LEVEL UP
www.level-up.one

Continuous Integration

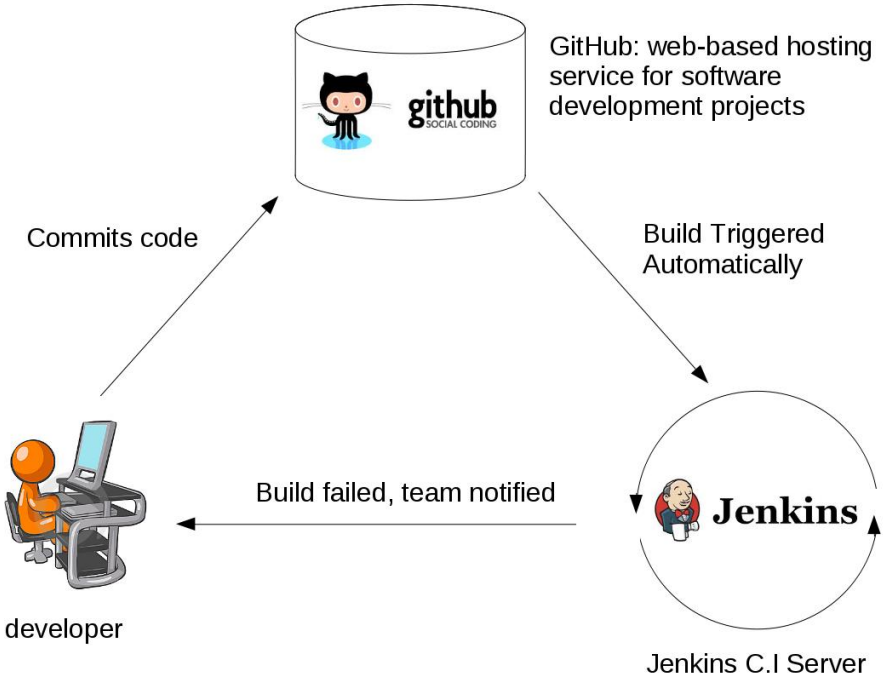
Jenkins

Continuous Integration

- What is Continuous Integration?
- Why do we need it?
- Different phases of adopting Continuous Integration

Jenkins

LEVEL UP
www.level-up.one



What is Continuous Integration?

- Developers commit code to a shared repository on a regular basis.
- Version control system is being monitored. When a commit is detected, a build will be triggered automatically.
- If the build is not green, developers will be notified immediately.

Why do we need Continuous Integration?

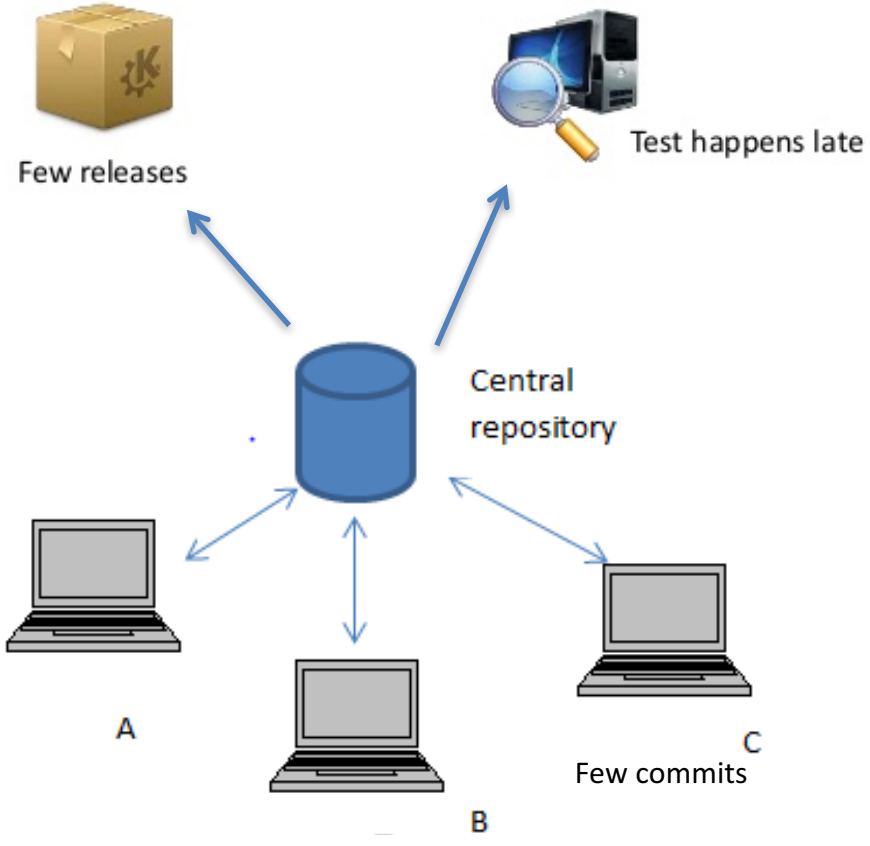
- Detect problems or bugs, as early as possible, in the development life cycle.

- Since the entire code base is integrated, built and tested constantly , the potential bugs and errors are caught earlier in the life cycle which results in better quality software.

Different stages of adopting Continuous Integration

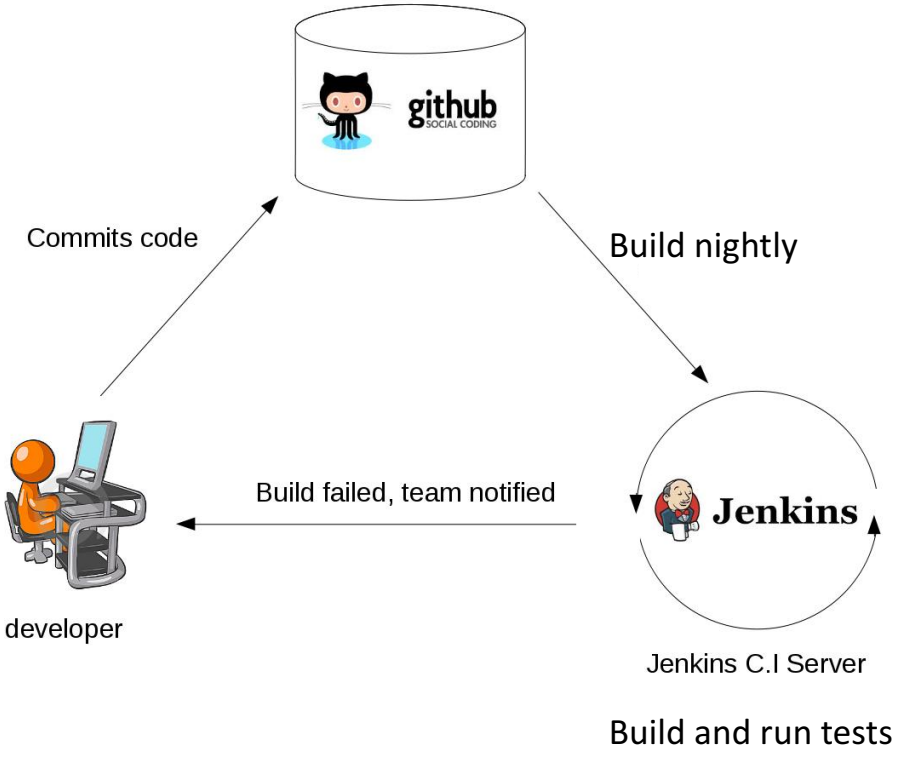
Jenkins

LEVEL UP
www.level-up.one



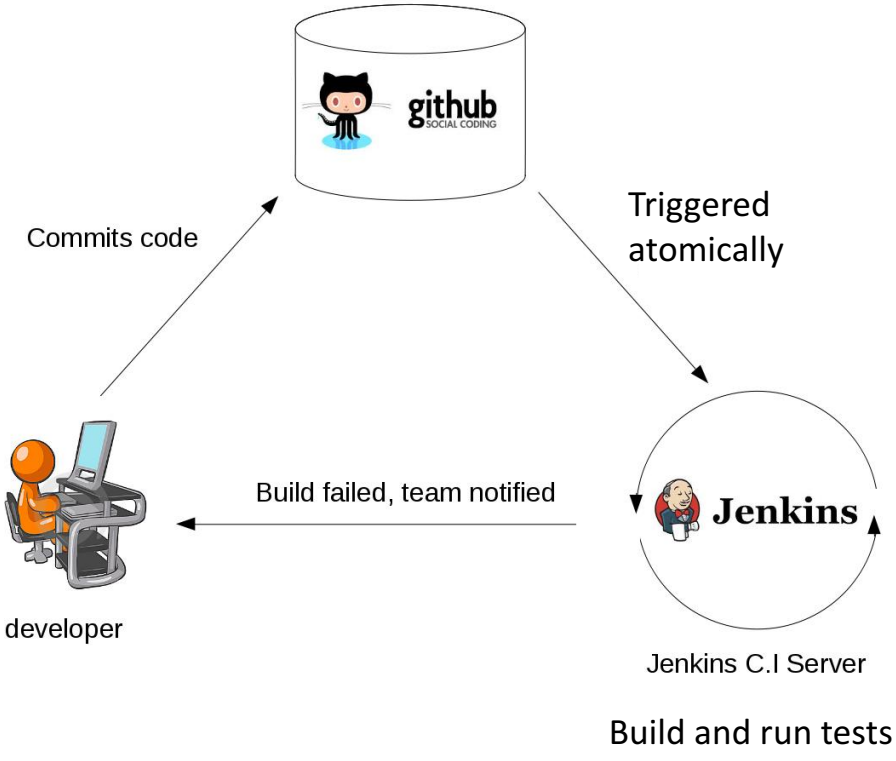
Stage 1:

- No build servers.
- Developers do NOT commit on a regular basis.
- Changes are integrated and tested manually.
- Few releases.



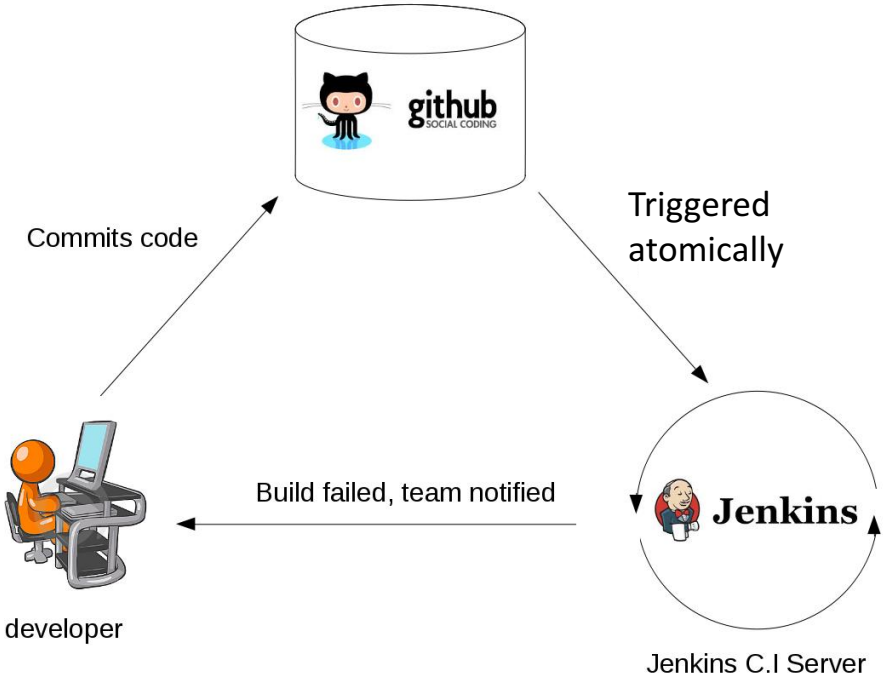
Stage 2:

- Automated builds are scheduled on a regular basis.
- Build script compiles the application and runs a set of automated tests.
- Developers now commit their changes regularly.
- Build servers would alert the team members in case of build failure.



Stage 3:

- A build is triggered whenever new code is committed to the central repository.
- Broken builds are usually treated as a high priority issue and are fixed quickly.



Build, run code quality and code coverage metrics along with tests

Stage 4:

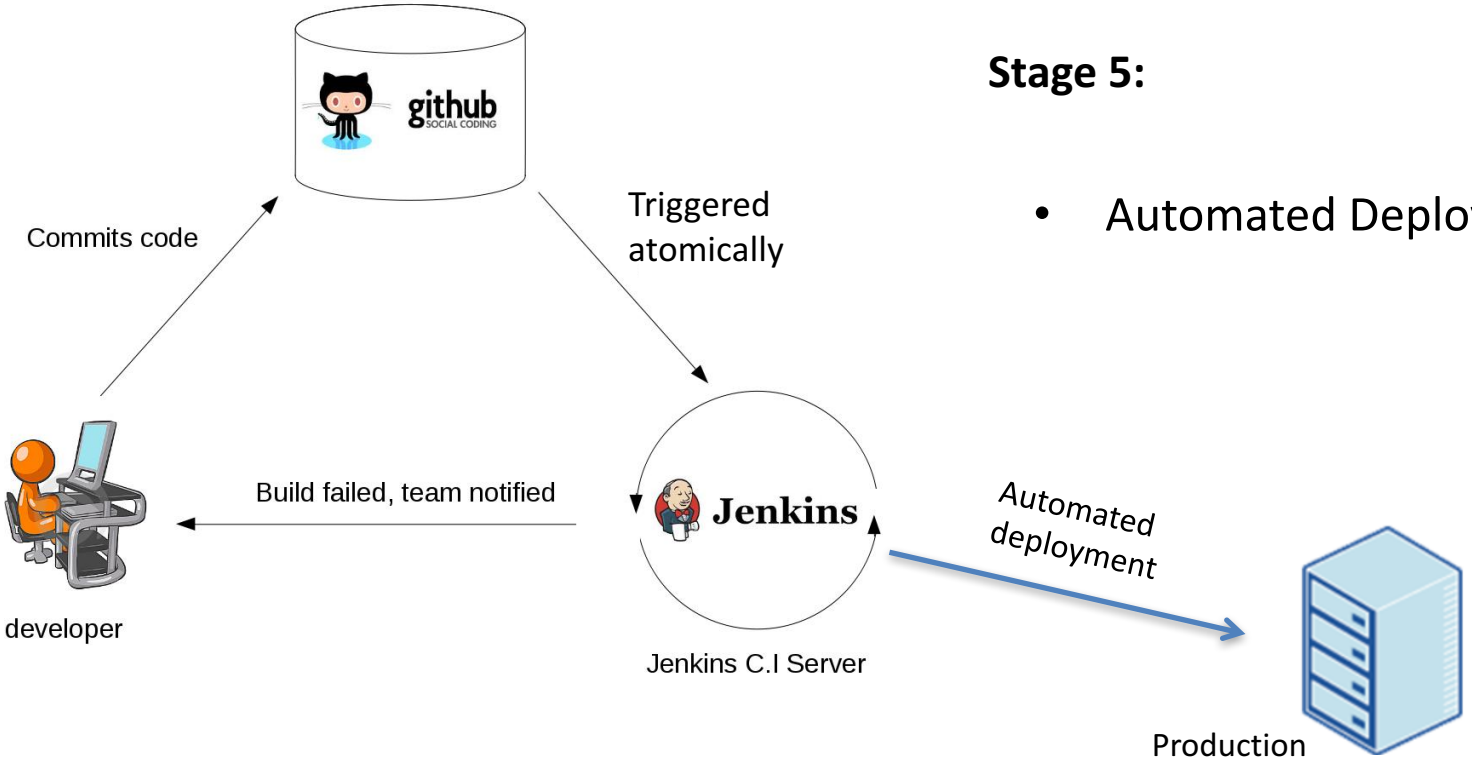
- Automated code quality and code coverage metrics are now run along with unit tests to continuously evaluate the code quality.

Is the code coverage increasing?

Do we have fewer and fewer build failures?

Stage 5:

- Automated Deployment



Continuous Integration

Continuous Delivery

Continuous Deployment

Jenkins

LEVEL UP
www.level-up.one

- **Continuous Integration**

  The practice of merging development work with the main branch constantly.

- **Continuous Delivery**

  Continual delivery of code to an environment once the code is ready to ship. This could be staging or production. The idea is the product is delivered to a user base, which can be QAs or customers for review and inspection.

- **Continuous Deployment**

  The deployment or release of code to production as soon as it is ready.

How to implement Continuous Integration?

Non-hosted solutions

circleci

Hosted solutions

Continuous Integration is also a mindset

- Fixing broken builds should be treated as a high priority issue for all team members.
- The deployment process should be automated, with no manual steps involved.
- All team members should focus on contributing to high-quality tests because the confidentiality of the CI process highly depends on the quality of the tests.

A Brief Introduction to Jenkins and the History of Jenkins

Jenkins

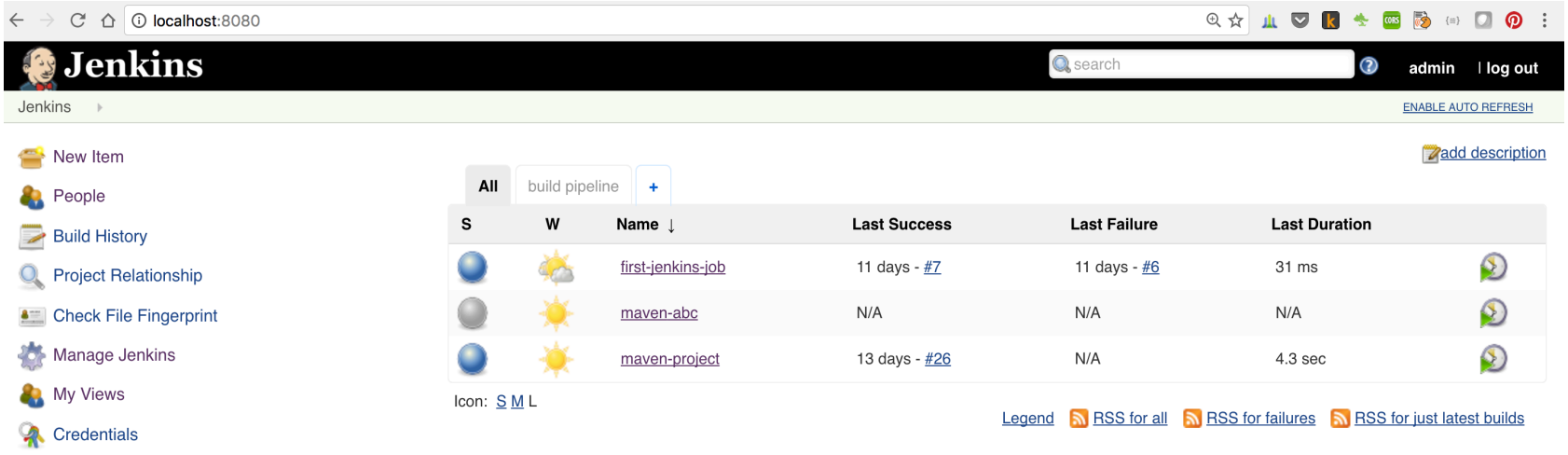
LEVEL UP
www.level-up.one

What is Jenkins

- Jenkins is a continuous integration and build server.
- It is used to manually, periodically, or automatically build software development projects.
- It is an open source Continuous Integration tool written in Java.
- Jenkins is used by teams of all different sizes, for projects with various languages.

Why Jenkins is popular

- Easy to use
- Great extensibility
  - Support different version control systems
  - Code quality metrics
  - Build notifiers
  - UI customization



Plugins by topic

Source code management

Jenkins has native support for Subversion and CVS as well as the following plugins:

- AccuRev Plugin — This plugin allows you to use AccuRev as a SCM.
- Anchore Container Image Scanner Plugin — Allows users to add a build step to run the Anchore container image scanner.
- archive-files-scm-plugin — ArchiveFilesSCM - This plugin for Jenkins checkouts archive files and extracts to Jenkins job workspace
- AWS CodePipeline Plugin — AWS CodePipeline is a continuous delivery service for fast and reliable application updates.
- Bazaar Plugin — This plugin integrates Bazaar version control system to Jenkins. The plugin requires the Bazaar binary (bzr) to be installed on the target machine.
- Bitbucket Branch Source Plugin — Multibranch projects and Team/Project folders from Bitbucket Cloud and Server. Please note that this plugin requires a server running BitBucket 4.0 or later; Stash 3.x and earlier are not supported.
- BitKeeper Plugin — Add BitKeeper support to Jenkins
- BlameSubversion — This plug-in provides utilities for getting svn info from upstream job to downstream job
- ClearCase Plugin — Integrates Jenkins with ClearCase.
- ClearCase UCM Baseline Plugin — Allows using ClearCase UCM baselines as the input of builds: When using this SCM, users will be asked at build-time to select the baseline on which the job has to work.
- ClearCase UCM Plugin — A Praqmatic integration to ClearCase UCM, simplifying continuous integration with Jenkins.
- Clone Workspace SCM Plugin — This plugin makes it possible to archive the workspace from builds of one project and reuse them as the SCM source for another project.
- CMVC Plugin — This plugin integrates CMVC to Hudson.
- Compuware Source Code Download for Endevor, PDS, and ISPW Plugin — The Compuware Source Code Download for Endevor, PDS, and ISPW plugin allows Jenkins users to download Endevor, PDS, or ISPW members from the mainframe to the PC.
- Config Rotator Plugin
- CVS Plugin — This bundled plugin integrates Jenkins with CVS version control system.
- Darcs Plugin — This plugin integrates Darcs version control system to Jenkins. The plugin requires the Darcs binary (darcs) to be installed on the target machine.
- Dimensions Plugin — This plugin integrates the Serena Dimensions CM SCM with Jenkins.
- File System SCM — Use File System as SCM.

The History of Jenkins

- Hudson was started in 2004 at Sun by Kohsuke Kawaguchi as a hobby project.
- First release in 2005.
- Kohsuke worked on Hudson full time in early 2008.
- Became the leading Continuous Integration solution with a market share of over 70% in 2010.
- Renamed to Jenkins in 2011.

Install Java on our local machine

Jenkins

LEVEL UP
www.level-up.one

Java Version Requirement

- Jenkins requires minimum Java 7 to be installed.
- It’s recommended to install Java 8.
- NOTE: Currently Java 9 is NOT supported for this course due to some Jenkins plugin incapability issues.

JAVA_HOME environment variable

- JAVA_HOME environment variable points to the installation path for the Java Development Kit.
- It is required by other applications which use Java.
- If you are using Mac, you can follow this lecture to configure JAVA_HOME.
- If you are using Windows or Linux, you can skip this lecture and follow the guide on the next text lecture.

Install Jenkins on our local machine

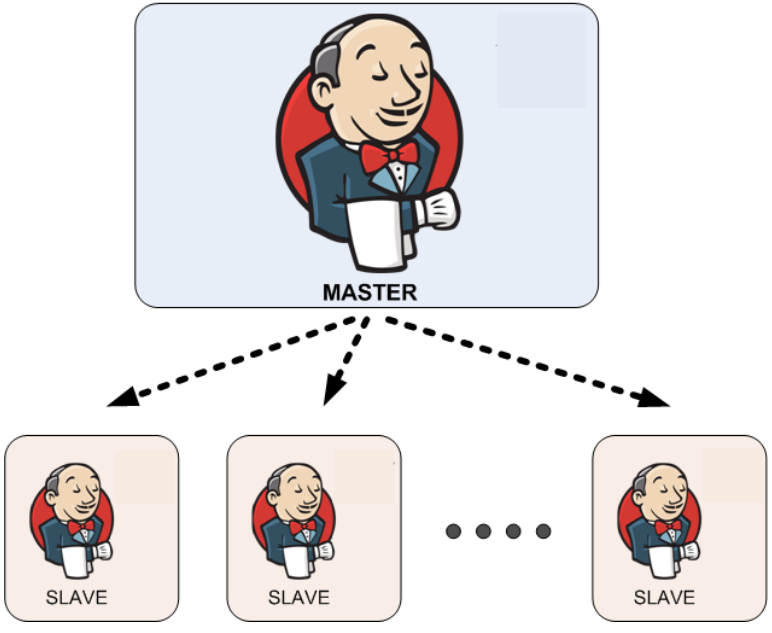
Jenkins

LEVEL UP
www.level-up.one

- Jenkins' Master and Slave Architecture
- Some Important Jenkins' Terminologies

Jenkins

Jenkins' Master and Slave Architecture



Master:

- Schedule build jobs.
- Dispatch builds to the slaves for the actual job execution.
- Monitor the slaves and record the build results.
- Can also execute build jobs directly.

Slave:

- Execute build jobs dispatched by the master.

Job / Project

- Those two terms are used interchangeably. They all refer to runnable tasks that are controlled / monitored by Jenkins.

Slave / Node

- Slaves are computers that are set up to build projects for a master.
- Jenkins runs a separate program called "slave agent" on slaves.
- When slaves are registered to a master, a master starts distributing loads to slaves.
- Node is used to refer to all machines that are part of Jenkins grid, slaves and master.

Executor

- Executor is a separate stream of builds to be run on a node in parallel.
- A Node can have one or more executors.

Build

- A build is a result of one of the projects.

Plugin

- A Plugin, like plugins on any other system, is a piece of software that extends the core functionality of the core Jenkins server.

Jenkins UI Overview

Jenkins

LEVEL UP
www.level-up.one

Create our first Jenkins Job

Jenkins

LEVEL UP
www.level-up.one

Run our first Jenkins job

Jenkins

LEVEL UP
www.level-up.one

Install GIT and GitHub plugin

Jenkins

LEVEL UP
www.level-up.one

Install and Configure Maven

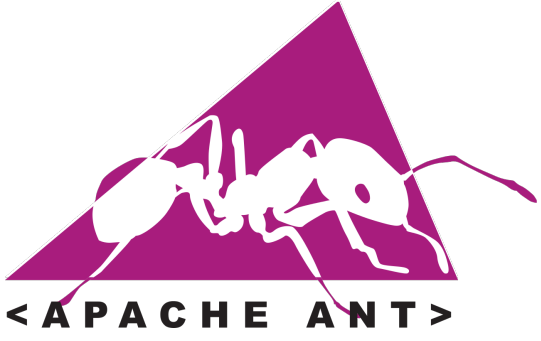
Jenkins

LEVEL UP
www.level-up.one

What does Maven do?

- Maven describes how the software is built.
- Maven describes the project's dependencies.

Java Build Tools



Add the bin directory of this new folder to the Path environment variable so that our operating system knows where to find our maven executable.

Configure Jenkins for our Maven-based project

Jenkins

LEVEL UP
www.level-up.one

Create our first Maven-based Jenkins project

Jenkins

LEVEL UP
www.level-up.one

Run our first Maven-based Jenkins project

Jenkins

LEVEL UP
www.level-up.one

Maven pom.xml file

- Describe the software project being built, including
  - The dependencies on other external modules.
  - The directory structures.
  - The required plugins.
  - The predefined targets for performing certain tasks such as compilation and packaging.

Different Phases in Maven Build Lifecycle

validate	Validate the project is correct and all necessary information is available.
compile	Compile the source code of the project.
test	Test the compiled source code using a suitable unit testing framework.
package	Take the compiled code and package it in its distributable format.
verify	Run any checks on results of integration tests to ensure quality criteria are met.
install	Install the package into the local repository, for use as a dependency in other projects locally.
deploy	Copy the final package to the remote repository for sharing with other developers and projects.

Maven Build Phases

- These lifecycle phases are executed sequentially to complete the default lifecycle.

- We want to specify the maven package command, this command would execute each default life cycle phase in order including validate, compile, test before executing package.

- We only need to call the last build phase to be executed.

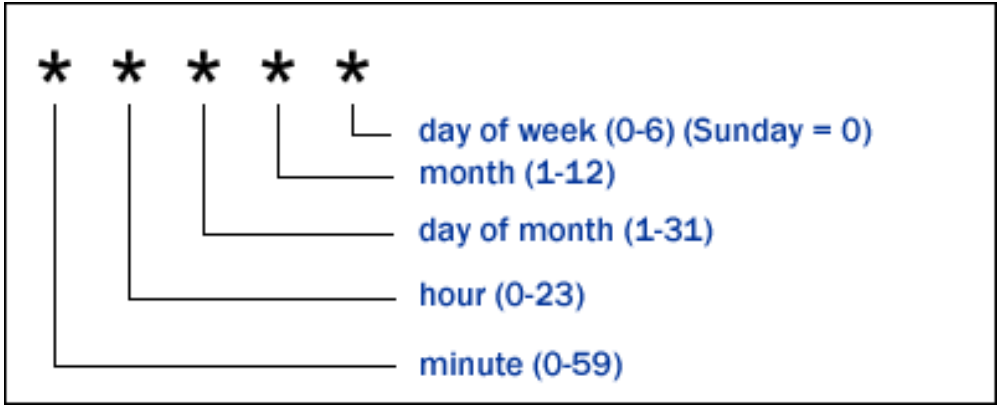
Configure Jenkins to poll source code changes periodically

Jenkins

LEVEL UP
www.level-up.one

Cron Syntax

- In Cron, each line consists of 5 fields separated by TAB or whitespace.



Cron Syntax

To specify multiple values for one field:

*****	all valid values
M-N	a range of values
A,B,Z	enumerates multiple values

Cron Syntax

Examples:

0 0 * * *	every day at midnight
0 2-4 * * *	2 am, 3 am and 4 am every day

Other Build Triggers of Jenkins

Jenkins

LEVEL UP
www.level-up.one

When to use scheduled build(build periodically option)?

- Very long running build jobs, where quick feedback is less critical.
- Some intensive load and performance tests which may take several hours to run.

Jenkins code quality metrics report

Jenkins

LEVEL UP
www.level-up.one

checkstyle

Last Published: 2016-10-30 | Version: 7.2

Overview

Checkstyle is a development tool to help programmers write Java code that adheres to a coding standard. It automates the process of checking Java code to spare humans of this boring (but important) task. This makes it ideal for projects that want to enforce a coding standard.

Checkstyle is highly configurable and can be made to support almost any coding standard. An example configuration files are supplied supporting the Sun Code Conventions, Google Java Style.

A good example of a report that can be produced using Checkstyle and Maven can be seen here.

Checkstyle is a code static analysis tool to help programmers to write Java code that adheres to a coding standard such as

- Avoiding multiple blank lines;
- Removing unused variables;
- Enforcing correct indentations;
- ...

Download

https://github.com/checkstyle/checkstyle

wnloaded from the SourceForge download page, or Maven central.

PMD Plugin

Edit Add Tools

2 Added by Ulli Hafner, last edited by Ulli Hafner on Jun 01, 2016 (view change)

Message from a Patron of Jenkins

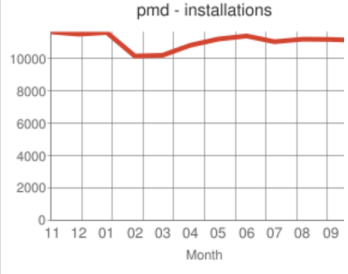
Jenkins

- Home
- Mailing lists
- Source code
- Bugtracker
- Security Advisories
- Events
- Donation
- Commercial Support
- Wiki Site Map

Documents

- Meet Jenkins
- Use Jenkins
- Extend Jenkins
- Plugins
- Servlet Container Notes

Plugin Information

Plugin ID	pmd
Latest Release **Latest Release Date** **Required Core** **Dependencies**	3.45 (archives) Jun 01, 2016 1.596.1 maven-plugin (version:2.9) matrix-project (version:1.2.1) token-macro (version:1.10, optional) analysis-core (version:1.77) dashboard-view (version:2.9.4, optional)
Usage	

Fork me on GitHub

PMD is a source code analyzer. It finds common programming flaws like unused variables, empty catch blocks, unnecessary object creation, and so forth. It supports Java, JavaScript, Salesforce.com Apex, PLSQL, Apache Velocity, XML, XSL.
Additionally it includes CPD, the copy-paste-detector. CPD finds duplicated code in Java, C, C++, C#, Groovy, PHP, Ruby, Fortran, JavaScript, PLSQL, Apache Velocity, Ruby, Scala, Objective C, Matlab, Python, Go, Swift and Salesforce.com Apex.

- Latest version(s)
- Get Involved
- Plugins
- Recent Announcements
- Next development version
- Previous versions

Latest version(s)

5.5.2 (5th November 2016)

- Release Notes
- Download (Sourcecode, Documentation)
- Online Documentation
- Requires at least java 7, Salesforce.com Apex requires at least java 8

5.4.3 (4th November 2016)

- Release Notes
- Download (Sourcecode, Documentation)
- Online Documentation
- Requires at least java 7

FindBugs Plugin

Edit Add Tools

12 Added by Ulli Hafner, last edited by Ulli Hafner on Jun 01, 2016 (view change)

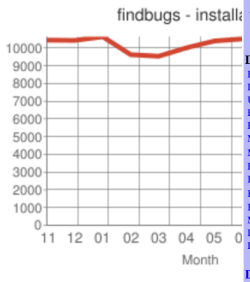
Jenkins

- Home
- Mailing lists
- Source code
- Bugtracker
- Security Advisories
- Events
- Donation
- Commercial Support
- Wiki Site Map

Documents

- Meet Jenkins
- Use Jenkins
- Extend Jenkins
- Plugins
- Servlet Container Notes

Plugin Information

Plugin ID	findbugs	**Changes**	In Latest Release Since Latest Release
Latest Release **Latest Release Date** **Required Core** **Dependencies**	4.65 (archives) Jun 01, 2016 1.596.1 matrix-project (version:1.2.1) analysis-core (version:1.77) maven-plugin (version:2.9) dashboard-view (version:2.9.4, op token-macro (version:1.10, option	**Source Code** **Issue Tracking** **Pull Requests**	GitHub Open Issues Pull Requests
Usage			

build passing

This plugin generates the trend report for FindBugs, an ope

FindBugs
because it's easy

Docs and Info
FindBugs 2.0
Demo and data
Users and supporters
FindBugs blog
Fact sheet
Manual
Manual(ja/日本語)
FAQ
Bug descriptions
Bug descriptions(ja/日本語)
Bug descriptions(fr)
Mailing lists
Documents and Publications
Links

Downloads

FindBugs Swag

Development
Open bugs
Reporting bugs
Contributing
Dev team
API [no frames]
Change log
SF project page
Browse source
Latest code changes

FindBugs™ - Find Bugs in Java Programs

This is the web page for FindBugs, a program which uses static analysis to look for bugs in Java code. It is free software, distributed under the terms of the Lesser GNU Public License. The name FindBugs™ and the FindBugs logo are trademarked by The University of Maryland. FindBugs has been downloaded more than a million times.

The current version of FindBugs is 3.0.1.

FindBugs requires JRE (or JDK) 1.7.0 or later to run. However, it can analyze programs compiled for any version of Java, from 1.0 to 1.8.

The current version of FindBugs is 3.0.1, released on 13:05:33 EST, 06 March, 2015. We are very interested in getting feedback on how to improve FindBugs. File bug reports on our sourceforge bug tracker

Changes | Talks | Papers | Sponsors | Support

FindBugs 3.0.1 Release

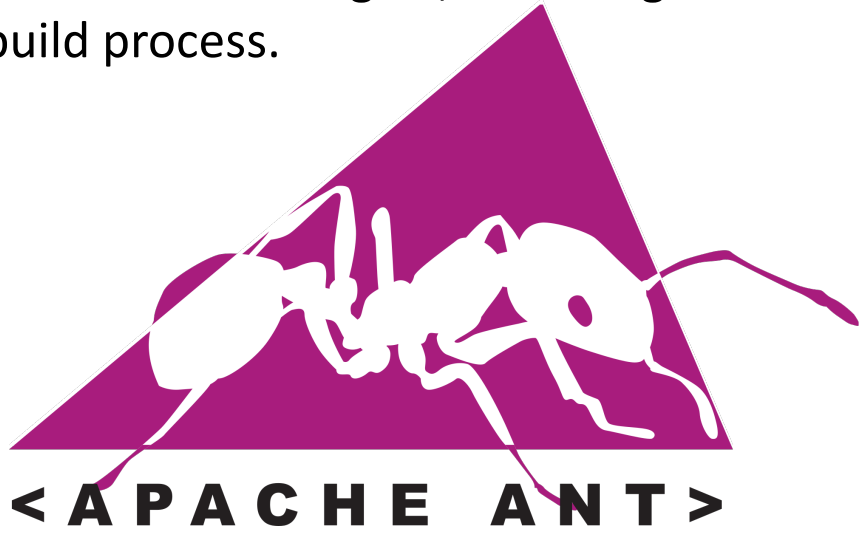
- A number of changes described in the changes document, including new bug patterns:
  - BSHIFT_WRONG_ADD_PRIORITY,
  - CO_COMPARETO_INCORRECT_FLOATING,
  - DC_PARTIALLY_CONSTRUCTED,
  - DM_BOXED_PRIMITIVE_FOR_COMPARE,
  - DM_INVALID_MIN_MAX,
  - ME_MUTABLE_ENUM_FIELD,
  - ME_ENUM_FIELD_SETTER,
  - MS_MUTABLE_COLLECTION,
  - MS_MUTABLE_COLLECTION_PKGPROTECT,
  - RANGE_ARRAY_INDEX,
  - RANGE_ARRAY_OFFSET,
  - RANGE_ARRAY_LENGTH,
  - RANGE_STRING_INDEX,
  - RV_RETURN_VALUE_IGNORED_NO_SIDE_EFFECT,
  - UC_USELESS_CONDITION,
  - UC_USELESS_CONDITION_TYPE,
  - UC_USELESS_OBJECT,
  - UC_USELESS_OBJECT_STACK,

Jenkins’ support for other build systems (Ant, Gradle and shell scripts)

Jenkins

Apache Ant

- Widely-used and very well-known build scripting language for Java.
- Flexible, extensible, relatively low-level scripting language.
- An Ant build script is made up of a number of targets, each target performs a particular job in the build process.



Gradle

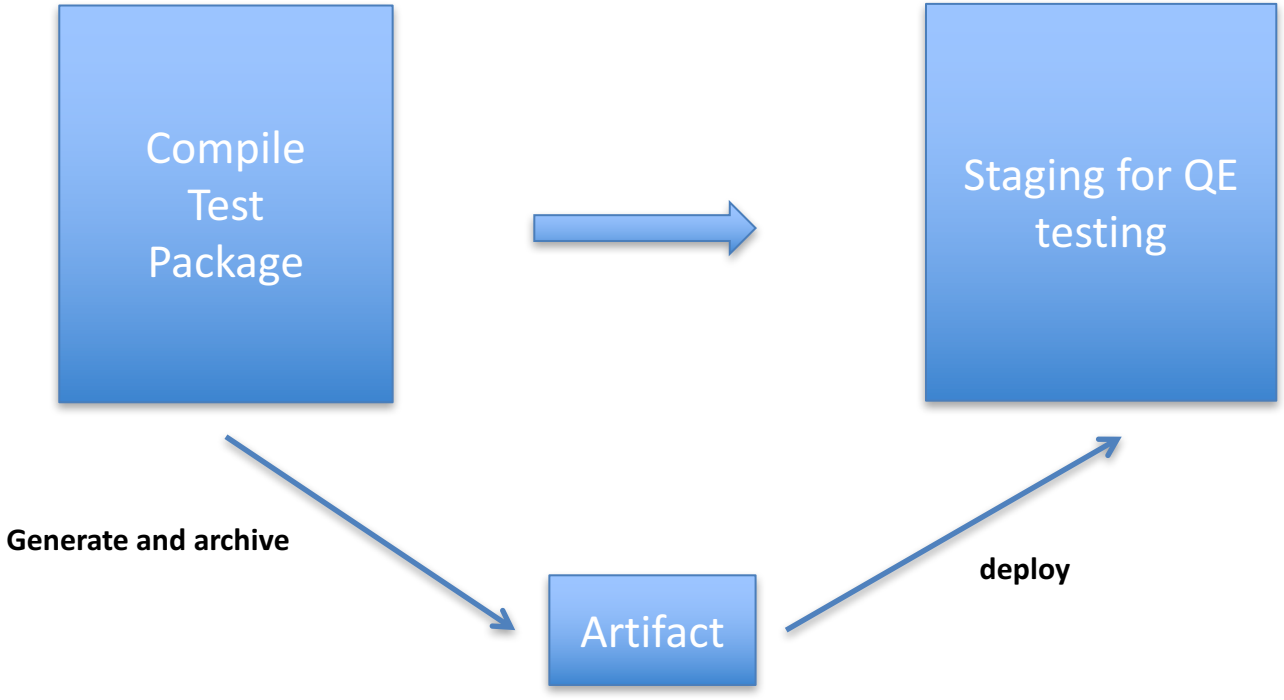
- Gradle is a relatively new open source build tool for the Java Virtual Machine.
- Build scripts for Gradle are written in a Domain Specific Language based on Groovy.
- The concise nature of Groovy scripting lets you write very expressive build scripts with very little code.

Archive generated artifacts

Jenkins

LEVEL UP
www.level-up.one

Continuous Integration Workflow



Install and configure Tomcat as a staging environment

Jenkins

LEVEL UP
www.level-up.one

Tomcat

Tomcat is an open-source web server and provides a "pure Java” HTTP web server environment in which Java code can run.

Apache Tomcat

Change Tomcat Server Port

- Jenkins runs on port 8080.
- The default port of Tomcat is also 8080.

- Install *copy artifact* and *deploy to container* plugins
- Deploy our application to staging environment

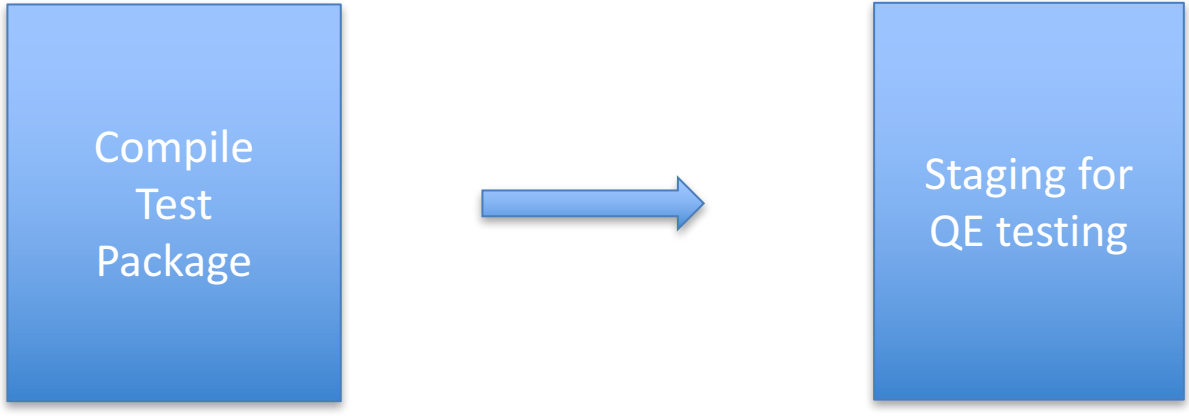
Jenkins

Jenkins Build Pipeline

Jenkins

LEVEL UP
www.level-up.one

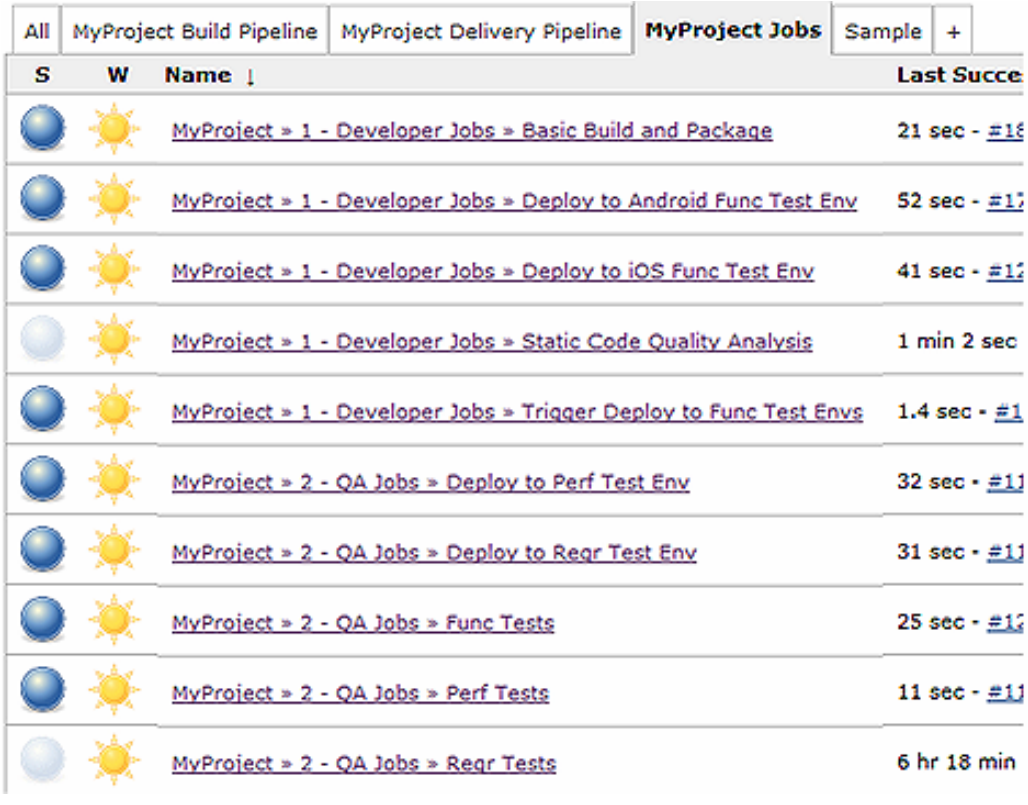
Our Current Build Pipeline



Upstream job

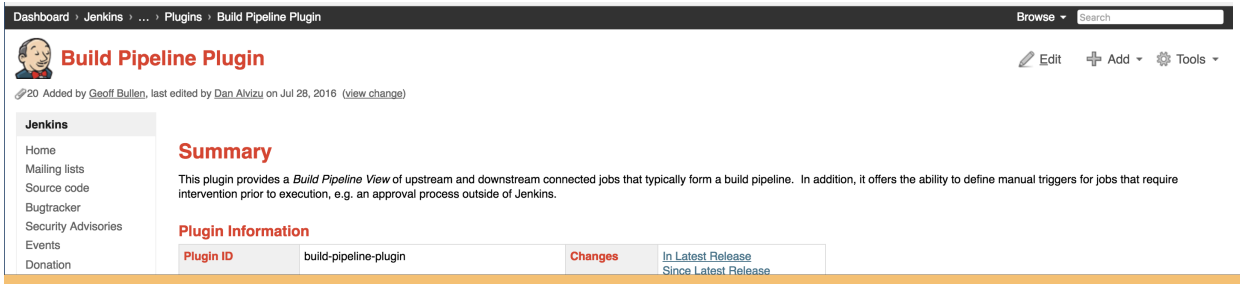
Downstream job

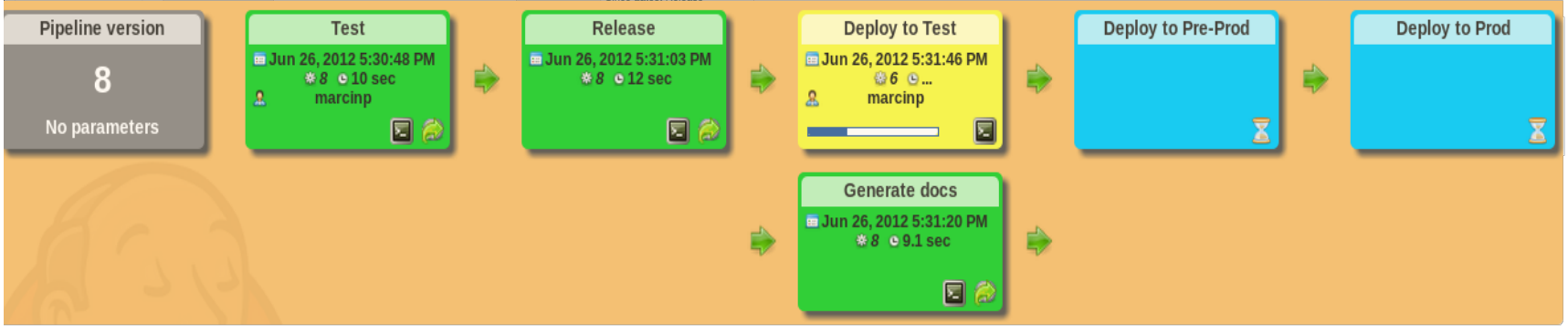
When the number of build jobs grows…



LEVEL UP
www.level-up.one

Build Pipeline Plugin





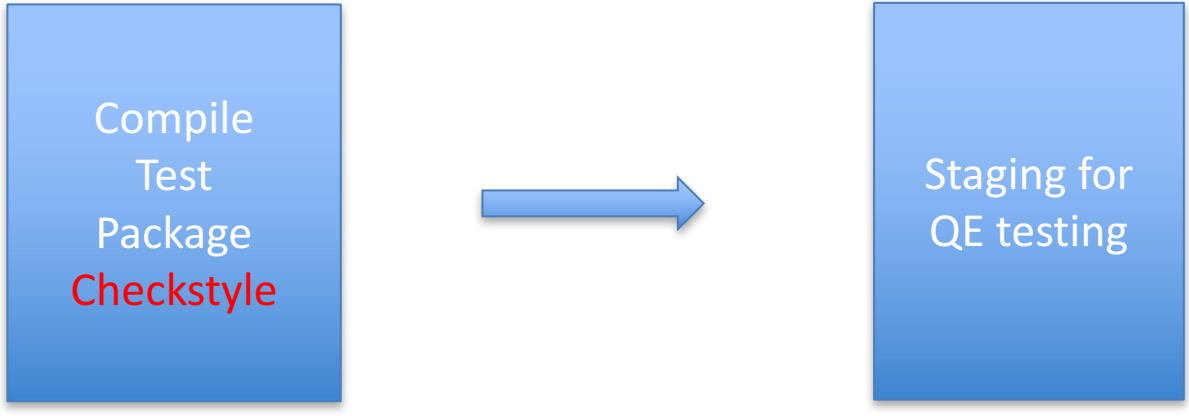
LEVEL UP
www.level-up.one

Parallel Jenkins Build

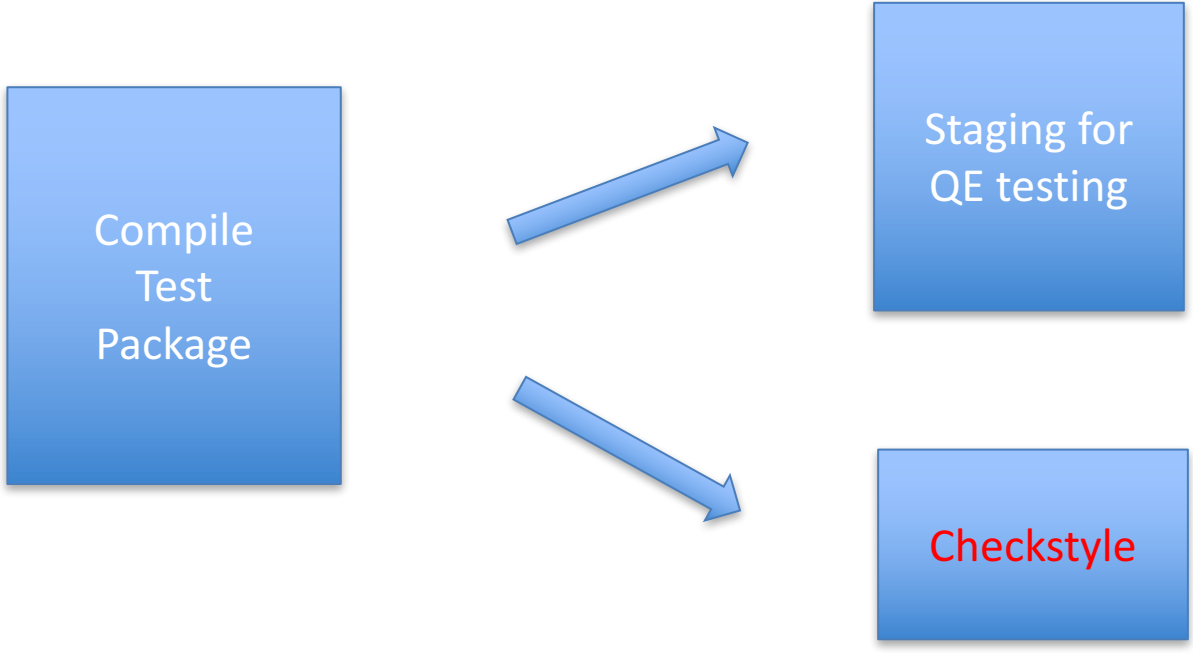
Jenkins

LEVEL UP
www.level-up.one

Linear Pipeline



Parallel Pipeline



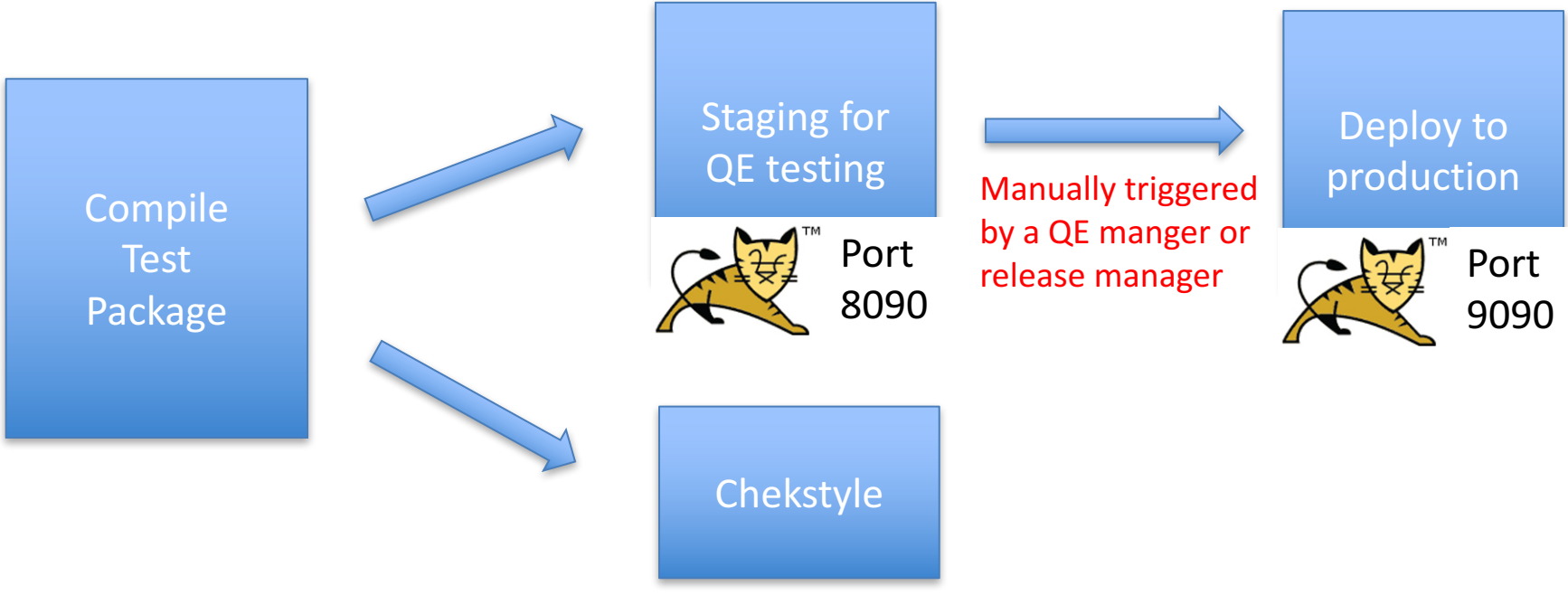
Continuous Delivery

Deploy our app to production

Jenkins

LEVEL UP
www.level-up.one

Full Continuous Delivery Pipeline



Pipeline as code

Carlos Rivas
Certified Cloud Architect| Linkedin.com/in/cjrivas

LEVEL UP
www.level-up.one

Benefits of a code-based pipeline

- Version control
- Best Practices
- Less error-prone execution of jobs
- Logic-based execution of steps

```
1
2   pipeline {
3       agent any
4       stages {
5           stage ('Initialize') {
6               steps {
7                   sh '''
8                       echo "PATH = ${PATH}"
9                       echo "M2_HOME = ${M2_HOME}"
10                  '''
11              }
12          }
13          stage ('Build') {
14              steps {
15                  echo 'Hello World!'
16              }
17          }
18      }
19  }
```

Pipeline as code

Automate our existing Jenkins pipeline

Pipeline as code

Taking our automation even further

Additional automation

- Setup Git repository polling
- Deployment to our tomcat servers
- We will setup a couple of tasks to run in parallel
- And we will briefly explain how to setup tomcat on EC2 in Amazon Web Services

Steps

- **Step 1:** Configure AWS security groups for Tomcat servers and create key pairs.
- **Step 2:** Provision EC2 instances on AWS to simulate staging and production environments.
- **Step 3:** Install and run Tomcat on created AWS instances.
- **Step 4:** Fully automate our existing Jenkins pipeline.

Steps

- **Step 1:** Configure AWS security groups for Tomcat servers and create key pairs.
- **Step 2:** Provision EC2 instances on AWS to simulate staging and production environments.
- **Step 3:** Install and run Tomcat on created AWS instances.
- **Step 4:** Fully automate our existing Jenkins pipeline.

Steps

- **Step 1:** Configure AWS security groups for Tomcat servers and create key pairs.
- **Step 2:** Provision EC2 instances on AWS to simulate staging and production environments.
- **Step 3:** Install and run Tomcat on created AWS instances.
- **Step 4:** Fully automate our existing Jenkins pipeline.

Steps

- **Step 1:** Configure AWS security groups for Tomcat servers and create key pairs.
- **Step 2:** Provision EC2 instances on AWS to simulate staging and production environments.
- **Step 3:** Install and run Tomcat on created AWS instances.
- **Step 4:** Fully automate our existing Jenkins pipeline.

Actions

- Next you will need to install two instances on AWS, one for staging and one for production environment. And install tomcats on them.
  - Those two tomcats can use the same 8080 port as they are running on different hosts.
  - There is NO need to configure the tomcat users and roles, as we won't use the deploy to container plugin anymore.

Local and Remote Servers

- scp to remote tomcat
- cp to local tomcat
- instead of remote ip address, use localhost

Introduction to Distributed Jenkins Builds

Jenkins

LEVEL UP
www.level-up.one

Digital Ocean is a cloud computing platform and virtual machine provider which is widely used in the industry.

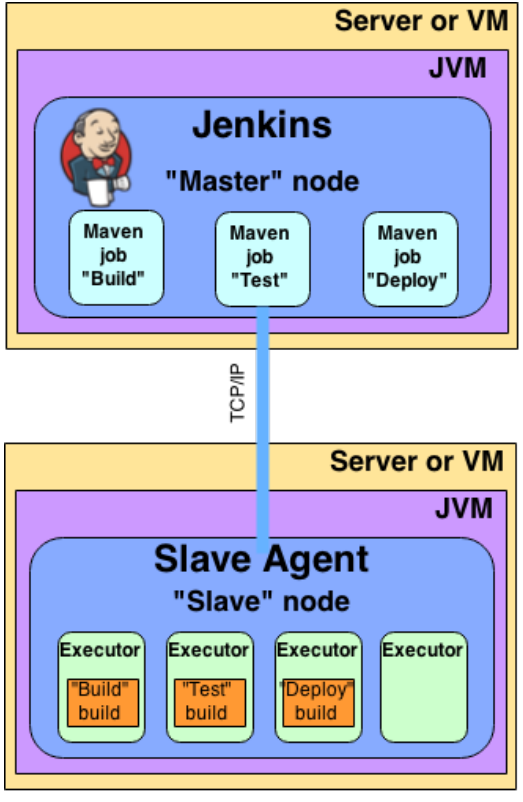
It is similar to AWS EC2 and Microsoft Azure, but much easier to get started with.

Install Jenkins Master in the Cloud

Jenkins

LEVEL UP
www.level-up.one

Jenkins Slave Agent



Different ways to start slave agent

- The master can start the slave agents via SSH.
- Start the slave agent manually using Java Web Start.
- Install the slave agent as a Window service.
- Start the slave agent directly from the command line on the slave machine.

Install Jenkins slaves in the cloud and form a Jenkins cluster

Jenkins

LEVEL UP
www.level-up.one

Start the slave agent

- Master node will start the slave agent on the slave machine via SSH .
- Automatic SSH login without password from master node to the slave node is needed.
- Master node will be running as a specific user called Jenkins to start the slave agent.

Executor

- A Jenkins executor is one of the basic building blocks which allow a build to run on a node.
- Think of an executor as a single “process ID”, or as the basic unit of resource that Jenkins executes on your machine to run a build.
- This number executors basically specifies the maximum number of concurrent builds that Jenkins may perform on this agent.
- A good value for the number of executors to start with would be the number of CPU cores on the machine.
- Setting a higher value would cause each build to take longer, but could increase the overall throughput.
- For example, one build might be CPU-bound, while a second build running at the same time might be I/O-bound — so the second build could take advantage of the spare I/O capacity at that moment.

Concurrent Builds on Jenkins Cluster

Label Jenkins Nodes

Jenkins

LEVEL UP
www.level-up.one

Label Nodes

- In real life scenarios, we might want a node to be reserved for certain kinds of jobs.
- For example, if you have jobs that run performance tests, you may want them to only run on a specially configured machine, while preventing all other jobs from using that machine.
- To do so, you would restrict where the test jobs may run by giving them a label expression matching that machine.