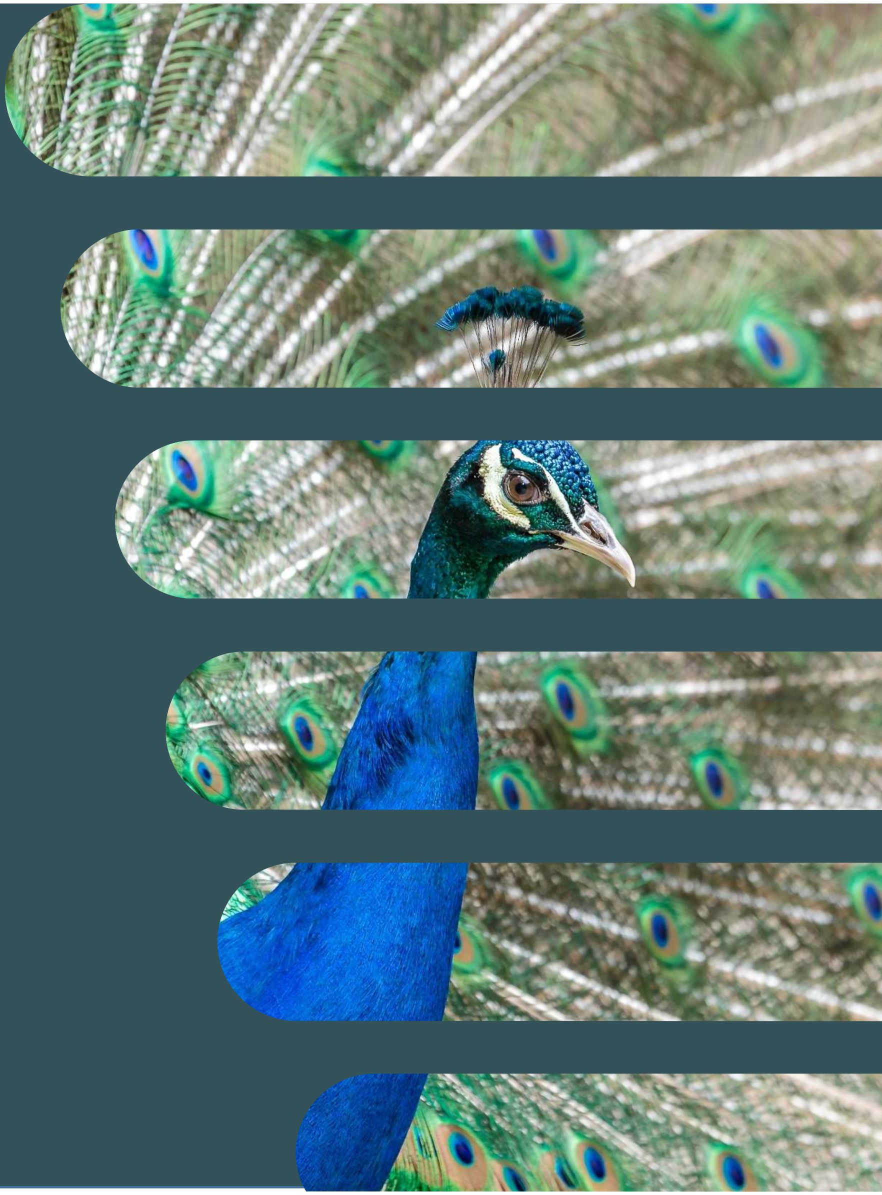
ZippyOPS

Making Automation Work

www.zippyops.com

Jenkins



Jenkins

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose.

Used to build and test your software projects continuously.

Continuously deliver your software by integrating with a large number of testing and deployment

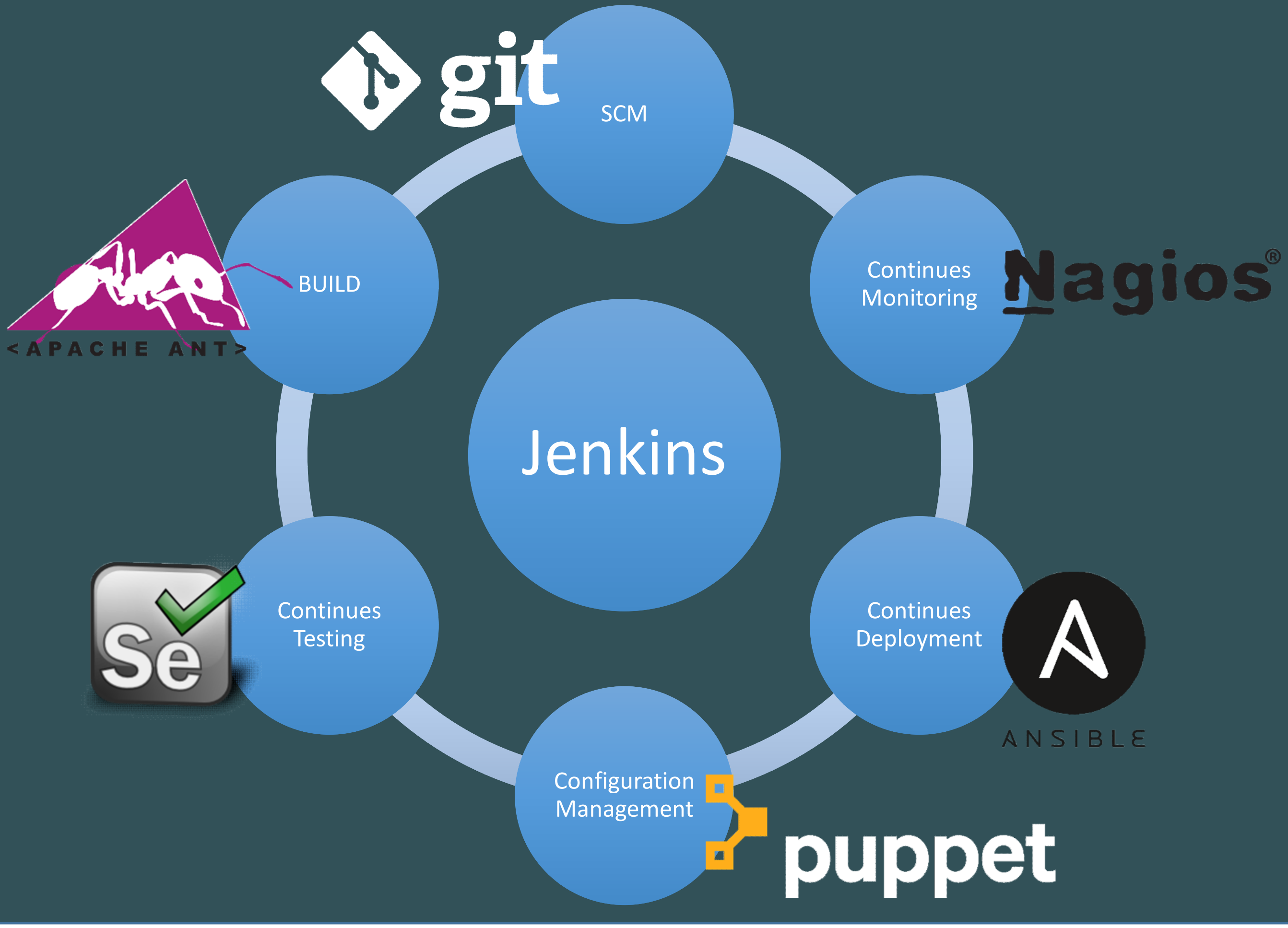
Jenkins integrates development life-cycle processes of all kinds, including build, document, test, package, stage, deploy, static analysis and much more.

Jenkins achieves Continuous Integration with the help of plugins

Example plugins: Git, Maven 2 project, Amazon EC2, HTML publisher etc.

ZippyOPS
Making Automation Work
www.zippyops.com

Jenkins integrating various DevOps stages



ZippyOPS
Making Automation Work
www.zippyops.com

Advantages of Jenkins

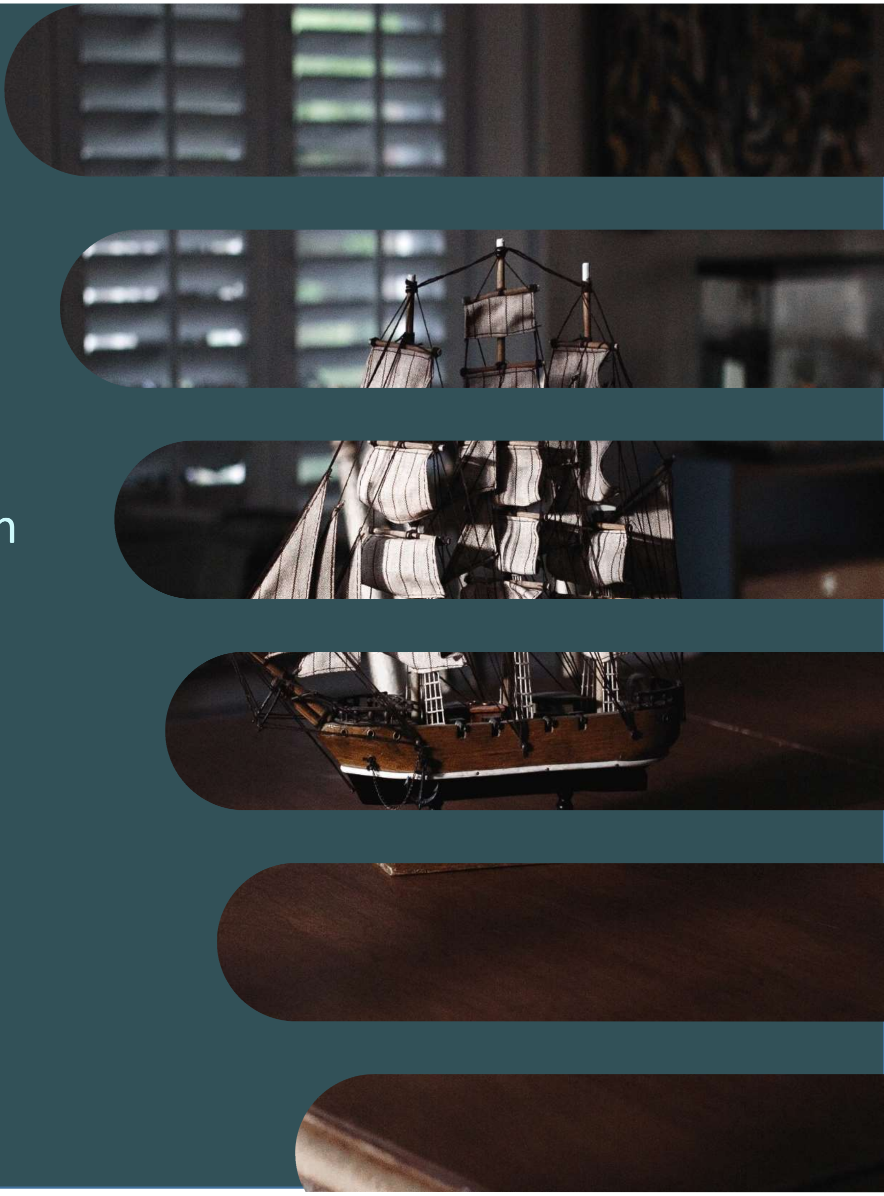
Its an open source tool with great community support.

It is easy to install.

It has around 900+ plugins to ease your work. If a plugin does not exist, just code it up and share with the community.

It is free of cost

Its built with Java and hence, it is portable on all major platforms.



ZippyOPS
Making Automation Work
www.zippyops.com

What can Jenkins Do

Can associate Jenkins with a version control server.

Can trigger builds by polling, Periodic etc.

Can execute bash scripts, shell scripts

Can trigger ANT and Maven Targets

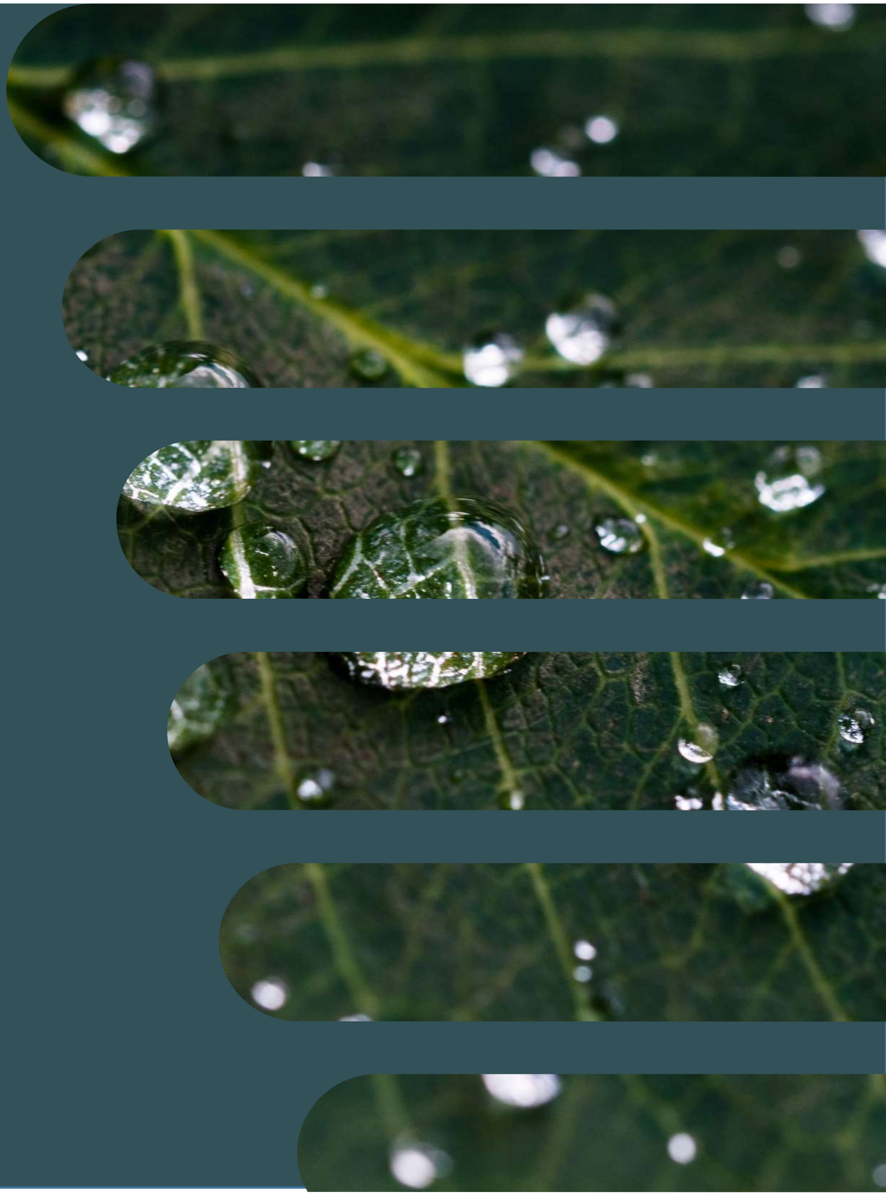
Can trigger testing jobs

Can create sequence and pipeline jobs

Can Publish results and send email notifications

Can triggers CD system for deployment

Can do lot more stuffs…



ZippyOPS
Making Automation Work
www.zippyops.com

uses of the Jenkins

Building snapshot and release artifacts for your application.

Deployment of the released artifact with custom scripts.

Continuous integration pipeline support for establishing software development life cycle work flow for your application.

Support for scheduled builds & automation test execution.

ZippyOPS
Making Automation Work
www.zippyops.com

What is Continuous Integration?

Continuous Integration is a development practice in which the developers are required to commit changes to the source code in a shared repository several times a day or more frequently.

Every commit made in the repository is then built.

This allows the teams to detect the problems early.

There are several other functions like deploying the build application on the test server, providing the concerned teams with the build and test results etc.

ZippyOPS
Making Automation Work
www.zippyops.com

Benefits of CI

Catch issue fast and hip them in the bud

Everyone can see what is happening

Automate the build

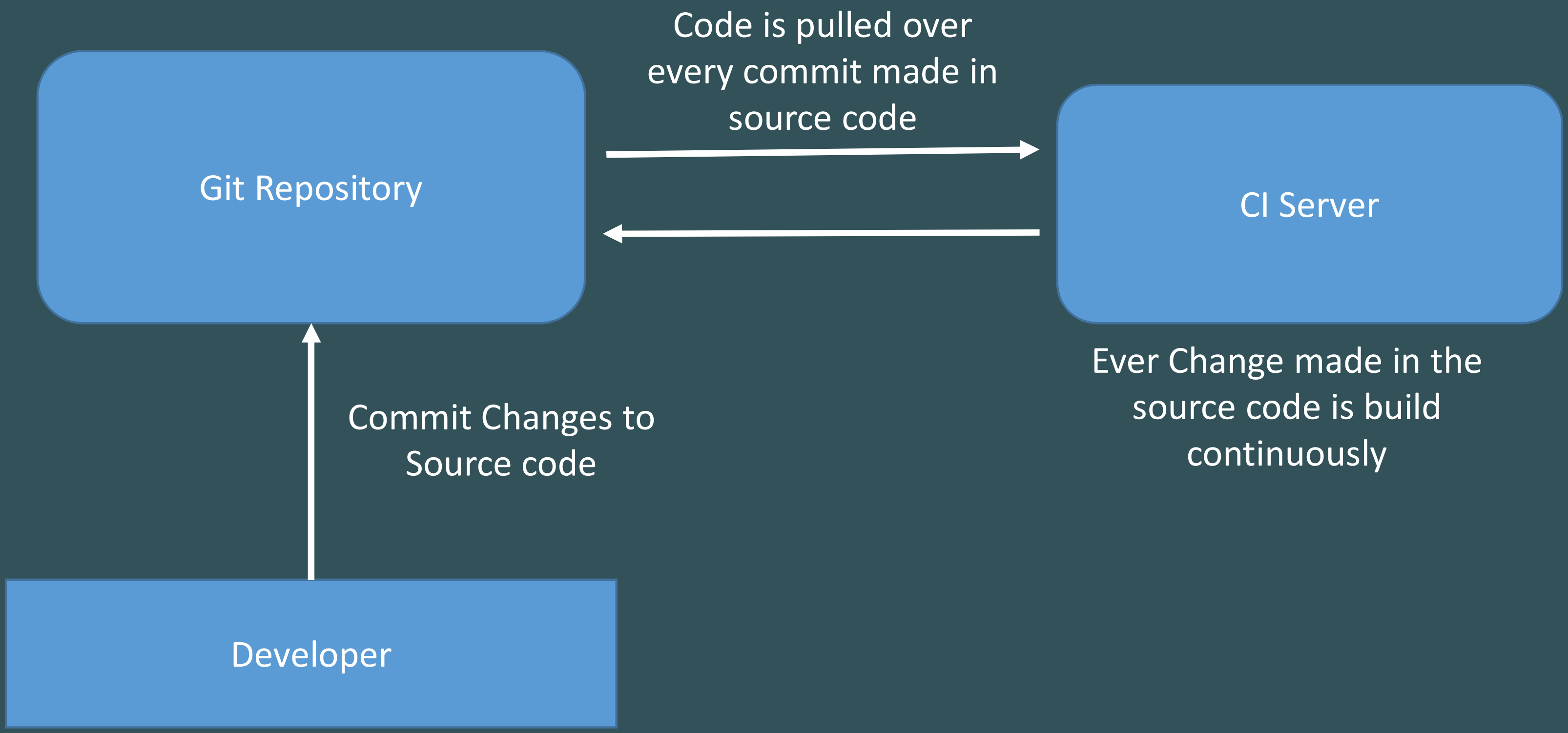
Keep the build fast

Stop waiting to find out if your codes going to work

CI Leeds to continuous deployment allowing you to deliver software more rapidly

ZippyOPS
Making Automation Work
www.zippyops.com

Continuous Integration



Git Repository

Code is pulled over every commit made in source code

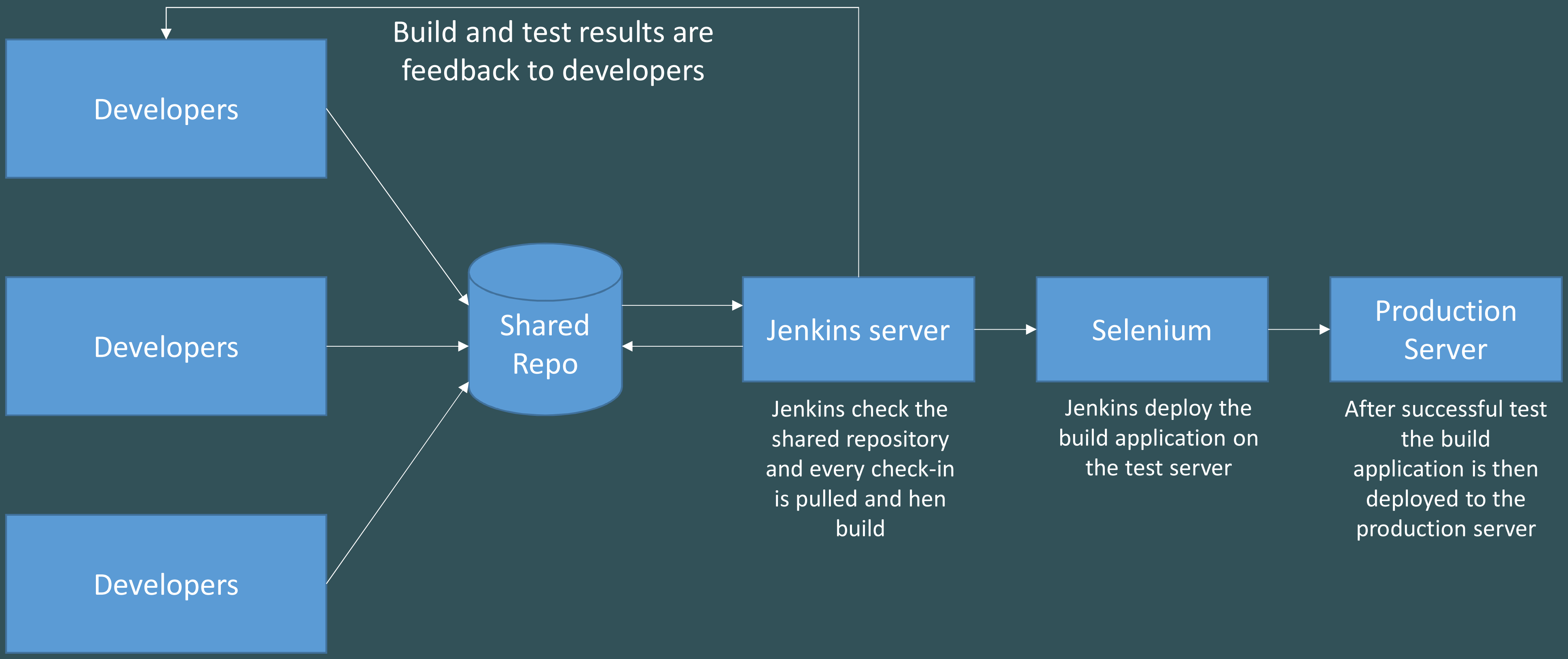
CI Server

Ever Change made in the source code is build continuously

Commit Changes to Source code

Developer

Continuous Integration With Jenkins :



Build and test results are feedback to developers

Developers

Developers

Developers

Shared Repo

Jenkins server

Jenkins check the shared repository and every check-in is pulled and hen build

Selenium

Jenkins deploy the build application on the test server

Production Server

After successful test the build application is then deployed to the production server

Continuous Integration With Jenkins

First, a developer commits the code to the source code repository. Meanwhile, the Jenkins server checks the repository at regular intervals for changes.

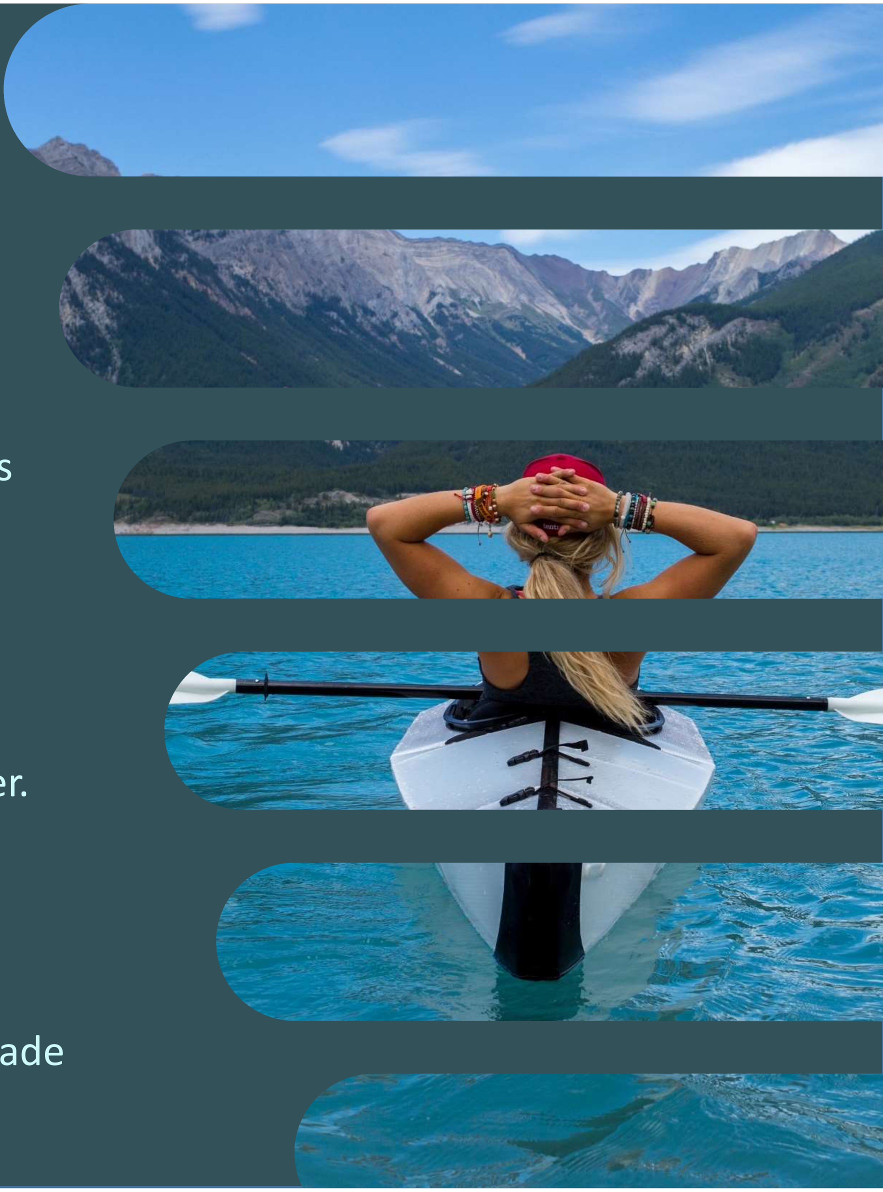
Soon after a commit occurs, the Jenkins server detects the changes that have occurred in the source code repository. Jenkins will pull those changes and will start preparing a new build.

If the build fails, then the concerned team will be notified.

If built is successful, then Jenkins deploys the built in the test server.

After testing, Jenkins generates a feedback and then notifies the developers about the build and test results.

It will continue to check the source code repository for changes made in the source code and the whole process keeps on repeating.



Before and After Jenkins

Before Jenkins	After Jenkins
The entire source code was built and then tested.	Every commit made in the source code is built and tested.
Developer have to wait for test result.	Developer know the test result of every commit made in the source code on the run.
No feed back	Feedback is present

ZippyOPS
Making Automation Work
www.zippyops.com

Jenkins Architecture

Jenkins uses a Master-Slave architecture to manage distributed builds.

In this architecture, Master and Slave communicate through TCP/IP protocol.

Jenkins Master :

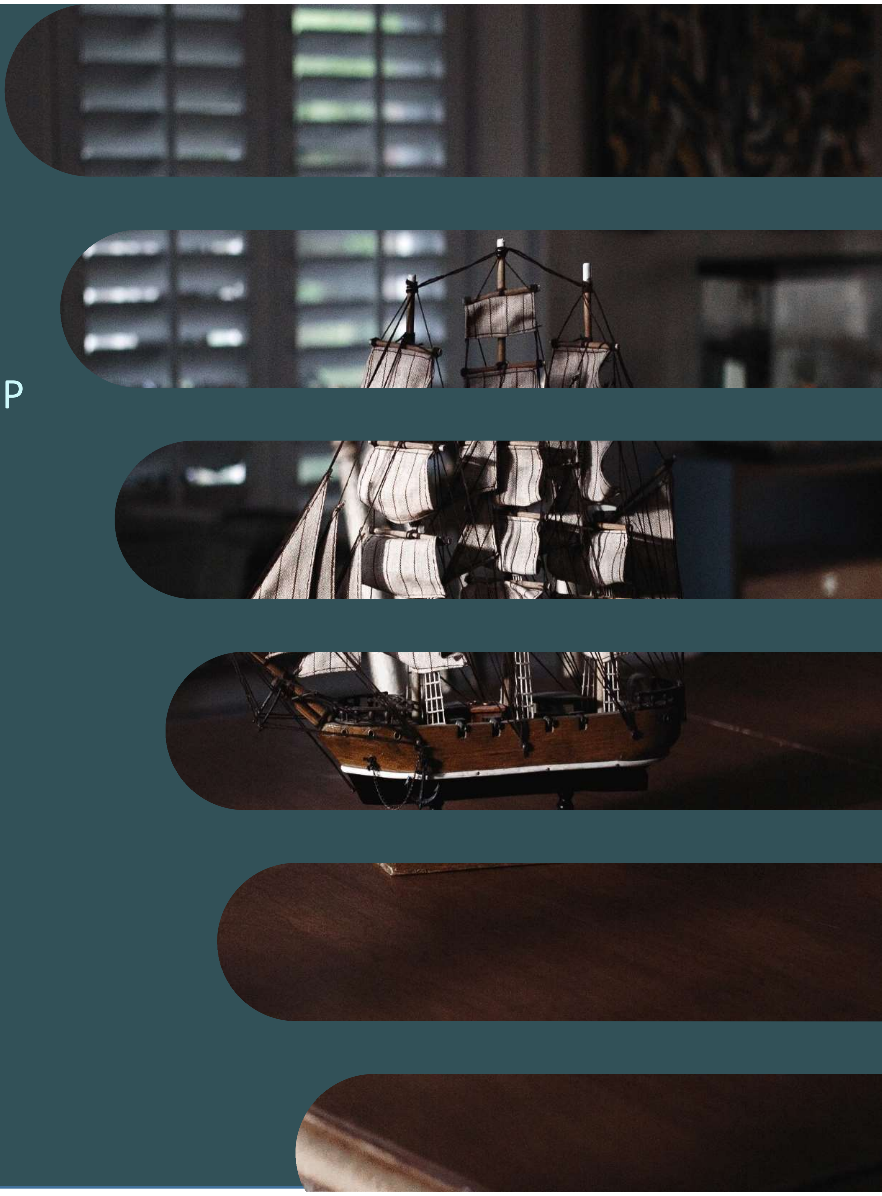
Scheduling build jobs.

Dispatching builds to the slaves for the actual execution.

Monitor the slaves (possibly taking them online and offline as required).

Recording and presenting the build results.

A Master instance of Jenkins can also execute build jobs directly.



Jenkins Architecture

Jenkins slave:

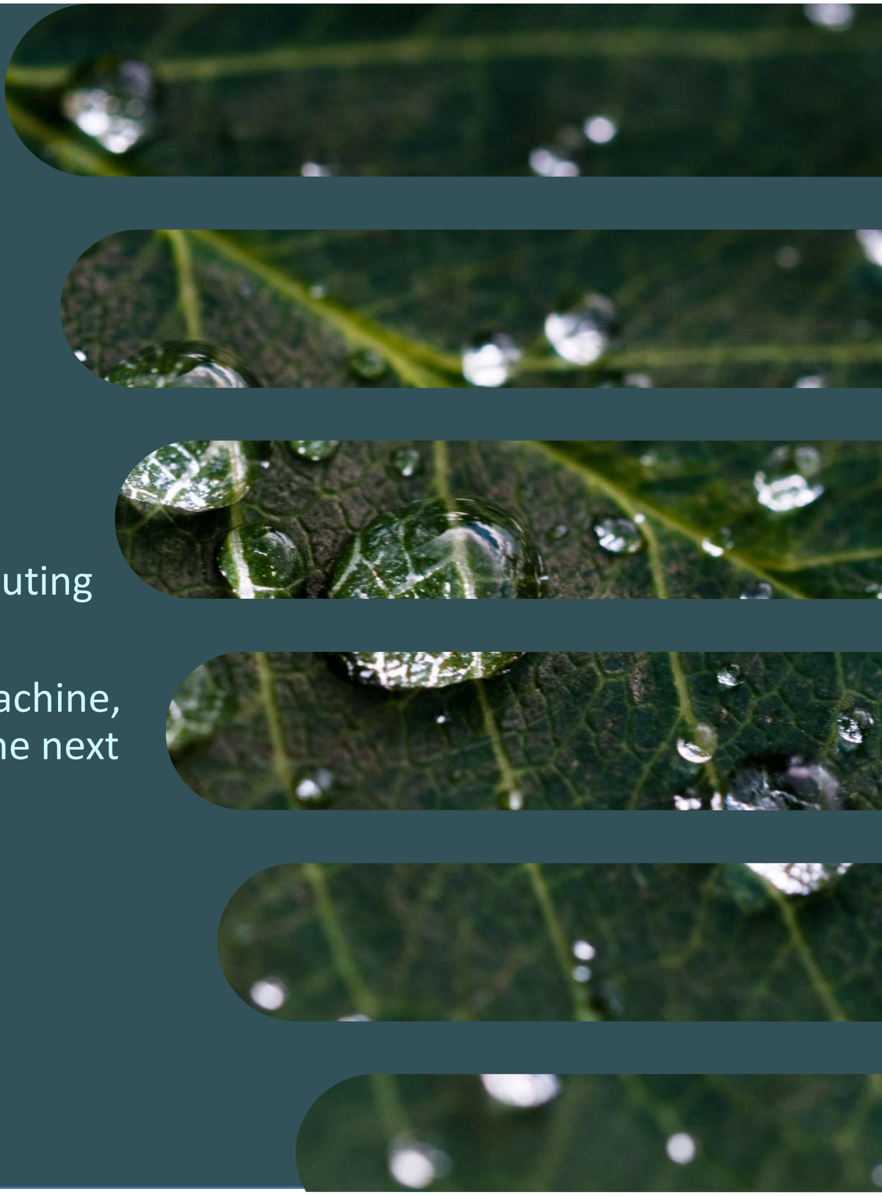
A Slave is a Java executable that runs on a remote machine.

It hears requests from the Jenkins Master instance.

Slaves can run on a variety of operating systems.

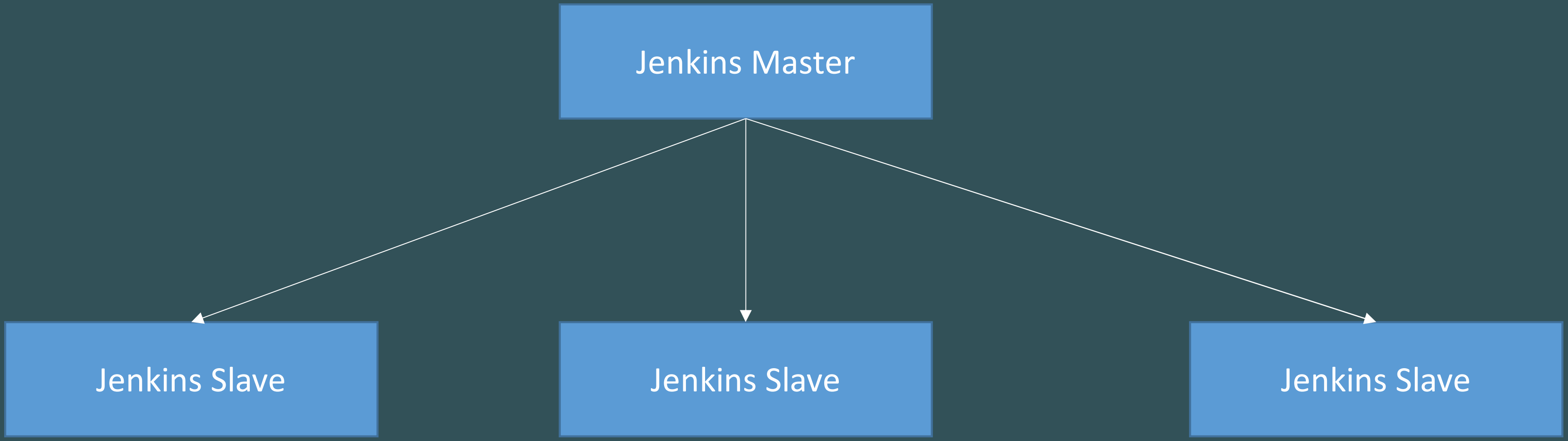
The job of a Slave is to do as they are told to, which involves executing build jobs dispatched by the Master.

You can configure a project to always run on a particular Slave machine, or a particular type of Slave machine, or simply let Jenkins pick the next available Slave.



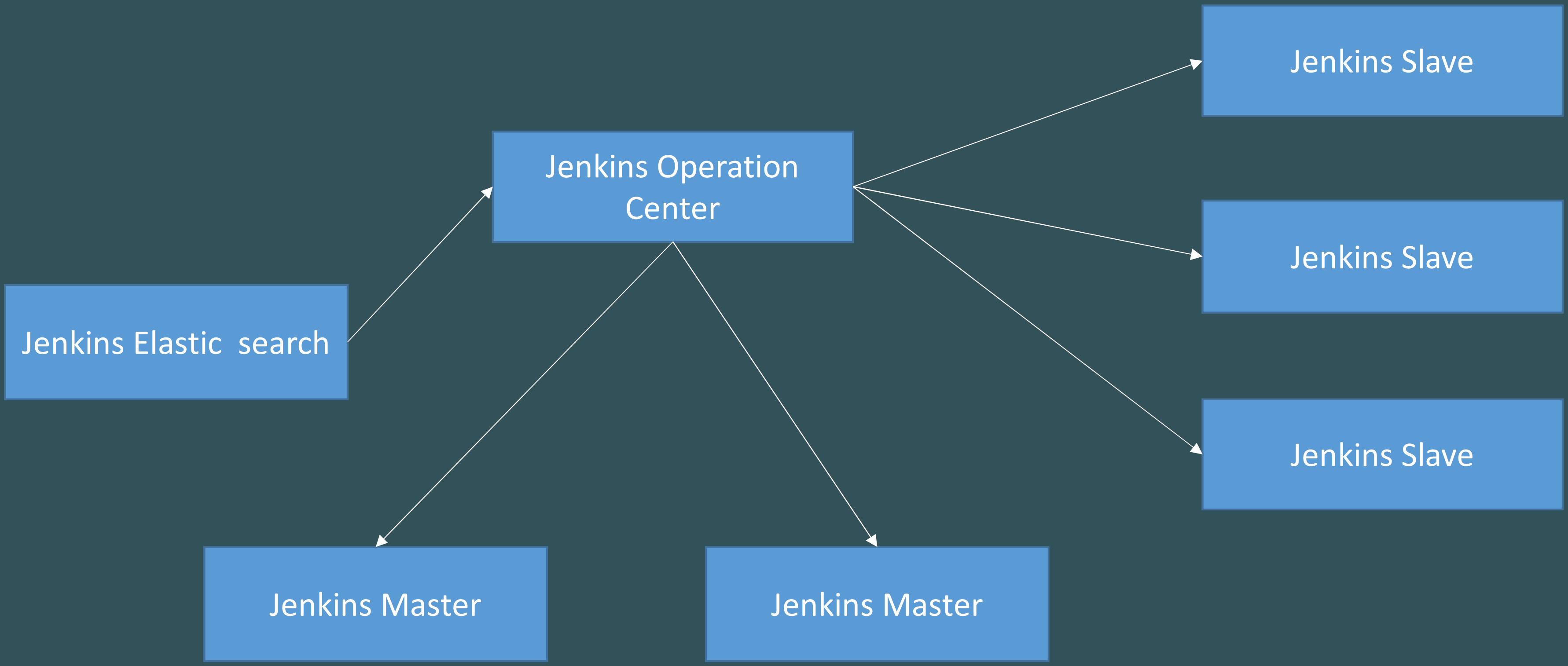
ZippyOPS
Making Automation Work
www.zippyops.com

Jenkins Architecture



ZippyOPS
Making Automation Work
www.zippyops.com

Jenkins Enterprise Architecture



Jenkins Build Pipeline

It is used to know which task Jenkins is currently executing.

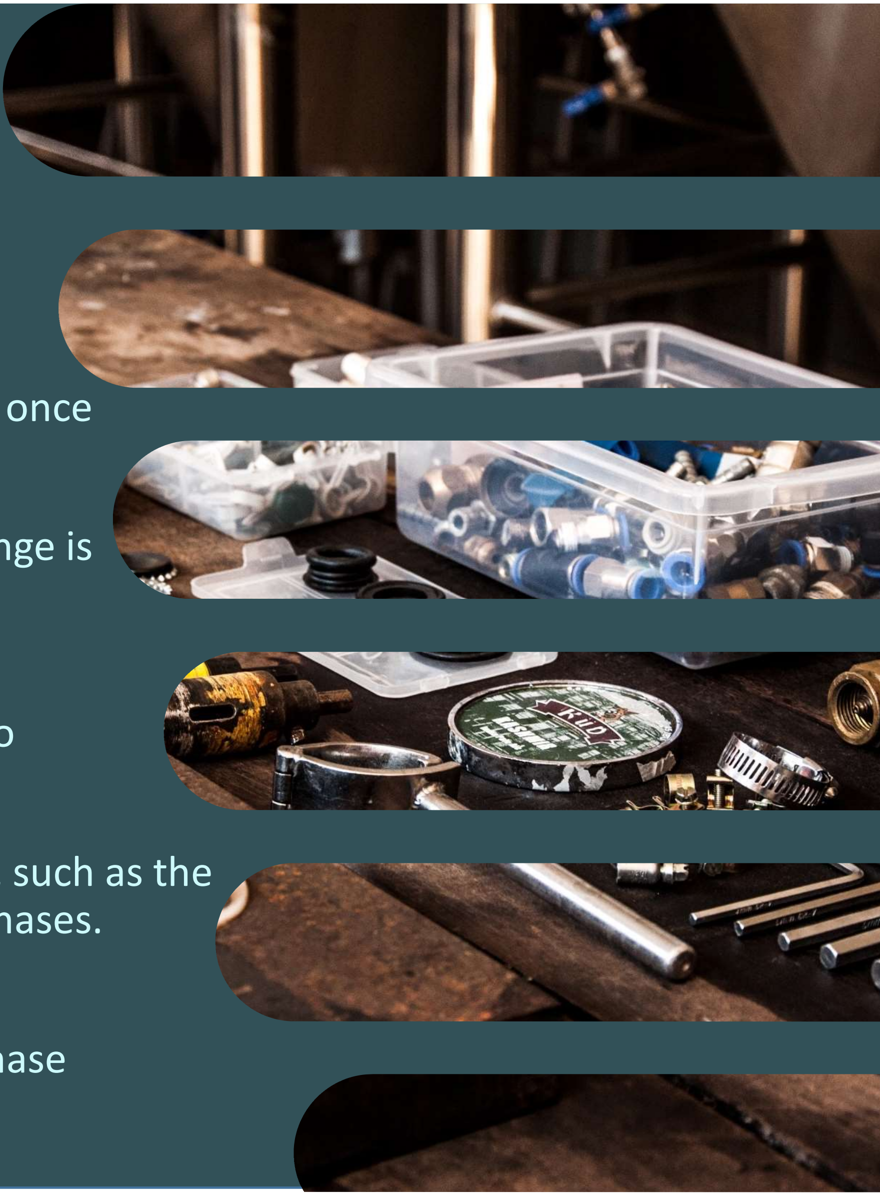
Often several different changes are made by several developers at once

so it is useful to know which change is getting tested or which change is sitting in the queue or which build is broken.

The Jenkins Pipeline gives you an overview of where tests are up to

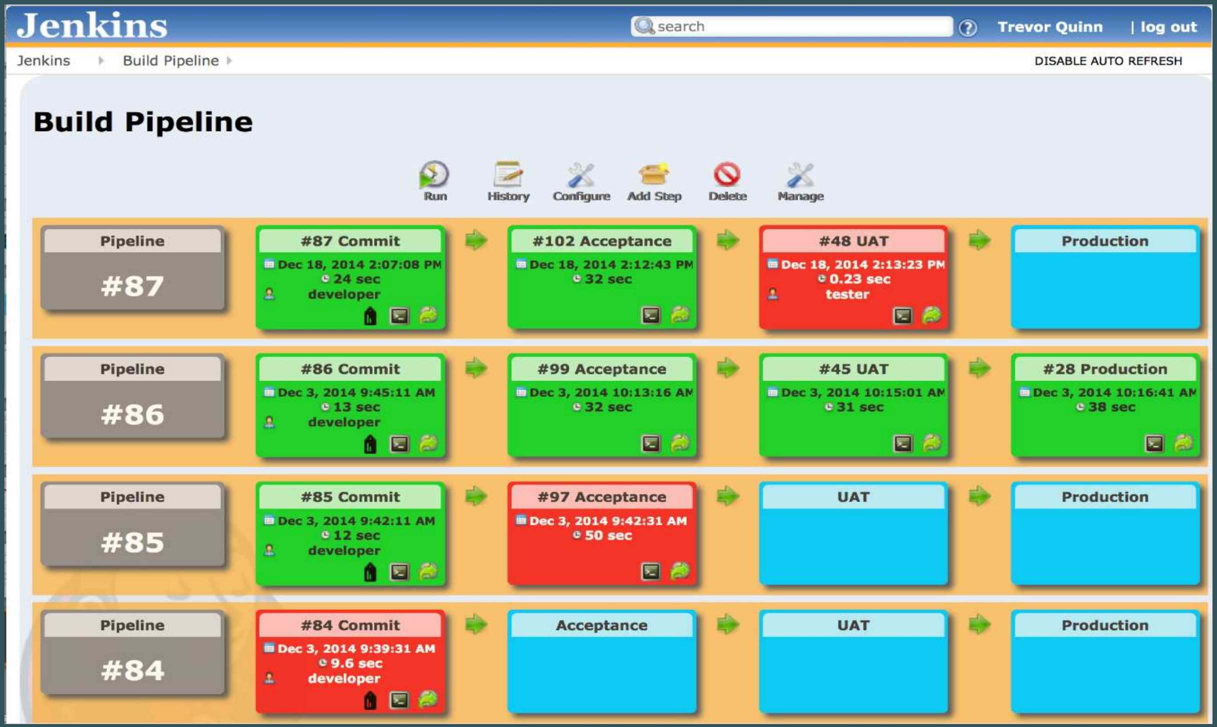
In build pipeline the build as a whole is broken down into sections, such as the unit test, acceptance test, packaging, reporting and deployment phases.

if one phase is successful, it automatically moves on to the next phase



ZippyOPS
Making Automation Work
www.zippyops.com

Jenkins Build Pipeline

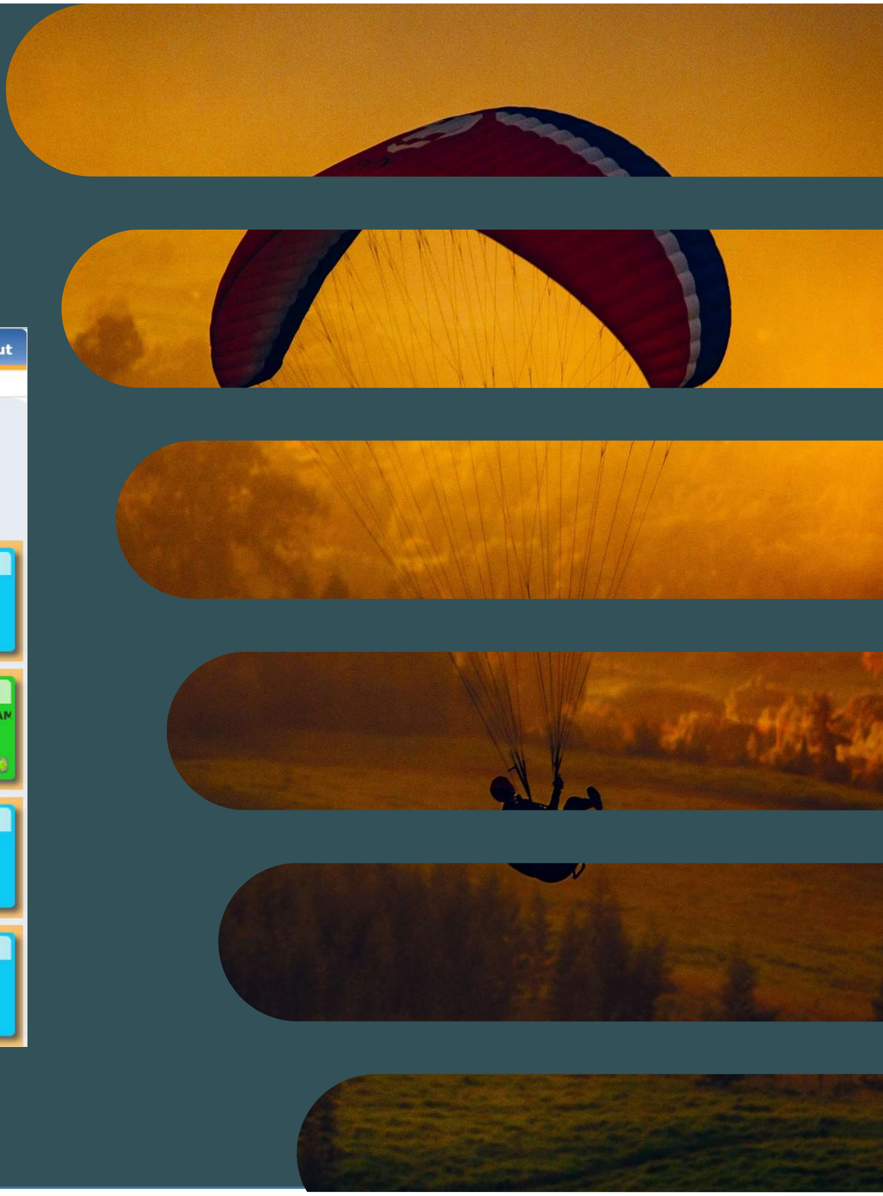


Jenkins search ? Trevor Quinn | log out

Jenkins › Build Pipeline › DISABLE AUTO REFRESH

Build Pipeline

Pipeline #87	#87 Commit Dec 18, 2014 2:07:08 PM 24 sec developer	#102 Acceptance Dec 18, 2014 2:12:43 PM 32 sec	#48 UAT Dec 18, 2014 2:13:23 PM 0.23 sec tester	Production
Pipeline #86	#86 Commit Dec 3, 2014 9:45:11 AM 13 sec developer	#99 Acceptance Dec 3, 2014 10:13:16 AM 32 sec	#45 UAT Dec 3, 2014 10:15:01 AM 31 sec	#28 Production Dec 3, 2014 10:16:41 AM 38 sec
Pipeline #85	#85 Commit Dec 3, 2014 9:42:11 AM 12 sec developer	#97 Acceptance Dec 3, 2014 9:42:31 AM 50 sec	UAT	Production
Pipeline #84	#84 Commit Dec 3, 2014 9:39:31 AM 9.6 sec developer	Acceptance	UAT	Production



Jenkins projects

By default 4 Projects are available

- Freestyle Project
- Multiconfiguration Job
- Monitor an External Job
- Maven Project

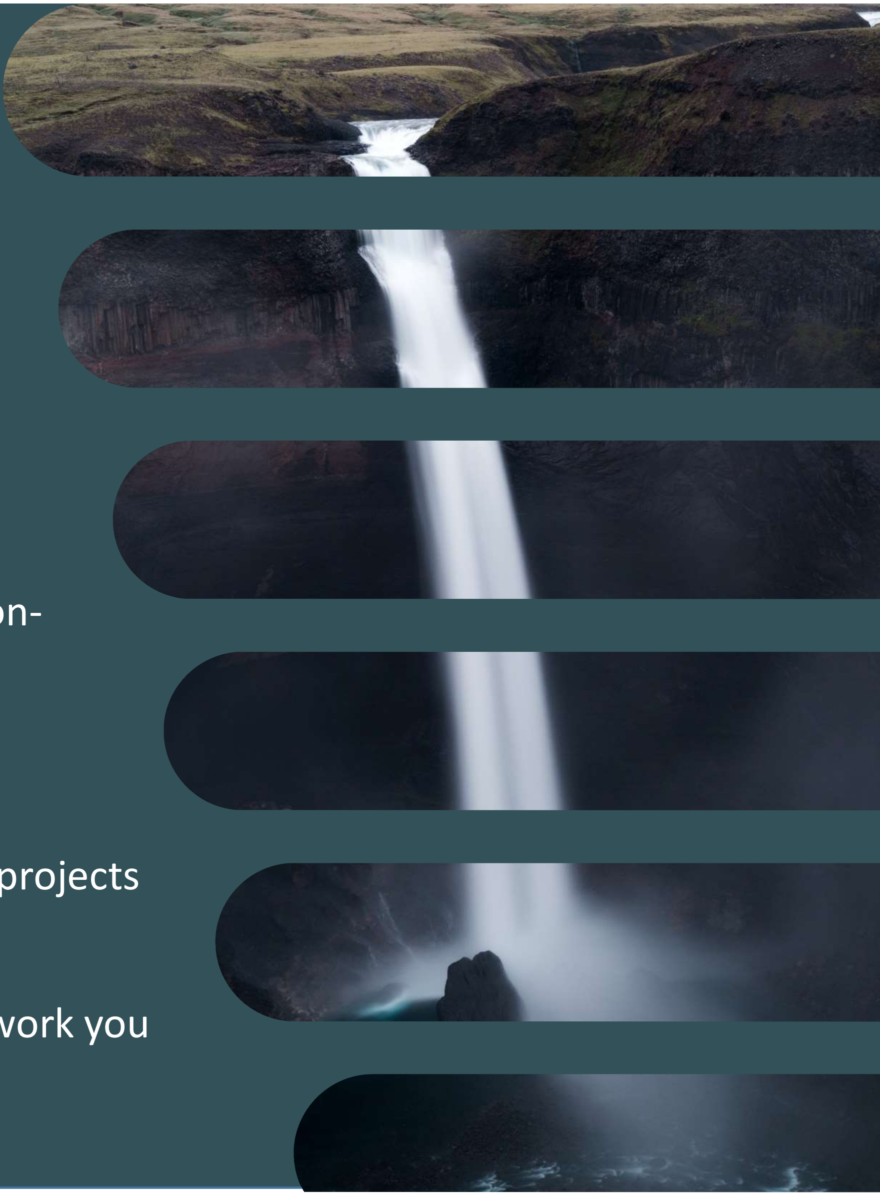
Freestyle Project

Freestyle build jobs are general-purpose build jobs, which provides maximum flexibility.

The freestyle build job is the most flexible and configurable option can be used for any type of project.

It is relatively straightforward to set up

Jenkins projects



Multi-configuration Job

It allows you run the same build job on different environments.

It is used for testing an application in different environments with different databases.

Monitor an External Job

The “Monitor an external job” build job lets you keep an eye on non-interactive processes, such as cron jobs.

Maven Project

The “maven2/3 project” is a build job specially adapted to Maven projects

Jenkins understands Maven pom files and project structures

Can use the information gleaned from the pom file to reduce the work you need to do to set up your project.

ZippyOPS
Making Automation Work
www.zippyops.com

Jenkins Pipeline

Jenkins Pipeline is written into a text file (called a Jenkinsfile) which in turn is checked into a project’s source control repository.This is the foundation of "Pipeline-as-Code"; treating the continuous delivery pipeline a part of the application to be versioned and reviewed like any other code. Creating a Jenkinsfile provides a number of immediate benefits:

- Automatically create Pipelines for all Branches and Pull Requests
- Code review/iteration on the Pipeline
- Audit trail for the Pipeline
- Single source of truth [3] for the Pipeline, which can be viewed and edited by multiple members of the project.

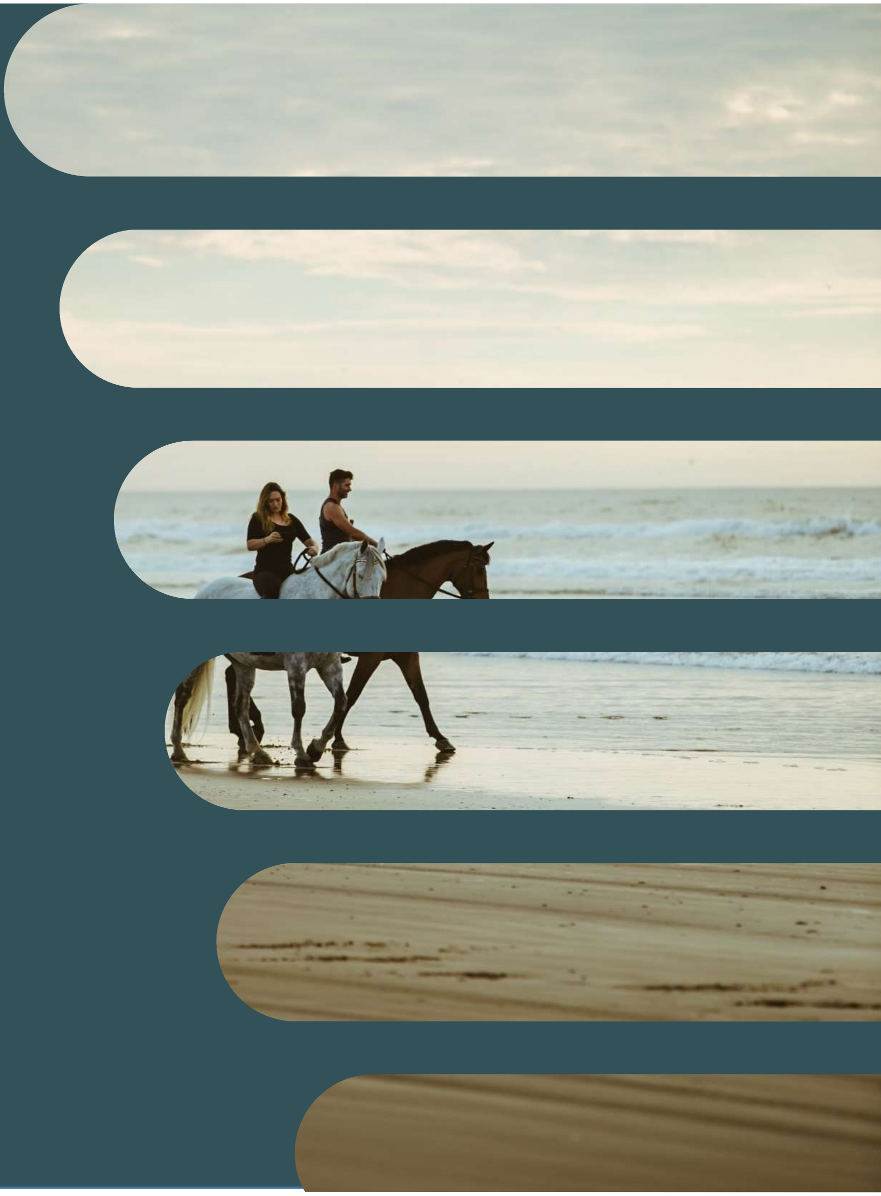
While the syntax for defining a Pipeline, either in the web UI or with a Jenkinsfile, is the same, it’s generally considered best practice to define the Pipeline in a Jenkinsfile and check that in to source control.

Jenkins Pipeline

Jenkinsfile (Declarative Pipeline)

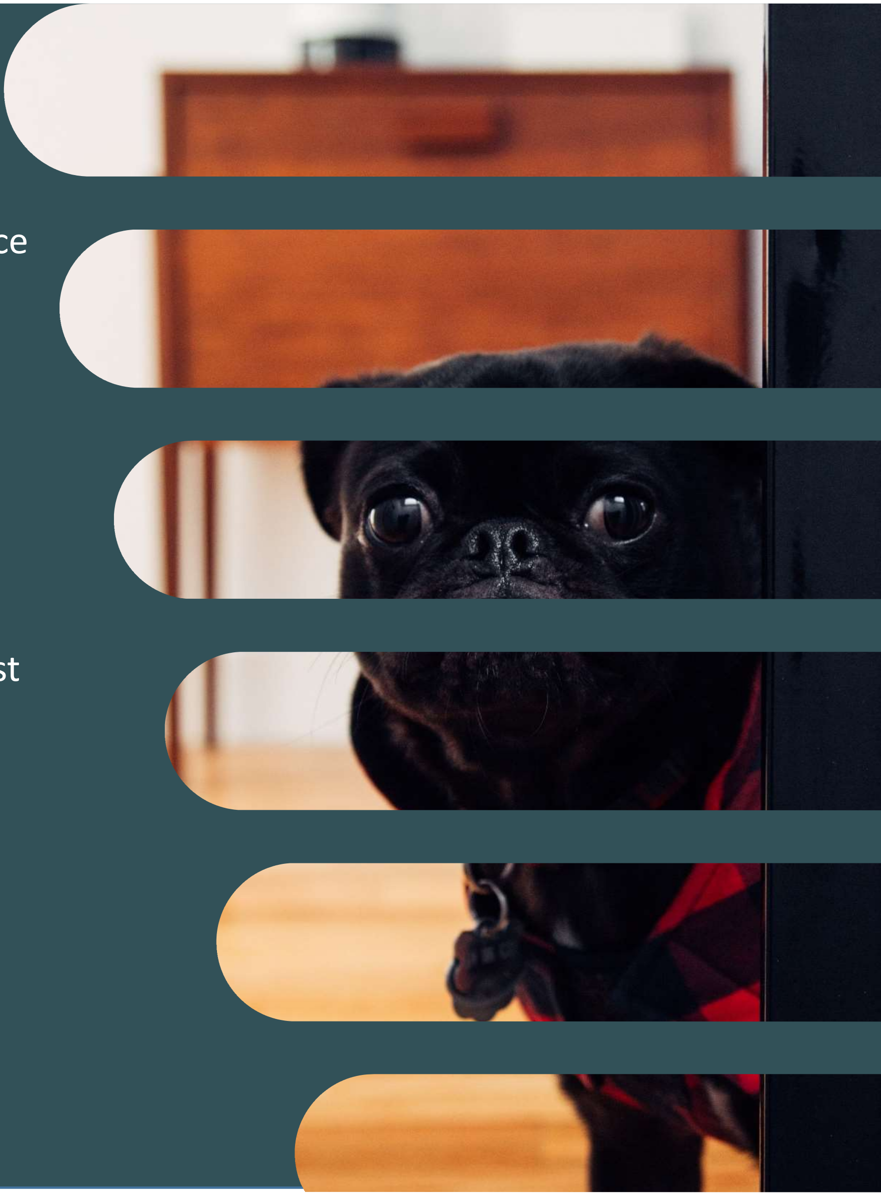
```
pipeline {
    agent any

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



Jenkins Pipeline

- **agent** indicates that Jenkins should allocate an executor and workspace for this part of the Pipeline.
- **stage** describes a stage of this Pipeline.
- **steps** describes the steps to be run in this stage
- **sh** executes the given shell command
- **junit** is a Pipeline step provided by the JUnit plugin for aggregating test report



Why Pipeline

Jenkins is, fundamentally, an automation engine which supports a number of automation patterns. Pipeline adds a powerful set of automation tools onto Jenkins, supporting use cases that span from simple continuous integration to comprehensive continuous delivery pipelines. By modeling a series of related tasks, users can take advantage of the many features of Pipeline:

- Code: Pipelines are implemented in code and typically checked into source control, giving teams the ability to edit, review, and iterate upon their delivery pipeline.
- Durable: Pipelines can survive both planned and unplanned restarts of the Jenkins master.
- Pausable: Pipelines can optionally stop and wait for human input or approval before continuing the Pipeline run.
- Versatile: Pipelines support complex real-world continuous delivery requirements, including the ability to fork/join, loop, and perform work in parallel.
- Extensible: The Pipeline plugin supports custom extensions to its DSL and multiple options for integration with other plugins.