**Kafka Manual**

**1. Installing Apache Kafka using Windows Subsystem Linux – Ubuntu**

**Step 1:** Install windows subsystem for linux

**Step 2:** Install latest ubuntu distribution

**Step 3:** Open ubuntu terminal & login sudo -s

**Step 4:** (Optional) creating new user, add it to sudo group & login to that user.

**Step 5:** Download jdk

**Step 6:** Make folder for kafka to download the tgz (binary file) & extract it.

* **sudo wget** [**https://downloads.apache.org/kafka/3.6.1/kafka\_2.13-3.6.1.tgz**](https://downloads.apache.org/kafka/3.6.1/kafka_2.13-3.6.1.tgz)

**Step 7:** (Optional) move the file to /opt/kafka folder:

* **mv kafka\_2.13-3.6.1 /opt/kafka**

**Step 8:**

**Option1:** Create systemd unit files & starting the Kafka server. systemd unit files help us to perform common service actions like starting, stopping, & restarting kafka/zookeeper

* Create the systemd unit file for **zookeeper**: sudo nano /etc/systemd/system/**zookeeper.service**

|  |
| --- |
| /etc/systemd/system/zookeeper.service [Unit] Description=Apache Zookeeper service Documentation=http://zookeeper.apache.org Requires=network.target remote-fs.target After=network.target remote-fs.target  [Service] Type=simple ExecStart=/opt/kafka/bin/zookeeper-server-start.sh /opt/kafka/config/zookeeper.properties ExecStop=/opt/kafka/bin/zookeeper-server-stop.sh Restart=on-abnormal  [Install] WantedBy=multi-user.target |

* **Reload the daemon to take effect:** sudo **systemctl** daemon-reload
* Create the systemd unit file for Kafka service: sudo nano /etc/systemd/system/**kafka.service**

|  |
| --- |
| [Unit]  Description=Apache Kafka Service Documentation=http://kafka.apache.org/documentation.html  Requires=zookeeper.service  [Service]  Type=simple  Environment="JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64" ExecStart=/opt/kafka/bin/kafka-server-start.sh /opt/kafka/config/server.properties ExecStop=/opt/kafka/bin/kafka-server-stop.sh  [Install]  WantedBy=multi-user.target |

* **Reload the daemon to take effect:** sudo **systemctl** daemon-reload

To start zookeeper & kafka service & check its status (if you’re using systemd files)

* **Start zookeeper first**: sudo **systemctl** start zookeeper
* **Status of zookeeper**: sudo **systemctl** status zookeeper
* **Start the kafka service**: sudo **systemctl** start kafka
* **Status of Kafka service**: sudo **systemctl** status kafka

**Option2:** To start zookeeper & kafka service & check its status (if you’re not using systemd files)

* cd /opt/kafka
* **Start zookeeper first**: bin/zookeeper-server-start.sh config/zookeeper.properties (From /opt/kafka)
* **Start the kafka service**: bin/kafka-server-start.sh config/server.properties (From /opt/kafka)

**Option3:** We can start Kafka & Zookeeper server in the background by write shell script

Shell Script: **kafkastart.sh**

|  |
| --- |
| #!/bin/bash  sudo nohup /opt/kafka/bin/zookeeper-server-start.sh -daemon /opt/kafka/config/zookeeper.properties > /dev/null 2>&1 &  sleep 5  sudo nohup /opt/kafka/bin/kafka-server-start.sh -daemon /opt/kafka/config/server.properties > /dev/null 2>&1 & |

Shell Script: **kafkastop.sh**

|  |
| --- |
| #!/bin/bash  sudo nohup /opt/kafka/bin/zookeeper-server-stop.sh -daemon /opt/kafka/config/zookeeper.properties > /dev/null 2>&1 &  sleep 5  sudo nohup /opt/kafka/bin/kafka-server-stop.sh -daemon /opt/kafka/config/server.properties > /dev/null 2>&1 & |

* Giving executable permissions to the file
* **sudo** chmod +x kafkastart.sh
* **sudo** chmod +x kafkastop.sh
* Go to folder containing kafkastart.sh, kafkastop.sh
* To run kafkastart.sh – **./kafkastart.sh**
* To run kafkastop.sh – **./kafkastop.sh**

|  |  |
| --- | --- |
| **No.** | **Command Description** |
| 1. | wsl --install (to install windows subsystem for Linux)  wsl --update (to update wsl)  wsl -l -v (to list down all the distributions) |
| 2. | sudo adduser [newUser] (to create new user)  groups [newUser] (to view groups to which newUser added to)  sudo adduser [newUser] sudo (adding newUser to the sudo group)  su -l [newUser] (logging/switching to newUser)  sudo deluser [newUser] sudo (Removing newUser from group ‘sudo’)  sudo deluser [newUser] (Remove newUser)  (nano /etc/sudoers) – Sudo rights file |
| 3. | nano [filename] (to view a file & edit) |
| 4. | sudo wget [downloadUrlLink] (to download from a particular url) |
| 5. | tar -xvzf [zipName] (unzip the zip file) |
| 6. | mv [unzipedFileName] [location] (move the file to required location) |
| 7. | netstat -ltnup  kill -9 [pid] |
| 8. | bin/zookeeper-server-start.sh config/zookeeper.properties & (& is for background)  bin/zookeeper-server-stop.sh |
|  | bin/kafka-server-start.sh config/server.properties &  bin/kafka-server-stop.sh |
| 9. | bin/kafka-topics.sh --create --topic transactions --bootstrap-server localhost:9092  bin/kafka-topics.sh --bootstrap-server localhost:9092 --create --topic demo-kafka1 --replication-factor 1 --partitions 1 |
| 10. | bin/kafka-topics.sh --bootstrap-server localhost:9092 --describe --topic [demo-kafka]  bin/kafka-topics.sh --bootstrap-server localhost:9092 --delete --topic [demo-kafka]  bin/kafka-topics.sh --list --bootstrap-server localhost:9092 |
| 11. | bin/kafka-console-producer.sh --topic transactions --bootstrap-server localhost:9092 |
| 12. | bin/kafka-console-consumer.sh --topic transactions --from-beginning --bootstrap-server localhost:9092 --group notif-consumer |
| 13. | jps (to view processes) |
|  |  |

Reference:

<https://www.digitalocean.com/community/tutorials/how-to-install-apache-kafka-on-ubuntu-20-04>

<https://www.devopshint.com/how-to-install-apache-kafka-on-ubuntu-22-04-lts/>

**2. Installing Confluent kafka community edition**

**Step 1:** Install windows subsystem for linux

**Step 2:** Install latest ubuntu distribution

**Step 3:** Open ubuntu terminal & login sudo -s

**Step 4:** (Optional) creating new user, add it to sudo group & login to that user.

**Step 5:** Download jdk

**Step 6:** Make folder for kafka to download the tgz (binary file) & extract it.

* **sudo** wget[**https://packages.confluent.io/archive/7.5/confluent-community-7.5.3.tar.gz**](https://packages.confluent.io/archive/7.5/confluent-community-7.5.3.tar.gz)
* **tar** -xvzf confluent-community-7.5.3.tar.gz

**Step 7:** (Optional) move the file to /opt/kafka folder:

* **mv** confluent-7.5.3 /opt/confluent
* mv confluent-7.5.3 ../
* mv confluent-7.5.3 confluent

**Step 8:** Download confluent CLI

* curl -sL --http1.1 https://cnfl.io/cli | sh -s -- latest (from /confluent/bin)

**Step 9:** Add confluent home & add path to run confluent command from anywhere.

* export CONFLUENT\_HOME=/opt/confluent
* export PATH=$PATH:$CONFLUENT\_HOME/bin
* echo $PATH
* echo $CONFLUENT\_HOME

#echo "export PATH=$PATH:/opt/confluent/bin" >> /etc/profile (Not required)

**Step 10:** Download confluent hub client in /confluent/bin

* wget <https://client.hub.confluent.io/confluent-hub-client-latest.tar.gz>
* tar -xvzf confluent-hub-client-latest.tar.gz bin (from /confluent folder)

export PATH=$PATH:/opt/kafka

**Step 11**: Check confluent command (confluent local services start/stop)

|  |  |
| --- | --- |
| No. | Command description |
| 1. | **confluent** local services start  (For starting all confluent services like zookeeper, kafka, schema registry, kafka REST, Connect, KsqlDB server) |
| 2. | confluent local services zookeeper start (For starting zookeeper service) |
| 3. | confluent local services kafka start (For starting kafka service) |
| 4. | confluent local services schema-registry start (For starting schema registry) |
| 5. | confluent local services kafka-rest start (For starting kafka rest) |
| 6. | confluent local services connect start (For starting connect) |
| 7. | confluent local services kql-server start (For starting KSQL server) |
| 8. | confluent local services status (For checking the status of services) |
| 9. | kafka-topics --create --bootstrap-server localhost:9092 --replication-factor 1 --partitions 1 --topic [topicName] |
| 10. | kafka-topics --list --bootstrap-server localhost:9092 |
| 11. | confluent local services connect connector list (Lists the available connectors) |

<https://docs.confluent.io/platform/current/installation/installing_cp/zip-tar.html#prod-kafka-cli-install>

**3. Kafka Configuration**

|  |  |
| --- | --- |
| **Configuration** | **Description** |
| broker.id | Id of broker & must be set to a unique integer for each broker. |
| Listeners | **a)** Comma-separated list of URIs we will listen on locally & the listener names. If the listener’s name is not security protocol, listener security protocol map must also be set. i.e., listener.security.protocol.map  **b)** Format: listeners = listener\_name://host\_name:port  **c)** Listener names & port numbers must be unique unless one listener is an IPV4 address & the other listener is an IPV6 address (for the same port)  **d)** Specify hostname as 0.0.0.0 to bind all interfaces & leave hostname empty to bind to default interface.  **e)** E.g. of legal listeners lists:  PLAINTEXT://myhost:9092, SSL://:9091, CLIENT://0.0.0.0:9092, REPLICATION://localhost:9093, PLAINTEXT://127.0.0.1:9092, SSL://[::1]:9092 |
| advertised.listeners | **a)** Listeners to publish to Zookeeper for clients to use, if different than the listeners config property. In IaaS environments, this may need to be different from the interface to which the broker binds.  **b)** If this is not set, the value for listeners will be used. Unlike listeners, it is not valid to advertise the 0.0.0.0 meta – address & there can be duplicated ports in this property, so that one listener can be configured to advertise another listener’s address.  **c)** This can be useful in some cases where external load balancers are used. |
| listener.security.protocol.map | Map between listener names & security protocols. This must be defined for the same security protocol to be usable in more than one port or IP.  For e.g., internal & external traffic can be separated even if SSL is required for both. Concretely, the user could define listeners with names INTERNAL & EXTERNAL and this property as INTERNAL:SSL, EXTERNAL:SSL |
| log.retention.bytes |  |
| log.retention.hours | The minimum age of a log file to be eligible for deletion due to age. |
| cleanup.policy | a) This config designates the retention policy to use on log segments.  b) The “delete” policy will discard old segments when their retention time or size limit has been reached.  c) The “compact” policy will enable log compaction, which retains the latest value for each key.  d) It is also possible to specify both policies in a comma – separated list (e.g., “delete”, “compact”). In this case, old segments will be discarded per the retention time & size configuration, while retained segments will be compacted. |

5. How to get started with Kafka/Confluent kafka in system (Refer User Manual)

(Here we’re using Linux Ubuntu wsl)

Step 1: Download Java & then Kafka from official store. Kafka 3.x.x compatible with Java 11+

Step 2: Extract & put it in /opt/kafka folder

Step 3:

**Option 1:** We can start zookeeper & then kafka directly from /opt/kafka using below commands

|  |
| --- |
| bin/zookeeper-server-start.sh config/zookeeper.properties (To start zookeeper) |
| bin/kafka-server-start.sh config/server.properties (To start kafka) |

**Option 2:** We can start Kafka & Zookeeper server in the background by writing shell script

Shell Script: **kafkastart.sh**

|  |
| --- |
| #!/bin/bash  sudo nohup /opt/kafka/bin/zookeeper-server-start.sh -daemon /opt/kafka/config/zookeeper.properties > /dev/null 2>&1 &  sleep 5  sudo nohup /opt/kafka/bin/kafka-server-start.sh -daemon /opt/kafka/config/server.properties > /dev/null 2>&1 & |

Shell Script: **kafkastop.sh**

|  |
| --- |
| #!/bin/bash  sudo nohup /opt/kafka/bin/zookeeper-server-stop.sh -daemon /opt/kafka/config/zookeeper.properties > /dev/null 2>&1 &  sleep 5  sudo nohup /opt/kafka/bin/kafka-server-stop.sh -daemon /opt/kafka/config/server.properties > /dev/null 2>&1 & |

* Giving executable permissions to the file
* **sudo** chmod +x kafkastart.sh
* **sudo** chmod +x kafkastop.sh
* Go to folder containing kafkastart.sh, kafkastop.sh
* To run kafkastart.sh – **./kafkastart.sh**
* To run kafkastop.sh – **./kafkastop.sh**

**Option 3:** Create the systemd unit file for **zookeeper**: sudo nano /etc/systemd/system/**zookeeper.service**

|  |
| --- |
| /etc/systemd/system/zookeeper.service [Unit] Description=Apache Zookeeper service Documentation=http://zookeeper.apache.org Requires=network.target remote-fs.target After=network.target remote-fs.target  [Service] Type=simple ExecStart=/opt/kafka/bin/zookeeper-server-start.sh /opt/kafka/config/zookeeper.properties ExecStop=/opt/kafka/bin/zookeeper-server-stop.sh Restart=on-abnormal  [Install] WantedBy=multi-user.target |

* **Reload the daemon to take effect:** sudo **systemctl** daemon-reload
* Create the systemd unit file for Kafka service: sudo nano /etc/systemd/system/**kafka.service**

|  |
| --- |
| [Unit]  Description=Apache Kafka Service Documentation=http://kafka.apache.org/documentation.html  Requires=zookeeper.service  [Service]  Type=simple  Environment="JAVA\_HOME=/usr/lib/jvm/java-11-openjdk-amd64" ExecStart=/opt/kafka/bin/kafka-server-start.sh /opt/kafka/config/server.properties ExecStop=/opt/kafka/bin/kafka-server-stop.sh  [Install]  WantedBy=multi-user.target |

* **Reload the daemon to take effect:** sudo **systemctl** daemon-reload

To start zookeeper & kafka service & check its status (if you’re using systemd files)

* **Start zookeeper first**: sudo **systemctl** start zookeeper
* **Status of zookeeper**: sudo **systemctl** status zookeeper
* **Start the kafka service**: sudo **systemctl** start kafka
* **Status of Kafka service**: sudo **systemctl** status kafka