OpenShift Dedicated 4

Operators

Operators in OpenShift Dedicated 4

Last Updated: 2020-06-18

OpenShift Dedicated 4 Operators

Operators in OpenShift Dedicated 4

Legal Notice

Abstract

This document details information on using Operators with OpenShfit Dedicated

Table of Contents

CHAPTER 1. UNDERSTANDING OPERATORS

Conceptually, *Operators* take human operational knowledge and encode it into software that is more easily shared with consumers.

Operators are pieces of software that ease the operational complexity of running another piece of software. They act like an extension of the software vendor's engineering team, watching over a Kubernetes environment (such as OpenShift Dedicated) and using its current state to make decisions in real time. Advanced Operators are designed to handle upgrades seamlessly, react to failures automatically, and not take shortcuts, like skipping a software backup process to save time.

More technically, *Operators* are a method of packaging, deploying, and managing a Kubernetes application.

A Kubernetes application is an app that is both deployed on Kubernetes and managed using the Kubernetes APIs and **kubectl** or **oc** tooling. To be able to make the most of Kubernetes, you require a set of cohesive APIs to extend in order to service and manage your apps that run on Kubernetes. Think of Operators as the runtime that manages this type of app on Kubernetes.

1.1. WHY USE OPERATORS?

Operators provide:

- Repeatability of installation and upgrade.
- Constant health checks of every system component.
- Over-the-air (OTA) updates for OpenShift components and ISV content.
- A place to encapsulate knowledge from field engineers and spread it to all users, not just one or two.

Why deploy on Kubernetes?

Kubernetes (and by extension, OpenShift Dedicated) contains all of the primitives needed to build complex distributed systems – secret handling, load balancing, service discovery, autoscaling – that work across on-premise and cloud providers.

Why manage your app with Kubernetes APIs and kubectl tooling?

These APIs are feature rich, have clients for all platforms and plug into the cluster's access control/auditing. An Operator uses the Kubernetes' extension mechanism, Custom Resource Definitions (CRDs), so your custom object, for example **MongoDB**, looks and acts just like the built-in, native Kubernetes objects.

How do Operators compare with Service Brokers?

A Service Broker is a step towards programmatic discovery and deployment of an app. However, because it is not a long running process, it cannot execute Day 2 operations like upgrade, failover, or scaling. Customizations and parameterization of tunables are provided at install time, versus an Operator that is constantly watching your cluster's current state. Off-cluster services continue to be a good match for a Service Broker, although Operators exist for these as well.

1.2. OPERATOR FRAMEWORK

The Operator Framework is a family of tools and capabilities to deliver on the customer experience described above. It is not just about writing code; testing, delivering, and updating Operators is just as important. The Operator Framework components consist of open source tools to tackle these

problems:

Operator SDK

The Operator SDK assists Operator authors in bootstrapping, building, testing, and packaging their own Operator based on their expertise without requiring knowledge of Kubernetes API complexities.

Operator Lifecycle Manager

The Operator Lifecycle Manager (OLM) controls the installation, upgrade, and role-based access control (RBAC) of Operators in a cluster. Deployed by default in OpenShift Dedicated 4.

Operator Registry

The Operator Registry stores ClusterServiceVersions (CSVs) and Custom Resource Definitions (CRDs) for creation in a cluster and stores Operator metadata about packages and channels. It runs in a Kubernetes or OpenShift cluster to provide this Operator catalog data to the OLM.

OperatorHub

The OperatorHub is a web console for cluster administrators to discover and select Operators to install on their cluster. It is deployed by default in OpenShift Dedicated.

Operator Metering

Operator Metering collects operational metrics about Operators on the cluster for Day 2 management and aggregating usage metrics.

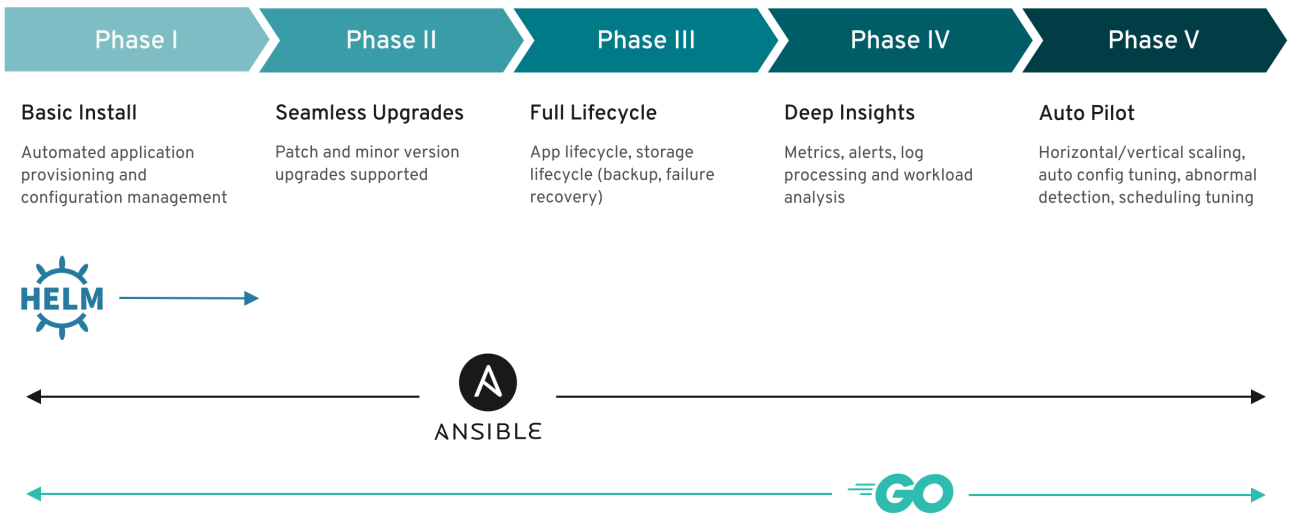
These tools are designed to be composable, so you can use any that are useful to you.

1.3. OPERATOR MATURITY MODEL

The level of sophistication of the management logic encapsulated within an Operator can vary. This logic is also in general highly dependent on the type of the service represented by the Operator.

One can however generalize the scale of the maturity of an Operator's encapsulated operations for certain set of capabilities that most Operators can include. To this end, the following Operator Maturity model defines five phases of maturity for generic day two operations of an Operator:

Figure 1.1. Operator maturity model



The above model also shows how these capabilities can best be developed through the Operator SDK's Helm, Go, and Ansible capabilities.

CHAPTER 2. ADDING OPERATORS TO A CLUSTER

This guide walks cluster administrators through installing Operators to an OpenShift Dedicated cluster and subscribing Operators to namespaces.

2.1. INSTALLING OPERATORS FROM THE OPERATORHUB

As a cluster administrator, you can install an Operator from the OperatorHub using the OpenShift Dedicated web console. You can then subscribe the Operator to the default **openshift-operators** namespace to make it available for developers on your cluster.

In OpenShift Dedicated clusters, a curated list of Operators is made available for installation from the OperatorHub. Administrators can only install Operators to the default **openshift-operators** namespace, except for the Logging Operator, which requires the **openshift-logging** namespace.

NOTE

Privileged and custom Operators cannot be installed.

During installation, you must determine the following initial settings for the Operator:

Installation Mode

In OpenShift Dedicated clusters, you can choose **All namespaces on the cluster (default)**to have the Operator installed on all namespaces. This makes the Operator available to all users and projects.

Update Channel

If an Operator is available through multiple channels, you can choose which channel you want to subscribe to. For example, to deploy from the **stable** channel, if available, select it from the list.

Approval Strategy

You can choose Automatic or Manual updates. If you choose Automatic updates for an installed Operator, when a new version of that Operator is available, the Operator Lifecycle Manager (OLM) automatically upgrades the running instance of your Operator without human intervention. If you select Manual updates, when a newer version of an Operator is available, the OLM creates an update request. As a cluster administrator, you must then manually approve that update request to have the Operator updated to the new version.

2.1.1. Installing from the OperatorHub using the web console

This procedure uses the Couchbase Operator as an example to install and subscribe to an Operator from the OperatorHub using the OpenShift Dedicated web console.

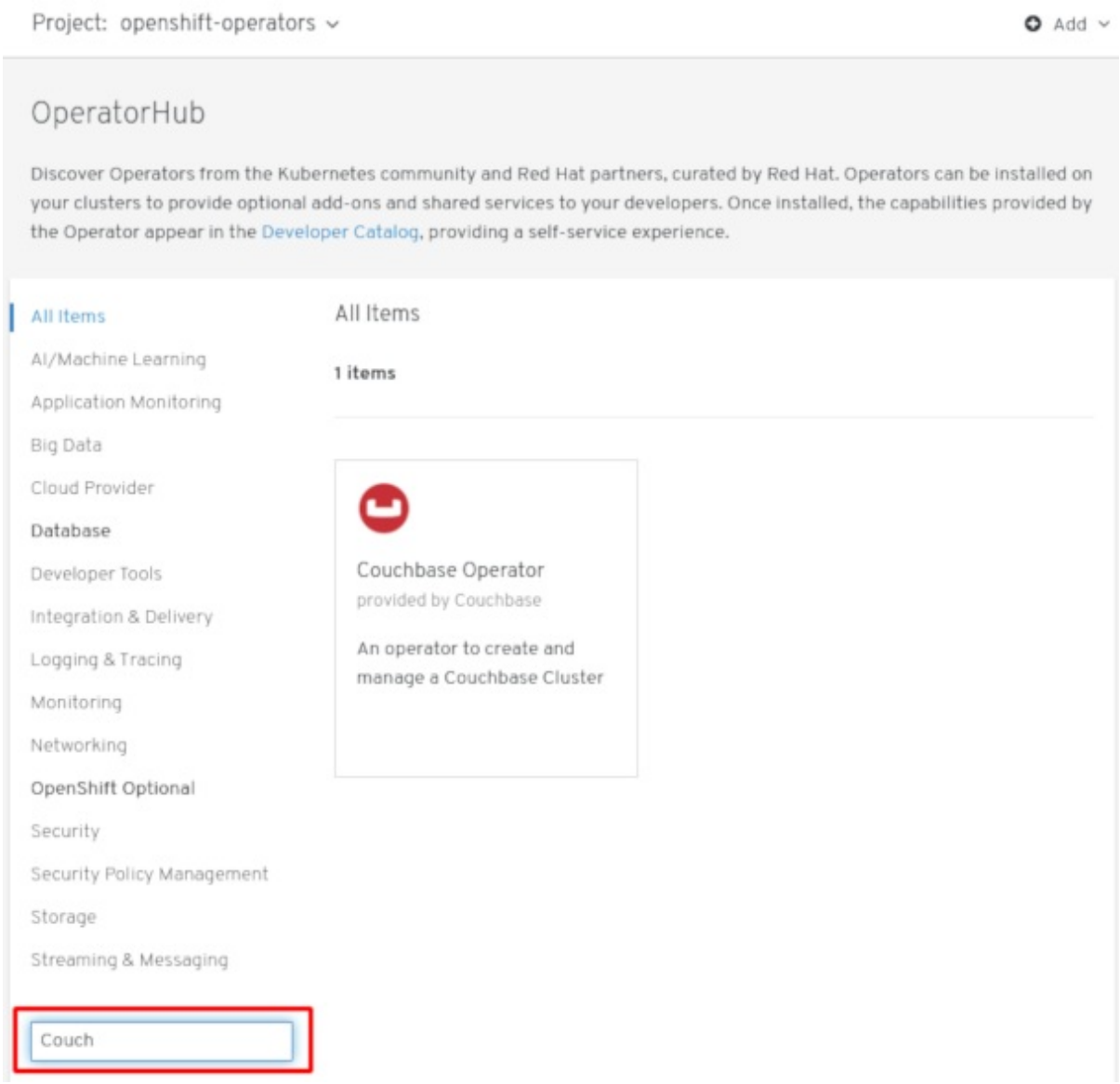
Prerequisites

- Access to an OpenShift Dedicated cluster using an account with **dedicated-admins-cluster** permissions.

Procedure

1. Navigate in the web console to the **Operators → OperatorHub** page.
2. Scroll or type a keyword into the **Filter by keyword** box (in this case, **Couchbase**) to find the Operator you want.

Figure 2.1. Filter Operators by keyword



3. Select the Operator. For a Community Operator, you are warned that Red Hat does not certify those Operators. You must acknowledge that warning before continuing. Information about the Operator is displayed.

4. Read the information about the Operator and click **Install**.

5. On the **Create Operator Subscription** page:

    a. Select one of the following:

    - **All namespaces on the cluster (default)** installs the Operator in the default **openshift-operators** namespace to watch and be made available to all namespaces in the cluster. This option is not always available.

    - **A specific namespace on the cluster** allows you to choose a specific, single namespace in which to install the Operator. The Operator will only watch and be made available for use in this single namespace. If you are installing the Cluster Logging Operator, choose this option to select the **openshift-logging** namespace.

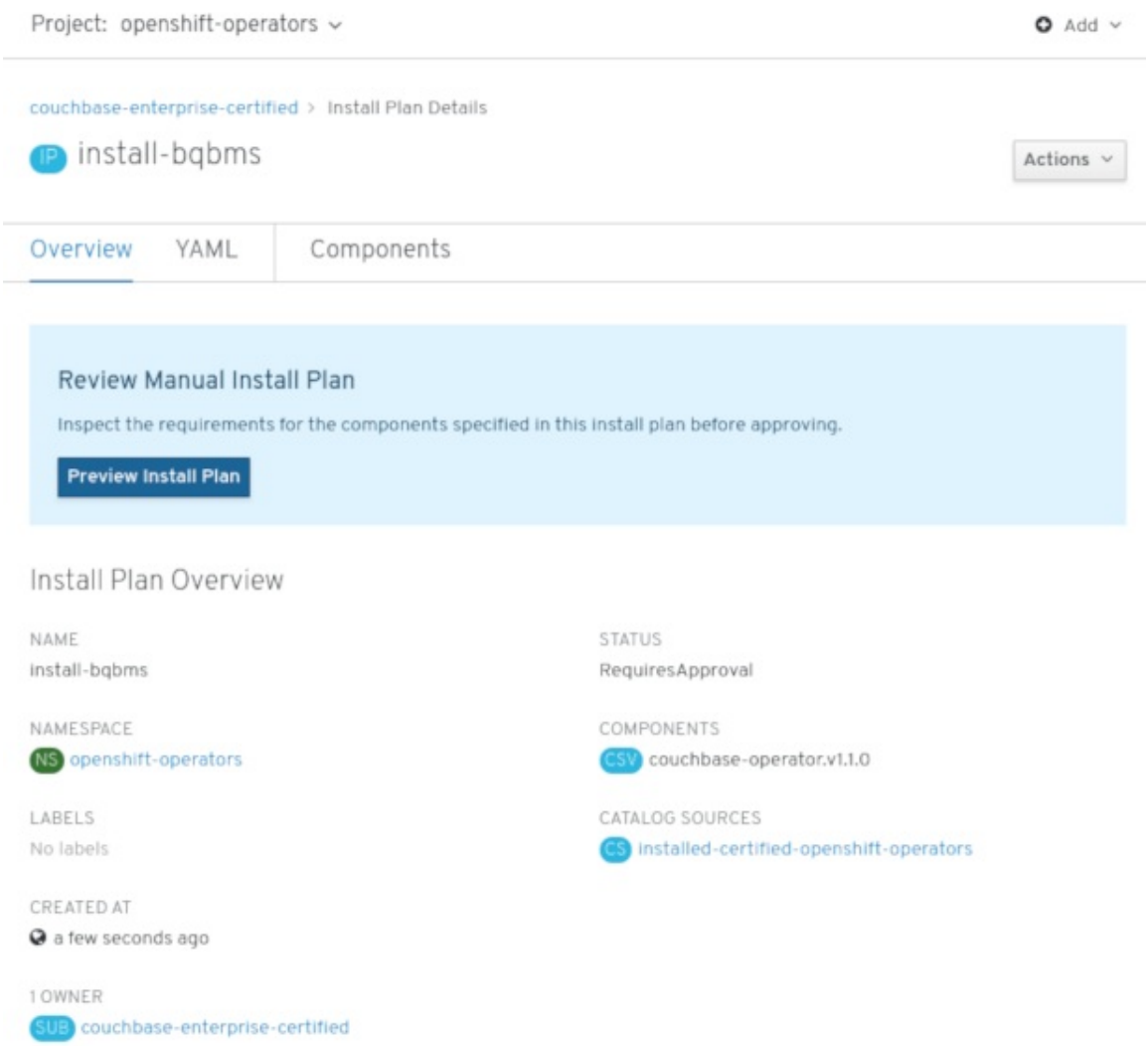
    b. Select an **Update Channel** (if more than one is available).

    c. Select **Automatic** or **Manual** approval strategy, as described earlier.

6. Click **Subscribe** to make the Operator available to the selected namespaces on this OpenShift Dedicated cluster.

   a. If you selected a Manual approval strategy, the Subscription's upgrade status will remain **Upgrading** until you review and approve its Install Plan.

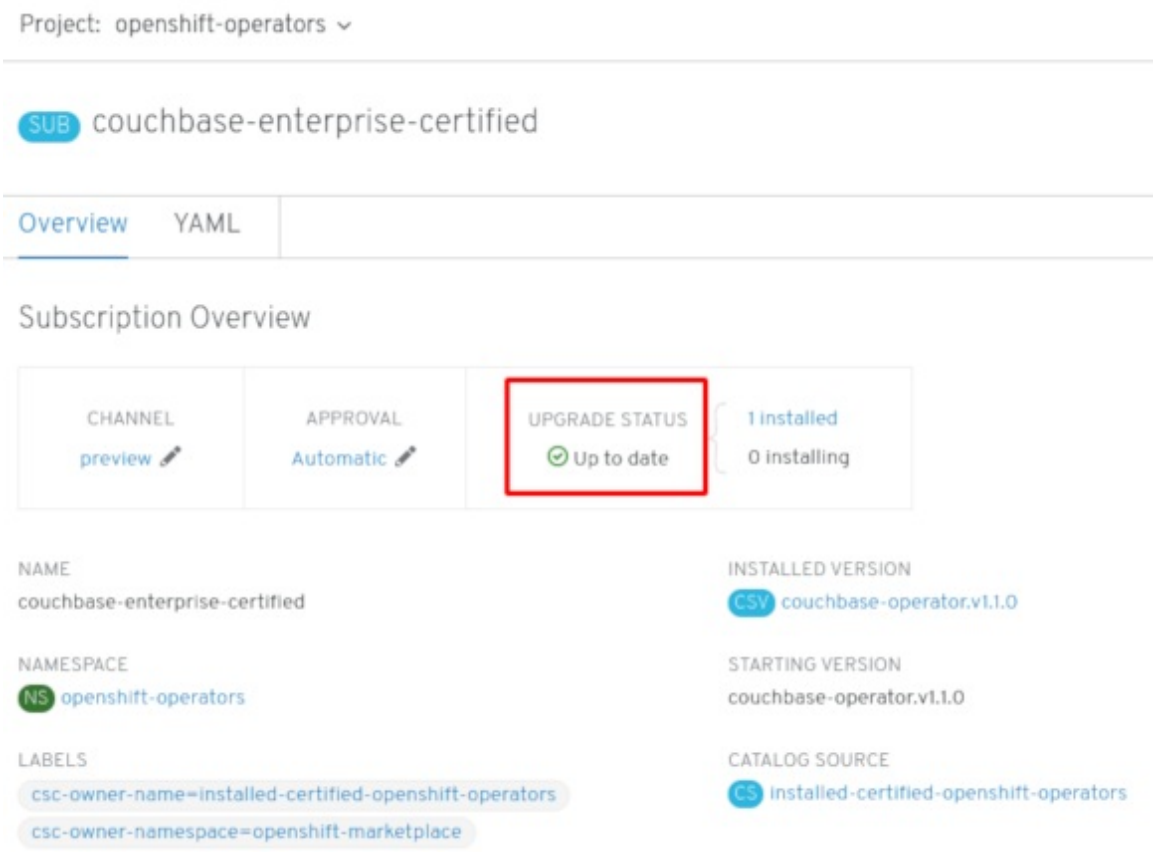
   Figure 2.2. Manually approving from theInstall Plan page



   After approving on the **Install Plan** page, the Subscription upgrade status moves to **Up to date**.

   b. If you selected an Automatic approval strategy, the upgrade status should resolve to **Up to date** without intervention.

Figure 2.3. Subscription upgrade status Up to date



7. After the Subscription's upgrade status is **Up to date**, select **Operators → Installed Operators** to verify that the **Couchbase** ClusterServiceVersion (CSV) eventually shows up and its **Status** ultimately resolves to **InstallSucceeded** in the relevant namespace.

NOTE

For the **All namespaces...** Installation Mode, the status resolves to **InstallSucceeded** in the **openshift-operators** namespace, but the status is **Copied** if you check in other namespaces.

If it does not:

a. Check the logs in any Pods in the **openshift-operators** project (or other relevant namespace if **A specific namespace...** Installation Mode was selected) on the **Workloads → Pods** page that are reporting issues to troubleshoot further.

CHAPTER 3. DELETING OPERATORS FROM A CLUSTER

The following describes how to delete Operators from a cluster using either the web console or the CLI.

3.1. DELETING OPERATORS FROM A CLUSTER USING THE WEB CONSOLE

Cluster administrators can delete installed Operators from a selected namespace by using the web console.

Prerequisites

- Access to an OpenShift Dedicated cluster web console using an account with **dedicated-admins-cluster** permissions.

Procedure

1. From the **Operators → Installed Operators** page, scroll or type a keyword into the **Filter by name** to find the Operator you want. Then, click on it.
2. On the right-hand side of the **Operator Details** page, select **Uninstall Operator** from the **Actions** drop-down menu.
3. When prompted by the **Remove Operator Subscription** window, you can select the **Also completely remove the Operator from the selected namespace** check box to remove all components related to the Operator. This removes the CSV, which in turn removes the Pods, Deployments, CRDs, and CRs associated with the Operator. Leaving this unchecked results in only the Subscription being removed.
4. Select **Remove**. This Operator will stop running and no longer receive updates.

3.2. DELETING OPERATORS FROM A CLUSTER USING THE CLI

Cluster administrators can delete installed Operators from a selected namespace by using the CLI.

Prerequisites

- Access to an OpenShift Dedicated cluster using an account with **dedicated-admins-cluster** permissions.
- **oc** command installed on workstation.

Procedure

1. Check the current version of the subscribed Operator (for example, **jaeger**) in the **currentCSV** field:

```
$ oc get subscription jaeger -n openshift-operators -o yaml | grep currentCSV
  currentCSV: jaeger-operator.v1.8.2
```

2. Delete the Operator's Subscription (for example, **jaeger**):

```
$ oc delete subscription jaeger -n openshift-operators
subscription.operators.coreos.com "jaeger" deleted
```

3. Delete the CSV for the Operator in the target namespace using the **currentCSV** value from the previous step:

```
$ oc delete clusterserviceversion jaeger-operator.v1.8.2 -n openshift-operators
clusterserviceversion.operators.coreos.com "jaeger-operator.v1.8.2" deleted
```

CHAPTER 4. CREATING APPLICATIONS FROM INSTALLED OPERATORS

This guide walks developers through an example of creating applications from an installed Operator using the OpenShift Dedicated web console.

4.1. CREATING AN ETCD CLUSTER USING AN OPERATOR

This procedure walks through creating a new etcd cluster using the etcd Operator, managed by the Operator Lifecycle Manager (OLM).

Prerequisites

- Access to an OpenShift Dedicated 4 cluster.
- The etcd Operator already installed cluster-wide by an administrator.

Procedure

1. Create a new project in the OpenShift Dedicated web console for this procedure. This example uses a project called **my-etcd**.
2. Navigate to the **Operators → Installed Operators** page. The Operators that have been installed to the cluster by the cluster administrator and are available for use are shown here as a list of ClusterServiceVersions (CSVs). CSVs are used to launch and manage the software provided by the Operator.

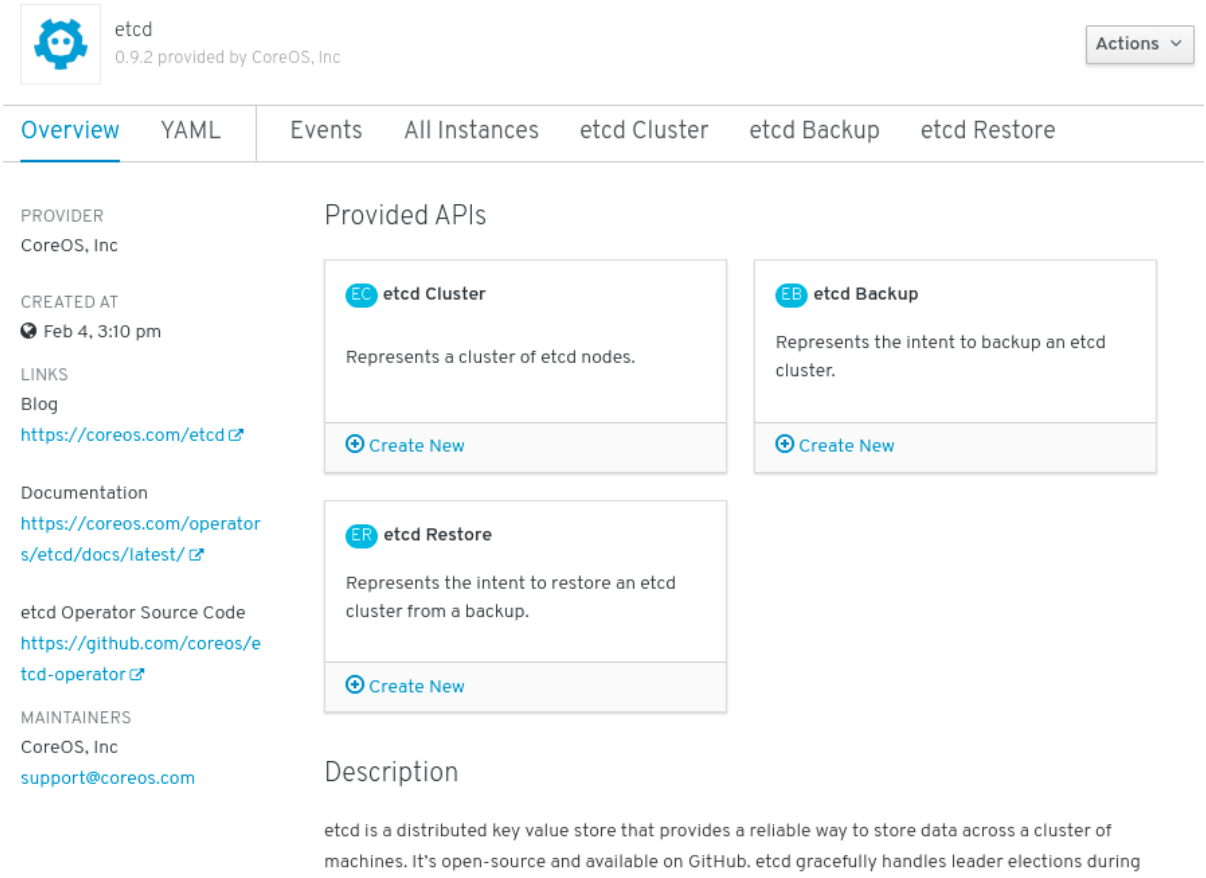
   TIP

   You can get this list from the CLI using:

   ```
   $ oc get csv
   ```

3. On the **Installed Operators** page, click **Copied**, and then click the etcd Operator to view more details and available actions:

Figure 4.1. etcd Operator overview



As shown under **Provided APIs**, this Operator makes available three new resource types, including one for an **etcd Cluster** (the **EtcdCluster** resource). These objects work similar to the built-in native Kubernetes ones, such as **Deployments** or **ReplicaSets**, but contain logic specific to managing etcd.

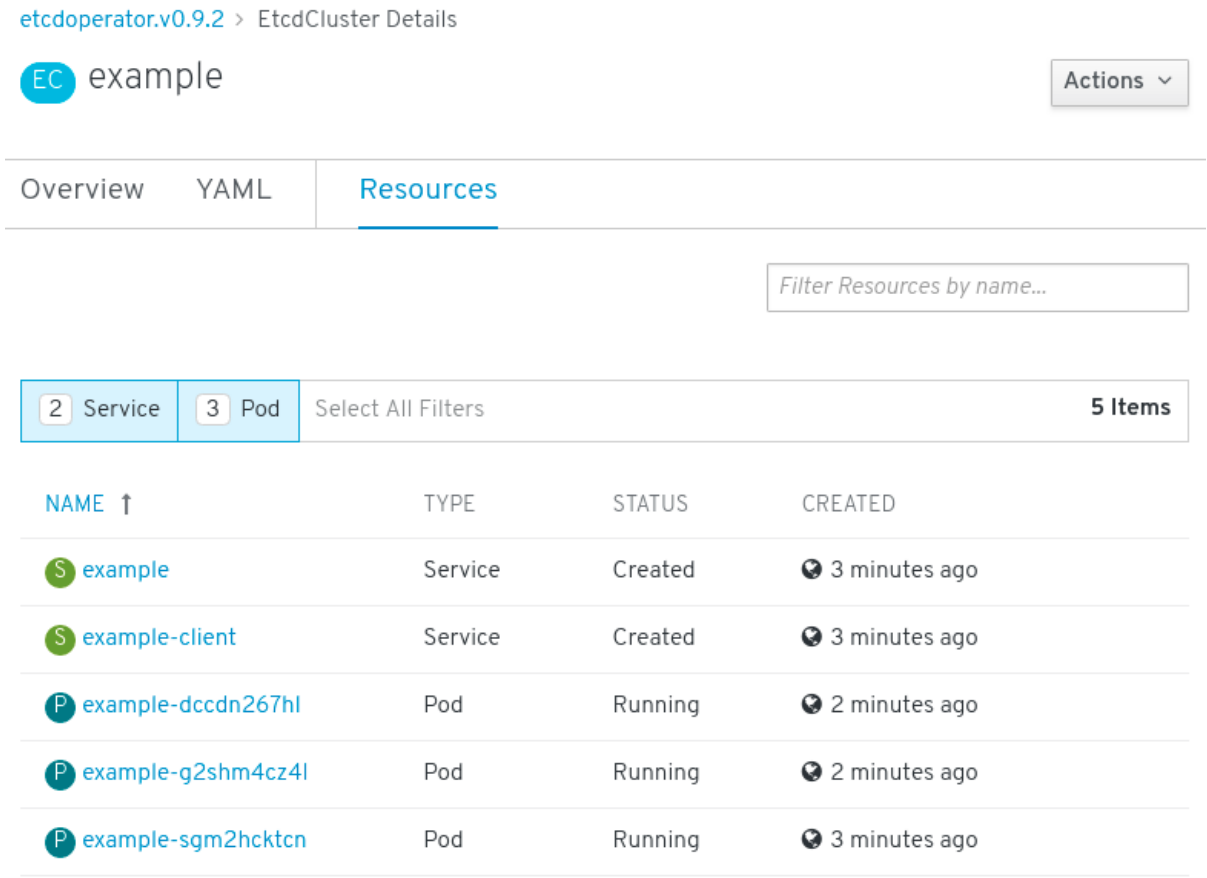
4. Create a new etcd cluster:

   a. In the **etcd Cluster** API box, click **Create New**.

   b. The next screen allows you to make any modifications to the minimal starting template of an **EtcdCluster** object, such as the size of the cluster. For now, click **Create** to finalize. This triggers the Operator to start up the Pods, Services, and other components of the new etcd cluster.

5. Click the **Resources** tab to see that your project now contains a number of resources created and configured automatically by the Operator.

Figure 4.2. etcd Operator resources



Verify that a Kubernetes service has been created that allows you to access the database from other Pods in your project.

6. All users with the **edit** role in a given project can create, manage, and delete application instances (an etcd cluster, in this example) managed by Operators that have already been created in the project, in a self-service manner, just like a cloud service. If you want to enable additional users with this ability, project administrators can add the role using the following command:

```
$ oc policy add-role-to-user edit <user> -n <target_project>
```

You now have an etcd cluster that will react to failures and rebalance data as Pods become unhealthy or are migrated between nodes in the cluster. Most importantly, cluster administrators or developers with proper access can now easily use the database with their applications.

CHAPTER 5. VIEWING OPERATOR STATUS

Understanding the state of the system in Operator Lifecycle Manager (OLM) is important for making decisions about and debugging problems with installed Operators. OLM provides insight into Subscriptions and related Catalog Source resources regarding their state and actions performed. This helps users better understand the healthiness of their Operators.

5.1. CONDITION TYPES

Subscriptions can report the following condition types:

Table 5.1. Subscription condition types

Condition	Description
CatalogSourcesUnhealthy	Some or all of the Catalog Sources to be used in resolution are unhealthy.
InstallPlanMissing	A Subscription's InstallPlan is missing.
InstallPlanPending	A Subscription's InstallPlan is pending installation.
InstallPlanFailed	A Subscription's InstallPlan has failed.

5.2. VIEWING OPERATOR STATUS USING THE CLI

You can view Operator status using the CLI.

Procedure

1. Use the **oc describe** command to inspect the Subscription resource:

```
$ oc describe sub <subscription_name>
```

2. In the command output, find the **Conditions** section:

```
Conditions:
  Last Transition Time:  2019-07-29T13:42:57Z
  Message:               all available catalogsources are healthy
  Reason:                AllCatalogSourcesHealthy
  Status:                False
  Type:                  CatalogSourcesUnhealthy
```

CHAPTER 6. CRDS