Multi-node production-ready deployments

Operators and developers who want to set up production-ready deployments can follow the workflows for On-Premises Deployments or Ansible Playbooks.

Single machine, multi-broker and multi-cluster configurations

To bridge the gap between the developer environment quick starts and full-scale, multi-node deployments, you can start by pioneering **multi-broker clusters** and **multi-cluster** setups on a single machine, like your laptop.

Trying out these different setups is a great way to learn your way around the configuration files for Kafka broker and Control Center, and experiment locally with more sophisticated deployments. These setups more closely resemble real-world configurations and support data sharing and other scenarios for Confluent Platform specific features like Replicator, Self-Balancing, Cluster Linking, and multi-cluster Schema Registry.

- For a single cluster with multiple brokers, you must configure and start a single ZooKeeper, and as many brokers as you want to run in the cluster.
- For a multi-cluster deployment, you need as many ZooKeepers as you want clusters, and multiple Kafka server properties files (one for each broker).

Does all this run on my laptop?

Yes! These examples show you how to run all clusters and brokers on a single laptop or machine.

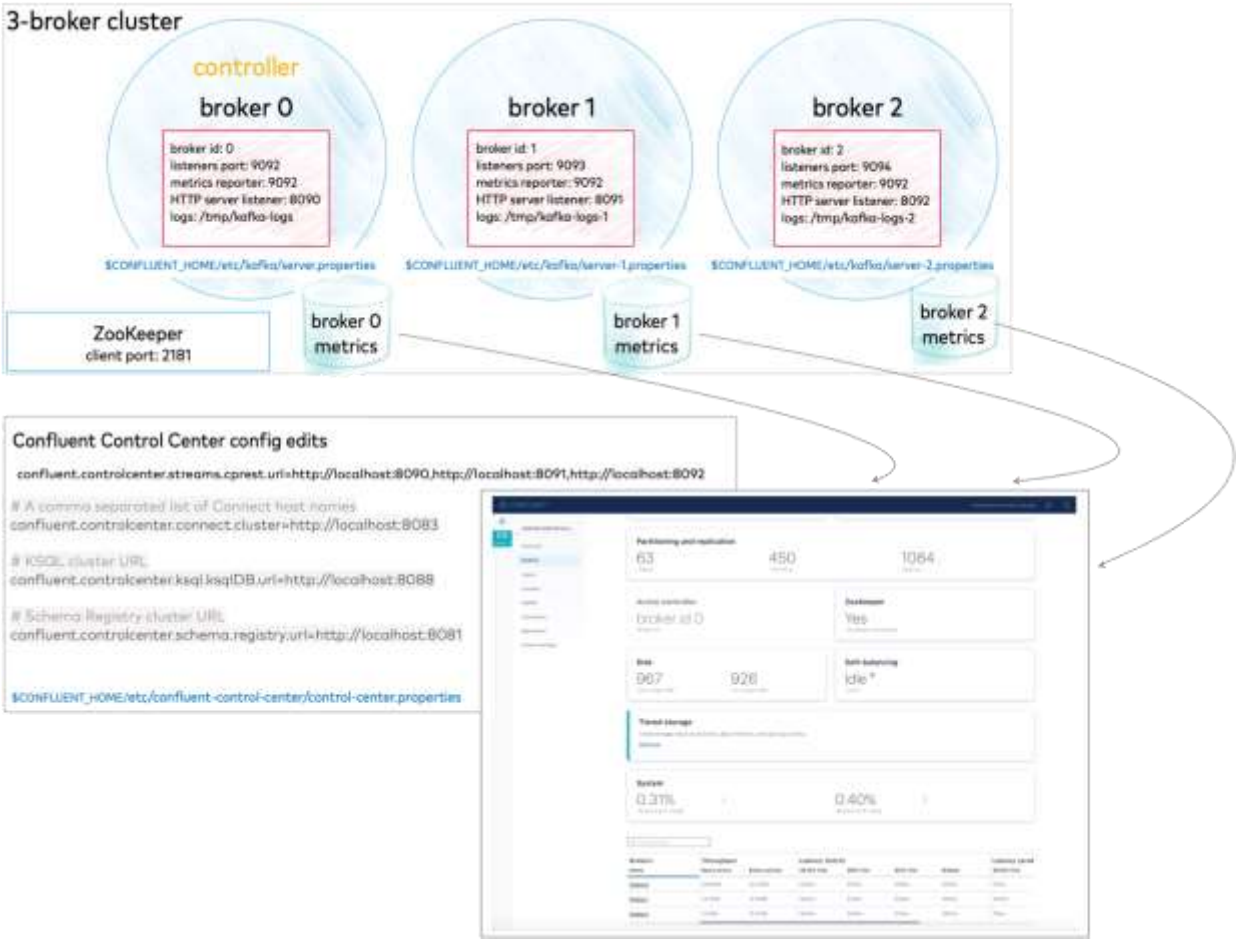
That said, you can easily extrapolate much of what you learn here to create similar deployments on your favorite cloud provider, using multiple virtual hosts. Use these examples as stepping stones to more complex deployments and feature integrations.

Run a multi-broker cluster

To run a single cluster with multiple brokers (3 brokers, for this example) you need:

- 1 ZooKeeper properties file
- 3 Kafka broker properties files with unique broker IDs, listener ports (to surface details for all brokers on Control Center), and log file directories.
- Control Center properties file with the REST endpoints for `controlcenter.cluster` mapped to your brokers.

- Metrics Reporter JAR file installed and enabled on the brokers. (If you start Confluent Platform as described below, from `$CONFLUENT_HOME/bin/`, the Metrics Reporter is automatically *installed* on the broker. Otherwise, you would need to add the path to the Metrics Reporter JAR file to your CLASSPATH.)
- Properties files for any other Confluent Platform components you want to run, with default settings to start with.



All of this is described in detail below.

Configure replication factors

The `server.properties` file that ships with Confluent Platform has replication factors set to `1` on several system topics to support development test environments and Confluent Quick Start scenarios. For real-world scenarios, however, a replication factor greater than `1` is preferable to support fail-over and auto-balancing capabilities on both system and user-created topics.

The following steps show you how to reset system topics replication factors and replicas to `2`, and uncomment the properties if needed so that your changes go into effect.

Make these changes, and then save the file.

1. Search in `$CONFLUENT_HOME/etc/kafka/server.properties` for all instances of `replication.factor` and set the values for these to a number that is less than the number of brokers but greater than 1. For this cluster, set all `replication.factor`'s to 2. Your search through `server.properties` should turn up these properties. If they are commented out, uncomment them:

```
offsets.topic.replication.factor=2
transaction.state.log.replication.factor=2
confluent.license.topic.replication.factor=2
confluent.metadata.topic.replication.factor=2
confluent.balancer.topic.replication.factor=2
```

2. In the same properties file, do a search to on `replicas`, uncomment these properties, and set their values to 2:

```
confluent.metrics.reporter.topic.replicas=2
confluent.security.event.logger.exporter.kafka.topic.replicas=2
```

3. If you want to run Connect, change replication factors in that properties file also. Search `$CONFLUENT_HOME/etc/kafka/connect-distributed.properties` for all instances of `replication.factor` and set the values for these to a number that is less than the number of brokers but greater than 1. For this cluster, set all `replication.factor`'s to 2. Your search through `connect-distributed.properties` should turn up these properties. If they are commented out, uncomment them:

```
offset.storage.replication.factor=2
config.storage.replication.factor=2
status.storage.replication.factor=2
```

Tip

- Limiting replicas and replication factors to `2` provides a safeguard for topic replication if you lose a broker, or the option to intentionally shrink the cluster by one broker using Self-Balancing Clusters. You can always expand back up to 3, or add more brokers. This configuration supports topic replication on a cluster of two or more brokers.
- When you create your topics, make sure that they also have the needed replication factor, depending on the number of brokers. Examples are shown in the section on creating topics in Kafka Commands Primer.

Create a basic configuration for a three-broker cluster

This example demos a cluster with three brokers.

Start with the `server.properties` file you updated for replication factors in the previous step, copy it and modify the configurations as shown below, renaming the new files to represent the other two brokers.

File	Configurations

server.properties	Use the defaults for these basics (if a value is commented out, leave it as such):
	
	`broker.id=0`
	
	`#listeners=PLAINTEXT://:9092`
	
	`log.dirs=/tmp/kafka-logs`
	
	Uncomment the following two lines to enable the Metrics Reporter and populate Control Center with broker metrics for all brokers. This same configuration can apply to all brokers in the cluster. There is no need to change this to match the listener port for each broker:

	`metric.reporters=io.confluent.metrics.reporter.ConfluentMetricsReporter` `confluent.metrics.reporter.bootstrap.servers=localhost:9092`
	Add the following listener configuration to specify the REST endpoint for this broker:
	`confluent.http.server.listeners=http://localhost:8090`

server-1.properties	Update the values for these basic properties to make them unique. (Be sure to uncomment `listeners`):
	
	`broker.id=1`
	
	`listeners=PLAINTEXT://:9093`
	
	`log.dirs=/tmp/kafka-logs-1`
	
	Make sure the following two lines are uncommented to enable the Metrics Reporter on this broker:

	`metric.reporters=io.confluent.metrics.reporter.ConfluentMetricsReporter`

File	Configurations

	`confluent.metrics.reporter.bootstrap.servers=localhost:9092`
	
	Add the listener configuration to specify the REST endpoint unique to this broker (if you copied `sever.properties`, just update the port number):
	
	`confluent.http.server.listeners=http://localhost:8091`

server-2.properties	Update the values for these basic properties to make them unique. (Be sure to uncomment `listeners`):
	
	`broker.id=2`
	
	`listeners=PLAINTEXT://:9094`
	
	`log.dirs=/tmp/kafka-logs-2`
	
	Make sure the following two lines are uncommented to enable the Metrics Reporter on this broker:
	
	`metric.reporters=io.confluent.metrics.reporter.ConfluentMetricsReporter`
	`confluent.metrics.reporter.bootstrap.servers=localhost:9092`
	
	Add the listener configuration to specify the REST endpoint unique to this broker (if you copied `sever.properties`, just update the port number):
	
	`confluent.http.server.listeners=http://localhost:8092`

When you have completed this step, you will have three server properties files in `$CONFLUENT_HOME/etc/kafka/`, one per broker:

- `server.properties` which corresponds to broker 0
- `server-1.properties` which corresponds to broker 1
- `server-2.properties` which corresponds to broker 2

Tip

In `server.properties` and other configuration files, commented out properties or those not listed at all, take the default values. For example, the commented out line

for `listeners` on broker 0 has the effect of setting a single listener to `PLAINTEXT://:9092`.

Configure Control Center with REST endpoints and advertised listeners (Optional)

This is an optional step, only needed if you want to use Confluent Control Center. It gives you a similar starting point as you get in the Quick Start for Confluent Platform (Local install), and an alternate way to work with and verify the topics and data you will create on the command line with `kafka-topics`.

You must tell Control Center about the REST endpoints for all brokers in your cluster, and the advertised listeners for the other components you may want to run. Without these configurations, the brokers and components will not show up on Control Center.

Make the following changes to `$CONFLUENT_HOME/etc/confluent-control-center/control-center.properties` and save the file.

1. Configure REST endpoints for the brokers.

   In the appropriate Control Center properties file, use `confluent.controlcenter.streams.cprest.url` to define the REST endpoints for `controlcenter.cluster`.

   In `$CONFLUENT_HOME/etc/confluent-control-center/control-center.properties`, uncomment the default value for the Kafka REST endpoint URL and modify it to match your multi-broker configuration as follows:

```
# Kafka REST endpoint URL
confluent.controlcenter.streams.cprest.url=http://localhost:8090,http://localh
ost:8091,http://localhost:8092
```

2. Uncomment the configurations for Kafka Connect, ksqlDB, and Schema Registry to provide Control Center with the default advertised URLs to for the component clusters.

```
# A comma separated list of Connect host names
confluent.controlcenter.connect.cluster=http://localhost:8083

# KSQL cluster URL
confluent.controlcenter.ksql.ksqlDB.url=http://localhost:8088

# Schema Registry cluster URL
confluent.controlcenter.schema.registry.url=http://localhost:8081
```

Tip

Component listeners are uncommented for you already in `control-center-dev.properties` which is used by `confluent local services start`, as in Quick Start for Confluent Platform (Local install), but in the file you are using here (`control-center.properties`), you must uncomment them.

Install the Datagen Connector (Optional)

Install the Kafka Connect Datagen source connector using the Confluent Hub client. This connector generates mock data for demonstration purposes and is not suitable for production. Confluent Hub is an online library of pre-packaged and ready-to-install extensions or add-ons for Confluent Platform and Kafka.

```
confluent-hub install \
   --no-prompt confluentinc/kafka-connect-datagen:latest
```

This is an optional step, but useful, as it gives you a similar starting point as you get in the Quick Start for Confluent Platform (Local install).

Start Confluent Platform

Follow these steps to start the servers in separate command windows.

1. Start ZooKeeper in its own command window.

```
./bin/zookeeper-server-start etc/kafka/zookeeper.properties
```

2. Start each of the brokers in separate command windows.

```
./bin/kafka-server-start etc/kafka/server.properties

./bin/kafka-server-start etc/kafka/server-1.properties

./bin/kafka-server-start etc/kafka/server-2.properties
```

3. Start each of these components in separate windows.

Tip

For this example, it is not necessary to start all of these. At a minimum, you will need ZooKeeper and the brokers (already started), and Kafka REST. However, it is useful to have all components running if you are just getting started with the platform, and want to explore everything. This gives you a similar starting point as you get in Quick Start for Confluent Platform (Local install), and enables you to work through the examples in that Quick Start in addition to the Kafka command examples provided here.

- Kafka REST

```
./bin/kafka-rest-start etc/kafka-rest/kafka-rest.properties
```

- (Optional) Kafka Connect

```
./bin/connect-distributed etc/kafka/connect-distributed.properties
```

- (Optional) ksqlDB Overview

```
./bin/ksql-server-start etc/ksqldb/ksql-server.properties
```

- (Optional) Schema Registry

```
./bin/schema-registry-start etc/schema-registry/schema-registry.properties
```

- (Optional) Finally, start Control Center in a separate command window.

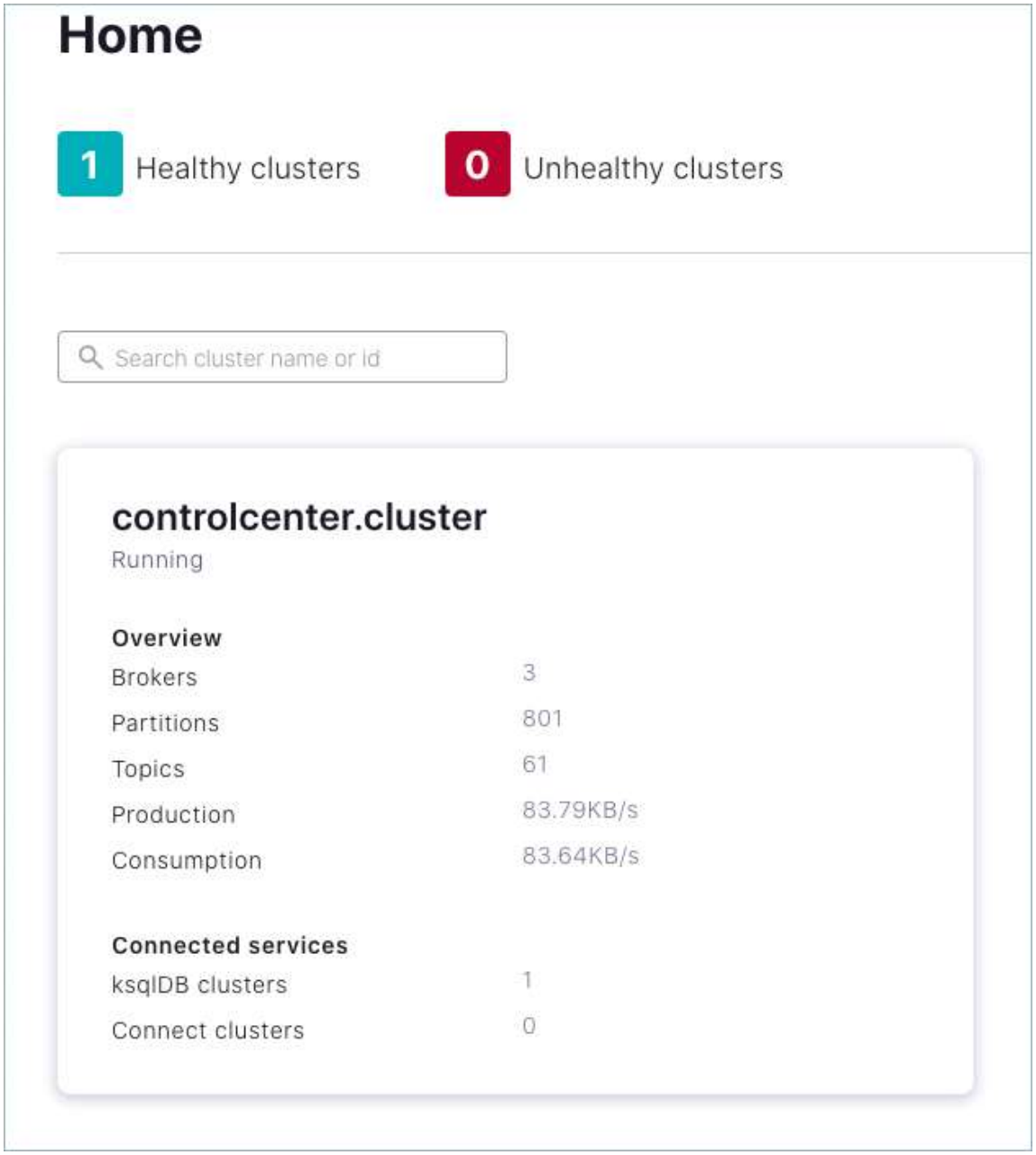
```
./bin/control-center-start etc/confluent-control-center/control-center.prop
erties
```

Explore Control Center (Optional)

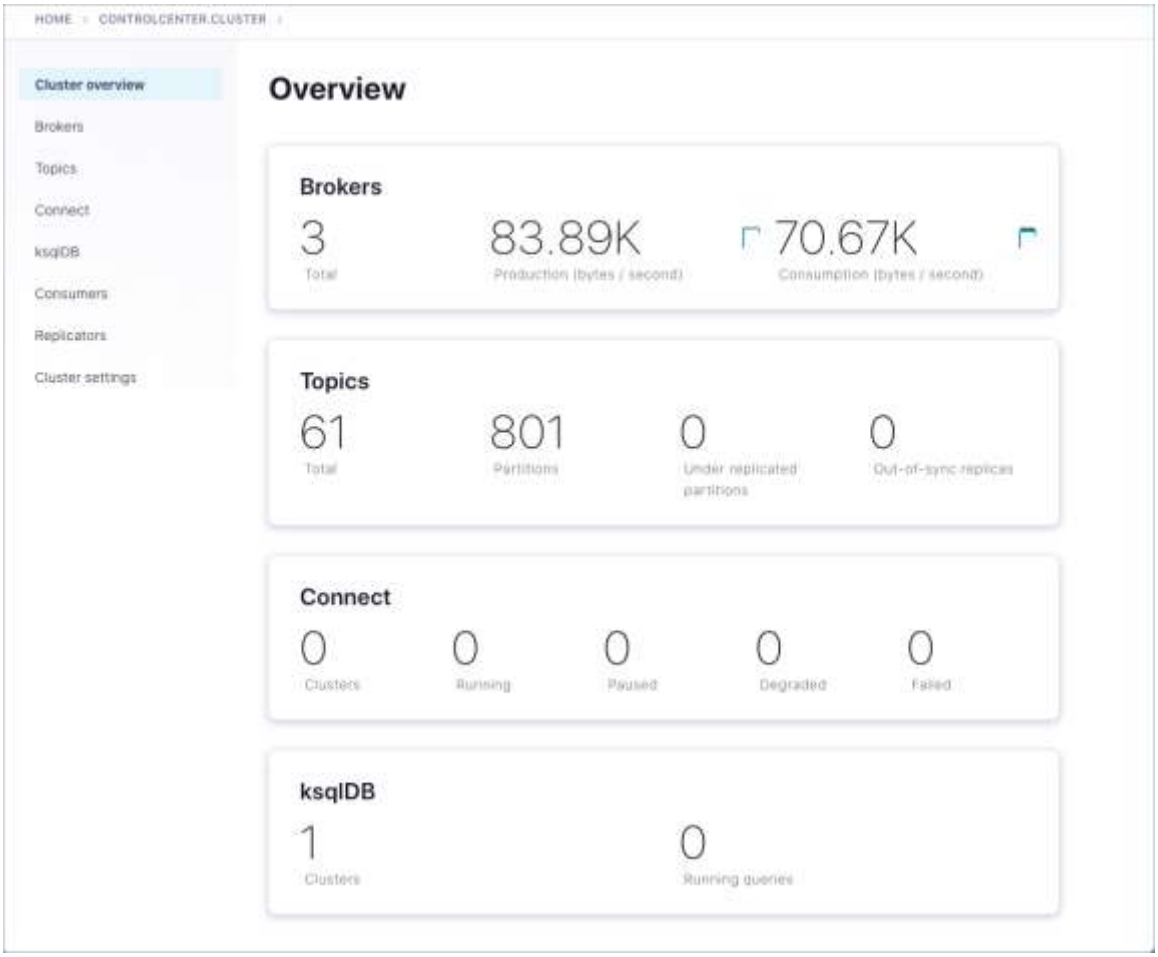
Bring up Confluent Control Center to verify the current status of your cluster, including lead broker (controller), topic data, and number of brokers. For a local deployment, Control Center is available at http://localhost:9021/ in your web browser.

The starting view of your **environment** in Control Center shows your cluster with 3 brokers.

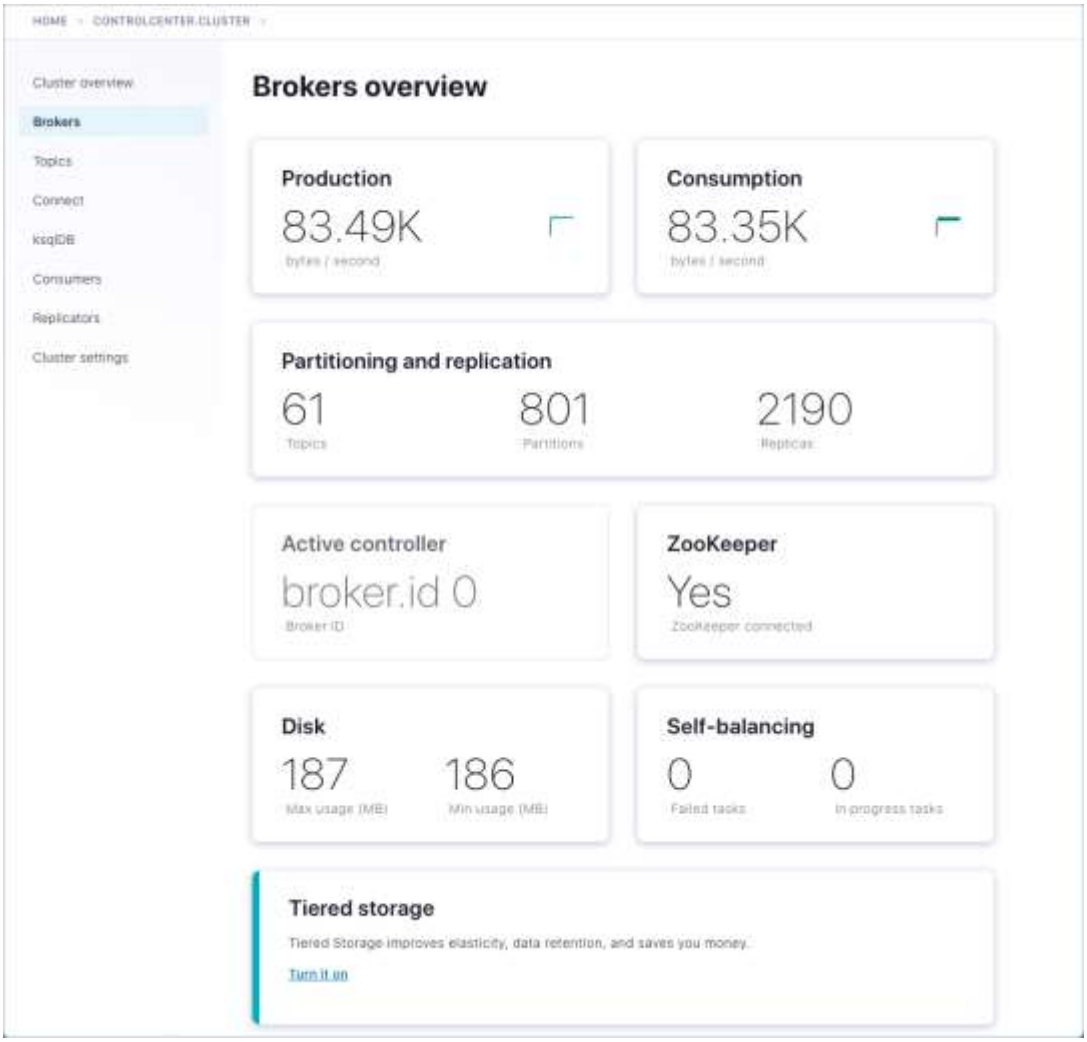
1. Click into the cluster card.



The cluster overview is displayed.



2. Click either the Brokers card or Brokers on the menu to view broker metrics.
    - Notice the card for **Active controller** indicating that the lead broker is **broker.id 0**, which was configured in `server.properties` when you specified `broker.id=0`. On a multi-broker cluster, the role of the controller can change hands if the current controller is lost.



- From the brokers list at the bottom of the page, you can view detailed metrics and drill down on each broker.

3. Finally, click **Topics** on the left menu.

   Note that only system (internal) topics are available at this point because you haven't created any topics of your own yet. The `default_ksql_processing_log` will show up as a topic if you configured and started ksqlDB.

Everything should work the same for the Quick Start steps. The only difference is that here you have a multi-broker cluster with replication factors set appropriately for additional examples, and the deployment in the quick starts (launched

with `confluent local services start`) is a single-broker cluster with replication factors set to `1` for a development-only environment.