# Spark setup

**The best materials:**

* [www.cloudduggu.com](https://www.cloudduggu.com/spark/installation-multi-node/) - spark multi node set up with hadoop.
* [www.confessionsofadataguy.com](https://www.confessionsofadataguy.com/create-your-very-own-apache-spark-hadoop-cluster-then-do-something-with-it/) - Setup of both Hadoop and Spark. there is no info about on which nodes (servers) execute which commands in terminal for setting up hadoop. But the part for Spark seems fine.

**Other materials:**

* [youtube](https://www.youtube.com/watch?v=f_XsaYcETnI) – spark multi node set up on hadoop
* [youtube](https://www.youtube.com/watch?v=-5TSKMXAygc) – spark multi node set up. I am not sure if it is using hadoop.
* [data-flair.training/blogs](https://data-flair.training/blogs/install-apache-spark-multi-node-cluster/) - spark multi node set up on hadoop

# Repositories

Here are my repositories related to Spark:

* [github - hadoop\_spark](https://github.com/bulka4/hadoop_spark) – Running a multinode HDFS, Yarn and Spark cluster on Azure Linux VMs.

# Modes of running Spark

## Local mode

When we run Spark in a local mode, that means it is running on a single machine. Then we have only one JVM (Java virtual machine) process.

## Cluster mode

When we run Spark in a cluster mode, that means it is running on multiple machines in a distributed way. We have then multiple JVM porcesses (master and worker) running on different machines.

# Spark theory

* [youtube - Data Engineering](https://www.youtube.com/watch?v=Tyg1FVNq40g&list=PLGhXxbu7qYooyn_aWk1DqpIF1CjBzaSUn&index=3) – Hadoop and Spark (9h video)
* [youtube - Data Engineering](https://www.youtube.com/watch?v=OgS0noWVPJ4&list=PLLa_h7BriLH0FzTY5aBFpH-vciOiEf4Br&index=4) - Spark

# Data partitioning

When Spark is saving a big table in an object storage, it automatically splits it into multiple smaller files (objects).

Later when Spark reads that table it knows which files to read to get data for the entire table.

# Submitting Spark jobs

## Materials

* [youtube - codeWithYu](https://www.youtube.com/watch?v=o_pne3aLW2w) – submit spark job through Airflow.
* [stackoverflow](https://stackoverflow.com/questions/53344285/is-there-a-way-to-submit-spark-job-on-different-server-running-master) - submit spark job through Airflow.

## Different ways of submitting Spark jobs on Kubernetes

### Spark-submit command

To run a Spark script using this option we need to follow those steps:

* Build a Docker image containing our Spark app and script
* Push our Docker image to a could container registry
* Use the spark-submit command in terminal to run that image on the Kubernetes cluster.

Instead of pushing Docker image to a container registry we can use an image saved on each Kubernetes’ node.

### SparkSession

In the SparkSession, in the Spark script, we can define parameters regarding Kubernetes cluster on which we want to run a Spark code.

It is similar to the spark-submit command. We provide the same parameters in the SparkSession as in the spark-submit command.

### SparkApplication YAML manifest

This is the best option to run Spark scripts in production.

Here are the steps we need to follow:

* Prepare a Spark script
* Build a Docker image containing our Spark script (we can use the official Spark image as the base image)
* Push that Docker image to the container registry
* Create a SparkApplication YAML manifest
* Deploy that manifest (run the Spark script) using kubectl or Airflow

In the SparkApplication YAML manifest we are specifying which Docker image we want to use and which Spark script we want to run from that image.

**Prerequisites:**

In order to use the SparkApplication YAML manifest we need to:

* Install Spark Operator using Helm
* Create a Service Account with proper permissions.

That Service Account will be used by the Spark Driver Pod to create Executors Pods.

Spark Operator will have automatically assigned a Service Account with proper permissions when using Helm.

**How this works in more detail:**

1. We prepare a SparkApplication YAML manifest
2. We deploy it, for example using the ‘kubectl apply’ command or using Airflow operator. That creates a SparkApplication resource.
3. Spark Operator notices that we created a SparkApplication resource and starts processing it.
4. Spark Operator creates a Spark Driver Pod.
5. Spark Driver Pod creates Spark Executor Pods.
6. Spark Pods executes Spark scripts.
7. Once the scripts are finished, Spark Pods are terminated.

# Code development

In order to develop Spark code we can run Spark in a Local mode (on a single machine) and Jupyter Notebook (or JupyterLab or JupyterHub) in a Docker container running on a Linux VM in cloud.

Then we can connect to the Jupyter through a browser from our local computer to develop and run a Spark code.

Additionally we can use VS code with the following extensions:

* Remote-SSH - In order to connect VS code to that VM
* Dev Containers – In order to access a filesystem inside of the Docker container running on the VM.

This will give us a possibility to browse a filesystem inside of the Docker container and use terminal through VS code.

It is used in the spark\_kubernetes repository.