**[What is Kafka ?](https://docs.datadoghq.com/integrations/faq/troubleshooting-and-deep-dive-for-kafka/" \l "what-is-kafka)**

Kafka is used for building real-time data pipelines and streaming apps. It is horizontally scalable, fault-tolerant, wicked fast, and runs in production in thousands of companies.”

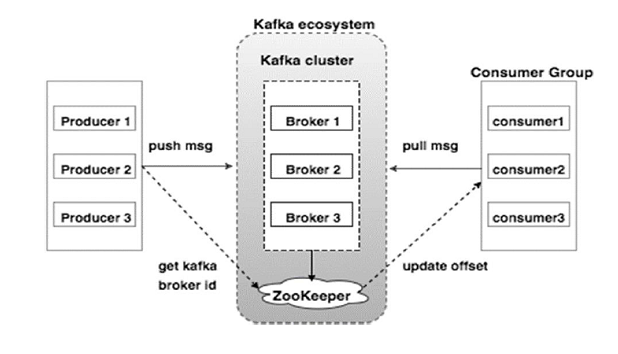
We are using Confluent Kafka to achieve the Messaging System, responsible for transferring data from one application to another.

**Confluent** is a fully managed Kafka service provider and enterprise stream processing platform Confluent Platform is a streaming platform that enables you to organize and manage data from many different sources with one reliable, high performance system.

## [Kafka Components:](https://docs.datadoghq.com/integrations/faq/troubleshooting-and-deep-dive-for-kafka/#kafka-components)

There are four main components to Kafka:

* **Broker**: Cluster of nodes responsible for establishing the mechanisms to write and read messages. (Main piece of Kafka, always in Java, usually managed by Apache Zookeeper)
* **Producer**: Application(s) that is writing the messages that you are interested in viewing. (Most commonly in Java, but possibly in other languages)
* **Consumer**: This is the application(s) that is receiving your set of messages. (Most commonly in Java, but possibly in other languages)
* **Topics** - Mailboxes of messages that Producers and Consumers subscribe to. When writing or reading a message in Kafka, specify which “topic” you are to read from. You can think of this like a channel in slack, you join the ones you want to post and read messages to. Each topic then has a list of offsets that informs you where you are in the number of messages you have read/have left to read.



## Role of Zookeeper

A critical dependency of Apache Kafka is Apache Zookeeper, which is a distributed configuration and synchronization service. Zookeeper serves as the coordination interface between the Kafka brokers and consumers. The Kafka servers share information via a Zookeeper cluster. Kafka stores basic metadata in Zookeeper such as information about topics, brokers, consumer offsets (queue readers) and so on.

Confluent Platform:

We are using Confluent Platform 5.3.1 , the Ansible playbooks to set up the cluster.

The Ansible playbooks provided by Confluent perform the following operations:

* Installs Confluent Platform packages.
* Starts services using systemd scripts.
* Provides variables for configuring various security settings between Confluent Platform component services:
  + SSL (one way and two way) with self-signed certs or custom certs

installed the services listed below using the Ansible playbooks provided by Confluent:

* Apache Zookeeper™
* Apache Kafka®
* Confluent Control Center
* Kafka Connect (distributed mode)

……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**Installation:**

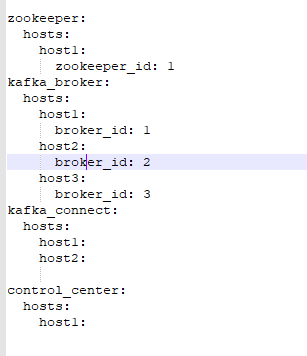
Using Ansible to install and configure Confluent Platform consists of a few steps. You create an inventory file, set up the variables you need, and then run the playbook.

Source code has been placed in our scm (gitlab)

<http://10.144.16.146/devops-itd2/sit-kafka.git>

Edit your inventory file and enter the correct hosts, SSH variables, and custom variables.

vi hosts**.**yml



Run the all.yml playbook.

ansible**-**playbook **-**i hosts**.**yml all**.**yml

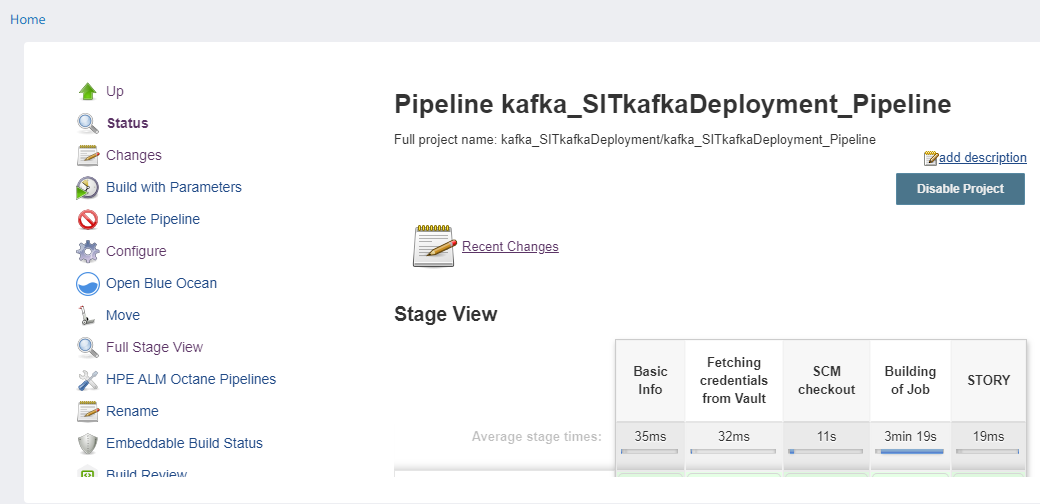
…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**IDP pipeline for Ansible installation:**

……………………………………………………………….

We have pipeline for Kafka installation for each environment.





Once the pipeline triggered it will run the ansible playbook and execute the respective roles in mentioned hosts which we have mentioned over hosts.yaml file.

**Kafka components and its properties:**

**Zookeeper:**

* Default configuration file: /etc/kafka/zookeeper.properties
* Log directory: /var/log/kafka

Log file: zookeeper-server.log

Broker:

* Default configuration file: /etc/kafka/server.properties
* Log directory: /var/log/kafka

Log file: server.log

Connector:

* Default configuration file: /etc/kafka/connect-distributed. Properties
* Log directory: /var/log/kafka

Log directory : connect-distributed.log

Control center:

* Default configuration file: config\_file: /etc/confluent-control-center/control-center-production.properties
* Log directory: /var/log/kafka

Log file : control-center.log

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………...

**Troubleshooting command :**

\*\* no command should run as root user\*\*

**Zookeeper:**

sudo systemctl status confluent-zookeeper.service

sudo systemctl start confluent-zookeeper.service

sudo systemctl stop confluent-zookeeper.service

**Broker:**

sudo systemctl status confluent-kafka.service

sudo systemctl start confluent-kafka.service

sudo systemctl stop confluent-kafka.service

**Connector:**

sudo systemctl status confluent-kafka-connect.service

sudo systemctl start confluent-kafka-connect.service

sudo systemctl stop confluent-kafka-connect.service

**Control center:**

sudo systemctl status confluent-control-center.service

sudo systemctl start confluent-control-center.service

sudo systemctl stop confluent-control-center.service

**Monitor cluster by control center:**

Confluent Control Center is a web-based tool for managing and monitoring Apache Kafka®. Control Center facilitates building and monitoring production data pipelines and streaming applications.

[**Clusters home page**](https://docs.confluent.io/current/control-center/userguide.html#controlcenter-userguide)

View healthy and unhealthy clusters at a glance, search for a cluster being managed by Control Center, and click on a cluster tile to drill into the [Brokers overview](https://docs.confluent.io/current/control-center/brokers.html#controlcenter-userguide-brokers) of critical metrics.

[**Alerts**](https://docs.confluent.io/current/control-center/alerts/index.html#controlcenter-userguide-alerts)

Define the trigger criteria to detect anomalous events in monitoring data and perform actions that trigger an alert when those events occur. Set triggers, actions, and view alert history across all clusters being managed by Control Center..

[**Brokers overview**](https://docs.confluent.io/current/control-center/brokers.html#controlcenter-userguide-brokers)

View essential Kafka metrics for brokers in a cluster.

[**Topics**](https://docs.confluent.io/current/control-center/topics/index.html#controlcenter-userguide-topics)

Add and edit topics, view production and consumption metrics for a topic, browse and download messages, manage Schema Registry for a topic, and edit topic configuration settings.

[**Connect**](https://docs.confluent.io/current/control-center/connect.html#controlcenter-userguide-connect)

Use Control Center to manage and monitor [Kafka Connect](https://docs.confluent.io/current/connect/index.html#kafka-connect), the toolkit for connecting external systems to Kafka. You can easily add new sources to load data from external data systems and new sinks to write data into external data systems. Additionally, you can manage, monitor, and configure connectors with Control Center. View the status of each connector and its tasks.

[**Consumers**](https://docs.confluent.io/current/control-center/consumers.html#controlcenter-userguide-consumers)

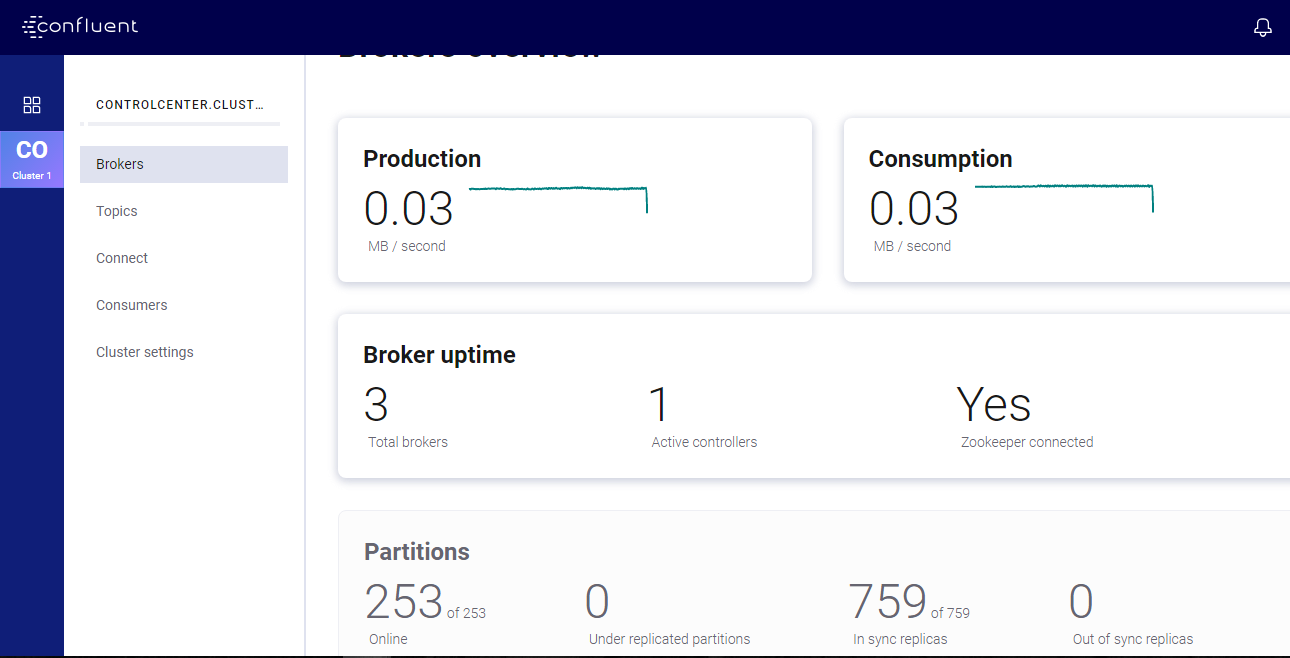
View all the consumer groups associated with a selected Kafka cluster, including the number of consumers per group and the number of topics being consumed. View Consumer Lag across all relevant topics. The Consumers feature contains the redesigned streams monitoring page.

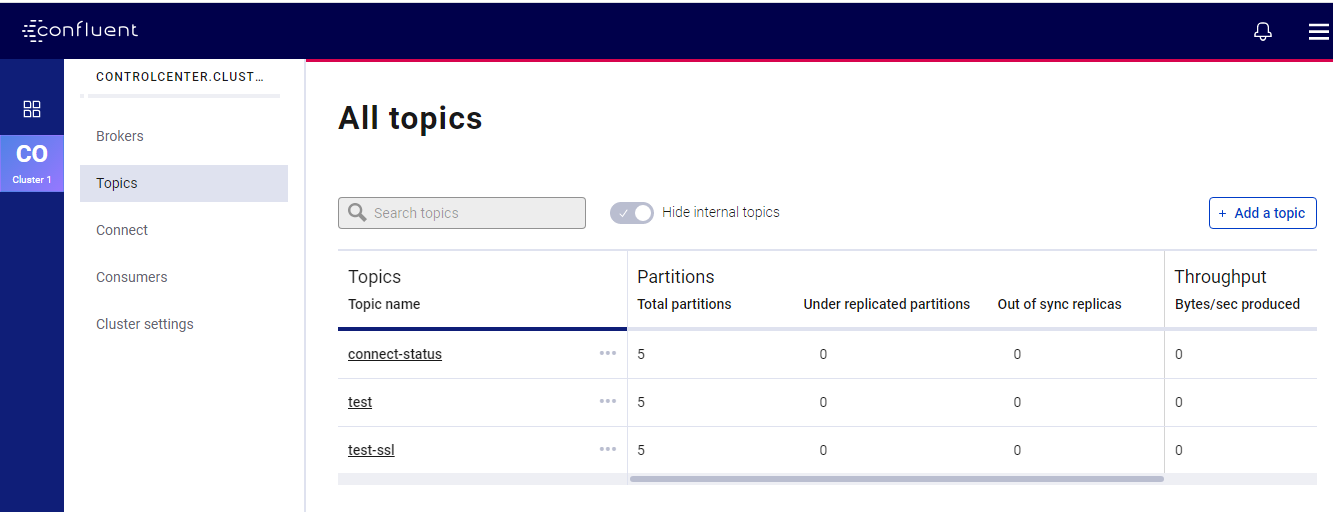
[**Data Streams**](https://docs.confluent.io/current/control-center/monitoring.html#controlcenter-userguide-monitoring)

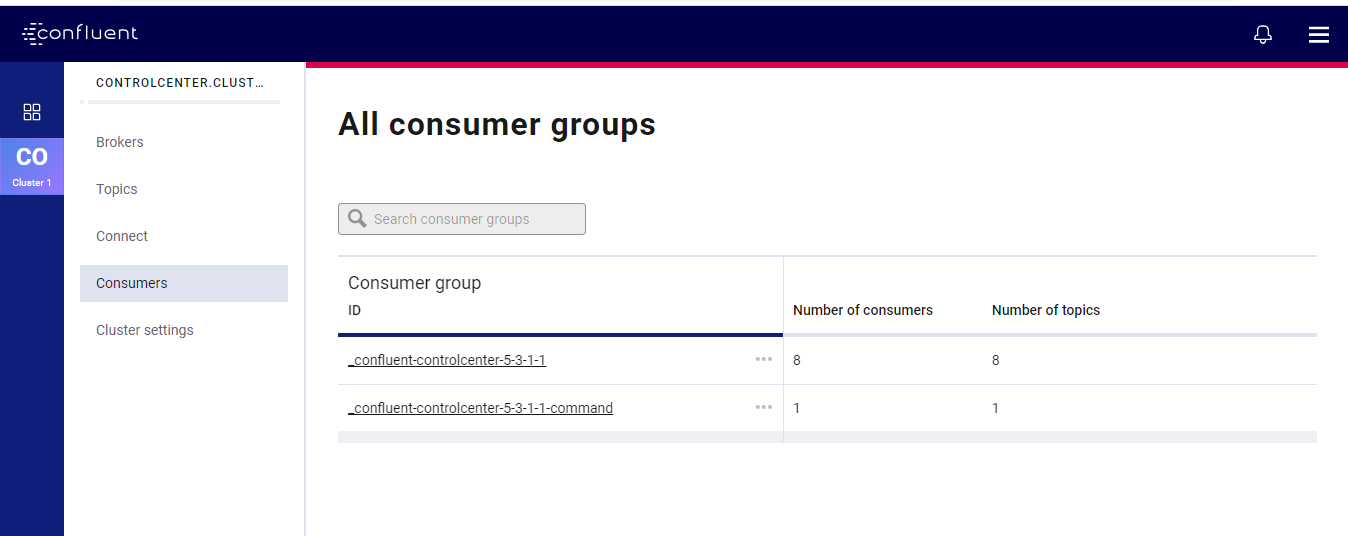
Use Control Center to monitor your data streams end-to-end, from producer to consumer. Use Control Center to verify that every message sent is received (and received only once), and to measure system performance end-to-end. Drill down to better understand cluster usage, and identify any problems.

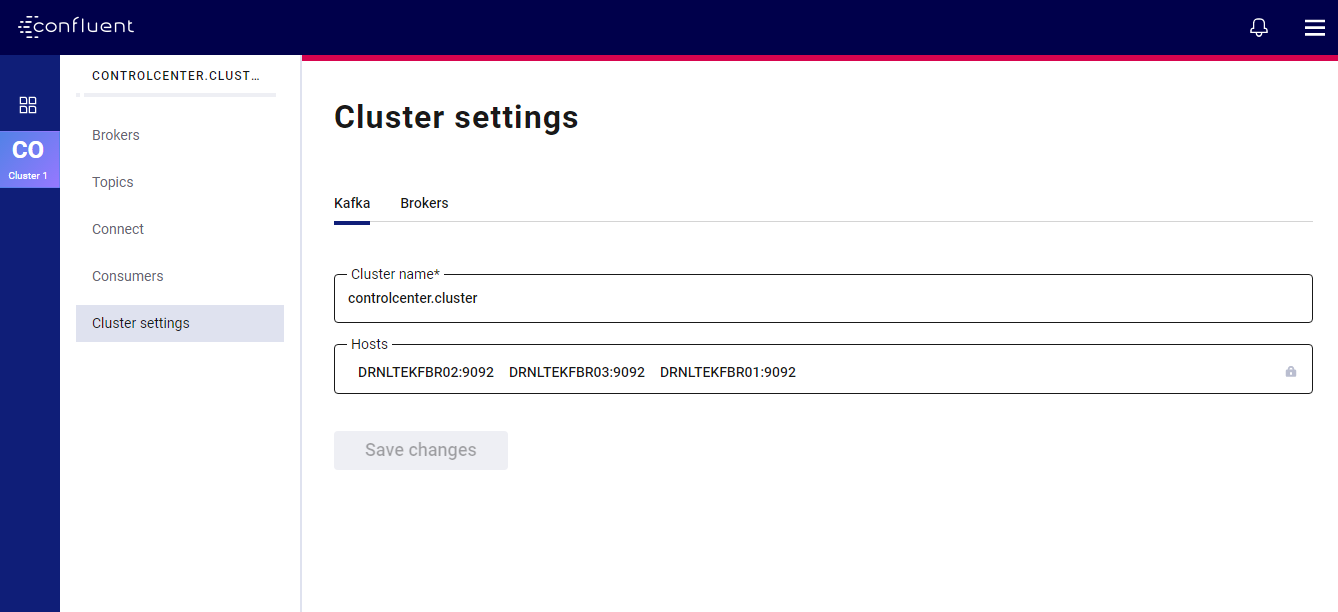
[**System Health**](https://docs.confluent.io/current/control-center/systemhealth.html#controlcenter-userguide-systemhealth)

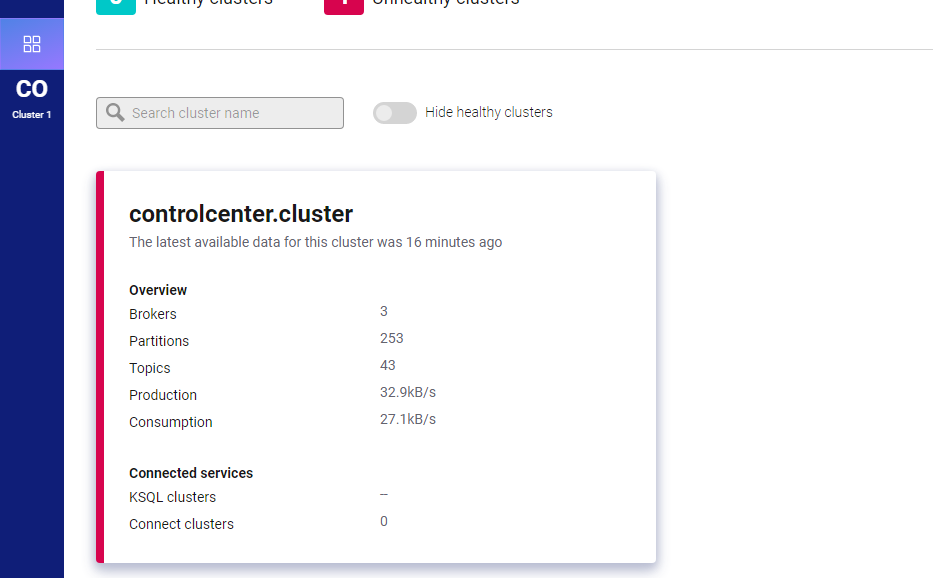
Control Center can monitor the health of your Kafka clusters. View trends for important broker and topic health metrics, and set alerts on important cluster key performance indicators (KPIs)











…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

…………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

**Downloads :**

1. You need to download the ansible playbook provided by confluent platform:

Clone the url to your work location :

<https://github.com/confluentinc/cp-ansible.git>

1. There are some packages provided by confluent team which you need to download before starting installation .
2. All the packages we have downloaded manually and kept in /**opt/pakages** location in our ansible host , ansible playbook automatically copies these packages and will do installation in respective hosts .

Please find the below list of packages :

1. ansible-2.8.5-1.el7.noarch.rpm
2. confluent-common-5.3.1-1.noarch.rpm
3. confluent-hub-client-5.3.1-1.noarch.rpm
4. confluent-control-center-fe-5.3.1-1.noarch.rpm
5. confluent-control-center-5.3.1-1.noarch.rpm
6. confluent-kafka-2.12-5.3.1-1.noarch.rpm
7. confluent-kafka-connect-jdbc-5.3.1-1.noarch.rpm
8. confluent-kafka-connect-elasticsearch-5.3.1-1.noarch.rpm
9. confluent-kafka-connect-jms-5.3.1-1.noarch.rpm
10. confluent-kafka-connect-s3-5.3.1-1.noarch.rpm
11. confluent-kafka-connect-replicator-5.3.1-1.noarch.rpm
12. confluent-kafka-connect-storage-common-5.3.1-1.noarch.rpm
13. confluent-kafka-mqtt-5.3.1-1.noarch.rpm
14. confluent-kafka-rest-5.3.1-1.noarch.rpm
15. confluent-platform-2.12-5.3.1-1.noarch.rpm
16. confluent-ksql-5.3.1-1.noarch.rpm
17. confluent-rebalancer-5.3.1-1.noarch.rpm
18. confluent-rest-utils-5.3.1-1.noarch.rpm
19. confluent-schema-registry-5.3.1-1.noarch.rpm
20. jmx\_prometheus\_javaagent-0.12.0.jar
21. confluent-common-5.3.1-1.noarch.rpm
22. confluent-control-center-fe-5.3.1-1.noarch.rpm
23. confluent-control-center-5.3.1-1.noarch.rpm
24. confluent-rebalancer-5.3.1-1.noarch.rpm
25. confluent-rest-utils-5.3.1-1.noarch.rpm
26. confluent-common-5.3.1-1.noarch.rpm
27. confluent-kafka-2.12-5.3.1-1.noarch.rpm
28. confluent-kafka-connect-elasticsearch-5.3.1-1.noarch.rpm
29. confluent-kafka-connect-jdbc-5.3.1-1.noarch.rpm
30. confluent-kafka-connect-jms-5.3.1-1.noarch.rpm
31. confluent-kafka-connect-replicator-5.3.1-1.noarch.rpm
32. confluent-kafka-connect-s3-5.3.1-1.noarch.rpm
33. confluent-kafka-connect-storage-common-5.3.1-1.noarch.rpm

.......................................................................................................................................................................................................................................................................................................