CONTENTS

// Tutorial //

How To Install Apache Kafka on CentOS 7

Published on August 15, 2018 · Updated on March 28, 2019

Apache Java Messaging CentOS

bsder

Not using CentOS 7?
Choose a different version or distribution.

CentOS 7

The author selected the Free and Open Source Fund to receive a donation as part of the Write for DOnations program.

Introduction

Apache Kafka is a popular distributed message broker designed to efficiently handle large volumes of real-time data. A Kafka cluster is not only highly scalable and fault-tolerant, but it also has a much higher throughput compared to other message brokers such as ActiveMQ and RabbitMQ. Though it is generally used as a *publish/subscribe* messaging system, a lot of organizations also use it for log aggregation because it offers persistent storage for published messages.

A publish/subscribe messaging system allows one or more producers to publish messages without considering the number of consumers or how they will process the messages. Subscribed clients are notified automatically about updates and the creation of new messages. This system is more efficient and scalable than systems where clients poll periodically to determine if new messages are available.

In this tutorial, you will install and use Apache Kafka 2.1.1 on CentOS 7.

Prerequisites

To follow along, you will need:

- One CentOS 7 server and a non-root user with sudo privileges. Follow the steps specified in this guide if you do not have a non-root user set up.
- At least 4GB of RAM on the server. Installations without this amount of RAM may cause the Kafka service to fail, with the Java virtual machine (JVM) throwing an "Out Of Memory" exception during startup.
- OpenJDK 8 installed on your server. To install this version, follow these instructions on installing specific versions of OpenJDK. Kafka is written in Java, so it requires a JVM; however, its startup shell script has a version detection bug that causes it to fail to start with JVM versions above 8.

Step 1 – Creating a User for Kafka

Since Kafka can handle requests over a network, you should create a dedicated user for it. This minimizes damage to your CentOS machine should the Kafka server be compromised. We will create a dedicated **kafka** user in this step, but you should create a different non-root user to perform other tasks on this server once you have finished setting up Kafka.

Logged in as your non-root sudo user, create a user called **kafka** with the `useradd` command:

```
$ sudo useradd kafka -m
```

Copy

The `-m` flag ensures that a home directory will be created for the user. This home directory, `/home/kafka`, will act as our workspace directory for executing commands in the sections below.

Set the password using `passwd`:

```
$ sudo passwd kafka
```

Copy

Add the **kafka** user to the `wheel` group with the `adduser` command, so that it has the privileges required to install Kafka's dependencies:

```
$ sudo usermod -aG wheel kafka
```

Copy

Your **kafka** user is now ready. Log into this account using `su`:

```
$ su -l kafka
```
Copy

Now that we've created the Kafka-specific user, we can move on to downloading and extracting the Kafka binaries.

Step 2 – Downloading and Extracting the Kafka Binaries

Let's download and extract the Kafka binaries into dedicated folders in our **kafka** user's home directory.

To start, create a directory in `/home/kafka` called `Downloads` to store your downloads:

```
$ mkdir ~/Downloads
```
Copy

Use `curl` to download the Kafka binaries:

```
$ curl "https://www.apache.org/dist/kafka/2.1.1/kafka_2.11-2.1.1.tgz" -o ~/Dow
```
Copy

Create a directory called `kafka` and change to this directory. This will be the base directory of the Kafka installation:

```
$ mkdir ~/kafka && cd ~/kafka
```
Copy

Extract the archive you downloaded using the `tar` command:

```
$ tar -xvzf ~/Downloads/kafka.tgz --strip 1
```
Copy

We specify the `--strip 1` flag to ensure that the archive's contents are extracted in `~/kafka/` itself and not in another directory (such as `~/kafka/kafka_2.11-2.1.1/`) inside of it.

Now that we've downloaded and extracted the binaries successfully, we can move on configuring to Kafka to allow for topic deletion.

Step – Configuring the Kafka Server

Kafka's default behavior will not allow us to delete a *topic*, the category, group, or feed name to which messages can be published. To modify this, let's edit the configuration file.

Kafka's configuration options are specified in `server.properties`. Open this file with `vi` or your favorite editor:

```
$ vi ~/kafka/config/server.properties
```

Copy

Let's add a setting that will allow us to delete Kafka topics. Press `i` to insert text, and add the following to the bottom of the file:

~/kafka/config/server.properties

```
delete.topic.enable = true
```

When you are finished, press `ESC` to exit insert mode and `:wq` to write the changes to the file and quit. Now that we've configured Kafka, we can move on to creating systemd unit files for running and enabling it on startup.

Step 4 – Creating Systemd Unit Files and Starting the Kafka Server

In this section, we will create systemd unit files for the Kafka service. This will help us perform common service actions such as starting, stopping, and restarting Kafka in a manner consistent with other Linux services.

Zookeeper is a service that Kafka uses to manage its cluster state and configurations. It is commonly used in many distributed systems as an integral component. If you would like to know more about it, visit the official Zookeeper docs.

Create the unit file for `zookeeper`:

```
$ sudo vi /etc/systemd/system/zookeeper.service
```

Copy

Enter the following unit definition into the file:

/etc/systemd/system/zookeeper.service

```
[Un
Requires=network.target remote-fs.target
After=network.target remote-fs.target
```

```
[Service]
Type=simple
User=kafka
ExecStart=/home/kafka/kafka/bin/zookeeper-server-start.sh /home/kafka/kafka/config/zooke
ExecStop=/home/kafka/kafka/bin/zookeeper-server-stop.sh
Restart=on-abnormal

[Install]
WantedBy=multi-user.target
```

The `[Unit]` section specifies that Zookeeper requires networking and the filesystem to be ready before it can start.

The `[Service]` section specifies that systemd should use the `zookeeper-server-start.sh` and `zookeeper-server-stop.sh` shell files for starting and stopping the service. It also specifies that Zookeeper should be restarted automatically if it exits abnormally.

Save and close the file when you are finished editing.

Next, create the systemd service file for `kafka`:

```
$ sudo vi /etc/systemd/system/kafka.service
```

Copy

Enter the following unit definition into the file:

/etc/systemd/system/kafka.service

```
[Unit]
Requires=zookeeper.service
After=zookeeper.service

[Service]
Type=simple
User=kafka
ExecStart=/bin/sh -c '/home/kafka/kafka/bin/kafka-server-start.sh /home/kafka/kafka/conf
ExecStop=/home/kafka/kafka/bin/kafka-server-stop.sh
Restart=on-abnormal

[Install]
WantedBy=multi-user.target
```

The `[Unit]` section specifies that this unit file depends on `zookeeper.service`. This will ensure that `zookeeper` gets started automatically when the `kafa` service starts.

The `[Service]` section specifies that systemd should use the `kafka-server-start.sh` and `kafka-server-stop.sh` shell files for starting and stopping the service. It also specifies that Kafka should be restarted automatically if it exits abnormally.

Save and close the file when you are finished editing.

Now that the units have been defined, start Kafka with the following command:

```
$ sudo systemctl start kafka
```

Copy

To ensure that the server has started successfully, check the journal logs for the `kafka` unit:

```
$ journalctl -u kafka
```

Copy

You should see output similar to the following:

```
Output
Jul 17 18:38:59 kafka-centos systemd[1]: Started kafka.service.
```

You now have a Kafka server listening on port `9092`.

While we have started the `kafka` service, if we were to reboot our server, it would not be started automatically. To enable `kafka` on server boot, run:

```
$ sudo systemctl enable kafka
```

Copy

Now that we've started and enabled the services, let's check the installation.

Step 5 – Testing the Installation

Let's publish and consume a **"Hello World"** message to make sure the Kafka server is behaving correctly. Publishing messages in Kafka requires:

- A *producer*, which enables the publication of records and data to topics.
- A *consumer*, which reads messages and data from topics.

First, create a topic named `TutorialTopic` by typing:

```
$ ~/kafka/bin/kafka-topics.sh --create --zookeeper localhost:2181 --replication
```
Copy

You will see the following output:

```
Output
Created topic "TutorialTopic".
```

You can create a producer from the command line using the `kafka-console-producer.sh` script. It expects the Kafka server's hostname, port, and a topic name as arguments.

Publish the string `"Hello, World"` to the `TutorialTopic` topic by typing:

```
$ echo "Hello, World" | ~/kafka/bin/kafka-console-producer.sh --broker-list loc
```
Copy

Next, you can create a Kafka consumer using the `kafka-console-consumer.sh` script. It expects the ZooKeeper server's hostname and port, along with a topic name as arguments.

The following command consumes messages from `TutorialTopic`. Note the use of the `--from-beginning` flag, which allows the consumption of messages that were published before the consumer was started:

```
$ ~/kafka/bin/kafka-console-consumer.sh --bootstrap-server localhost:9092 --top
```
Copy

If there are no configuration issues, you should see `Hello, World` in your terminal:

```
Output
Hello, World
```

The script will continue to run, waiting for more messages to be published to the topic. Feel free to open a new terminal and start a producer to publish a few more messages. You should be able to see them all in the consumer's output.

When you are done testing, press `CTRL+C` to stop the consumer script. Now that we have tested the installation, let's move on to installing KafkaT.

Step 6 – Installing KafkaT (Optional)

KafkaT is a tool from Airbnb that makes it easier for you to view details about your Kafka cluster and perform certain administrative tasks from the command line. Because it is a Ruby gem, you will need Ruby to use it. You will also need `ruby-devel` and build-related packages such as `make` and `gcc` to be able to build the other gems it depends on. Install them using `yum`:

```
$ sudo yum install ruby ruby-devel make gcc patch
```

Copy

You can now install KafkaT using the gem command:

```
$ sudo gem install kafkat
```

Copy

KafkaT uses `.kafkatcfg` as the configuration file to determine the installation and log directories of your Kafka server. It should also have an entry pointing KafkaT to your ZooKeeper instance.

Create a new file called `.kafkatcfg`:

```
$ vi ~/.kafkatcfg
```

Copy

Add the following lines to specify the required information about your Kafka server and Zookeeper instance:

~/.kafkatcfg

```
{
  "kafka_path": "~/kafka",
  "log_path": "/tmp/kafka-logs",
  "zk_path": "localhost:2181"
}
```

Save and close the file when you are finished editing.

You are now ready to use KafkaT. For a start, here's how you would use it to view details

```
Topic                Partition   Leader      Replicas        ISRs
TutorialTopic        0             0         [0]             [0]
__consumer_offsets       0                     0               [0]
...
...
```

You will see `TutorialTopic`, as well as `__consumer_offsets`, an internal topic used by Kafka for storing client-related information. You can safely ignore lines starting with `__consumer_offsets`.

To learn more about KafkaT, refer to its GitHub repository.

Step 7 – Setting Up a Multi-Node Cluster (Optional)

If you want to create a multi-broker cluster using more CentOS 7 machines, you should repeat Step 1, Step 4, and Step 5 on each of the new machines. Additionally, you should make the following changes in the `server.properties` file for each:

- The value of the `broker.id` property should be changed such that it is unique throughout the cluster. This property uniquely identifies each server in the cluster and can have any string as its value. For example, `"server1"`, `"server2"`, etc.
- The value of the `zookeeper.connect` property should be changed such that all nodes point to the same ZooKeeper instance. This property specifies the Zookeeper instance's address and follows the `<HOSTNAME/IP_ADDRESS>:<PORT>` format. For example, `"203.0.113.0:2181"`, `"203.0.113.1:2181"` etc.

If you want to have multiple ZooKeeper instances for your cluster, the value of the `zookeeper.connect` property on each node should be an identical, comma-separated string listing the IP addresses and port numbers of all the ZooKeeper instances.

Step 8 – Restricting the Kafka User

Now that all of the installations are done, you can remove the **kafka** user's admin privileges. Before you do so, log out and log back in as any other non-root sudo user. If you are still running the same shell session you started this tutorial with, simply type `exit`.

Remove the **kafka** user from the sudo group:

```
$ sudo gpasswd -d kafka wheel
```
Copy

To further improve your Kafka server's security, lock the **kafka** user's password using the `passwd` command. This makes sure that nobody can directly log into the server using this account:

```
$ sudo passwd kafka -l
```
Copy

At this point, only **root** or a `sudo` user can log in as `kafka` by typing in the following command:

```
$ sudo su - kafka
```
Copy

In the future, if you want to unlock it, use `passwd` with the `-u` option:

```
$ sudo passwd kafka -u
```
Copy

You have now successfully restricted the **kafka** user's admin privileges.

Conclusion

You now have Apache Kafka running securely on your CentOS server. You can make use of it in your projects by creating Kafka producers and consumers using Kafka clients, which are available for most programming languages. To learn more about Kafka, you can also consult its documentation.

About the authors

bsder Author

Kathleen Juell Editor

Still looking for an answer? Ask a question

Search for more help

Was this helpful? Yes No

Comments

4 Comments

Leave a comment...

This t x defaults to using `Markdown` to format your answer.

You can type `!ref` in this text area to quickly search our full set of tutorials, documentation & marketplace offerings and insert the link!

Sign In or Sign Up to Comment

javieraguasvivas • December 29, 2021

Please update endpoint for download to the latest updated one: curl “https://downloads.apache.org/kafka/3.0.0/kafka_2.13-3.0.0.tgz” -o ~/Downloads/kafka.tgz

Reply

dshipps • July 13, 2021

There are a couple of steps that are out of date.

In Step 2 the line for downloading kafka should be

curl “https://downloads.apache.org/kafka/2.8.0/kafka_2.13-2.8.0.tgz” -o ~/Downloads/kafka.tgz

In Step 4 the zookeeper service needs the ExecStart line update to the following:

ExecStart=/bin/sh -c ‘/home/kafka/kafka/bin/zookeeper-server-start.sh /home/kafka/kafka/config/zookeeper.properties’

In Step 6 I have been unable to get KafkaT working

Reply

annguyendl • February 13, 2021

value of the broker.id property must be an integer as Apache Kafka 2.7’s do entation.

Reply

iakko • September 25, 2018

This post was very helpful for me. Thanks :)

The only suggestion I have is adding a little note on paragraph 7 saying that in this case, adding a firewall rule for kafka service in each machine is needed in order to let the nodes on the cluster communicate. I spent 1 hour wondering why I was not able to make them talk to each other!

Anyway, great post, thank!

Reply

Popular Topics

Ubuntu

Linux Basics

JavaScript

Python

MySQL

Docker

Kubernetes

All tutorials →

Free Managed Hosting →

Congratulations on unlocking the whale ambience easter egg! Click the whale button in the bottom left of your screen to toggle some ambient whale noises while you read.

Thank you to the Glacier Bay National Park & Preserve and Merrick079 for the sounds behind this easter egg.

Interested in whales, protecting them, and their connection to helping prevent climate change? We recommend checking out the Whale and Dolphin Conservation.

Reset easter egg to be discovered again / Permanently dismiss and hide easter egg

Get our biweekly newsletter

Sign up for Infrastructure as a Newsletter.

Sign up →

Hollie's Hub for Good

Working on improving health and education, reducing inequality, and spurring economic growth? We'd like to help.

Learn more →

Become a contributor

You get paid; we donate to tech nonprofits.

Learn more →

Featured on Community

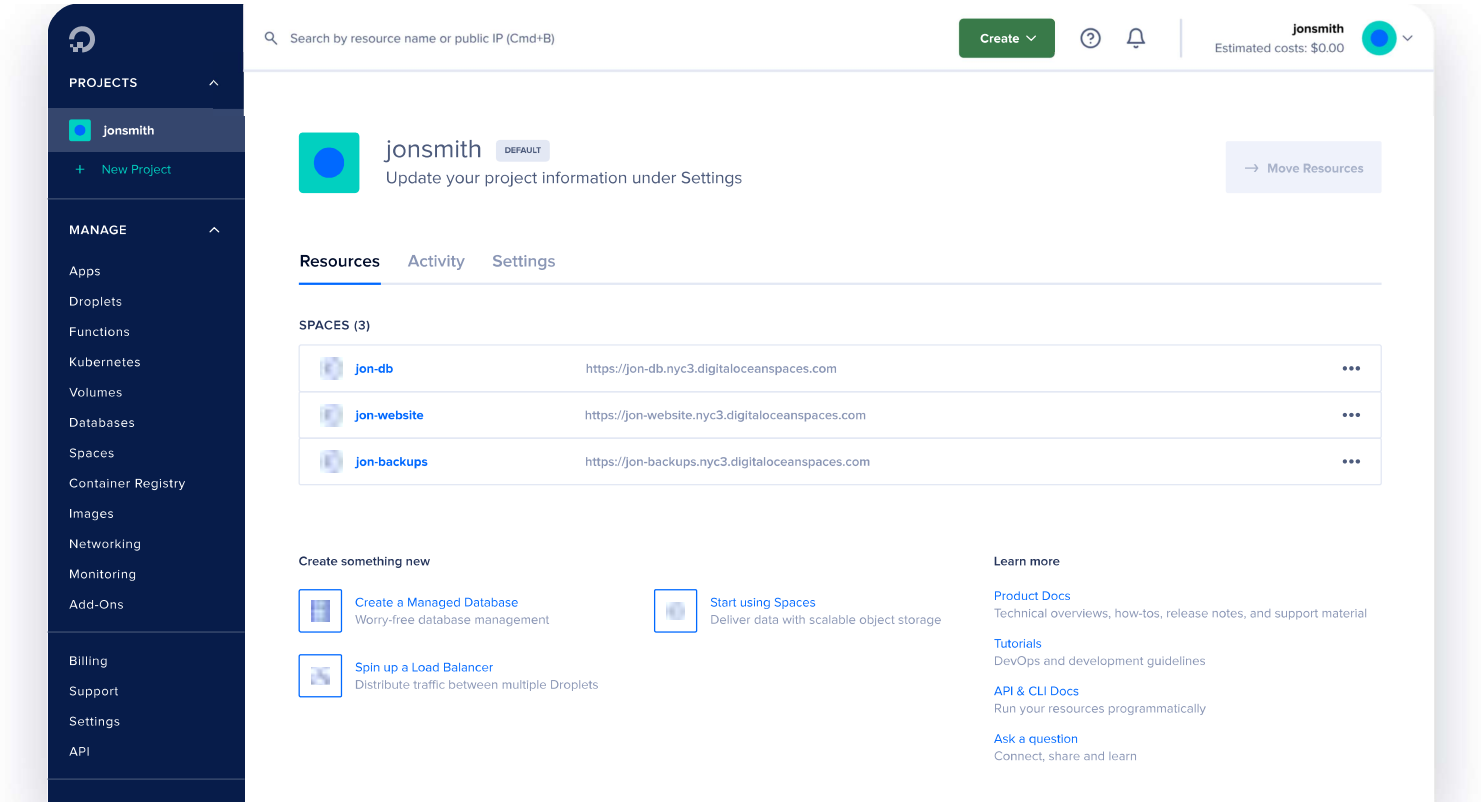
Kubernetes Course Learn Python 3 Machine Learning in Python

Getting started with Go Intro to Kubernetes

DigitalOcean Products

Cloudways Virtual Machines Managed Databases Managed Kubernetes

Block Storage Object Storage Marketplace VPC Load Balancers



Get started for free

Enter your email to get $200 in credit for your first 60 days with DigitalOcean.

Email address

Send My Promo

New accounts only. By submitting your email you agree to our Privacy Policy.

Company

Pro

Community

Solutions

Contact