Continuous Integration and Continuous Delivery with Docker

Lev Epshtein

Continuous Delivery with Jenkins Pipeline

Lev Epshtein

Continuous Integration (CI) and Continuous Delivery (CD)

Set of operating principles, and collection of practices that enable application development teams to deliver code changes more frequently and reliably. The implementation is also known as the ***CI/CD* pipeline**.

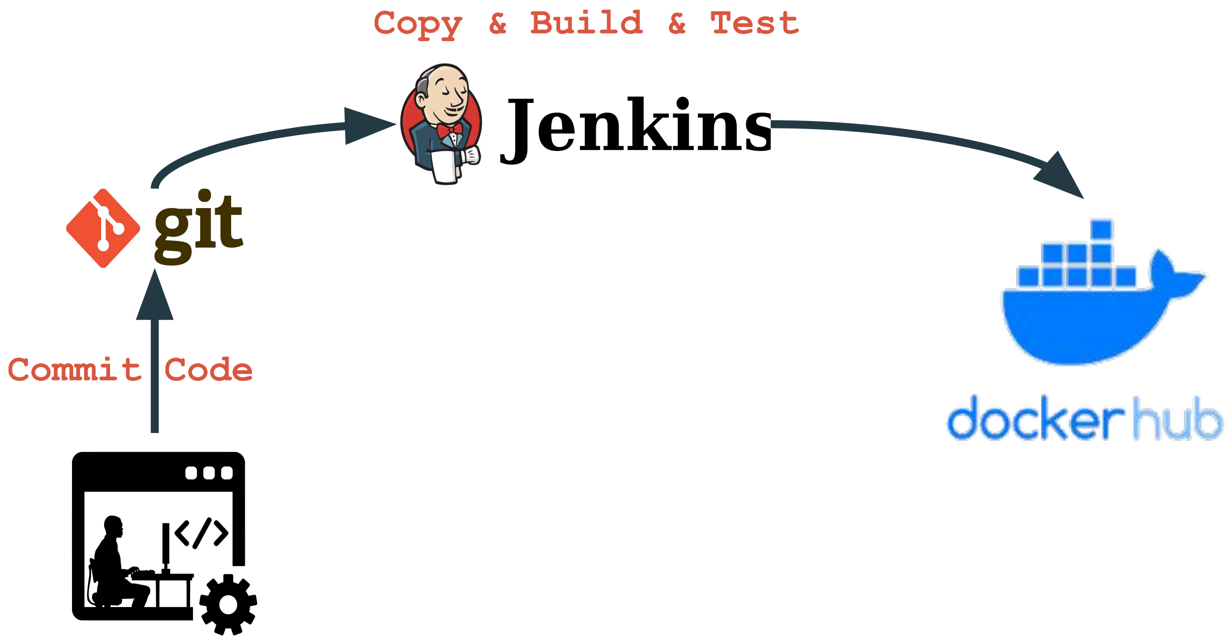
Continuous Integration (CI)

Continuous integration is a **DevOps software development practice where developers regularly merge their code changes into a central repository**, after which automated builds and tests are run.

Continuous Integration (CI)

Continuous integration is a **DevOps software development practice where developers regularly merge their code changes into a central repository**, after which automated builds and tests are run.

Pushing Docker image to docker hub **pipeline**



What is Jenkins

Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

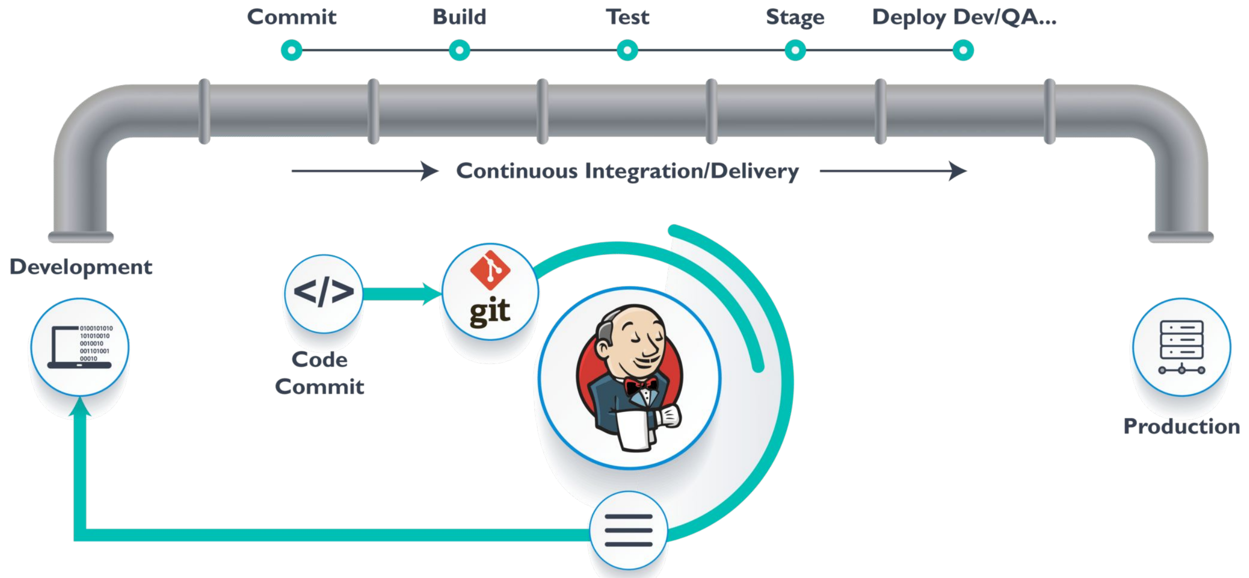
Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) installed

Environment

```
cd Jenkins\Environment

docker-compose up -d
```

Jenkins pipelines

Jenkins pipelines are based on groovy programming languages



Scripted

```
node {
    // groovy script
}
```

Declarative

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {

            }
        }
    }
}
```

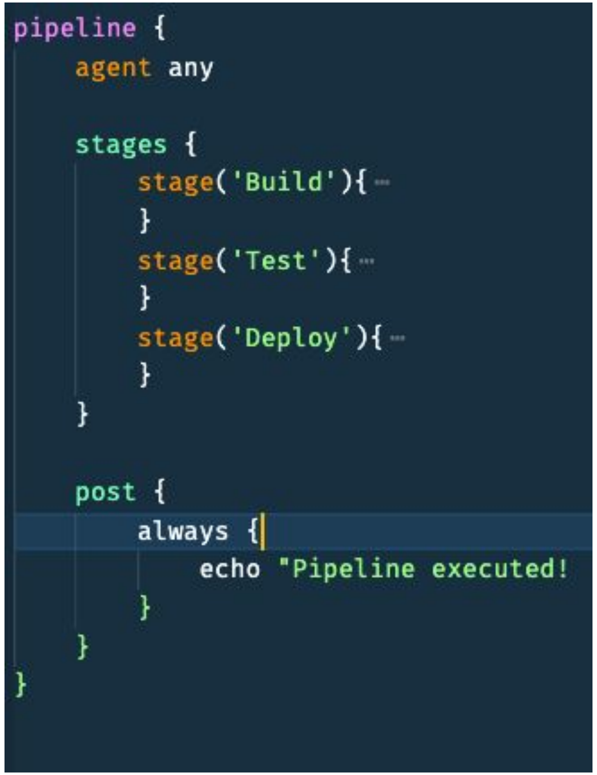
Declarative Pipeline

Declarative Pipeline

- Simplified and opinionated syntax on top of the pipeline sub-systems
- Follows the groovy syntax
- The top-level of the pipeline must be a pipeline { .. } block
- Inside of pipeline the blocks consist of sections, directives, steps, or assignment statements.

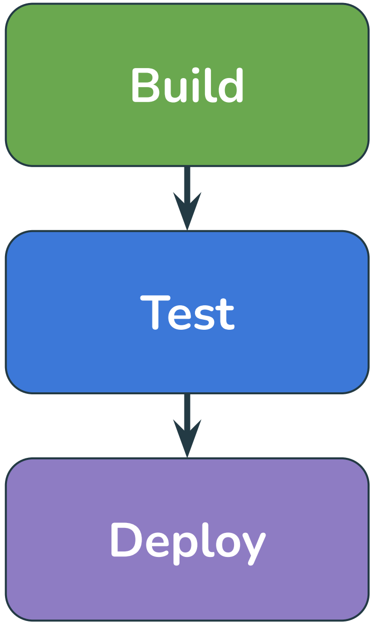
Sections

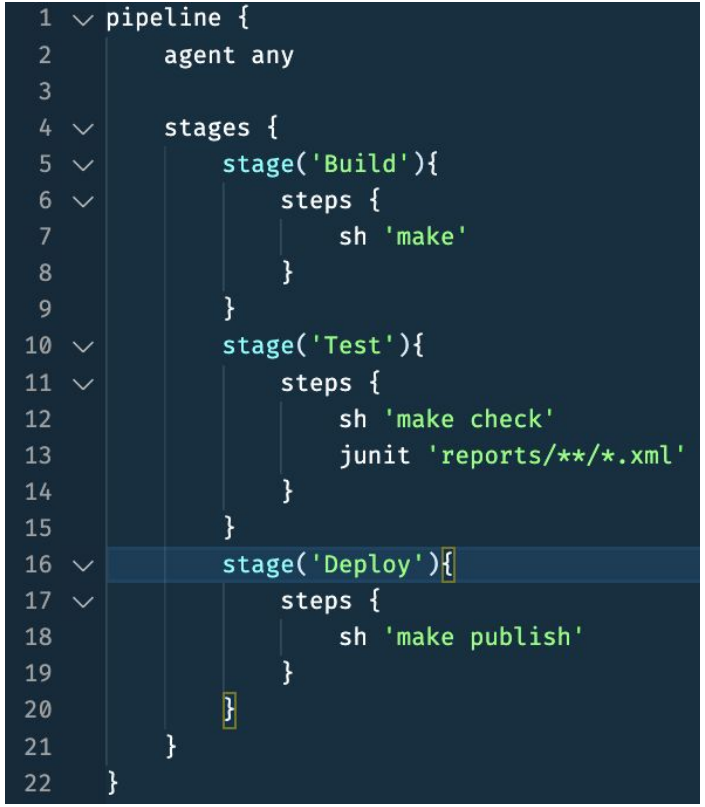
- Usually contains directive or steps
- Common sections:
  - **Agent** - defines on which agent the pipeline or the stage will be executed
  - **Stage** - define the stages of the pipeline
  - **Post** - define the post build steps



```
pipeline {
    agent any

    stages {
        stage('Build'){⋯
        }
        stage('Test'){⋯
        }
        stage('Deploy'){⋯
        }
    }

    post {
        always {
            echo "Pipeline executed!
        }
    }
}
```

Pipeline as Code - Example

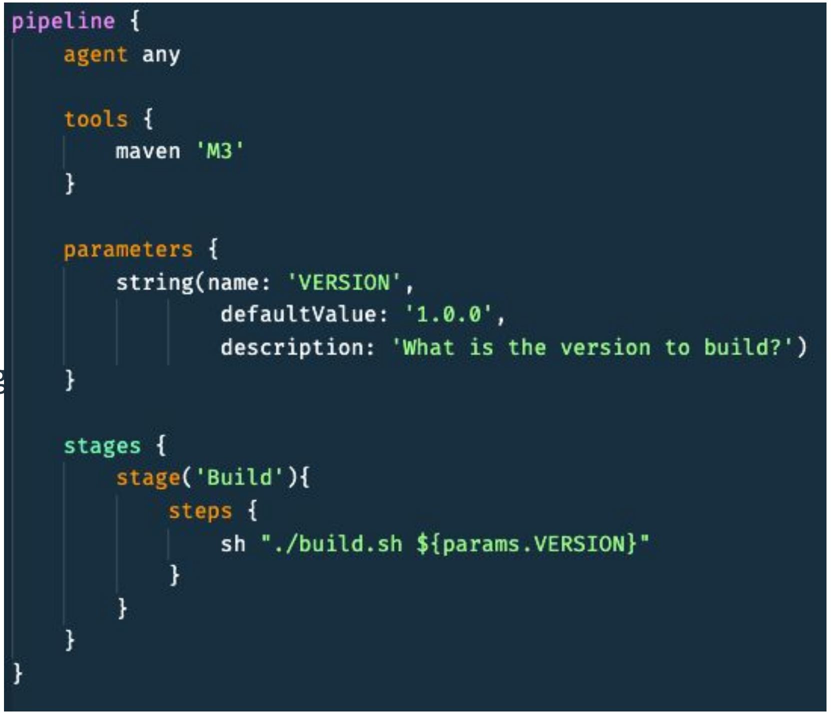




```
pipeline {
    agent any

    stages {
        stage('Build'){
            steps {
                sh 'make'
            }
        }
        stage('Test'){
            steps {
                sh 'make check'
                junit 'reports/**/*.xml'
            }
        }
        stage('Deploy'){
            steps {
                sh 'make publish'
            }
        }
    }
}
```

Directives

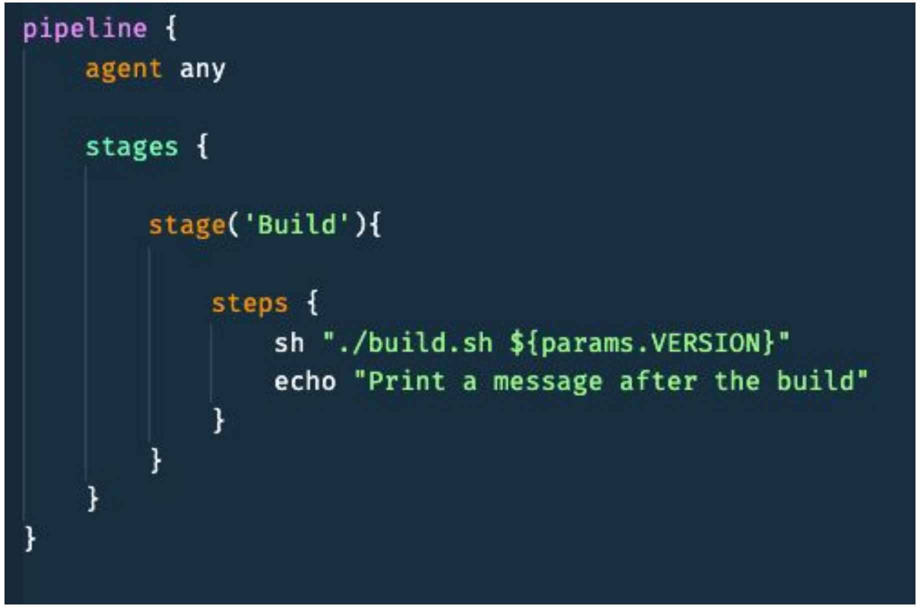
- Can stay inside of a section or pipeline block
- Common directives:
  - **Triggers** - automatically triggers the pipeline based on definition.
  - **Parameters** - define one or more parameters which an user should provide when triggering the pipeline
  - **Tools** - define the tools configured on Jenkins
  - **Stage** - inside of stage section; contains steps and agent
  - **When** - gives control to what should be executed in the pipeline



```
pipeline {
    agent any

    tools {
        maven 'M3'
    }

    parameters {
        string(name: 'VERSION',
                defaultValue: '1.0.0',
                description: 'What is the version to build?')
    }

    stages {
        stage('Build'){
            steps {
                sh "./build.sh ${params.VERSION}"
            }
        }
    }
}
```

Steps

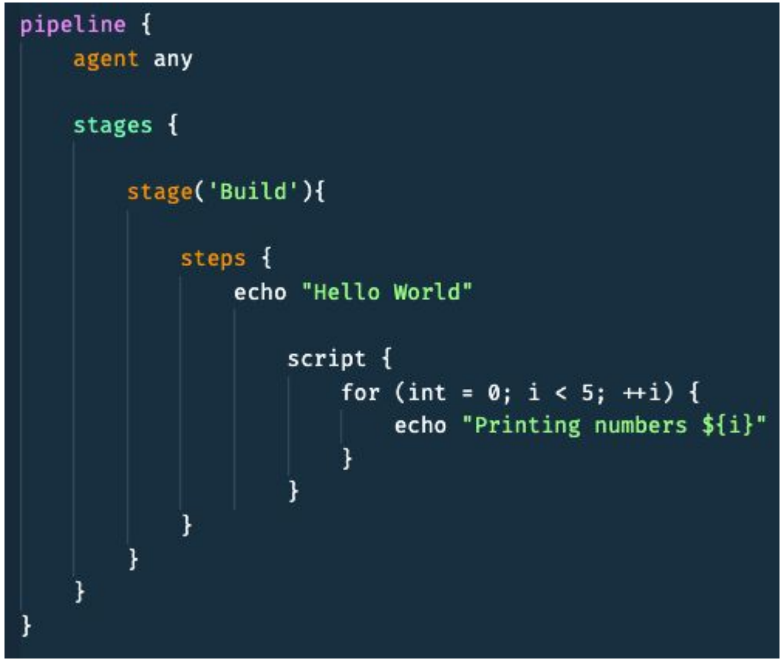
Defines all the steps of given stage



```
pipeline {
    agent any

    stages {

        stage('Build'){

            steps {
                sh "./build.sh ${params.VERSION}"
                echo "Print a message after the build"
            }
        }
    }
}
```

Script Inside of a Declarative Pipeline

Script is a step to use scripted pipeline inside of a declarative pipeline



```
pipeline {
    agent any

    stages {

        stage('Build'){

            steps {
                echo "Hello World"

                    script {
                        for (int = 0; i < 5; ++i) {
                            echo "Printing numbers ${i}"
                        }
                    }
            }
        }
    }
}
```

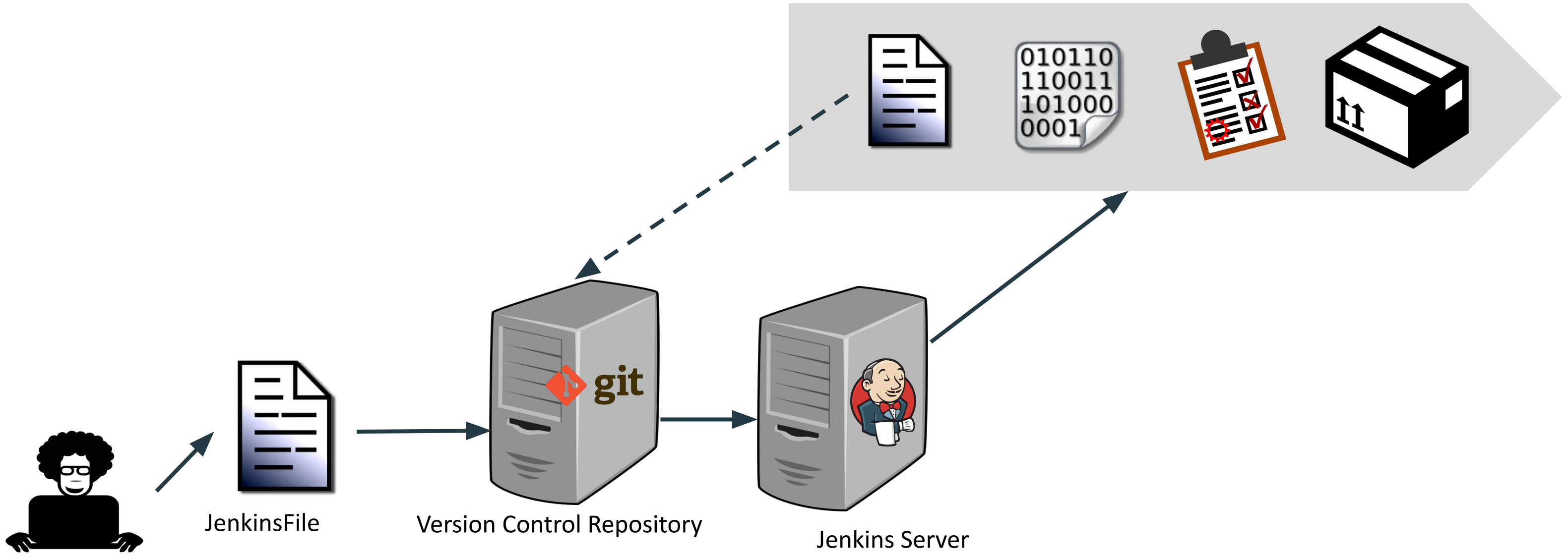
https://www.jenkins.io/doc/book/pipeline/

Building with Docker

https://www.jenkins.io/doc/book/pipeline/docker/

Understanding the Jenkins File

How the Pipeline Works with JenkinsFile?



push to docker hub Demo

https://github.com/levep/spring-petclinic-jenkins.git