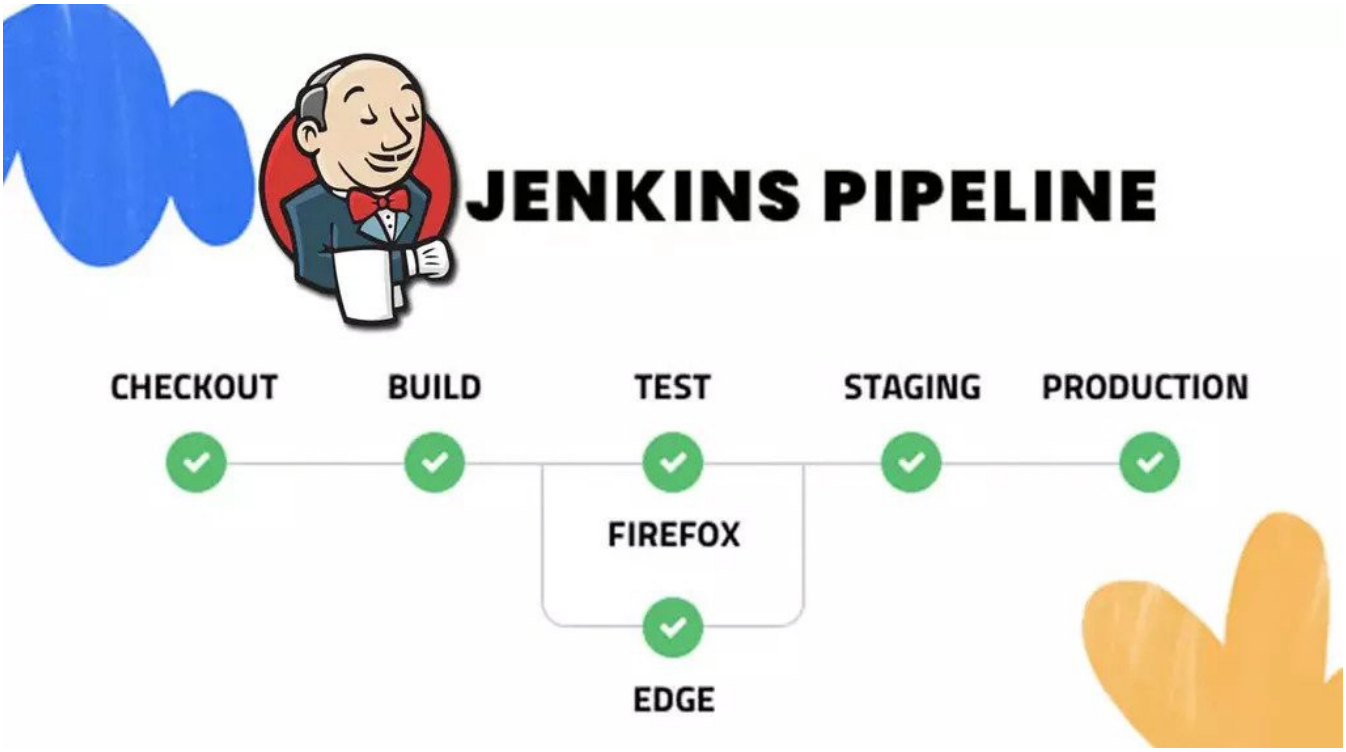
Jenkins Tutorial — Part 1 — Pipelines

Saeid Bostandoust
Jul 18 · 3 min read



Jenkins is one of the most popular CI/CD platforms used to automate CI/CD jobs. Today I intend to talk about Jenkins pipelines. So let's get started.

First of all, if you need to install Jenkins stack on Kubernetes follow jenkins-stack-kubernetes on my GitHub profile.

What is Pipeline?

A Pipeline is a definition of a job that has some stages executed respectively. Each stage has a steps section that includes commands which should be executed respectively to complete that stage. With the pipeline, we define how Jenkins should complete the CI/CD process for our applications. So to speak, the pipeline runs all processes that were run formerly by human hands.

Scripted vs. Declarative Pipelines:

Jenkins supports two types of pipeline definition.
1- Scripted pipelines which are written in Groovy language.
2- Declarative pipelines which are written in Jenkins DSL language.
We intend to talk about the Jenkins DSL language which is more readable and easy to learn. Furthermore, within this DSL language, we can write Groovy scripts too.

Jenkins Pipeline Syntax Introduction:

Here is a basic pipeline example. Let's describe it.

```
pipeline {
    agent any
    stages {
        stage("Build") {
            steps {
                echo "Build stage."
            }
        }
        stage("Test") {
            steps {
                echo "Test stage."
            }
        }
        stage("Release") {
            steps {
                echo "Release stage."
            }
        }
    }
}
```

intro1.pipeline hosted with ♡ by GitHub view raw

Each pipeline is started with "pipeline" and each of them needs two mandatory sections "agent" and "stages" which define where to run pipeline stages and what should be run to complete the pipeline. Each stage should define with a unique name and the "steps" sections of them may contain one or more commands which be run within that stage.

So, let's describe pipeline directives more deeply…

Pipeline directive: **agent**

This directive specifies where to run the entire pipeline or a specific stage.

`agent any` execute the pipeline or a specific stage on any available agent.

`agent none` is used to disable global agent for the entire pipeline. Instead, we should define the agent in each stage. Here is an example:

```
pipeline {
    agent none
    stages {
        stage("Build") {
            agent any
            steps {
                echo "Build stage."
            }
        }
    }
}
```

intro2.pipeline hosted with ♡ by GitHub view raw

`agent label` is used to execute the whole pipeline or a specific stage on a node with a specific label. Here is an example:

```
pipeline {
    agent {
        label "linux"
    }
    stages {
        stage("Build") {
            steps {
                echo "Build stage."
            }
        }
    }
}
```

intro3.pipeline hosted with ♡ by GitHub view raw

`agent docker` is used to execute the entire pipeline or a specific stage inside a docker container. Here is an example:

```
pipeline {
    agent {
```

```
        docker "alpine"
    }
    stages {
        stage("Build") {
            steps {
                sh "hostname"
            }
        }
    }
}
```

intro4.pipeline hosted with ♡ by GitHub view raw

All stages of the above pipeline are executed within an instance of an alpine container.

`agent dockerfile` this directive is a special Docker related directive that is used to build a docker image from an existing Dockerfile within the source of the project and run the entire pipeline or a specific stage inside an instance of this built image. To use this special directive, you should connect Jenkins to your source code repository to able Jenkins to fetch the project source in conjunction with Dockerfile. Here is a simple example:

```
pipeline {
    agent {
        dockerfile true
    }
    stages {
        stage("Build") {
            steps {
                sh "hostname"
            }
        }
    }
}
```

intro5.pipeline hosted with ♡ by GitHub view raw

Other agents exist but at this moment, these described agents should be enough. Another agent like Kubernetes will described later.

Pipeline directive: **stages**

`stages` section contains one or more `stage` directives. Each stage should be defined with a unique name. Inside each stage, we should define `steps` section that contains

one or more steps to be executed.

Pipeline directive: **steps**

The `steps` section contains one or more steps that should be executed as a job of this stage. All tasks that should be run to automate the CI/CD process are defined in this section.

In the next article, I will describe essential commands that can be run in `steps` section. Follow my profile.

Cicd Cicd Pipeline Jenkins Jenkins Pipeline Kubernetes