JENKINS & CI/CD

JOHN BRYCE
Leading in IT Education
a matrix company

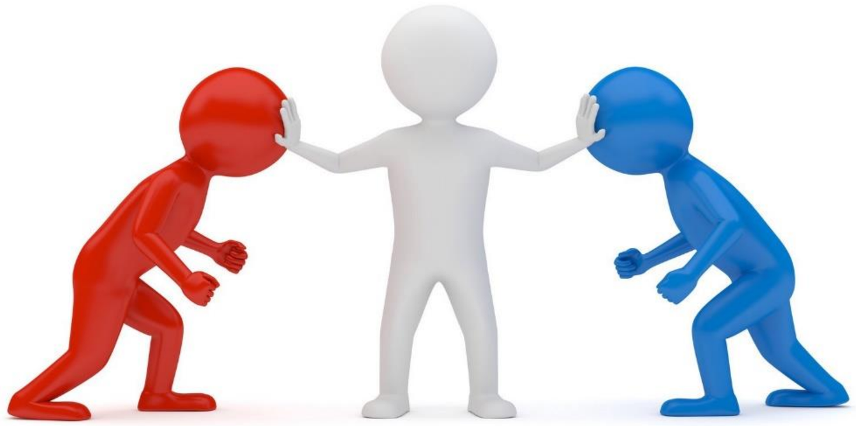
Topics

1. Continuous Integration (CI)
2. A Typical CI Flow
3. What is Jenkins?
4. Installing Jenkins
5. How Jenkins Works?
6. It’s All In the Command Line!
7. Types of Jenkins Jobs
8. Not Only a Nice UI for Scripting
9. Live Demo: Configuring a Job

Continuous Integration (CI)

The Problem:

- Large software projects are developed by more than one developer.
- Work done by one developer may **conflict** with the work that was done by other developers.
- Developers might introduce **new errors and bugs** to the central code repository and **break** each other's work.
- The longer one developer works on his copy of the project without integrating it with the work done by other team members, the greater the risk of multiple **integration conflicts** and failures.
- The greater the difference between the code base and one specific developer's copy of the project, the more work has to be done to integrate between the versions and **resolve conflicts**.



Continuous Integration (CI)

The Goal:

- Prevent integration problems (known as: "integration hell" or "merge hell")
- Avoid one developer's work-in-progress breaking another developer's copy
- Reduce cost and time by reducing re-work

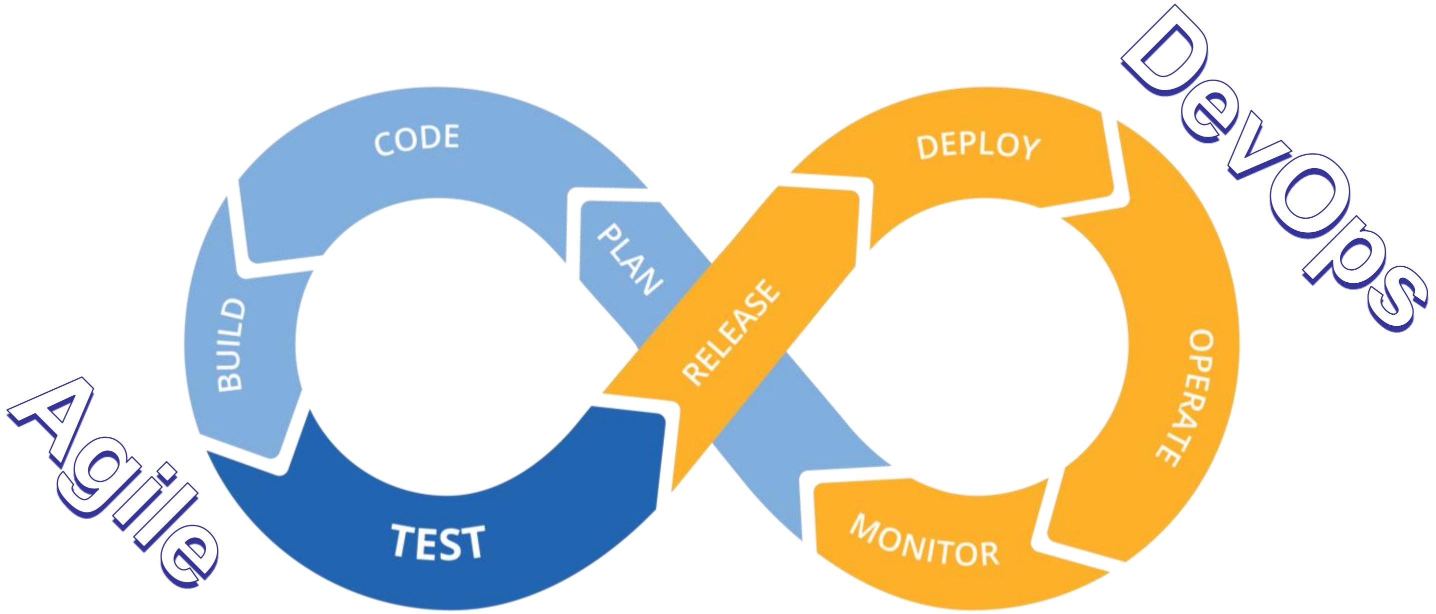
The Solution: CI - Integrating Early and Often

- Merge all developers' work to a shared mainline several times a day
- Use **automated unit and integration tests** to verify that there is no regression
- Use build servers to run the unit tests periodically or after every commit, and report the results to the developers

Continuous Delivery (CD) - further extends CI by making sure the software checked in on the mainline is always in a state that can be deployed to users and makes the deployment process very rapid.

A Typical CI Flow

1. A software developer checks-in his code changes to the central repository (e.g. git push), and then the CI system (e.g. Jenkins) steps in:
2. Pull latest code from VCS
3. Build project
4. Run automated tests
5. Send notifications
6. If all previous steps passed – Release & Deploy



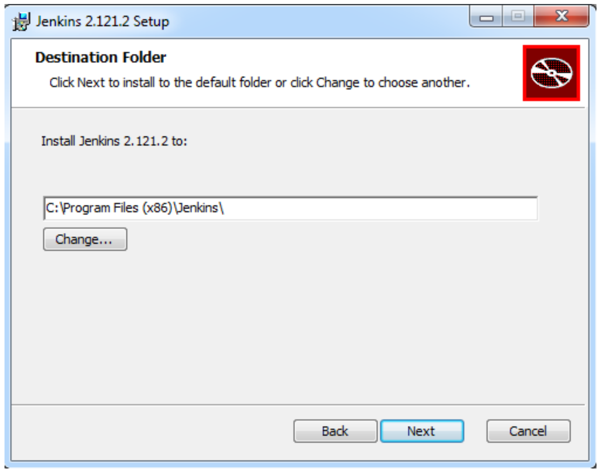
What is Jenkins?

- Jenkins is an open source **automation server**
- Jenkins is a **Java application** with a Web user interface
- Jenkins can be used to automate all sorts of tasks related to software:
  - **Building**
  - **Testing**
  - **Delivering** or **Deploying**
- Provides hundreds of **plugins** which allow work with practically every tool in the continuous integration (CI) and continuous delivery (CD) **toolchain**
- Can be used as a simple CI server or turned into the continuous delivery hub for any project

Installing Jenkins

JOHN BRYCE
Leading in IT Education
a matrix company

- Verify that you have **Java 8** installed on your OS
- Download a Jenkins installer appropriate for your OS from: https://jenkins.io/download/
- On Windows, extract the downloaded ZIP file and run: **jenkins.msi**
- Follow the instructions of the installer and accept the defaults
- When the installer launches a browser at: http://localhost:8080 – follow the instructions in the browser, and select "Install Suggested Plugins"
- Jenkins will install a bunch of plugins – a process which takes a few minutes



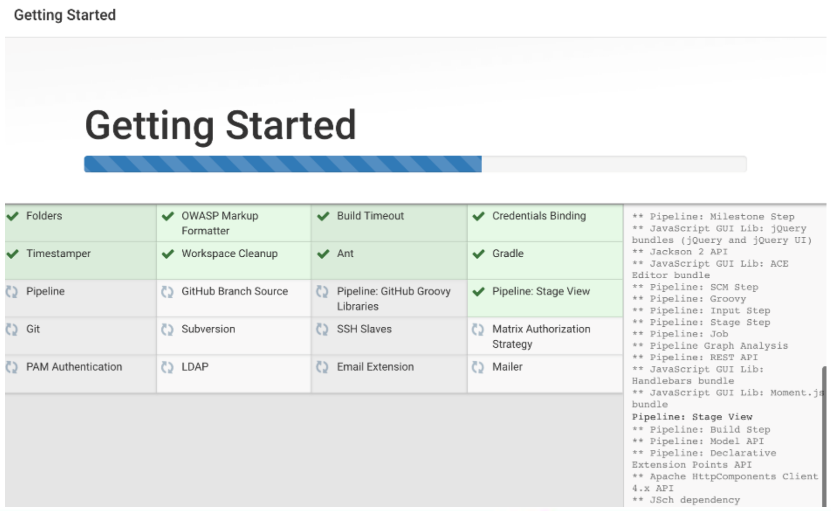
Jenkins 2.121.2 Setup

Destination Folder

Click Next to install to the default folder or click Change to choose another.

Install Jenkins 2.121.2 to:

C:\Program Files (x86)\Jenkins\



Getting Started

Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding
✓ Timestamper	✓ Workspace Cleanup	✓ Ant	✓ Gradle
Pipeline	GitHub Branch Source	Pipeline: GitHub Groovy Libraries	✓ Pipeline: Stage View
Git	Subversion	SSH Slaves	Matrix Authorization Strategy
PAM Authentication	LDAP	Email Extension	Mailer

```
** Pipeline: Milestone Step
** JavaScript GUI Lib: jQuery
bundles (jQuery and jQuery UI)
** Jackson 2 API
** JavaScript GUI Lib: ACE
Editor bundle
** Pipeline: SCM Step
** Pipeline: Groovy
** Pipeline: Input Step
** Pipeline: Stage Step
** Pipeline: Job
** Pipeline Graph Analysis
** Pipeline: REST API
** JavaScript GUI Lib:
Handlebars bundle
** JavaScript GUI Lib: Moment.js
bundle
Pipeline: Stage View
** Pipeline: Build Step
** Pipeline: Model API
** Pipeline: Declarative
Extension Points API
** Apache HttpComponents Client
4.x API
** JSch dependency
```

How Jenkins Works?

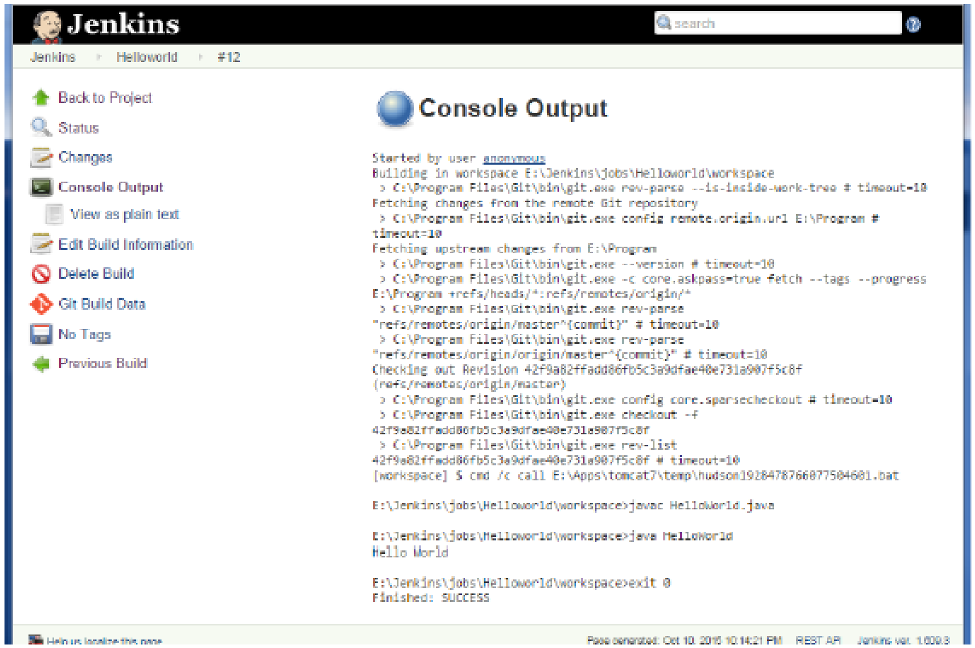
- A task executed by Jenkins is called a **Job**
- A Jenkins job consists of a series of **command-line utilities** that are invoked in a useful manner
  Example: **git** (pull code) → **mvn** (build the project) → **java** (run the code)
- When configuring a job, command-line utilities can be invoked from simple console scripts (**batch scripts** on Windows; **shell scripts** on Linux), or by using a dedicate plugin for each utility that we need to run (if such plugin exists).
- Jenkins plugins provide a friendly way to invoke command-line utilities and provide CLI arguments, but they are also eventually translated to command-line scripts when a job runs.

It's All In The Command Line!

REMEMBER: It's all about the command line!

- If you figure out the command lines for each of the steps you need Jenkins to perform – you'll have no problem implementing them as part of your Jenkins job.

- In order to **debug** a Jenkins job as you work on its configuration – you need to examine the job's **console output** and understand which command lines were executed by Jenkins and what were the output / error messages printed to the console.



Console Output

```
Started by user anonymous
Building in workspace E:\Jenkins\jobs\Helloworld\workspace
 > C:\Program Files\Git\bin\git.exe rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > C:\Program Files\Git\bin\git.exe config remote.origin.url E:\Program #
timeout=10
Fetching upstream changes from E:\Program
 > C:\Program Files\Git\bin\git.exe --version # timeout=10
 > C:\Program Files\Git\bin\git.exe -c core.askpass=true fetch --tags --progress
E:\Program +refs/heads/*:refs/remotes/origin/*
 > C:\Program Files\Git\bin\git.exe rev-parse
"refs/remotes/origin/master^{commit}" # timeout=10
 > C:\Program Files\Git\bin\git.exe rev-parse
"refs/remotes/origin/origin/master^{commit}" # timeout=10
Checking out Revision 42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f
(refs/remotes/origin/master)
 > C:\Program Files\Git\bin\git.exe config core.sparsecheckout # timeout=10
 > C:\Program Files\Git\bin\git.exe checkout -f
42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f
 > C:\Program Files\Git\bin\git.exe rev-list
42f9a82ffadd86fb5c3a9dfae40e731a907f5c8f # timeout=10
[workspace] $ cmd /c call E:\Apps\tomcat7\temp\hudson1928478766077504601.bat

E:\Jenkins\jobs\Helloworld\workspace>javac HelloWorld.java

E:\Jenkins\jobs\Helloworld\workspace>java HelloWorld
Hello World

E:\Jenkins\jobs\Helloworld\workspace>exit 0
Finished: SUCCESS
```

Types of Jenkins Jobs

Freestyle project

- This kind of job is entirely configurable from the Jenkins web UI
- Mosely relies on invocation of existing Jenkins plugins, but can also run batch or shell scripts

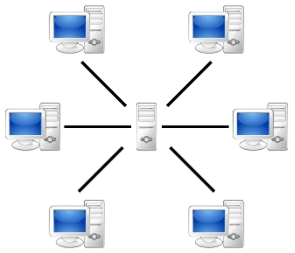
---

Pipeline

- Implemented with **Groovy** programming language scripts
- Much more powerful and flexible compared to a freestyle job
- Allows implementing complex execution flows and logic
- Supports invocation of other jobs (one job calls another job)

- Both types of jobs can invoke different Jenkins plugins, or specify general **batch** or **shell scripts** (depending on the OS type where the job should be executed).
- Both types of jobs can be **parameterized** – allowing the user to inject different values, and thus make the job configurable.

Not Only a Nice UI for Scripting



Distributed System

Jenkins jobs can be executed on the Jenkins server machine (also known as the “master” – this is the machine where Jenkins is installed), or on “nodes” (also called “slaves”) – other computers connected to the Jenkins server.
This allows **parallel** execution of Jenkins jobs on multiple “executors” at the same time.

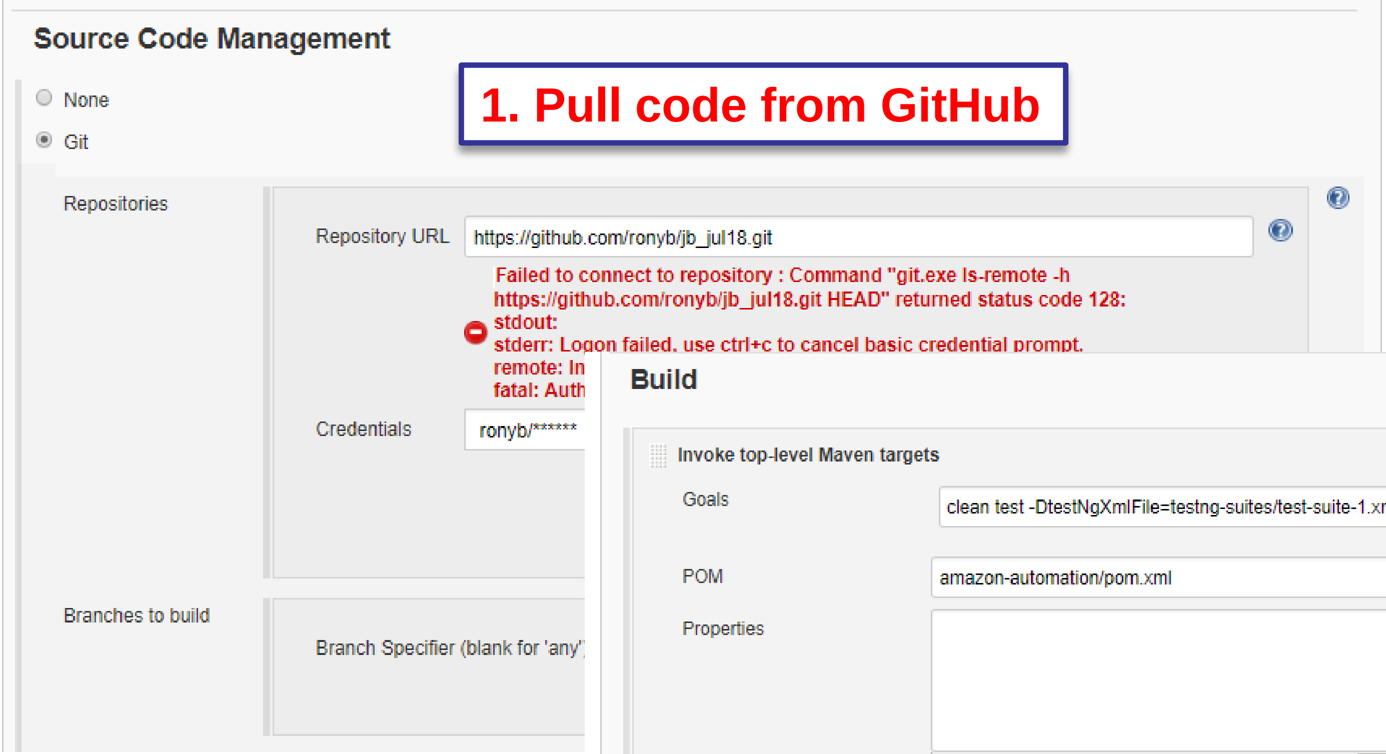
Scheduling and Triggering

Jenkins jobs can be invoked **manually** or they can be configured to run automatically according to a certain **schedule,** or **triggered** by certain events. e.g. a new commit to the repository.

Notifications

Jenkins jobs can be configured to send automatic **email notifications** after each successful / failed build.

Live Demo: Configuring a Job



Source Code Management

None

Git

Repositories

Repository URL https://github.com/ronyb/jb_jul18.git

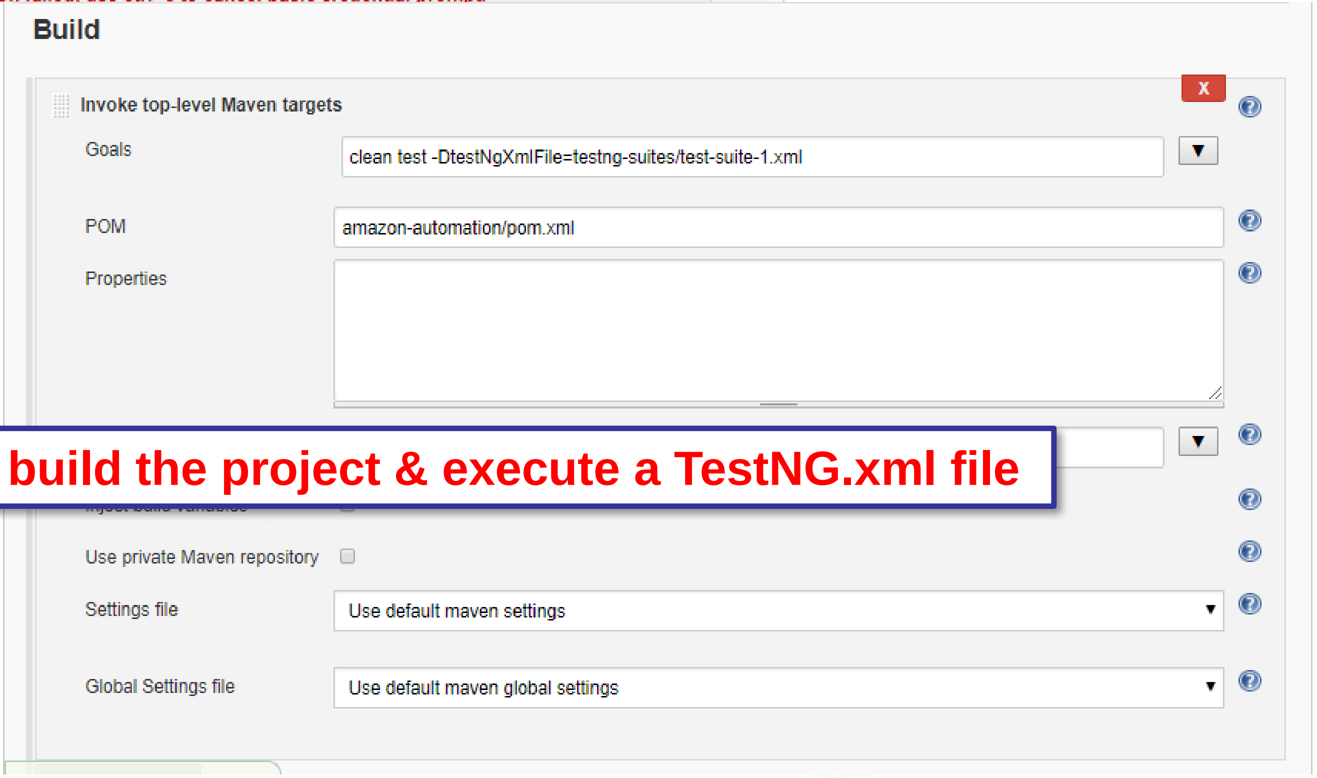
Failed to connect to repository : Command "git.exe ls-remote -h https://github.com/ronyb/jb_jul18.git HEAD" returned status code 128:
stdout:
stderr: Logon failed, use ctrl+c to cancel basic credential prompt.
remote: In
fatal: Auth

Credentials ronyb/******

Branches to build

Branch Specifier (blank for 'any')

1. Pull code from GitHub



Build

Invoke top-level Maven targets

Goals	clean test -DtestNgXmlFile=testng-suites/test-suite-1.xml
POM	amazon-automation/pom.xml
Properties	
Use private Maven repository	☐
Settings file	Use default maven settings
Global Settings file	Use default maven global settings

2. Invoke Maven to build the project & execute a TestNG.xml file

JOHN BRYCE
Leading in IT Education
a matrix company

Thank You!