Introduction to Jenkins pipeline

Internal reference: topics/03-1.md

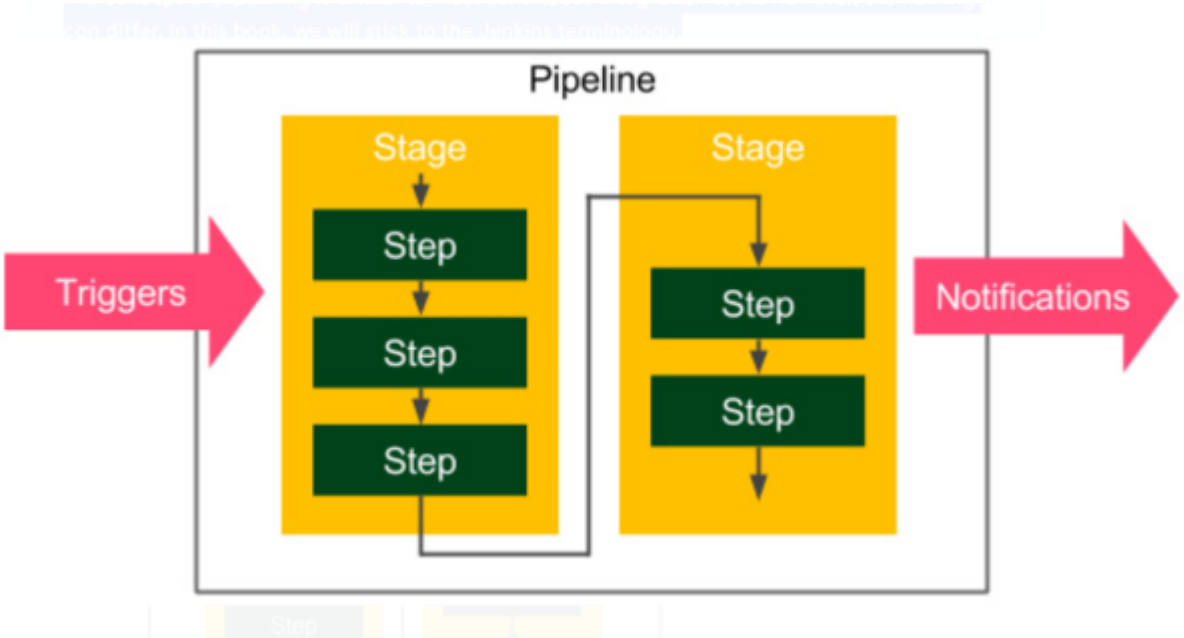
Introduction

A pipeline is a sequence of automated operations that usually represents a part of the software delivery and quality assurance process. It can be seen as a chain of scripts that provide the following additional benefits:

- `Operation grouping`: Operations are grouped together into stages (also known as gates or quality gates) that introduce a structure into a process and clearly define a rule – if one stage fails, no further stages are executed.
- `Visibility`: All aspects of a process are visualized, which helps in quick failure analysis and promotes team collaboration.
- `Feedback`: Team members learn about problems as soon as they occur so that they can react quickly

The pipeline structure

A Jenkins pipeline consists of two kinds of elements – a stage and a step. The following diagram shows how they are used:



The following are the basic pipeline elements:

Step: A single operation that tells Jenkins what to do – for example, check out code from the repository and execute a script Stage: A logical separation of steps that groups conceptually distinct sequences of steps – for example, build, test, and deploy, used to visualize the Jenkins pipeline progressInformationTechnically, it's possible to create parallel steps; however, it's better to treat

them as an exception that is only used for optimization purposes.

A multi-stage Hello World

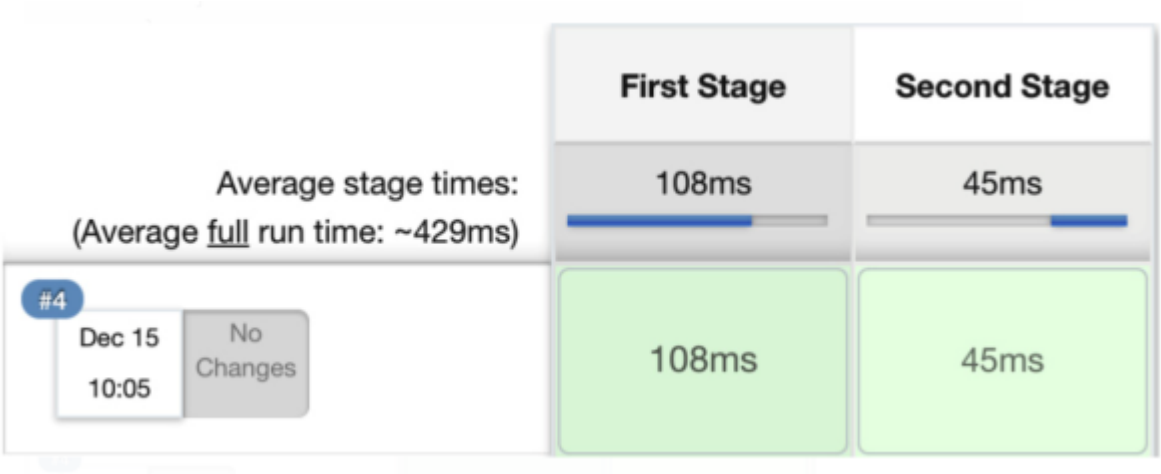
As an example, let's extend the Hello World pipeline to contain two stages:

<code> pipeline {

```
   agent any
   stages {
        stage('First Stage') {
             steps {
                  echo 'Step   -Hello World'
             }
        }
        stage('Second Stage') {
             steps {
                  echo 'Step   -Second time Hello'
                  echo 'Step   -Third time Hello'
             }
        }
   }
```

}

<code> The pipeline has no special requirements in terms of environment, and it executes three steps inside two stages. When we click on Build Now, we should see a visual representation:



The pipeline succeeded, and we can see the step execution details by clicking on the console. If any of the steps failed, processing would stop, and no further steps would run. Actually, the sole reason for a pipeline is to prevent all further steps from execution and visualize the point of failure.

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