

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



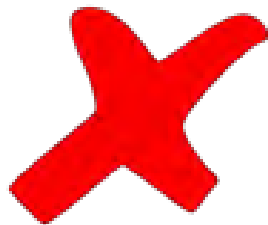
What will we **NOT** cover today?

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)

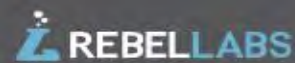
1000 Employees over 14 offices

19,000 customers, 27M interactions / Month



LIVEPERSON

THE 2014 LEADERBOARD OF JAVA TOOLS & TECHNOLOGIES



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

67.5%
Hibernate*°

THE TOP
ORM
FRAMEWORK
USED

65% **Java 7**

THE INDUSTRY LEADER FOR
SE DEVELOPMENT

56%
MongoDB*

THE NOSQL
TECHNOLOGY OF CHOICE

69% **Git***

#1 VERSION CONTROL
TECHNOLOGY OUT THERE

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



maven

Over **two in three (68%)** devs use Maven as their main build tool



Over **two in three (68%)** devs use Git as their version control



Java 8

Almost **two in three (62%)** devs use Java 8 in production



Jenkins

Three in five (60%) devs use Jenkins for CI



IntelliJ IDEA

Almost **one in two (46%)** devs use IntelliJ, the most popular IDE in the survey.



spring

Over **two in five (43%)** devs use Spring MVC



Tomcat

Over **two in five (42%)** devs use Tomcat server in production



ORACLE[®] DATABASE

Almost **two in five (39%)** devs use Oracle DB in production



Microservices

Over **one in three (34%)** devs have adopted a microservices architecture



docker

Almost **one in three (32%)** devs use Docker in production



Java EE 7

Over **three in ten (31%)** devs use Java EE 7



spring

Almost **three in ten (29%)** devs use Spring Boot



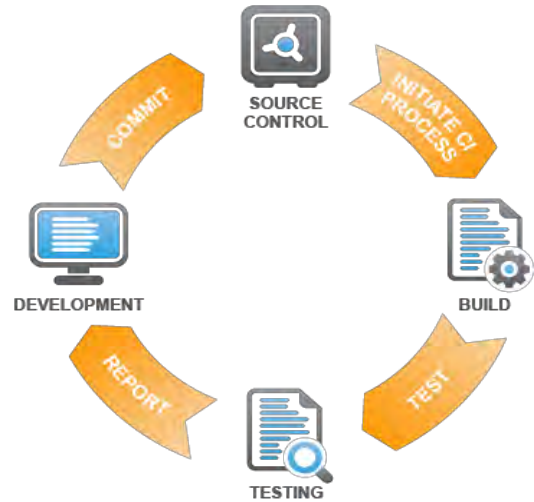
Introduction to **CONTINUOUS INTEGRATION / DELIVERY**

What is Continuous Integration?

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

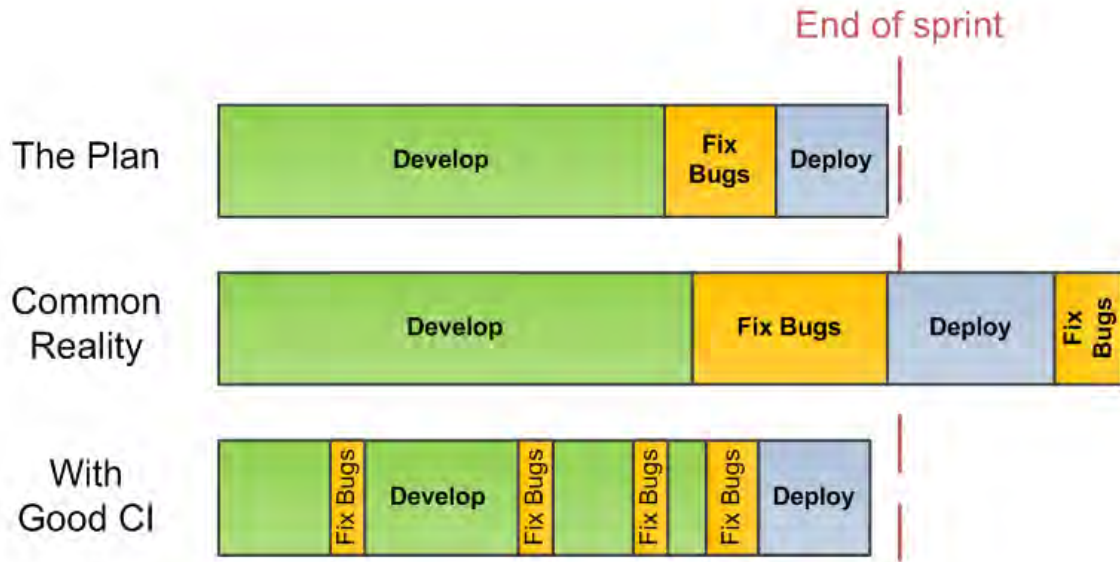
Martin Fowler, Continuous Integration:

<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

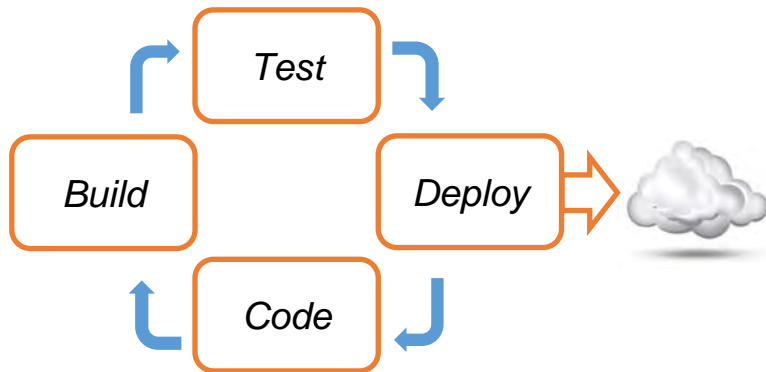
- **Automate** the build (single command), **Automate all**.
- Build is **self testable**.
- Baseline **branch is open consistently**.
- **Every commit** should be built.
- Build should be **fast**.
- Test Env is **clone of Production** (as much as we can).
- Everyone can see the status – **Transparency**.

What is

CONTINUOUS DELIVERY / DEPLOYMENT

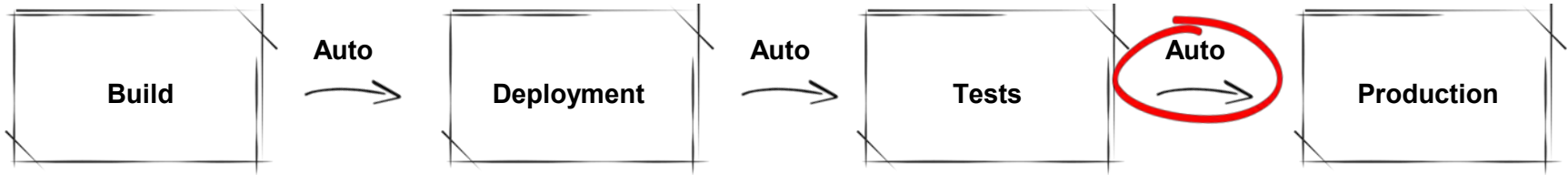
Continuous Delivery & Deployment

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**.

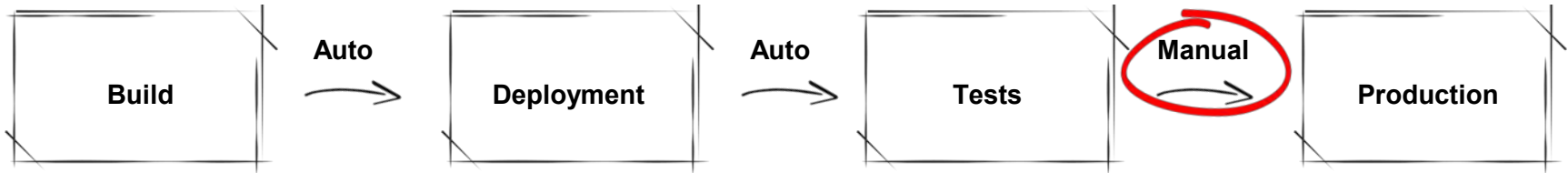


Continuous Deployment means that every change goes through the pipeline and automatically gets put into production, resulting in many production deployments every day.

Continuous **Deployment**



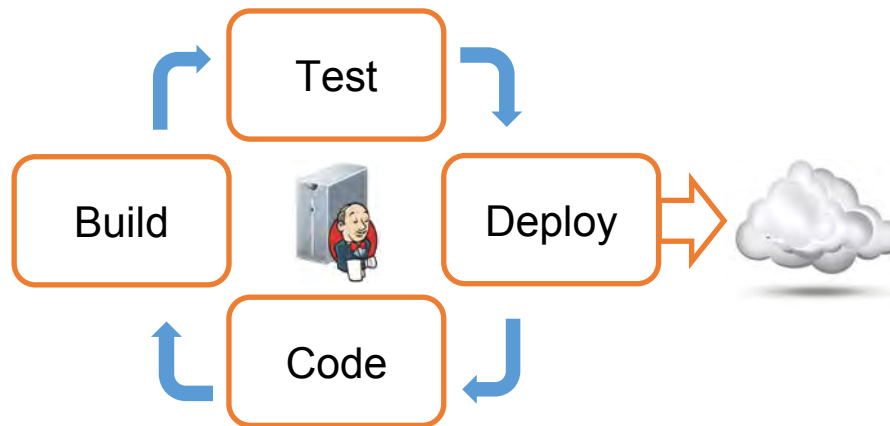
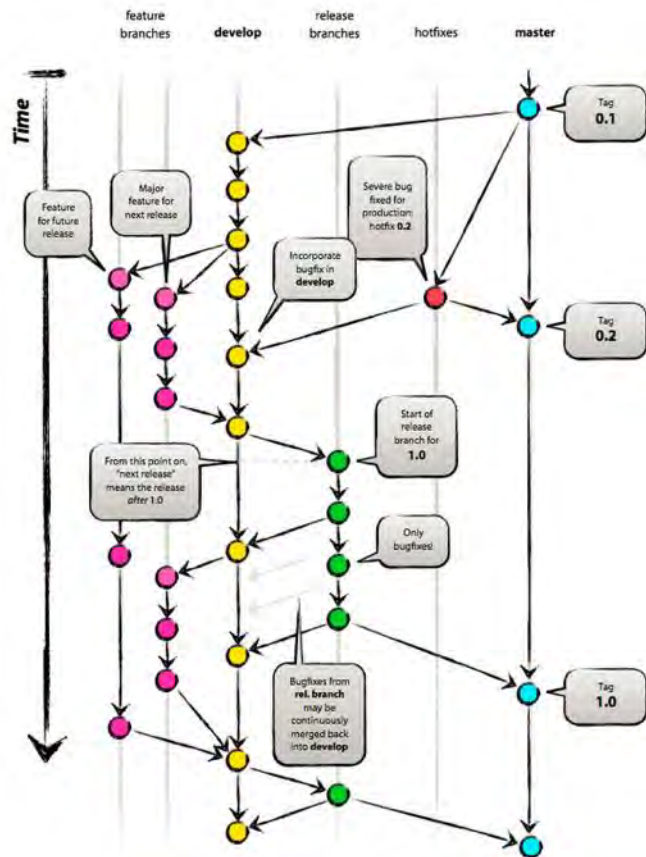
Continuous **Delivery**



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



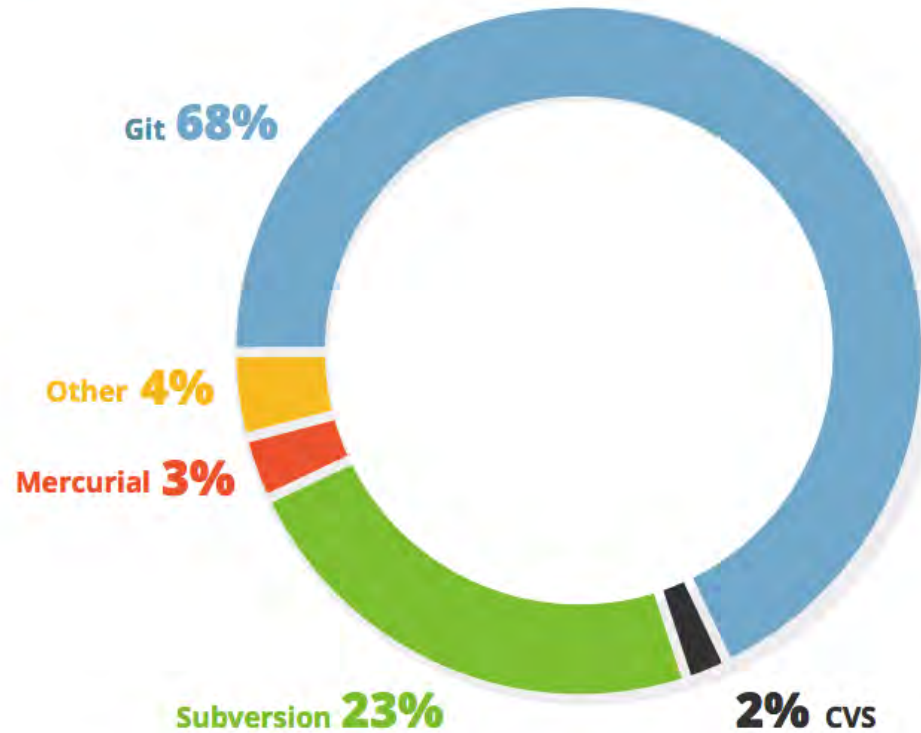
CD is not only in Computer Software

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



[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

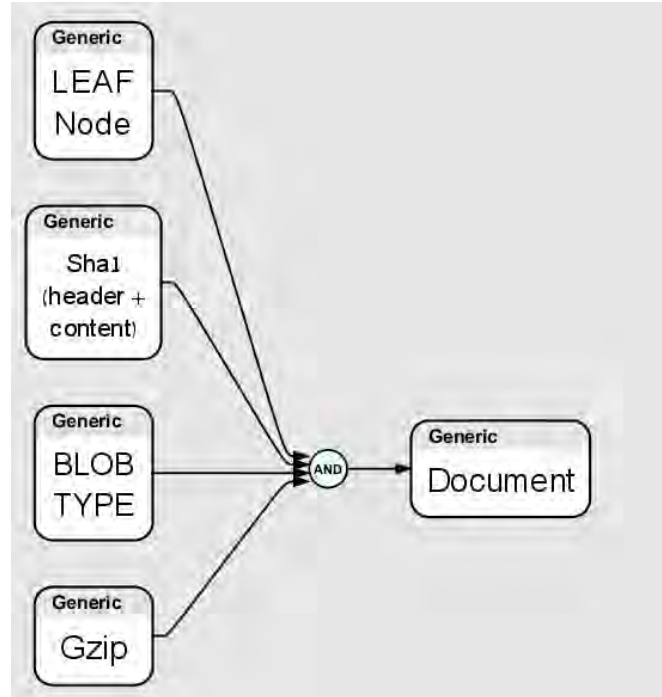
First release @ 2005 for Linux Kernel source code.
New free alternative for BitKeeper.

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**.

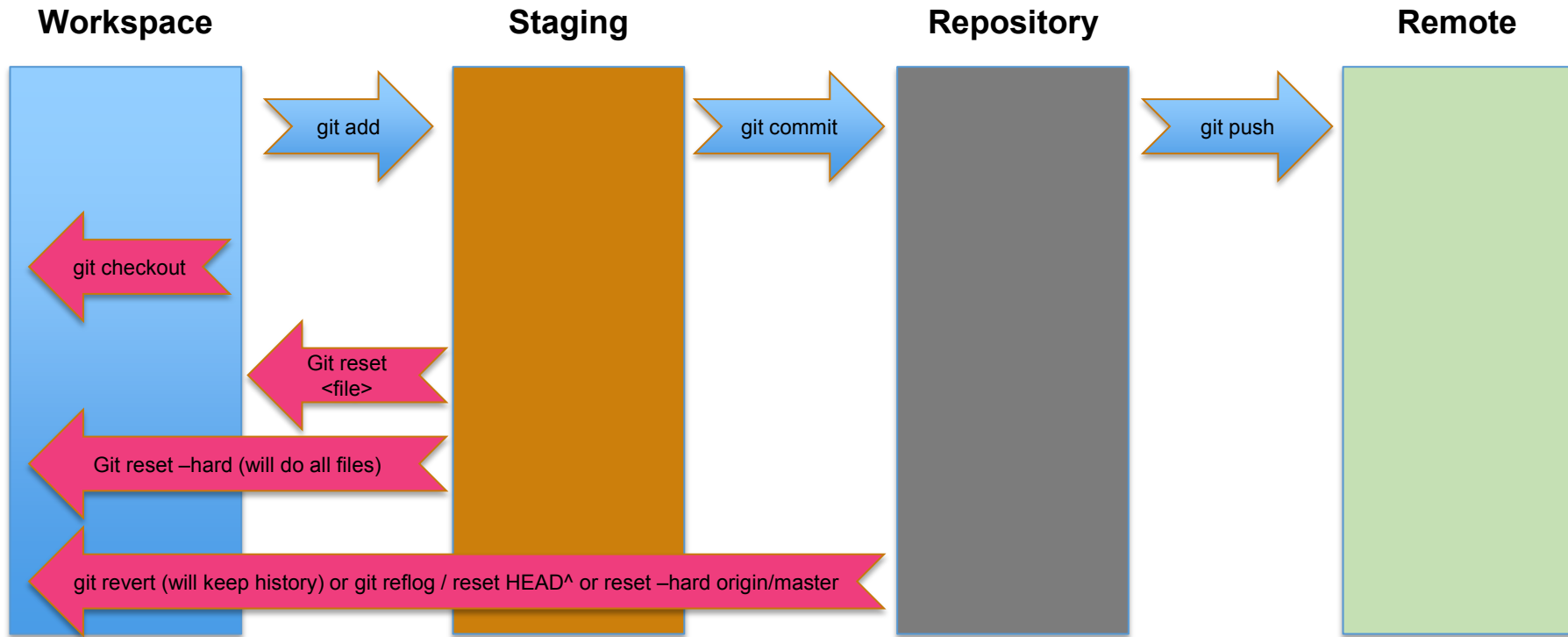
CLONE

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

FORK

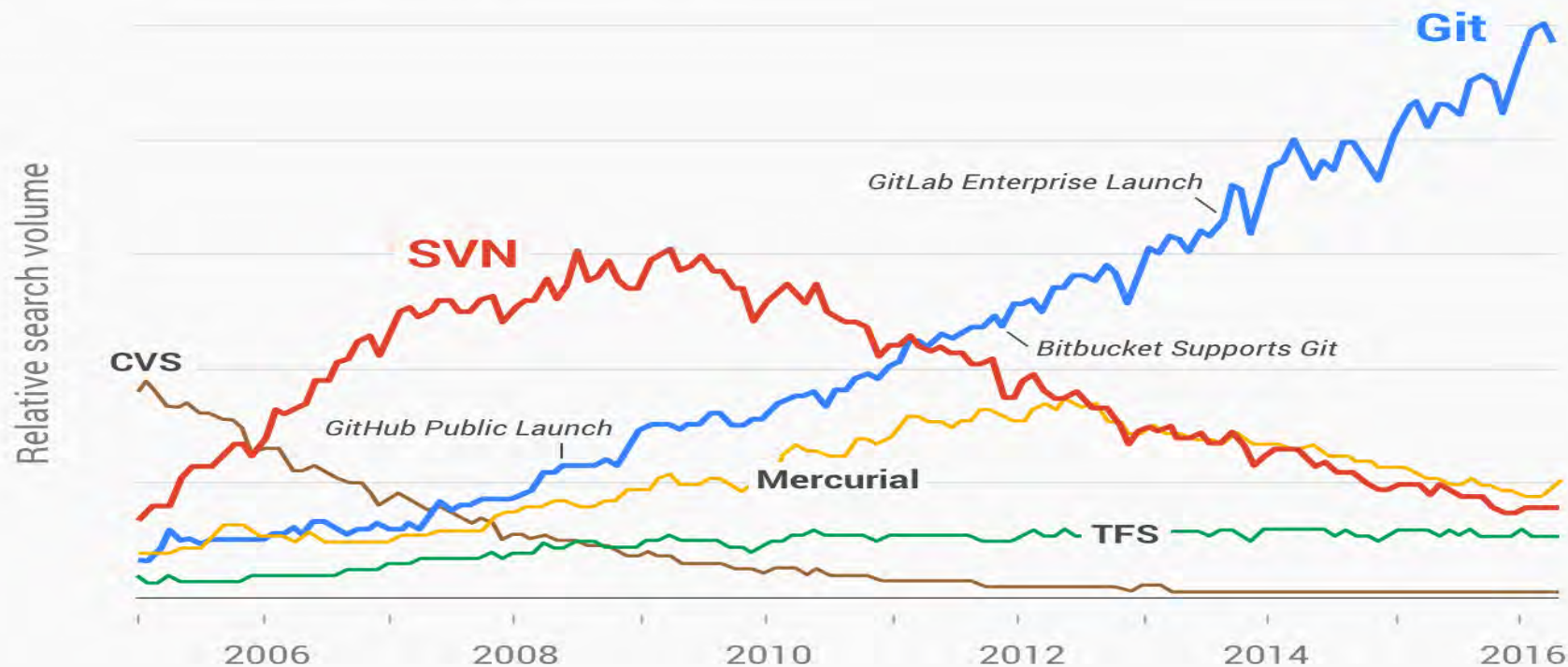
A **personal copy** of a repository that lives **remotely** on a user account and holds relationship with the original repo..

Git: Areas



How Did SVN Manage to Lose?

Version control interest over time



Companies & Projects Using Git

Google

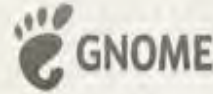
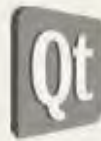
facebook

Microsoft

twitter

LinkedIn

NETFLIX



Perforce new products
contains integration to
Git

Work Locally, Scale Globally

We've partnered with GitLab to introduce
our GitSwarm ecosystem...

- Merge-request workflow
- Easy repo management
- Project visibility & security
- Automatically mirrors work into
Helix mainline repository

GET IT NOW



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

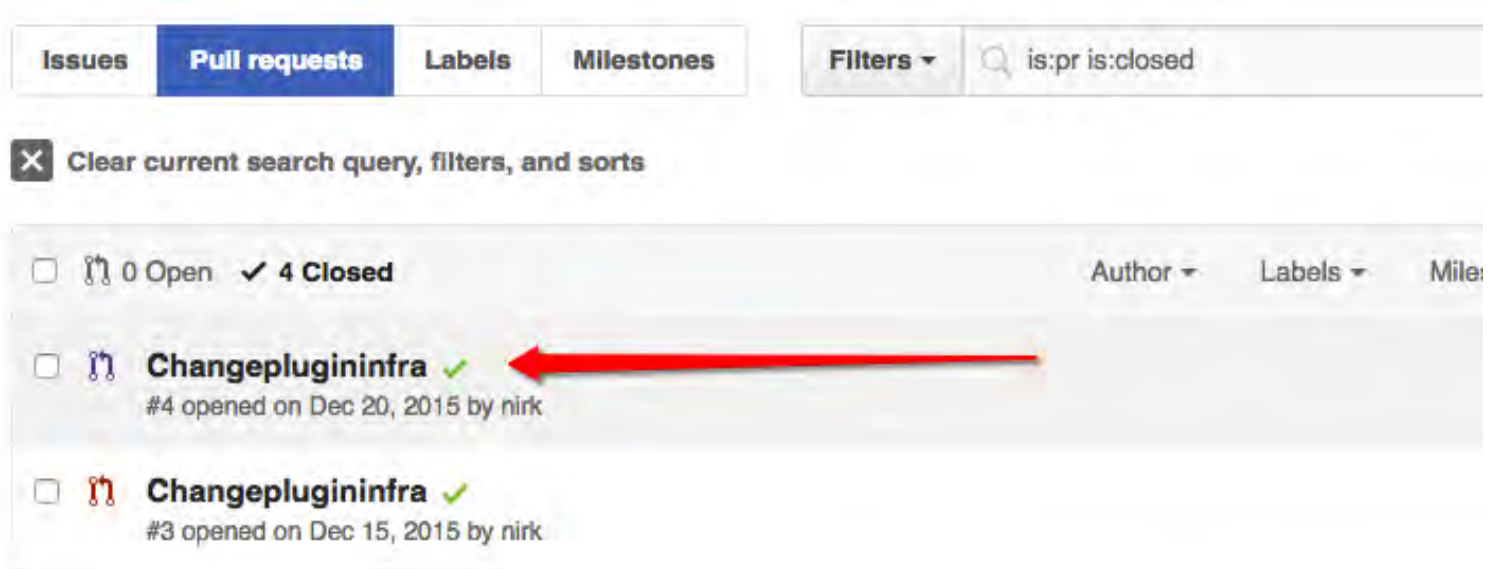
Blame

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



The screenshot displays the GitHub interface for pull requests. At the top, there are tabs for 'Issues', 'Pull requests' (which is selected and highlighted in blue), 'Labels', and 'Milestones'. To the right of these tabs is a search bar with the text 'is:pr is:closed' and a 'Filters' dropdown menu. Below the tabs, there is a button with a close icon and the text 'Clear current search query, filters, and sorts'. The main content area shows a list of pull requests. At the top of this list, it says '0 Open' and '4 Closed'. The first pull request in the list is titled 'Changeplugininfra' and has a green checkmark next to it. A red arrow points to this green checkmark. Below the title, it says '#4 opened on Dec 20, 2015 by nirk'. The second pull request in the list is also titled 'Changeplugininfra' and has a green checkmark next to it. Below its title, it says '#3 opened on Dec 15, 2015 by nirk'. On the right side of the pull request list, there are dropdown menus for 'Author', 'Labels', and 'Milestones'.

Issues Pull requests Labels Milestones

Filters is:pr is:closed

Clear current search query, filters, and sorts

0 Open ✓ 4 Closed

Author Labels Miles

Changeplugininfra ✓ #4 opened on Dec 20, 2015 by nirk

Changeplugininfra ✓ #3 opened on Dec 15, 2015 by nirk

Code Search engine

The screenshot shows the GitHub Code Search interface. At the top, there's a navigation bar with 'Pull requests', 'Issues', and 'Gist'. Below this, the 'Search' section is active. On the left, a sidebar contains filters: 'Repositories', 'Code' (highlighted with a red arrow and showing 2 results), 'Issues', and 'Users'. Below the filters, the 'Languages' section shows 'Java' selected (also highlighted with a red arrow and showing 2 results). At the bottom left, there are links for 'Advanced search' and 'Cheat sheet'. The main search area has a search bar containing the text '"class Createrepo"' (highlighted with a red arrow). Below the search bar, two search results are displayed. The first result is titled 'C/p-promote-plugin - CreateRepo.java' and shows the top two matches, last indexed on Feb 25. The code snippet shows imports for MojoExecutionException and MojoFailureException, followed by a public class CreateRepo extending AbstractPreAlpha. The second result is titled 'C/p-yum-plugin - CreateRepo.java' and shows the top match, last indexed 27 days ago. The code snippet shows imports for java.util.Properties and a Mojo class, followed by a public class CreateRepo extending AbstractLPYUMMojo with a @Parameter annotation and a private String reponame attribute.

Pull requests Issues Gist

Search

"class Createrepo"

Repositories

Code 2

Issues

Users

Languages

Java 2

Advanced search Cheat sheet

C/p-promote-plugin - CreateRepo.java
Showing the top two matches. Last indexed on Feb 25.

```
3 import org.apache.maven.plugin.MojoExecutionException;
4 import org.apache.maven.plugin.MojoFailureException;
5
6 public class CreateRepo extends AbstractPreAlpha{
```

C/p-yum-plugin - CreateRepo.java
Showing the top match. Last indexed 27 days ago.

```
13 import java.util.Properties;
14
15 @Mojo(name = "createrepo", threadSafe = true, defaultPhase = LifecyclePhase.INSTALL)
16 public class CreateRepo extends AbstractLPYUMMojo {
17
18     @Parameter(alias = "reponame", required = true)
19     private String reponame;
```

Rich documentation and API

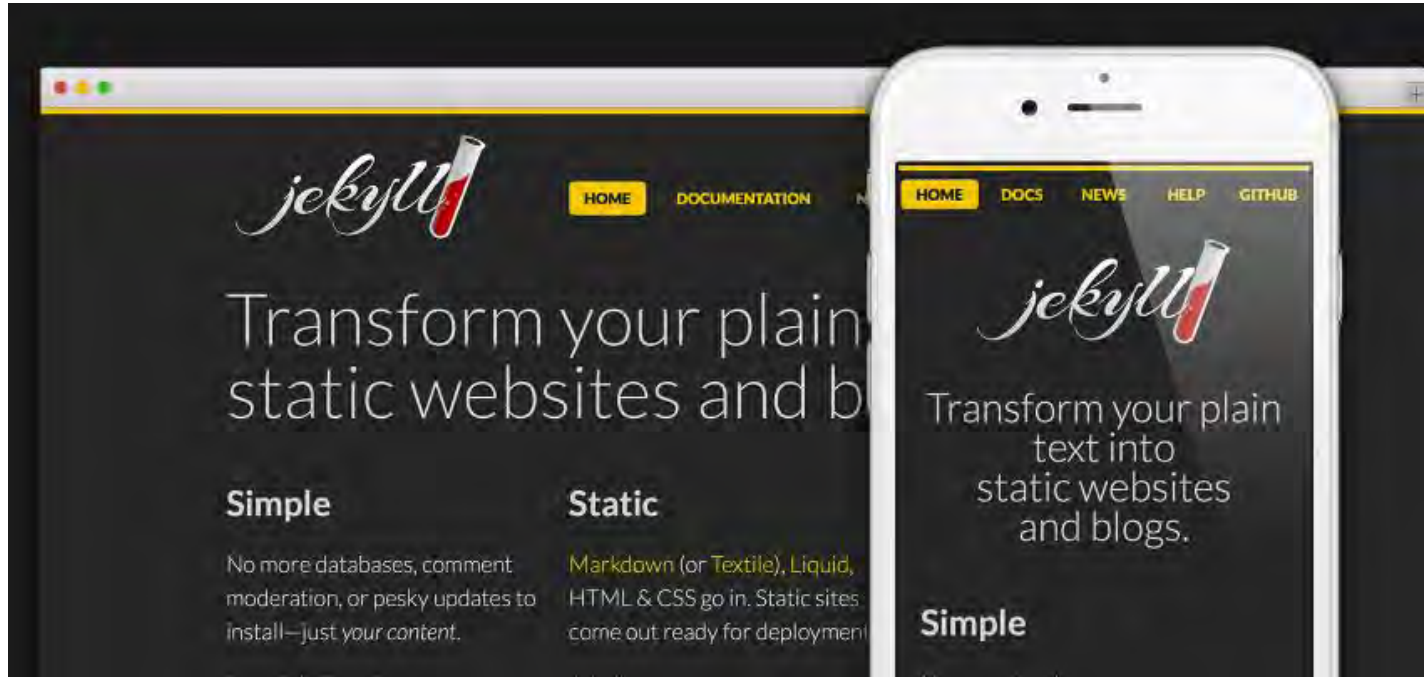
The screenshot displays a REST client interface with the following components:

- Request Method:** GET
- URL:** `https://api/v3/repos/CI/ci-dot-utils/commits?sha=master&author=vifat...`
- Headers (1):**

Key	Value
Authorization	Basic bmlyazpNeU1pa2EwMDk5IQ==
- Body:** The response is shown in JSON format, displaying a commit object with the following structure:

```
1 [
2   {
3     "sha": "54e000405918e0665c7780ed8690b73178b91721",
4     "commit": {
5       "author": {
6         "name": "Vladi Ifat",
7         "email": "vifat@liveperson.com",
8         "date": "2018-03-25T12:02:06Z"
9       },
10      "committer": {
11        "name": "Vladi Ifat",
12        "email": "vifat@liveperson.com",
13        "date": "2018-03-25T12:02:06Z"
14      },
15    }
16  }
```

GitHub Pages



GitHub WIKI



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

fabric8io / docker-maven-plugin

Watch 75

Star 842

Fork 347

<> Code


Issues 253

Pull requests 14


Projects 0

Wiki

Insights

 **Want to submit an issue to fabric8io/docker-maven-plugin?** [Dismiss](#)

If you have a bug or an idea, read the [contributing guidelines](#) before opening an issue.

Filters  is:issue is:open


Labels


Milestones


New issue

253 Open ✓ 391 Closed


Author Labels Projects Milestones Assignee Sort


 **How to add arguments while building a docker image using plugin**
#1011 opened 5 days ago by rajat-garg

 **log file output broken in 0.24.0, 0.25.0, 0.25.1, 0.25.2**
#1010 opened 5 days ago by chonton

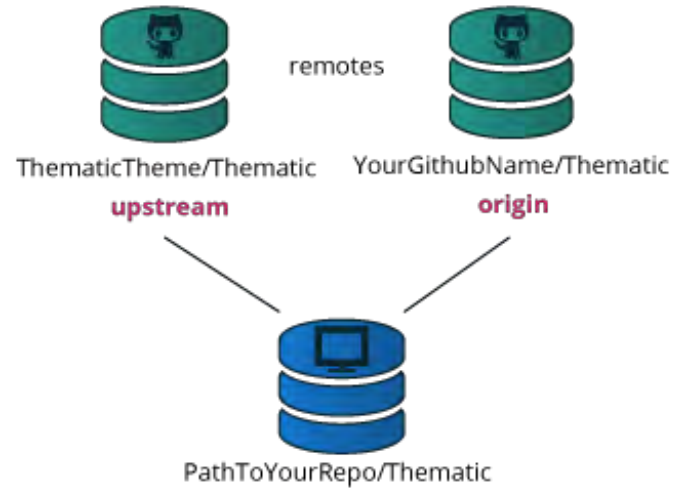
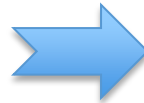
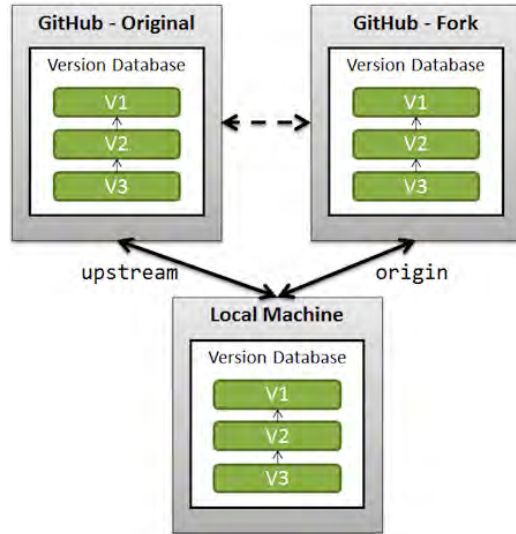
 **Dockerfile with zero-config**
#1008 opened 7 days ago by burtsevyg

2

 **Add support for passint --init parameter to docker when starting containers**
#1003 opened 13 days ago by thrawn-sh

 **Change default phase for docker:build to "package"**

Forks model – Open Source work model









Pull Request

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



Communicate with Emojis

 dannyfritz New emoji for deprecation notices 🗑️	
 INTEGRATIONS.md	 Update emoji packages required for Atom
 LICENSE	 Initial commit
 README.md	New emoji for deprecation notices 🗑️

Manage your tasks in Board

The screenshot displays the Board project management interface for a repository named "Production / D6R". The interface includes a navigation bar with links for "Code", "Issues 29", "Pull requests 0", "Projects 1", "Wiki", "Pulse", "Graphs", and "Settings". Below the navigation bar, the "Self Service" section is visible, showing a Kanban board with four columns: "To Do" (19 tasks), "In Progress" (6 tasks), "BLOCKED" (1 task), and "Completed" (49 tasks). Each task card includes a title, a brief description, an issue number, and the assignee's name.

Column	Task Title	Issue Number	Assignee
To Do (19)	Add discovery for service health per BB.	#96	liranc
	Move scripter code to main DSL code as func.	#86	liranc
	Investigate about dashboarding / transparency of the activity of the Jenkins / jobs	#78	nirk
	when choosing alpha segment and pci yes then show only alpha servers and not va	#69	tomerb
	cobrowse_app lpnova role does not match module		
In Progress (6)	AC Metadata is identified as jboss instead of tomcat causes build to fail	#95	tomerb
	if rnd-a person created pull request cannot assign to rnd-b person in lpgithub	#80	tomerb
	Support for PCI enhancement	#46	liranc
	puppet module d6r_web	#4	liranc
BLOCKED (1)	ssh to Servers	#7	liranc
Completed (49)	PCI: svvr-mng40 is not open to ca/io/am which causes D6R to fail	#94	tomerb
	SY-R10K no such command found [root@rmor-mng32 ~]# r10k -bash: r10k: command not found	#87	tomerb
	SY-R10K itself fails on sydney see logs	#90	tomerb
	deployer validation	#44	tomerb
	Integrate Orca with system health	#26	kobyh

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.

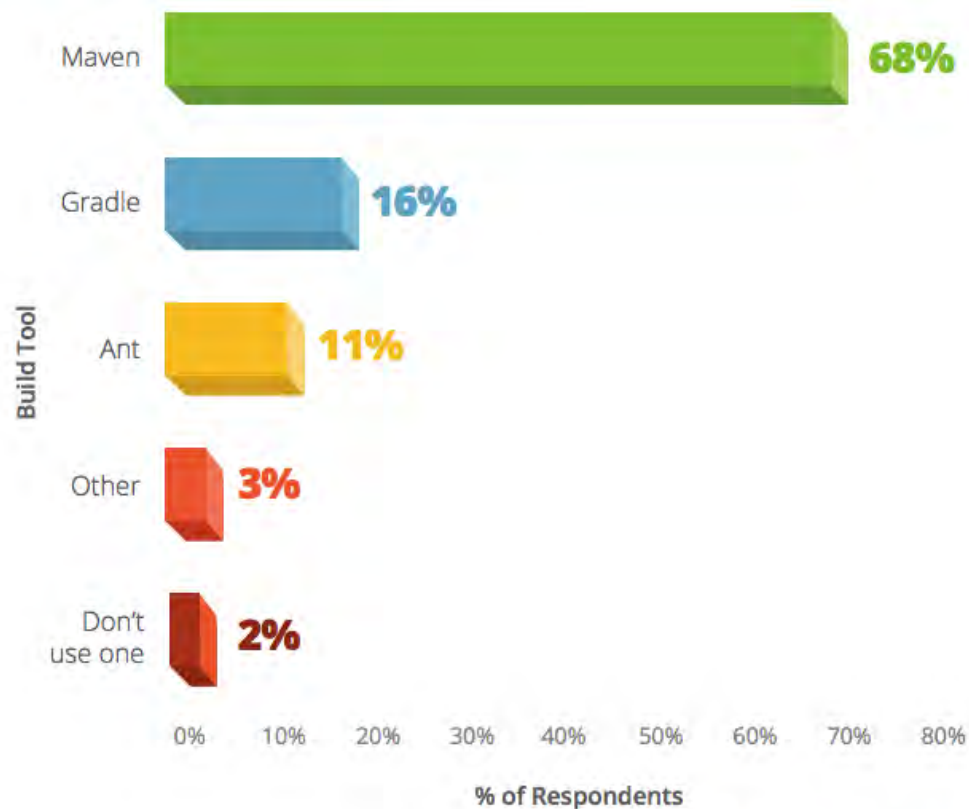
CONTRIBUTOR

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

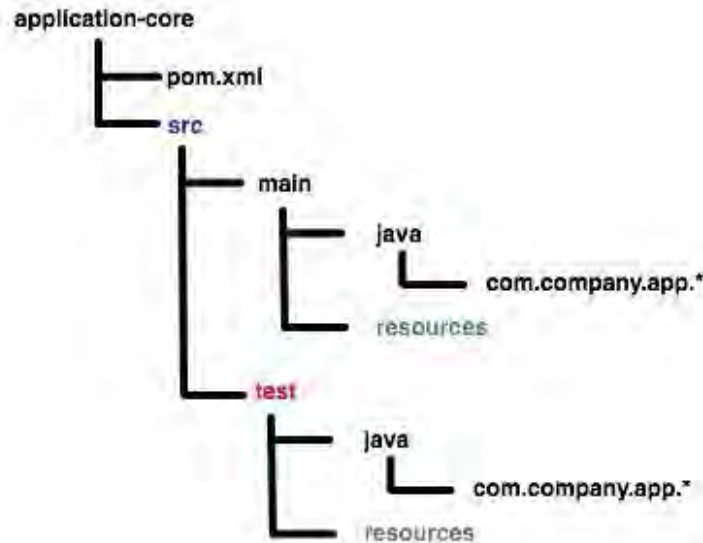
THE EVOLUTION OF BUILD TOOLS: 1977 - 2013 (AND BEYOND)

Visual timeline

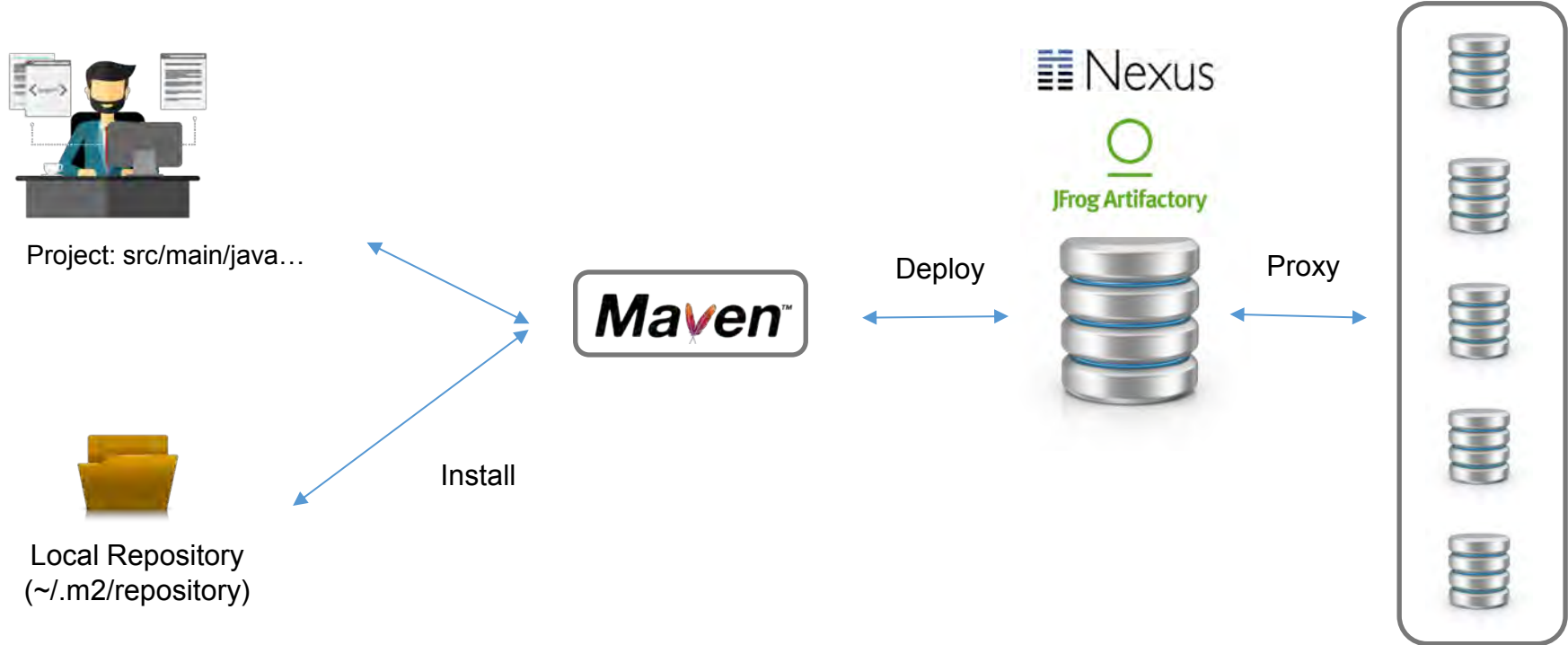


Maven directories structure

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



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>
        <someproperty>xxx</someproperty>
      </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

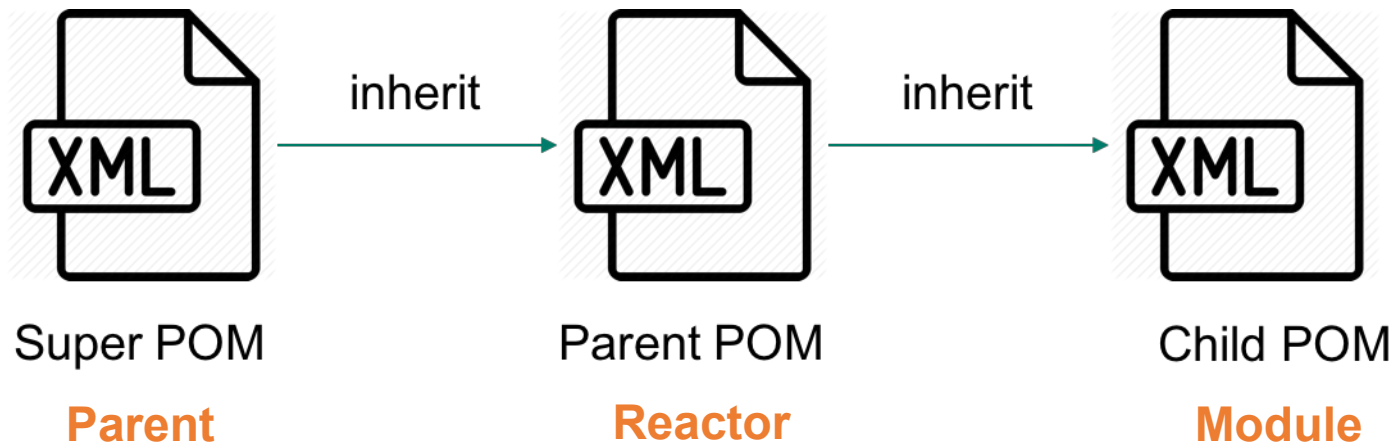
Maven SNAPSHOT Update Policy

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



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



PARENT POM



POM



SERVICE #1

POM



SERVICE #2

POM



SERVICE #3

POM



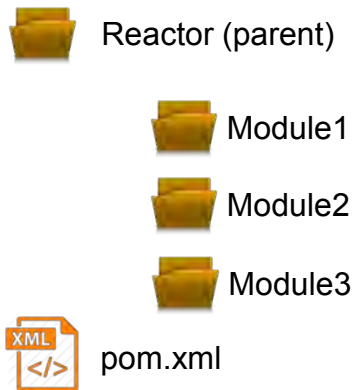
SERVICE #4

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**

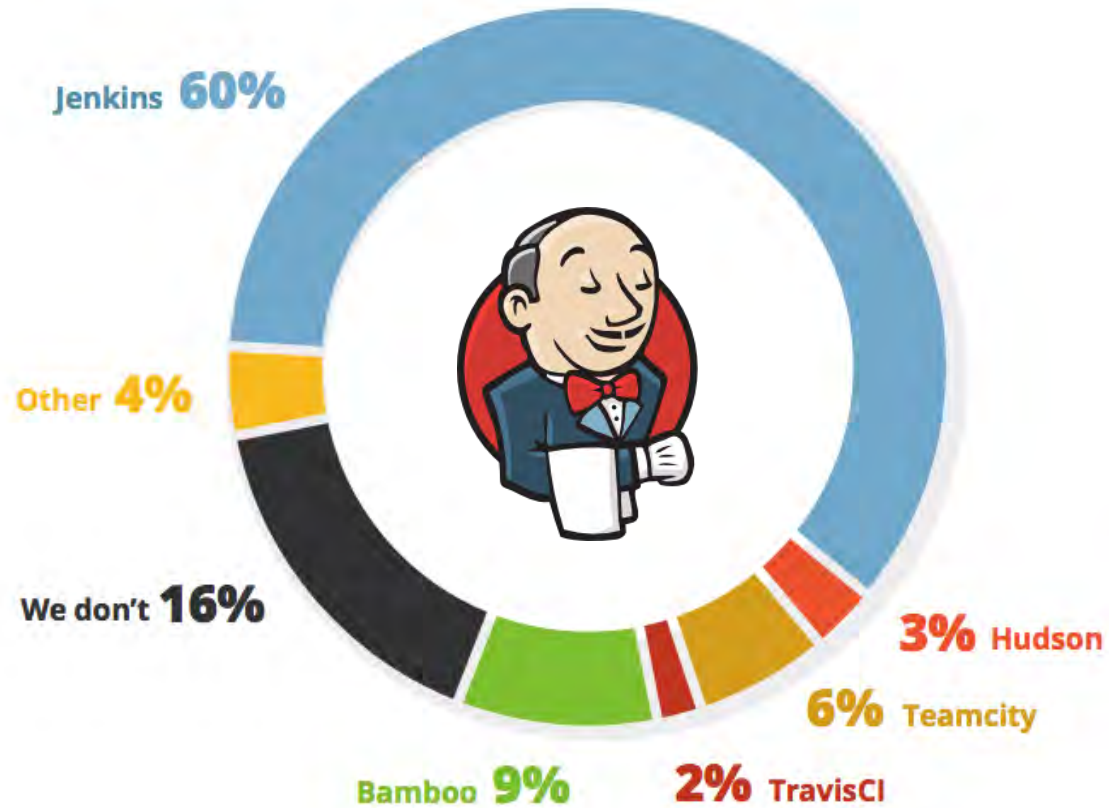


Figure 1.17 Continuous Integration Server Usage

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:its
```



What is Jenkins?

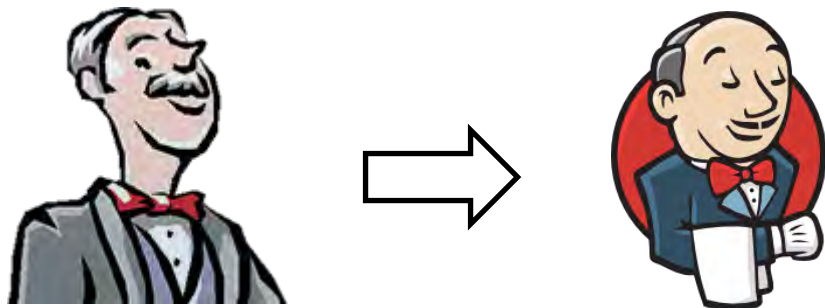
Jenkins is **open source automation server** which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.

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.



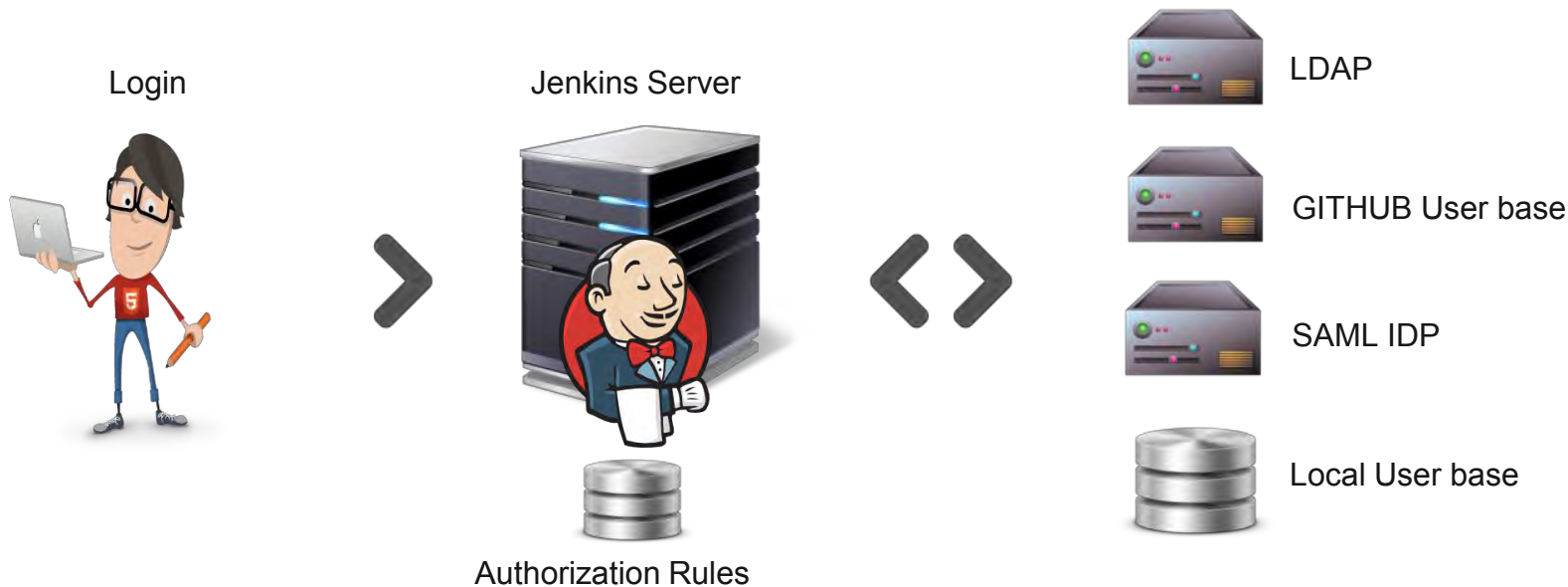
Why Jenkins?

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





Provides built-in Authentication & Authorization





Jenkins

Open platform

1000+ plugins



The screenshot shows the Jenkins web interface. At the top, there's a navigation bar with the Jenkins logo and a breadcrumb trail: Jenkins > myJenkinsJob. Below this, a sidebar on the left contains a list of actions: Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, Configure, Rebuild Last, Favorite, and Job Config History. The main content area is titled 'Project myJenkinsJob' and includes links for 'Workspace' and 'Recent Changes'. Below these links is a 'Permalinks' section with a list of build links: 'Last build (#3), 0.9 sec ago', 'Last stable build (#3), 0.9 sec ago', 'Last successful build (#3), 0.9 sec ago', and 'Last completed build (#3), 0.9 sec ago'. At the bottom, there's a 'Build History' table with columns for build number, status, and time. It shows three builds: #3 (green), #2 (green), and #1 (green), all from Nov 8, 2017. At the very bottom, there are RSS feed links for 'all' and 'failures'.

JSON API

```
{
  "_class": "hudson.model.FreeStyleProject",
  "actions": [6],
  "description": "",
  "displayName": "myJenkinsJob",
  "displayNameOrNull": null,
  "fullDisplayName": "myJenkinsJob",
  "fullName": "myJenkinsJob",
  "name": "myJenkinsJob",
  "url": "http://ctvr-jenkins:8080/job/myJenkinsJob/",
  "buildable": true,
  "builds": [
    {
      "_class": "hudson.model.FreeStyleBuild",
      "number": 1,
      "url": "http://ctvr-jenkins:8080/job/myJenkinsJob/1/"
    }
  ]
}
```

XML API

```
<?xml version='1.0' encoding='UTF-8'>
<project class="hudson.model.FreeStyleProject">
  <action/>
  <action/>
  <action class="hudson.plugins.jobConfigHistory.JobConfigHistoryProjectAction"/>
  <action/>
  <description></description>
  <displayName>myJenkinsJob</displayName>
  <fullDisplayName>myJenkinsJob</fullDisplayName>
  <fullName>myJenkinsJob</fullName>
  <name>myJenkinsJob</name>
  <url>http://ctvr-jenkins:8080/job/myJenkinsJob/</url>
  <buildable true</buildable>
  <build class="hudson.model.FreeStyleBuild">
    <number 3</number>
    <url>http://ctvr-jenkins:8080/job/myJenkinsJob/3/</url>
  </build>
  <build class="hudson.model.FreeStyleBuild">
    <number 2</number>
    <url>http://ctvr-jenkins:8080/job/myJenkinsJob/2/</url>
  </build>
  <build class="hudson.model.FreeStyleBuild">
    <number 1</number>
    <url>http://ctvr-jenkins:8080/job/myJenkinsJob/1/</url>
  </build>
</project>
</xml>
```



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





Jenkins

Short-time (rapid) Development



Practice