**Creating CI/CD pipelines using YAML script involves the following steps:**

**Define the pipeline:** First, you need to define the pipeline by specifying the stages, jobs, and tasks required to build, test, and deploy your application. You can do this by creating a **YAML file** and adding the necessary steps.

**Define triggers:** You can define **triggers** to automatically kick off the pipeline when changes are made to the code **repository** or when specific events occur, such as a **pull request** being created or a new release being tagged.

**Define environment variables:** You can define environment variables that your pipeline can use to store sensitive data such as **API keys**, passwords, and other secrets.

**Define artifacts:** You can define artifacts that are created during the build process and can be used in subsequent stages, such as the **deployment stage**.

**Define deployment stages:** You can define deployment stages that deploy your application to the desired environment, such as **staging** or **production**.

**Define notifications:** You can define notifications that are sent to relevant parties when the **pipeline** fails or succeeds.

Here is an example **YAML script** that demonstrates the basic structure of a CI/CD pipeline:

Graphical user interface, text, application

Description automatically generated

In this example, the **pipeline** is **triggered** whenever changes are made to the **master branch**. The **pipeline** is executed on a virtual machine running the latest version of Ubuntu. The pipeline then executes two scripts that print a message to the console.

You can add more steps to this pipeline, such as **compiling the code**, **running tests**, **packaging the application**, and **deploying it to a production environment**. By defining a CI/CD pipeline using YAML script, you can automate the entire software development process and ensure that your application is always built, tested, and deployed in a consistent and repeatable manner.