# Introduction

This document outlines how Kubernetes can be leveraged in the data analytics domain. It includes real-world examples, benefits, and the reasons why Kubernetes is a strong fit for modern, distributed data processing workflows.

# Use cases

Generally Kubernetes is useful when we want to run multiple applications distributed across multiple servers. Using Kubernetes we can set up and manage environments (containers) in which each application is running.

For example we can run on one Kubernetes cluster the following services:

* Airflow – Workflow orchestration
* Data ingestion processes – Python scripts
* Spark - Distributed data transformation
* MiniIO - Distributed object storage
* Mlflow – Managing ML models lifecycle
* Custom automation scripts – Interacting with APIs of tools like PowerBI, Databricks or Azure.

Each component of those services can run in an isolated containerized environment, for example:

* **Airflow**
  + Webserver
  + Scheduler
  + Workers
* **Spark**
  + Driver
  + Executors
* **Each data ingestion process**

# Benefits

Benefits which Kubernetes offers are:

* **Isolation**
  + Each application is running in a separate container with dedicated dependecies.
  + Application dependencies do not conflict.
  + Application’s failures don’t affect other applications.
* **High Availability and Fault Tolerance**
  + We can create multiple replicas of an application.
  + When application crashes then another replica takes over.
  + Zero-downtime deployments of apps’ updates. Instead of following this practice:
    - Stop the application
    - Start the updated application

We follow this:

* Prepare new application
* Once new application is ready then replace the old application with the new one (immediately)
* **Networking**
  + Kubernetes can assign a static IP address and DNS names to the application (a container running an app)
  + Even after restart of a container, it will still have the same IP address and DNS name.
* **Declarative applications deployments using YAML files**
  + Configuration files define how application will be deployed, they allow for example:
    - Specifying how many resources that application can use (CPU, RAM)
    - Define environmental variables
    - Executing bash scripts before application starts
    - What volumes to mount
  + This enhances reproducibility, automation and supports version control with Git.