Jenkins' Commit Pipeline

Internal reference: topics/03-3.md

Introduction

The most basic continuous integration process is called a commit pipeline. This classic phase, as its name indicates, starts with commit (or push in Git) to the main repository and results in a report about the build success or failure. Since it runs after each change in the code, the build should take no more than 5 minutes and should consume a reasonable amount of resources. The commit phase is always the starting point of the continuous delivery process and provides the most important feedback cycle in the development process – constant information if the code is in a healthy state.

The commit phase works as follows: a developer checks in the code to the repository, the continuous integration server detects the change, and the build starts. The most fundamental commit pipeline contains three stages:

- Checkout: This stage downloads the source code from the repository.
- Compile: This stage compiles the source code.
- Unit test: This stage runs a suite of unit tests.

Jenkins file

So far, we've created all the pipeline code directly in Jenkins. This is, however, not the only option. We can also put the pipeline definition inside a file called `Jenkins file` and commit it to the repository, together with the source code. This method is even more consistent because the way your pipeline looks is strictly related to the project itself.

For example, if you don't need the code compilation because your programming language is interpreted (and not compiled), you won't have the Compile stage. The tools you use also differ, depending on the environment. We are going to use `Node.js`. However, in the case of a project written in Python, you can use PyBuilder. This leads to the idea that the pipelines should be created by the same people who write the code – the developers. Also, the pipeline definition should be put together with the code, in the repository.

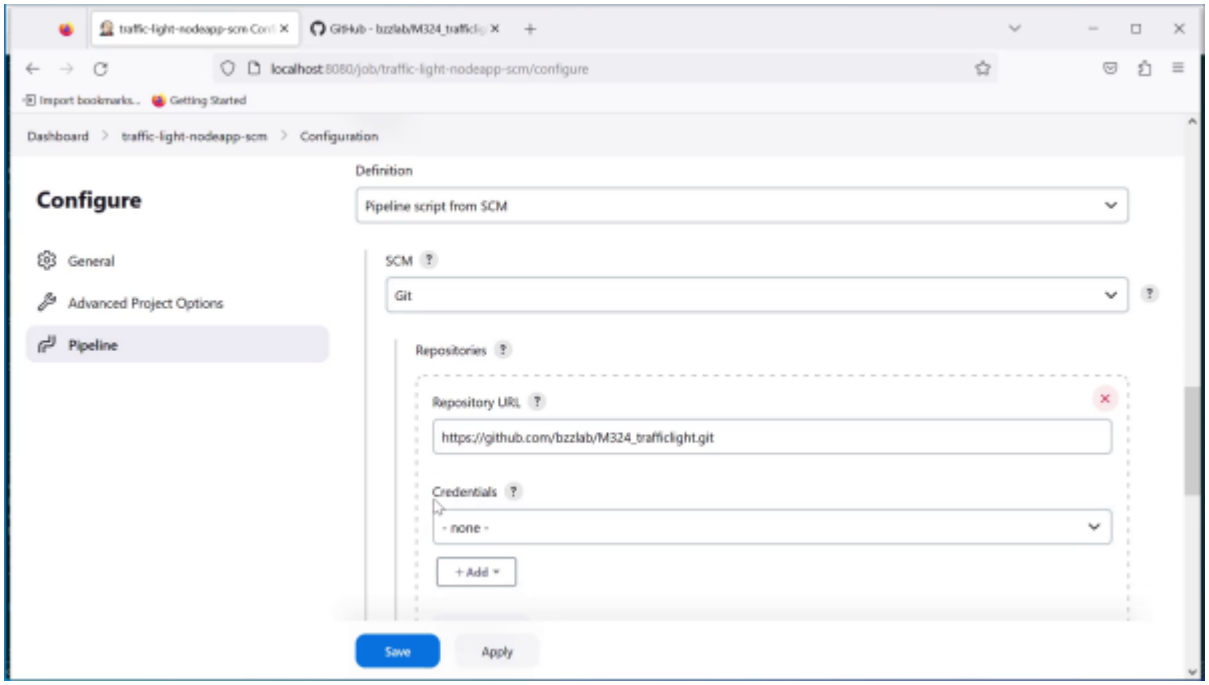
This approach brings immediate benefits, as follows:

- In the case of a Jenkins failure, the pipeline definition is not lost (because it's stored in the code repository, not in Jenkins).
- The history of the pipeline changes is stored.
- Pipeline changes go through the standard code development process (for example, they are subjected to code reviews).
- Access to the pipeline changes is restricted in exactly the same way as access to the source code.

Let's see how it all looks in practice by creating a Jenkins file.

Creating the Jenkins file

We can create the Jenkins file and push it into our Repository. Its content is almost the same as the commit pipeline we wrote. The only difference is that the checkout stage becomes redundant because Jenkins has to first check out the code (together with Jenkins file) and then read the pipeline structure (from Jenkins file). This is why Jenkins needs to know the repository address before it reads Jenkins file.



Based on the book: „Continuous Delivery with Docker and Jenkins, 3rd Edition - Third Edition By Leszko“
