From Zero to CI/CD

Building your first CI Process

Nir Koren
DevOps CI/CD Team Leader, LivePerson Israel

My challenge / goals today

Each one of you will understand:

What is CI / CD

What is Git / GitHub and why most of the industry use it.

Same goes to Apache Maven

Same goes to Jenkins CI

Each one of you will have:

“hello-world” CI process on his/ her Laptop.

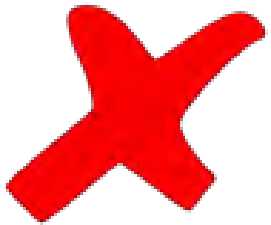
What will we cover today?

- Introduction
- Introduction to CI/CD
- **Introduction & hands-on** to Git and GitHub Basics
- **Introduction & hands-on** to Maven Basics
- **Introduction & hands-on** to Jenkins Basics
- Connect all together

What will we **NOT** cover today?

We are not going to develop anything today.

Not all of us comes from the same background and this not in the Workshop agenda.



Containerized applications: Docker / Kubernetes

Not enough time BUT it goes the same for those technologies as well.

Jenkins Pipelines and Shared libs

Not enough time but we will play with the pipeline in a nutshell.

Agenda

09:00 – 10:30: Introduction to CI / CD

10:30 – 11:00: Coffee Break 

11:00 – 12:30: Introduction to Git / GitHub

12:30 – 13:30: Lunch 

13:30 – 15:00: Introduction to Maven

15:00 – 15:30: Coffee Break 

15:30 – 17:00: Introduction to Jenkins & connect all

WE WILL HAVE TO BE FLEXIBLE

$ whoami

Nir Koren, DevOps CI/CD Team Leader, LivePerson Israel
Doing and implementing DevOps and CI / CD for 10+ years


https://il.linkedin.com/in/nirkoren


https://www.facebook.com/koren.nir


@KorenNir


@nir_koren

I work for LivePerson

100% Cloud SaaS company

Founded in 1995, Headquartered in NYC

Public since 2000 (NASDAQ: LPSN)

1000 Employees over 14 offices

19,000 customers, 27M interactions / Month

THE 2014 LEADERBOARD
OF JAVA TOOLS & TECHNOLOGIES

REBELLABS

82.5%
JUnit*
TOP TESTING FRAMEWORK USED BY DEVELOPERS

70%
Jenkins°
MOST USED CI SERVER IN THE INDUSTRY

64%
Maven
MOST USED BUILD TOOL IN JAVA

64%
Nexus°
THE MAIN REPOSITORY USED BY DEVELOPERS

69% Git*
#1 VERSION CONTROL TECHNOLOGY OUT THERE

67.5%
Hibernate*/°
THE TOP ORM FRAMEWORK USED

65% Java 7
THE INDUSTRY LEADER FOR SE DEVELOPMENT

56%
MongoDB*
THE NOSQL TECHNOLOGY OF CHOICE

55%
FindBugs*/°
MOST-USED STATIC CODE ANALYSIS TOOL

50%
Tomcat*
THE MOST POPULAR APPLICATION SERVER

49%
Java EE 6*
FOUND IN THE MOST ENTERPRISES

48%
Eclipse
THE IDE USED MORE THAN ANY OTHER

40% **Spring MVC*/°**
MOST COMMONLY USED WEB FRAMEWORK

32% **MySQL***
THE MOST POPULAR SQL TECHNOLOGY

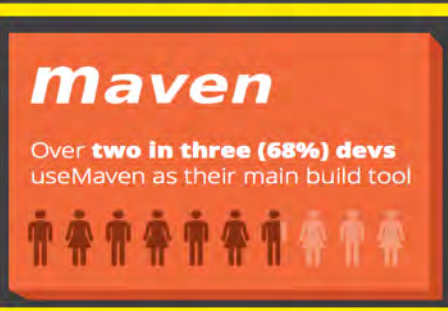
* *Multiple selections possible*

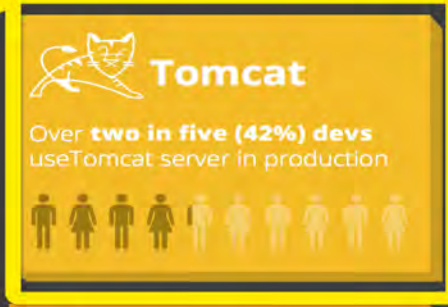
° *Normalized to exclude non-user base*

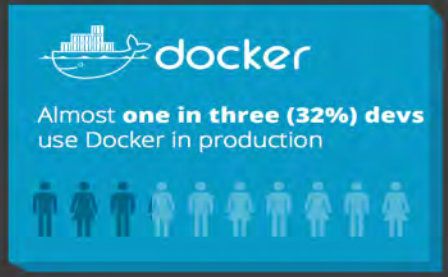
Sample population of 2164 Java professionals, sample error 2.1%

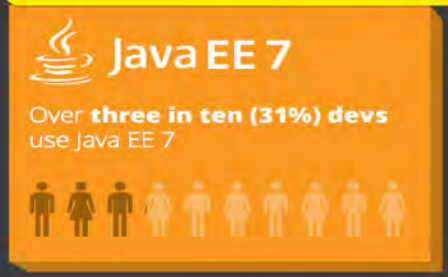
RebelLabs Tools and Technologies Leaderboard 2016

REBELLABS by ZEROTURNAROUND









Introduction to

CONTINUOUS INTEGRATION / DELIVERY

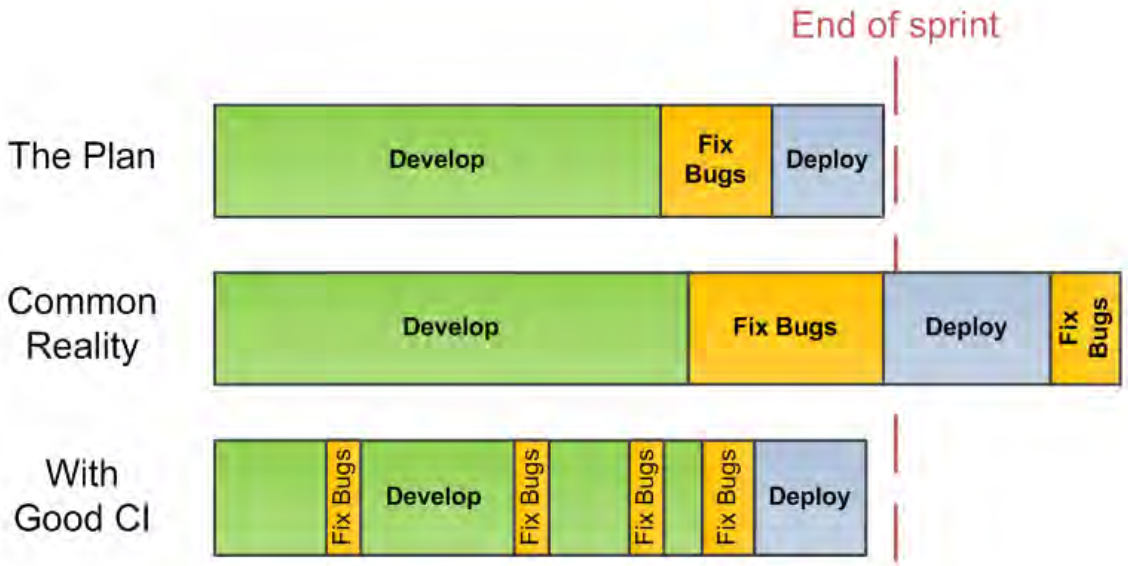
What is Continuous Integration?

"A key **software development practice** where members of a team **integrate their work frequently**."

Martin Fowler, Continuous Integration:
http://martinfowler.com/articles/continuousIntegration.html

What is Continuous Integration?

Continuous integration involves integrating early and often, in order to avoid "integration hell“



CI / CD Main Goal

Provides a **rapid feedback**.
If a defect is introduced into the code base, it can be **identified** and **corrected ASAP**.

Why CI/CD?

- Frequent changes ➔ Less integration problems.
- Bugs are detected earlier ➔ Saves money.
- Avoid last minute chaos.
- Transparency to all.
- Testing on Production-Like env.
- Easy to rollback in case of any issue.
- Enforce of automation culture.
- Enforce DevOps culture.
- Make the developers accountable and take ownership.

CI Principles

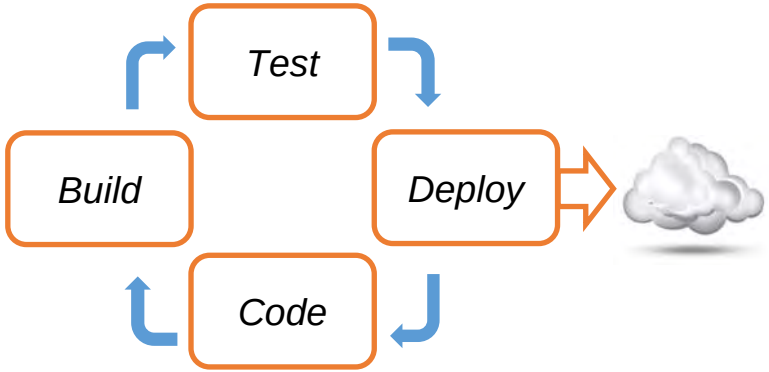
- **Automate** the build (single command), **Automate all**.
- Build is **self testable**.
- Baseline **branch is open consistently**.
- **Every commit** should be built.
- Build should be **fast**.
- Test Env is **clone of Production** (as much as we can).
- Everyone can see the status – **Transparency**.

What is

CONTINUOUS DELIVERY / DEPLOYMENT

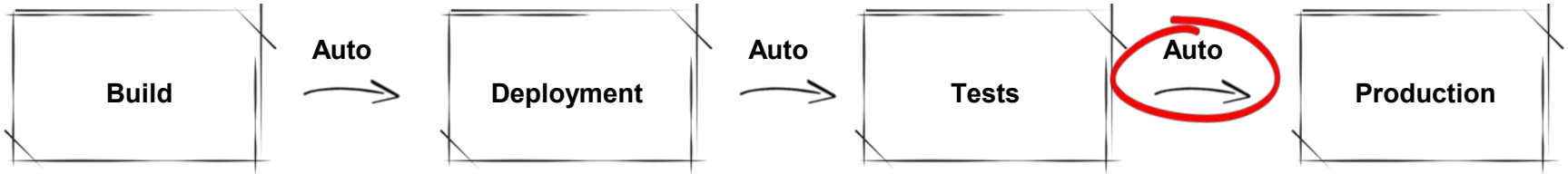
Continuous Delivery & Deployment

Continuous Delivery (CD) is a software development discipline where you build software in such a way that the software **can be released to production at any time**.

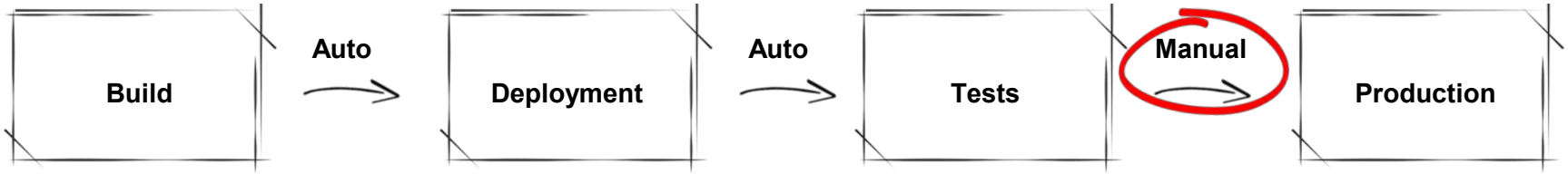


Continuous Deployment means that every change goes through the pipeline and automatically gets put into production, resulting in many production deployments every day.

Continuous **Deployment**



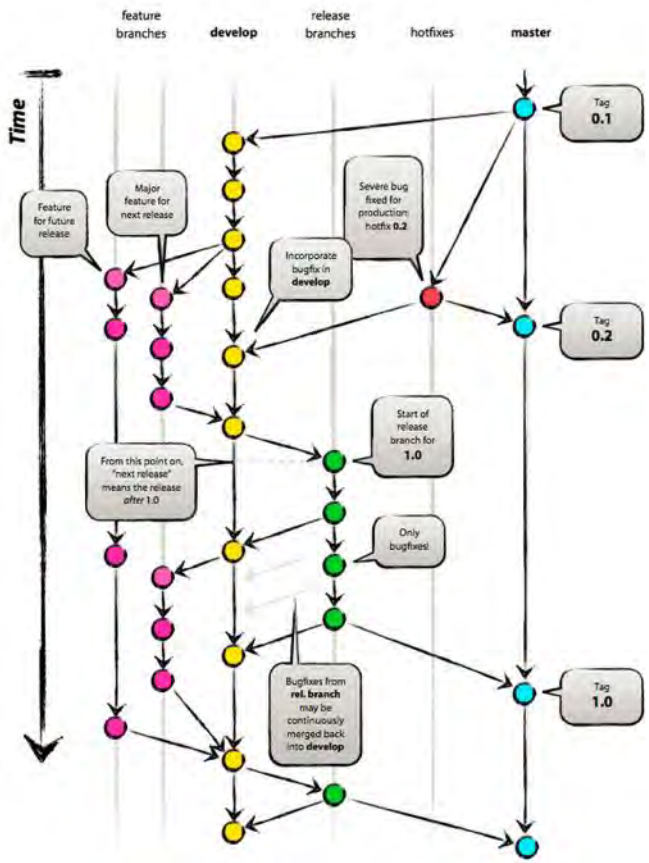
Continuous **Delivery**

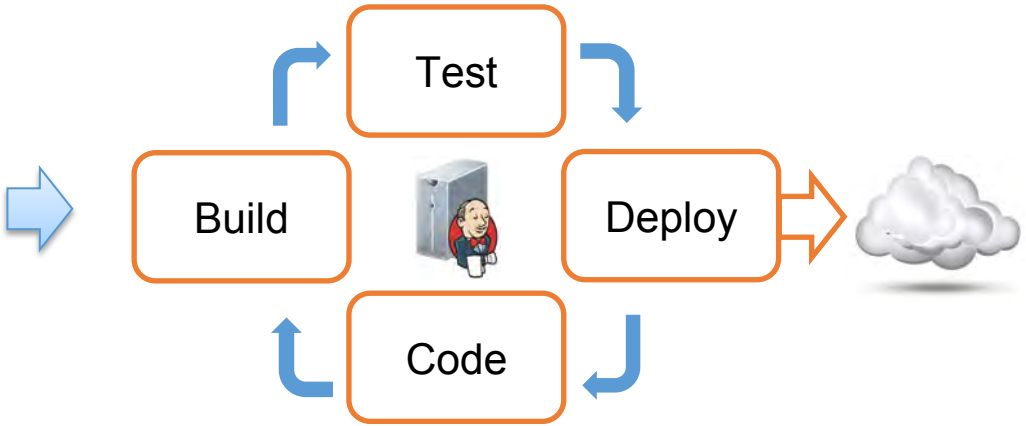


CD Principles

- The Software is always **deployable** through it's lifecycle.
- Anybody can get **fast and automated state** of Production.
- You can perform **push-button deployments** any time

CD Common structure



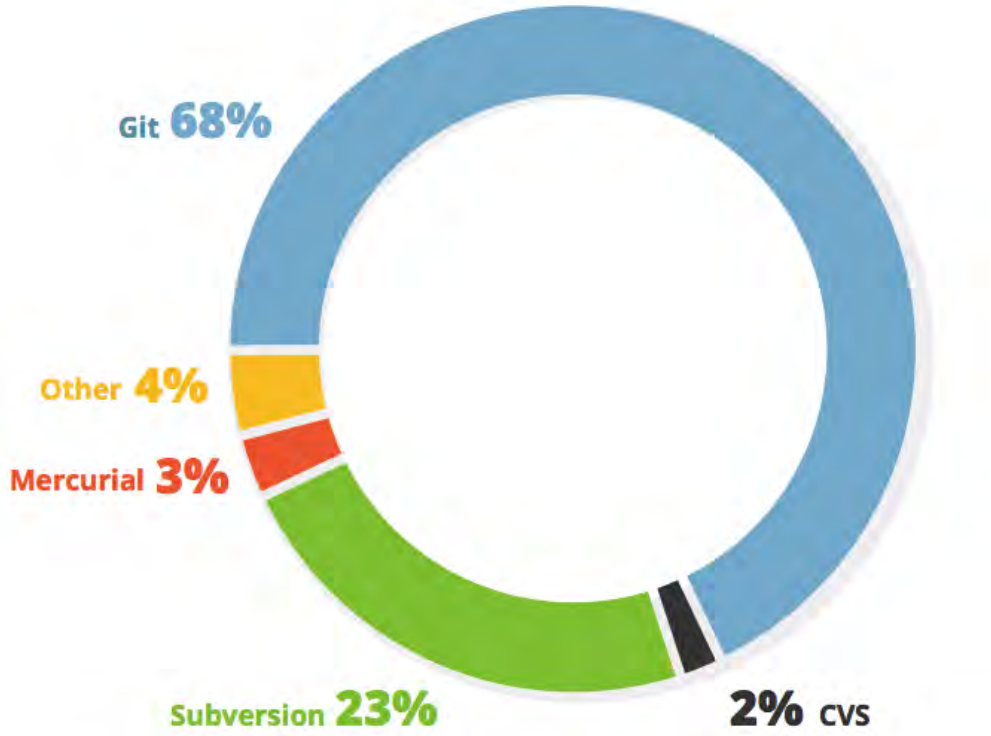


CD is not only in Computer Software

Tesla Model S gets firmware updates on regular basis for both UI and major elements (suspension, acceleration and more)

Introduction to
Git & GitHub

Figure 1.18 Most Commonly Used VCS



Lets install **git** & create **GitHub** account

Linux based: $ sudo apt-get install git

Windows: Download from https://git-scm.com/

Mac: $ brew install git Or download from https://git-scm.com/

Initial configurations:

$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com

Check your configurations:

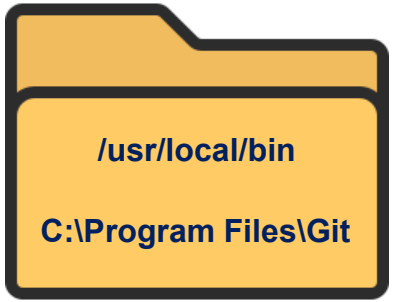
$ git config -l

https://github.com/

Sign Up

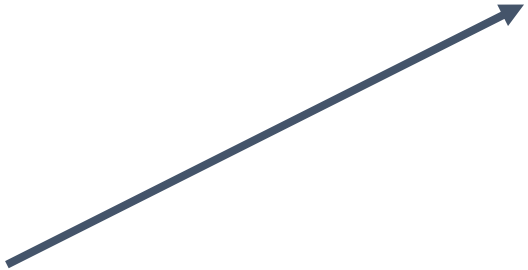
Git installation / configuration files.

INSTALLATION DIRECTORY



CONFIGURATION DIRECTORY

CODE REPOSITORY



```
[user]
        name = Nir Koren
        email = nirk@liveperson.com

[merge]
  tool = p4merge
…
```

What is Git?

Git is an **Open Source**, **Distributed** and **popular** version control system.
Designed for **Speed** and **Efficiency**

Open Source

- Free
- Popular
- Highly contributed by the community and GitHub

Popular

- Multi-platform support
- Multiple GIU’s
- Fully command lined – great for CI / Automation
- Contains any IDE / Integrated plugins

Fast

No Network needed.

- Performing a diff
- Viewing file history
- Committing changes
- Merging branches
- Obtaining any other revision of a file
- Switching branches

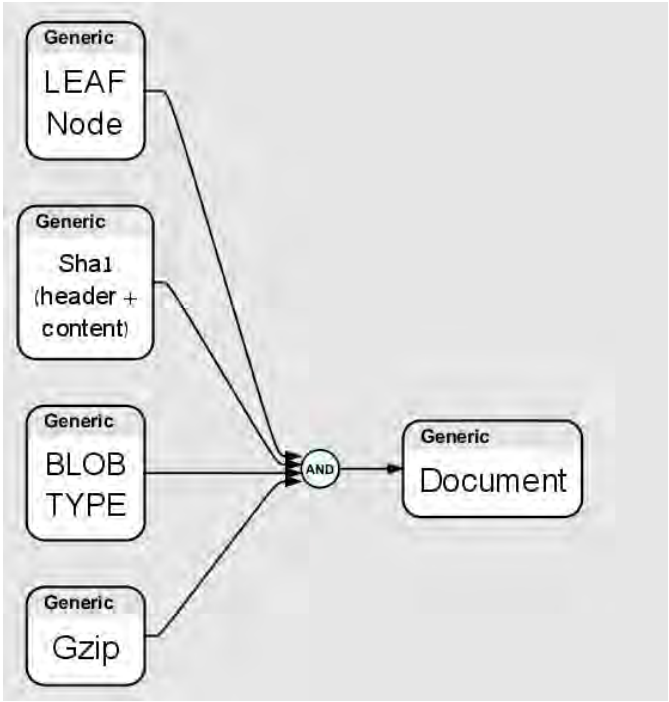
History & Founder

First release @ 2005 for Linux Kernel source code.
New free alternative for BitKeeper.

I'm an egotistical bastard, and I name all my projects after myself. First “Linux” and now “git”.

Linus Torvalds

Git: Under the hood



Key Concepts

COMMIT

A **Snapshot** in the repository
State of the project in a certain time. Point of a tree

BRANCH

A label with **user-friendly name**. Point to a **commit**.

TAG

A descriptive name to a **list of commits**.

Technically – it’s all the same

Key Concepts

REMOTE

Can be **origin** (where it cloned from) or **upstream** (custom)
The version that hosted in the server. Can be **remote host** or **local clone**.

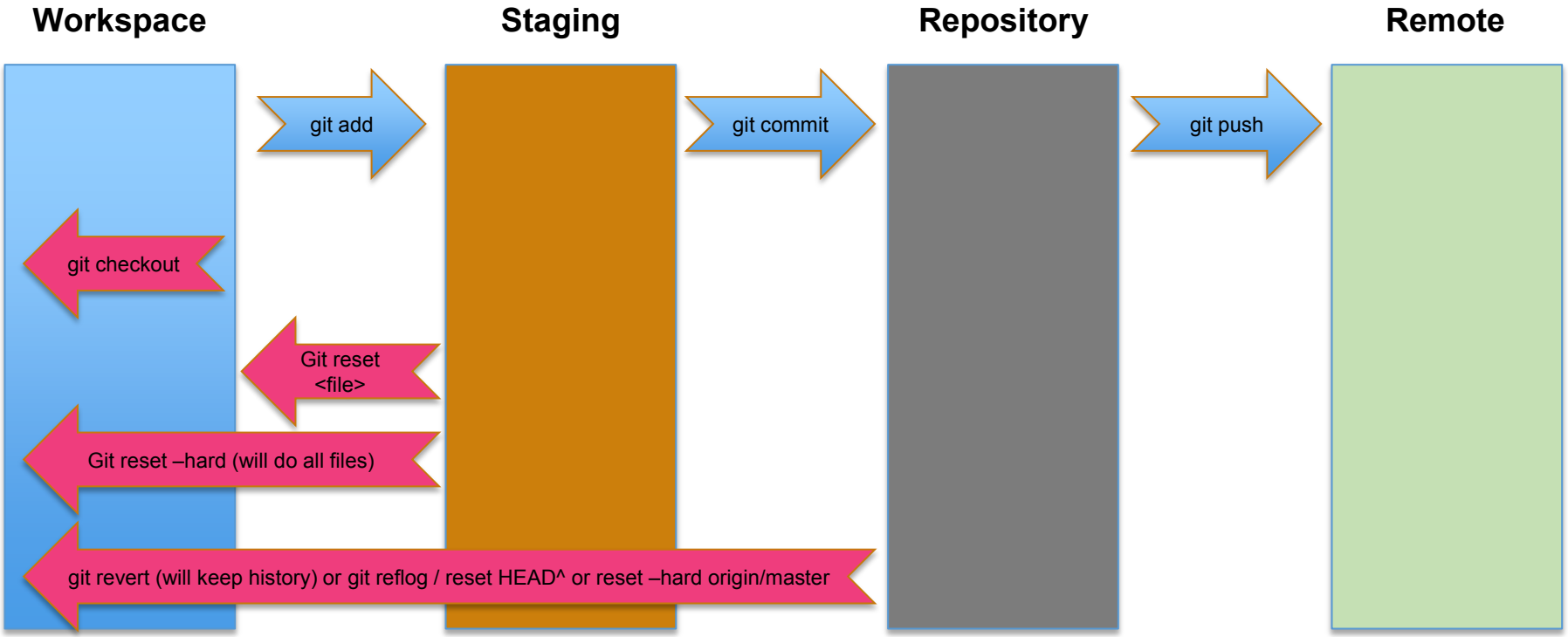
CLONE

Copy of a repository locally on your laptop. Comes with metadata relationship to the remote repo

FORK

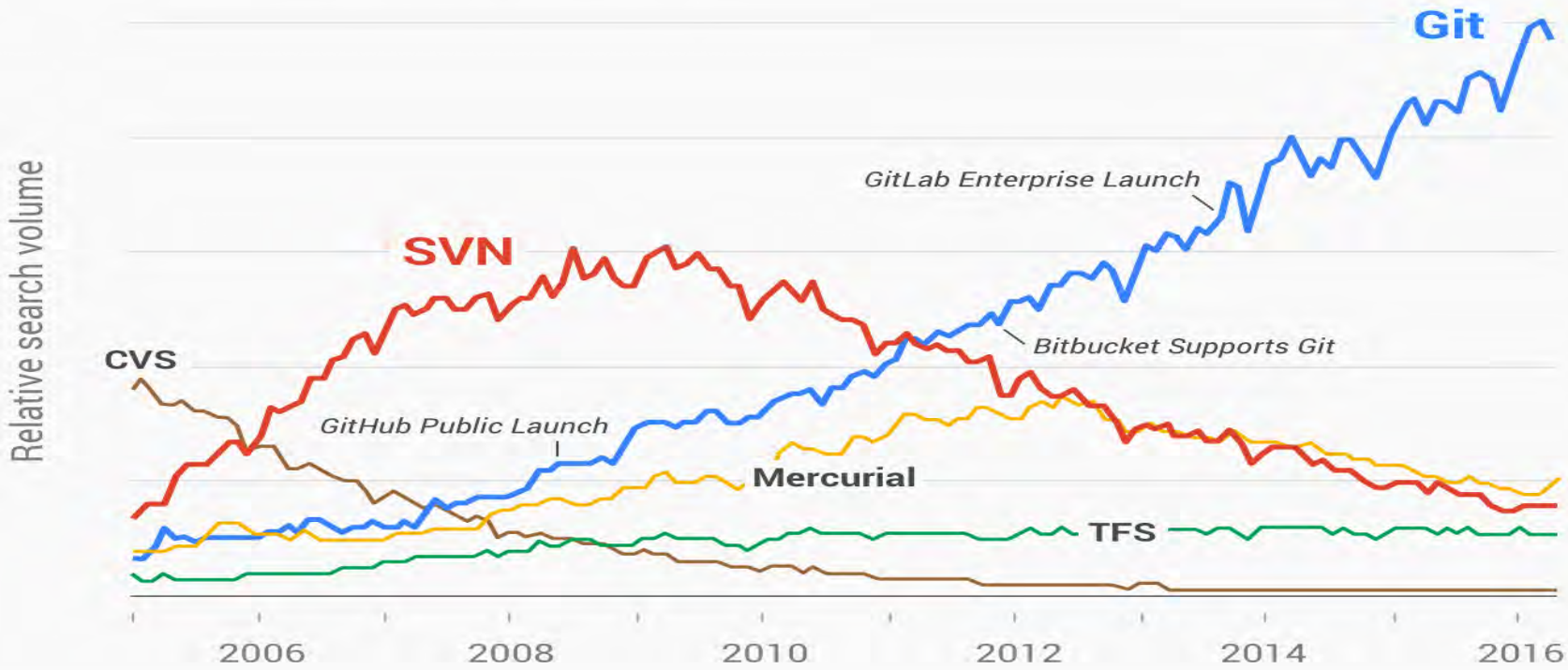
A **personal copy** of a repository that lives **remotely** on a user account and holds relationship with the original repo..

Git: Areas


Workspace
Staging
Repository
Remote
git add
git commit
git push
git checkout
Git reset
<file>
Git reset –hard (will do all files)
git revert (will keep history) or git reflog / reset HEAD^ or reset –hard origin/master

How Did SVN Manage to Lose?

Version control interest over time



GitPrime

Data: Google Trends, GitHub, Atlassian, GitLab

Perforce new products contains integration to Git

The best way to learn Git

FORGET Perforce
FORGET ClearCase
FORGET SVN
FORGET Any VCS you know

It’s much easier to teach a person who doesn’t know any VCS

Philosophy

- While **SVN trunk** is development, **GIT Master** is Release.
- Anything in **origin master is deployable**.
- Create a descriptively **named branch** for new feature development
- **Commit early and often** to new branch locally
- Ensure that **each commit represents one idea** or complete change.
- Push your new branch work to the **same named branch on origin**

Introduction to

What is GitHub?

Web-based hosting service for version control using Git.
The largest source code in the world

GitHub Company

- Founded in 2007.
- Headquartered in San Francisco, 800+ employees
- GitHub.com > 35M users, 100M Repositories
- Core GitHub developers are core Git OS contributors
- Cool brand (shop, merchandizing, crazy company)

Pricing models

- **Free:** your code is open and visible to all
- **Personal:** your code is private and you can share.
- **Business (hosted):** private in the cloud Github.com
- **Business (Enterprise):** On-premise. Install and maintain on your own.

Partial features list of

WHY GITHUB?

In-place editing using Ace Editor

devopscon / build-war / src / main / java / org / devopscon / maven / demo / **HelloHandler.java**

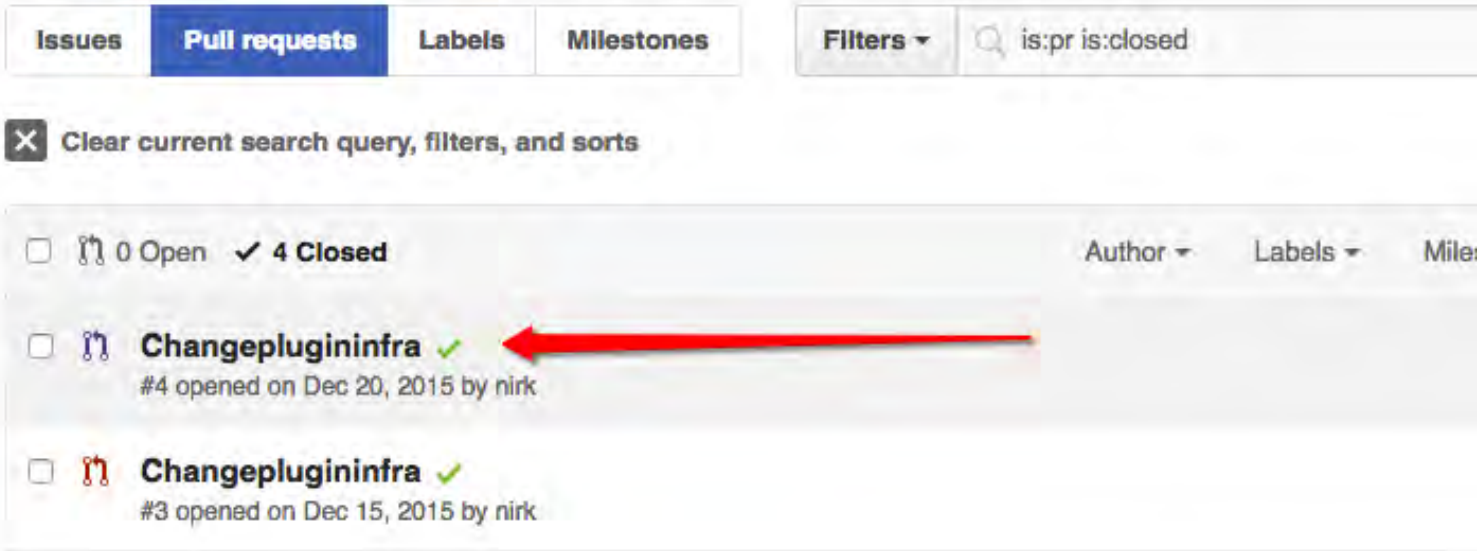
liveperson relocate build project — f71bccd 15 days ago

1 contributor

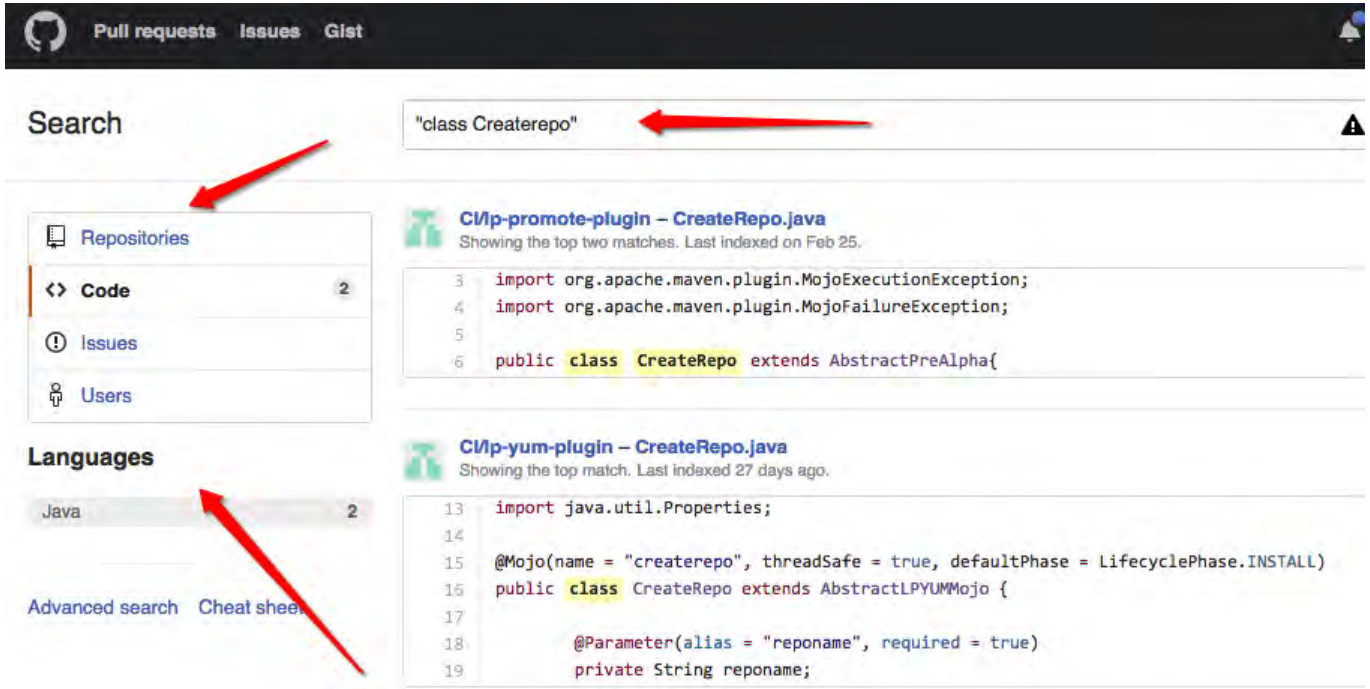
10 lines (7 sloc) | 182 Bytes — Raw | Blame | History

```
1  package org.devopscon.maven.demo;
2
3  public class HelloHandler {
4
5          public String sayHello() {
6                  String hello = "Hi there? What do you think about this WAR file?";
7                  return hello;
8          }
9
10 }
```

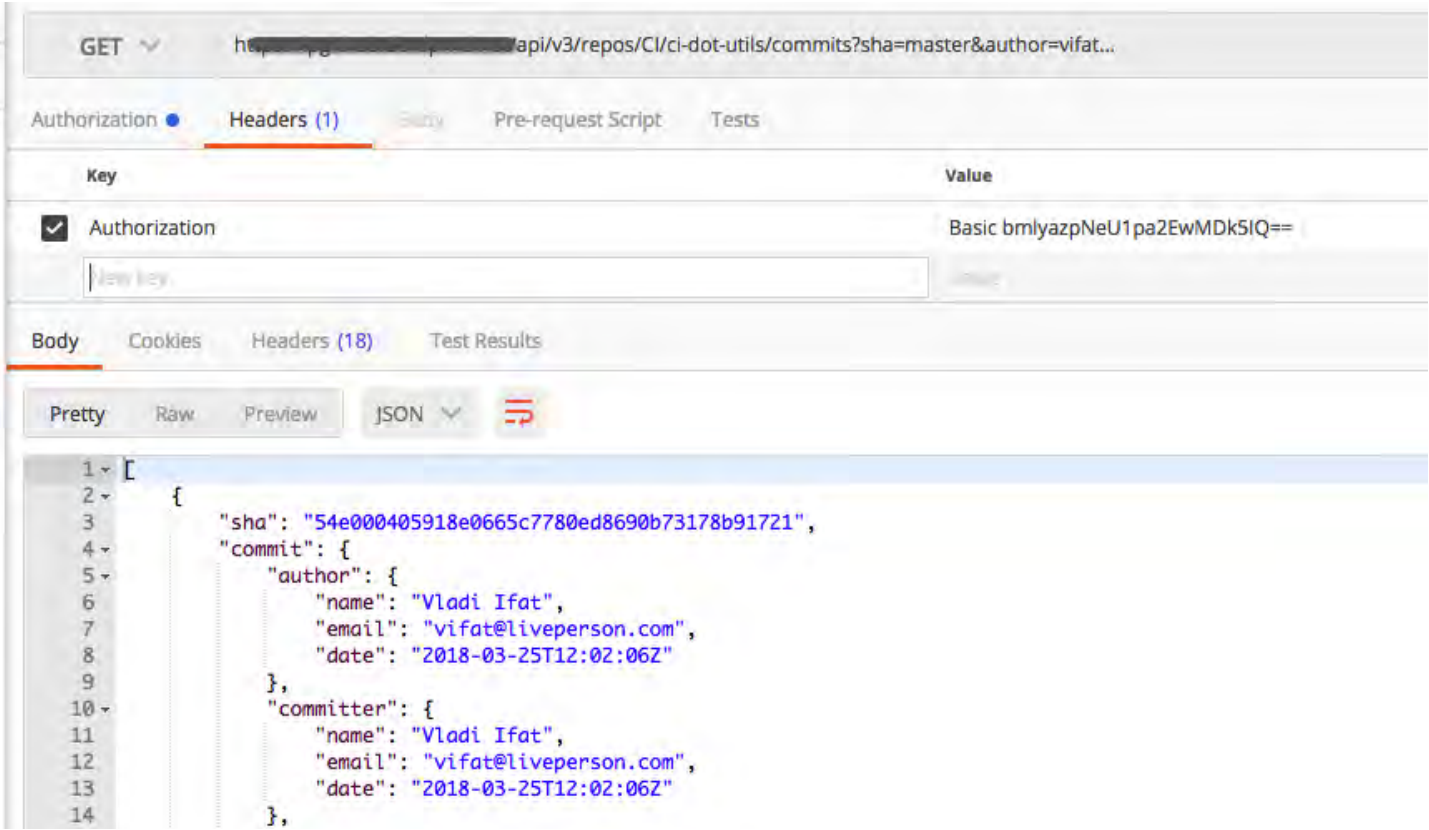
Both Machine and Human Code Review



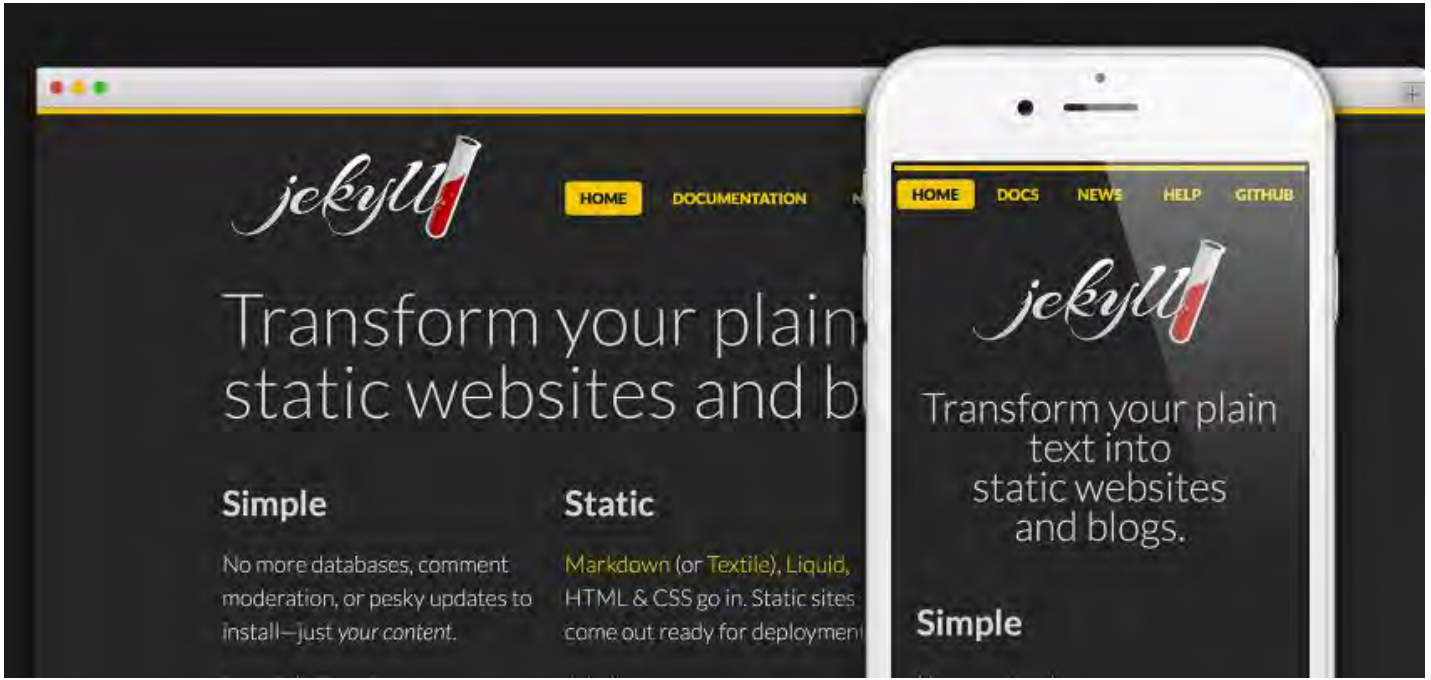
Code Search engine



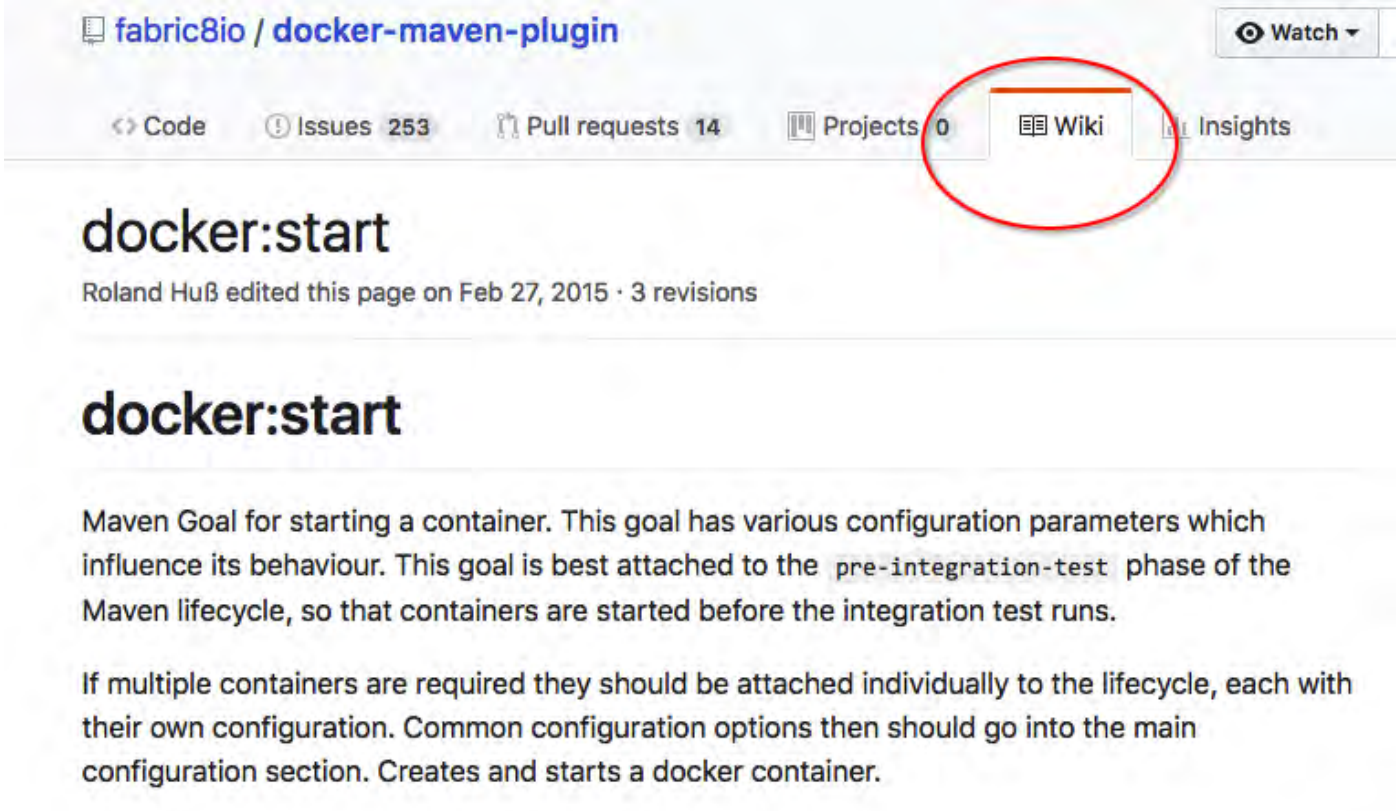
Rich documentation and API



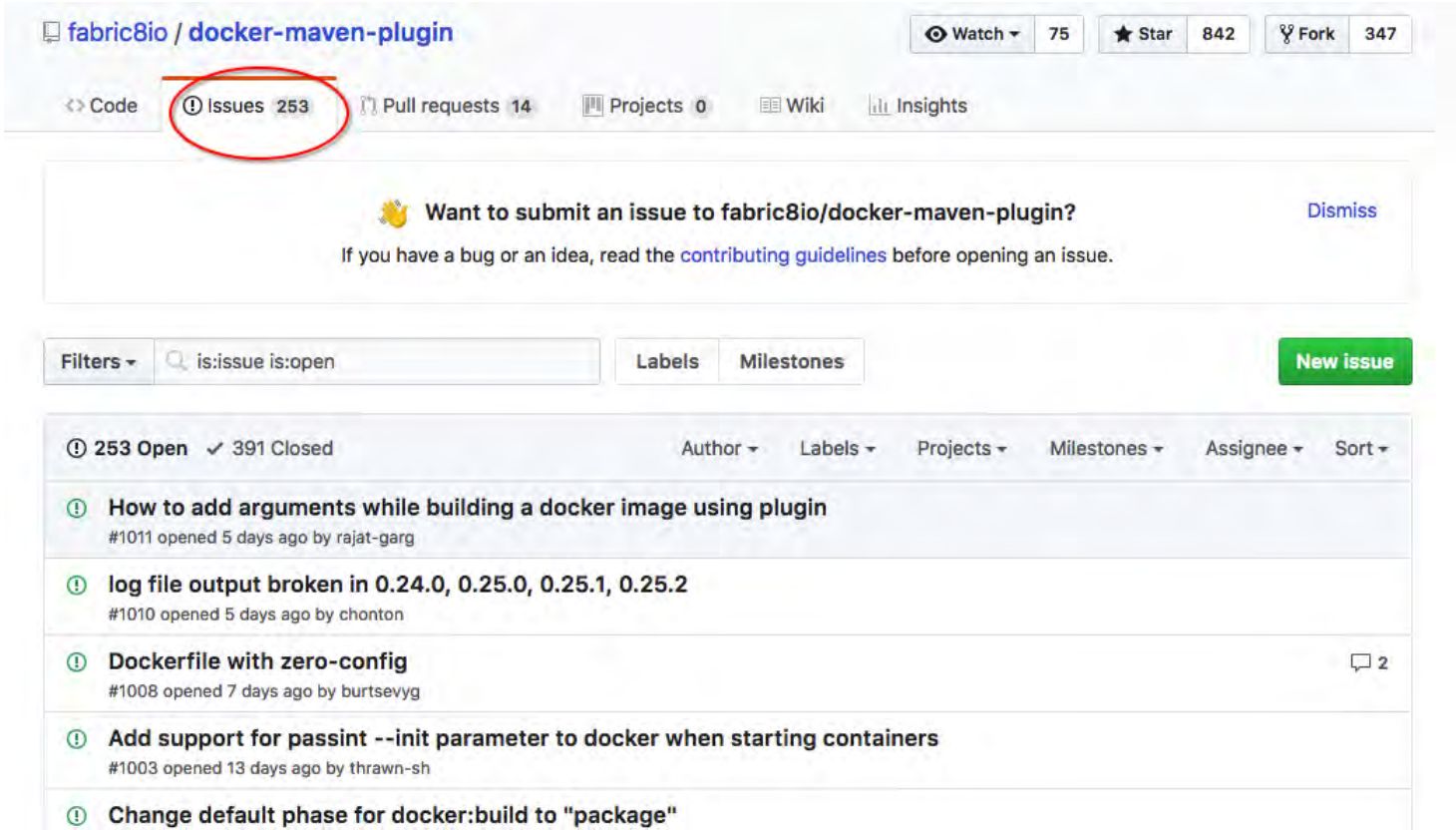
GitHub Pages



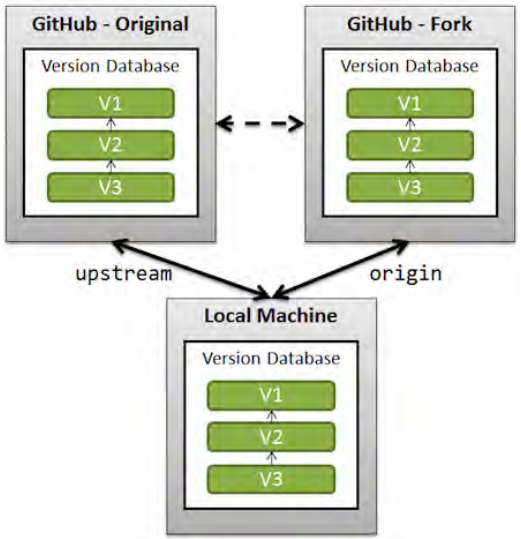
GitHub WIKI

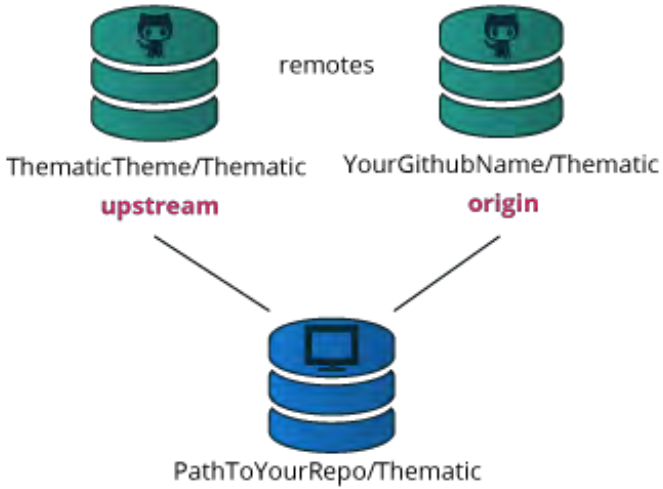

fabric8io / docker-maven-plugin
Watch
Code
Issues 253
Pull requests 14
Projects 0
Wiki
Insights
docker:start
Roland Huß edited this page on Feb 27, 2015 · 3 revisions
docker:start
Maven Goal for starting a container. This goal has various configuration parameters which influence its behaviour. This goal is best attached to the pre-integration-test phase of the Maven lifecycle, so that containers are started before the integration test runs.
If multiple containers are required they should be attached individually to the lifecycle, each with their own configuration. Common configuration options then should go into the main configuration section. Creates and starts a docker container.

GitHub Issues



Forks model – Open Source work model

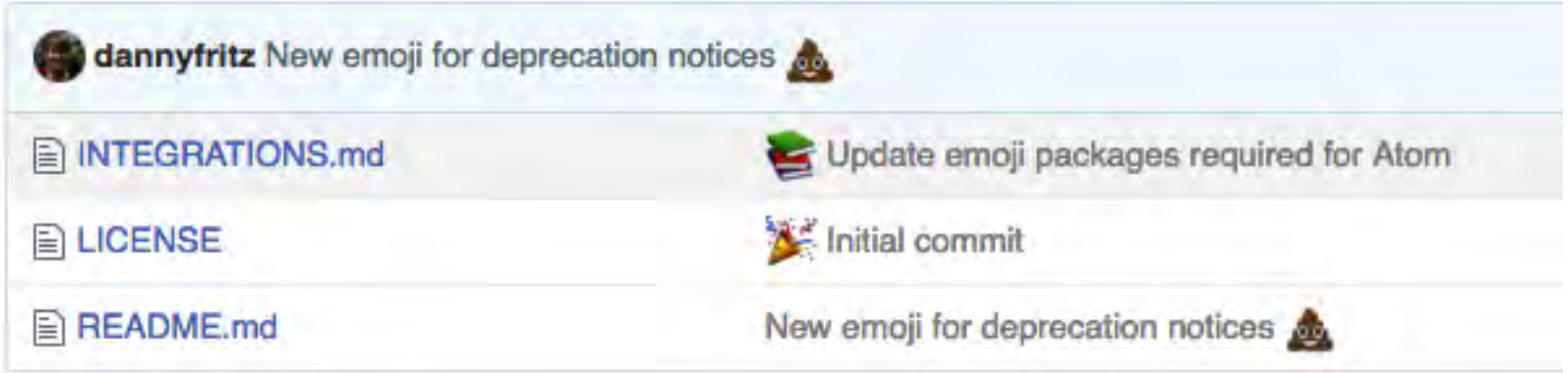




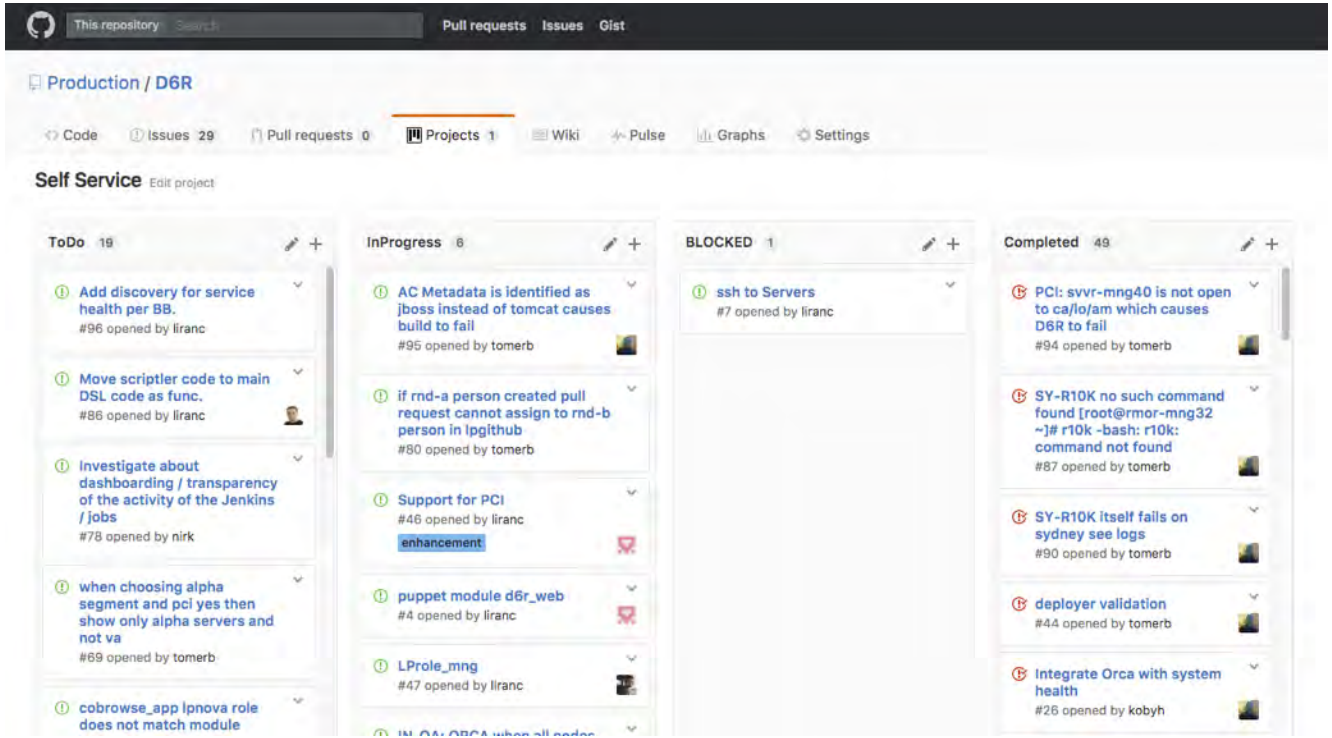
Pull Request

Tell others about your change, Let human / machine to review your code.

Communicate with Emojis



Manage your tasks in Board



Key Concepts

ORGANIZATION

Group of **2 or more users** that typically acts like a real org / company. It administrated and contains teams, users and repos.

REPOSITORY

Collection of files with metadata (.git folder) that contains history, revisions etc’

COLLABORATOR

A person who had read / write permissions to a repository.

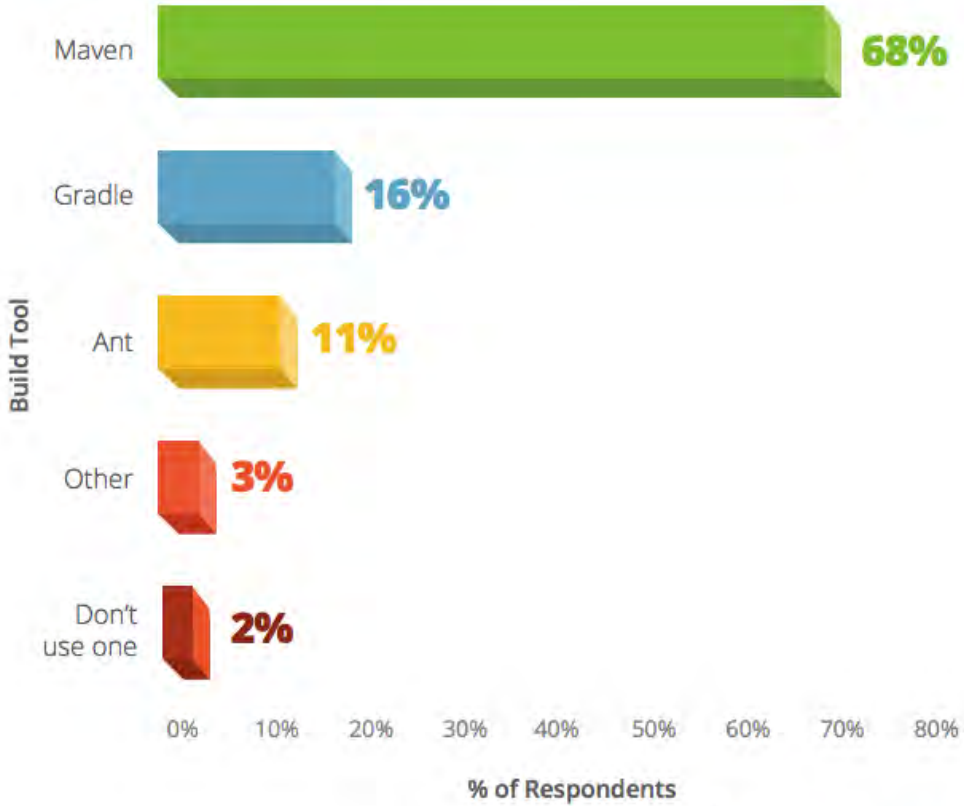
CONTRIBUTOR

A person who contributes to a repository but has no collaborator access. Only PULL requests.

Practice

Introduction to
Apache Maven

Figure 1.12 Battle of the build tools



Let’s install Maven first

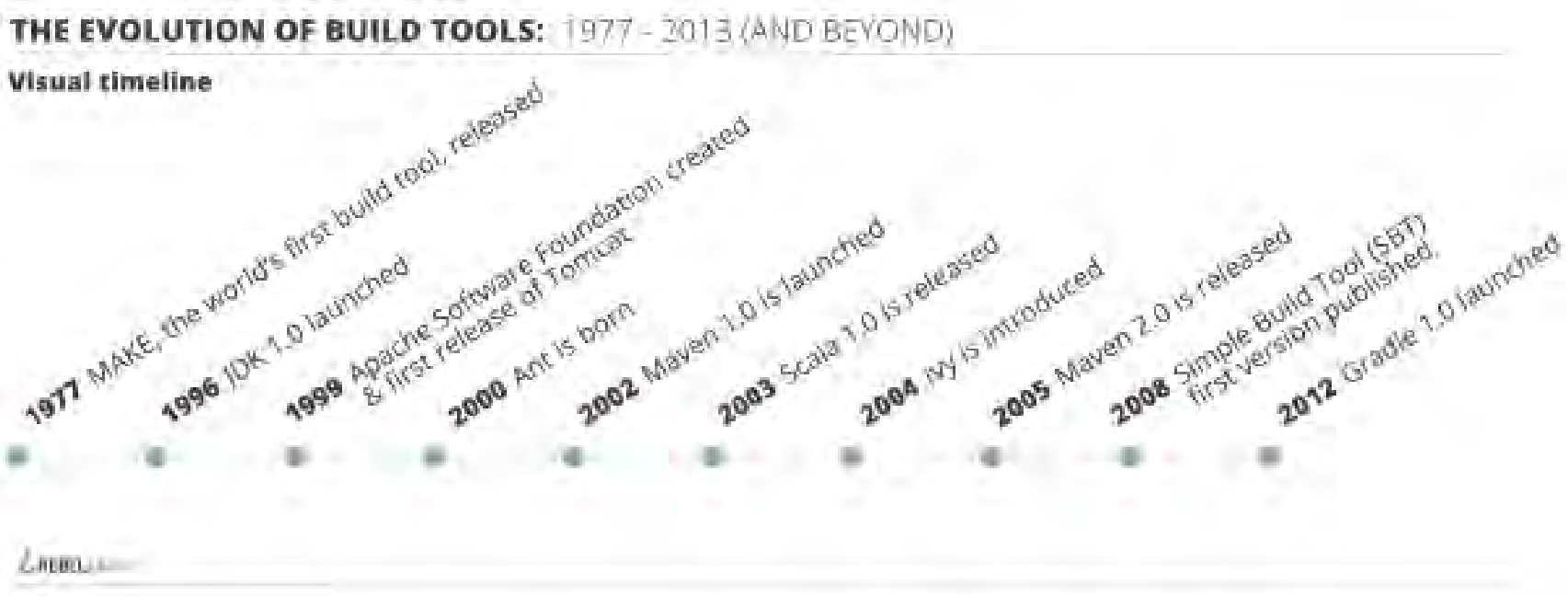
- Make sure you have JDK8
- Ensure JAVA_HOME env var is pointed to the JDK8
- **Download Maven** https://maven.apache.org/download.cgi
- Place (unzip / extract) the Maven somewhere and add the <MAVEN>/bin to your env var PATH
- Run “mvn -v” to verify the installation

What is Apache Maven?

- In Yiddish: accumulator of knowledge
- Releases **1.x**: 2002, **2.x**: 2005, **3.x** (Current): 2009
- Successor of Apache ANT (*)
- A standard way to build projects (Java mainly).
- A Framework (open & extendable)

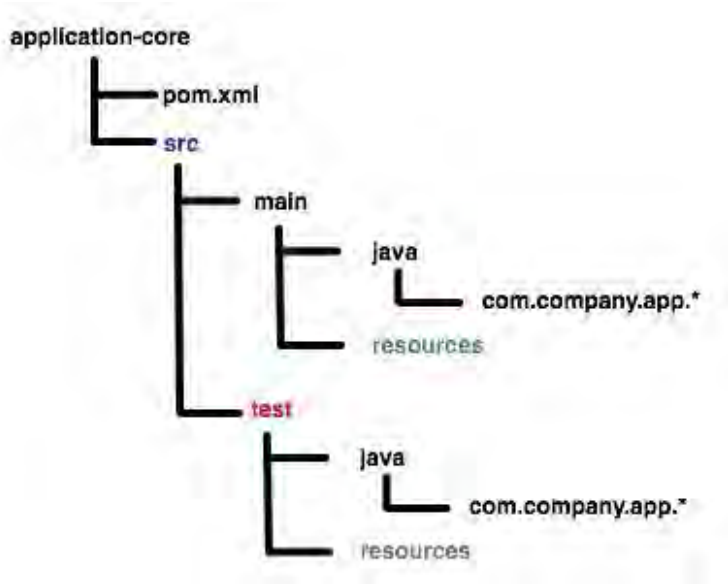
What is “Build”?

History of build tools



Maven directories structure

src/main/java	Application sources
src/main/resources	Application resources
src/main/webapp	Web application sources
src/test/java	Test sources
src/test/resources	Test resources



Maven Architecture

Project: src/main/java…

Local Repository
(~/.m2/repository)

Install

Maven

Deploy

Nexus

JFrog Artifactory

Proxy

Maven Configuration

General Configuration: ${maven.home}/conf/settings.xml
User Configuration: ${user.home}/.m2/settings.xml

```
<settings>
<localRepository/>
<interactiveMode/>
<offline/>
<pluginGroups/>
<servers/>
<mirrors/>
<proxies/>
<profiles/>
<activeProfiles/>
</settings>
```

Maven it’s just a
core framework
for **Maven Plugins**

Maven Plugins

```
<build>
    <plugins>
        <plugin>
            <groupId>com.someorg.somegroup</groupId>
            <artifactId>some-name</artifactId>
            <version>1.0.0.0</version>
            <executions>
                <execution>
                <id>id</id>
                <phase>package</phase>
                    <goals>
                        <goal>any_goal</goal>
                    </goals>
                </execution>
            </executions>
            <configuration>
                <somepropery>xxx</somepropery>
            </configuration>
        </plugin>
    </plugins>
</build>
```

$ mvn com.someorg.somegroup:some-name:1.0.0.0:any_goal –Dsomeproperty=xxx

Maven Built-in Plugins (partial List)

clean
Clean up after the build.

compiler
Compiles Java sources.

deploy
Deploy the built artifact to the remote repository.

install
Install the built artifact into the local repository.

resources
Copy the resources to the output directory for including in the JAR.

surefire
Run the JUnit unit tests in an isolated classloader.

Maven Lifecycle (partial list)

validate	validate the project is correct and all necessary information is available.
compile	validate the project is correct and all necessary information is available.
process-resources	copy and process the resources before build
test	run tests using a suitable unit testing framework
package	Take the compiled code and package it in its distributable format
install	Package + put it in the local repo
deploy	Install + upload to the artifact repository

Maven Versions

General Structure (In LP we use 4 digits):

1.0.0.0-SNAPSHOT

Major Version – Main content of a release
Minor Version – Enhancements (Service Packs)
Micro version – Bug fixes
HotFix version – Emergency fixes
Latest version – Latest version we have

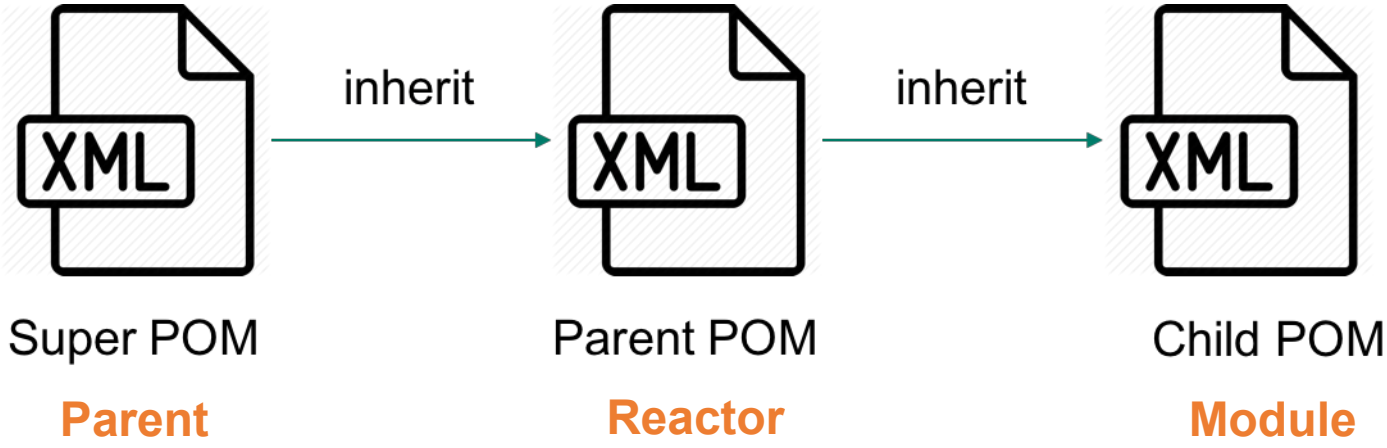
Maven SNAPSHOT Update Policy

How often should we check for SNAPSHOT update?
Can be configured in the **settings.xml** By default is “daily”.

- “always”
- “daily”
- “interval:XXX” (in minutes)
- “never”

- Can be done always also via CMD with –U flag

POM


XML
inherit
XML
inherit
XML
Super POM
Parent POM
Child POM
Parent
Reactor
Module

Maven POM Structure

```
<project>
        <modelVersion>4.0.0</modelVersion>
        <!-- The Basics -->
        <groupId>...</groupId>
        <artifactId>...</artifactId>
        <version>...</version>
        <packaging>...</packaging>
        <dependencies>...</dependencies>
        <parent>...</parent>
        <dependencyManagement>...</dependencyManagement>
        <modules>...</modules>
        <properties>...</properties>
        <!-- Build Settings -->
        <build>...</build>
        <profiles>...</profiles>
</project>
```

Maven POM structure

General artifact information

```
<groupId>com.company.something</groupId>
<artifactId>name-of-artifact</artifactId>
<version>xxx-SNAPSHOT</version>
<packaging>jar</packaging>
```

Maven POM structure

Dependencies

```
<dependency>
        <groupId>io.fabric8</groupId>
        <artifactId>kubernetes-api</artifactId>
        <version>xxx</version>
        <scope>${scope}</scope>
</dependency>
```

Scopes: compile (default), provided, runtime, test (partial list)

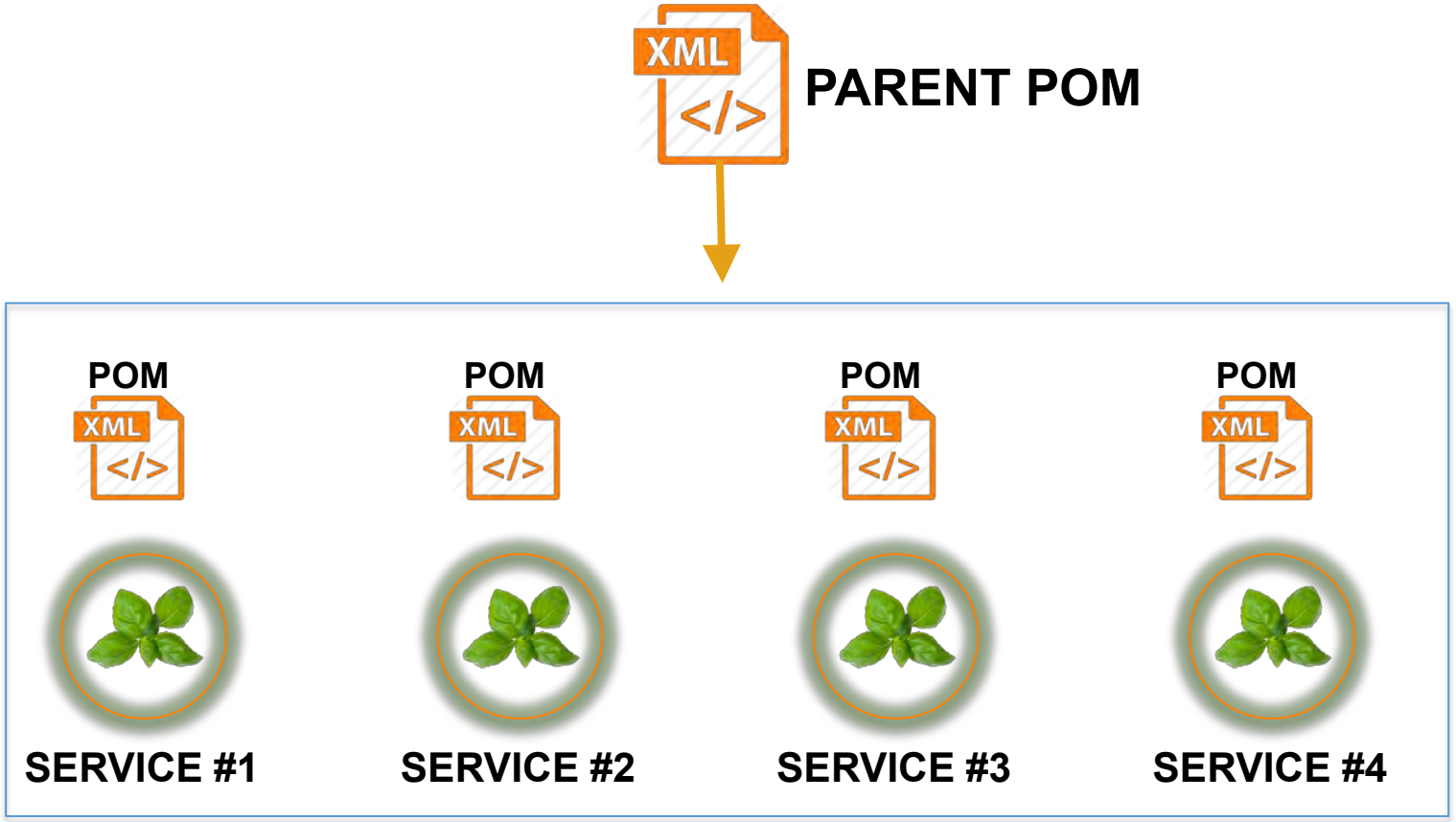
Maven POM structure

Parent

```
<parent>
        <groupId>com.mycompany.xxx</groupId>
        <artifactId>my-parent</artifactId>
        <version>123</version>
</parent>
```

Get main definitions to all child projects.

Parent

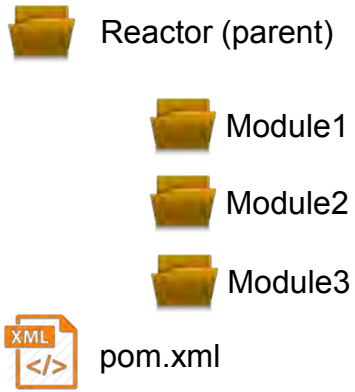

XML
PARENT POM
POM
POM
POM
POM
SERVICE #1
SERVICE #2
SERVICE #3
SERVICE #4

Multi module project

- Collects all the available modules to build
- Sorts the projects into the correct build order
- Builds the selected projects in order

Reactor POM

```
<modules>
  <module>module1</module>
  <module>module2</module>
  <module>module3</module>
</modules>
```



Dependency / plugin management

Allows project authors to directly specify the **versions** or **configuration** of artifacts or plugins where there is no version / configuration. Can be overwritten in the plugin itself.

```
<pluginManagement>
  <plugin>
      <groupId>com.myorg</groupId>
      <artifactId>myplugin</artifactId>.
      <version>1.1.1.1</version>
  </plugin>
</pluginManagement>
```

```
<plugin>
    <groupId>com.myorg</groupId>
    <artifactId>myplugin</artifactId>.
</plugin>
```

Maven Profiles

Subset of elements in the POM that can be triggered in many ways.
Can be defined both in the **pom.xml** or in the **settings.xml**

```
<profiles>
   <profile>
      <id>my_profile_id</id>
      …
   </profile>
</profiles>
```

```
$ mvn clean install -Pmy_profile_id
```

Maven Profiles Activation

```
<profiles>
    <profile>
        <id>my_profile_id</id>
        <activation>
            <activeByDefault>true</activeByDefault>
        </activation>
    </profile>
</profiles>
```

Maven Profiles Activation

```
<profiles>
    <profile>
        <id>my_profile_id</id>
        <activation>
            <jdk>1.8</jdk>
        </activation>
    </profile>
</profiles>
```

Practice

Introduction to
Jenkins CI Server

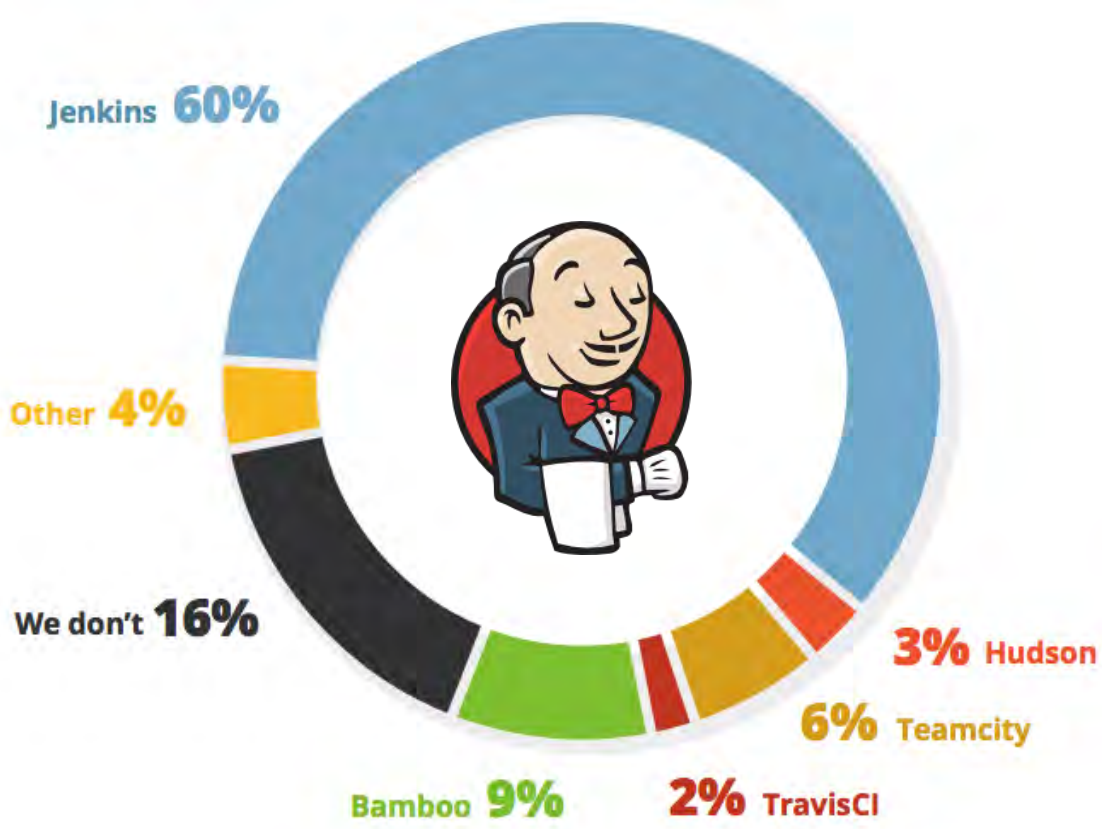


Figure 1.17 Continuous Integration Server Usage

Let’s install Jenkins first

- Download Jenkins (Preferred LTS) https://jenkins.io/
- Open up a terminal in the download directory.
- Run java -jar jenkins.war --httpPort=9090.
- Browse to http://localhost:9090.
- Follow the instructions to complete the installation.

Docker?

$ docker run -d -p 9090:8080 -u root -v /Users/nirk/jenkins_home:/var/jenkins_home jenkins/jenkins:lts

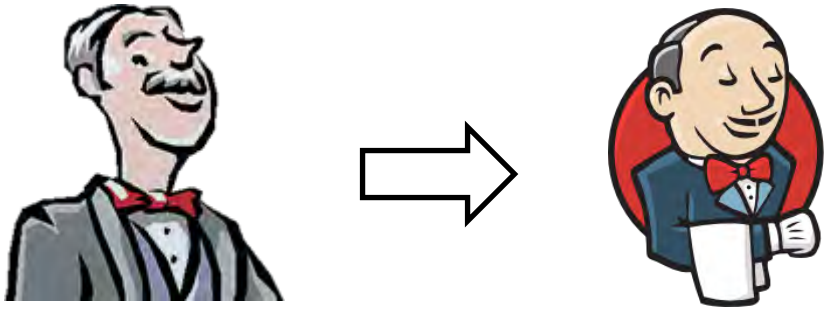
What is Jenkins?

Jenkins is **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.

History of Jenkins

- Funded by **Kohsuke Kawaguchi** as Hudson CI (in that time he was a developer in Sun Microsystems).
- First Hudson release was in 2004.
- Oracle acquired Sun in 2010 and claimed the right to the ”**Hudson**” name and applied for a trademark in December 2010.
- Kohsuke forked Hudson open source and created the name “**Jenkins”** – the rest is history.

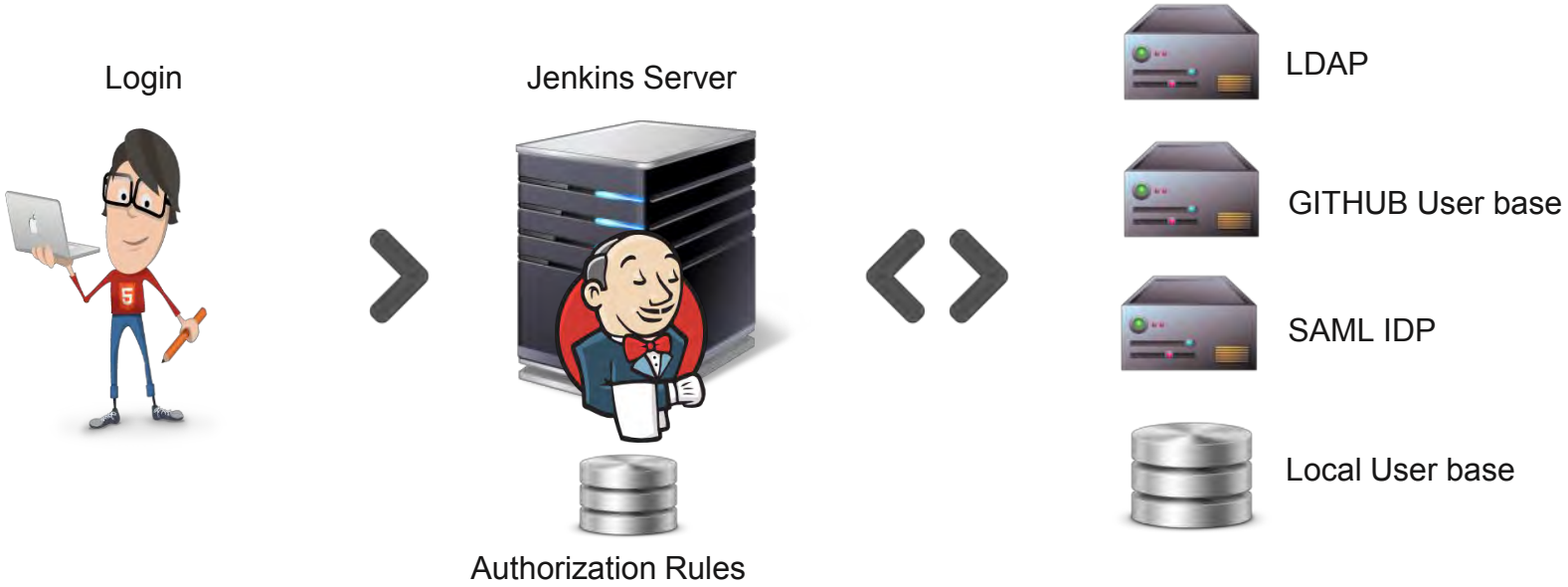


Why Jenkins?

A platform, Free, Open source. Easy to install and easy to use

Jenkins

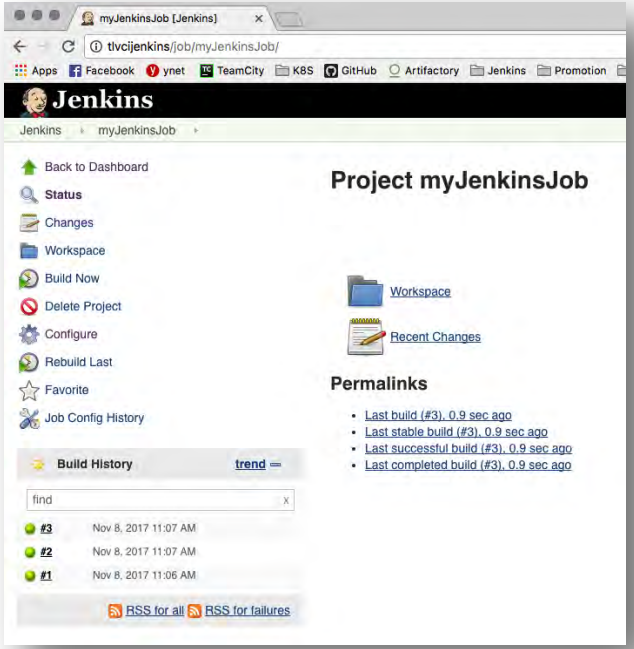
Provides built-in Authentication & Authorization



Jenkins

Open platform

1000+ plugins



JSON API

XML API

Jenkins

Extendable

Write your own plugin

Support Groovy DSL

Groovy declarative pipelines

```
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

Popular by the community

Short-time (rapid) Development

Practice