OpenShift Log Aggregation

OpenShift Log Aggregation

In this lab you will explore the logging aggregation capabilities of OpenShift.

An extremely important function of OpenShift is collecting and aggregating logs from the environments and the application pods it is running. OpenShift ships with an elastic log aggregation solution: **EFK**. (**E**lasticSearch, **F**luentd and **K**ibana)

The cluster logging components are based upon Elasticsearch, Fluentd, and Kibana (EFK). The collector, Fluentd, is deployed to each node in the OpenShift cluster. It collects all node and container logs and writes them to Elasticsearch (ES). Kibana is the centralized, web UI where users and administrators can create rich visualizations and dashboards with the aggregated data. Administrators can see and search through all logs. Application owners and developers can allow access to logs that belong to their projects. The EFK stack runs on top of OpenShift.

This lab requires that you have completed the infra-nodes lab. The logging stack will be installed on the `infra` nodes that were created in that lab.

More information may be found on the official OpenShift documentation site found here:

```
https://docs.openshift.com/container-platform/4.1/logging/efk-logging.html
```

This exercise is done almost entirely using the OpenShift web console. All of the interactions with the web console are effectively creating or manipulating API objects in the background. It is possible to fully automate the process and/or do it using the CLI or other tools, but these methods are not covered in the exercise or documentation at this time.

Deploying OpenShift Logging

OpenShift Container Platform cluster logging is designed to be used with the default configuration, which is tuned for small to medium sized OpenShift Container Platform clusters. The installation instructions that follow include a sample Cluster Logging Custom Resource (CR), which you can use to create a cluster logging instance and configure your cluster logging deployment.

If you want to use the default cluster logging install, you can use the sample CR directly.

If you want to customize your deployment, make changes to the sample CR as needed. The following describes the configurations you can make when installing your cluster logging instance or modify after installtion. See the Configuring sections for more information on working with each component, including modifications you can make outside of the Cluster Logging Custom Resource.

Create the `openshift-logging` namespace

OpenShift Logging will be run from within its own namespace `openshift-logging`. This namespace does not exist by default, and needs to be created before logging may be installed. The namespace is represented in yaml format as:

openshift_logging_namespace.yaml

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: openshift-logging
  annotations:
    openshift.io/node-selector: ""
  labels:
    openshift.io/cluster-logging: "true"
    openshift.io/cluster-monitoring: "true"
```

To create the namespace, run the following command:

Make sure you are logged-in as kubeadmin

```
oc login -u kubeadmin -p xxx
```

```bash
oc create -f /content/support/openshift_logging_namespace.yaml
```

Install the `Elasticsearch` and `Cluster Logging` Operators in the cluster

In order to install and configure the `EFK` stack into the cluster, additional operators need to be installed. These can be installed from the `Operator Hub` from within the cluster via the GUI.

When using operators in OpenShift, it is important to understand the basics of some of the underlying principles that make up the Operators. `CustomResourceDefinion (CRD)` and `CustomResource (CR)` are two Kubernetes objects that we will briefly describe. `CRDs` are generic pre-defined structures of data. The operator understands how to apply the data that is defined by the `CRD`. In terms of programming, `CRDs` can be thought as being similar to a class. `CustomResource (CR)` is an actual implementations of the `CRD`, where the structured data has actual values. These values are what the operator will use when configuring it's service. Again, in programming terms, `CRs` would be similar to an instantiated object of the class.

The general pattern for using Operators is first, install the Operator, which will create the necessary `CRDs`. After the `CRDs` have been created, we can create the `CR` which will tell the operator how to act, what to install, and/or what to configure. For installing `openshift-logging`, we will follow this pattern.

To begin, log-in to the OpenShift Cluster's GUI. `https://console-openshift-console.apps.notrealcluster.com`

Then follow the following steps:

1. Install the Elasticsearch Operator:
   a. In the OpenShift console, click `Catalog` → `OperatorHub`.
   b. Choose `Elasticsearch Operator` from the list of available Operators, and click `Install`.

   > You may receive a warning about Community Operators. By the time OpenShift 4 is generally available, the various EFK operators required for the log aggregation solution will no longer be "community" status. Accept the warning by clicking "Continue".

   c. On the `Create Operator Subscription` page, select `All namespaces on the cluster` under `Installation Mode`. Then, click `Subscribe`.

   This makes the Operator available to all users and projects that use this OpenShift Container Platform cluster.
2. Install the Cluster Logging Operator:

   > The `Cluster Logging` operator needs to be installed in the `openshift-logging` namespace. Please ensure that the `openshift-logging` namespace was created from the previous steps

   a. In the OpenShift console, click `Catalog` → `OperatorHub`.
   b. Choose `Cluster Logging` from the list of available Operators, and click `Install`.
   c. On the `Create Operator Subscription` page, Under ensure `Installation Mode` that `A specific namespace on the cluster namespaces on the cluster` is selected, and choose `openshift-logging`. Then, click `Subscribe`.
3. Verify the operator installations:
   a. Switch to the `Catalog` → `Installed Operators` page.
   b. Ensure that `Cluster Logging` and `Elasticsearch Operator` are listed on the `InstallSucceeded` tab with a Status of `InstallSucceeded`. Change the project to `all projects` if necessary.

   > During installation an operator might display a `Failed` status. If the operator then installs with an `InstallSucceeded` message, you can safely ignore the `Failed` message.

4. Troubleshooting (optional)

   If either operator does not appear as installed, to troubleshoot further:

   - On the Copied tab of the Installed Operators page, if an operator show a Status of Copied, this indicates the installation is in process and is expected behavior.
   - Switch to the Catalog → Operator Management page and inspect the Operator Subscriptions and Install Plans tabs for any failure or errors under Status.
   - Switch to the Workloads → Pods page and check the logs in any Pods in the openshift-logging and openshift-operators projects that are reporting issues.

Create the Loggging `CustomResource (CR)` instance

Now that we have the operators installed, along with the `CRDs`, we can now kick off the logging install by creating a Logging `CR`. This will define how we want to install and configure logging.

1. In the OpenShift Console, switch to the the `Administration` → `Custom Resource Definitions` page.
2. On the `Custom Resource Definitions` page, click `ClusterLogging`.
3. On the `Custom Resource Definition Overview` page, select `View Instances` from the `Actions` menu.

   > If you see a `404` error, don't panic. While the operator installation succeeded, the operator itself has not finished installing and the `CustomResourceDefinition` may not have been created yet. Wait a few moments and then refresh the page.

4. On the `Cluster Loggings` page, click `Create Cluster Logging`.
5. In the `YAML` editor, replace the code with the following:

openshift_logging_cr.yaml

```yaml
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      storage: {}
      redundancyPolicy: "SingleRedundancy"
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      resources:
        request:
          memory: 4G
  visualization:
    type: "kibana"
    kibana:
      replicas: 1
      nodeSelector:
        node-role.kubernetes.io/infra: ""
  curation:
    type: "curator"
    curator:
      schedule: "30 3 * * *"
      nodeSelector:
        node-role.kubernetes.io/infra: ""
  collection:
    logs:
      type: "fluentd"
      fluentd: {}
      nodeSelector:
        node-role.kubernetes.io/infra: ""
```

Then click `Create`.

Verify the Loggging install

Now that Logging has been created, let's verify that things are working.

1. Switch to the `Workloads` → `Pods` page.
2. Select the `openshift-logging` project.

You should see pods for cluster logging, Elasticsearch, and Fluentd, as shown in the following CLI output:

Alternatively, you can verify from the command line by using the following command:

```bash
oc get pods -n openshift-logging
```

You should eventually see something like:

```
NAME                                           READY   STATUS    RESTARTS   AGE
cluster-logging-operator-cb795f8dc-xkckc       1/1     Running   0          32m
elasticsearch-cdm-b3nqzchd-1-5c6797-67kfz      2/2     Running   0          14m
elasticsearch-cdm-b3nqzchd-2-6657f4-wtprv      2/2     Running   0          14m
elasticsearch-cdm-b3nqzchd-3-588c65-clg7g      2/2     Running   0          14m
fluentd-2c7dg                                  1/1     Running   0          14m
fluentd-9z7kk                                  1/1     Running   0          14m
fluentd-br7r2                                  1/1     Running   0          14m
fluentd-fn2sb                                  1/1     Running   0          14m
fluentd-pb2f8                                  1/1     Running   0          14m
fluentd-zqgqx                                  1/1     Running   0          14m
kibana-7fb4fd4cc9-bvt4p                        2/2     Running   0          14m
```

The *Fluentd* **Pods** are deployed as part of a **DaemonSet**, which is a mechanism to ensure that specific **Pods** run on specific **Nodes** in the cluster at all times:

```bash
oc get daemonset -n openshift-logging
```

You will see something like:

```
NAME      DESIRED   CURRENT   READY     UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
fluentd   8         8         8         8            8           <none>          15m
```

You should expect 1 `fluentd` **Pod** for every **Node** in your cluster. Remember that **Masters** are still **Nodes** and `fluentd` will run there, too, to slurp the various logs.

You will also see the storage for ElasticSearch being automatically provisioned from the default storage service if you query the PersistentVolumeClaim objects in this project

> Much like with the Metrics solution, we defined the appropriate `NodeSelector` in the Logging configuration (`CR`) to ensure that the Logging components only landed on the infra nodes. That being said, the `DaemonSet` ensures FluentD runs on **all** nodes. Otherwise we would not capture all of the container logs.

Accessing *Kibana*

As mentioned before, *Kibana* is the front end and the way that users and admins may access the OpenShift Logging stack. To reach the *Kibana* user interface, first determine its public access URL by querying the **Route** that got set up to expose Kibana's **Service**:

To find and access the *Kibana* route:

1. In the OpenShift console, click on the `Networking` → `Routes` page.
2. Select the `openshift-logging` project.
3. Click on the `Kibana` route.
4. In the `Location` field, click on the URL presented.
5. Click through and accept the SSL certificates

Alternatively, this can be obtained from the command line:

```bash
oc get route -n openshift-logging
```

You will see something like:

```
NAME     HOST/PORT                                                    PATH   SERVICES   PORT    TERMINATION          WILDCARD
kibana   kibana-openshift-logging.apps.notrealcluster.com          kibana     <all>   reencrypt/Redirect   None
```

Or, you can control+click the link:

https://kibana-openshift-logging.apps.notrealcluster.com

There is a special authentication proxy that is configured as part of the EFK installation that results in Kibana requiring OpenShift credentials for access.

Queries with *Kibana*

Once the *Kibana* web interface is up, we are now able to do queries. *Kibana* offers a the user a powerful interface to query all logs that come from the cluster.

By default, *Kibana* will show all logs that have been received within the the last 15 minutes. This time interval may be changed in the upper right hand corner. The log messages are shown in the middle of the page. All log messages that are received are indexed based on the log message content. Each message will have fields associated that are associated to that log message. To see the fields that make up an individual message, click on the arrow on the side of each message located in the center of the page. This will show the message fields that are contained.

First, set the default index pattern to `.all` . On the left hand side towards the top, in the drop down menu select the `.all` index pattern.

To select fields to show for messages, look on left hand side fore the `Available Fields` label. Below this are fields that can be selected and shown in the middle of the screen. Find the `hostname` field below the `Available Fields` and click `add` . Notice now, in the message pain, each message's hostname is displayed. More fields may be added.

To create a query for logs, the `Add a filter +` link right below the search box may be used. This will allow us to build queries using the fields of the messages. For example, if we wanted to see all log messages from the `openshift-logging` namespace, we can do the following:

1. Click on `Add a filter +` .
2. In the `Fields` input box, start typing `kubernetes.namespace_name` . Notice all of the available fields that we can use to build the query
3. Next, select `is` .
4. In the `Value` field, type in `openshift-logging`
5. Click the "Save" button

Now, in the center of the screen you will see all of the logs from the `openshift-logging` namespace.

Of course, you may add more filters to refine the query.

One other neat option that Kibana allows you to do is save queries to use for later. To save a query do the following:

1. click on `Save` at the top of the screen.
2. Type in the name you would like to save it as. In this case, let's type in `openshift-logging Namespace`

Once this has been saved, it can be used at a later time by hitting the `Open` button and selecting this query.

Please take time to explore the *Kibana* page and get experience by adding and doing more queries. This will be helpful when using a production cluster, you will be able to get the exact logs that you are looking for in a single place.

External (LDAP) Authentication Providers, Users, and Groups