Configuring the Elasticsearch log store

You can use Elasticsearch 6 to store and organize log data.

You can make modifications to your log store, including:

- Storage for your Elasticsearch cluster
- Shard replication across data nodes in the cluster, from full replication to no replication
- External access to Elasticsearch data

Configuring log storage

You can configure which log storage type your logging uses by modifying the `ClusterLogging` custom resource (CR).

Prerequisites

- You have administrator permissions.
- You have installed the OpenShift CLI (`oc`).
- You have installed the Red Hat OpenShift Logging Operator and an internal log store that is either the LokiStack or Elasticsearch.
- You have created a `ClusterLogging` CR.

> The Logging 5.9 release does not contain an updated version of the OpenShift Elasticsearch Operator. If you currently use the OpenShift Elasticsearch Operator released with Logging 5.8, it will continue to work with Logging until the EOL of Logging 5.8. As an alternative to using the OpenShift Elasticsearch Operator to manage the default log storage, you can use the Loki Operator. For more information on the Logging lifecycle dates, see Platform Agnostic Operators.

Procedure

- Modify the `ClusterLogging` CR `logStore` spec:

  `ClusterLogging` CR example

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
# ...
spec:
# ...
  logStore:
    type: <log_store_type>  (1)
    elasticsearch:  (2)
      nodeCount: <integer>
      resources: {}
      storage: {}
      redundancyPolicy: <redundancy_type>  (3)
    lokistack:  (4)
      name: {}
# ...
```

1. Specify the log store type. This can be either `lokistack` or `elasticsearch`.
2. Optional configuration options for the Elasticsearch log store.
3. Specify the redundancy type. This value can be `ZeroRedundancy`, `SingleRedundancy`, `MultipleRedundancy`, or `FullRedundancy`.
4. Optional configuration options for LokiStack.

Example `ClusterLogging` CR to specify LokiStack as the log store

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogging
metadata:
  name: instance
  namespace: openshift-logging
spec:
  managementState: Managed
  logStore:
    type: lokistack
    lokistack:
      name: logging-loki
# ...
```

- Apply the `ClusterLogging` CR by running the following command:

```
$ oc apply -f <filename>.yaml
```

Forwarding audit logs to the log store

In a logging deployment, container and infrastructure logs are forwarded to the internal log store defined in the `ClusterLogging` custom resource (CR) by default.

Audit logs are not forwarded to the internal log store by default because this does not provide secure storage. You are responsible for ensuring that the system to which you forward audit logs is compliant with your organizational and governmental regulations, and is properly secured.

If this default configuration meets your needs, you do not need to configure a `ClusterLogForwarder` CR. If a `ClusterLogForwarder` CR exists, logs are not forwarded to the internal log store unless a pipeline is defined that contains the `default` output.

Procedure

To use the Log Forward API to forward audit logs to the internal Elasticsearch instance:

- Create or edit a YAML file that defines the `ClusterLogForwarder` CR object:
  - Create a CR to send all log types to the internal Elasticsearch instance. You can use the following example without making any changes:

```
apiVersion: logging.openshift.io/v1
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  pipelines: (1)
  - name: all-to-default
    inputRefs:
    - infrastructure
    - application
    - audit
    outputRefs:
    - default
```

1 A pipeline defines the type of logs to forward using the specified output. The default output forwards logs to the internal Elasticsearch instance.

> You must specify all three types of logs in the pipeline: application, infrastructure, and audit. If you do not specify a log type, those logs are not stored and will be lost.

- If you have an existing `ClusterLogForwarder` CR, add a pipeline to the default output for the audit logs. You do not need to define the default output. For example:

```
apiVersion: "logging.openshift.io/v1"
kind: ClusterLogForwarder
metadata:
  name: instance
  namespace: openshift-logging
spec:
  outputs:
   - name: elasticsearch-insecure
     type: "elasticsearch"
     url: http://elasticsearch-insecure.messaging.svc.cluster.local
     insecure: true
   - name: elasticsearch-secure
     type: "elasticsearch"
     url: https://elasticsearch-secure.messaging.svc.cluster.local
     secret:
       name: es-audit
   - name: secureforward-offcluster
     type: "fluentdForward"
     url: https://secureforward.offcluster.com:24224
     secret:
       name: secureforward
  pipelines:
   - name: container-logs
     inputRefs:
     - application
     outputRefs:
     - secureforward-offcluster
   - name: infra-logs
     inputRefs:
     - infrastructure
     outputRefs:
     - elasticsearch-insecure
   - name: audit-logs
     inputRefs:
     - audit
     outputRefs:
     - elasticsearch-secure
     - default (1)
```

1 This pipeline sends the audit logs to the internal Elasticsearch instance in addition to an external instance.

Additional resources

- About log collection and forwarding

Configuring log retention time

You can configure a *retention policy* that specifies how long the default Elasticsearch log store keeps indices for each of the three log sources: infrastructure logs, application logs, and audit logs.

To configure the retention policy, you set a `maxAge` parameter for each log source in the `ClusterLogging` custom resource (CR). The CR applies these values to the Elasticsearch rollover schedule, which determines when Elasticsearch deletes the rolled-over indices.

Elasticsearch rolls over an index, moving the current index and creating a new index, when an index matches any of the following conditions:

- The index is older than the `rollover.maxAge` value in the `Elasticsearch` CR.
- The index size is greater than 40 GB × the number of primary shards.
- The index doc count is greater than 40960 KB × the number of primary shards.

Elasticsearch deletes the rolled-over indices based on the retention policy you configure. If you do not create a retention policy for any log sources, logs are deleted after seven days by default.

Prerequisites

- The Red Hat OpenShift Logging Operator and the OpenShift Elasticsearch Operator must be installed.

Procedure

To configure the log retention time:

- Edit the `ClusterLogging` CR to add or modify the `retentionPolicy` parameter:

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
...
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    retentionPolicy: (1)
      application:
        maxAge: 1d
      infra:
        maxAge: 7d
      audit:
        maxAge: 7d
    elasticsearch:
      nodeCount: 3
...
```

1 Specify the time that Elasticsearch should retain each log source. Enter an integer and a time designation: weeks(w), hours(h/H), minutes(m) and seconds(s). For example, `1d` for one day. Logs older than the `maxAge` are deleted. By default, logs are retained for seven days.

- You can verify the settings in the `Elasticsearch` custom resource (CR).

  For example, the Red Hat OpenShift Logging Operator updated the following `Elasticsearch` CR to configure a retention policy that includes settings to roll over active indices for the infrastructure logs every eight hours and the rolled-over indices are deleted seven days after rollover. OpenShift Container Platform checks every 15 minutes to determine if the indices need to be rolled over.

```
apiVersion: "logging.openshift.io/v1"
kind: "Elasticsearch"
metadata:
  name: "elasticsearch"
spec:
...
  indexManagement:
    policies: (1)
      - name: infra-policy
        phases:
          delete:
            minAge: 7d (2)
          hot:
            actions:
              rollover:
                maxAge: 8h (3)
        pollInterval: 15m (4)
...
```

1 For each log source, the retention policy indicates when to delete and roll over logs for that source.

2 When OpenShift Container Platform deletes the rolled-over indices. This setting is the `maxAge` you set in the `ClusterLogging` CR.

3 The index age for OpenShift Container Platform to consider when rolling over the indices. This value is determined from the `maxAge` you set in the `ClusterLogging` CR.

4 When OpenShift Container Platform checks if the indices should be rolled over. This setting is the default and cannot be changed.

> Modifying the `Elasticsearch` CR is not supported. All changes to the retention policies must be made in the `ClusterLogging` CR.

The OpenShift Elasticsearch Operator deploys a cron job to roll over indices for each mapping using the defined policy, scheduled using the `pollInterval` .

```
$ oc get cronjob
```

Example output

```
NAME                     SCHEDULE       SUSPEND   ACTIVE   LAST SCHEDULE   AGE
elasticsearch-im-app     */15 * * * *   False     0        <none>          4s
elasticsearch-im-audit   */15 * * * *   False     0        <none>          4s
elasticsearch-im-infra   */15 * * * *   False     0        <none>          4s
```

Configuring CPU and memory requests for the log store

Each component specification allows for adjustments to both the CPU and memory requests. You should not have to manually adjust these values as the OpenShift Elasticsearch Operator sets values sufficient for your environment.

> In large-scale clusters, the default memory limit for the Elasticsearch proxy container might not be sufficient, causing the proxy container to be OOMKilled. If you experience this issue, increase the memory requests and limits for the Elasticsearch proxy.

Each Elasticsearch node can operate with a lower memory setting though this is **not** recommended for production deployments. For production use, you should have no less than the default 16Gi allocated to each pod. Preferably you should allocate as much as possible, up to 64Gi per pod.

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

- Edit the `ClusterLogging` custom resource (CR) in the `openshift-logging` project:

```
$ oc edit ClusterLogging instance
```

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
....
spec:
    logStore:
      type: "elasticsearch"
      elasticsearch: (1)
        resources:
          limits: (2)
            memory: "32Gi"
          requests: (3)
            cpu: "1"
            memory: "16Gi"
        proxy: (4)
          resources:
            limits:
              memory: 100Mi
            requests:
              memory: 100Mi
```

1 Specify the CPU and memory requests for Elasticsearch as needed. If you leave these values blank, the OpenShift Elasticsearch Operator sets default values that should be sufficient for most deployments. The default values are `16Gi` for the memory request and `1` for the CPU request.

2 The maximum amount of resources a pod can use.

3 The minimum resources required to schedule a pod.

4 Specify the CPU and memory requests for the Elasticsearch proxy as needed. If you leave these values blank, the OpenShift Elasticsearch Operator sets default values that are sufficient for most deployments. The default values are `256Mi` for the memory request and `100m` for the CPU request.

When adjusting the amount of Elasticsearch memory, the same value should be used for both `requests` and `limits`.

For example:

```
      resources:
        limits: (1)
          memory: "32Gi"
        requests: (2)
          cpu: "8"
          memory: "32Gi"
```

1 The maximum amount of the resource.

2 The minimum amount required.

Kubernetes generally adheres the node configuration and does not allow Elasticsearch to use the specified limits. Setting the same value for the `requests` and `limits` ensures that Elasticsearch can use the memory you want, assuming the node has the memory available.

Configuring replication policy for the log store

You can define how Elasticsearch shards are replicated across data nodes in the cluster.

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

- Edit the `ClusterLogging` custom resource (CR) in the `openshift-logging` project:

```
$ oc edit clusterlogging instance
```

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"

....

spec:
  logStore:
    type: "elasticsearch"
    elasticsearch:
      redundancyPolicy: "SingleRedundancy" (1)
```

Specify a redundancy policy for the shards. The change is applied upon saving the changes.

- **FullRedundancy**. Elasticsearch fully replicates the primary shards for each index to every data node. This provides the highest safety, but at the cost of the highest amount of disk required and the poorest performance.
- **MultipleRedundancy**. Elasticsearch fully replicates the primary shards for each index to half of the data nodes. This provides a good tradeoff between safety and performance.

1
- **SingleRedundancy**. Elasticsearch makes one copy of the primary shards for each index. Logs are always available and recoverable as long as at least two data nodes exist. Better performance than MultipleRedundancy, when using 5 or more nodes. You cannot apply this policy on deployments of single Elasticsearch node.
- **ZeroRedundancy**. Elasticsearch does not make copies of the primary shards. Logs might be unavailable or lost in the event a node is down or fails. Use this mode when you are more concerned with performance than safety, or have implemented your own disk/PVC backup/restore strategy.

The number of primary shards for the index templates is equal to the number of Elasticsearch data nodes.

Scaling down Elasticsearch pods

Reducing the number of Elasticsearch pods in your cluster can result in data loss or Elasticsearch performance degradation.

If you scale down, you should scale down by one pod at a time and allow the cluster to re-balance the shards and replicas. After the Elasticsearch health status returns to `green`, you can scale down by another pod.

If your Elasticsearch cluster is set to `ZeroRedundancy`, you should not scale down your Elasticsearch pods.

Configuring persistent storage for the log store

Elasticsearch requires persistent storage. The faster the storage, the faster the Elasticsearch performance.

Using NFS storage as a volume or a persistent volume (or via NAS such as Gluster) is not supported for Elasticsearch storage, as Lucene relies on file system behavior that NFS does not supply. Data corruption and other problems can occur.

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

- Edit the `ClusterLogging` CR to specify that each data node in the cluster is bound to a Persistent Volume Claim.

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
# ...
spec:
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      storage:
        storageClassName: "gp2"
        size: "200G"
```

This example specifies each data node in the cluster is bound to a Persistent Volume Claim that requests "200G" of AWS General Purpose SSD (gp2) storage.

If you use a local volume for persistent storage, do not use a raw block volume, which is described with `volumeMode: block` in the `LocalVolume` object. Elasticsearch cannot use raw block volumes.

Configuring the log store for emptyDir storage

You can use emptyDir with your log store, which creates an ephemeral deployment in which all of a pod's data is lost upon restart.

When using emptyDir, if log storage is restarted or redeployed, you will lose data.

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

- Edit the `ClusterLogging` CR to specify emptyDir:

```
spec:
   logStore:
     type: "elasticsearch"
     elasticsearch:
       nodeCount: 3
       storage: {}
```

Performing an Elasticsearch rolling cluster restart

Perform a rolling restart when you change the `elasticsearch` config map or any of the `elasticsearch-*` deployment configurations.

Also, a rolling restart is recommended if the nodes on which an Elasticsearch pod runs requires a reboot.

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.

Procedure

To perform a rolling cluster restart:

- Change to the `openshift-logging` project:

```
$ oc project openshift-logging
```

- Get the names of the Elasticsearch pods:

```
$ oc get pods -l component=elasticsearch
```

- Scale down the collector pods so they stop sending new logs to Elasticsearch:

```
$ oc -n openshift-logging patch daemonset/collector -p '{"spec":{"template":{"spec":{"nodeSelector":
{"logging-infra-collector": "false"}}}}}'
```

- Perform a shard synced flush using the OpenShift Container Platform **es_util** tool to ensure there are no pending operations waiting to be written to disk prior to shutting down:

```
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --query="_flush/synced" -XPOST
```

For example:

```
$ oc exec -c elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6  -c elasticsearch -- es_util --query="_flush/synced" -X
```

Example output

```
{"_shards":{"total":4,"successful":4,"failed":0},".security":{"total":2,"successful":2,"failed":0},".kibana_1":{
```

- Prevent shard balancing when purposely bringing down nodes using the OpenShift Container Platform es_util tool:

```
$ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --query="_cluster/settings" -XPUT -d '{ "persi
```

For example:

```
$ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --query="_cluster/settings" -X
```

Example output

```
{"acknowledged":true,"persistent":{"cluster":{"routing":{"allocation":
{"enable":"primaries"}}}},"transient":
```

- After the command is complete, for each deployment you have for an ES cluster:
  - By default, the OpenShift Container Platform Elasticsearch cluster blocks rollouts to their nodes. Use the following command to allow rollouts and allow the pod to pick up the changes:

    ```
    $ oc rollout resume deployment/<deployment-name>
    ```

    For example:

    ```
    $ oc rollout resume deployment/elasticsearch-cdm-0-1
    ```

    Example output

    ```
    deployment.extensions/elasticsearch-cdm-0-1 resumed
    ```

    A new pod is deployed. After the pod has a ready container, you can move on to the next deployment.

    ```
    $ oc get pods -l component=elasticsearch-
    ```

    Example output

    ```
    NAME                                            READY   STATUS    RESTARTS   AGE
    elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6k    2/2     Running   0          22h
    elasticsearch-cdm-5ceex6ts-2-f799564cb-l9mj7    2/2     Running   0          22h
    elasticsearch-cdm-5ceex6ts-3-585968dc68-k7kjr   2/2     Running   0          22h
    ```

  - After the deployments are complete, reset the pod to disallow rollouts:

    ```
    $ oc rollout pause deployment/<deployment-name>
    ```

    For example:

    ```
    $ oc rollout pause deployment/elasticsearch-cdm-0-1
    ```

    Example output

    ```
    deployment.extensions/elasticsearch-cdm-0-1 paused
    ```

  - Check that the Elasticsearch cluster is in a `green` or `yellow` state:

    ```
    $ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --query=_cluster/health?pretty=true
    ```

    > If you performed a rollout on the Elasticsearch pod you used in the previous commands, the pod no longer exists and you need a new pod name here.

    For example:

    ```
    $ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --query=_cluster/health?p
    ```

```
{
 "cluster_name" : "elasticsearch",
 "status" : "yellow", (1)
 "timed_out" : false,
 "number_of_nodes" : 3,
 "number_of_data_nodes" : 3,
 "active_primary_shards" : 8,
 "active_shards" : 16,
 "relocating_shards" : 0,
 "initializing_shards" : 0,
 "unassigned_shards" : 1,
 "delayed_unassigned_shards" : 0,
 "number_of_pending_tasks" : 0,
 "number_of_in_flight_fetch" : 0,
 "task_max_waiting_in_queue_millis" : 0,
 "active_shards_percent_as_number" : 100.0
}
```

1 Make sure this parameter value is `green` or `yellow` before proceeding.

- If you changed the Elasticsearch configuration map, repeat these steps for each Elasticsearch pod.
- After all the deployments for the cluster have been rolled out, re-enable shard balancing:

  ```
  $ oc exec <any_es_pod_in_the_cluster> -c elasticsearch -- es_util --query="_cluster/settings" -XPUT -d '{ "persi
  ```

  For example:

  ```
  $ oc exec elasticsearch-cdm-5ceex6ts-1-dcd6c4c7c-jpw6 -c elasticsearch -- es_util --query="_cluster/settings" -X
  ```

  Example output

  ```
  {
    "acknowledged" : true,
    "persistent" : { },
    "transient" : {
      "cluster" : {
        "routing" : {
          "allocation" : {
            "enable" : "all"
          }
        }
      }
    }
  }
  ```

- Scale up the collector pods so they send new logs to Elasticsearch.

  ```
  $ oc -n openshift-logging patch daemonset/collector -p '{"spec":{"template":{"spec":{"nodeSelector":
  {"logging-infra-collector": "true"}}}}}'
  ```

Exposing the log store service as a route

By default, the log store that is deployed with logging is not accessible from outside the logging cluster. You can enable a route with re-encryption termination for external access to the log store service for those tools that access its data.

Externally, you can access the log store by creating a reencrypt route, your OpenShift Container Platform token and the installed log store CA certificate. Then, access a node that hosts the log store service with a cURL request that contains:

- The `Authorization: Bearer ${token}`
- The Elasticsearch reencrypt route and an Elasticsearch API request.

Internally, you can access the log store service using the log store cluster IP, which you can get by using either of the following commands:

```
$ oc get service elasticsearch -o jsonpath={.spec.clusterIP} -n openshift-logging
```

Example output

```
172.30.183.229
```

```
$ oc get service elasticsearch -n openshift-logging
```

Example output

```
NAME            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)    AGE
elasticsearch   ClusterIP   172.30.183.229   <none>        9200/TCP   22h
```

You can check the cluster IP address with a command similar to the following:

```
$ oc exec elasticsearch-cdm-oplnhinv-1-5746475887-fj2f8 -n openshift-logging -- curl -tlsv1.2 --insecure
-H "Authorization: Bearer ${token}" "https://172.30.183.229:9200/_cat/health"
```

Example output

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    29  100    29    0     0    108      0 --:--:-- --:--:-- --:--:--   108
```

Prerequisites

- The Red Hat OpenShift Logging and Elasticsearch Operators must be installed.
- You must have access to the project to be able to access to the logs.

Procedure

To expose the log store externally:

- Change to the `openshift-logging` project:

  ```
  $ oc project openshift-logging
  ```

- Extract the CA certificate from the log store and write to the ***admin-ca*** file:

  ```
  $ oc extract secret/elasticsearch --to=. --keys=admin-ca
  ```

  Example output

  ```
  admin-ca
  ```

- Create the route for the log store service as a YAML file:

  - Create a YAML file with the following:

    ```
    apiVersion: route.openshift.io/v1
    kind: Route
    metadata:
      name: elasticsearch
      namespace: openshift-logging
    spec:
      host:
      to:
        kind: Service
        name: elasticsearch
      tls:
        termination: reencrypt
        destinationCACertificate: | (1)
    ```

    1 Add the log store CA certifcate or use the command in the next step. You do not have to set the `spec.tls.key`, `spec.tls.certificate`, and `spec.tls.caCertificate` parameters required by some reencrypt routes.

  - Run the following command to add the log store CA certificate to the route YAML you created in the previous step:

    ```
    $ cat ./admin-ca | sed -e "s/^/      /" >> <file-name>.yaml
    ```

  - Create the route:

```
$ oc create -f <file-name>.yaml
```

Example output

```
route.route.openshift.io/elasticsearch created
```

- Check that the Elasticsearch service is exposed:
  - Get the token of this service account to be used in the request:

```
$ token=$(oc whoami -t)
```

  - Set the **elasticsearch** route you created as an environment variable.

```
$ routeES=`oc get route elasticsearch -o jsonpath={.spec.host}`
```

  - To verify the route was successfully created, run the following command that accesses Elasticsearch through the exposed route:

```
curl -tlsv1.2 --insecure -H "Authorization: Bearer ${token}" "https://${routeES}"
```

The response appears similar to the following:

Example output

```
{
  "name" : "elasticsearch-cdm-i40ktba0-1",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "0eY-tJzcR3KOdpgeMJo-MQ",
  "version" : {
  "number" : "6.8.1",
  "build_flavor" : "oss",
  "build_type" : "zip",
  "build_hash" : "Unknown",
  "build_date" : "Unknown",
  "build_snapshot" : true,
  "lucene_version" : "7.7.0",
  "minimum_wire_compatibility_version" : "5.6.0",
  "minimum_index_compatibility_version" : "5.0.0"
},
  "<tagline>" : "<for search>"
}
```

Removing unused components if you do not use the default Elasticsearch log store

As an administrator, in the rare case that you forward logs to a third-party log store and do not use the default Elasticsearch log store, you can remove several unused components from your logging cluster.

In other words, if you do not use the default Elasticsearch log store, you can remove the internal Elasticsearch `logStore` and Kibana `visualization` components from the `ClusterLogging` custom resource (CR). Removing these components is optional but saves resources.

Prerequisites

- Verify that your log forwarder does not send log data to the default internal Elasticsearch cluster. Inspect the `ClusterLogForwarder` CR YAML file that you used to configure log forwarding. Verify that it *does not* have an `outputRefs` element that specifies `default`. For example:

```
outputRefs:
- default
```

Suppose the `ClusterLogForwarder` CR forwards log data to the internal Elasticsearch cluster, and you remove the `logStore` component from the `ClusterLogging` CR. In that case, the internal Elasticsearch cluster will not be present to store the log data. This absence can cause data loss.

Procedure

- Edit the `ClusterLogging` custom resource (CR) in the `openshift-logging` project:

```
$ oc edit ClusterLogging instance
```

- If they are present, remove the `logStore` and `visualization` stanzas from the `ClusterLogging` CR.
- Preserve the `collection` stanza of the `ClusterLogging` CR. The result should look similar to the following example:

```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  collection:
    type: "fluentd"
    fluentd: {}
```

- Verify that the collector pods are redeployed:

```
$ oc get pods -l component=collector -n openshift-logging
```