 集群服务端证书.....	17
10. 对接 Logstash 和 Kafka.....	29

1. 创建 OpenSearch 实例

本节介绍如何在 DMP 数据库管理平台上创建 RadonDB OpenSearch 实例。

前提条件

- 在 KubeSphere 企业版平台加入一个项目。
- KubeSphere 企业版平台需要安装“数据库管理平台”扩展组件。

操作步骤


- 以 platform-admin 角色登录 KubeSphere 企业版 Web 控制台并进入数据库管理平台。
- 在左侧导航栏选择 **OpenSearch**。
- 在页面右侧选择**新建 OpenSearch 实例**。
- 在**基本信息**页签，设置 RadonDB OpenSearch 实例的基本信息和位置，然后点击**下一步**。

参数	描述
名称	OpenSearch 实例的名称。名称只能包含小写字母、数字和连字符（-），必须以小写字母开头并以小写字母或数字结尾，最长 32 个字符。
描述	OpenSearch 实例的描述信息。描述可包含任意字符，最长 256 个字符。
位置	OpenSearch 实例所属的企业空间和项目。

- 在**应用设置**页签，设置 OpenSearch 实例的参数。

参数	描述
版本	OpenSearch 应用的版本。
内核版本	OpenSearch 数据库的内核版本。
主节点/热节点/温节点/冷节点/控制面板节点/Logstash 节点设置	<p>资源：分配给每个节点的 CPU 和内存资源。</p> <p>主节点/热节点/温节点/冷节点/控制面板节点/Logstash 节点数量：OpenSearch 实例各类型节点的数量。主节点的数量默认为 3 且不能修改，其他节点数量可根据需要设置。</p> <p>存储类：节点使用的存储系统对应的存储类。默认值为 local。如果下拉列表中没有符合需要的存储类，请联系平台管理员创建存储类。</p> <p>卷：每个节点使用的卷大小，单位为 GiB。每个主节点的卷大小默认为 20 GiB。控制面板节点和 Logstash 节点不能挂载卷。其他节点的卷大小可根据需要设置。</p>

点击**高级设置**，设置数据库的用户名和密码（默认用户名为 admin，密码为 RadonDB@123），启用或禁用**反亲和性配置**和 **node 亲和设置**。

设置	描述
反亲和性配置	开启反亲和性配置后， pod 将必须调度到不同的物理节点。
node 亲和设置	<p>开启 node 亲和设置后，添加匹配 Node 的键值对，可以将 pod 部署在特定条件的 node 上。</p> <ul style="list-style-type: none">• 点击添加可以设置多个键值对。• 在已添加的键值对右侧点击  可删除键值对。

提示

您还可以打开**应用设置**页签右上角的**编辑 YAML**，在 yaml 文件中自定义配置。

6. 点击**确定**。RadonDB OpenSearch 实例创建后，将显示在 OpenSearch 页面的实例列表中。

2. 查看运行状态

本节介绍如何查看 RadonDB OpenSearch 实例的运行状态。

前提条件

- 已创建 RadonDB OpenSearch 实例。更多信息，请参阅[创建 OpenSearch 实例](#)。

操作步骤

- 以 platform-admin 角色登录 KubeSphere 企业版 Web 控制台并进入数据库管理平台。
- 在左侧导航栏选择 **OpenSearch**。

已创建的 OpenSearch 实例将显示在 OpenSearch 实例列表中。

参数	描述
名称	OpenSearch 实例在 KubeSphere 企业版平台的名称。
状态	OpenSearch 实例的运行状态。 <ul style="list-style-type: none">创建中：正在创建。更新中：正在更新配置。已完成：更新完成，即将开始运行。失败：创建过程中出现错误。运行中：运行正常。
企业空间	OpenSearch 实例所属的企业空间。
项目	OpenSearch 实例所属的项目。
内核版本	OpenSearch 数据库的内核版本。
创建时间	OpenSearch 实例的创建时间。

- 在 OpenSearch 实例列表中，点击一个实例名称打开其详情页面。
- 在详情页面左侧的**详情**区域查看其资源属性。

参数	描述
集群	OpenSearch 实例所属的集群。
企业空间	OpenSearch 实例所属的企业空间。
项目	OpenSearch 实例所属的项目。
名称	OpenSearch 实例在 KubeSphere 企业版平台的名称。
状态	OpenSearch 实例的运行状态。
应用模板	创建 OpenSearch 实例所使用的模板类型。
内核版本	OpenSearch 数据库的内核版本。
创建时间	OpenSearch 实例的创建时间。
访问地址	OpenSearch 数据库的访问地址。每个数据库可提供多个不同用途的访问地址。

5. 在详情页面右侧的**概览**页签，查看数据库的基本信息和节点。

参数	描述
基本信息	显示 OpenSearch 数据库的运行状态、各节点数量。
节点	显示 OpenSearch 数据库的所有节点及其创建时间、运行状态、容器组 IP 地址、节点类型、CPU 和内存资源、卷容量。 每个节点被创建为一个容器组。

3. 访问 OpenSearch 数据库

Dashboards 是 OpenSearch 的可视化工具，提供面向用户的 OpenSearch 插件管理服务，包括安全性、警报、索引状态管理、SQL 管理等插件。

本节介绍如何访问 OpenSearch 数据库的 Dashboards。

前提条件

- 如果使用外部读写地址，取决于您的网络环境，您可能需要配置流量转发和防火墙规则。有关具体操作请联系您的网络环境管理员。
- RadonDB OpenSearch 实例状态为**运行中**。
- 已获取 RadonDB OpenSearch 数据库用户名和密码。如果您在创建 RadonDB OpenSearch 实例时未修改过用户名和密码，此处默认用户名为 **admin**，默认密码为 **RadonDB@123**。如果您修改过用户名和密码，请使用修改后的用户名和密码登录。

操作步骤

1. 以 platform-admin 角色登录 KubeSphere 企业版 Web 控制台并进入项目。
2. 点击**应用负载 > 服务**，搜索并点击新建的 OpenSearch 实例的 Dashboards 服务进入详情页面。
3. 点击**更多操作 > 编辑外部访问**。
4. 将 Dashboards 服务的访问模式设置为 **NodePort**，点击**确定**。
5. 进入 DMP 数据库管理平台，在左侧导航栏点击 **OpenSearch**，然后点击已创建的 OpenSearch 实例进入详情页面。
6. 在页面左侧的**访问地址**区域，查看控制面板地址。
7. 在浏览器输入 http://dashboard_IP:dashboard_port/ 进入 OpenSearch Dashboards 登录页面，输入用户名和密码。



Please login to OpenSearch Dashboards

If you have forgotten your username or password, please ask your system administrator



Log In

备注

- 请将 **dashboard_IP** 和 **dashboard_port** 替换成实际的值。
- 取决于 Kubernetes 集群的部署位置，您可能需要在安全组中放行端口并配置相关的端口转发规则。

4. 修改数据库配置参数

本节介绍如何修改 RadonDB OpenSearch 数据库的配置参数。

RadonDB OpenSearch 实例支持编辑配置参数，通过管理配置参数可调优数据库性能，并可启用数据库高可用性能。

警告

修改部分配置参数后，OpenSearch 服务会重启，造成业务中断。请谨慎操作。

前提条件

- RadonDB OpenSearch 实例状态为**运行中**。

修改配置参数

- 以 platform-admin 角色登录 KubeSphere 企业版 Web 控制台并进入数据库管理平台。
- 在左侧导航栏选择 **OpenSearch**。
- 在 OpenSearch 实例列表中，点击一个实例名称打开其详情页面。
- 点击页面右侧的**参数管理**页签，点击**编辑**即可修改配置参数。
- 点击**确定**保存。

参数简介

参数	说明
thread_pool.write.queue_size	线程池写队列的大小。取值范围为 1000 ~ 20000，默认值为 10000 。
thread_pool.search.queue_size	线程池搜索队列的大小。取值范围为 1000 ~ 20000，默认值为 1000 。
action.destructive_requires_name	是否在删除索引时禁止使用通配符和 _all。您可以将此参数设置为 false，以允许使用通配符和 _all。默认值为 true 。
http.cors.enabled	是否启用跨域资源共享（CORS）。默认值为 false 。
http.cors.allow-origin	可用于跨域资源共享（CORS）的域（origin）。
indices fielddata.cache.size	可用作字段数据缓存的最大堆内存。默认值为 -1b 。
indices.memory.index_buffer_size	分配给节点并由所有分片作为索引缓冲区共享的堆内存大小。参数值可以为百分比或字节数。默认值为 10% 。
indices.queries.cache.size	过滤器缓存的内存大小。取值可以为百分比（例如 5%）或确切值（例如 512mb）。默认值为 10%。默认值为 10% 。
indices.requests.cache.size	在节点级管理的分片请求缓存大小。默认值为堆内存的 1%。
reindex.remote.whitelist	列入白名单以进行 Reindex 操作的远程节点的地址。您可以用使用半角逗号（,）分隔多个地址（例如 otherhost:9200, 192.168.1.:9200, localhost:）。

5. 扩展数据库节点

本节介绍如何扩展 RadonDB OpenSearch 数据库的节点。

警告

- 若选择节点数低于当前节点数，将删除超出的节点。
- 扩展节点可能会中断数据库服务，请在非高峰时段执行此操作。

前提条件

- RadonDB OpenSearch 实例状态为**运行中**。

操作步骤

1. 以 platform-admin 角色登录 KubeSphere 企业版 Web 控制台并进入数据库管理平台。
2. 在左侧导航栏选择 **OpenSearch**。
3. 在 OpenSearch 实例列表中，点击一个实例名称打开其详情页面。
4. 在页面左上角选择**更多操作 > 扩展**。
5. 在**扩展**对话框，选择节点角色并设置节点的数量，然后点击**确定**。待实例状态切换为**运行中**，则新增节点完成。

6. 调整 CPU 和内存资源

本节介绍如何调整 RadonDB OpenSearch 数据库的 CPU 和内存资源。

警告

此操作可能会中断数据库服务，请在非高峰时段执行此操作。

前提条件

- RadonDB OpenSearch 实例状态为**运行中**。

操作步骤

1. 以 platform-admin 角色登录 KubeSphere 企业版 Web 控制台并进入数据库管理平台。
2. 在左侧导航栏选择 **OpenSearch**。
3. 在 OpenSearch 实例列表中，点击一个实例名称打开其详情页面。
4. 在页面左上角选择**更多操作 > 调整资源**。
5. 在**调整资源**对话框，选择节点角色，设置分配给每个节点的 CPU 和内存资源数量，然后点击**确定**。

7. 扩展卷

本节介绍如何扩展挂载到数据库节点的卷。

备注

- 目前不支持对使用 KubeSphere 企业版集群本地存储创建的卷（local）进行扩展。
- 目前仅支持增加卷容量，不支持减少卷容量。
- 此操作可能会中断数据库服务，请在非高峰时段执行此操作。

前提条件

- RadonDB OpenSearch 实例状态为**运行中**。

操作步骤

1. 以 platform-admin 角色登录 KubeSphere 企业版 Web 控制台并进入数据库管理平台。
2. 在左侧导航栏选择 **OpenSearch**。
3. 在 OpenSearch 实例列表中，点击一个实例名称打开其详情页面。
4. 在页面左上角选择**更多操作 > 扩展卷**。
5. 在**扩展卷**对话框，选择节点角色，设置卷的容量，然后点击**确定**。

8. 删除 OpenSearch 实例

本节介绍如何删除 RadonDB OpenSearch 实例。

警告

此操作将删除 RadonDB OpenSearch 数据库和数据库中的所有数据，请谨慎执行此操作。

前提条件

- 已创建 RadonDB OpenSearch 实例。
- 为避免数据丢失，请提前备份数据。

操作步骤

删除单个实例

1. 以 platform-admin 角色登录 KubeSphere 企业版 Web 控制台并进入数据库管理平台。
2. 在左侧导航栏选择 **OpenSearch**。
3. 在 RadonDB OpenSearch 实例右侧点击 **:**，然后在下拉列表中选择**删除**。
4. 在**删除**对话框，输入 RadonDB OpenSearch 实例的名称，然后点击**确定**。

批量删除实例

1. 以 platform-admin 角色登录 KubeSphere 企业版 Web 控制台并进入数据库管理平台。
2. 在左侧导航栏选择 **OpenSearch**。
3. 选择需要删除的实例左侧的复选框，然后在列表上方点击**删除**。
4. 在确认删除对话框中，输入要删除的实例名称，点击**确定**。

备注

请使用半角逗号 (,) 和空格分隔多个名称。

9. 配置 OpenSearch 集群证书

OpenSearch 集群自生成的证书不支持在 Web 页面上下载，可通过 secret 手动获取证书。本章介绍如何配置 OpenSearch 集群服务端和客户端证书。

9.1. 配置 OpenSearch 集群客户端证书

本节介绍如何对接配置 OpenSearch 集群客户端证书。

创建 OpenSearch 集群

1. 修改 `opensearch.yaml` 文件。

a. 在 `spec.general.additionalConfig` 区域，增加 `plugins.security.ssl.http.clientauth_mode`:

OPTIONAL 配置。

b. 集群开启 TLS 认证。

```
apiVersion: opensearch.opster.io/v1 kind: OpenSearchCluster
metadata:
  name: os-001 spec:
  general:
    version: 2.3.0 httpPort: 9200 vendor: opensearch serviceName: os-001 additionalConfig:
    plugins.security.ssl.http.clientauth_mode: OPTIONAL dashboards:
    version: 2.3.0 enable: true replicas: 1 resources:
    requests: memory: "1Gi" cpu: "500m"
    limits:
    memory: "1Gi" cpu: "500m"
    tls:
    enable: true generate: true
    confMgmt: smartScaler: true
    security: config:
    securityConfigSecret:
    # Pre create this secret with required security configs, to override the default
    settings
    name: securityconfig-secret adminCredentialsSecret:
    name: admin-credentials-secret tls:
    transport: generate: true
    http:
    generate: true
    nodePools:
    - component: masters
    replicas: 3 diskSize: "50Gi" persistence:
    pvc:
    storageClass: csi-qingcloud accessModes:
    - ReadWriteOnce nodeSelector:
```

```
resources: requests:
memory: "1Gi"
cpu: "500m" limits:
memory: "1Gi"
cpu: "500m" roles:
- "cluster_manager"
- component: data-hot replicas: 2
diskSize: "80Gi" persistence:
pvc:
storageClass: csi-qingcloud accessModes:
- ReadWriteOnce additionalConfig:
node.attr.datatier: "hot" # warm, cold resources:
requests: memory: "1Gi" cpu: "500m"
limits:
memory: "1Gi" cpu: "500m"
roles:
- "data"
--- apiVersion: v1 kind: Secret metadata:
name: admin-credentials-secret type: Opaque
data:
username: c2h1YWliMTIz password: c2h1YWliMTIz
```

2. 在 `securityconfig.yaml` 中增加 `clientcert_auth_domain` 配置。

```
config.yml: |- _meta:
type: "config"
config_version: "2" config:
dynamic: http:
anonymous_auth_enabled: false authc:
basic_internal_auth_domain: http_enabled: true transport_enabled: true order: "4"
http_authenticator:
type: basic
challenge: true authentication_backend:
type: internal clientcert_auth_domain:
http_enabled: true transport_enabled: true order: 1 http_authenticator:
type: clientcert #config:
#username_attribute: cn #optional, if omitted DN becomes username challenge: false
authentication_backend: type: noop
```

3. 执行以下命令创建集群。

```
root@master1:/home/ubuntu/opensearch# kubectl -n opensearch apply -f ./security-
config.yaml
root@master1:/home/ubuntu/opensearch# kubectl -n opensearch apply -f ./os-ssl-
true.yaml
```

生成客户端证书

1. 获取 CA 证书。集群自生成的 CA 证书名称为 `<cluster-name>-ca`。

```
root@master1:/home/ubuntu/opensearch# kubectl -n opensearch get secret os-001-ca -
oyaml
echo <ca.crt 内容> | base64 -d > root-ca.pem
echo <ca.key 内容> | base64 -d > root-ca-key.pem
```

2.

2. 签发客户端证书。

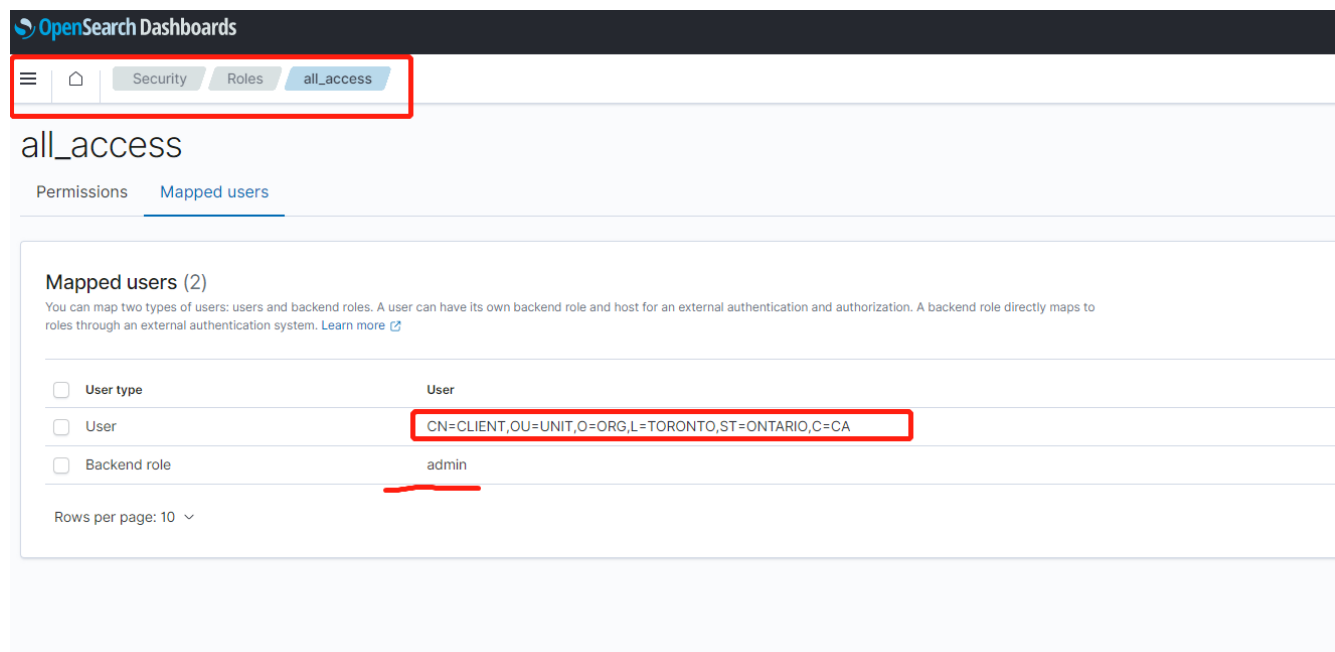
```
openssl genrsa -out client-key-temp.pem 2048
openssl pkcs8 -inform PEM -outform PEM -in client-key-temp.pem -topk8 -nocrypt -v1
PBE-SHA1-3DES -out client-key.pem
openssl req -new -key client-key.pem -subj
"/C=CA/ST=ONTARIO/L=TORONTO/O=ORG/OU=UNIT/CN=CLIENT" -out client.csr
openssl x509 -req -in client.csr -CA root-ca.pem -CAkey root-ca-key.pem -
CAcreateserial -sha256 -out client.pem -days 730
```

其中 **-subj** 后的内容可以自定义。

3. 获取客户端证书的专有名称（DN）。

```
root@master1:/home/ubuntu/opensearch/certs# openssl x509 -subject -nameopt RFC2253 -
noout -in client.pem
subject=CN=CLIENT,OU=UNIT,O=ORG,L=TORONTO,ST=ONTARIO,C=CA
```

4. 在 OpenSearch Dashboards 将在上一步获得的映射给用户。



5. 使用证书访问集群。

```
root@i-abmm2sr0:~/test# curl -k --cert client.pem --key client-key.pem
https://172.22.9.17:30201/_cat/nodes?pretty
```

其中：

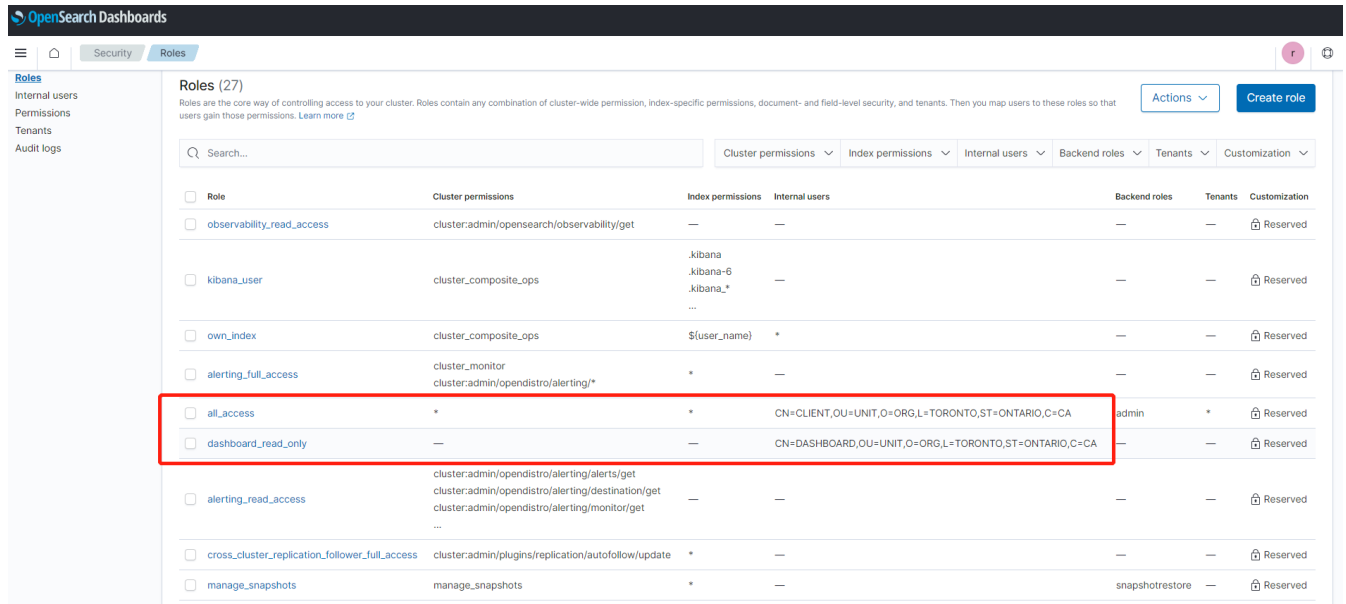
- **-k**：忽略服务端证书验证。
- **--cert**：指定客户端证书。
- **--key**：指定客户端证书的 key。
- **172.22.9.17**：任意节点 IP。
- **30201**：9200 暴露在集群外部的端口号。

多用户访问集群

用户使用客户端证书绑定的 role 来获取相应的权限，根据实际情况，不同用户可配置不同的 role，同时生成对应的证书访问集群。

下面举例如何将生成的证书绑定在 all_access 和 dashboard_read_only 两个 role 上。关于如何生成证书，请参阅[生成客户端证书](#)。

1. 进入 OpenSearch Dashboards 管理页面，在左侧导航栏选择 **Security** > **Roles**。
2. 在右侧的 **Roles** 页面，点击 **Create role**，输入角色名称并设置权限，点击 **Submit**。



3. 测试绑定是否成功。
 - a. 使用 all_access 证书访问集群，如果出现以下信息，则代表访问集群成功。

```
root@i-abmm2sr0:~/test# curl -k --cert client.pem --key client-key.pem https://172.22.9.17:30201/_cat/nodes?pretty
10.10.2.141 46 99 9 0.92 0.93 1.33 d data - os-001-data-hot-1
10.10.215.154 38 94 4 0.86 0.83 0.79 m cluster_manager * os-001-masters-1
10.10.215.153 34 96 4 0.86 0.83 0.79 d data - os-001-data-hot-0
10.10.194.133 53 98 5 0.95 1.10 1.27 m cluster_manager - os-001-masters-0
10.10.194.134 41 98 7 0.95 1.10 1.27 m cluster_manager - os-001-masters-2
```

- b. 使用 dashboard_read_only 证书访问 Dashboard，如果出现以下信息，则代表访问集群成功。

```
root@i-abmm2sr0:~/test# curl -k --cert dashboard.pem --key dashboard-key.pem https://172.22.9.17:32053
root@i-abmm2sr0:~/test# curl -k --cert dashboard.pem --key dashboard-key.pem https://172.22.9.17:32053
root@i-abmm2sr0:~/test#
```

- c. 使用 dashboard_read_only 证书访问集群。如果出现报错，则符合预期。

```
root@i-abmm2sr0:~/test# curl -k --cert dashboard.pem --key dashboard-key.pem https://172.22.9.17:30201/_cat/health?pretty
{
  "error": {
    "root_cause": [
      {
        "type": "security_exception",
        "reason": "no permissions for [cluster:monitor/health] and User [name=CN=DASHBOARD,OU=UNIT,O=ORG,L=TORONTO,ST=ONTARIO,C=CA, backend_roles=[], requestedTenant=null]"
      }
    ],
    "type": "security_exception",
    "reason": "no permissions for [cluster:monitor/health] and User [name=CN=DASHBOARD,OU=UNIT,O=ORG,L=TORONTO,ST=ONTARIO,C=CA, backend_roles=[], requestedTenant=null]"
  },
  "status": 403
}
```

9.2. 配置 OpenSearch 集群服务端证书

本节介绍如何配置 OpenSearch 集群服务端证书。

服务端证书有以下两种生成方式：

- OpenSearch 集群内部自动生成证书(默认)，具有以下特点：
 - 有效期为 10 年。
 - 暂不支持证书更换或续签。
 - 该证书仅能在 Kubernetes 集群内部使用，无法用于 Kubernetes 外部访问。
- 用户自定义证书，具有以下特点：
 - 自定义有效期。
 - 暂不支持证书更换或续签。
 - 该证书可用于从 Kubernetes 集群外部访问 OpenSearch 集群。

OpenSearch 集群内部自动生成证书

1. 在 OpenSearch Dashboards 上开启 TLS 认证。

- 将 **spec.dashboards.tls.enable** 的值设置为 **true**。
- 将 **spec.dashboards.tls.generate** 的值设置为 **true**。

2. 在 OpenSearch 集群上开启 TLS 认证。

- 将 **spec.security.tls.transport.generate** 的值设置为 **true**。
- 将 **spec.security.tls.http.generate** 的值设置为 **true**。

配置文件如下所示：

```
apiVersion: opensearch.opster.io/v1
kind: OpenSearchCluster
metadata:
  name: os-001
spec:
  general:
    version: 2.3.0
    httpPort: 9200
    vendor: opensearch
    serviceName: os-001

  dashboards:
    version: 2.3.0
    enable: true
    replicas: 1
    resources:
      requests:
        memory: "1Gi"
        cpu: "500m"
```

```
memory: "1Gi"
  cpu: "500m"
  tls:
    enable: true
    generate: true

confMgmt:
  smartScaler: true
security:
  config:
    securityConfigSecret:
# Pre create this secret with required security configs, to override the default
settings
  name: securityconfig-secret
  adminCredentialsSecret:
    name: admin-credentials-secret
  tls:
    transport:
      generate: true

  http:
    generate: true

nodePools:
- component: masters
  replicas: 3
  diskSize: "50Gi"
  persistence:
    pvc:
      storageClass: csi-qingcloud
      accessModes:
        - ReadWriteOnce
  nodeSelector:
  resources:
    requests:
      memory: "1Gi"
      cpu: "500m"
    limits:
      memory: "1Gi"
      cpu: "500m"
  roles:
    - "cluster_manager"

- component: data-hot
  replicas: 2
  diskSize: "80Gi"
  persistence:
    pvc:
      storageClass: csi-qingcloud
      accessModes:
        - ReadWriteOnce
  additionalConfig:
```

```
node.attr.datatier: "hot" # warm, cold
resources:
  requests:
    memory: "1Gi"
    cpu: "500m"
  limits:
    memory: "1Gi"
    cpu: "500m"
roles:
  - "data"

---
apiVersion: v1
kind: Secret
metadata:
  name: admin-credentials-secret
type: Opaque
data:
  username: c2h1YWliMTIz
  password: c2h1YWliMTIz
```

3. 在 master 节点上执行以下命令创建集群。

```
root@master1:/home/ubuntu/opensearch# kubectl -n opensearch apply -f ./os-ssl-true.yaml
```

4. 在 master 节点执行以下命令，并访问 <https://cluster-name.namespace:9200> 验证 OpenSearch 证书是否可用。

```
root@master1:/home/ubuntu/opensearch# kubectl -n opensearch exec -it pod/os-001-masters-0 -c opensearch -- /bin/bash [opensearch@os-001-masters-0 ~]$ curl https://os-001.opensearch.svc:9200/_cat/nodes -u root:U61yyB86o2QbyyFo --cacert /usr/share/opensearch/config/tls-http/ca.crt
10.10.194.117 40 93 7 1.03 1.04 1.02 d data - os-001-data-hot-1
10.10.215.129 46 88 9 0.52 0.65 0.84 m cluster_manager - os-001-masters-1
10.10.215.128 51 94 5 0.52 0.65 0.84 d data - os-001-data-hot-0 10.10.2.137 45 100 8
1.98 1.21 1.12 m cluster_manager * os-001-masters-0 10.10.194.118 44 88 17 1.03 1.04
1.02 m cluster_manager - os-001-masters-2
```

此处 `--cacert` 指定了 CA 证书的路径 `/usr/share/opensearch/config/tls-http/ca.crt`。

5. 在 Dashboard 节点执行以下命令，并访问 <https://cluster-name-dashboards:5601> 验证 Dashboard 证书是否可用。

```
root@master1:/home/ubuntu/opensearch# kubectl -n opensearch exec -it os-001-dashboards-7d9886689d-zd7jb -- /bin/bash
[opensearch-dashboards@os-001-dashboards-7d9886689d-zd7jb ~]$ curl https://os-001-dashboards:5601 -u dashboarduser:dashboarduser
curl: (60) SSL certificate problem: unable to get local issuer certificate
More details here: https://curl.se/docs/sslcerts.html

curl failed to verify the legitimacy of the server and therefore could not
establish a secure connection to it. To learn more about this situation and
```

```
how to fix it, please visit the web page mentioned above.
[opensearch-dashboards@os-001-dashboards-7d9886689d-zd7jyb ~]$ curl https://os-001-
dashboards:5601 -u dashboarduser:dashboarduser --cacert /usr/share/opensearch-
dashboards/certs/tls.crt
[opensearch-dashboards@os-001-dashboards-7d9886689d-zd7jyb ~]$
```

此处 **--cacert** 指定了 CA 证书的路径 `/usr/share/openssl/config/tls-http/ca.crt`。

集群内部生成的证书名称为 `<cluster-name>-ca` 和 `<cluster-name>-dashboards-cert`，如下图所示。

```
root@master1:/home/ubuntu/opensearch# kubectl -n opensearch get secrets
NAME                                TYPE                                DATA  AGE
admin-credentials-secret            Opaque                             2      2d4h
admin-default-secret               Opaque                             2      39m
default-token-679c3b7d             kubernetes.io/service-account-token 3      2d4h
os-001-admin-certificate            kubernetes.io/tls                  3      39m
os-001-ca                          Opaque                             2      39m
os-001-dashboards-cert             Opaque                             3      38m
os-001-http-cert                  kubernetes.io/tls                  3      39m
os-001-root-password              Opaque                             2      39m
os-001-transport-cert             kubernetes.io/tls                  3      39m
securityconfig-secret              Opaque                             8      39m
root@master1:/home/ubuntu/opensearch#
```

6. 执行以下命令获取 OpenSearch 集群 CA 证书并保存到 **ca-test.cert** 文件中。

```
root@master1:/home/ubuntu# kubectl -n opensearch get secrets os-001-ca -o jsonpath='{.data}' | jq -r '"ca.crt"' | base64 -d > test-ca.crt
```

```
root@master1:/home/ubuntu/openssl# kubectl -n openssl get secrets os-001-ca -ojsonpath='{.data}' | jq -r '"ca.crt"' | base64 -d > test-ca.crt
root@master1:/home/ubuntu/openssl# cat test-ca.crt
-----BEGIN CERTIFICATE-----
MIIE7zCCategAwIBAgIRAJa9J4GEdeZHs1yaYzKvp1K0mDQYJKoZIhvcNAQELBQAw
ETEPMA0GA1UEAxMGB3MtM0tAdXA0MDEzMDIyNDIyMDIyNDIyMDIyNDIyNDIyNDIy
MFowETEPMA0GA1UEAxMGB3MtM0tAdXA0MDEzMDIyNDIyMDIyNDIyNDIyNDIyNDIy
CgKCAgEAA1wLQCTSUUVIyaG94Do1kyJ0FPg7um/4vU67KhzbadQFeX79HTudCF10MP
Z8F58KS1Dec+SoHKLCPmFH8BDL5wajv9Zq7mG5z6u7YLApmHqnJ6+if09qveHZ
JG4faQd7qFgoJSR7D1nWkXqLRTpWdvsb2MygAP/ASFcmnhyqKkL74ek5o/6HRU
usijxQoFj1ESLgK9NzYd60PKQ0w4wFteo51Nyfg4M1K+7P3VCz0PS/cPxbzPlxow
939vcjh+v3aFPiQpGyB0RafxdX4mFGwia+1/Bqp2L8WNI1Tx0BwS3J051VmHeVAnQE
ao9ihFnW31h3SjEwud/CXjs5+xxY/EKjmtHEX6Jzz0XdwCg6EPLfPpJpPmNx8BDN
vm7QBLa31AasquUjBt911J/TmSP74zmIdtZ6cu0spa32ImgzF3Jot2V6C9z/r5JDM
7e20Y0qKHi1CX9kMaTpohdQEDKLQKkRkL5SeybjSE7CBZqDL+5vLnmm+Fa5fwG80Y
IeOX4GJV9h8QeAGNRR2TV45K3tXBKbvi5rkal7aHt13kgVUS4Ho6SAioqAmMqXg
Njpr/ot8K1RYQA0cJG6grzhi7AxosVNoYwlp7G01S548yzkSBBAdi6bTi4RxFyFZyx
WX3dYDRC4w1Mctf7mCyEocFe1Hrh5M+tmRJAG1JOIGHpqGhw4a0CAwEAEAACMEAw
DgYVR0PAAQ/BAQDAAGGMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0BBYEBori1icp
MahcLPj/yY2Lv3KHLA4YMA0GCqG5Ib3DQEBCCAAJATCAQDTayyINmQm60DSHrtG
ofyFxd88x6dboJSQzaxA/ue7r3pr4mRh+26056vdEDB1LK245kA/ZpG0XK56YG0S
AztvcMBRINo5Q/RGcXZrDnwF7TLWtJC/w81WpTbZyTZQZP6V1fpr9JUT071PbFz1
Lj2HXnCyTNSHhdkUwHmBz05mJKVQFC0yIF18bV3tuo0Iq05iIlu+8wEKw5zsxLDI
wUfrvuzdaJICEC2T3LEAm2Bvf4pUJF65MkKsR6qIgzHyZ3FxmT9G5q3W3h+Dcdk+
gdu4dIamv0TLj8M8mr18qW5s0L1Bg8D/EQEYxR1gHF2pd5i3BpkZmcx2y9R21e8g
LjF9hd60bvU4hkD67yuVVL1H1h25U821NImF96WFIUVVjyPM8fsuKkZoarRhDsUp
ua2H7uWiz5q+r8e10uQuB+tIjnngrYLUR87hjJmBJ96665TQgJuvVwkJor0bdkRqC
a2NMPcb85kfpDA6IILeFo6QMkMTvVEDk8xvBMQromoglvambudpqRkGgpb2x3TEEq
gBL0DcyT9eT5N8Xm1sGe6zMsUbgLP8bfnZC/p6z4sV3AcvJY07cp5PLHiNaFi0XN
nQ6z212dqU4fMw+u4k5DwKrqAqPfoE9R+/XMDejB7ETmRKCLzSGVh/sU0/sZ2ap1
D9gkU16fcDBYekwIqz38ux9gCw==
-----END CERTIFICATE-----
root@master1:/home/ubuntu/openssl#
```

用户自定义证书

1. 使用 OpenSSL 签发 Dashboard 和 OpenSearch 证书。

a. 规划证书。

项目	内容
CN	*.<cluster-name>.<namespace>.svc.cluster.local
SAN	*.<cluster-name>.<namespace>.svc.cluster.local .<cluster-name>-dashboards.

b. 创建 CA 证书 **root-ca.pem** 和 **root-ca-key.pem**。

```
openssl genrsa -out root-ca-key.pem 2048
openssl req -new -x509 -sha256 -key root-ca-key.pem -subj
"/C=CN/ST=STATE/L=CITY/O=ORG/OU=UNIT/CN=demo@nowhere.com" -out root-ca.pem -days
3650
```

c. 创建证书配置文件 **san.cnf**。

```
[req]
distinguished_name = req_distinguished_name
req_extensions = req_ext
prompt = no
[req_distinguished_name]
CN = *.os002.opensearch.svc.cluster.local
[req_ext]
subjectAltName = @alt_names
[alt_names]
DNS.1 = *.os002.opensearch.svc.cluster.local
DNS.2 = *.os002-dashboards.*
```

备注

此处集群名称 os002，namespace 名称 opensearch 仅为示例，请根据实际情况替换。

d. 创建证书 **my.crt** 和 **my.key**。

```
openssl req -out my.csr -newkey rsa:2048 -nodes -keyout my.key -config san.cnf
openssl x509 -req -in my.csr -CA root-ca.pem -CAkey root-ca-key.pem -CAcreateserial
-sha256 -out my.crt -days 36 50 -extensions req_ext -extfile san.cnf
```

e. 验证证书，含 CN 和 SAN 项。

```
openssl x509 -in my.crt -noout -text
```

```

root@i-abmm2sr0:~/test# openssl genrsa -out root-ca-key.pem 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....+++++
.....+++++
e is 65537 (0x010001)
root@i-abmm2sr0:~/test# openssl req -new -x509 -sha256 -key root-ca-key.pem -subj "/C=CN/ST=STATE/L=CITY/O=ORG/OU=UNIT/CN=demo@nowhere.com" -out root-ca.pem -days 3650
root@i-abmm2sr0:~/test# vim san.cnf
root@i-abmm2sr0:~/test# openssl req -out my.csr -newkey rsa:2048 -nodes -keyout my.key -config san.cnf
Generating a RSA private key
.....+++++
.....+++++
writing new private key to 'my.key'
-----
root@i-abmm2sr0:~/test# openssl x509 -req -in my.csr -CA root-ca.pem -CAkey root-ca-key.pem -CAcreateserial -sha256 -out my.crt -days 3650 -extensions req_ext -extfile san.cnf
Signature ok
subject=CN = *.os002.opensearch.svc.cluster.local
Getting CA Private Key
root@i-abmm2sr0:~/test# ls
my.crt my.csr my.key root-ca-key.pem root-ca.pem root-ca.srl san.cnf
root@i-abmm2sr0:~/test# openssl x509 -in my.crt -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            27:c3:5b:1c:1d:ae:aa:e8:4f:52:1d:57:35:e5:47:69:aa:67:46:2d
        Signature Algorithm: sha256withRSAEncryption
        Issuer: C = CN, ST = STATE, L = CITY, O = ORG, OU = UNIT, CN = demo@nowhere.com
        Validity
            Not Before: Feb 24 08:27:29 2023 GMT
            Not After : Feb 21 08:27:29 2033 GMT
        Subject: CN = *.os002.opensearch.svc.cluster.local
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public-Key: (2048 bit)
            Modulus:
                00:d8:a5:c5:1d:4f:c5:61:72:ea:00:35:8a:95:a1:
                ba:3e:a8:25:75:60:83:3e:27:69:1a:0d:14:bc:d8:
                5a:8a:45:40:7f:5e:81:ec:23:5d:6f:83:9e:2b:03:
                08:b3:9b:5e:2f:49:fe:b6:81:e9:41:7b:75:8c:28:
                37:07:5c:5e:13:44:b4:ab:39:f1:16:28:24:6f:4f:
                b9:4f:b0:3a:96:2d:5b:02:66:e0:9d:5d:7d:25:df:
                5e:20:54:e9:23:5d:58:73:7c:86:dc:54:bc:34:7d:
                1f:83:9a:01:26:ee:5b:c0:7c:09:32:6d:be:58:0e:

```

2. 通过 my.crt 和 my.key 创建 HTTP 证书。

```

root@master1:/home/ubuntu/opensearch/certs# kubectl -n opensearch create secret tls
cluster-http-cert --cert=my.crt --key=my.key
secret/cluster-http-cert created

```

3. 通过 root-ca.pem 创建 CA 证书。

```

root@master1:/home/ubuntu/opensearch/certs# kubectl -n opensearch create secret
generic cluster-ca --from-file=ca.crt=./root-ca.pem -- from-file=ca.key=./root-ca-
key.pem
secret/cluster-ca created

```

4. 参照以下内容配置 CR 文件。

- 开启 Dashboard TLS 认证
 - 将 `spec.dashboards.tls.enable` 设置为 `true`。
 - `spec.dashboards.tls.generate` 设置为 `false`。
 - `spec.dashboards.tls.secret` 设置为 HTTP 证书名称，此处为 `cluster-http-cert`。
 - `spec.dashboards.tls.caSecret` 设置为 CA 证书名称，此处为 `cluster-ca`。
- 开启 OpenSearch 集群 TLS 认证
 - `spec.security.tls.transport.generate` 设置为 `true`。
 - `spec.security.tls.transport.caSecret` 设置为 CA 证书名称，此处为 `cluster-ca`。
 - `spec.security.tls.http.generate` 设置为 `false`。
 - `spec.security.tls.http.caSecret` 设置为 CA 证书名称，此处为 `cluster-ca`。
 - `spec.security.tls.http.secret` 设置为 HTTP 证书名称，此处为 `cluster-http-cert`。

配置文件如下所示：

```
apiVersion: opensearch.opster.io/v1
kind: OpenSearchCluster
metadata:
  name: os002
spec:
  general:
    version: 2.3.0
    httpPort: 9200
    vendor: opensearch
    serviceName: os002

  dashboards:
    version: 2.3.0
    enable: true
    replicas: 1
    resources:
      requests:
        memory: "1Gi"
        cpu: "500m"
      limits:
        memory: "1Gi"
        cpu: "500m"
    tls:
      enable: true
      generate: false
      secret:
        name: cluster-http-cert
      caSecret:
        name: cluster-ca
  confMgmt:
    smartScaler: true
  security:
    config:
      securityConfigSecret:
# Pre create this secret with required security configs, to override the default
settings
      name: securityconfig-secret
      adminCredentialsSecret:
        name: admin-credentials-secret
    tls:
      transport:
        generate: true
        caSecret:
          name: cluster-ca
      http:
        generate: false
        secret:
          name: cluster-http-cert
        caSecret:
          name: cluster-ca

  nodePools:
```

```

- component: masters
  replicas: 3
  diskSize: "50Gi"
  persistence:
    pvc:
      storageClass: csi-qingcloud
      accessModes:
        - ReadWriteOnce
  nodeSelector:
  resources:
    requests:
      memory: "1Gi"
      cpu: "500m"
    limits:
      memory: "1Gi"
      cpu: "500m"
  roles:
    - "cluster_manager"

- component: data-hot
  replicas: 2
  diskSize: "80Gi"
  persistence:
    pvc:
      storageClass: csi-qingcloud
      accessModes:
        - ReadWriteOnce
  additionalConfig:
    node.attr.datatier: "hot" # warm, cold
  resources:
    requests:
      memory: "1Gi"
      cpu: "500m"
    limits:
      memory: "1Gi"
      cpu: "500m"
  roles:
    - "data"

---
apiVersion: v1
kind: Secret
metadata:
  name: admin-credentials-secret
type: Opaque
data:
  username: c2h1YWliMTIz
  password: c2h1YWliMTIz

```

5. 创建 OpenSearch 集群。

```
root@master1:/home/ubuntu/opensearch# kubectl -n opensearch apply -f os-ssl-false.yaml
```

6.

6. 验证 OpenSearch 集群证书。

```

root@master1:/home/ubuntu/opensearch# kubectl -n opensearch exec -it os002-masters-0 -
- /bin/bash
Defaulted container "opensearch" out of: opensearch, init (init)
[opensearch@os002-masters-0 ~]$ curl https://os002-masters-
0.os002.opensearch.svc.cluster.local:9200/_cat/nodes --cacert /usr/share
/opensearch/config/tls-http/ca.crt -u root:x804NqPd3tz7HBzr
10.10.2.138 63 95 17 2.16 1.68 1.42 m cluster_manager * os002-masters-0 10.10.194.120
49 94 6 4.88 2.75 1.75 d data - os002-data-hot-1 10.10.194.121 35 87 7 4.88 2.75 1.75
m cluster_manager - os002-masters-2 10.10.215.132 53 94 13 1.58 0.93 0.86 d data -
os002-data-hot-0 10.10.215.133 28 88 8 1.58 0.93 0.86 m cluster_manager - os002-
masters-1 [opensearch@os002-masters-0 ~]$

```

7. 外部验证 OpenSearch 集群证书。

备注

- 在 2.0.16 之前的 operator 版本，OpenSearch 集群 Service 类型默认为 ClusterIP，且不支持修改，故无法修改 service 类型为 NodePort 供外部访问集群。若需要外部访问，可以通过额外增加一个 NodePort 类型的 service 实现。
- 在 2.0.16 及以后的 operator 版本，service 默认类型已调整为 NodePort。

a. 获取 **service/opensearch** 的 YAML 配置，并重命名。

```

root@master1:~# kubectl -n opensearch get service/opensearch -oyaml > opensearch-
external.yaml

```

b. 修改 **opensearch-external.yaml**。

```

apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/service: opensearch
    opster.io/exporter-monitor: "yes"
    opster.io/opensearch-cluster: opensearch
name: opensearch-external
namespace: opensearch
ownerReferences:
- apiVersion: opensearch.opster.io/v1
  blockOwnerDeletion: true
  controller: true
  kind: OpenSearchCluster
  name: opensearch
  uid: 5f16ee6c-35ba-460f-910a-fa88f4f32f86
spec:
  clusterIP: 10.96.220.19
  clusterIPs:
  - 10.96.220.19
  internalTrafficPolicy: Cluster
  ipFamilies:
  - IPv4

```

```

ipFamilyPolicy: SingleStack
ports:
  - name: http
    port: 9200
    protocol: TCP
    targetPort: 9200
  - name: transport
    port: 9300
    protocol: TCP
    targetPort: 9300
  - name: metrics
    port: 9600
    protocol: TCP
    targetPort: 9600
  - name: rca
    port: 9650
    protocol: TCP
    targetPort: 9650
selector:
  opster.io/opensearch-cluster: opensearch
sessionAffinity: None
type: NodePort
status:
  loadBalancer: {}

```

- metadata 里保留 labels、name、namespace、ownerReferences 属性。
- 修改 metadata.name 字段，自定义 service 名称。
- 删除 spec.clusterIP 和 spec.clusterIPs 字段，防止 IP 冲突。
- 修改 spec.type 为 NodePort。

可参考以下示例：

```

apiVersion: v1
kind: Service
metadata:
  labels:
    app.kubernetes.io/service: opensearch
    opster.io/exporter-monitor: "yes"
    opster.io/opensearch-cluster: opensearch
  name: opensearch-external
  namespace: opensearch
  ownerReferences:
    - apiVersion: opensearch.opster.io/v1
      blockOwnerDeletion: true
      controller: true
      kind: OpenSearchCluster
      name: opensearch
      uid: 5f16ee6c-35ba-460f-910a-fa88f4f32f86
spec:
  internalTrafficPolicy: Cluster
  ipFamilies:

```

```
- IPv4
  ipFamilyPolicy: SingleStack
  ports:
  - name: http
    port: 9200
    protocol: TCP
    targetPort: 9200
  - name: transport
    port: 9300
    protocol: TCP
    targetPort: 9300
  - name: metrics
    port: 9600
    protocol: TCP
    targetPort: 9600
  - name: rca
    port: 9650
    protocol: TCP
    targetPort: 9650
  selector:
    opster.io/opensearch-cluster: opensearch
  sessionAffinity: None
  type: NodePort
status:
  loadBalancer: {}
```

c. 创建 opensearch-external 服务。

```
root@master1:~# kubectl -n opensearch apply -f ./opensearch-external.yaml
```

d. 检查服务是否创建成功。

```
root@master1:~# kubectl -n opensearch get svc
```

如果您能够在回显中找到 **opensearch-external** 服务，则该服务创建成功。

```
root@master1:~# kubectl -n opensearch get svc
NAME                                TYPE                CLUSTER-IP      EXTERNAL-IP      PORT(S)                                     AGE
opensearch                         ClusterIP           10.96.220.19    <none>            9200/TCP,9300/TCP,9600/TCP,9650/TCP       29h
opensearch-dashboards              NodePort            10.96.59.35     <none>            5601:31004/TCP                           29h
opensearch-data-plot               ClusterIP           None            <none>            9200/TCP,9300/TCP                         29h
opensearch-discovery               ClusterIP           None            <none>            9300/TCP                                   29h
opensearch-external                NodePort            10.96.53.135    <none>            9200:31427/TCP,9300:31768/TCP,9600:30872/TCP,9650:32312/TCP 65s
opensearch-masters                 ClusterIP           None            <none>            9200/TCP,9300/TCP                         29h
root@master1:~#
```

8. 配置 /etc/hosts 文件。

```
echo "172.22.9.17 os002-masters-0.os002.os002.opensearch.svc.cluster.local" >>
/etc/hosts
```

其中，**172.22.9.17** 为任意节点名称，**os002-masters-0.os002.os002.opensearch.svc.cluster.local** 为其中一个 master 节点名称，可任意配置。

9. 执行以下命令进行外部访问。

```
root@i-abmm2sr0:~/test# curl https://os002-masters-
0.os002.opensearch.svc.cluster.local:30255/_cat/nodes --cacert ./root-ca.pem -u
```

```
root:x804NqPd3tz7HBzr
10.10.194.121 61 88 6 0.49 0.80 1.28 m cluster_manager - os002-masters-2
10.10.215.133 57 89 7 0.35 0.60 0.79 m cluster_manager - os002-masters-1
10.10.194.120 57 94 4 0.49 0.80 1.28 d data - os002-data-hot-1 10.10.2.138 34 95 5
3.41 2.13 1.80 m cluster_manager * os002-masters-0 10.10.215.132 57 95 8 0.35 0.60
0.79 d data - os002-data-hot-0
```


10. 对接 Logstash 和 Kafka

Logstash 是一个开源数据收集引擎，支持从多个数据源采集数据、转换数据、并将数据写入到指定存储中。

本节介绍如何对接 Logstash 和 Kafka。

前提条件

- 已获取 KubeSphere 企业版平台登录账号和密码，且已获取平台管理权限。
- 已经创建好 Kafka 和 OpenSearch 集群，且集群状态处于运行中。

Kafka 集群操作步骤

- 以 platform-admin 角色登录 KubeSphere 企业版 Web 控制台并进入数据库管理平台。
- 在左侧导航栏选择 **Kafka**。
- 在 Kafka 集群列表中，点击一个集群名称打开其详情页面。
- 在页面右侧点击**主题**。
- 在**创建主题**对话框，输入主题名称，并设置分区数量和副本数量，点击**确定**。
- 点击 **Kafka 用户**，然后点击**创建**。
- 在**创建用户**对话框，设置用户名为 **logstash**，然后点击**确定**。用户创建完成后将显示在用户列表中。
- 在页面左侧的**传输加密**区域，下载 CA 证书并保存密码。
- 在页面右侧点击 **Kafka 用户** 页签，下载用户证书并保存密码。



- 上传证书到服务器，根据 keystore 和 truststore 创建 secret 或 configmap。

```
# 创建 secret
kubectl -n p1 create secret generic test-kafka-ssl --from-file
./cluster.keystore.p12 --from-file ./cluster.truststore.p12
# 创建 configmap
```

```
kubect1 -n p1 create cm test-cm-kafka --from-file=/tmp/cluster.keystore.p12 --from-
file=/tmp/cluster.truststore.p12
```

OpenSearch 集群操作步骤

1. 配置 Logstash CR 文件 logstash-kafka-ssl.yml。

```
apiVersion: opensearch.opster.io/v1
kind: Logstash
metadata:
  name: radondb-os
spec:
  replicas: 1
  config:
    jvm: "-Xms512m -Xmx512m"
    openSearchInfo:
      openSearchCluster:
        name: radondb-os # opensearch 集群名称
        namespace: p1 # opensearch 所在的 ns
      ports: [8080]
      # bootstrap_servers      : kafka                : 读写地址
      # security_protocol      : 认证类型              : 固定值 SSL
      # ssl_keystore_location   : 用户证书的全路径     : 固定值
      /usr/share/ssl/cluster.keystore.p12
      # ssl_keystore_password   : 用户密码              : 上述 logstash 用户证书密码
      # ssl_keystore_type       : 用户证书格式          : PKCS12 (kse中的 kafka 用此格式, 根据实际
      情况调整)
      # ssl_truststore_location : ca 证书的全路径       : 固定值
      /usr/share/ssl/cluster.truststore.p12
      # ssl_truststore_password : ca 证书的密码          : 上述 ca 证书密码
      # ssl_truststore_type     : ca 证书的格式          : PKCS12 (kse中的 kafka 用此格式, 根据实
      际情况调整)
      # ssl_endpoint_identification_algorithm : 固定值 ""
    pipelineConfig:
      inputs: |-
        kafka {
          topics => ["test-topic", "test-topic-1"]
          bootstrap_servers => "radondb-85z8fi-kafka-external-bootstrap.p1:9092"
          security_protocol => "SSL"
          ssl_keystore_location => "/usr/share/ssl/cluster.keystore.p12"
          ssl_keystore_password => "HZznFT_xv1BXDA6NkqBC7PbT9_vzI1t2"
          ssl_keystore_type => "PKCS12"
          ssl_truststore_location => "/usr/share/ssl/cluster.truststore.p12"
          ssl_truststore_password => "HZznFT_xv1BXDA6NkqBC7PbT9_vzI1t2"
          ssl_truststore_type => "PKCS12"
          ssl_endpoint_identification_algorithm => ""
        }
      filters: |-
        mutate {
          add_field => { "LogstashHost" => "${HOSTNAME}" }
        }
      outputs:
        openSearchIndex: |- # 保存到 opensearch 的索引设置
```

```

index => "logstash-%{+YYYY.MM.dd}"
podTemplate:
  spec:
    containers:
      - name: logstash
        resources:
          requests:
            memory: "1Gi"
            cpu: "500m"
          limits:
            memory: "1Gi"
            cpu: "500m"
        volumeMounts:
          - name: certification-volume
            mountPath: /usr/share/ssl
    volumes:
      - name: certification-volume
        # test-secret-kafka Secret 需要提前创建, 命令如下
        # kubectl -n p1 create secret generic test-kafka-ssl --from-file
        ./cluster.keystore.p12 --from-file ./cluster.truststore.p12
        secret:
          secretName: test-secret-kafka
        # 或者使用 configmap 挂载, 也需要提前创建, 命令如下
        # kubectl -n p1 create cm test-cm-kafka --from-
        file=/tmp/cluster.keystore.p12 --from-file=/tmp/cluster.truststore.p12
        # configMap:
        #   name: test-cm-kafka

```

2. 执行以下命令创建 Logstash。

```
kubectl -n p1 apply -f ./logstash-kafka-ssl.yml
```

验证 Input 是否生效

1. 在客户端节点执行以下命令，向主题发送消息。

```
./kafka-console-producer.sh --broker-list {连接地址} --topic {Topic 名称}
```

- **连接地址**：所连接的 Kafka 集群的地址，格式为 host_ip1:port,host_ip2:port,host_ip3:port。
host_ip 为 Kafka 节点的 IP 地址，port 为客户端节点的访问端口 9092。
- **Topic 名称**：创建的主题名称。

2. 输入需要发送的消息内容，按 **Enter** 发送消息，每一行的内容都将作为一条消息发送到 Kafka。

示例如下：

```

./kafka-console-producer.sh --broker-list
172.22.2.124:9092,172.22.2.125:9092,192.22.2.126:9092 --topic test-topic
>hello world
>test msg1

```

3. 在 OpenSearch Dashboards 查看数据写入情况，如果能够找到索引 **logstash-yyyy.MM.dd**，则证明消息写入成功。

```
GET _cat/indices
```

History	Settings	Help
1	GET _cat/indices	1 green open security-auditlog-2022.08.16 Ph9iEI5JT8qqWjFWdolcKg 1 1 1103 0 3.1mb 1.2mb
2		2 green open .kibana_92668751_admin_1 KowccR8ASceLNuywZ0dcSg 1 1 1 0 10.1kb 5kb
3		3 green open logstash-2022.08.16 bnKtZQjdQ5eUh5Qn3F95Kw 1 1 5 0 16.5kb 8.2kb
4		4 green open .opendistro-job-scheduler-lock dJD53vi1Re04zEC4UBtGnA 1 1 0 0 26.4kb 26.1kb
5		5 green open .opendistro-reports-definitions pFKnmStoTJqP6X36s8B-QQ 1 1 0 0 416b 208b
6		6 green open .kibana_1 48wn9mEqQjy84LtyL9WSvA 1 1 0 0 416b 208b
7		7 green open .opendistro_security 3LF0ak4J50GEa6A6m788mQ 1 1 9 0 116kb 58kb
8		8 green open .opendistro-reports-instances 3wRByCZJRaaArwbM41EhEA 1 1 0 0 416b 208b
9		
10		
11		

4. 执行以下命令查看写入到主题的信息是否完整。

```
GET /logstash-2022.08.16/_search
```

Console

History	Settings	Help
1	GET /logstash-2022.08.16/_search	1 {
2		2 "took" : 2,
3		3 "timed_out" : false,
4		4 "shards" : {
5		5 "total" : 1,
6		6 "successful" : 1,
7		7 "skipped" : 0,
8		8 "failed" : 0
9		9 },
10		10 "hits" : {
11		11 "total" : {
12		12 "value" : 5,
13		13 "relation" : "eq"
14		14 },
15		15 "max_score" : 1.0,
16		16 "hits" : [
17		17 {
18		18 "_index" : "logstash-2022.08.16",
19		19 "_type" : "_doc",
20		20 "_id" : "OgjcpIIBKrNnTyrxcJT",
21		21 "_score" : 1.0,
22		22 "_source" : {
23		23 "@timestamp" : "2022-08-16T04:15:09.729Z",
24		24 "message" : "hello world",
25		25 "@version" : "1"
26		26 }
27		27 },
28		28 {
29		29 "_index" : "logstash-2022.08.16",
30		30 "_type" : "_doc",
31		31 "_id" : "OwjcpIIBKrNnTyrx2cIr",
32		32 "_score" : 1.0,
33		33 "_source" : {
34		34 "@timestamp" : "2022-08-16T04:15:13.080Z",
35		35 "message" : "test msg1",
36		36 "@version" : "1"
37		37 }
38		38 }.