

CONTINUOUS DELIVERY WITH THE CLOUDBEES JENKINS PLATFORM

USER TRAINING

TOC

- CI & CD, Jenkins Overview and CloudBees Introduction
- Jenkins Jobs/Projects and Setting up Freestyle Projects
- Monitoring and Organizing your jobs
- Plugins Management
- Working with Version Control
- Code Quality and Coverage Metrics
- Parameterized Builds
- Automated Deployment
- Folders & Folders Plus plugins and Security + RBAC plugins
- Playing with Git: Validated Merge Plugin and Pull-Request Builder for GitHub
- Pipeline (formerly known as Workflow)
- Template Plugin
- CloudBees Support
- Introduction to CloudBees Jenkins Operation Center





CLOUDBEES
UNIVERSITY

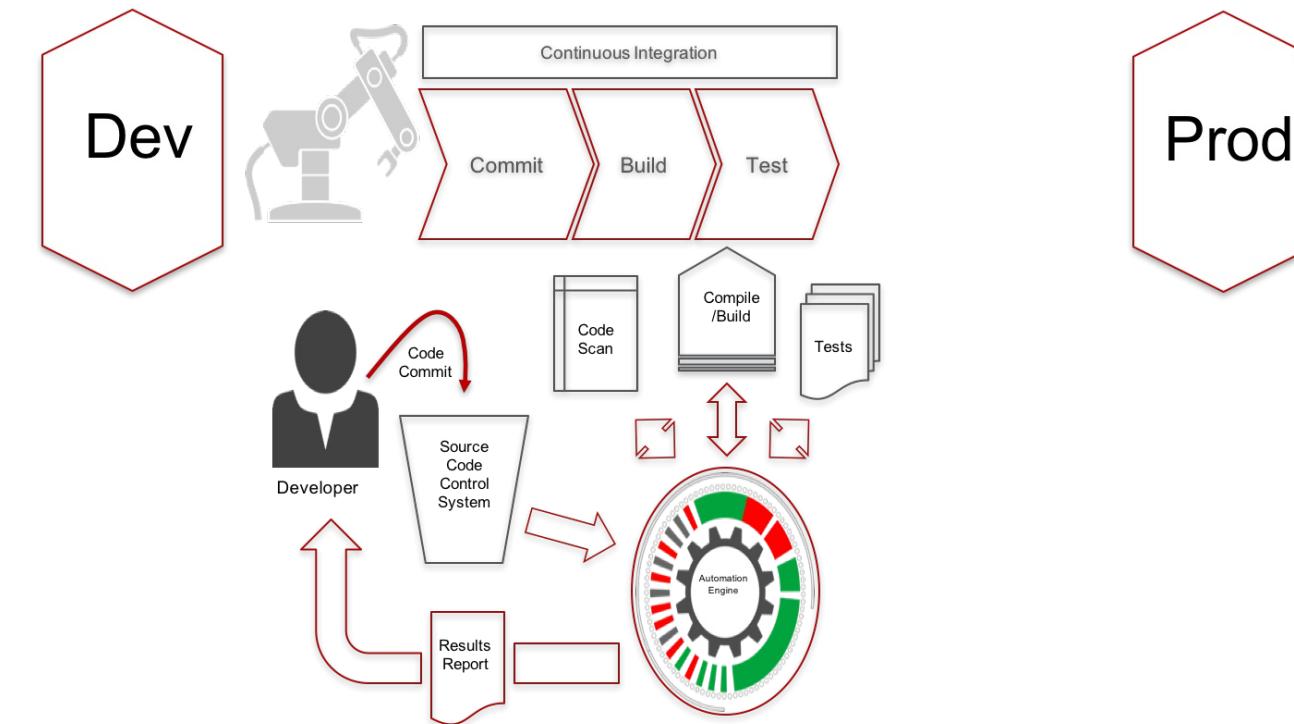
CI / CD FUNDAMENTALS

WHAT IS CONTINUOUS INTEGRATION (CI)?

- Original answer by Martin Fowler:
 - Integrate code often, at least daily, to make integration a non-event
 - Developers follow repository tip closely
- Modern Answer
 - Automate build and test executions of components
 - Build very often (e.g., per commit) to detect regressions quickly
 - Test often (unit and integration tests)
 - Automate regular code quality analysis executions

CONTINUOUS INTEGRATION (CI)

- Automate Build & Test phases
- Continually build and integrate



WHAT IS CONTINUOUS DELIVERY (CD) ?

- While CI lets you automate the software build, scan and test process, CD automates the full application delivery pipeline taking new features and code from development to staging to production
- With automated building, testing, environment provisioning, and deployment, CD becomes the foundation of your DevOps transformation

WHAT IS CONTINUOUS DELIVERY (CD) ?

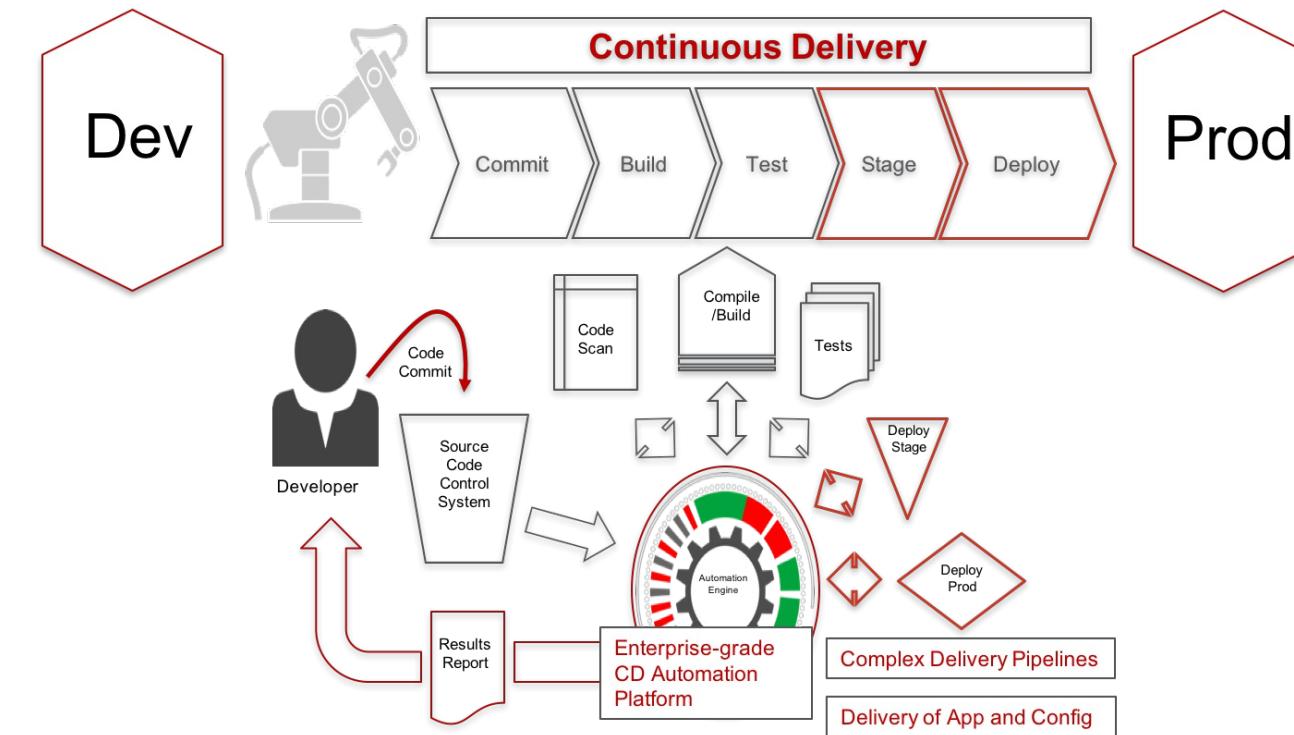
The ability to get:

- Features
- Configuration changes
- Bug fixes
- Experiments

into production or into the hands of users safely & quickly in a sustainable way.

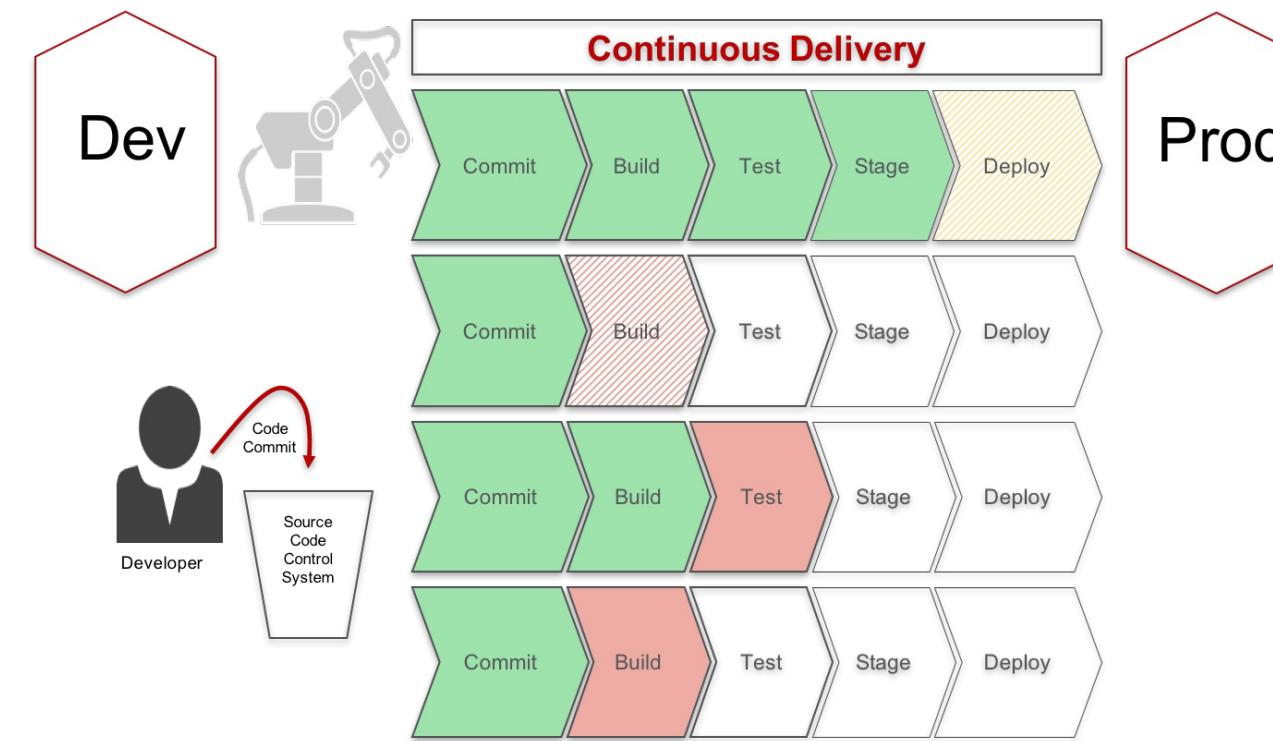
CONTINUOUS DELIVERY (CD)

- CD starts with CI
- It then extends automation across the entire application lifecycle



CONTINUOUS DELIVERY (CD)

- Enables continuous execution and monitoring of the entire software development pipeline



CONTINUOUS DELIVERY (CD), WHY DO IT ?

- Make releases painless, low risk events
- Reduce time to market
- Increase software quality & stability
- Reduce cost of ongoing software development
- Increase customer & employee satisfaction
- Speed up the feedback loop

RELEASE CADENCE

Mary Poppendieck



Tom Poppendieck



How long would it take to your organization to deploy a change
that involves just one single line of code?



**CLOUDBEES
UNIVERSITY**

JENKINS OVERVIEW

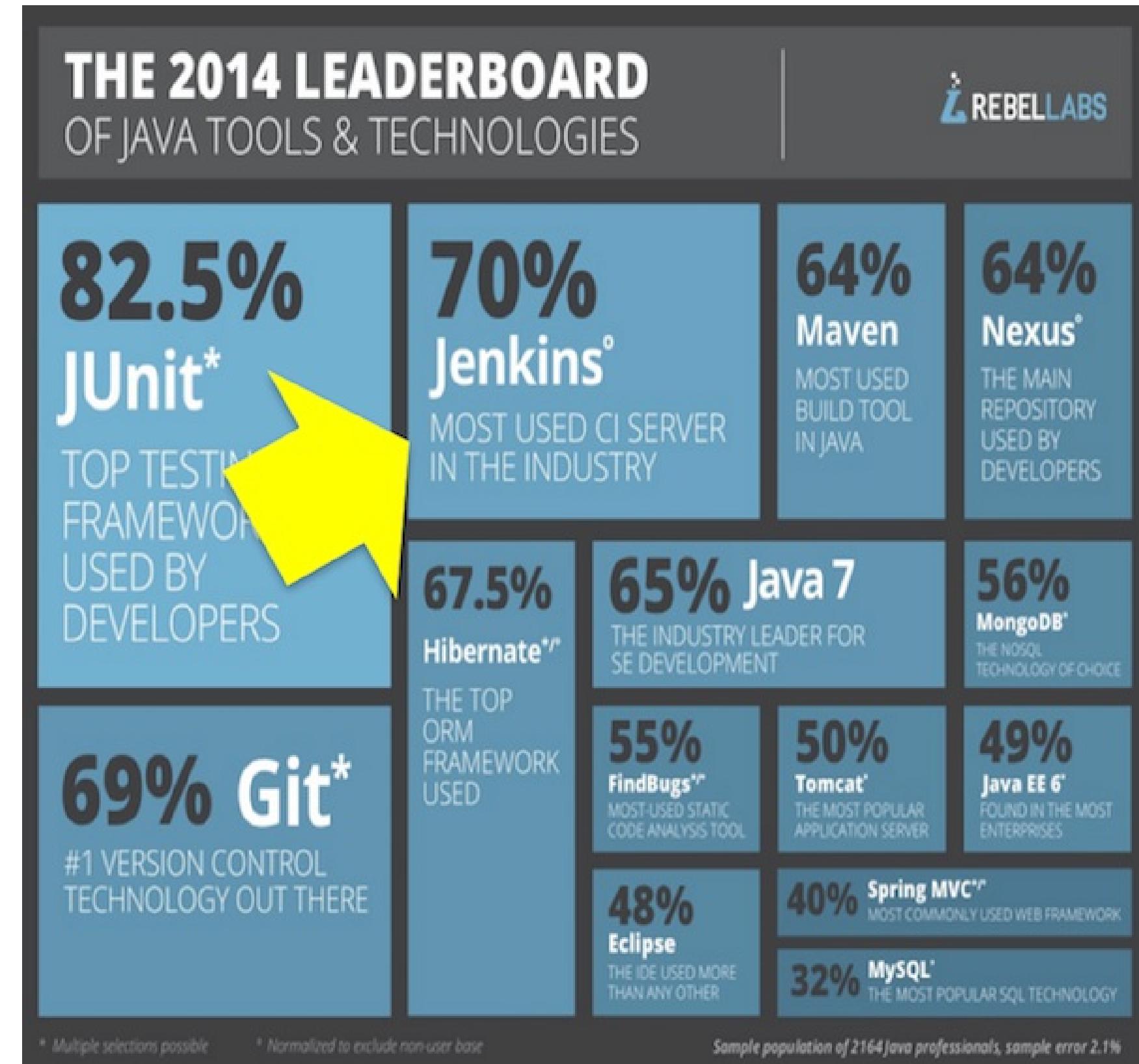
MEET JENKINS...

- #1 Continuous Integration and Delivery server
- Created by Kohsuke Kawaguchi
- An independent and active community (jenkins-ci.org)
- 10 years old
- 500+ releases to date
- 100,000 active installations
- 300,000 Jenkins servers
- 1,200+ plugins



Source: 2014 Java Tools and Technologies Landscape – Rebel Labs

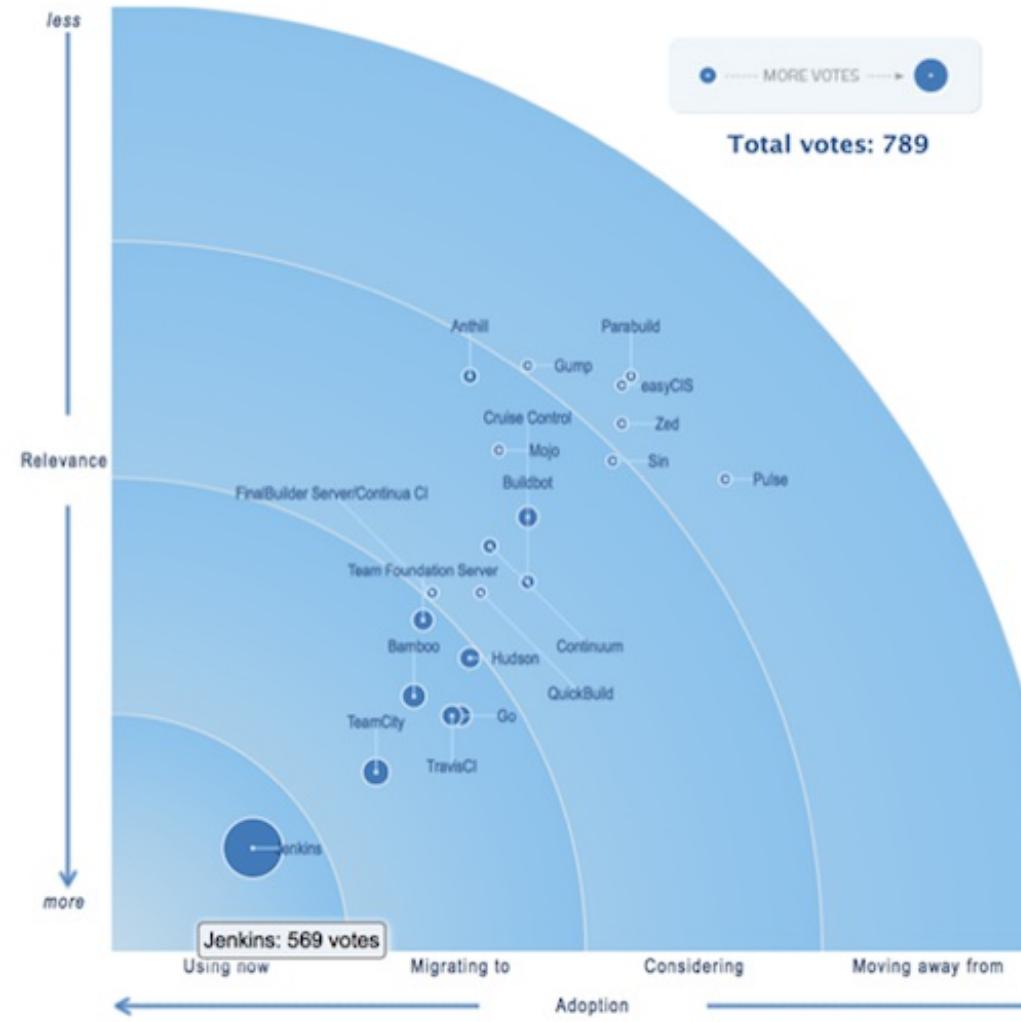






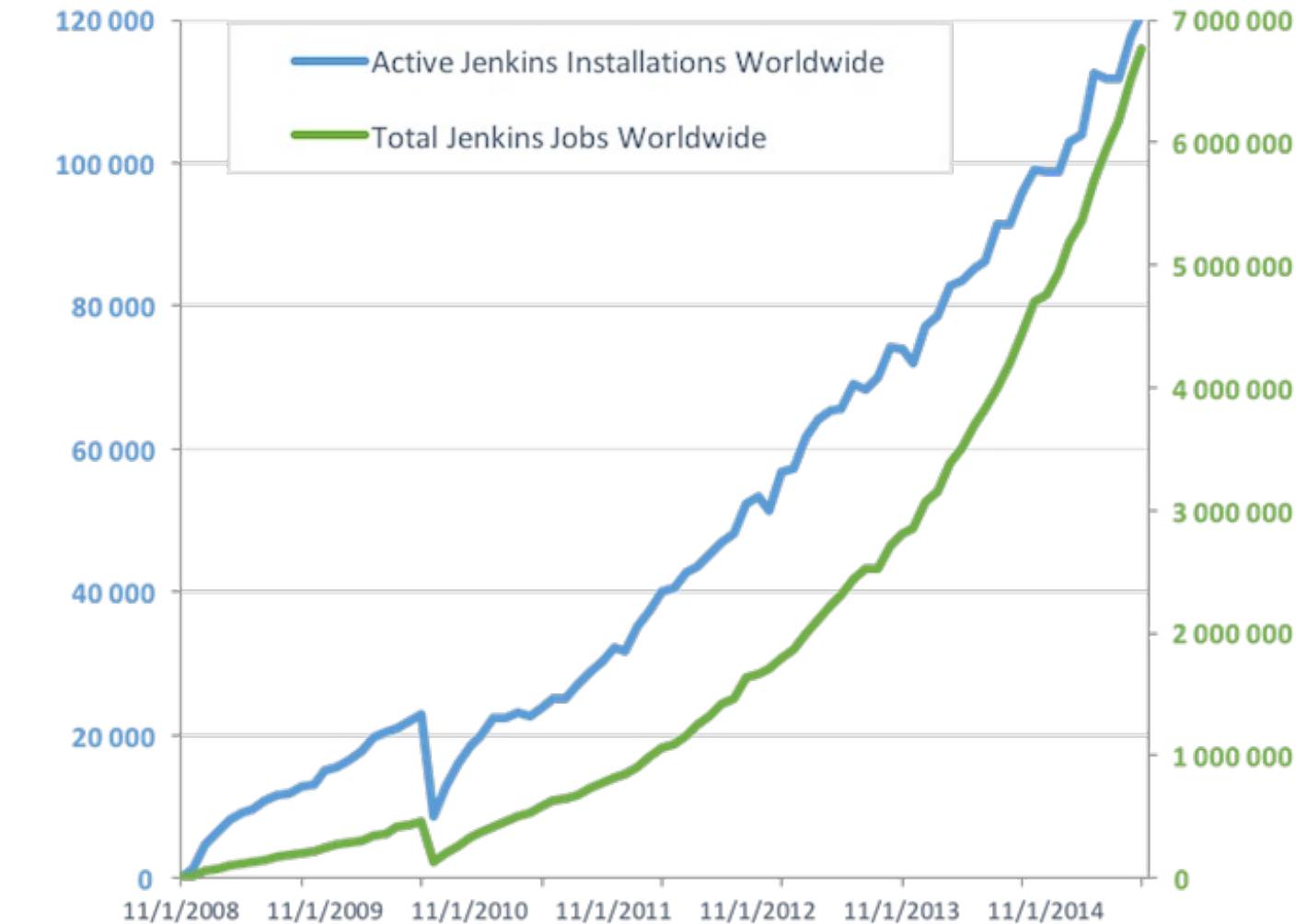
CLOUDBEES
UNIVERSITY

JENKINS POPULARITY: THROUGH THE ROOF



<http://stats.jenkins-ci.org/jenkins-stats>

<http://www.infoq.com/research/ci-server>





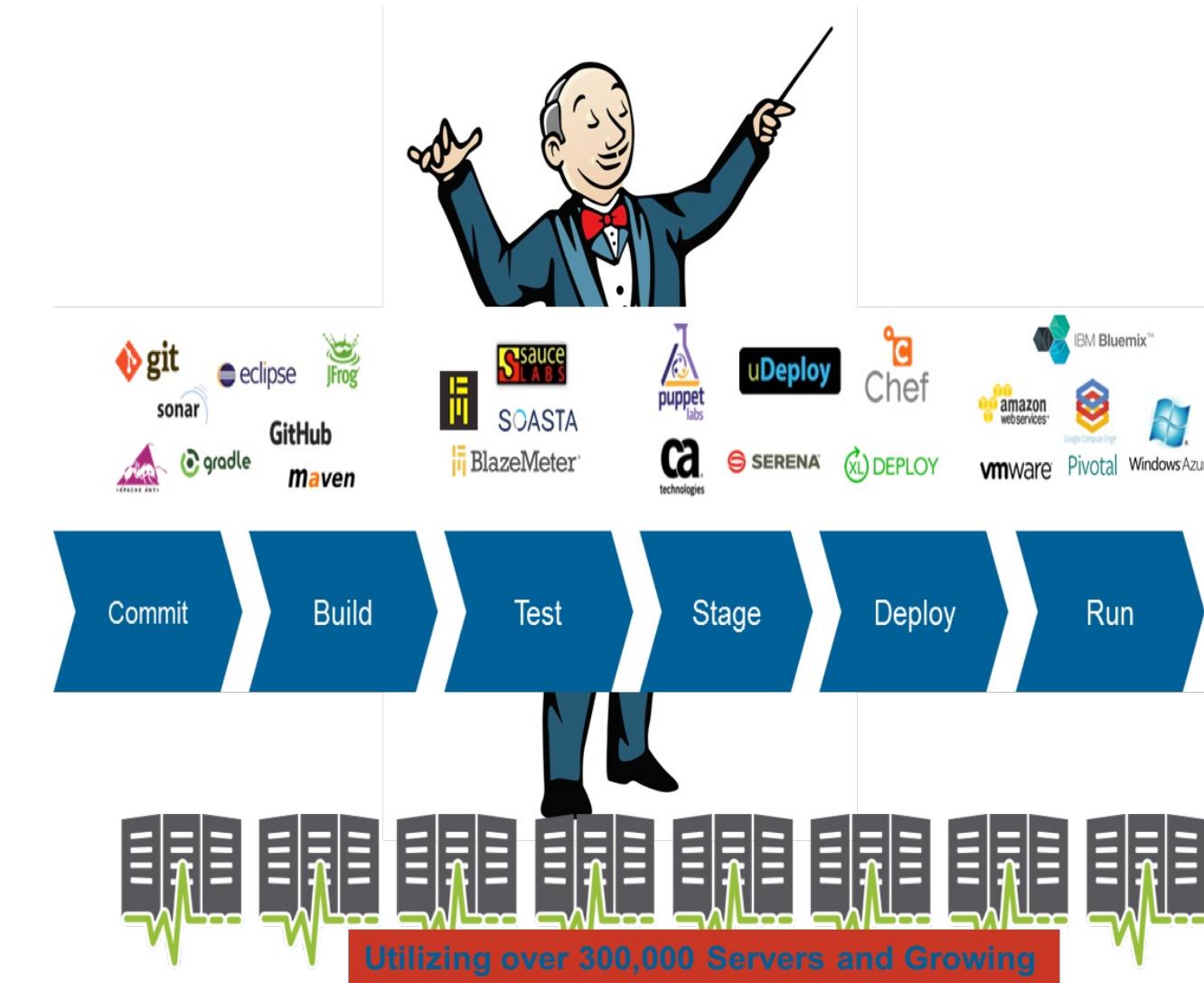
CLOUDBEES
UNIVERSITY

WORLDWIDE ADOPTION





JENKINS IS THE CD ORCHESTRATOR





CLOUDBEES
UNIVERSITY

CLOUDBEES INTRODUCTION

CLOUDBEES AND THE JENKINS COMMUNITY

- Kohsuke Kawaguchi
 - Community leader and CloudBees' CTO
- Code and Releases
 - CloudBees partners with the community on development
 - CloudBees engineers contribute a majority of Jenkins OSS code
 - CloudBees partners with the community on releases
 - CloudBees contributes fixes back to the community
- Produces Jenkins Quarterly Newsletter
- Conducts Jenkins User Conferences



CLOUDBEES INC.

- Incorporated in April 2010
- ~ 200 people as of July 2016
- Headcount and revenue tripled between January 2015 and July 2016
- Presence in 14 countries
- Offices:
 - USA:
 - Raleigh, Richmond, San Jose
- Europe:
 - Sevilla (Spain), Reading (UK), Neuchatel (Switzerland)

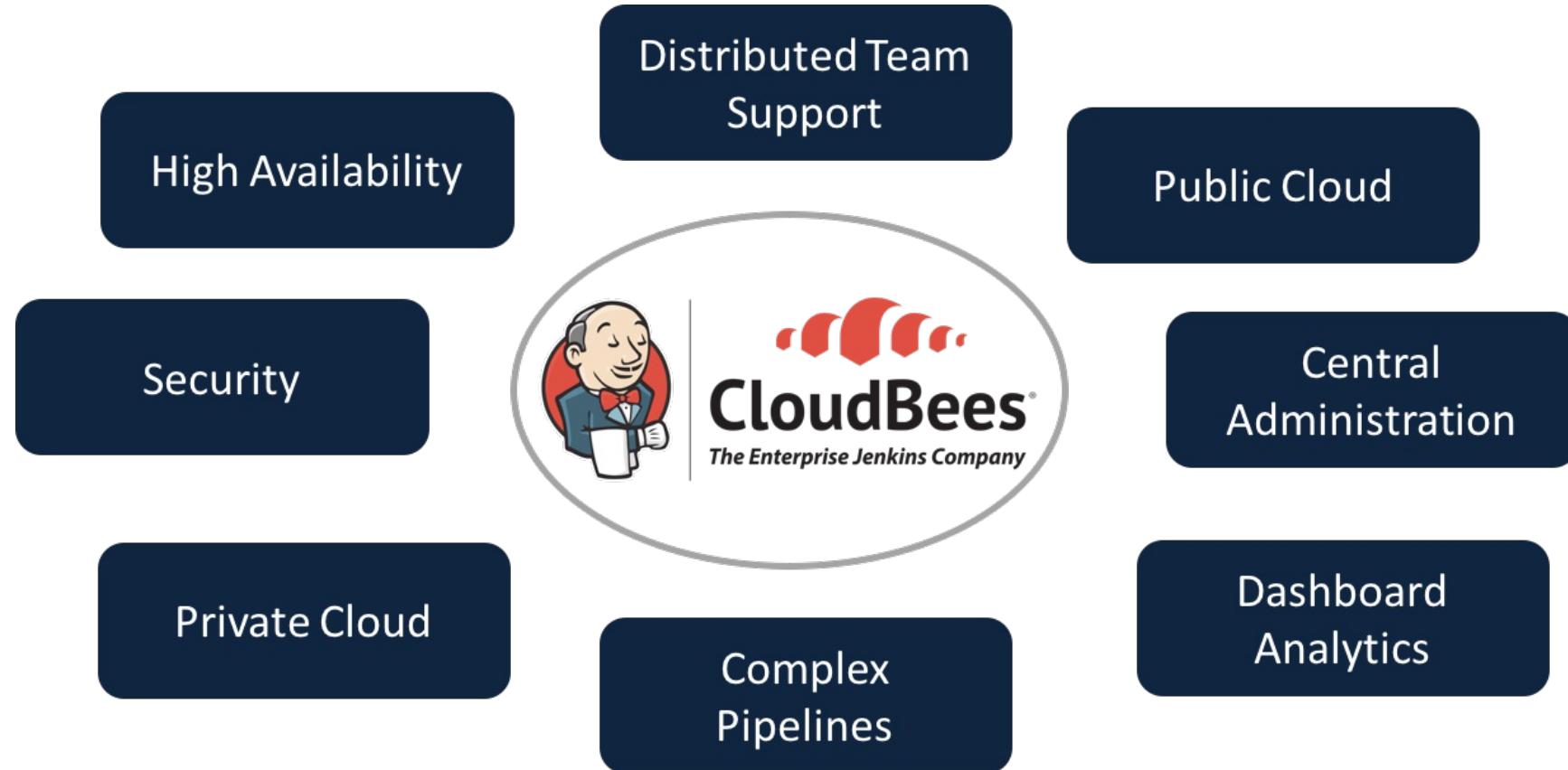


**CLOUDBEES
UNIVERSITY**

FOLLOW-THE-SUN 24/7 SUPPORT LOCATIONS



WE DELIVER JENKINS @ ENTERPRISE SCALE





CLOUDBEES JENKINS PLATFORM

Enterprise Edition

Scalable, Secure and Manageable

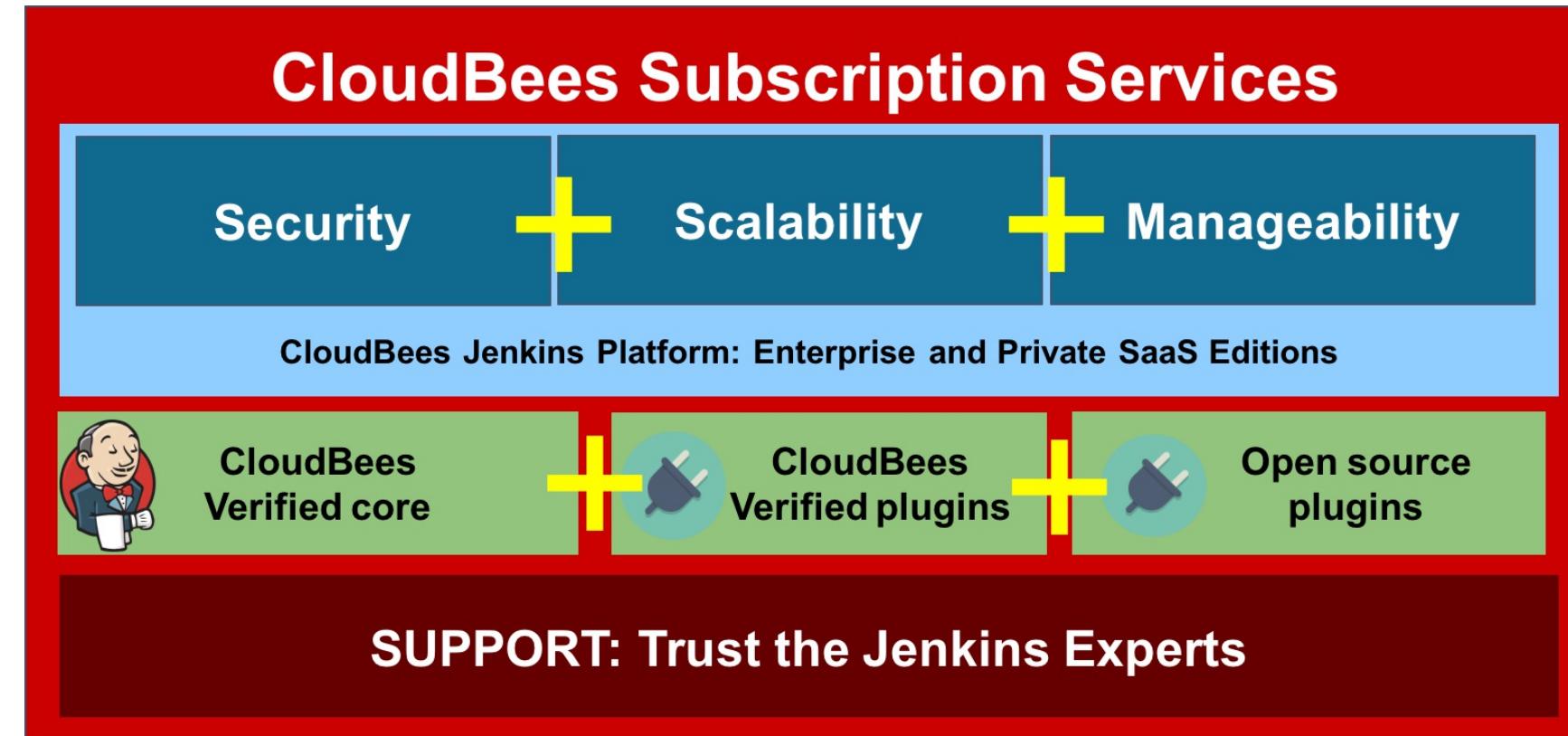
- High Availability
- Cluster Operations - multi-master environment control
- Shared Build Nodes
- Role Based Access Control and single sign-on
- Pipeline management
- Templates to automate best practices
- Analytics and actionable information

Private SaaS Edition

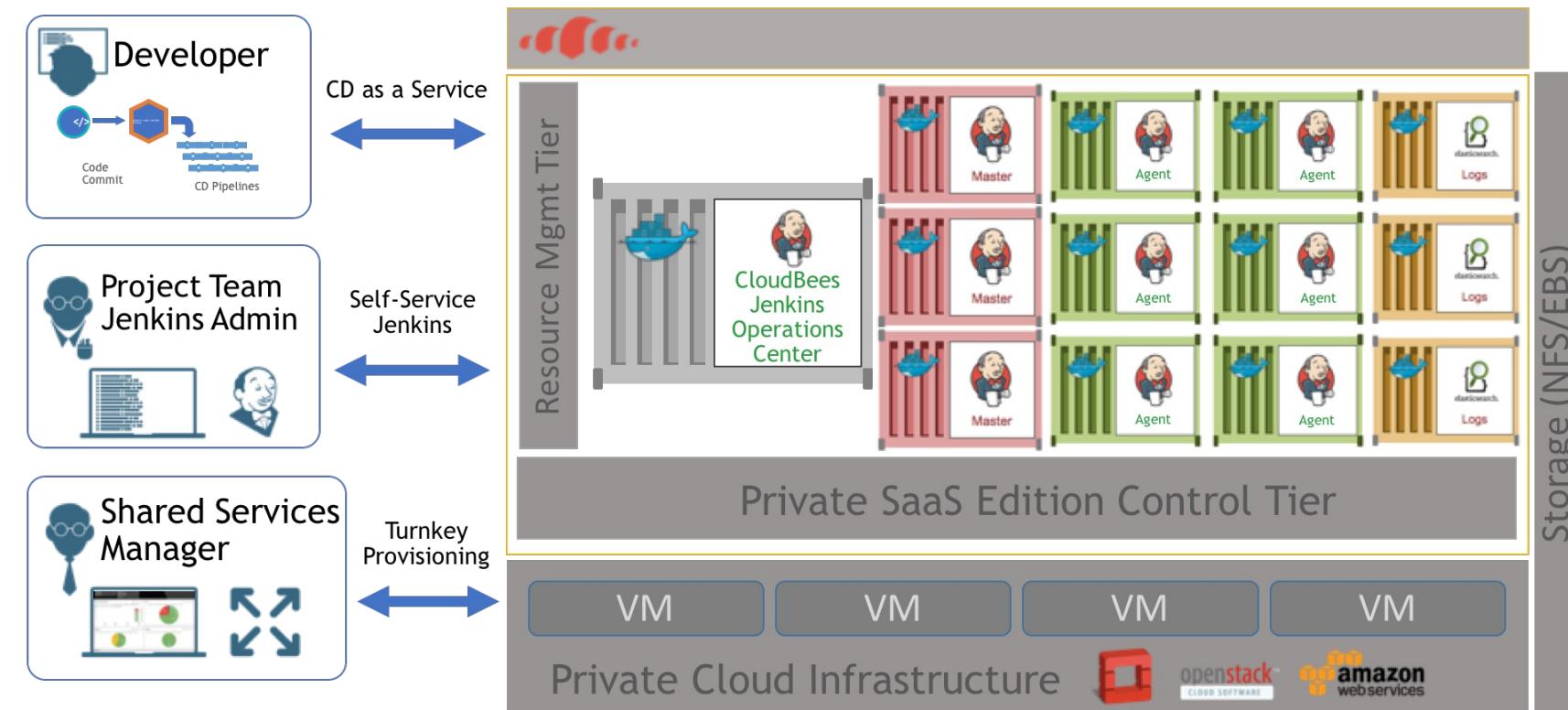
Continuous Delivery as a Service

- Deploy on your private cloud or VPC
- Federated, shared services across projects
- Deploy new Jenkins services in seconds
- Resilient cloud-based HA/DR
- **Automated storage backups**
- Optimize infrastructure cost & usage
- Operational visibility & analysis
- Oversight of total workflow & complex pipelines

CLOUDBEES JENKINS PLATFORM: ENTERPRISE EDITION



CLOUDBEES JENKINS PLATFORM: PRIVATE SAAS EDITION



JENKINS SERVICES FROM CLOUDBEES



Jenkins Certified Training



Jenkins Certification



Jenkins Consulting



Jenkins Newsletter



Jenkins User Conferences

CLOUDBEES SUBSCRIPTION MODEL

- CloudBees sells subscriptions that entitle you to receive support for:
 - CloudBees Jenkins Platform
 - OSS Jenkins including 1,200+ Open Source Plugins
- Customers are charged an annual subscription fee per installation
 - Multi-year subscriptions are available
- A typical service subscription includes:
 - Software updates, bug fixes, and upgrades
 - Technical support
 - stable versions, stable APIs, and more

CLOUDBEES RESOURCES

- Customer Engagement
 - Support
 - Knowledge Base
 - Diagnostics
- Professional Services
 - Architecture Assessment & CD Guidance
 - Bootstrap Implementation Services
 - Migration Assistance
- Certified Partners
 - Training
 - DevOps transformation
 - DevOps integration
 - Custom Development



JENKINS JOBS/PROJECTS

WHAT IS A JENKINS JOB/PROJECT

- Jobs are at the heart of the Jenkins build process.
- A Jenkins job is most of the time a particular task or step in your build process.
- An application may require several jobs
- A single job may have multiple build steps

KEY JENKINS PROJECT (JOB) TYPES

- Pipeline Project (Brand New!)
- Freestyle Project
- Maven Project
- Monitor External Job
- Several more types supported

FREESTYLE PROJECTS

- Most flexible
- Build any type of projects (Ant, Maven, Makefile, Shell script...)
- Supports most reporting plugins

MAVEN PROJECTS

- Optimized for Maven 2/3 projects
- Jenkins reads the pom.xml file to make setting up the project easier.
- Build projects in the correct order based on the Maven dependencies
- Understands multi-module projects
- Default reporting of tests and code analysis tools

SETTING UP FREESTYLE PROJECTS

NEW FREESTYLE PROJECT: GENERAL INFO

- Discard old build data
 - Useful to cap the disk consumption
 - You can mark some to be kept forever
- Configure specific tools for the build to run with
 - Specify JDK
 - Specify Maven/Ant versions

NEW FREESTYLE PROJECT: BUILD STEPS

- Build steps do the actual work
- Out of the box, many types available:
 - Unix shell or Windows batch script
 - Invoke an Ant script
 - Invoke a Maven script
- You can have several build steps
 - A shell script followed by an Ant or Maven build
 - An Ant build followed by a Maven build
- Can be reordered via Drag & Drop
- Plugins can add new build steps

ENVIRONMENT VARIABLES

- Jenkins also sets environment variables, eg.
 - `BUILD_NUMBER` (eg “123”)
 - `JOB_NAME` (eg. “tax-api-unit-tests”)
 - `BUILD_TAG` (eg. “Jenkins-tax-api-unit-tests-123”)
 - `SVN_REVISION` or `CVS_BRANCH`
- You can use these in your build script

POST-BUILD ACTIONS

- Collect the results of builds
 - Archiving
 - Reporting
 - Trigger other builds
 - Notifications

POST-BUILD ACTION: ARCHIVING

- Store build artifacts for future reference
- Use Ant-style paths `build/**/*.jar` to determine what files to archive Stored artifacts are displayed on a build's page

POST-BUILD ACTION: JUNIT REPORTS

- The build needs to run the tests and generate an XML JUnit report
- Use Ant-style path expressions
- Results appear on the project home page

POST-BUILD ACTION: JAVADOC API

- The build needs to generate Javadoc
- Jenkins will capture javadoc
- Javadoc link appears on the build home page

POST-BUILD ACTION: E-MAIL NOTIFICATION

- Email notification strategies
 - Sent when a build fails or is deemed “unstable”
 - And on the first successful build after a series of failures
- To fine-tune the behaviour, use “email-ext”

POST-BUILD ACTION: TRIGGER OTHER BUILDS

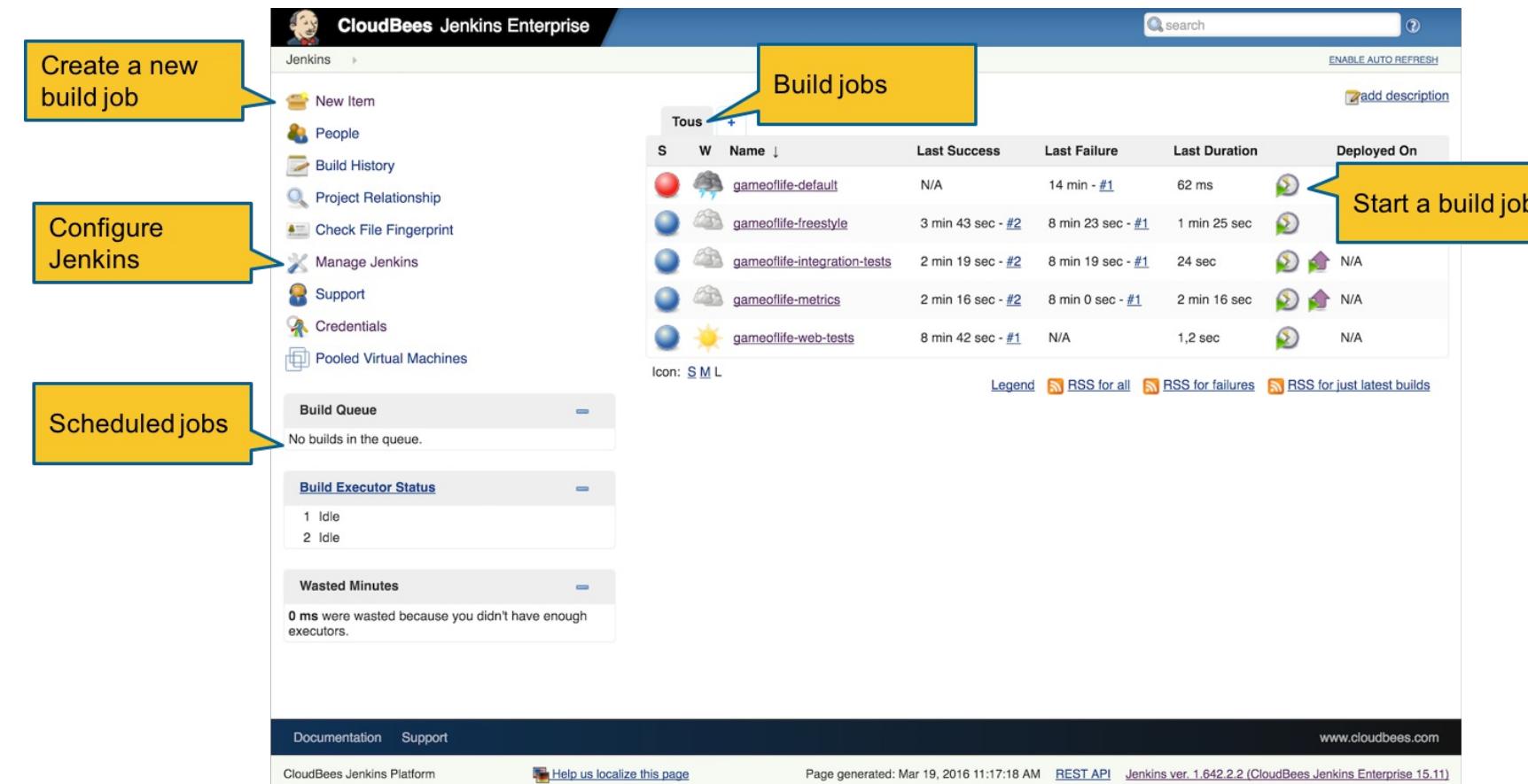
- Specify other builds to be started when this build succeeds
- Useful for cascading builds
- Different view of the same data as “build trigger”

LAB EXERCISES:

- [Lab 1: Creating A Freestyle build Job](#)
- [Lab 2: Creating A Maven build Job](#)

MONITORING BUILD JOBS

THE JENKINS DASHBOARD



The screenshot shows the CloudBees Jenkins Enterprise dashboard. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins', 'Support', 'Credentials', and 'Pooled Virtual Machines'. Below the sidebar are sections for 'Build Queue' (No builds in the queue), 'Build Executor Status' (1 Idle, 2 Idle), and 'Wasted Minutes' (0 ms wasted). The main area is titled 'Build jobs' and lists five build jobs: 'gameoflife-default', 'gameoflife-freestyle', 'gameoflife-integration-tests', 'gameoflife-metrics', and 'gameoflife-web-tests'. Each job entry includes columns for Status (red circle), Last Success, Last Failure, Last Duration, and Deployed On. A yellow callout box labeled 'Create a new build job' points to the 'New Item' link in the sidebar. Another yellow callout box labeled 'Configure Jenkins' points to the 'Manage Jenkins' link. A third yellow callout box labeled 'Scheduled jobs' points to the 'Build Queue' section. A fourth yellow callout box labeled 'Start a build job' points to the 'Deployed On' column for the 'gameoflife-web-tests' job.

S	W	Name	Last Success	Last Failure	Last Duration	Deployed On
🔴	⚡	gameoflife-default	N/A	14 min - #1	62 ms	⌚ N/A
🔵	⚡	gameoflife-freestyle	3 min 43 sec - #2	8 min 23 sec - #1	1 min 25 sec	⌚ N/A
🔵	⚡	gameoflife-integration-tests	2 min 19 sec - #2	8 min 19 sec - #1	24 sec	⌚ N/A
🔵	⚡	gameoflife-metrics	2 min 16 sec - #2	8 min 0 sec - #1	2 min 16 sec	⌚ N/A
🔵	☀️	gameoflife-web-tests	8 min 42 sec - #1	N/A	1,2 sec	⌚ N/A



MONITORING BUILD JOBS

- What is the last build result?

The screenshot shows the CloudBees Jenkins Enterprise dashboard. On the left, there's a sidebar with various links: New Item, People, Build History, Project Relationship, Checks, Manage, Support, Credentials, and Pooled Virtual Machines. Below the sidebar, there are three collapsed sections: Build Queue (No builds in the queue), Build Executor Status (1 Idle, 2 Idle), and Wasted Minutes (0 ms wasted). The main area features a table titled "Tous" (All) with columns: S (Status), W (Waste), Name, Last Success, Last Failure, Last Duration, and Deployed On. A yellow callout box points to the first row of the table, which shows a red circle icon and the text "Failed meoflife-default". To the right of the table is a legend box with three entries: "Build successful" (blue circle), "Build unstable" (yellow circle), and "Build failed" (red circle). At the bottom of the dashboard, there's a footer with links for Documentation, Support, and www.cloudbees.com, along with a note about the page being generated on March 19, 2016.

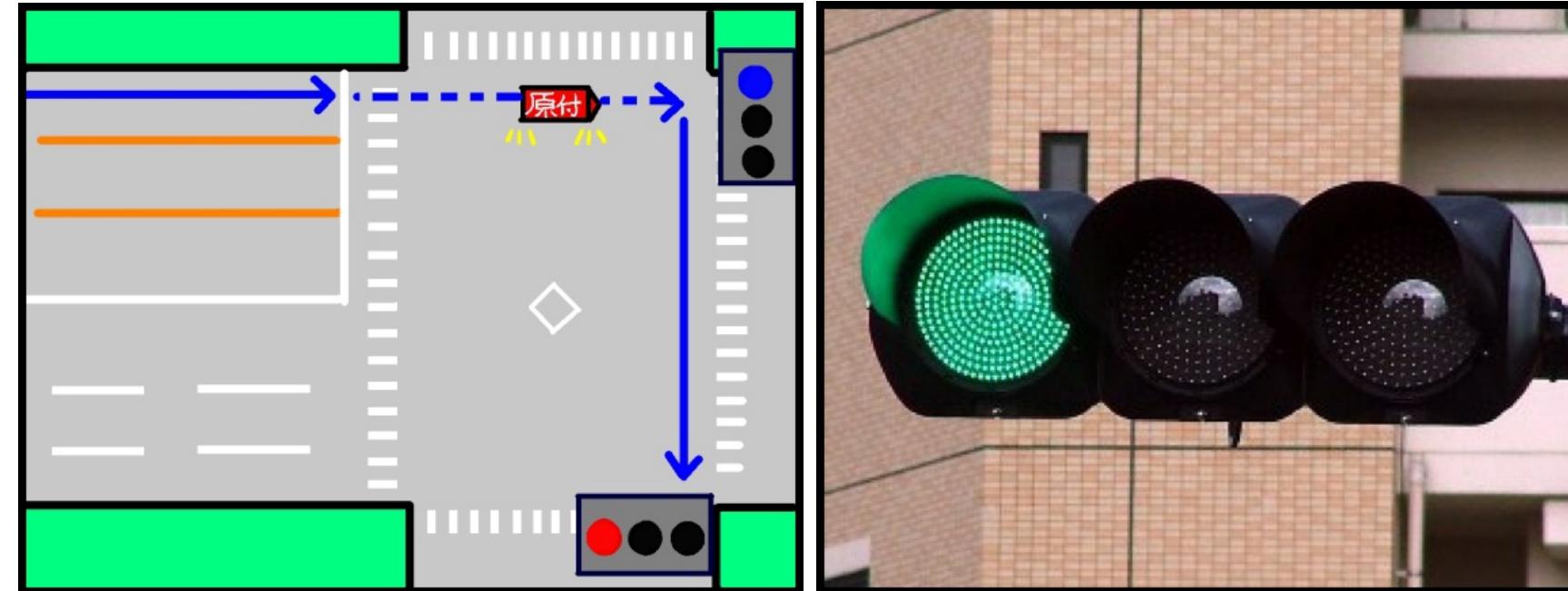
S	W	Name	Last Success	Last Failure	Last Duration	Deployed On
		Failed meoflife-default	N/A	14 min - #1	62 ms	
		gameoflife-free			1 min 25 sec	
		gameoflife-inte			24 sec	
		gameoflife-me			2 min 16 sec	
		gameoflife-wel			1,2 sec	

Legend: RSS for all | RSS for failures | RSS for just latest builds



WHY BLUE?

- In Japan, it is really called "blue light"
 - So people draw them as such
 - Even though the actual color is green!



BENEFITS: QUICK OVERVIEW

- **KISS** principle:
 - Color for last build status
 - Weather for the build status trend
- One screen to see them all



MONITORING BUILD JOBS

- How my jobs are behaving for the last 5 builds?

The screenshot shows the CloudBees Jenkins Enterprise dashboard. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', etc. The main area displays a table of build jobs:

Tous	S	W	Name ↓	Last Success	Last Failure	Last Duration	Deployed On
	●	●	gameoflife-default	[cloud icon]	[cloud with rain icon]	62 ms	N/A
	●	●	gameoflife-freestyle	[cloud icon]	[cloud with rain icon]	1 min 25 sec	N/A
	●	●	gameoflife-integration-	[cloud icon]	[cloud with rain icon]	24 sec	N/A
	●	●	gameoflife-metrics	[cloud icon]	[cloud with rain icon]	2 min 16 sec	N/A
	●	●	gameoflife-web-tests	[sun icon]	[sun icon]	1,2 sec	N/A

A large callout box highlights the 'gameoflife-default' job, which has a red icon. The callout box contains the text 'Build unstable' at the top and 'Build stable' at the bottom, with a large green arrow pointing downwards.

At the bottom of the dashboard, there are sections for 'Build Queue' (No builds in the queue), 'Build Executor Status' (1 Idle, 2 Idle), and 'Wasted Minutes' (0 ms wasted).

Footer information includes: Documentation, Support, www.cloudbees.com, CloudBees Jenkins Platform, Help us localize this page, Page generated: Mar 19, 2016 11:17:18 AM, REST API, Jenkins ver. 1.642.2.2 (CloudBees Jenkins Enterprise 15.11).





MONITORING BUILD JOBS

- What jobs are currently executed?

The screenshot shows the CloudBees Jenkins Enterprise dashboard. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', etc. The main area displays a table of build jobs:

S	W	Name	Last Success	Last Failure	Last Duration	Deployed On
●	⚡	gameoflife-default	N/A	14 min - #1	62 ms	N/A
●	⚡	gameoflife-freestyle	3 min 43 sec - #2	8 min 23 sec - #1	1 min 25 sec	N/A
●	⚡	gameoflife-integration-tests	2 min 19 sec - #2	8 min 19 sec - #1	24 sec	N/A
●	⚡	gameoflife-metrics	2 min 16 sec - #2	8 min 0 sec - #1	2 min 16 sec	N/A
●	☀	gameoflife-web-tests	8 min 42 sec - #1	N/A	1,2 sec	N/A

Below the table, there's a 'Build Queue' section stating 'No builds in the queue.' and a 'Build Executor Status' section showing '1 gameoflife-freestyle #10' and '2 Idle'. A yellow callout box points to the progress bar of the first executor entry with the text: 'Click on progress bar to jump to the console'.

This screenshot shows the same dashboard as above, but with a different focus. It highlights the 'Build Executor Status' section, which now shows '1 gameoflife-freestyle #10' and '2 Idle'. A yellow callout box points to the progress bar of the first executor entry with the text: 'Click on progress bar to jump to the console'.

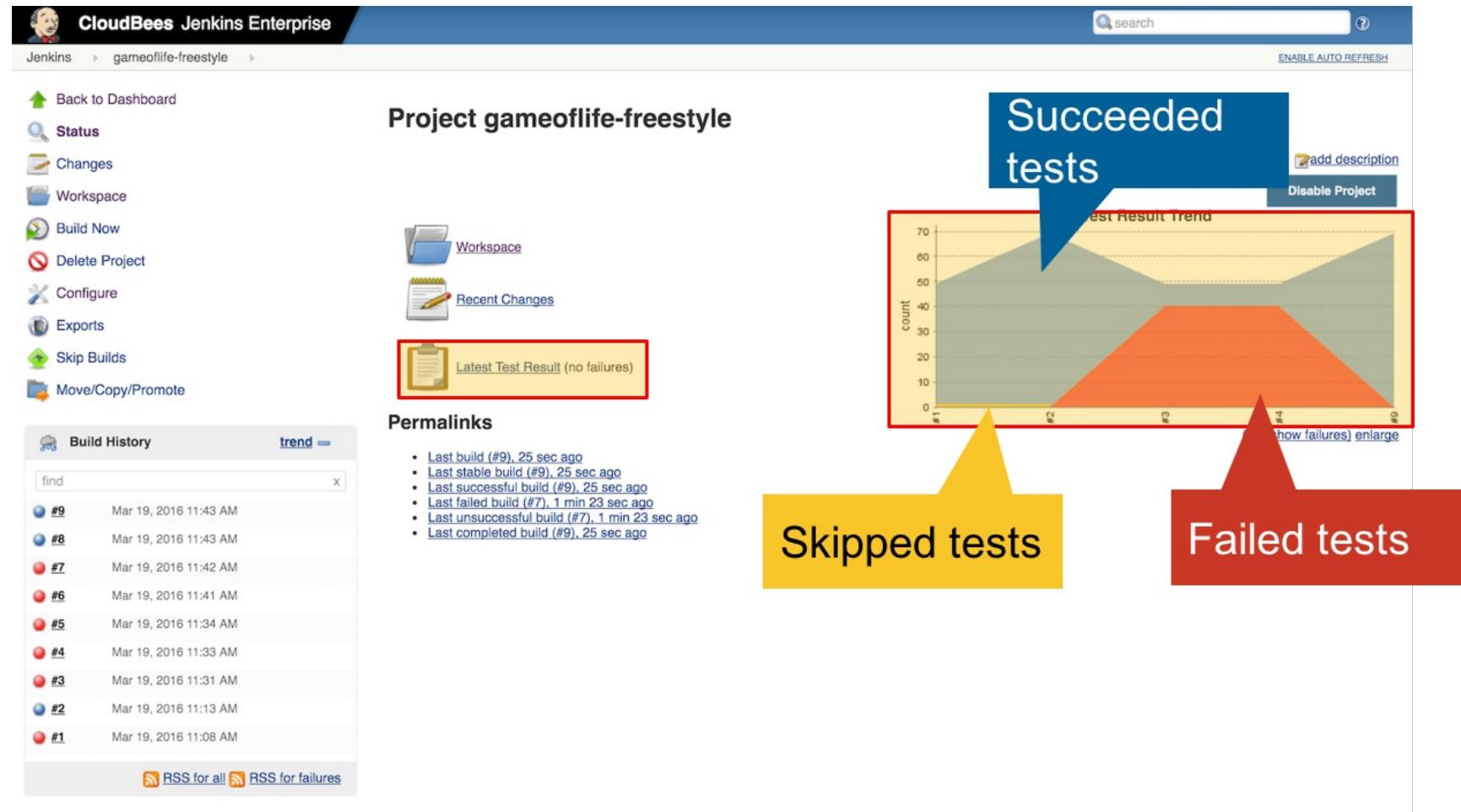
The footer of the page includes links for 'Documentation' and 'Support', the URL 'www.cloudbees.com', and a 'Help us localize this page' button. It also shows the page was generated on Mar 19, 2016 at 11:17:18 AM, and provides REST API and Jenkins version information.





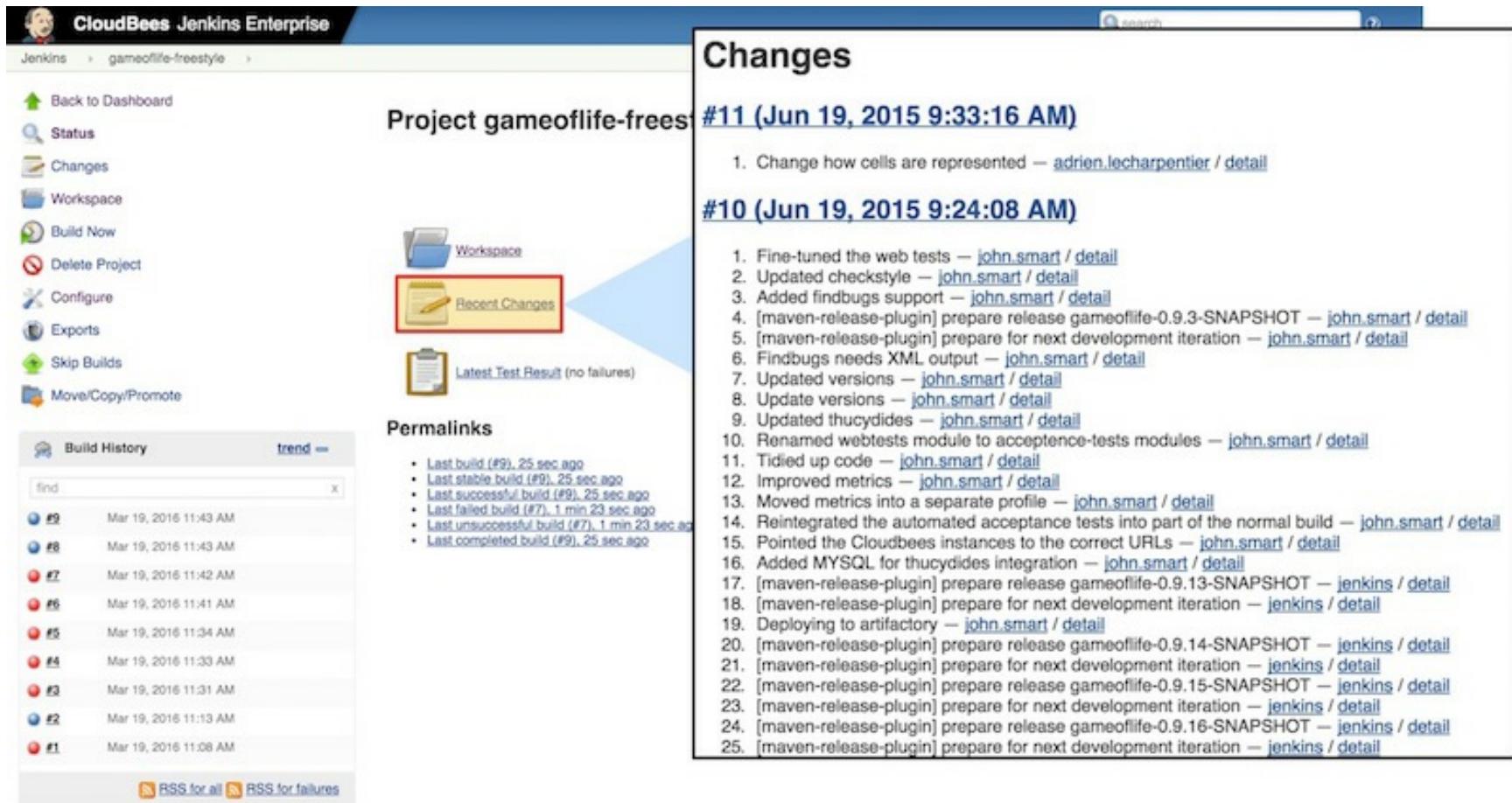
BUILD DETAILS

- See the tests results:



BUILD DETAILS

- See what has changed:



The screenshot shows the Jenkins interface for the project "gameoflife-freestyle". On the left, there's a sidebar with links like Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, Configure, Exports, Skip Builds, and Move/Copy/Promote. Below that is the "Build History" section, which lists builds from Mar 19, 2016, at 11:43 AM (build #9) to Mar 19, 2016, at 11:08 AM (build #1). The "Recent Changes" link is highlighted with a yellow box. The main content area shows the "Changes" page for build #11, dated Jun 19, 2015, at 9:33:16 AM. It lists 25 changes made by users like adrien.lecharpentier and john.smart, including updates to workspace, checkstyle, findbugs support, and various Maven release plugin configurations.



BUILD DETAILS

- See what has changed:

The screenshot shows the Jenkins interface for a job named "gameoflife-freestyle". The main content area is titled "Changes" and displays the log for build #11, which was triggered on Jun 19, 2015, at 9:33:16 AM. The log entries are as follows:

- Commit 12b8d87ad7543eb67e333889d6af8f4ae3b0401b by john.smart
- Fine-tuned the web tests
- [gameoflife-webtests/src/test/java/com/wakaleo/gameoflife/webtests/WhenTheUserEntersAnInitialGrid.java]
- [gameoflife-webtests/src/test/java/com/wakaleo/gameoflife/webtests/WhenTheUserSpawnsTheNextGeneration.java]
- [gameoflife-webtests/src/test/java/com/wakaleo/gameoflife/webtests/WhenTheUserGoesToTheHomePage.java]
- [naminglife-webtests/nom.xml]

Below the changes, there is a "Permalinks" section with links to various builds:

- Last build (#9), 25 sec ago
- Last stable build (#9), 25 sec ago
- Last successful build (#9), 25 sec ago
- Last failed build (#7), 1 min 23 sec ago
- Last unsuccessful build (#7), 1 min 23 sec ago
- Last completed build (#9), 25 sec ago

On the left side of the screen, there is a sidebar with options like "Exports", "Skip Builds", and "Move/Copy/Promote". Below the sidebar, there is a "Build History" section showing a list of builds from Mar 19, 2016, at 11:43 AM to Mar 19, 2016, at 11:08 AM. The history table includes columns for status, build number, and timestamp.

BUILD DETAILS

- Download latest archived artifact:



The screenshot shows the Jenkins interface for a build of the 'gameoflife-web' project. The build number is #3, and it was run on Mar 19, 2016, at 11:53:31 AM. A red box highlights the 'Build Artifacts' section, which contains two files: 'gameoflife-web-0.0.68.pom' (6.44 KB) and 'gameoflife-web-0.0.68.war' (3.27 MB). Below this, a note says 'No changes. Changes in dependency'. The sidebar on the left includes links for Back to Project, Status, Changes, Console Output, Edit Build Information, Delete Build, Executed Mojos, Deploy Now, Test Result, See Fingerprints, Redeploy Artifacts, and Previous Build.



BUILD DETAILS

- Build history:

CloudBees Jenkins Enterprise

Jenkins > gameoflife-freestyle >

search ENABLE AUTO REFRESH

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build Now](#)

[Delete Project](#)

[Configure](#)

[Exports](#)

[Skip Builds](#)

[Move/Copy/Promote](#)

Project gameoflife-freestyle

[Workspace](#)

[Recent Changes](#)

[Latest Test Result \(no failures\)](#)

[add description](#)

[Disable Project](#)

Test Result Trend

count

#1 #2 #3 #4 #5

(just show failures) [enlarge](#)

Build History

trend =

find

- Last build (#9), 25 sec ago
- Last stable build (#9), 25 sec ago
- Last successful build (#9), 25 sec ago
- Last failed build (#7), 1 min 23 sec ago
- Last unsuccessful build (#7), 1 min 23 sec ago
- Last completed build (#9), 25 sec ago

#9 Mar 19, 2016 11:43 AM

#8 Mar 19, 2016 11:43 AM

#7 Mar 19, 2016 11:42 AM

#6 Mar 19, 2016 11:41 AM

#5 Mar 19, 2016 11:34 AM

#4 Mar 19, 2016 11:33 AM

#3 Mar 19, 2016 11:31 AM

#2 Mar 19, 2016 11:13 AM

#1 Mar 19, 2016 11:08 AM

[RSS for all](#) [RSS for failures](#)





BUILD DETAILS

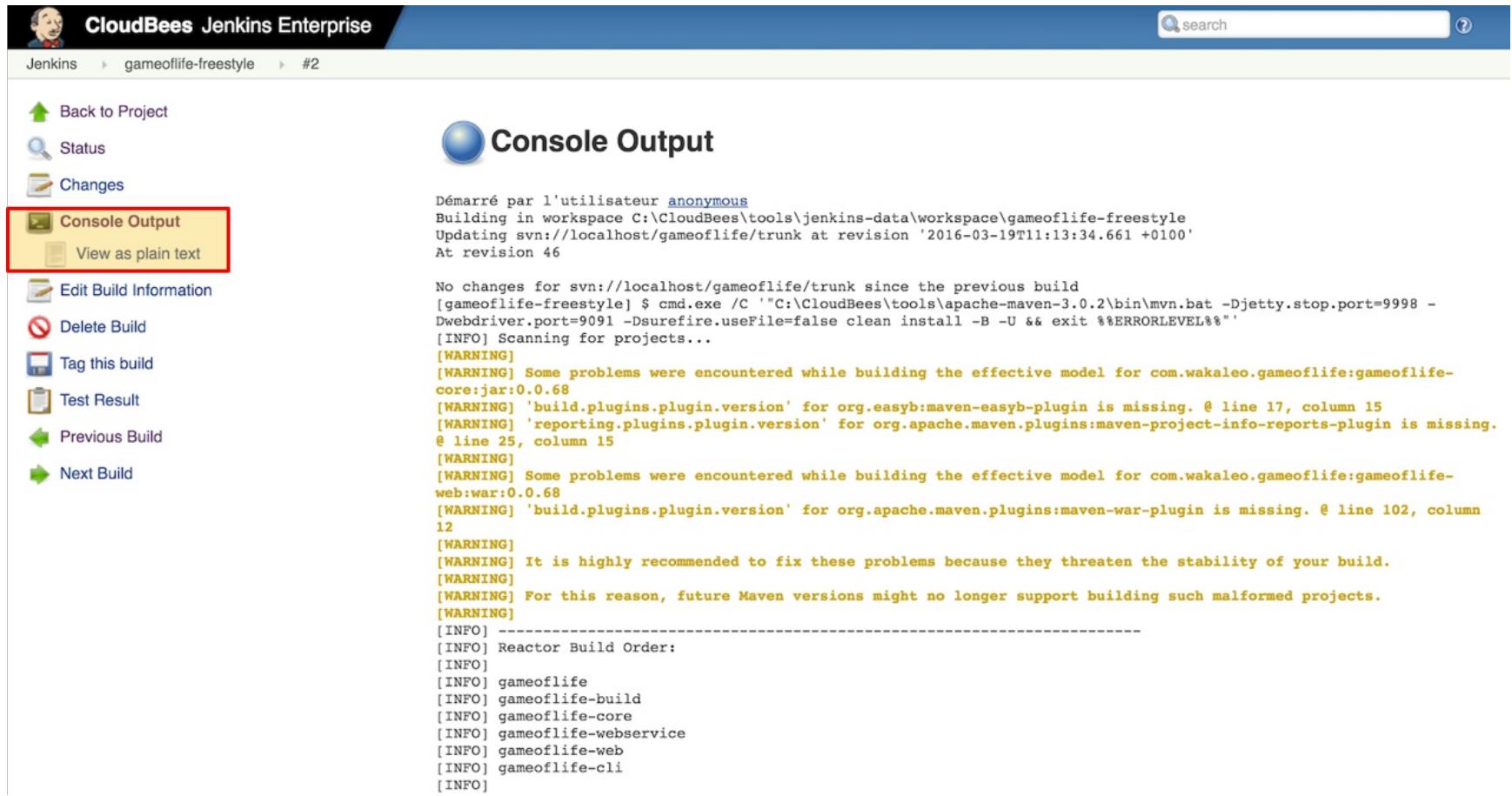
- View the details of any particular build:

The screenshot shows the Jenkins build details page for build #3 of the gameoflife-default project. The page is titled "Build #3 (Mar 19, 2016 11:53:26 AM)". Key sections include:

- Build number and date:** Build #3 (Mar 19, 2016 11:53:26 AM)
- Who / what triggered this build:** Started by anonymous user
- Test results:** Test Result (no failures)
- Module Builds:**
 - gameoflife: 0,81 sec
 - gameoflife-build: 1,4 sec
 - gameoflife-cli: 0,34 sec
 - gameoflife-core: 3,1 sec
 - gameoflife-web: 15 sec
 - gameoflife-webservice: 0,36 sec
- Downstream Builds:**
 - gameoflife-integration-tests: #4
 - gameoflife-metrics: (none)
- The changes of this build:** Revision: 48 Changes
 - 1. Reverting Dead Cell code ([detail/Sventon 2.x](#))
- Build timings:** This run spent:
 - 30 ms waiting in the queue;
 - 31 sec building on an executor;
 - 31 sec total from scheduled to completion.
- Module dependencies (Maven):** (List of dependencies shown in the screenshot)

BUILD DETAILS

- View the output of the build:



The screenshot shows the Jenkins interface for a project named "gameoflife-freestyle" under build #2. On the left, there's a sidebar with links: Back to Project, Status, Changes, **Console Output** (which is highlighted with a red box), View as plain text, Edit Build Information, Delete Build, Tag this build, Test Result, Previous Build, and Next Build. The main content area is titled "Console Output". It displays the Maven build logs. The logs show the build was started by an anonymous user and was updated from revision 45 to 46. It notes no changes since the previous build and lists several warning messages related to missing plugin versions for org.easyb:maven-easyb-plugin and org.apache.maven.plugins:maven-project-info-reports-plugin. It also mentions a warning about the stability of the build due to these issues. The logs end with reactor build order information for gameoflife, gameoflife-build, gameoflife-core, gameoflife-webservice, gameoflife-web, and gameoflife-cli.

```
Démarré par l'utilisateur anonymous
Building in workspace C:\CloudBees\tools\jenkins-data\workspace\gameoflife-freestyle
Updating svn://localhost/gameoflife/trunk at revision '2016-03-19T11:13:34.661 +0100'
At revision 46

No changes for svn://localhost/gameoflife/trunk since the previous build
[gameoflife-freestyle] $ cmd.exe /C '"C:\CloudBees\tools\apache-maven-3.0.2\bin\mvn.bat -Djetty.stop.port=9998 -
Dwebdriver.port=9091 -Dsurefire.useFile=false clean install -B -U && exit %ERRORLEVEL%'

[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for com.wakaleo.gameoflife:gameoflife-
core:jar:0.0.68
[WARNING] 'build.plugins.plugin.version' for org.easyb:maven-easyb-plugin is missing. @ line 17, column 15
[WARNING] 'reporting.plugins.plugin.version' for org.apache.maven.plugins:maven-project-info-reports-plugin is missing.
@ line 25, column 15
[WARNING]
[WARNING] Some problems were encountered while building the effective model for com.wakaleo.gameoflife:gameoflife-
web:war:0.0.68
[WARNING] 'build.plugins.plugin.version' for org.apache.maven.plugins:maven-war-plugin is missing. @ line 102, column
12
[WARNING]
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO] -----
[INFO] Reactor Build Order:
[INFO]
[INFO] gameoflife
[INFO] gameoflife-build
[INFO] gameoflife-core
[INFO] gameoflife-webservice
[INFO] gameoflife-web
[INFO] gameoflife-cli
[INFO]
```

ORGANIZING BUILD JOBS

ORGANIZING YOUR BUILD JOBS

- Don't do everything in one job:
 - Create many job and divide things up
- Progressive refinement:
 - Do cheap things first and get fast feedback:
 - Compilation, unit tests
 - Do expensive things later:
 - Only after cheap things run successfully
 - Integration tests, cross-platform tests



USING VIEWS

Views appear as tabs

Create a new view

S	W	Name ↓	Last Success
		gameoflife-default	4 min 40 sec - #2
		gameoflife-freestyle	21 min - #2
		gameoflife-integration-tests	2 min 41 sec - #3
		gameoflife-metrics	3 min 48 sec - #3
		gameoflife-web-tests	26 min - #1

Icon: [S](#) [M](#) [L](#)

LIST VIEW

- Contains a list of projects:

View name

Dashboard
Customizable view that contains various portlets containing information about your job(s)

Groovy Script View
This view renders HTML produced by a Groovy script.

List View
Shows items in a simple list format. You can choose which jobs are to be displayed in which view.



CREATING A LIST VIEW

The screenshot shows the Jenkins 'Create New View' configuration page. The 'Name' field is set to 'Test list'. The 'Job Filters' section includes a 'Status Filter' set to 'All selected jobs' and a 'Jobs' section listing several Jenkins job names: gameoflife-default, gameoflife-freestyle, gameoflife-integration-tests, gameoflife-metrics, and gameoflife-web-tests. A checkbox for 'Use a regular expression to include jobs into the view' is unchecked. Below the filters is an 'Add Job Filter' button. The 'Columns' section lists three columns: 'Status', 'Weather', and 'Name', each with a red 'Delete' button to its right. Three callout bubbles point to these features:

- A blue callout bubble points to the 'Jobs' section with the text "Choose the jobs".
- A blue callout bubble points to the 'Add Job Filter' button with the text "Or use regular expression".
- A blue callout bubble points to the 'Name' column with the text "Control what columns to use".



CUSTOMIZE LIST VIEW

- “View Job Filters” plugin
- Columns:
 - “Favorite” mark
 - Compact status column

All	Compact Columns Example				v1	+
S	W	Job ↓	Last Statuses			
●		funk	N/A			
●	●	jobby	4.8 days > 4.8 days			
●	●	junit	4.8 days > 9 days > 9.1 days			
●	●	more jobs	9.1 days			
●	●	successes	9.1 days			

Icon: S M L

Build #3 (Latest Build)

- Built @ 2:03 PM, 05/28/2010
- Ended 9.1 days ago
- Lasted 0.11 sec
- Stable

OTHER VIEW TYPES

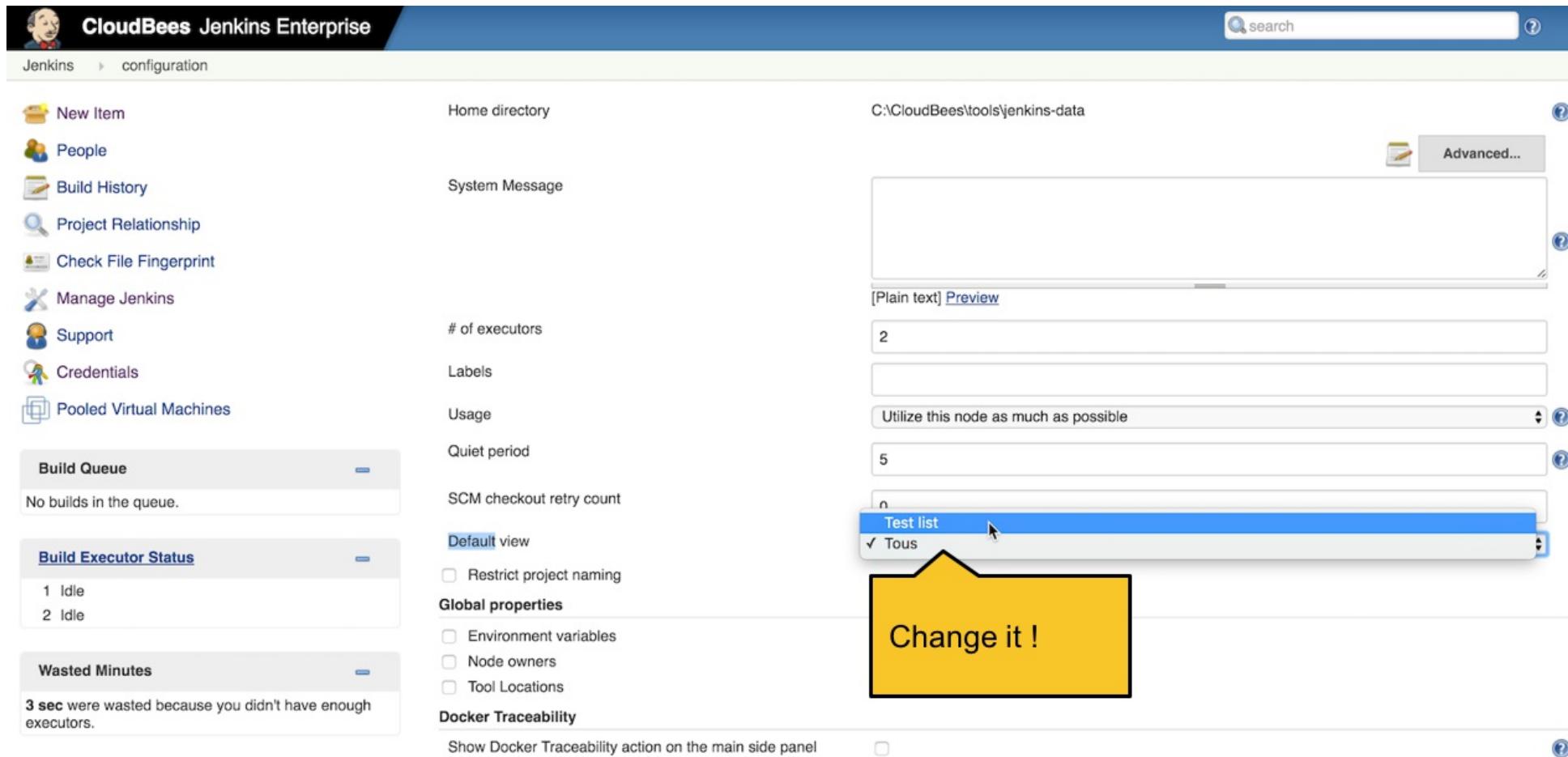
Radiator View Plugin:

- Highly-visible display of build status
 - Visible from a distance
 - May show all builds
 - Or only broken ones



DEFAULT VIEW

- **All** view is the default, but if more views exist, this can be changed in `$JENKINS_URL/configure`:



The screenshot shows the Jenkins configuration interface for CloudBees Jenkins Enterprise. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', etc. The main area has sections for 'Home directory' (set to 'C:\CloudBees\tools\jenkins-data'), 'System Message', and various Jenkins management settings. A prominent dropdown menu titled 'Default view' is open, showing options like 'All', 'Tous', and 'None'. A yellow callout bubble with the text 'Change it!' points to this dropdown. The top navigation bar shows 'CloudBees Jenkins Enterprise' and a search bar.

LAB EXERCISES:

- [Lab 3: Displaying Test Results](#)
- [Lab 4: Creating an Integration Tests Build](#)



PLUGIN MANAGEMENT

JENKINS PLUGINS

- Jenkins can be extended by writing plugins
- They are mainly used to integrate with 3rd-party tools
- Many already available
- Check out the Jenkins website

PLUGIN EXAMPLES

- Source code management tools
- Build Tools
- Reporting tools
 - Code coverage, static code analysis
- Online source code browsers
- Issue tracker
- Notification tools
- Views and UI customizations
- Distributed builds



INSTALLING PLUGINS

The screenshot shows the Jenkins management interface. On the left, there's a sidebar with links like 'New Item', 'People', 'Build History', 'Project Relationship', 'Check File Fingerprint', 'Manage Jenkins' (which is selected), 'Support', 'Credentials', and 'Pooled Virtual Machines'. Below this are three summary boxes: 'Build Queue' (No builds in the queue), 'Build Executor Status' (1 Idle, 2 Idle), and 'Wasted Minutes' (3 sec were wasted because you didn't have enough executors). The main content area is titled 'Manage Jenkins' and lists several configuration options with icons. One option, 'Manage Plugins' (represented by a green puzzle piece icon), is circled in red.

- [Configure System](#)
Configure global settings and paths.
- [Configure Global Security](#)
Secure Jenkins; define who is allowed to access/use the system.
- [Configure Analytics](#)
Configure the storage and reporting of Analytics
- [Reload Configuration from Disk](#)
Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.
- [Manage Plugins](#)
Add, remove, disable or enable plugins that can extend the functionality of Jenkins. (**updates available**)
- [System Information](#)
Displays various environmental information to assist trouble-shooting.
- [System Log](#)
System log captures output from `java.util.logging` output related to Jenkins.
- [Load Statistics](#)
Check your resource utilization and see if you need more computers for your builds.
- [Jenkins CLI](#)
Access/manage Jenkins from your shell, or from your script.
- [Script Console](#)
Executes arbitrary script for administration/trouble-shooting/diagnostics.
- [Manage Nodes](#)
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- [High Availability Status](#)
Monitor the status of the Jenkins Enterprise HA cluster
- [Plugin Usage](#)
Summarizes usage of installed plugins insofar as that can be determined. *Beta status*; report bugs and suggestions to [CloudBees Support](#).



WHAT PLUGINS ARE AVAILABLE?

The screenshot shows the Jenkins Plugin Manager interface. At the top, there's a navigation bar with the CloudBees Jenkins Enterprise logo, a search bar, and tabs for 'Updates', 'Available' (which is selected), 'Installed', and 'Advanced'. Below the tabs is a table with columns for 'Name' and 'Version'. A yellow callout bubble points to the 'Available' tab. The table lists various plugins under sections like '.NET Development' and 'Android Development'. At the bottom of the page are buttons for 'Install without restart', 'Download now and install after restart', and 'Check now'.

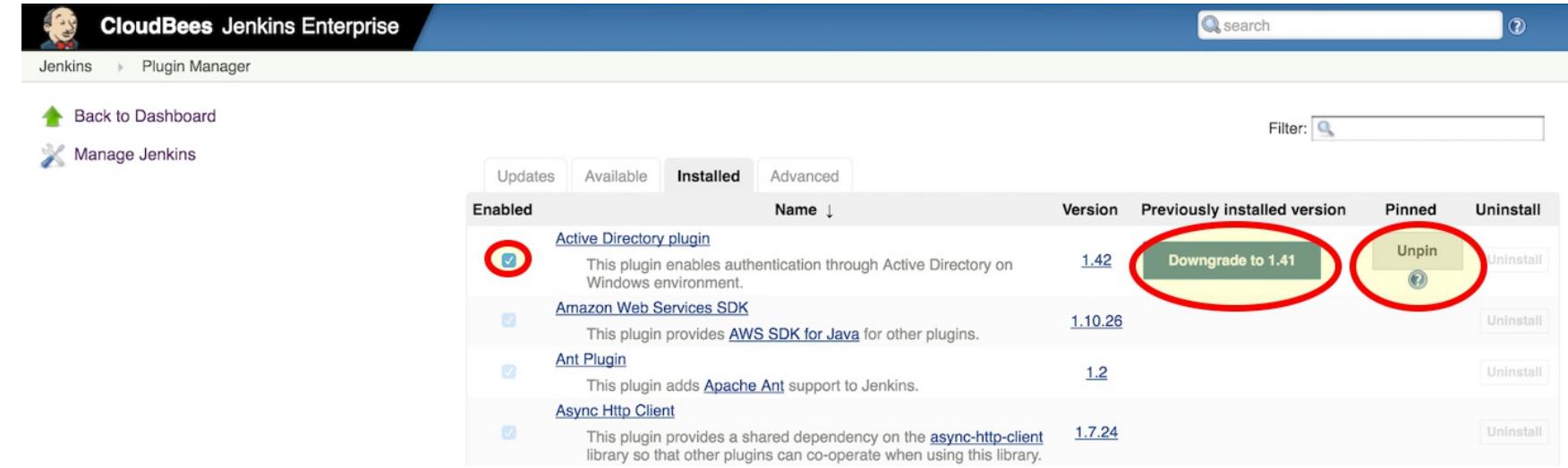
Name	Version
CCM Plug-in	3.1
FxCop Runner plugin	1.1
MSBuild Plugin	1.25
MSTest plugin	0.19
MSTestRunner plugin	1.2.0
PowerShell plugin	1.3
Violation Comments to Bitbucket Server Plugin	1.12
Visual Studio Code Metrics Plugin	1.7
VSTest Runner plugin	1.0.4
change-assembly-version-plugin	1.5.1
Android Emulator Plugin	2.13.1
Appetize.io Plugin	

Install without restart Download now and install after restart Check now

Update information obtained: 56 min ago

INSTALLED PLUGINS

- You can downgrade, pin/unpin or disable plugins:

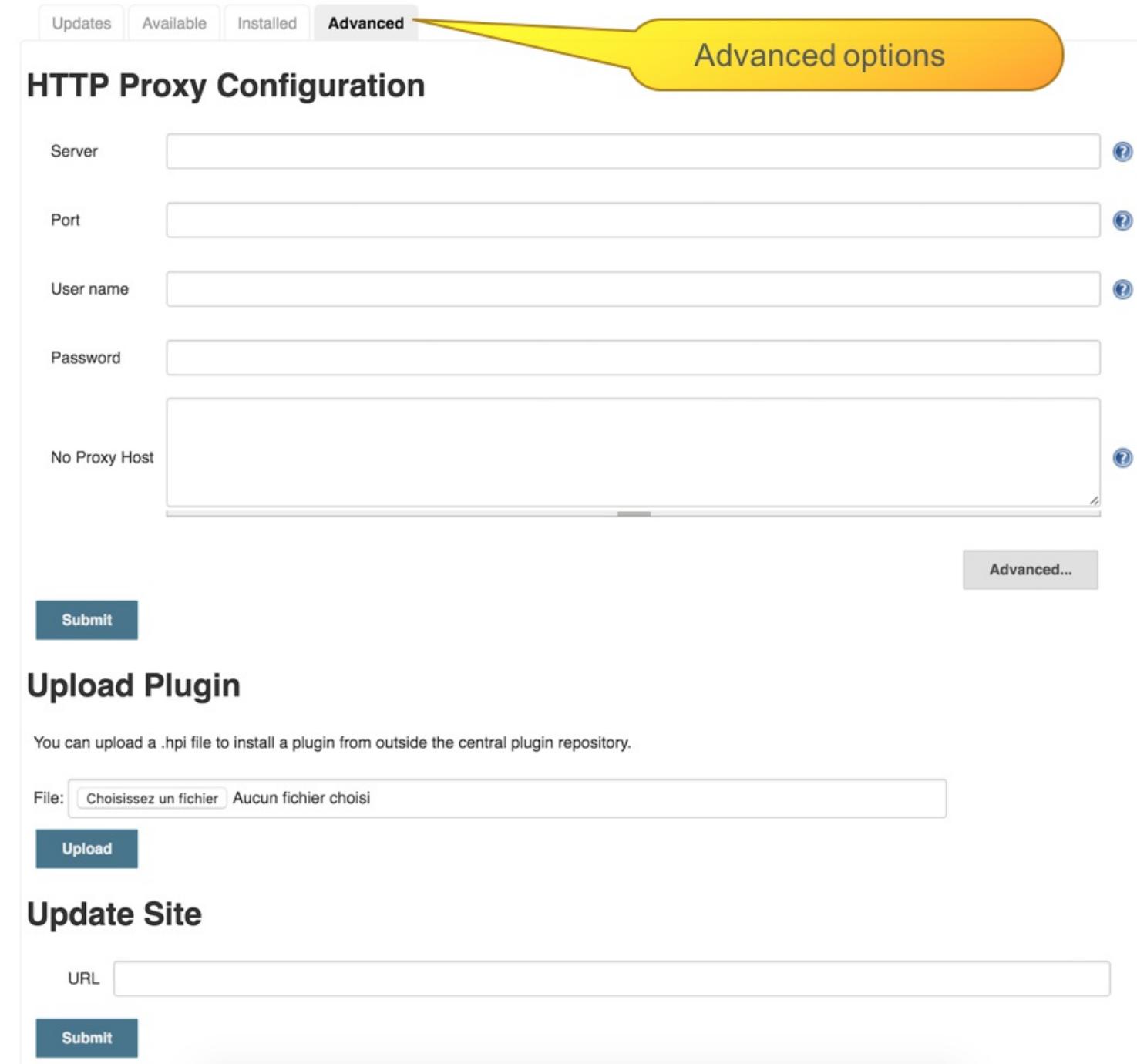


Enabled	Name	Version	Previously installed version	Pinned	Uninstall
<input checked="" type="checkbox"/>	Active Directory plugin This plugin enables authentication through Active Directory on Windows environment.	1.42	1.41	Downgrade to 1.41	Unpin
<input checked="" type="checkbox"/>	Amazon Web Services SDK This plugin provides AWS SDK for Java for other plugins.	1.10.26		Uninstall	
<input checked="" type="checkbox"/>	Ant Plugin This plugin adds Apache Ant support to Jenkins.	1.2		Uninstall	
<input checked="" type="checkbox"/>	Async Http Client This plugin provides a shared dependency on the async-http-client library so that other plugins can co-operate when using this library.	1.7.24		Uninstall	

PINNING PLUGINS:

- When you update bundled plugin, they get pinned
- Pinned plugins don't get overwritten automatically
- On disk, you'll see a file named foo.hpi.pinned

MANUALLY INSTALLING PLUGINS



The screenshot shows the Jenkins Manage Plugins interface. At the top, there are tabs: Updates, Available, Installed, and Advanced (which is highlighted). A yellow callout bubble points to the Advanced tab with the text "Advanced options".

HTTP Proxy Configuration

This section contains fields for proxy settings:

- Server
- Port
- User name
- Password
- No Proxy Host

An "Advanced..." button is located at the bottom right of this section.

Upload Plugin

You can upload a .hpi file to install a plugin from outside the central plugin repository.

File: Choisissez un fichier Aucun fichier choisi

Update Site

URL:

Submit

TOC 

EXAMPLE: VIOLATIONS PLUGIN

- Reports on code quality metrics:
- Checkstyle, CPD, FindBugs, PMD, and also FxCop and pylint
- Your build script needs to generate the raw XML data

WORKING WITH VERSION CONTROL

WORKING WITH VERSION CONTROL

- Integrating with your SCM
 - Build triggers
 - Integrating with a SCM browser
 - Integrate with SCM users



SCM BROWSER INTEGRATION

- Why?
 - See details of what changes went to builds
 - You can share the pointer with colleagues
- Jenkins Integrates with browsers for many modern SCMs
 - Subversion: Trac, ViewSVN, WebSVN, Sventon, Fisheye, Collabnet,...
 - Git: gitweb, github,...
 - Mercurial: bitbucket, googlecode, hgweb,...



EXAMPLE USING SVENTON

Source Code Management

None
 Git
 Mercurial
 Subversion

Modules

Repository URL: `svn://localhost/gameoflife/trunk`

Credentials: `harry*****`

Local module directory: `.`

Repository depth: `infinity`

Additional

Check-out Strategy: `Use 'svn' as much as possible` (Use 'svn' whenever possible, making the build faster. But this causes the artifacts from the previous build to remain in the workspace.)

Repository browser: `Sventon 2.x`

URL: `http://localhost:8888/svn/`

Repository Instance: `gameoflife`

Repository browser: `FishEye`

URL: `http://www.cloudbees.com/fisheye/browse/team1/`

Root module: `ameoflife`

Advanced...

Possible browsers depend on your SCM

Depending on browser, you get different config options



EXAMPLE USING SVENTON

CloudBees Jenkins Enterprise

Jenkins > gameoflife-default > #3

[Back to Project](#)
[Status](#)
[Changes](#)
[Console Output](#)
[Edit Build Information](#)
[Delete Build](#)
[Tag this build](#)
[Redeploy Artifacts](#)
[Test Result](#)
[See Fingerprints](#)
[Previous Build](#)

Build #3 (Mar 19, 2016 11:53:26 AM)

Revision: 48 Changes
1. Reverting Dead Cell code ([detail/Sventon 2.x](#))

Started by a [redacted]
This run spent:
• 30 m
• 31 sec building on an executor,
• 31 sec total from scheduled to completion.

[Test Result \(no failures\)](#)



EXAMPLE USING SVENTON

sventon subversion web client - <http://www.sventon.org>

[show recent changes]

Rev: HEAD (48) - svn://localhost/gameoflife / gameoflife / trunk / gameoflife-core / src / main / java / com / wakaleo / gameoflife / domain / Cell.java

Diff View - Cell.java

Show log Show file Toggle line wrap Diff to previous Inline UTF-8

```
↓ /gameoflife/trunk/gameoflife-core/src/main/java/com/wakaleo/gameoflife/domain/Cell.java @ revision 47
↓ /gameoflife/trunk/gameoflife-core/src/main/java/com/wakaleo/gameoflife/domain/Cell.java @ revision 48
1 1 package com.wakaleo.gameoflife.domain;
2 2
3 3 public enum Cell {
4 4     - LIVE_CELL("**"), DEAD_CELL("-");
5 5     + LIVE_CELL("**"), DEAD_CELL(".");
6 6
7 7     private String symbol;
8 8
9 9     private Cell(String symbol) {
10 10         this.symbol = symbol;
11 11     }
12 12     @Override
13 13     public String toString() {
14 14         return symbol;
15 15     }
16 16
17 17     static Cell fromSymbol(String symbol) {
18 18         Cell cellRepresentedBySymbol = null;
19 19         for (Cell cell : Cell.values()) {
20 20             if (cell.symbol.equals(symbol)) {
21 21                 cellRepresentedBySymbol = cell;
22 22                 break;
23 23             }
24 24         }
25 25         return cellRepresentedBySymbol;
26 26     }
27 27
28 28     public String getSymbol() {
29 29         return symbol;
30 30     }
31 31 }
```

sventon 2.1.5



LAB EXERCISE:

Lab 5: Integrating with a SCM Browser

CODE QUALITY METRICS

CODE QUALITY METRICS

- Jenkins is very good at reporting on code quality metrics:
 - Static analysis (Checkstyle, FindBugs,...)
 - Code Coverage (Cobertura, Clover,...)
 - Code quality data mining(SonarQube)



STATIC ANALYSIS

- Analyze code without running it
- Find potential bugs
- Examples:
 - Checkstyle
 - PMD
 - Findbugs
- Use the Jenkins 'Violations' plugin

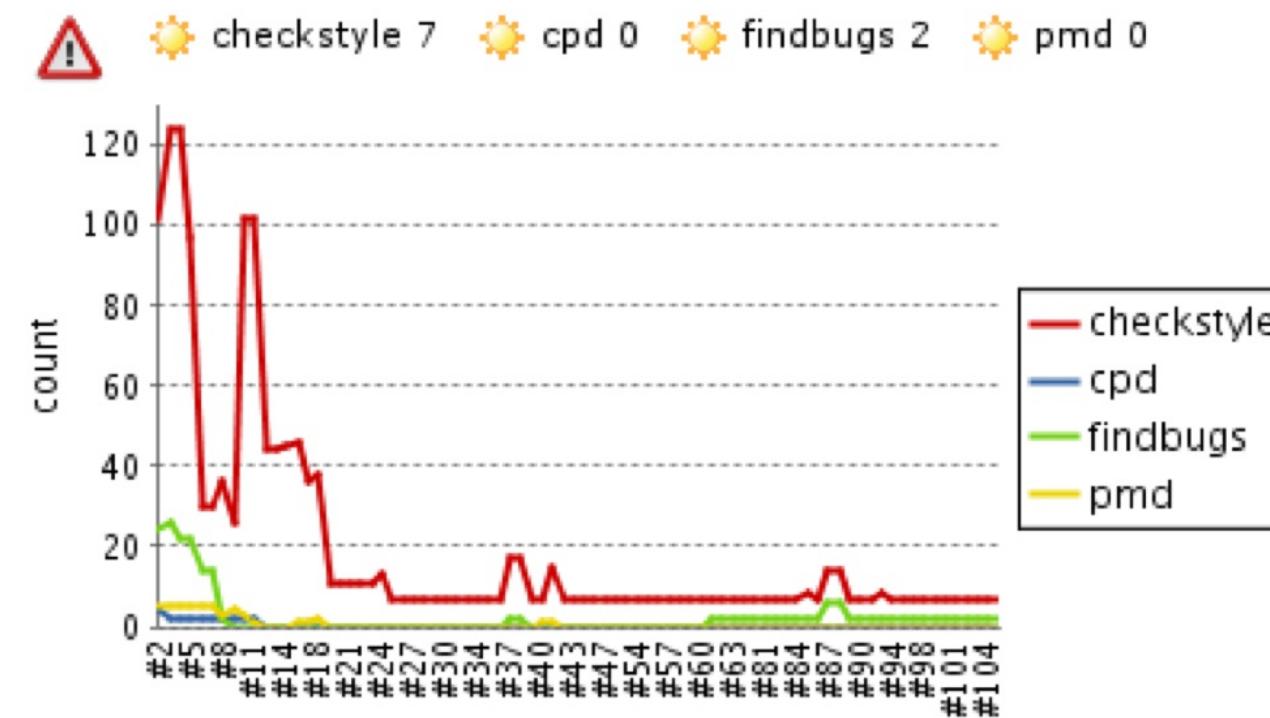
INTEGRATING CHECKSTYLE REPORTS

- Works on source code:
 - Enforce coding standards, javadoc, etc.
- Customizable rules
- Supports Ant, Maven, etc.

```
<project>
  <reporting>
    ...
    <plugin>
      <artifactId>maven-checkstyle-plugin</artifactId>
      <configuration>
        <configLocation>${basedir}/src/main/config/my-checkstyle-rules.xml</configLocation>
      </configuration>
    </plugin>
    ...
  </plugins>
</reporting>
</project>
```

INTEGRATING CHECKSTYLE REPORTS

- Checkstyle statistics are included in the Violations graph



CONFIGURING CHECKSTYLE PLUGIN

- Configured as a post-build plugin

Activate the plugin

Report Violations

Per file display limit

This is the number of violations to display per file (per type)

type	sun icon	cloud icon	XML filename pattern
checkstyle	10	999	planeshowcase/target/checkstyle-result.xml
cpd	10	999	planeshowcase/target/cpd.xml
findbugs	10	999	planeshowcase/target/findbugs-result.xml
fxcop	10	999	
pmd	10	999	planeshowcase/target/pmd-result.xml
pylint	10	999	

Location of the generated report file

Source Path Pattern (Optional)

This is a file name pattern that can be used to resolve classes to sourcefiles (for example `**/src/main/java`).

Faux Project Path (Optional)



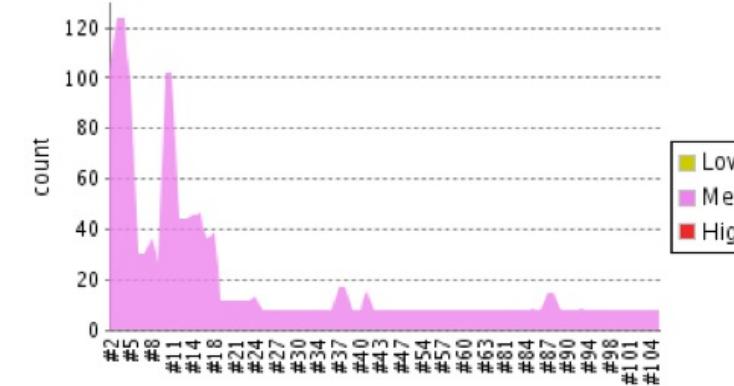
CHECKSTYLE DETAILS

Checkstyle violation count

⚠️ Violations Report for build 105

Type	Violations	Files in violation
checkstyle	7	4
cpd	0	0
findbugs	2	1
pmd	0	0

checkstyle



Detailed statistics

filename
data/home/hudson/jobs/oia-debt-calculator-portlet-site/workspace/debtcalculator-portlet/src/main/java/nz/govt/ird/oia/debtcalculator/portle
data/home/hudson/jobs/oia-debt-calculator-portlet-site/workspace/debtcalculator-portlet/src/main/java/nz/govt/ird/oia/debtcalculator/portle
data/home/hudson/jobs/oia-debt-calculator-portlet-site/workspace/debtcalculator-portlet/src/main/java/nz/govt/ird/oia/debtcalculator/portle
data/home/hudson/jobs/oia-debt-calculator-portlet-site/workspace/debtcalculator-portlet/src/main/java/nz/govt/ird/oia/debtcalculator/portle





CHECKSTYLE SOURCE CODE OVERLAY

- View details for a particular class:

!

data/home/hudson/jobs/oia-debt-calculator-portlet-site/workspace/debtcalculator-portlet

checkstyle 4 violations

21	⚠	Utility classes should not have a public or default constructor.
34	⚠	Avoid inline conditionals.
60	⚠	'32' is a magic number.
112	⚠	Missing a Javadoc comment.

File: HashCodeUtil.java Lines 12 to 44

```
12 *     int result = HashCodeUtil.SEED;
13 *     //collect the contributions of various fields
14 *     result = HashCodeUtil.hash(result, fPrimitive);
15 *     result = HashCodeUtil.hash(result, fObject);
16 *     result = HashCodeUtil.hash(result, fArray);
17 *     return result;
18 * }
19 * </pre>
20 */
```

⚠ 21 public final class HashCodeUtil {

Type	Class	Description
checkstyle	HideUtilityClassConstructorCheck	Utility classes should not have a public or default constructor. is added contributions

```
25     * from fields. Using a non-zero value decreases collisions of <code>hashCode</code>
26     * values.
27     */
28     public static final int SEED = 23;
```

List of violations

Details in the source code

VIOLATIONS PLUGIN SUPPORTS MORE

- PMD
 - Works on source code
 - Based on xpath matches from source tree.
 - Find coding problems
- CPD
 - Works on source code
 - Detects copy/paste in code
- FindBugs
 - Works on bytecode
 - Good at finding errors involving types

LAB EXERCISE:

Lab 6: Code Quality Metrics

TOC

CODE COVERAGE METRICS:

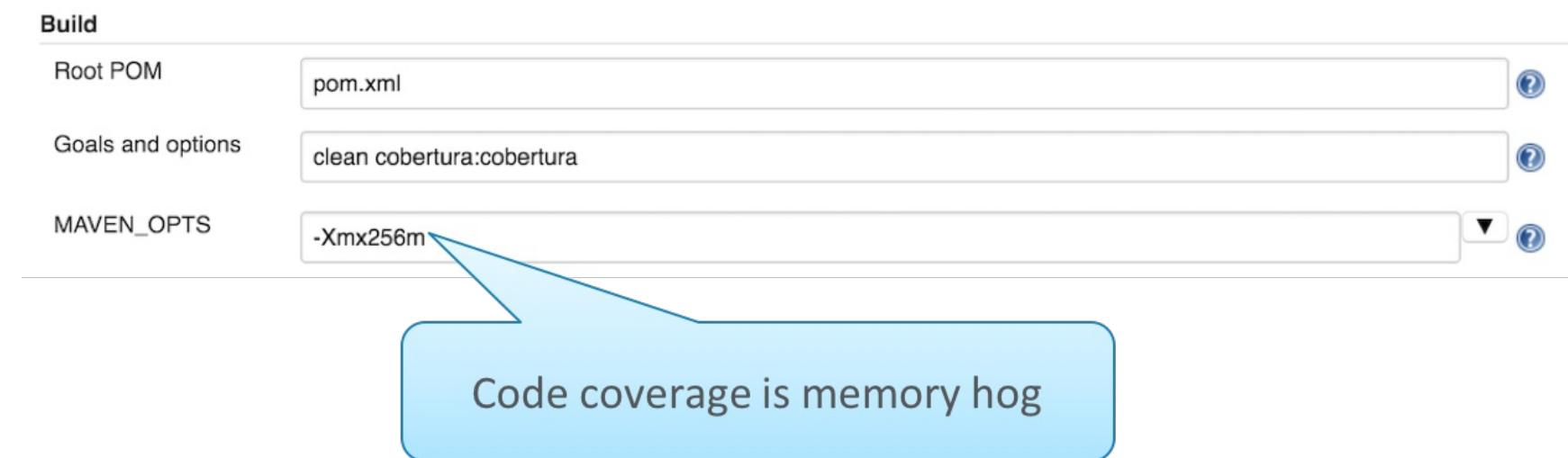
Dynamic Analysis

DYNAMIC ANALYSIS

- Most commonly used code coverage
 - But there are others, like deadlock analysis
- Code coverage measures how much code is executed by your tests
- Useful in guiding unit test effort
- Jenkins integrates with several Code Coverage tools:
 - JaCoCo, Clover, Cobertura, Emma, Crap4j

INTEGRATING WITH COBERTURA

- Configuring projects to work with Cobertura:



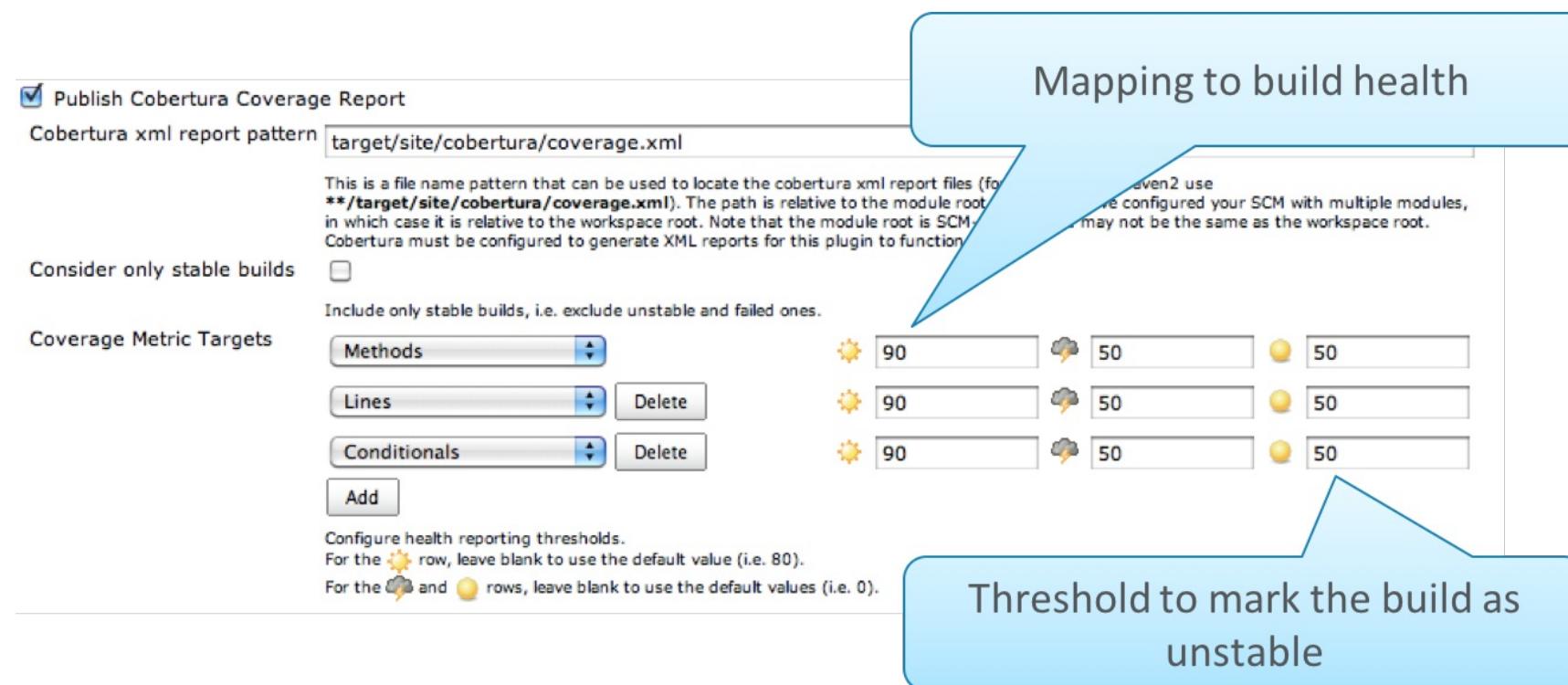
The screenshot shows the Jenkins build configuration for a project. The 'Build' section includes fields for 'Root POM' (set to 'pom.xml'), 'Goals and options' (set to 'clean cobertura:cobertura'), and 'MAVEN_OPTS' (set to '-Xmx256m'). A blue callout box with the text 'Code coverage is memory hog' points to the 'MAVEN_OPTS' field.

Build	
Root POM	pom.xml
Goals and options	clean cobertura:cobertura
MAVEN_OPTS	-Xmx256m

Code coverage is memory hog

INTEGRATING WITH COBERTURA

- Configure Cobertura in the Post-build Actions:



Publish Cobertura Coverage Report

Cobertura xml report pattern `target/site/cobertura/coverage.xml`

This is a file name pattern that can be used to locate the cobertura xml report files (for `**/target/site/cobertura/coverage.xml`). The path is relative to the module root, in which case it is relative to the workspace root. Note that the module root is SCM, Cobertura must be configured to generate XML reports for this plugin to function.

Consider only stable builds

Include only stable builds, i.e. exclude unstable and failed ones.

Coverage Metric Targets

Metric	Sun (90)	Cloud (50)	Circle (50)
Methods	90	50	50
Lines	90	50	50
Conditionals	90	50	50

Add

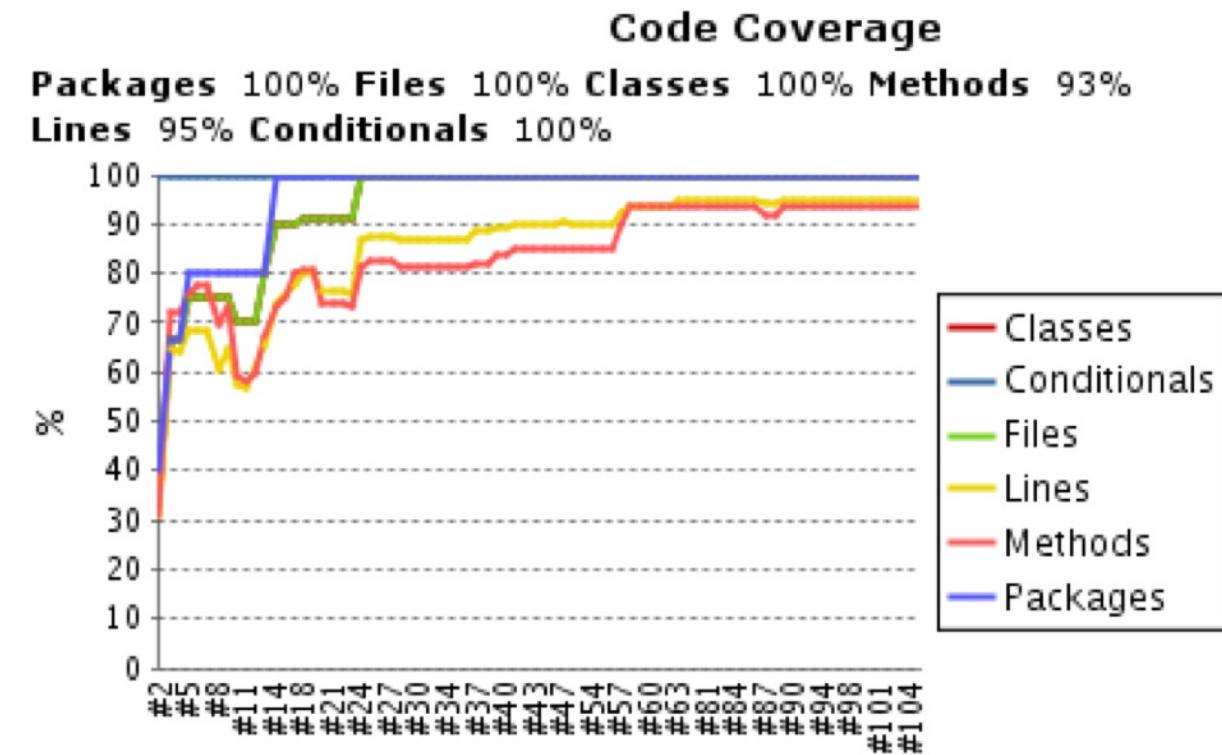
Configure health reporting thresholds.
For the Sun row, leave blank to use the default value (i.e. 80).
For the Cloud and Circle rows, leave blank to use the default values (i.e. 0).

Mapping to build health

Threshold to mark the build as unstable

CODE COVERAGE REPORTS

- Code coverage statistics are displayed on the build results page



CODE COVERAGE REPORT DRILL-DOWN

CloudBees Jenkins Enterprise

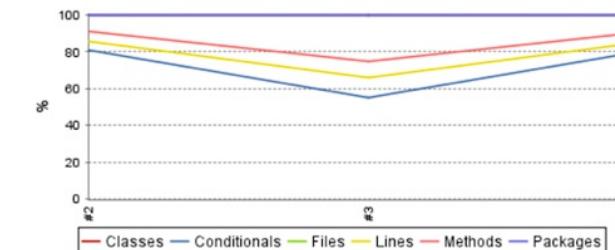
Jenkins gameoflife-metrics #4

Back to Project Status Changes Console Output Edit Build Information Delete Build Deploy Now Tag this build Violations Redeploy Artifacts Test Result Coverage Report See Fingerprints Previous Build

Code Coverage

Cobertura Coverage Report

Trend



Metric	B2 (%)	B3 (%)	B4 (%)
Classes	~85	~75	~90
Conditionals	~80	~60	~85
Files	~80	~75	~85
Lines	~80	~70	~85
Methods	~85	~75	~90
Packages	~80	~70	~85

Project Coverage summary

Name	Packages	Files	Classes	Methods	Lines	Conditionals
Cobertura Coverage Report	100% 2/2	100% 7/7	100% 7/7	91% 51/56	86% 154/180	81% 68/84

Coverage Breakdown by Package

Name	Files	Classes	Methods	Lines	Conditionals
com.wakaleo.gameoflife.domain	100% 5/5	100% 5/5	91% 40/44	86% 118/137	80% 56/70
com.wakaleo.gameoflife.web.controllers	100% 2/2	100% 2/2	92% 11/12	84% 36/43	86% 12/14

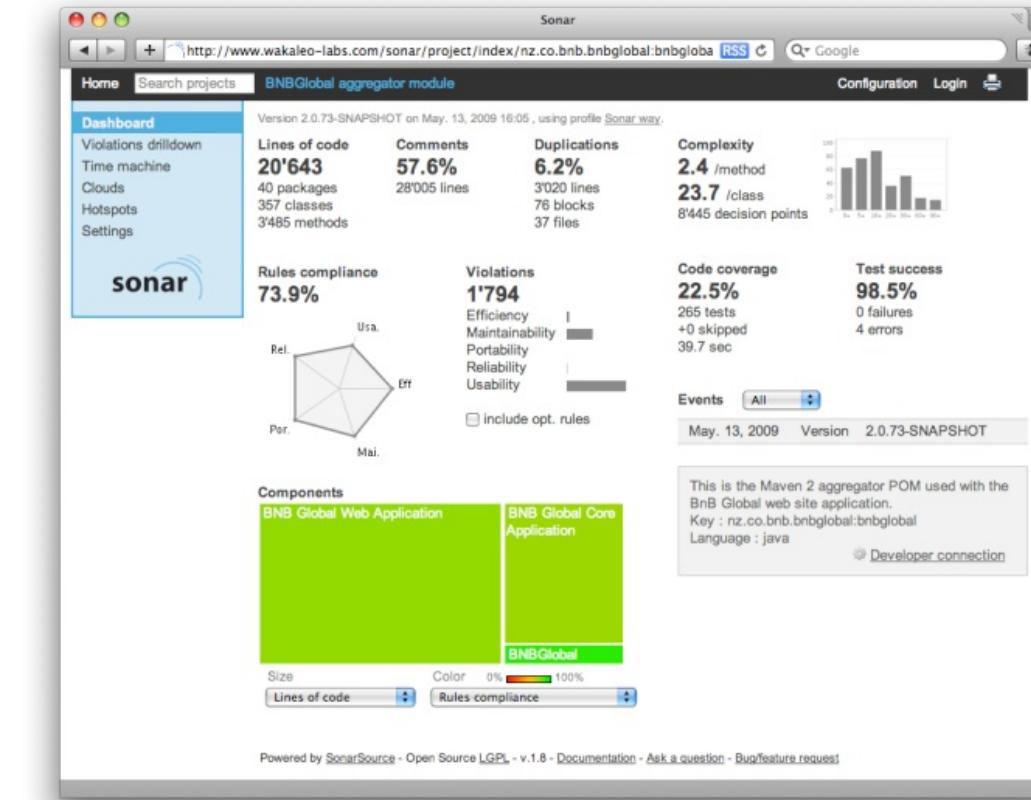


CODE COVERAGE LINE-BY-LINE REPORT

```
43     */
44     public Manifest getManifest() {
45     3         if (manifest == null) {
46     0             loadMetaInf();
47         }
48     3         return manifest;
49     }
50
51     public void setManifest(Manifest manifest) {
52     0         this.manifest = manifest;
53     }
54
55     public ServletContext getContext() {
56     0         return context;
57     }
58
59     public void setContext(ServletContext context) {
60     3         this.context = context;
61     3     }
62
63     /**
64      * Load the META-INF file.
65      */
66     public void loadMetaInf() {
67     3         String appServerHome = context.getRealPath("/");
68     3         File manifestFile = new File(appServerHome, "META-INF/MANIFEST.MF");
--
```

COMBINED METRICS: SONARQUBE

- SonarQube gives overview of project metrics
- Projects publish metrics data to the SonarQube database
- Jenkins is obvious place to run them



PUBLISHING METRICS TO SONARQUBE

Step 1

Configure the Jenkins SonarQube plugin:



Step 2

Configure SonarQube:



A screenshot of the Jenkins Configure System page under the 'Sonar' section. It shows a configuration for a 'Sonar installation'. The 'Name' field is set to 'Wakaleo-labs'. The 'Disable' checkbox is unchecked. The 'Version' field is set to '1.8'. Below the version field is a note: 'Must match the Sonar server version (displayed into the footer)'. There are 'Advanced...' and 'Delete' buttons at the bottom right.



CONFIGURING SONARQUBE PLUGIN

Sonar

Sonar installation	Name	<input type="text" value="Wakaleo Sonar Server"/>
	Disable	<input type="checkbox"/> Check to quickly disable Sonar on all jobs.
	Version	<input type="text" value="1.8"/> Must match the Sonar server version (displayed into the footer)
	Server URL	<input type="text" value="http://localhost/sonar"/> Default is http://localhost:9000
	Database URL	<input type="text" value="jdbc:mysql://localhost:3306/sonar?useUnicode=true&characterEncoding=utf8"/> 
	Database login	<input type="text"/> Do not set if you use the default embedded database on localhost.
	Database password	<input type="text"/> Default is sonar.
	Database driver	<input type="text" value="com.mysql.jdbc.Driver"/> Default is sonar.
	Additional properties	<input type="text"/> Additional properties to be passed to the mvn executable (example : -Dsome.property=some.value)

URL of SonarQube server

How to connect to database

LAB EXERCISE:

Lab 7: Code Coverage metrics

PARAMETERIZED BUILDS

WHAT ARE PARAMETERIZED BUILDS?

- For the moment, jobs were no-arguments methods call

```
void doCIOfProjectGameOfLife() {...}
```

- but it is possible to add parameter to this method

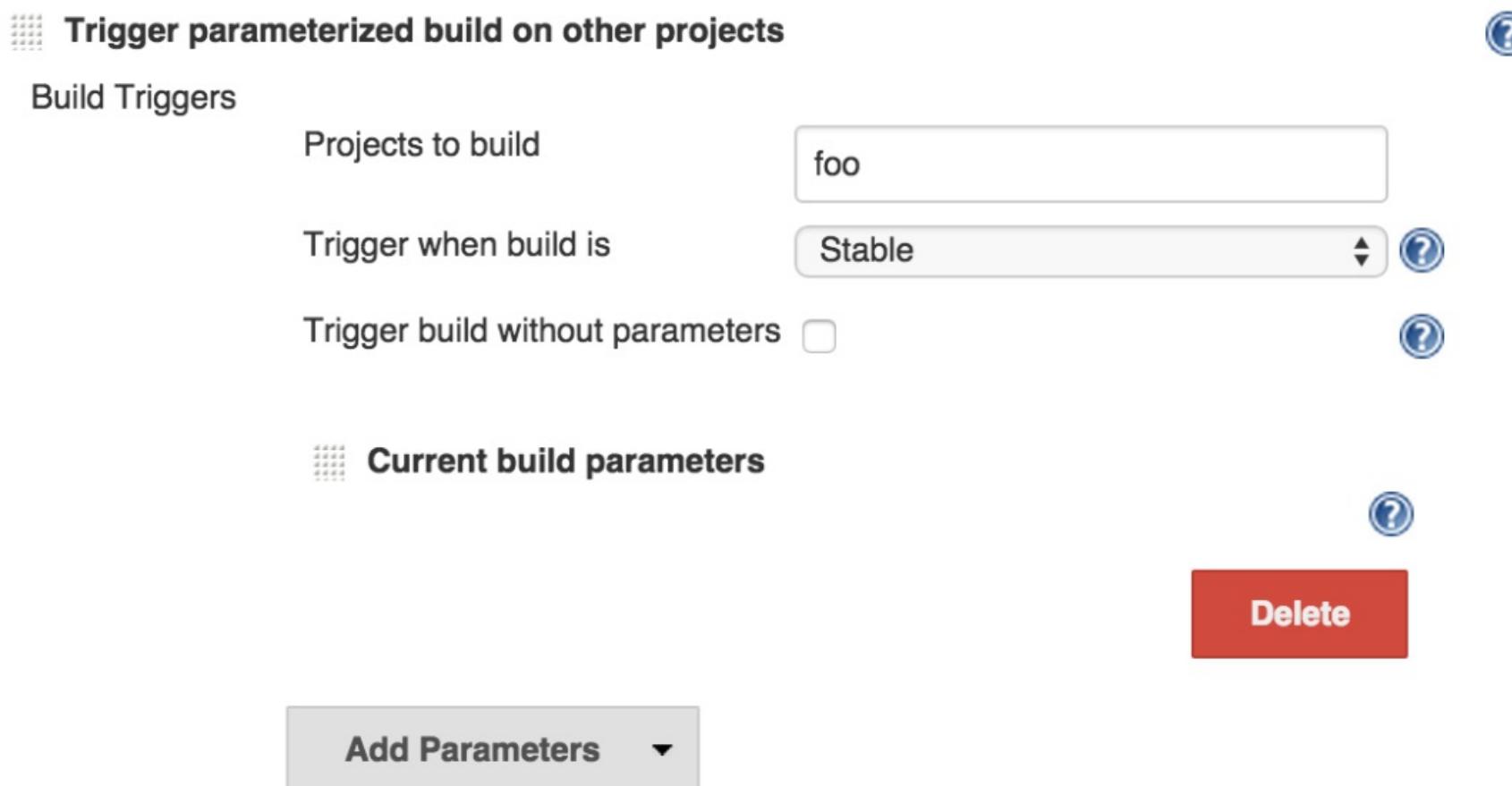
```
void doCIOfProjectGameOfLife(testPlatform) {...}
```

BENEFITS: GENERIC JOBS

- You don't need to create a job for every platform you want to test / use
 - **DRY** : Do not Repeat Yourself
 - Maintenance is easier

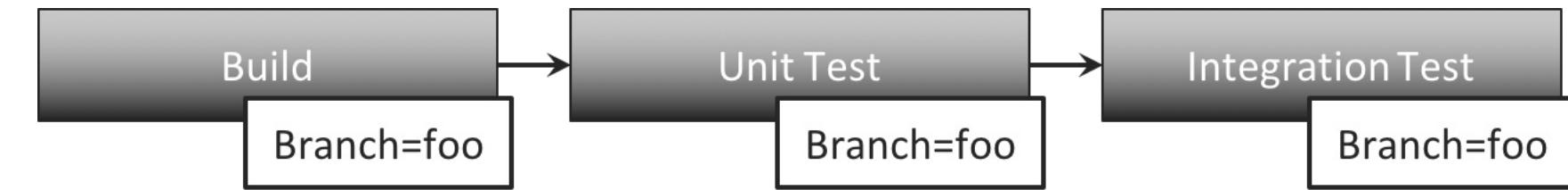
BENEFITS: CREATE A « PIPELINE »

- Using Parameterized Trigger Plugin, you can start another job with the current parameters:



The screenshot shows the Jenkins Parameterized Trigger configuration page. It includes sections for "Trigger parameterized build on other projects" and "Current build parameters". The "Build Triggers" section contains fields for "Projects to build" (set to "foo"), "Trigger when build is" (set to "Stable"), and "Trigger build without parameters" (unchecked). A "Delete" button is visible in the "Current build parameters" section, along with an "Add Parameters" dropdown.

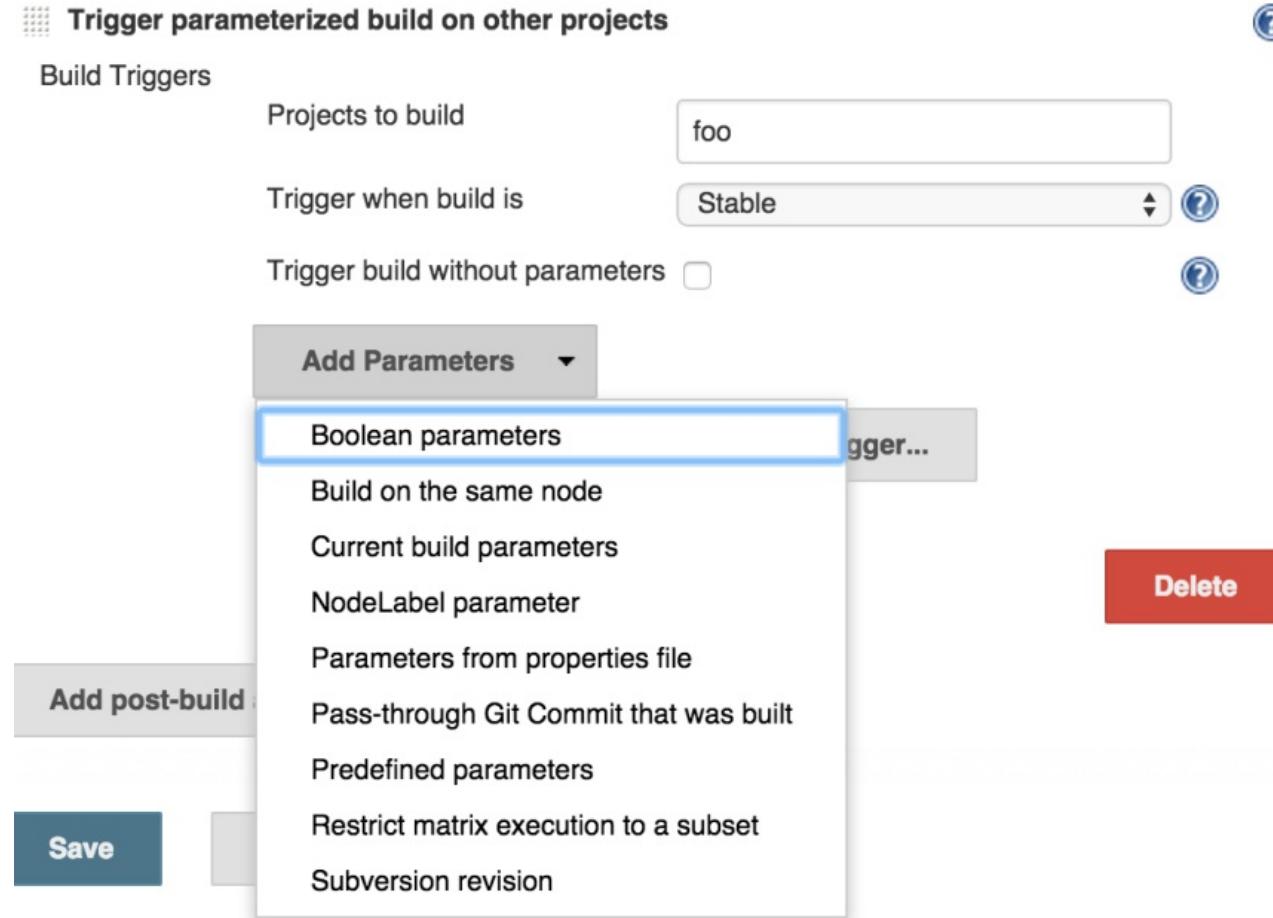
BENEFITS: CREATE A "PIPELINE"



- Pass current parameters:
- Pass SCM revision/branch/tags
 - This makes downstream builds check out the exact same revision
 - E.g., Ensure unit tests and builds are synced
- Similar feature available for all SCMs

BENEFITS: CREATE A « PIPELINE »

- It's also possible to provide more or different parameters to the next job:



The screenshot shows the Jenkins Pipeline configuration interface. In the 'Trigger parameterized build on other projects' section, the 'Projects to build' field contains 'foo' and the 'Trigger when build is' dropdown is set to 'Stable'. Below these, there is a checkbox for 'Trigger build without parameters'. An 'Add Parameters' dropdown menu is open, with 'Boolean parameters' selected. Other options in the menu include 'Build on the same node', 'Current build parameters', 'NodeLabel parameter', 'Parameters from properties file', 'Pass-through Git Commit that was built', 'Predefined parameters', 'Restrict matrix execution to a subset', and 'Subversion revision'. A red 'Delete' button is visible on the right side of the menu. At the bottom left, there are 'Save' and 'Add post-build' buttons.

WHERE DO WE START? #1

- Define parameters

This build is parameterized ?

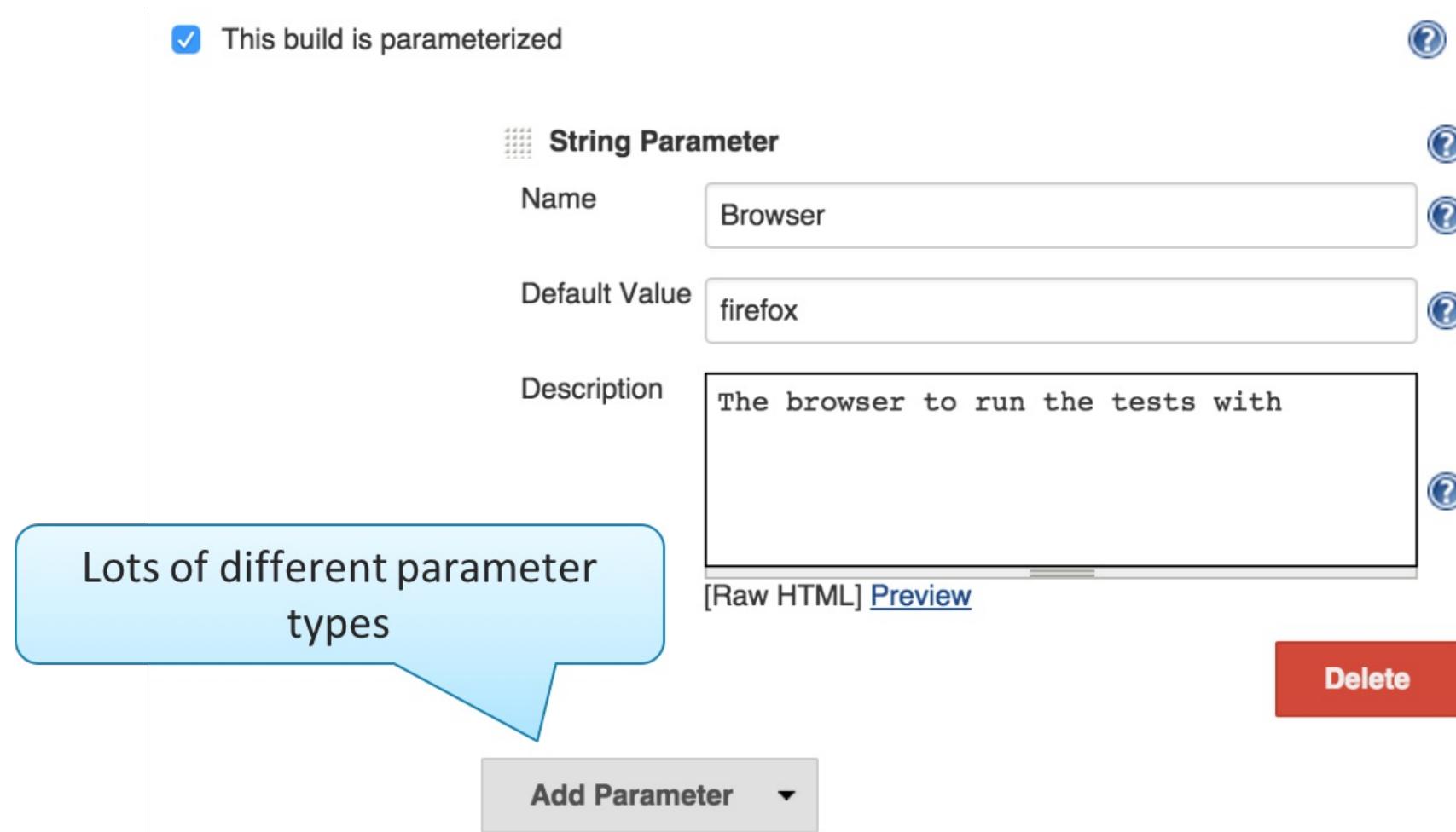
String Parameter ?

Name	Browser	?
Default Value	firefox	?
Description	The browser to run the tests with	?

[Raw HTML] [Preview](#) Delete

Lots of different parameter types

Add Parameter ▾



WHERE DO WE START? #2

- Refer to parameter values
 - As variable expansions: \${Browser}
 - As environment variables from your builds
 - Watch out for cases!
 - Some parameter types expose data in different ways
 - File parameter

WHERE DO WE START? #3

- Specify actual values when you launch a build:

Maven project gameoflife-integration-tests

This build requires parameters:

Browser

Browser to run the integration tests with.

Build

- From curl/wget, use query parameter:

```
/job/foo/buildWithParameters?Browser=firefox
```

- From CLI:

```
-p Browser=firefox"
```

LAB EXERCISE:

Lab 8: Parameterized Builds

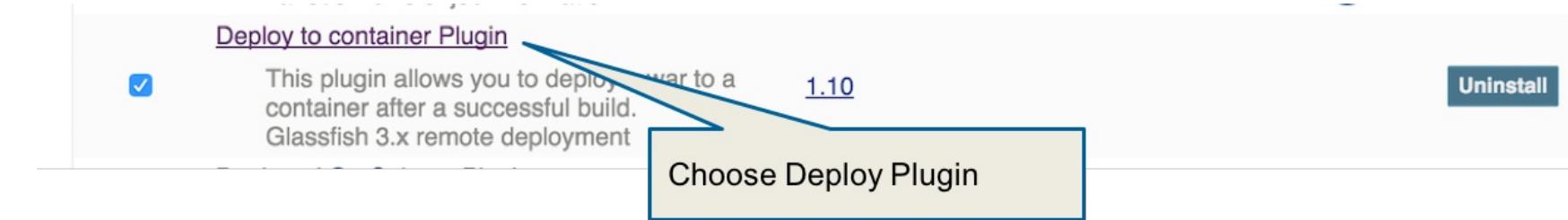
AUTOMATED DEPLOYMENT

WHAT IS AUTOMATED DEPLOYMENT

- A way to deploy your application to a container:
 - Continuous Deployment
 - Manual Deployment to test servers

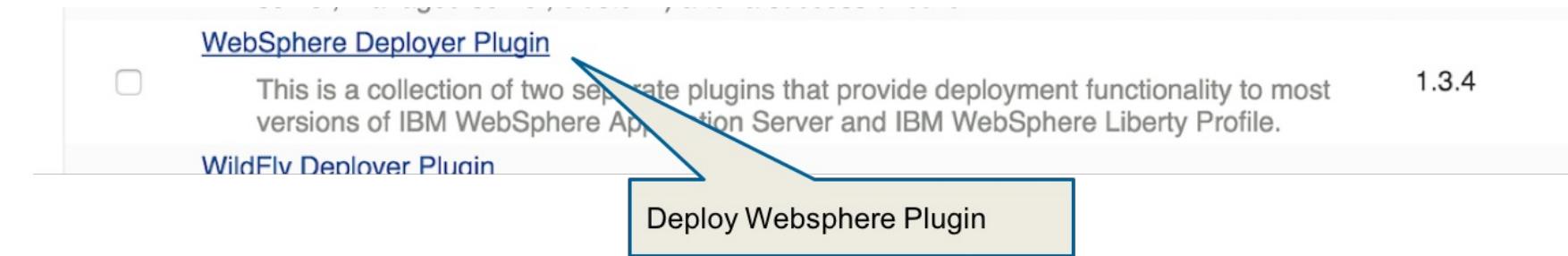
WHERE DO WE START?

- Install Deploy Plugin
 - supports several application servers (Tomcat, JBoss, GlassFish)



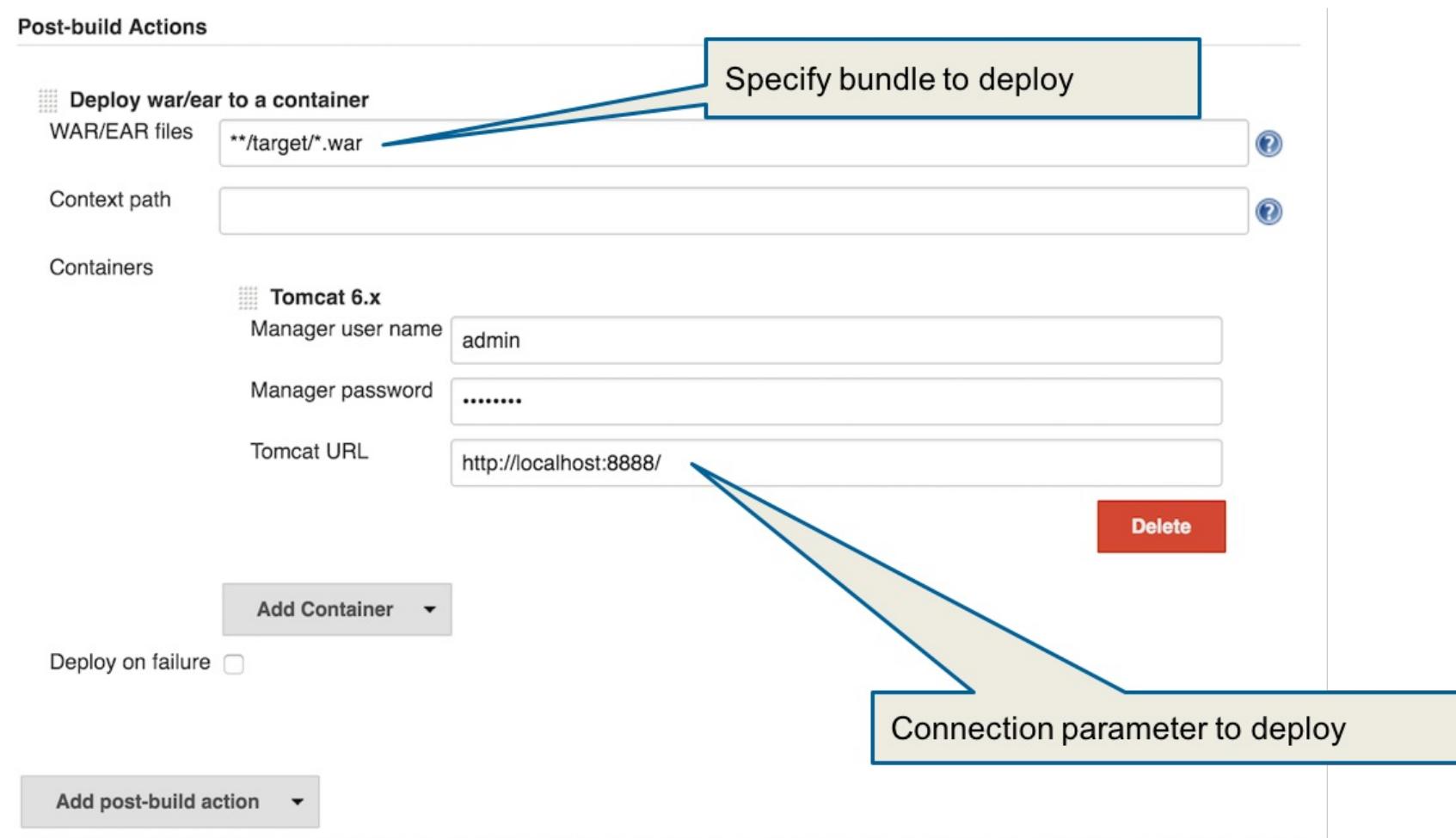
WHERE DO WE START?

- Install Deploy Plugin Additions:
 - Some additional containers available as separate plugins



WHERE DO WE START?

- Configure the WAR or EAR to be deployed



The screenshot shows the 'Post-build Actions' configuration screen. A blue callout box labeled 'Specify bundle to deploy' points to the 'WAR/EAR files' field containing the value '**/target/*.war'. Another blue callout box labeled 'Connection parameter to deploy' points to the 'Tomcat URL' field containing the value 'http://localhost:8888/'. A red 'Delete' button is visible next to the container configuration.

Post-build Actions

Deploy war/ear to a container

WAR/EAR files: **/target/*.war

Context path:

Containers

Tomcat 6.x

Manager user name: admin

Manager password:

Tomcat URL: http://localhost:8888/

Add Container ▾

Deploy on failure

Add post-build action ▾

LAB EXERCISE:

Lab 9: Automated Deployment to Tomcat



**CLOUDBEES
UNIVERSITY**

FOLDERS AND FOLDERS PLUS PLUGINS

WHAT IS FOLDERS/FOLDERS PLUS?

- Just like operating system folders, unlimited nesting
 - Folder 1→Folder 2→Folder 3→...→Folder n
 - Help you model **custom taxonomies**
- Creates namespace
 - "build" job in Folder A is different than "build" job in Folder B
- Folders have their own views.

BENEFITS: ISOLATION AND INHERITANCE

- When combined with the [Role-Based Access Control \(RBAC\)](#) plugin, Folders can have their own groups and permissions, effectively allowing teams to have their own “world” without needing their own Jenkins master
- Nested folders can inherit permissions from their parent folder, as well as templates in the parent folder.

BENEFIT: GRANULAR CONTROL #1

- Each folder **supports groups and roles** configured on a particular folder.
- Permissions from root be inherited (default), but can also be excluded by filtering out all filterable root-level groups.
- Folders can **limit what types of objects** can be created in the folder.

BENEFIT: GRANULAR CONTROL #2

- Folders allow for arbitrary key-value pairs to be set as **environment variables**.
 - Can be referenced from builds within folder
 - E.g.: `$BRANCH_NAME` or `$DEPLOY_URL`
- "Folders Plus" supports **linking agents to specific folders**, such that agents will only run that folders' builds.

BENEFITS: EASY BRANCHING

- When a branch of an existing project needs its own pipeline, the folder with the parent project's pipeline can simply be copied
 - Local references within the branch are kept intact
- Folders can also have environment variables for arbitrary key-value pairs, which jobs can reference
 - E.g.: \$BRANCH_NAME or \$DEPLOY_URL



WHERE DO WE START? #1

CloudBees Jenkins Enterprise

Jenkins > Tous >

New Item Item name: Departement A

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

Pipeline
Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

Auxiliary Template
Auxiliary templates are used to create nested structures embedded within other templates as attribute values. For example, if you are modeling a CD, you'd define an auxiliary template called "Track, and then your "CD" would have an attribute called "tracks" that contain a list of "Track"s.

Backup
Performs back up of Jenkins

Bitbucket Team
Scans a Bitbucket team for all repositories matching some defined markers.

Builder Template
Builder template lets you define a custom builder by defining a number of attributes and describing how it translates to the configuration of another existing builder. This allows you to create a locked down version of a builder. It also lets you change the definition of the translation without redoing all the use of the template.

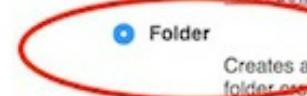
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

External Job
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard of your existing automation system. See [the documentation for more details](#).

Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

Folder Template
A folder template is useful to model a domain-specific concept that contains other "stuff" in it.

OK



TOC



WHERE DO WE START? #2

Name: Departement A

Display Name:

Description:

Views Tab Bar: [Plain text] Preview Default Views TabBar

Icon: Default Icon

Health metrics: Health metrics...

Credentials: Credentials...

Environment Variables:

Restrict the kind of children in this folder: This folder can contain the following items:

- Freestyle project
- Maven project
- Pipeline
- Auxiliary Template
- Backup
- Bitbucket Team
- Builder Template
- Multi-configuration project

Save Apply



SECURITY & ROLES BASE ACCESS CONTROL (RBAC)

ENABLING SECURITY

Manage Jenkins



[Configure System](#)
Configure global settings and paths.



[Configure Global Security](#)
Secure Jenkins; define who is allowed to access/use the system.



[Configure Analytics](#)
Configure the storage and reporting of Analytics

Jenkins can integrate its security to a lot of existing assets:

- Active Directory
- Delegate to servlet container
- Google Apps SSO (with OpenID)
- Jenkins' own user database
- LDAP
- OpenID SSO
- Operations Center Single Sign On
- Unix user/group database



PER-PROJECT ACCESS CONTROL

- If your Jenkins installation hosts sensitive projects that must be visible to a restricted set of people, define permissions at the individual project level so that different people have access to different sets of projects.
- Or if you just want to have a single set of access rights across all your projects, Jenkins can be configured that way, too.

WHAT IS RBAC?

- Role-based access control can be configured as an **authorization policy**.
- RBAC comes as a CJE plugin.
- RBAC allows:
 - Creation of re-usable global and object-only roles.
 - Role define very granular permissions on all Jenkins objects (projects, view, folders, etc.)

HOW DO I USE RBAC?

- Roles that are created must be mapped to **groups** defined in Jenkins:
 - These groups can be "shadow" groups for external groups in LDAP/AD
- Groups/roles can be defined on different objects in Jenkins:
 - Jobs
 - Maven modules
 - Agents
 - Views
 - Folders

BENEFITS: GRANULARITY

- CJE RBAC supports very granular permissions for roles.
 - E.g. "Browse" role only has overall read permissions while "QA" role supports read/write/execute permissions on jobs only in a given folder.
- Roles and groups can be combined with folders and other Jenkins objects to limit users' permissions within those objects.
 - E.g. A team-only project folder containing all jobs in a pipeline, along with a job that is hidden from anyone not in the "QA Team" group.

WHERE DO WE START?

- Ensure security is turned on and a valid security realm has been selected.
- Opt to use "role-based matrix authorization strategy" to enable CloudBees RBAC plugin to your authorization policy.
- Create groups and roles on your CJE master configuration:
 - Assign roles to these groups
 - Add users to groups

LAB EXERCISE:

Lab 10: Job Organization and Security with Folders

VALIDATED MERGE PLUGIN

aka the “Unbreakable Build”

WHAT IS THE VALIDATED MERGE PLUGIN?

- How can you **delegate more** to Jenkins?
- Does your CI server shift work from laptops to servers?
 - Not if you have to share the changes to test: You end up testing locally before committing
= > **FAIL**
- OK, let's test it before sharing changes
 - By taking advantages of **Git**

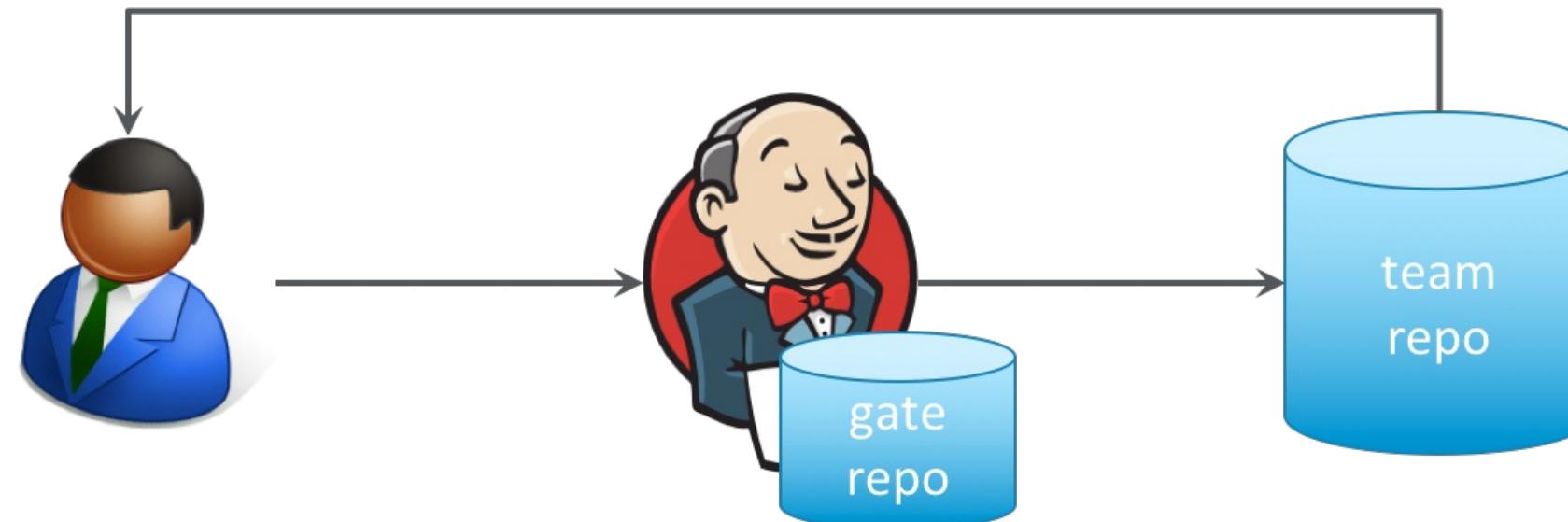
WHAT IS THE VALIDATED MERGE PLUGIN?

- Model **before** validated merge:
 - You push to team repo, then test



WHAT IS THE VALIDATED MERGE PLUGIN?

- Model **with** Validated Merge:
 - Jenkins becomes git repo
 - You push to Jenkins
 - Jenkins tests, then push to team repo



BENEFITS: NO BROKEN BUILDS

- A mistake will not block others
- Only push after changes are validated
- It is the Unbreakable Build pattern

BENEFITS: TESTS ASYNCHRONOUSLY

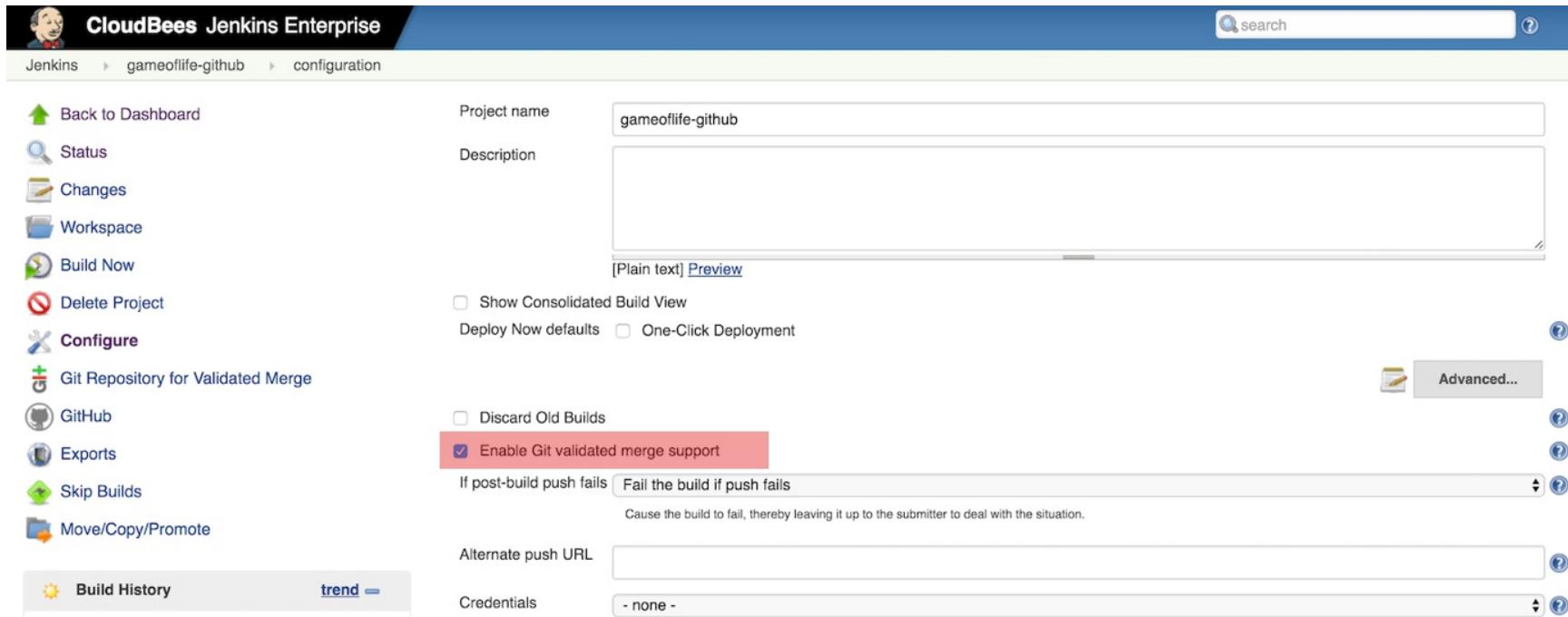
Make
change and move on
Let
Jenkins follow through

BENEFITS: TESTS ON THE SERVER

Your laptop has more important things to do Highly concurrent, environment-dependent test executions

WHERE DO WE START?

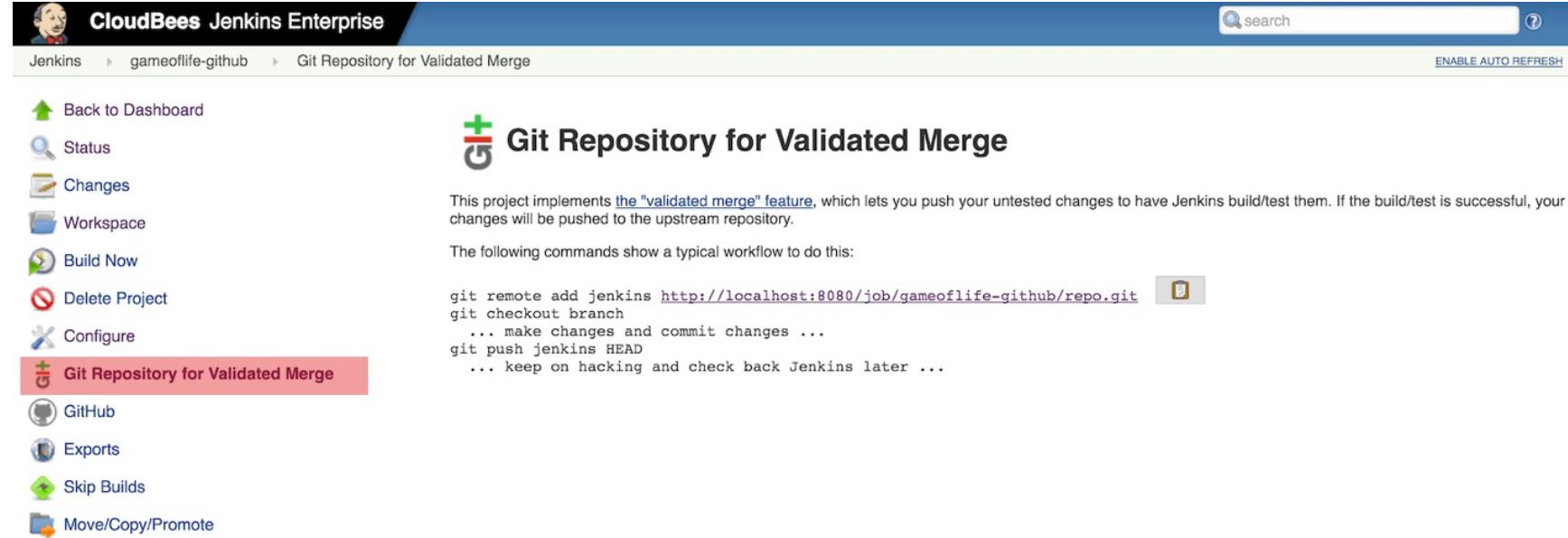
- To enable, just one checkbox:



The screenshot shows the Jenkins configuration interface for a project named "gameoflife-github". On the left, there's a sidebar with various Jenkins-related links like Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, Configure, Git Repository for Validated Merge, GitHub, Exports, Skip Builds, and Move/Copy/Promote. The main panel has fields for Project name (set to "gameoflife-github") and Description (empty). Below these are several checkboxes: "Show Consolidated Build View" (unchecked), "Deploy Now defaults" (unchecked), "One-Click Deployment" (unchecked), "Discard Old Builds" (unchecked), and "Enable Git validated merge support" (checked). There's also a "Plain text" link and a "Preview" button. At the bottom, there are sections for "If post-build push fails" (with options "Fail the build if push fails" and "Cause the build to fail, thereby leaving it up to the submitter to deal with the situation"), an "Alternate push URL" field, and a "Credentials" dropdown set to "- none -". A "Advanced..." button is also present.

WHERE DO WE START?

- Git access coordinate from project page:



The screenshot shows the Jenkins interface for a project named "gameoflife-github". The main title is "Git Repository for Validated Merge". On the left, there's a sidebar with various Jenkins management links like Back to Dashboard, Status, Changes, Workspace, Build Now, Delete Project, Configure, GitHub, Exports, Skip Builds, and Move/Copy/Promote. The "Git Repository for Validated Merge" link is highlighted with a red box. The main content area describes the validated merge feature and provides a command-line workflow example:

```
git remote add jenkins http://localhost:8080/job/gameoflife-github/repo.git
git checkout branch
... make changes and commit changes ...
git push jenkins HEAD
... keep on hacking and check back Jenkins later ...
```

WHERE DO WE START?

- Two protocols supported:

SSH

See system config to fix the port number for production use

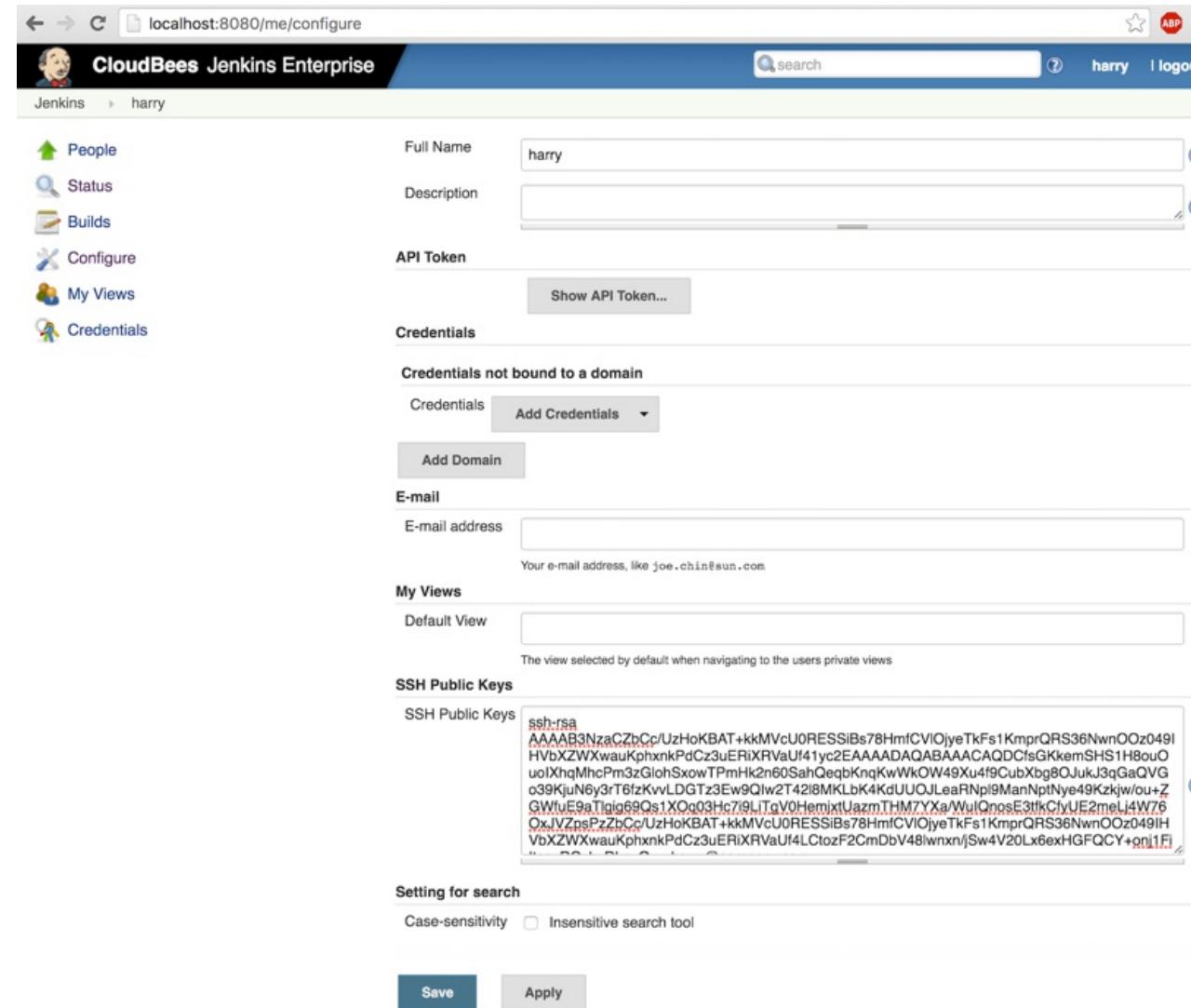
HTTP

The same URL as the “Git repository” page in Jenkins Awkward to handle authentication

WHERE DO WE START?

Developer Workflow

Register SSH public key : http://<JENKINS_URL>/me/configure



The screenshot shows the Jenkins 'Configure User' page at localhost:8080/me/configure. The user 'harry' is selected. In the 'SSH Public Keys' section, there is a text input field containing an SSH RSA key:

```
ssh-rsa AAAAB3NzaC1eZbCpUzHoKBAT+kkMVCu0RESSIBs78HmfCVIOjyeTkFs1KmprQRS36NwnOOz0491Hv4XZWxwauKphnxnPdcz3uERiXRVaU41yc2EAAAQABAAACAQDCfGKkemSHS1H8ouOuo1XhqMhcPm3zGlohsxowTPmHk2n60SahQeqbKngKwWkOW49Xu4fCubXbg8OJuKj3gGaQVGc39KjuN6y3rT6lzkVvLDGTz3Ew9Qlw2T42l8MKLbK4KdUUQJLeaRNpI9ManNptNye49Kzkjw/ou+ZGWfuE9aTlg69Qs1XOq03Hc7i9LITgVOHemixtJazmtTHM7YXa/WuIQnosE3tkCfyUE2meLj4W76OxIVZpsPzBCC/UzHoKBAT+kkMVCu0RESSIBs78HmfCVIOjyeTkFs1KmprQRS36NwnOOz0491Hv4XZWxwauKphnxnPdcz3uERiXRVaU4LCtozF2CmDbV48lwrxn/Sw4V20Lx6exHGFQCY+on1Fj
```

Below the key, there are 'Save' and 'Apply' buttons.



WHERE DO WE START?

Developer Workflow

- Adds Jenkins as a new git remote repository

```
$ git remote add jenkins ssh://server:port/JOBNAME/
```

- Get changes from team repo

```
$ git pull origin mybranch
```

- Hack

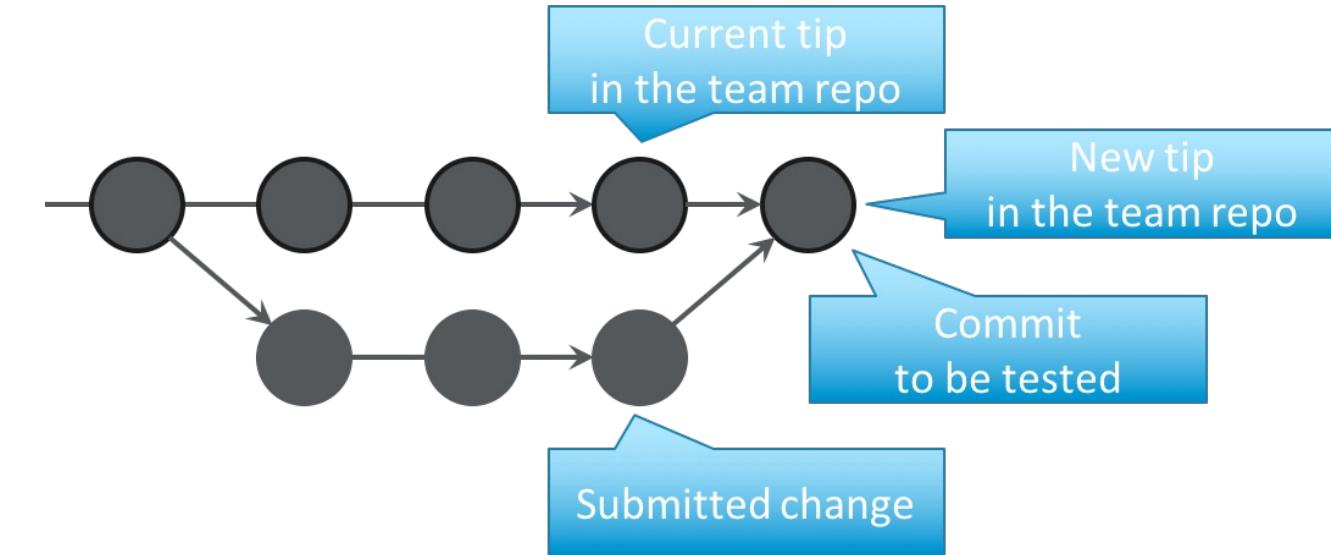
```
$ vi .  
$ git commit
```

- Push to the « Jenkins » remote instead of team one

```
$ git push jenkins mybranch
```

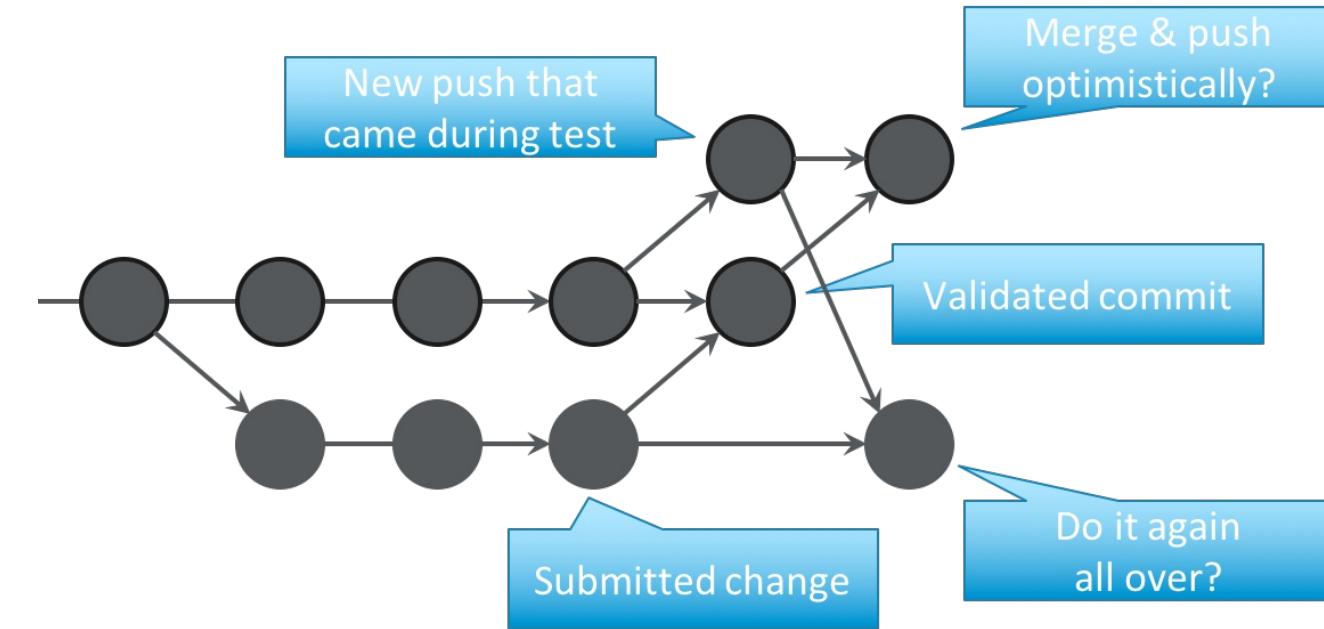
WHERE DO WE START?

- Pre-build merge behavior
 - Jenkins needs to test what becomes the new tip:



WHERE DO WE START?

- Post-build merge behavior
 - What if new push is made to the team repo?



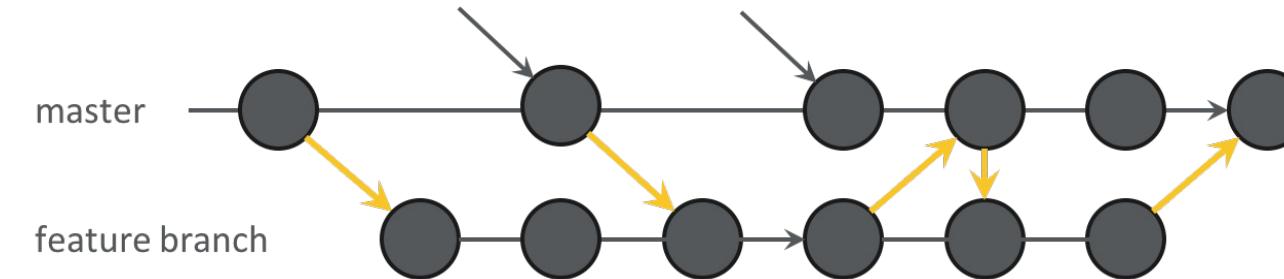
Or fail and leave it to the developer to resolve

ALTERNATIVES: GERRIT

- Gerrit is a code review tool
- Gerrit trigger plugin for Jenkins:
 - Jenkins acts as a code reviewer (review====test)
- Works great if you want code review
- **Problems:**
 - Gerrit needs to own the repository
 - More client side changes required

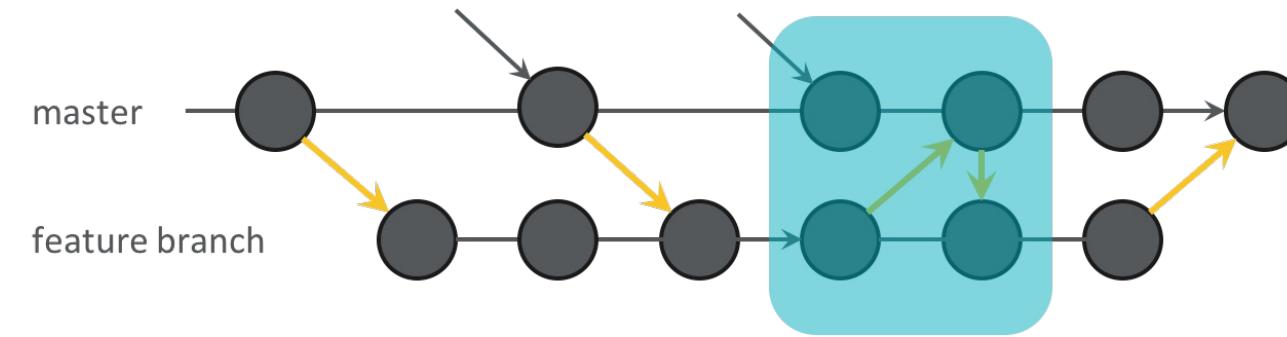
ALTERNATIVES: SUBVERSION?

- Yes! Svnmerge plugin to the rescue
 - It's not Git, but Subversion has improved
- You commit to branches
- Jenkins will merge and commit to upstream



CAVEAT

- Push and rebase need to happen in one go:



LAB EXERCISE:

Lab 11: Validated Merge

PULL-REQUEST BUILDER FOR GITHUB

WHAT IS PULL-REQUEST BUILDER FOR GITHUB

The Pull-Request Builder for GitHub plugin let you configure Jenkins to verify pull-request on a GitHub project and check the proposed changes validates Continuous Integration criteria set on your Jenkins instance.

BENEFITS: EFFICIENCY

The GitHub Pull Requester Plugin can:

- automatically run unit/integration tests.
- merge the pull request upstream if it passes the unit/integration tests

WHERE DO WE START?

- Download and install the GitHub Pull Request Builder Plugin.
- Configure the Plugin, add users to the Admin, and White List

PIPELINE

Formerly known as Workflow

WHAT IS A « JENKINS WORKFLOW »

CD is about setting a “pipeline”, from SCM commit to deployed application



Jenkins can chain jobs this way, but ...

JENKINS CD PIPELINE

a real world CD pipeline is way more complex !

- requires (complex) conditional logic
- requires resources allocation and cleanup
- involves human interaction for manual approval
- should be resumable at some point on failure

JENKINS CD PIPELINE

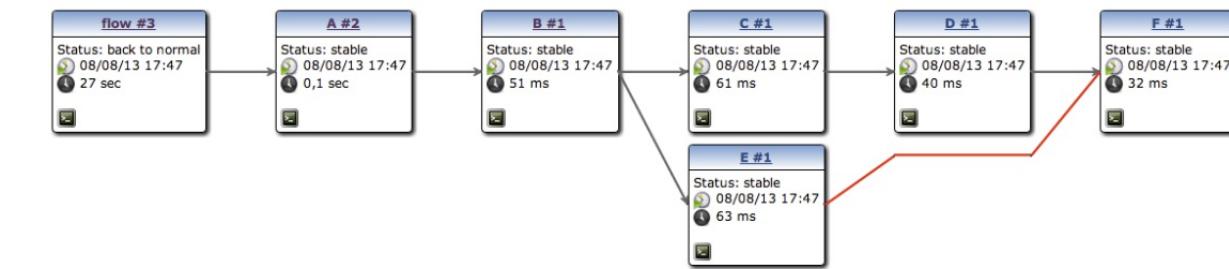
- many related plugins
 - conditional build steps
 - parameterized trigger
 - promotions
 - ...
- Major issue : pipeline configuration is scattered in various jobs

CD PIPELINE VISUALIZATION

- pipeline plugin to render “linear” pipeline

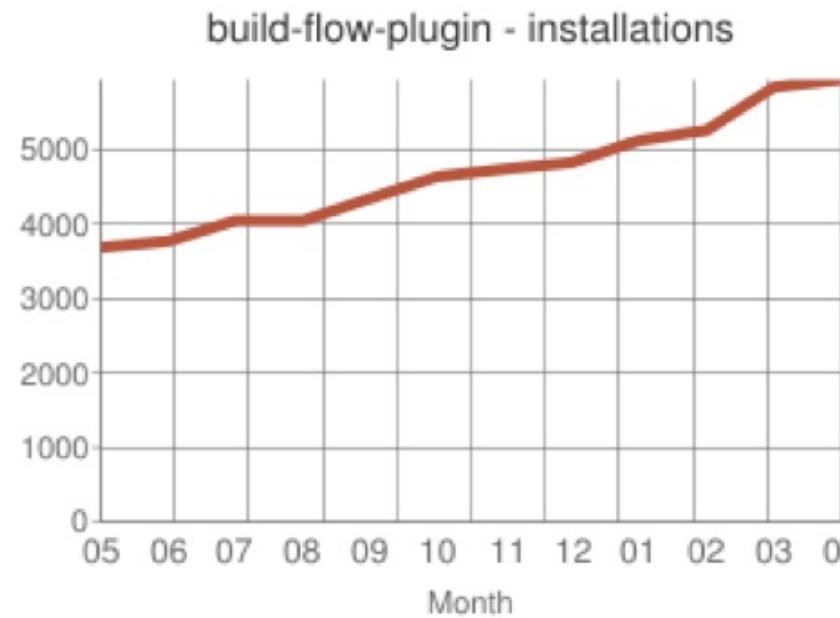


- build-graph-view to render complex ones



BUILD-FLOW PLUGIN

- OSS, considered a proof of concept



- define CD pipeline as (possibly complex) job orchestration written with a Groovy DSL
- largely adopted
 - **demonstrates direction is correct**
- But major technical limitations

CLOUDBEES PIPELINE PLUGIN

- inspired by build-flow
- Groovy DSL to orchestrate build steps (optionally jobs)
- **extensible** DSL syntax
- do supports **savepoints**
- advanced **visualization**
- plain text : Can be stored in SCM

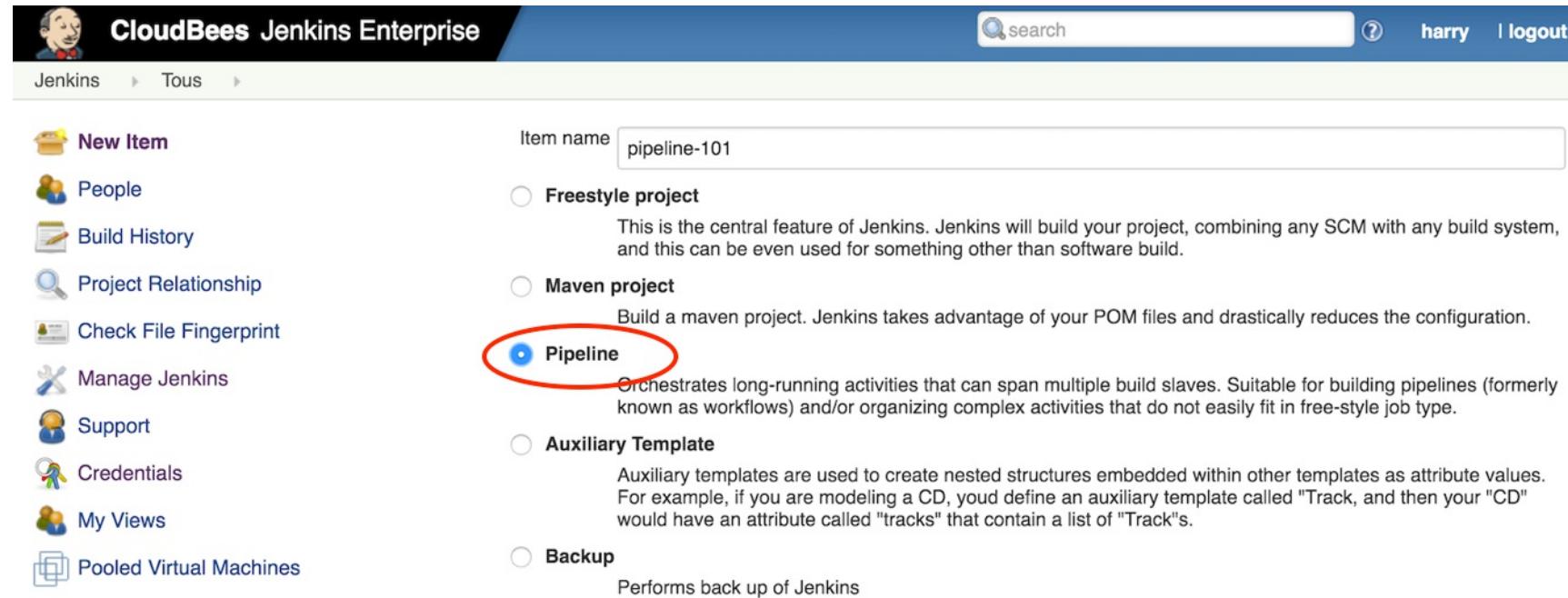
GROOVY DSL ?

Groovy based “Domain Specific Language” to
manage build step orchestration

- Groovy is widely used in Jenkins ecosystem.
You can leverage on Groovy experience
- If you don’t like/care about Groovy, just consider
the DSL as a dedicated simple syntax

PIPELINE 101

- A Pipeline is a new type of **job**:



The screenshot shows the Jenkins web interface for creating a new item. On the left, there's a sidebar with various Jenkins management links like 'New Item', 'People', 'Build History', etc. The main area is titled 'CloudBees Jenkins Enterprise' and shows a form for creating a new job. The 'Item name' field contains 'pipeline-101'. Below it, there are several project types to choose from:

- Freestyle project**: This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Maven project**: Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Pipeline**: Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Auxiliary Template**: Auxiliary templates are used to create nested structures embedded within other templates as attribute values. For example, if you are modeling a CD, you'd define an auxiliary template called "Track, and then your "CD" would have an attribute called "tracks" that contain a list of "Track"s.
- Backup**: Performs back up of Jenkins

PIPELINE 101

Can define DSL in Jenkins, or store in SCM
=> possibly distinct pipeline per branch

Pipeline

Definition

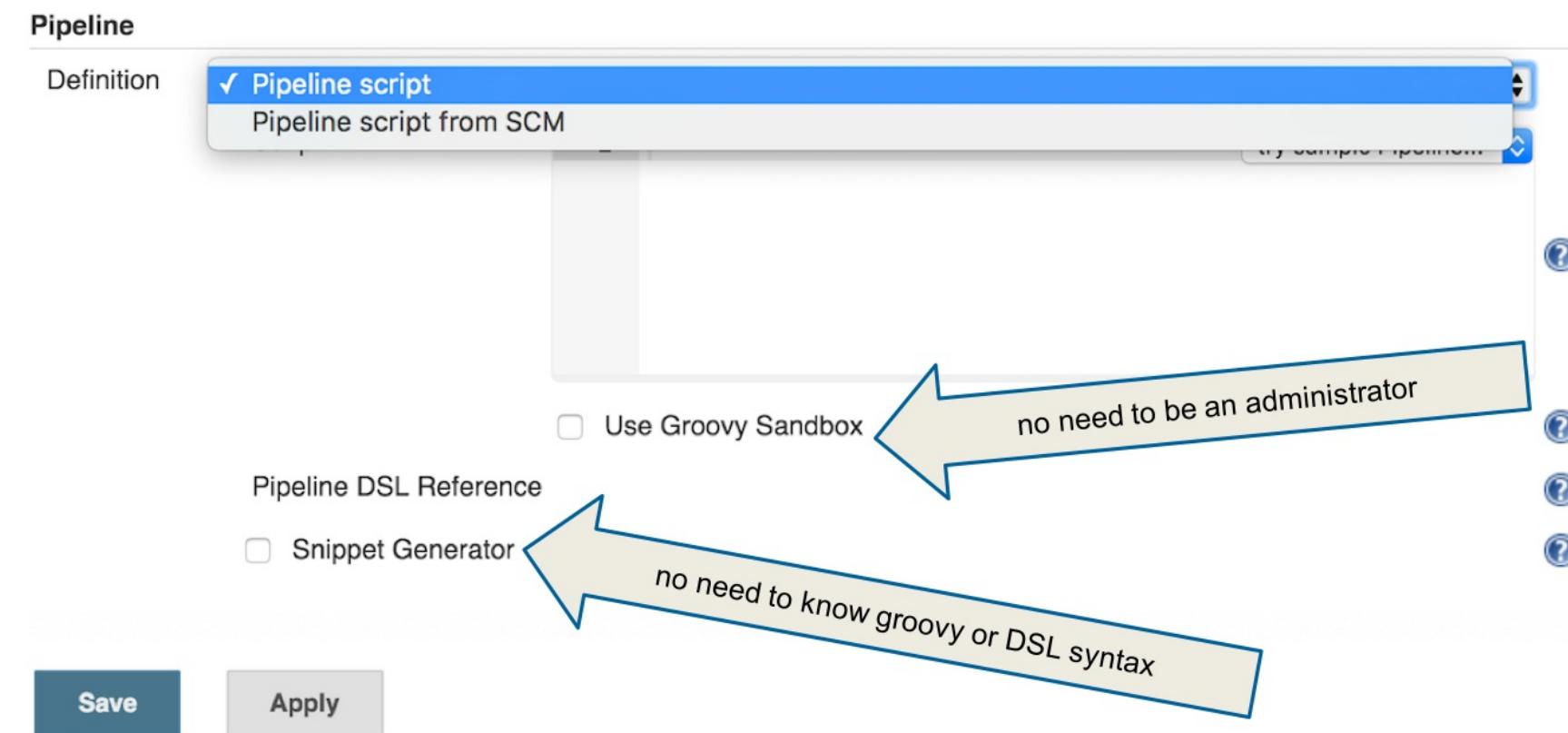
Pipeline script
Pipeline script from SCM

Use Groovy Sandbox

Pipeline DSL Reference

Snippet Generator

Save Apply



PIPELINE 101 - BASIC DSL SYNTAX

- DSL is used to define when/how to run build steps
- Those (most of them) will run on a **node**, i.e. a build executor
- Example with **sh**:

```
node('linux') { // run on node that are labelled as "linux"  
  sh 'echo hello world' // a DSL keyword to run a shell script build step:  
}
```



PIPELINE SNIPPET GENERATOR

- No need to learn DSL syntax !

Snippet Generator ?

Steps

Sample Step node: Allocate node ?

Label rhel ?

Label is serviced by 1 node

Generate Groovy

```
node('rhel') {
    // some block
}
```



Pipeline 101 - BUILD STEPS

- Some have native DSL support

Snippet Generator ?

Steps

Sample Step git: Git ?

Repository URL: `https://github.com/jenkinsci/docker.git` ?

Branch: `feature-01`

Credentials: `harry/*********`  Add

Include in polling? Include in changelog?

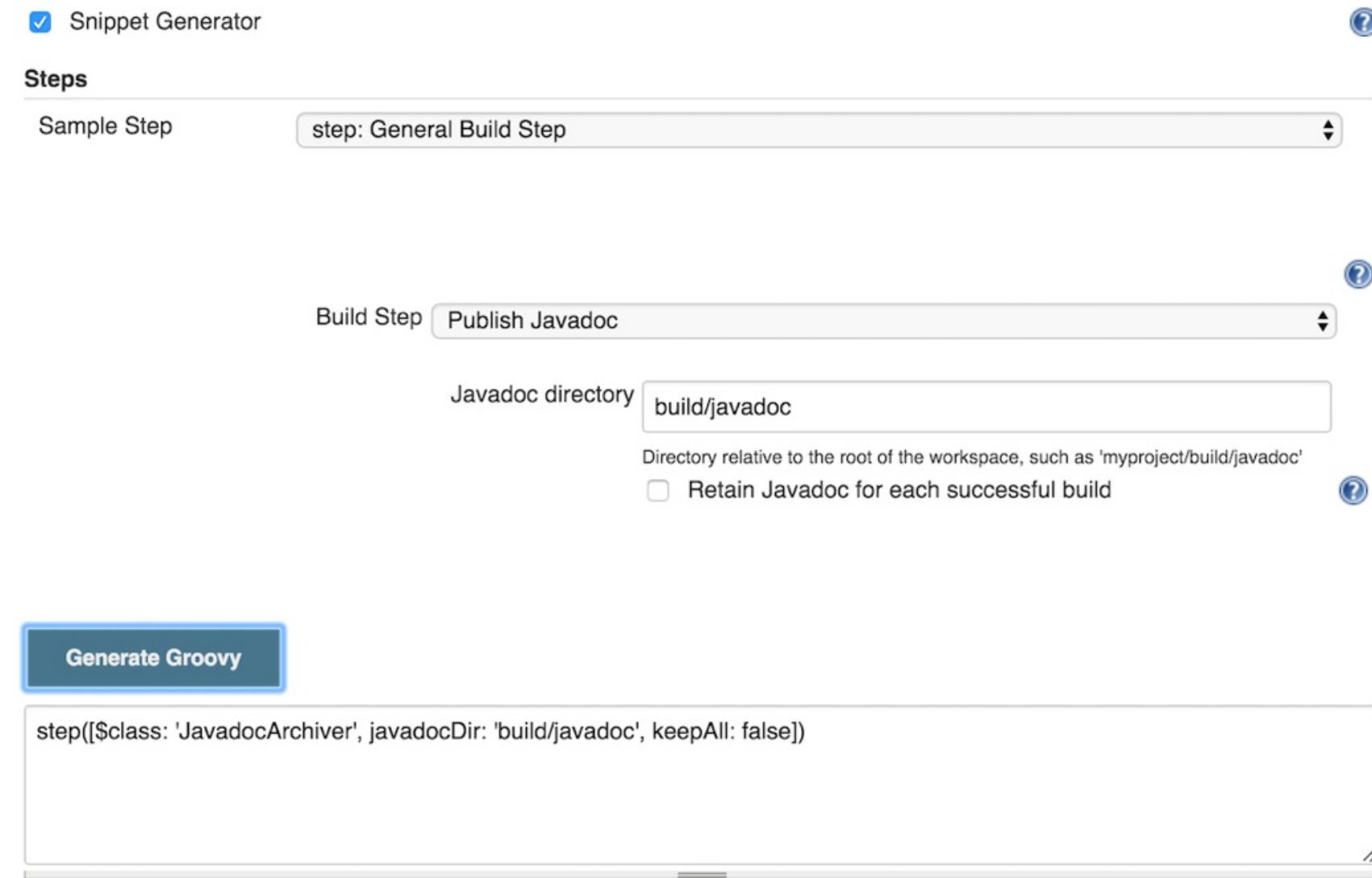
Generate Groovy

```
git branch: 'feature-01', credentialsId: '475118bd-bbc3-4284-82f2-f38f50cb67aa', url: 'https://github.com/jenkinsci/docker.git'
```



PIPELINE 101 - BUILD STEPS

Some don't (yet) have native DSL support and require some hack-ish invocation syntax - but **still work**:



The screenshot shows the CloudBees Pipeline interface. At the top, there is a checkbox labeled "Snippet Generator" which is checked. Below it, under the "Steps" section, there is a "Sample Step" and a dropdown menu set to "step: General Build Step".

Further down, there is a "Build Step" section with a "Publish Javadoc" option selected. It includes a "Javadoc directory" field containing "build/javadoc" and a note: "Directory relative to the root of the workspace, such as 'myproject/build/javadoc'". There is also a checkbox for "Retain Javadoc for each successful build".

At the bottom, there is a "Generate Groovy" button, and below it is a code editor containing the following Groovy code:

```
step([$class: 'JavadocArchiver', javadocDir: 'build/javadoc', keepAll: false])
```



INVOCATION SYNTAX

- Most build steps can be used this way **today**, they may evolve to offer native pipeline support:

```
step( // DSL keyword to invoke arbitrary step
    [ // groovy map
        $class: 'build_step_class_name', // Same as in job's config.xml on disk
        constructor_argument: 'value',
        constructor_argument_2: 'value'
    ]
)
```

HUMAN INTERACTION

A real world CD pipeline involves human interaction:

- manual validation steps
- approval

Jenkins users used to rely on Promoted Builds Plugin
but then lost the “pipeline execution flow” view

HUMAN INTERACTION

```
input // DSL keyword for interactive step  
message: 'Is it ok to deploy ?',  
ok: 'Yes!', // Button caption  
submitter: 'boss' // Role restriction
```

- Will give this "output":



Console Output

```
Started by user Nicolas De loof  
Running: Input  
Is it ok to push to server ?  
Yes ! or Abort
```

EXECUTION CONTROL

- Try up to N times:

```
retry(10) {  
    // some block  
}
```

- Pause the flow:

```
sleep time: 10,  
unit: 'MINUTES'
```

- Wait for event:

```
waitFor {  
    // some block  
}
```

- Timeout an operation:

```
timeout(time: 100,  
       unit: 'SECONDS') {  
    // some block  
}
```



EXECUTION CONTROL

You can also rely on Groovy control flow syntax:

```
while(something) {  
    // do something  
    if (something_else) {  
        // do something else  
    }  
}
```

CONCURRENCY

Pipeline can execute tasks in parallel :

```
parallel
  firstBranch: {
    // do something
  },
  secondBranch: {
    // do something else
  }
```

FILE SYSTEM

- read file:

```
readFile file: 'some/file', encoding: 'UTF-8'
```

- write file:

```
writeFile file: 'some/file', text: 'hello'
```

BUILD-FLOW COMPATIBILITY

- build-flow might be deprecated at some time
- Pipeline can trigger a job too:

```
build job: 'foo', wait: false
```

so **could** run a build flow script with some minor changes

PIPELINE STAGE VIEW

WHAT IS PIPELINE STAGE VIEW?

- When you have complex builds pipelines, it is useful to be able to see the progress of each stage.
- To that end, CJE includes an extended visualization of pipeline build history on the index page of a flow project.

Tip : (You can also click on Full Stage View to get a full-screen view.)

DEFINING STAGES

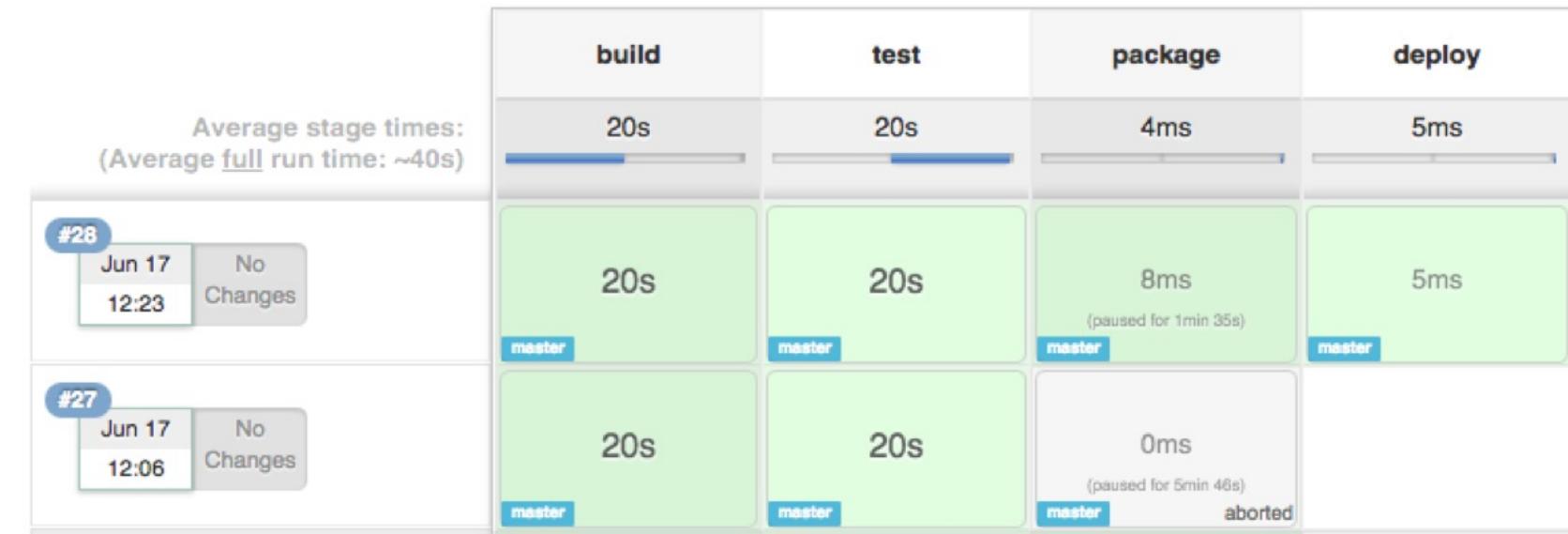
- The stage keyword let you label parts of the pipeline :

```
stage 'build'  
  // Steps to build your application  
  
stage 'test'  
  // Steps to run tests against application  
  
stage 'package'  
  // Steps to packaging  
  
stage 'deploy'  
  // GO in production
```

This will be rendered by visualization

VISUALIZING STAGES

After the first build, you can see your stages (and their states, logs, etc.) :



CONCURRENT STAGES

Stages also let you control concurrency:

```
stage name: 'load-tests',
  concurrency: 4 // spread across 4 nodes
```

BENEFITS: SIMPLICITY

By adding Stage View to your Pipeline jobs it allows users to easily identify the progress of each stage within the job.

CHECKPOINTS

WHAT ARE CHECKPOINTS?

- The checkpoints plugin, offered through CloudBees Jenkins enterprise package, allows for Pipelines to be resumed at specified points.
- By adding a simple line of code to a pipeline script, a failed Pipeline build can be started at that specific checkpoint.
- Data is always preserved with restarted Pipelines and it is easy to pick up where the build left off.

BENEFITS: RESTART PIPELINE WHERE THEY FAILED

- Without Checkpoints, failed long running Pipebuilds will need to be re-run from the start.
- With Checkpoints, you can simply place a checkpoint before a high risk operations. And if the build fails, Checkpoint will allow the build to be resumed from the checkpoint just before the failure.
- Caution: Checkpoints take additional disk space. You can delete a specific checkpoint to save space

WHERE DO WE START?

- Create or modify an existing Pipeline to include a Checkpoint. Where is there a high risk operation? A task that depends on network communications or a large amount of disk space?
- Add a checkpoint before the high risk operation. For example:

```
checkpoint 'Built & Tested'
```

WHERE DO WE START?

A note about Checkpoints:

- As started earlier, data is always preserved with Checkpoint'ed Pipelines and it is easy to pick up where the build left off
- This is accomplished by serializing the entire workspace and the pipeline script itself (hence the impact on disk usage)
- So updating a pipeline script and resuming it after a checkpoint will not run the new version of the pipeline
- This feature is really intended to resume the exact same pipeline from the checkpoint after an external issue has been fixed or resolved
- For example, compile, test, build stages pass, but deploy stage may fail due to connection issue to the deployment environment. You could resume from a checkpoint placed after the test stage to re-deploy once the connectivity issue is resolved

LAB EXERCISE:

Lab 12: Pipelines

TEMPLATE PLUGIN

Template Plugin Overview

WHAT ARE TEMPLATES?

- A template is a **reusable** piece of job configuration that only **exposes** a minimal and pre-defined set of parameters to users
- A template is internally translated to a “standard” Jenkins job configuration.
- A template **can extend** another template
- Changes to a template **immediately apply** to **everywhere** the template is used.

WHAT ARE TEMPLATES?

attribute

named & typed variable

model

collection of attributes (& super model)

instance

particular values for all attributes in a model

transformer

translates model instance into real item

template

model + transformer

WHAT ARE TEMPLATES?

- 5 different kinds of templates
 - Builder
 - Publisher
 - Job
 - Folder
 - Auxiliary (model only)

BENEFITS: MAINTAINABILITY

- Manage a large set of jobs **without duplicating** configuration:
 - Configuration update is easily done in one place instead of doing a dangerous mass update using the groovy console for example
 - Easier onboarding of new teams/applications
- Manage a large set of jobs **without unexpected subtle differences**
 - You are explicitly controlling what may be different/customized

BENEFITS: DELEGATION

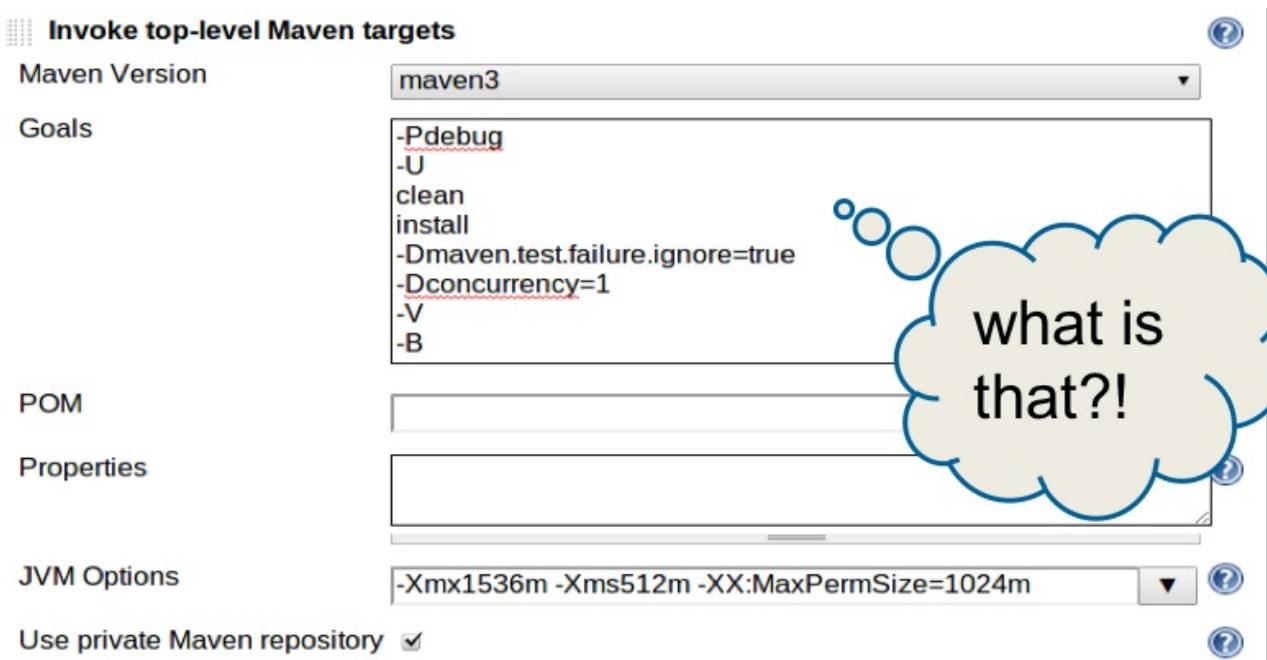
- Delegate jobs creation **without allowing users to do anything**:
 - They can create jobs but they can customize only what the administrator allows
 - Enforce common enterprise standard across all applications (for example, Static Code Analysis to be done all projects)
- Delegate jobs creation **without exposing** to users all advanced options and switches that Jenkins may provide
 - Configuration settings are limited by the administrator who can specialize them to his targeted audience

TEMPLATE PLUGIN

Builder templates

WHAT ARE BUILDER TEMPLATES?

- A builder template allows to create **your own simplified and reusable** build step



WHAT ARE BUILDER TEMPLATES?

- How does it work?
 - transformer produces a Jenkins build step
 - often a shell/batch script but need not be
 - transformation run during build, so can access live data
 - new builder immediately run

BENEFITS: SIMPLICITY

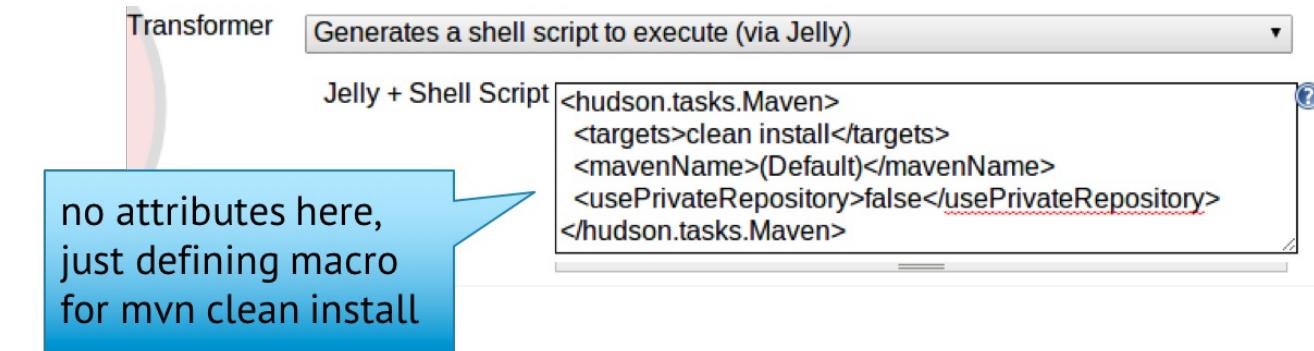
- By using builder templates you avoid
 - copying & modifying too complex build steps
 - writing heavyweight Jenkins plugins (learning curve)

WHERE DO WE START?

- Using general transformers in a builder
 - awkward to do simple things
 - but can generate any builder: Gradle, wsadmin, &c.
 - useful if builder's plugin does tricky things
 - copy XML from a prototype job, then templatize
 - RFE in progress to load sample XML for you

WHERE DO WE START?

- Using general transformers in a builder
 - Groovy for scripting power;
 - Jelly for XML hygiene (deprecated)

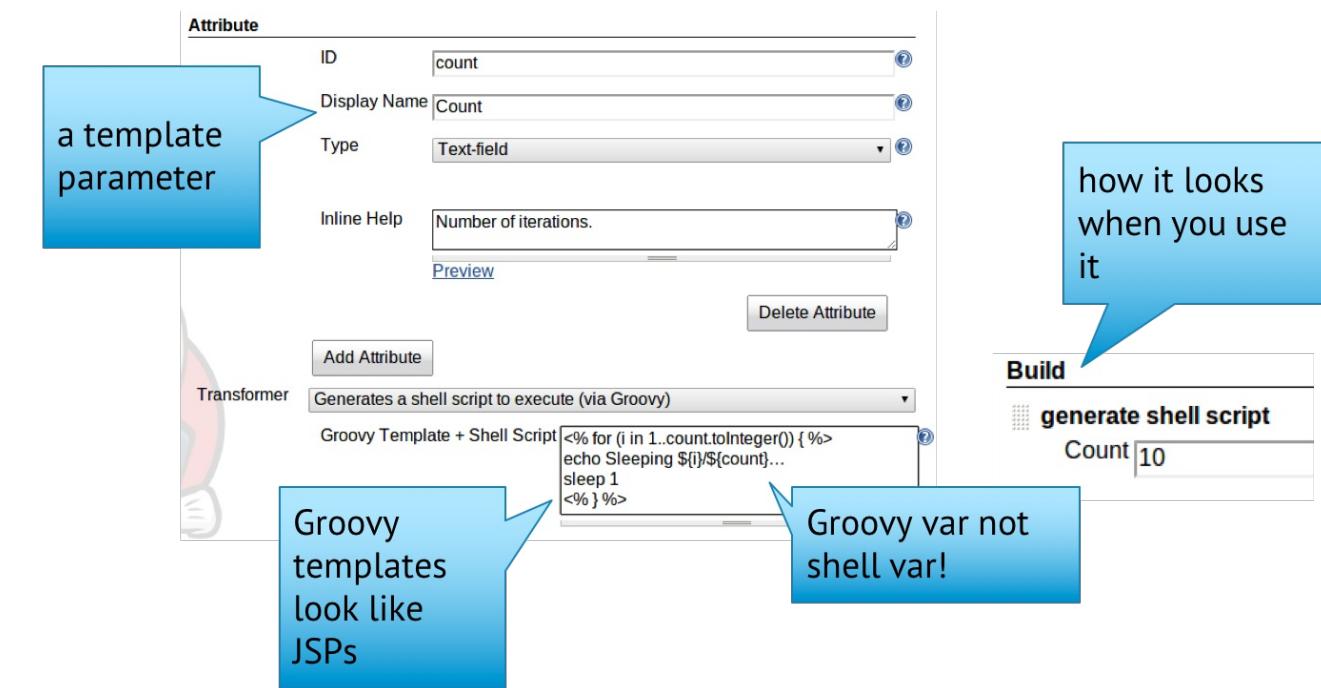


WHERE DO WE START?

- Generate shell scripts and run
 - easier if you would normally use a shell script anyway
 - write a template of a shell script (usually in Groovy)
 - **do not confuse Groovy variables with shell variables**

WHERE DO WE START?

- Generate shell scripts and run

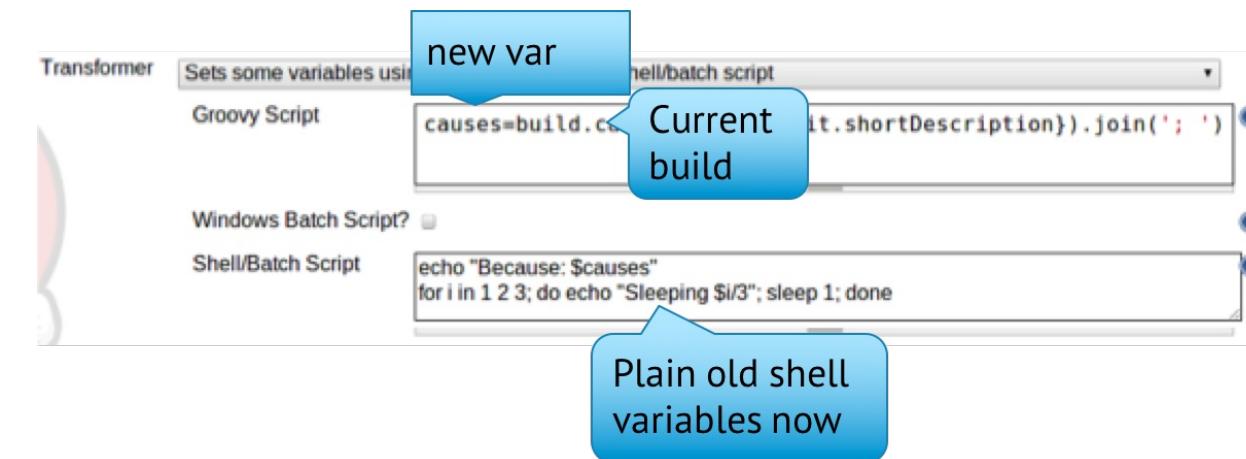


WHERE DO WE START?

- Predefine variables then run script
 - alternative keeping Groovy and shell syntax separated
 - supports Windows batch scripts as well
 - special characters (spaces, quotes, ...) preserved
 - all control flow must be done by shell

WHERE DO WE START?

- Predefine variables then run script
 - Definition:



- Output:

```
[workspace] $ /bin/sh -e /tmp/hudson6506427550849825755.sh
Because: Started by user anonymous
Sleeping 1/3
Sleeping 2/3
Sleeping 3/3
```

TEMPLATE PLUGIN

Publisher templates

WHAT ARE PUBLISHER TEMPLATES?

- much like builders, except for **post-build actions**
- publish **HTML or test results**, send email, ...
- only general transformers (Groovy/Jelly) supported



WHERE DO WE START?

The screenshot shows the Jenkins configuration interface for a job. On the left, there's a 'Transformer' section with an 'Attribute' panel for a 'dir' attribute. The 'Type' is set to 'Text-field' and the 'Inline Help' provides an example: 'Name of source directory, e.g. <code>src</code>'.

A blue callout bubble points to the 'Script' field of the transformer, containing the Groovy code: "\${java.util.regex.Pattern.quote(dir)}". The code is used to escape the directory name for use as a regex pattern.

To the right, under 'Post-build Actions', there's a 'Scan for Warnings' action with a 'Source Directory' set to 'sources'. A thought bubble says 'this is easy!'

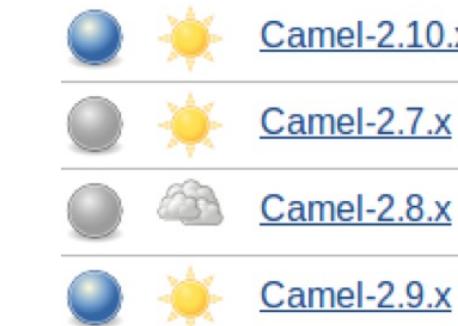
```
<hudson.plugins.warnings.WarningsPublisher>
  <thresholdLimits>
    <pluginName>Java Compiler (javac)</pluginName>
    <defaultEncoding>UTF-8</defaultEncoding>
    <thresholds>
      <dontComputeNew>true</dontComputeNew>
      <doNotResolveRelativePaths>true</doNotResolveRelativePaths>
      <includePattern>.+${java.util.regex.Pattern.quote(dir)}/.+</includePattern>
    </thresholds>
  </pluginName>
</hudson.plugins.warnings.WarningsPublisher>
```

TEMPLATE PLUGIN

Job templates

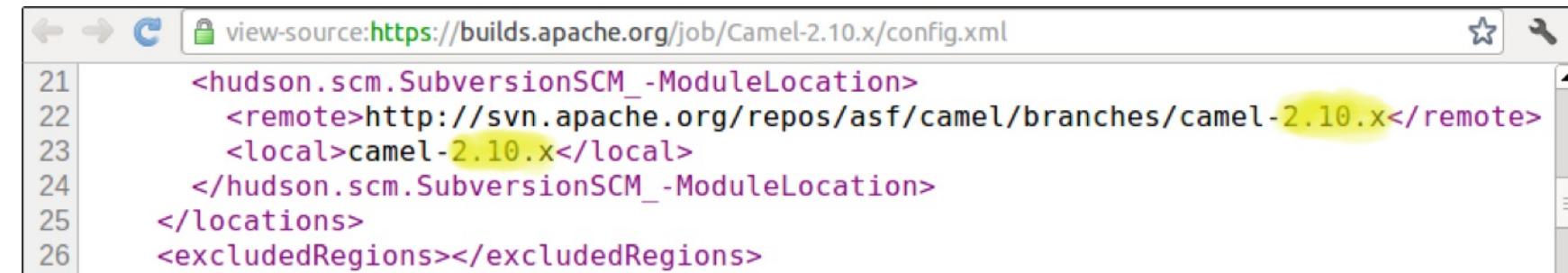
WHAT ARE JOB TEMPLATES?

- A maze of twisty little projects, all alike
 - many Jenkins jobs are very similar
 - want to capture the common parts
 - not everyone creating a job is a Jenkins expert
 - managers want to know what varies and what does not
 - security setup may prohibit arbitrary job configs
 - job templates to the rescue!



WHERE DO WE START?

- Defining a job template
 - start with config.xml from a working job
 - decide which parts should be customizable



The screenshot shows a browser window displaying the configuration XML for a Jenkins job. The URL in the address bar is <https://builds.apache.org/job/Camel-2.10.x/config.xml>. The code is color-coded for syntax highlighting:

```
21 <hudson.scm.SubversionSCM_-ModuleLocation>
22   <remote>http://svn.apache.org/repos/asf/camel/branches/camel-2.10.x</remote>
23   <local>camel-2.10.x</local>
24 </hudson.scm.SubversionSCM_-ModuleLocation>
25 </locations>
26 <excludedRegions></excludedRegions>
```

A yellow highlighter has been used to emphasize the string "camel-2.10.x" in the local path element.



WHERE DO WE START?

- Defining a job template
 - define and document attributes

ID	<input type="text" value="version"/> ?
Display Name	<input type="text" value="Version"/> ?
Type	<input type="text" value="Text-field"/> ?
Inline Help	<div><p>Version of Camel, such as <code>2.7.x</code>.</p>?</div>
	Preview Hide preview
	Version of Camel, such as 2.7.x.
	Delete Attribute



WHERE DO WE START?

- Defining a job template
 - pick a transformation language (Groovy, Jelly)
 - make config.xml into a template

Transformer Jelly-based transformation

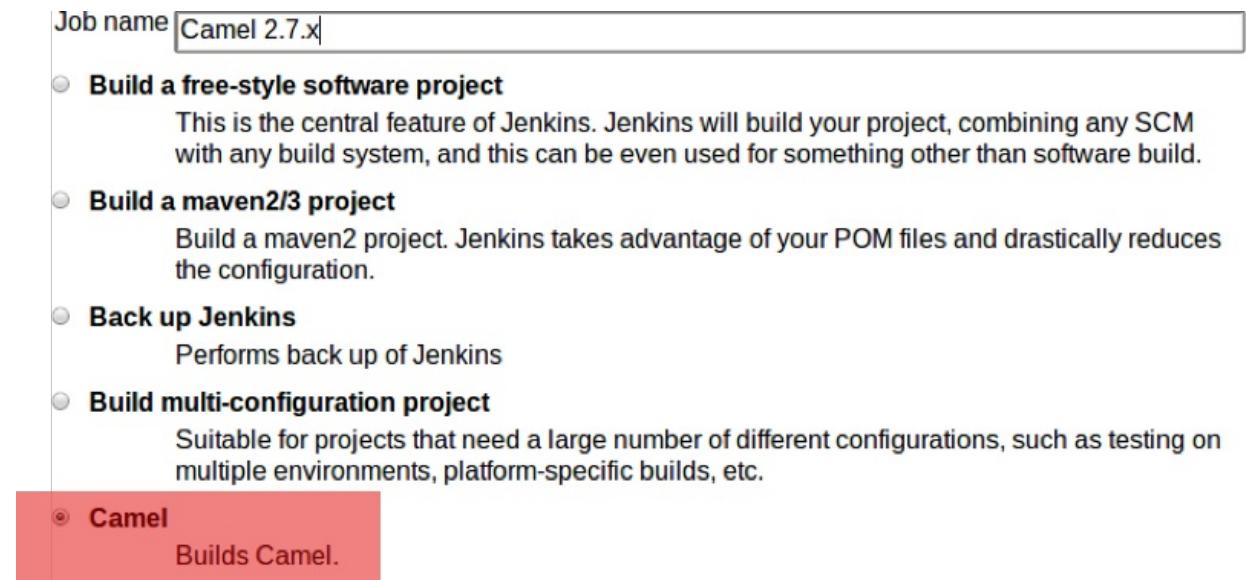
Script

```
<locations>
  <hudson.scm.SubversionSCM_-ModuleLocation>
    <remote>http://svn.apache.org/repos/asf/camel/branches/camel-${version}</remote>
    <local>camel-${version}</local>
  </hudson.scm.SubversionSCM_-ModuleLocation>
</locations>
```

Variable reference in Jelly

WHERE DO WE START?

- Instantiating a job template
 - “New Job” page:



Job name

- Build a free-style software project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Build a maven2/3 project**
Build a maven2 project. Jenkins takes advantage of your POM files and drastically reduces the configuration.
- Back up Jenkins**
Performs back up of Jenkins
- Build multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Camel**
Builds Camel.

- Configuration screen:



Name

Version ?

Save

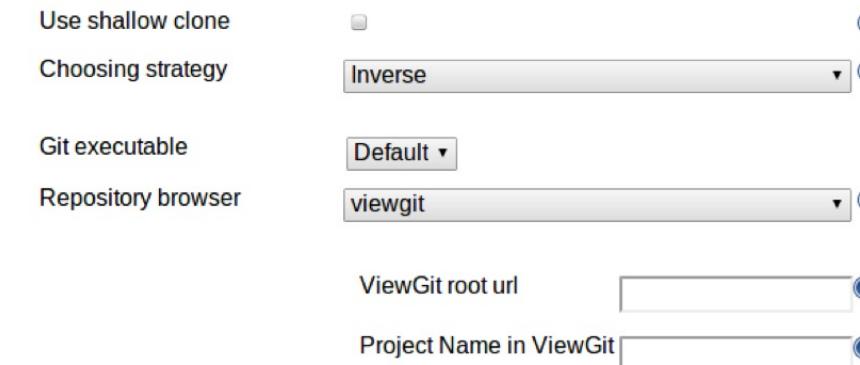


TEMPLATE PLUGIN

Auxiliary models and fancier template attributes

WHAT ARE ATTRIBUTES?

- Attribute types
 - so far all attributes have been simple text fields
 - complex templates demand more!
 - after all, plain Jenkins jobs have a rich config UI...



WHAT ARE ATTRIBUTES?

- Some basic attribute types:
 - text field ~ String (what we have seen)
 - text area ~ String (same but multiline)
 - checkbox ~ boolean



WHAT ARE ATTRIBUTES?

- Select from enumerated items:

The screenshot shows the configuration of a Deployment Target in the CloudBees Jenkins Platform. The target is named "target" and has a type of "Select a string among many". It includes three options: "None", "Staging", and "Production". Each option has a display name, value, and inline help message. The "Production" option is currently selected, with its inline help message being "Deploy into production.". There is also a note at the bottom stating "Where to deploy the build product (if anywhere.)". A "Save" button is visible at the bottom right.

ID	target																								
Display Name	Deployment Target																								
Type	Select a string among many																								
UI Mode	Radio button (can show inline help, but takes up more space)																								
Options	<table border="1"><tr><td>Display Name</td><td>None</td></tr><tr><td>Value</td><td>-</td></tr><tr><td>Inline Help</td><td>Do not deploy.</td></tr><tr><td colspan="2">Preview</td></tr><tr><td>Display Name</td><td>Staging</td></tr><tr><td>Value</td><td>stage</td></tr><tr><td>Inline Help</td><td>Deploy to staging server.</td></tr><tr><td colspan="2">Preview</td></tr><tr><td>Display Name</td><td>Production</td></tr><tr><td>Value</td><td>release</td></tr><tr><td>Inline Help</td><td>Deploy into production.</td></tr><tr><td colspan="2">Preview</td></tr></table>	Display Name	None	Value	-	Inline Help	Do not deploy.	Preview		Display Name	Staging	Value	stage	Inline Help	Deploy to staging server.	Preview		Display Name	Production	Value	release	Inline Help	Deploy into production.	Preview	
Display Name	None																								
Value	-																								
Inline Help	Do not deploy.																								
Preview																									
Display Name	Staging																								
Value	stage																								
Inline Help	Deploy to staging server.																								
Preview																									
Display Name	Production																								
Value	release																								
Inline Help	Deploy into production.																								
Preview																									
Add	Add																								
Inline Help	Where to deploy the build product (if anywhere.)																								

WHAT ARE ATTRIBUTES?

- Jenkins-specific attribute types
 - “item” (~ job/project)
 - installation of some type of tool (e.g. JDK)
 - “component” of some type, such as builder

WHAT ARE ATTRIBUTES?

- Jenkins-specific attribute types:

- Definition:

ID 

Display Name 

Type  

Fully-qualified class name of a Describable subtype 

Inline Help 

[Preview](#)

- What users see:

Build Steps

[Save](#) [Add ▾](#)

- Copy artifacts from another project
- Execute Windows batch command
- Execute shell
- Import artifacts from an exported permalink
- Invoke Ant**
- Invoke top-level Maven targets
- Take backup
- Trigger/call builds on other projects

com.cloudbees.hudson.plugins.modeling.impl.builder.RuntimeShellScriptBuilder

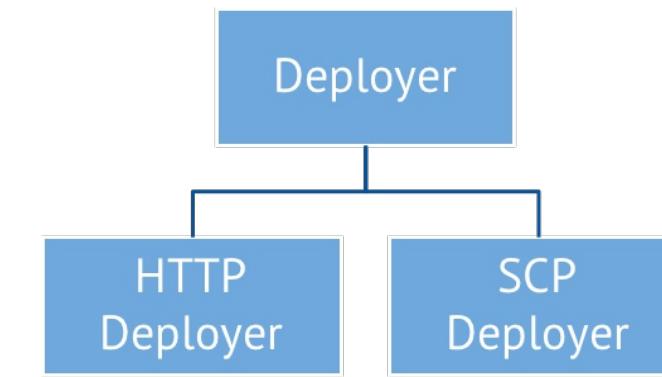


WHAT ARE AUXILIARY MODELS?

- Auxiliary models and inheritance:
 - like defining your own struct/class (with config GUI)
 - model only used inside other models
 - object-oriented style
 - “abstract” models
 - subtypes
 - polymorphic attributes
 - virtual attributes
 - used for complex, structured configuration

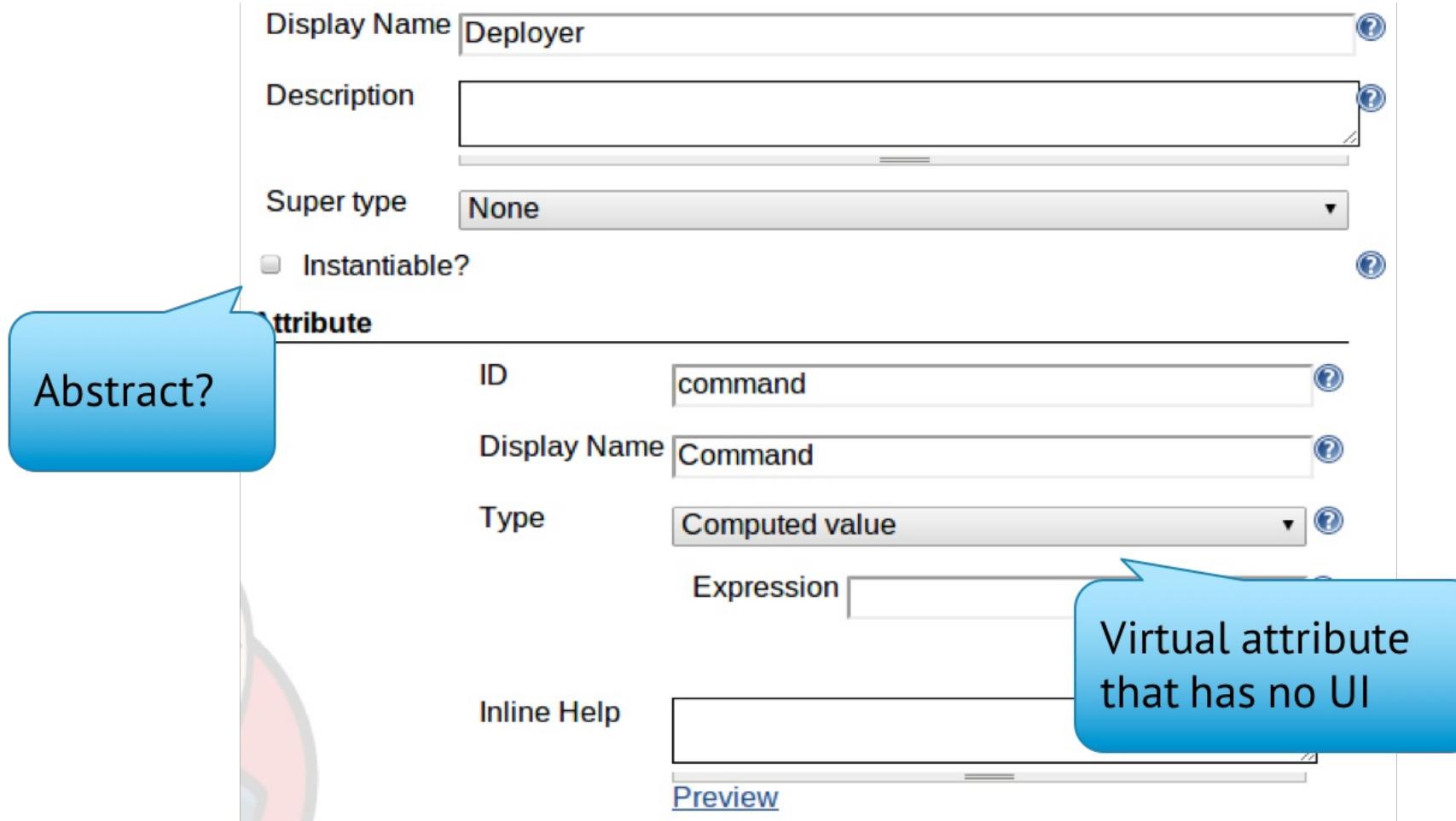
WHERE DO WE START?

- Auxiliary model example:
 - Scenario
 - We build webapp
 - Team decides where to deploy
 - Two types of deployment (via HTTP or SCP)
 - How we model this
 - Via 3 models
 - With inheritance



WHERE DO WE START?

- Defining an abstract auxiliary model:



The screenshot shows a configuration interface for defining an auxiliary model. On the left, there's a sidebar with a red icon and the text "CloudBees Jenkins Platform". The main area has a header "Define Model" with tabs "Model" and "Build Step". Below the tabs, there are sections for "Model Name" (set to "Deployer") and "Model Type" (set to "Abstract"). A large blue callout bubble labeled "Abstract?" points to the "Model Type" section. The "Attributes" section contains fields for "ID" (set to "command"), "Display Name" (set to "Command"), "Type" (set to "Computed value"), and "Expression" (empty). A blue callout bubble labeled "Virtual attribute that has no UI" points to the "Expression" field. Other sections include "Super type" (set to "None"), "Instantiable?" (unchecked), "Attribute" (with ID "command" and display name "Command"), and "Inline Help" (empty). Buttons at the bottom include "Preview" and "Save".

WHERE DO WE START?

- Defining auxiliary model subtypes
 - Marked as “instantiable”
 - Extends from “Deployer”
 - “command” attribute concretely defined

Attribute	
ID	command
Display Name	Command
Type	Computed value
Expression	<code>curl -T <FILE> \${url}</code>
Variable reference	
Delete Attribute	



WHERE DO WE START?

- Defining auxiliary model subtypes
 - Another subtype defined similarly

ID	server	
Display Name	Server	
Type	Text-field	
Inline Help	Server name.	
Preview		
Delete Attribute		
ID	user	
Display Name	User Name	
Type	Text-field	
Inline Help	Login name on that server.	
Preview		

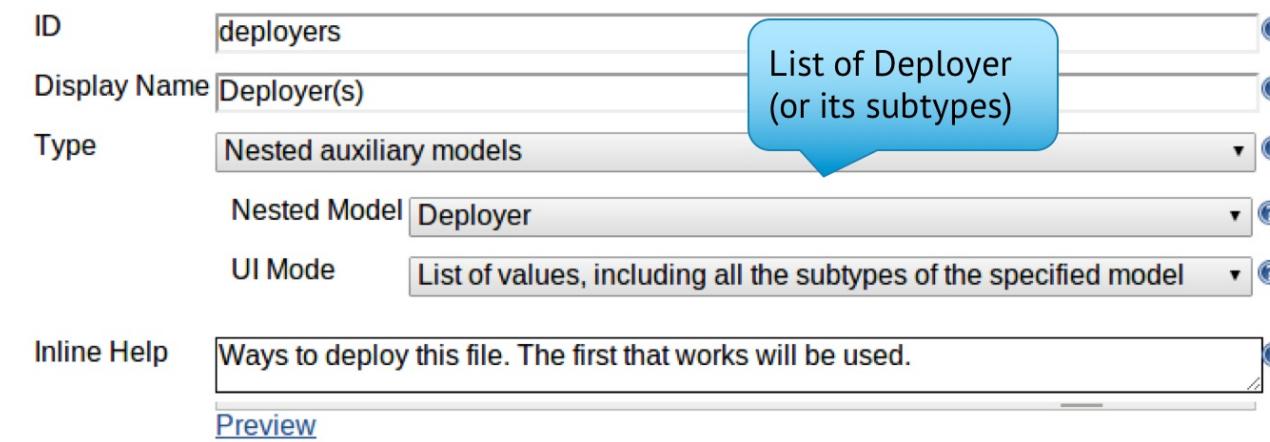
WHERE DO WE START?

- Use deployer from builder template
 - First, define the file attribute:

Attribute	
ID	file ?
Display Name	File Path & Name ?
Type	Text-field ?
Inline Help	Location of a file in the workspace. ?
Preview	

WHERE DO WE START?

- Then let users specify deployers



The screenshot shows a configuration interface for a 'Deployer' model. The fields are as follows:

- ID: deployers
- Display Name: Deployer(s)
- Type: Nested auxiliary models
- Nested Model: Deployer
- UI Mode: List of values, including all the subtypes of the specified model
- Inline Help: Ways to deploy this file. The first that works will be used.

A blue callout bubble points to the 'Type' field, which contains the value 'Nested auxiliary models'. The bubble contains the text 'List of Deployer (or its subtypes)'. Below the 'Type' field is a dropdown arrow icon.

WHERE DO WE START?

- Produce command that runs
 - Bit of Groovy to generate the command to run

```
cmd = deployers.  
    toList()*.  
    command*.  
    toString()*.  
    replace('<FILE>',file).  
    join('\n')
```

Transformer Sets some variables using Groovy then runs a shell/batch script

Groovy Script	<code>cmd = deployers.toList().collect({it.command.toString().replace('<FILE>', file)}).join(' ')</code>
Windows Batch Script?	<input type="checkbox"/>
Shell/Batch Script	<code>echo + \$cmd</code>

WHERE DO WE START?

- And the user experience ...

Build

Deploy File
File Path & Name ?

Deployer(s) **SSH Deployer**
Server ?
User Name ?

Delete

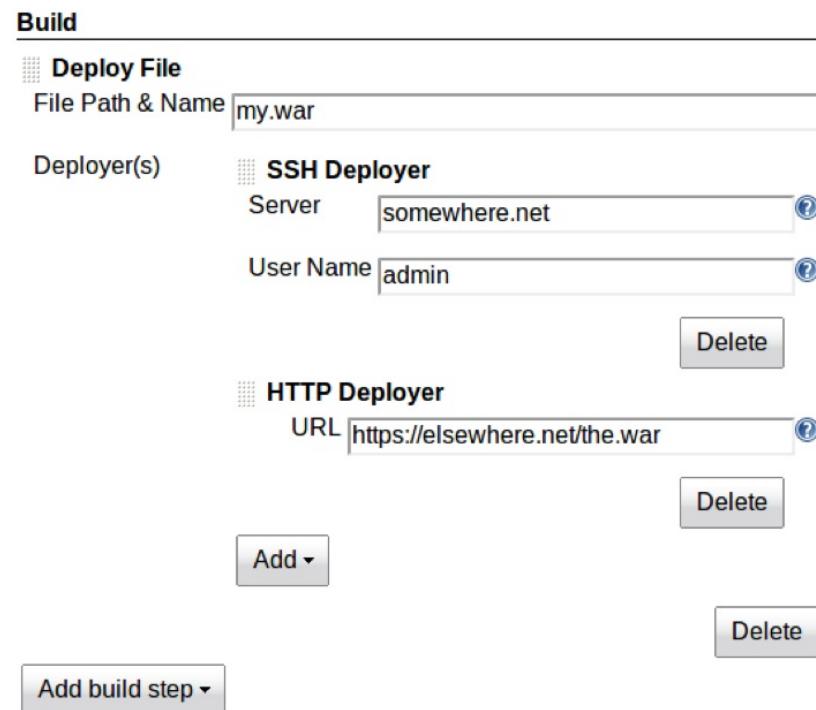
HTTP Deployer
URL ?

Delete

Add ▾

Delete

Add build step ▾



```
[workspace] $ /bin/sh -e /tmp/hudson8063706505027028169.sh
+ scp my.war admin@somewhere.net:~ || curl -T my.war https://elsewhere.net/the.war
Finished: SUCCESS
```

TEMPLATE PLUGIN

Folder templates

WHAT IS A FOLDER TEMPLATE?

- like a job template, but produces a whole folder
- may predefine:
 - job type restrictions
 - icon
 - columns
 - variables passed to jobs
- template produces config.xml as for a job
- can also define initial list of jobs
- admins can control what happens inside!

WHERE DO WE START?

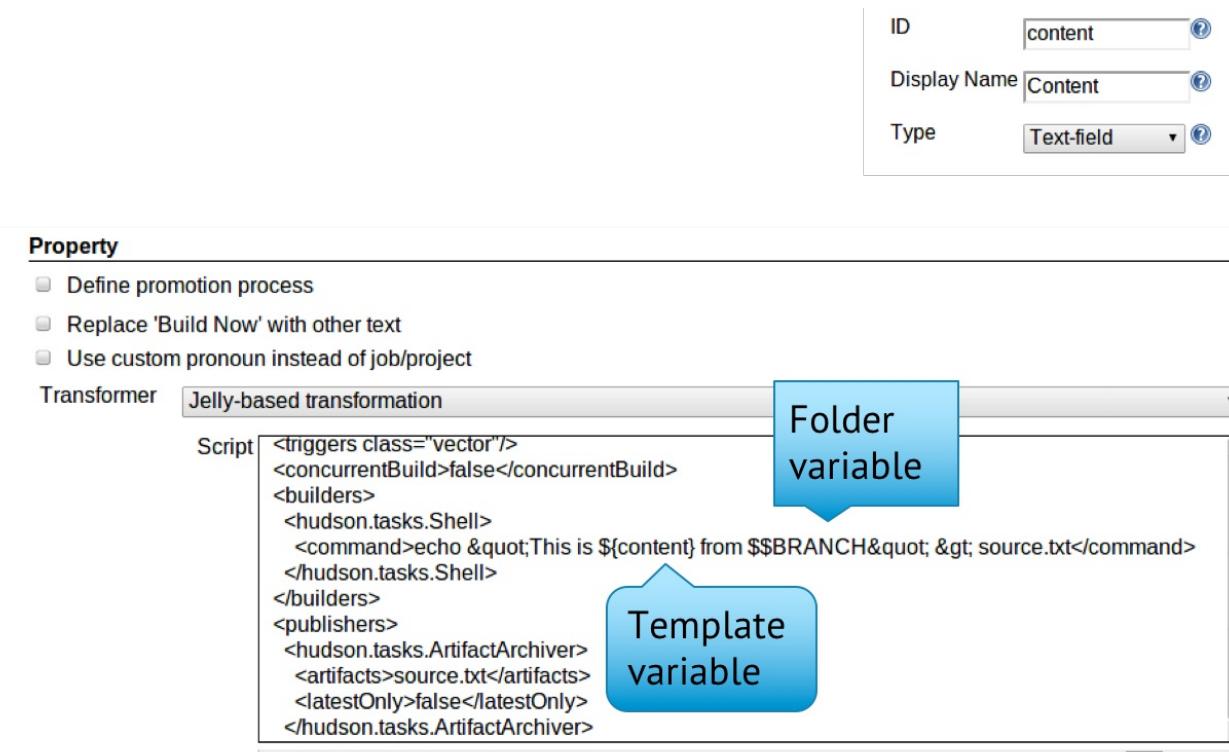
- Folder template example
- Scenario:
 - Team uses one branch for one sprint
 - Each branch requires build & test jobs
- How we model this:
 - Let's map one branch to one folder
 - Create a folder template so that build & test jobs are there from get-go

WHERE DO WE START?

- How to create them ?
- The same as before... by example!
 - Manually create a folder
 - Manually create two jobs in them
 - Take their config.xml and create templates

WHERE DO WE START?

- Setup: job template build
 - Define “content” attribute



The screenshot shows the configuration of a Jenkins job template. At the top, there's a panel for defining a 'Folder variable' with fields for ID ('content'), Display Name ('Content'), and Type ('Text-field'). Below this, the 'Property' section contains three unchecked checkboxes: 'Define promotion process', 'Replace 'Build Now' with other text', and 'Use custom pronoun instead of job/project'. Under 'Transformer', it's set to 'Jelly-based transformation'. The 'Script' field contains the following Jenkinsfile code:

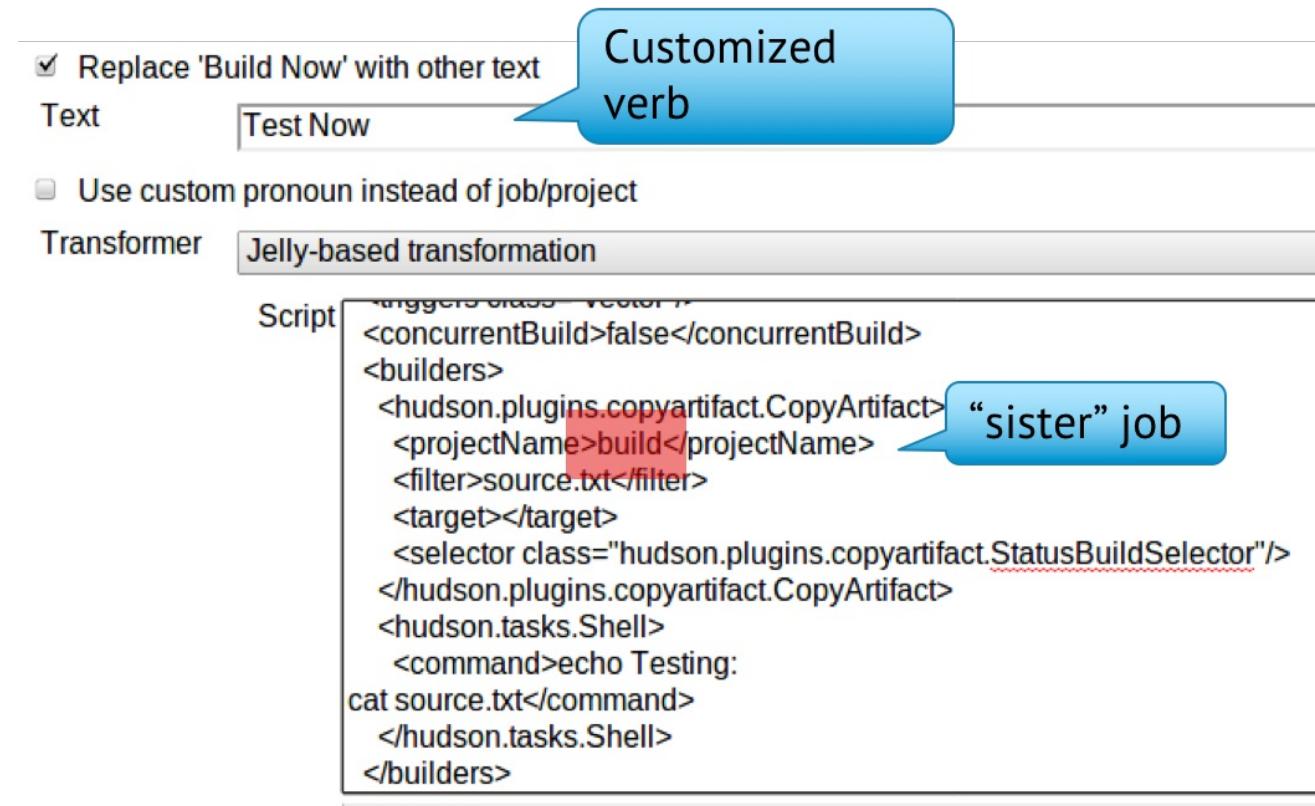
```
<triggers class="vector"/>
<concurrentBuild>false</concurrentBuild>
<builders>
  <hudson.tasks.Shell>
    <command>echo &quot;This is ${content} from $$BRANCH&quot; &gt; source.txt</command>
  </hudson.tasks.Shell>
</builders>
<publishers>
  <hudson.tasks.ArtifactArchiver>
    <artifacts>source.txt</artifacts>
    <latestOnly>false</latestOnly>
  </hudson.tasks.ArtifactArchiver>

```

Two blue callout boxes point to specific parts of the configuration: one labeled 'Folder variable' points to the 'content' ID in the top panel, and another labeled 'Template variable' points to the \${content} placeholder in the 'Script' field.

WHERE DO WE START?

- Setup: job template test



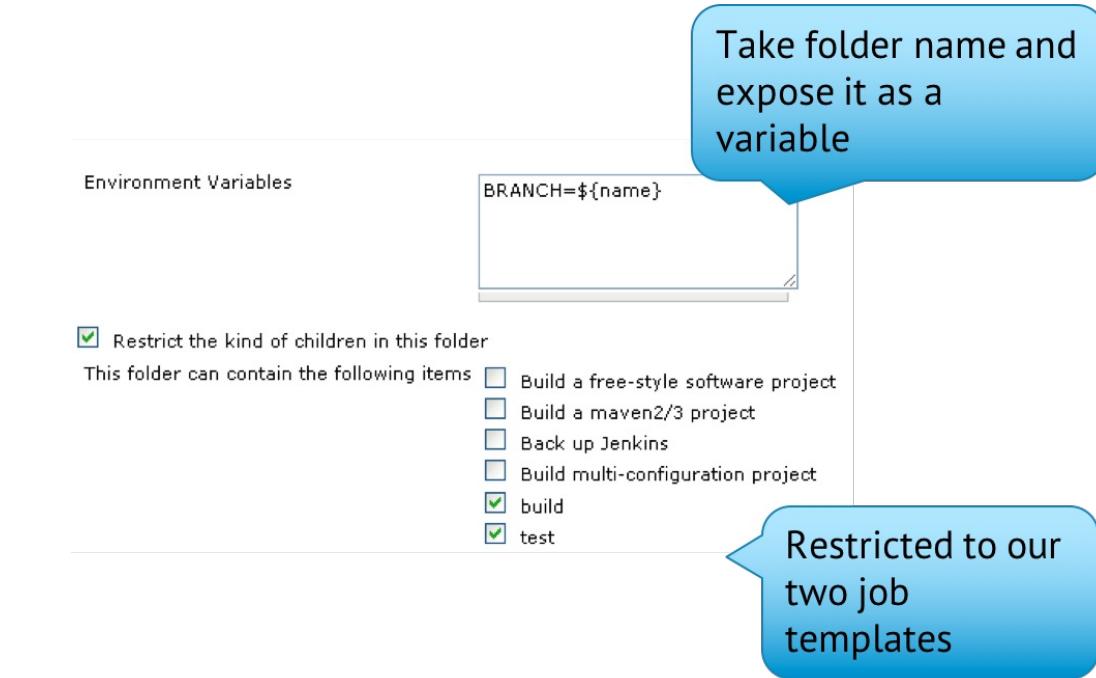
The screenshot shows the 'Text' and 'Script' sections of a Jenkins job template configuration. The 'Text' section contains a checked checkbox for 'Replace 'Build Now' with other text' and a text input field containing 'Test Now'. A blue callout bubble labeled 'Customized verb' points to the 'Text' input field. The 'Script' section contains a Jelly-based transformation script:

```
<concurrentBuild>false</concurrentBuild>
<builders>
<hudson.plugins.copyartifact.CopyArtifact>
<projectName>build</projectName>
<filter>source.txt</filter>
<target></target>
<selector class="hudson.plugins.copyartifact.StatusBuildSelector"/>
</hudson.plugins.copyartifact.CopyArtifact>
<hudson.tasks.Shell>
<command>echo Testing:
cat source.txt</command>
</hudson.tasks.Shell>
</builders>
```

A blue callout bubble labeled '“sister” job' points to the word 'build' in the script.

WHERE DO WE START?

- Create a folder that becomes a template
 - Create a folder by any name
 - Configure it the way we like



- Copy config.xml into clipboard

WHERE DO WE START?

- Folder template branch
 - Define the branch name attribute:

ID	<input type="text" value="name"/> ?
Display Name	<input type="text" value="Branch name"/> ?
Type	<input type="text" value="Text-field"/> ?
Inline Help	Specify the subversion branch name for this sprint. ?



WHERE DO WE START?

- Folder template branch:

Property	
<input checked="" type="checkbox"/> Use custom pronoun instead of job/project	
Text	Branch
Transformer	Jelly-based transformation
Script	<pre><?xml version='1.0' encoding='UTF-8'?> <com.cloudbees.hudson.plugins.folder.Folder> <actions/> <description></description> <properties> <com.cloudbees.hudson.plugins.folder.properties.EnvVarsFolderProperty> <properties>BRANCH=\${name}</properties> </com.cloudbees.hudson.plugins.folder.properties.EnvVarsFolderProperty> <com.cloudbees.hudson.plugins.folder.properties.SubItemFilterProperty> <allowedTypes> <string>test</string> <string>build</string> </allowedTypes> </com.cloudbees.hudson.plugins.folder.properties.SubItemFilterProperty> <com.cloudbees.hudson.plugins.folder.properties.FolderProxyGroupContainer/> <com.cloudbees.jenkins.plugins.foldersplus.SecurityGrantsFolderProperty> <securityGrants/> </com.cloudbees.jenkins.plugins.foldersplus.SecurityGrantsFolderProperty> </properties> <icon class="com.cloudbees.hudson.plugins.folder.icons.StockFolderIcon"/></pre>

A blue callout bubble points to the 'Branch' value in the 'Text' field, containing the text 'Customized display'.

WHERE DO WE START?

- Folder template branch
 - Create build & test jobs when a folder is created:

Initial Activities

Name of the new item	<input type="text" value="build"/>
Type	<input type="text" value="build"/>

Name of the new item	<input type="text" value="test"/>
Type	<input type="text" value="test"/>



WHERE DO WE START?

- Going one more level!
 - Scenario:
 - Many teams work on different products in your org
 - Let's define "product":
 - As a collection of branches

WHERE DO WE START?

- Create a folder that becomes a template
 - Create a folder by any name
 - Configure it the way we like
 - Restrict children to “branch”
 - Custom folder icon



WHERE DO WE START?

- Configure it the way we like (cont'd)
 - Column configuration:

Columns

Name	Column Title	Nested Job Name	Column To Apply
Pull information from nested job	Last Build	build	Last Success
Pull information from nested job	Last Test	test	Last Success
Pull information from nested job	Build	build	Build Button

Show information about job inside the folder





WHERE DO WE START?

- Take config.xml and paste that as XML:

more custom display

✓ Use custom pronoun instead of job/project

Text Product

Transformer Jelly-based transformation

Script

```
<?xml version='1.0' encoding='UTF-8'?>
<com.cloudbees.hudson.plugins.folder.Folder>
<actions/>
<description></description>
<properties>
<com.cloudbees.hudson.plugins.folder.properties.EnvVarsFolderProperty>
<properties></properties>
</com.cloudbees.hudson.plugins.folder.properties.EnvVarsFolderProperty>
<com.cloudbees.hudson.plugins.folder.properties.SubItemFilterProperty>
<allowedTypes>
<string>branch</string>
</allowedTypes>
<com.cloudbees.hudson.plugins.folder.properties.SubItemFilterProperty>
<com.cloudbees.hudson.plugins.folder.properties.FolderProxyGroupContainer/>
<com.cloudbees.jenkins.plugins.foldersplus.SecurityGrantsFolderProperty>
<securityGrants/>
<com.cloudbees.jenkins.plugins.foldersplus.SecurityGrantsFolderProperty>
</properties>
<icon class="com.cloudbees.hudson.plugins.folder.icons.BuiltinFolderIcon">
<baseName>package</baseName>
</icon>
<views>
<hudson.model.ListView>
<owner class="com.cloudbees.hudson.plugins.folder.Folder" reference=".//.."/>
<name>All</name>
<filterExecutors>false</filterExecutors>
<filterQueue>false</filterQueue>
<properties class="hudson.model.View$PropertyList"/>
<jobNames class="tree-set">
<comparator class="hudson.util.CaseInsensitiveComparator"/>
</jobNames>
<jobFilters/>
```



WHERE DO WE START?

- Making a “product” and “branches”
 - initially empty...

Custom icon

My Program

All +

Up

Status

Configure

Move

Controlled Slaves

New branch

Delete Product

People

Build History

Edit View

Project Relationship

Check File Fingerprint

only “branches” offered as additions

TOC

WHERE DO WE START?

- Adding a branch and customizing its build job:



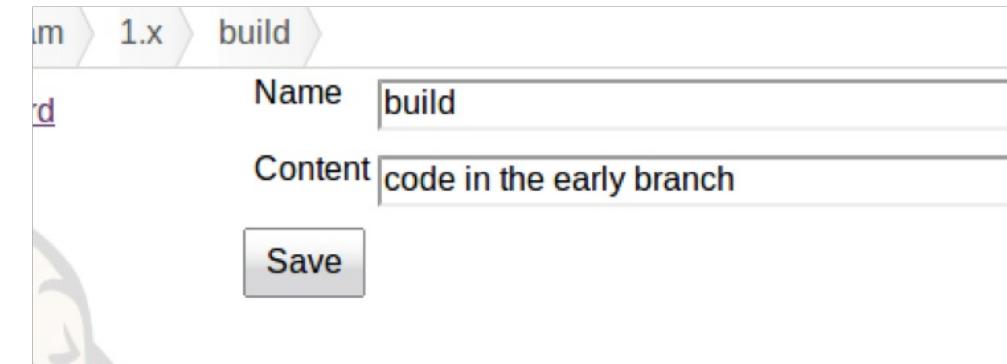
The screenshot shows the Jenkins web interface. On the left, there is a sidebar with the following links:

- Up
- Status
- Configure
- Move
- Controlled Slaves
- New Item
- Delete Branch
- People

On the right, there is a list of branches under the heading "1.x".

Name	Content	Last Success
build		N/A
test		N/A

A "New Item" button is visible at the top right. Below this, a detailed view of the "build" branch is shown in a modal window.



The modal window displays the configuration for the "build" branch:

- Name: build
- Content: code in the early branch
- Save button





WHERE DO WE START?

- Running build and test jobs inside a branch:

1.x

Name	Content	Last Success
build	code in the early branch	11 sec (#2)
test		6.9 sec (#2)

[add description](#)

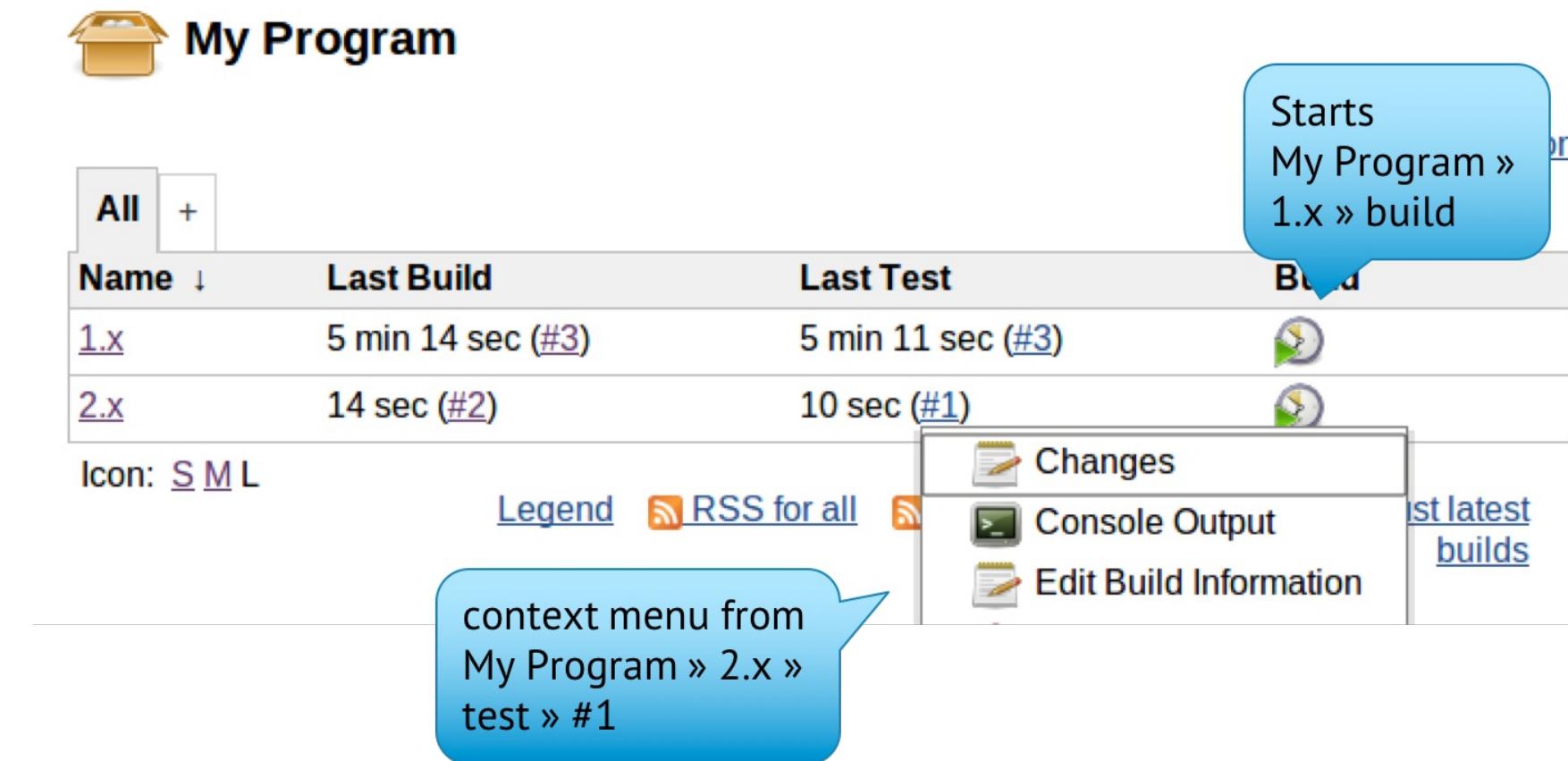
Console Output

```
Started by user anonymous
Building in workspace /home/jglick/src/jdbc/work/jobs/My
Program/jobs/1.x/jobs/build/workspace
[workspace] $ /bin/sh -xe /tmp/hudson6714744848822535328.sh
+ echo This is code in the early branch from 1.x
Archiving artifacts
Finished: SUCCESS
```



WHERE DO WE START?

- Product folder now shows overview of branches:



My Program

Name	Last Build	Last Test	Build
1.x	5 min 14 sec (#3)	5 min 11 sec (#3)	
2.x	14 sec (#2)	10 sec (#1)	

Icon: [S](#) [M](#) [L](#)

[Legend](#) [RSS for all](#) [List latest builds](#)

Starts My Program » 1.x » build

context menu from My Program » 2.x » test » #1

LAB EXERCISE:

Lab 13: Templates

CLOUDBEES SUPPORT PLUGIN

Accelerate diagnosis with CloudBees Support

WHAT IS CLOUDBEES SUPPORT

- CloudBees customers have access to a whole set of services including **Professional Support** by Jenkins experts who are themselves developers and Jenkins maintainers.
- The **Support** plugin provides the ability to generate a bundle of all the commonly requested information used by CloudBees when resolving support issues.
 - Those bundles will be used within the **CloudBees Support Portal**

WHERE DO WE START?

- Simply go to the plugin's release notes, pick the latest version, and click the link in the Download section, such as to <http://jenkins-updates.cloudbees.com/download/plugins/cloudbees-support/latest/cloudbees-support.hpi>). You can install the downloaded plugin using Manage Jenkins » Manage Plugins » Advanced » Upload Plugin
- To generate a bundle, simply go to the CloudBees Support link on the Jenkins instance.



WHERE DO WE START?(CONT.)

CloudBees Jenkins Operations Center

Jenkins > Support

New Item

People

Build History

Manage Jenkins

Alerts

Support

Analytics Dashboard Creator

Credentials

Pooled Virtual Machines

Cluster Operations

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

CloudBees Support

In order to assist CloudBees Technical Support in providing you with an efficient response to your support requests please generate a bundle of the commonly requested support information and attach this bundle to your support ticket.

It is best to include all of the following information in the support bundle, but if you are unable to provide some of the information due to corporate policy, you can either deselect the information to exclude prior to generating the bundle or edit the generated bundle to remove the specific information that you must exclude. Each bundle is a simple ZIP file containing mostly plain text files and you can examine and/or modify the bundle prior to sharing the bundle if you have any concerns about the information contained within.

Log Recorders

About browser

About Jenkins

About user (basic authentication details only)

Administrative monitors

Build queue

Dump slave export tables (could reveal some memory leaks)

Environment variables

File descriptors (Unix only)

JVM process system metrics (Linux only)

Load Statistics

All loggers currently enabled.

Metrics

Networking Interface

Root CAs

System configuration (Linux only)

System properties

Update Center

Slow Request Records

Deadlock Records

Thread dumps

Generate Bundle

LAB EXERCISE

Lab 14: CloudBees Support

INTRODUCTION TO CJOC

CloudBees Jenkins Operations Center



**CLOUDBEES
UNIVERSITY**

JENKINS @ SCALE



CLOUDBEES
UNIVERSITY

HOW JENKINS GROWS IN AN ORGANIZATION ?



Horizontally... virally



Vertically

VERTICAL SCALING ISSUES

- How long can I keep scaling Jenkins?
- Single point of failure:

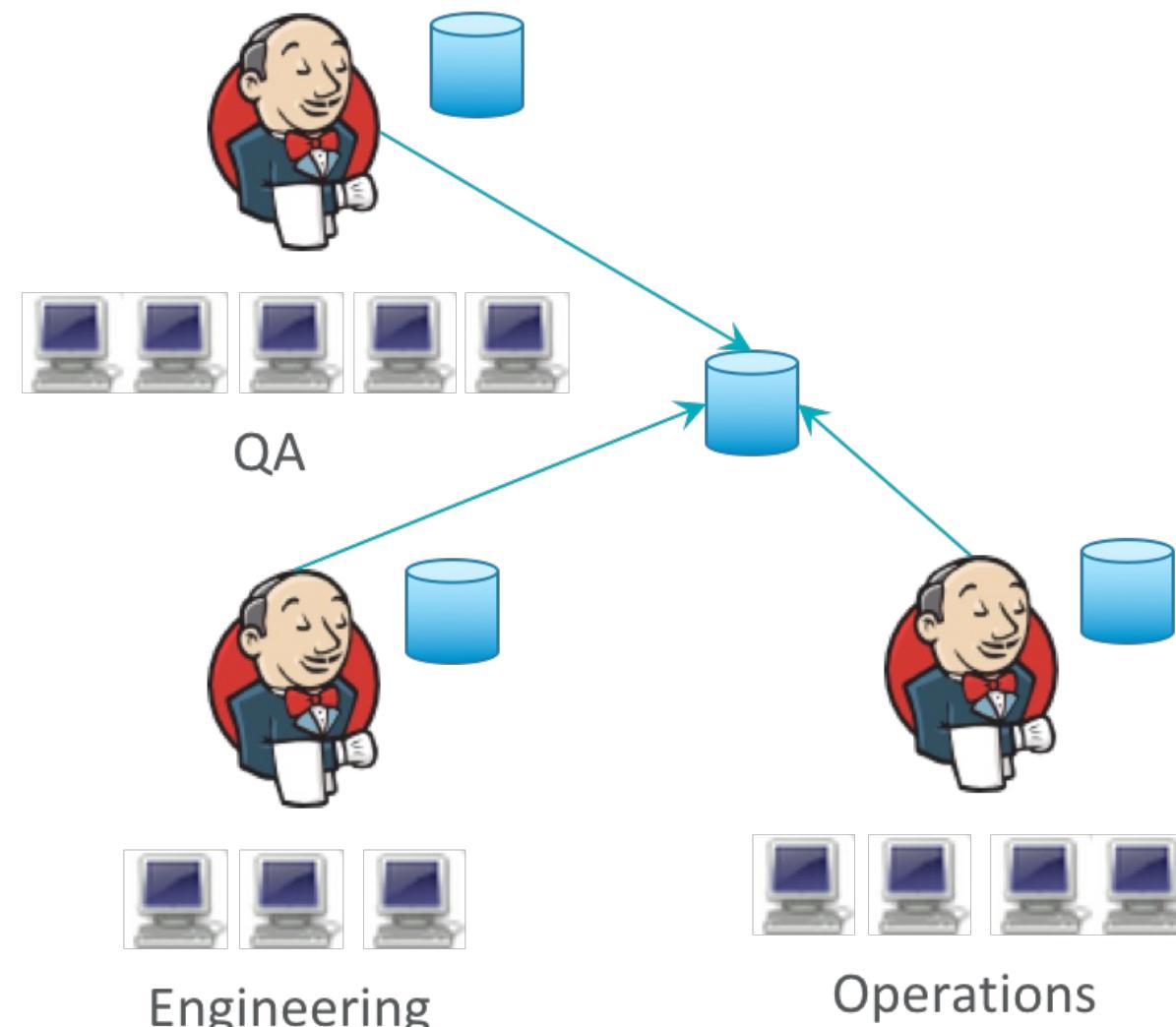




HORIZONTAL SCALING ISSUES

- I would like to...
 - Share security information: Security Realm and roles
 - Easily navigate between masters

<http://sca12-3530-qe.cloudbees.com/jenkins>



INTRODUCING CLOUDBEES JENKINS OPERATIONS CENTER

aka "CJOC"

WHAT IS CLOUDBEES JENKINS OPERATIONS CENTER?

- Operations console for Jenkins
 - Consolidated navigation
 - Shared agents
 - Centralized authentication and authorization
 - License management
 - Security enforcement
 - Analytics
 - Cluster Operations

WHAT IS CLOUDBEES JENKINS OPERATIONS CENTER?

- Without CJOC, scaling with Jenkins can be painful.
 - Two ways to scale today:
 - Vertical
 - Horizontal
- CJOC provides alternative scaling model to avoid the pitfalls of a single point of failure and master sprawl.

WHAT IS CLOUDBEES JENKINS OPERATIONS CENTER?

- CJOC treats agents and Jenkins **masters** like Jenkins jobs.
- Maintenance tasks, like plugin updates, can also be run as Jenkins jobs using a CJOC-only job type called “Cluster Operations”.
- CJOC tracks agent and master usages in a report format.

CLOUDBEES JENKINS OPERATIONS CENTER

OVERVIEW

Scale

- Easily configured client masters
- Shared build resources
- Shared clouds

Secure

- Role-based Access Control (RBAC)
- Integration with LDAP and ActiveDirectory
- Single-Sign On
- Setting enforcement
- Plugin version enforcement

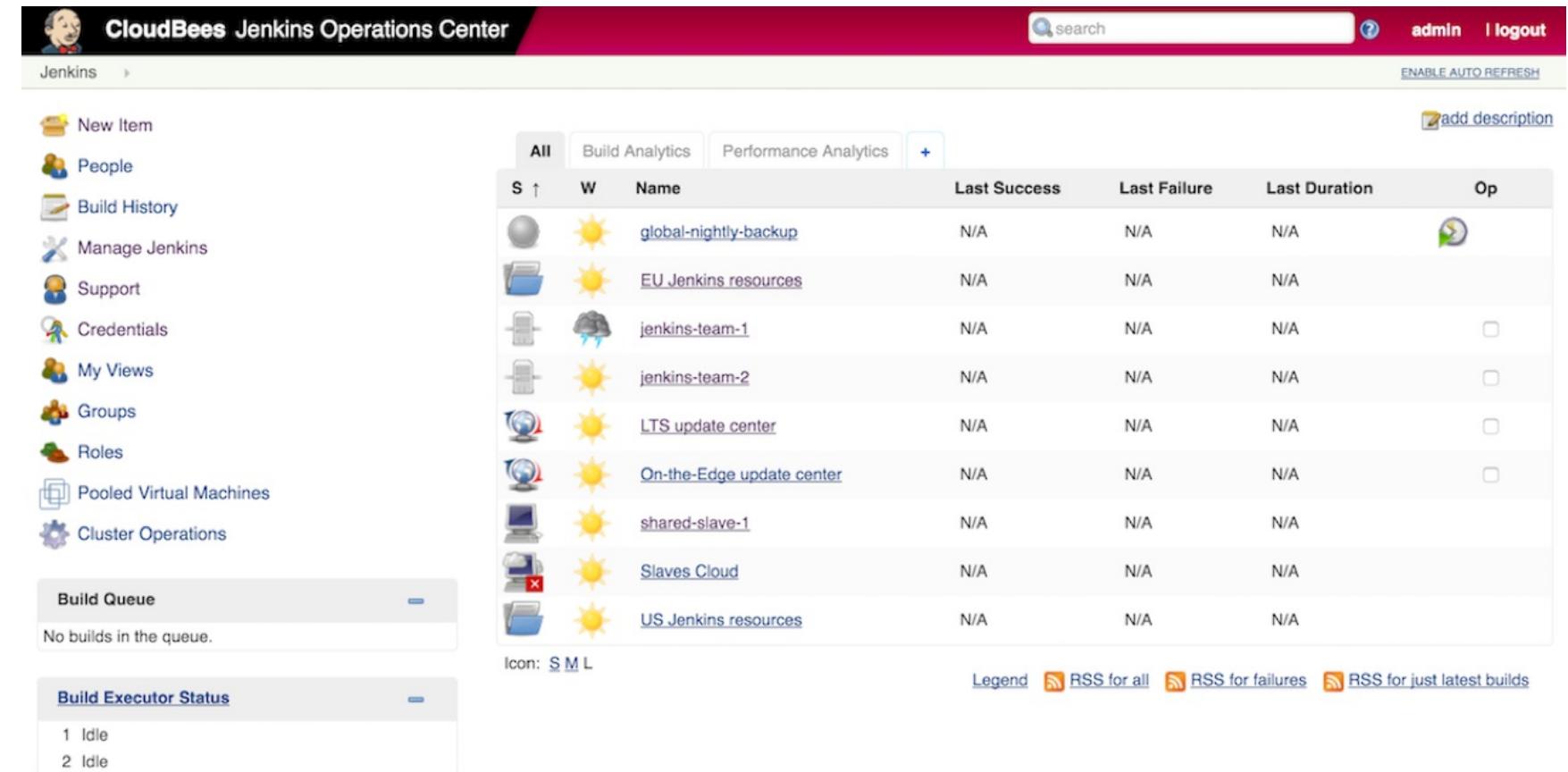
Manage

- License management
- CloudBees Jenkins Analytics
- Cluster Operations
- Test instance management



CONCEPTS

- Operations Center server:



The screenshot shows the CloudBees Jenkins Operations Center interface. On the left, there's a sidebar with various Jenkins management links like New Item, People, Build History, Manage Jenkins, Support, Credentials, My Views, Groups, Roles, Pooled Virtual Machines, and Cluster Operations. Below these are sections for Build Queue (empty) and Build Executor Status (1 Idle, 2 Idle). The main area displays a table of Jenkins jobs with columns for S (Status), W (Weather icon), Name, Last Success, Last Failure, Last Duration, and Op (Operations icon). The jobs listed are: global-nightly-backup, EU Jenkins resources, jenkins-team-1, jenkins-team-2, LTS update center, On-the-Edge update center, shared-slave-1, Slaves Cloud, and US Jenkins resources. All jobs show N/A for Last Success, Last Failure, and Last Duration. The Operations Center also includes a search bar, user authentication (admin), and auto-refresh options.

S	W	Name	Last Success	Last Failure	Last Duration	Op
Grey circle	Sunny	global-nightly-backup	N/A	N/A	N/A	
Folder icon	Sunny	EU Jenkins resources	N/A	N/A	N/A	
File icon	Cloud with rain	jenkins-team-1	N/A	N/A	N/A	
File icon	Sunny	jenkins-team-2	N/A	N/A	N/A	
Globe icon	Sunny	LTS update center	N/A	N/A	N/A	
Globe icon	Sunny	On-the-Edge update center	N/A	N/A	N/A	
Computer icon	Sunny	shared-slave-1	N/A	N/A	N/A	
Computer icon	Sunny	Slaves Cloud	N/A	N/A	N/A	
Folder icon	Sunny	US Jenkins resources	N/A	N/A	N/A	

TOC

CONCEPTS

- Client master:

Operations Center Connector

Enable

Connection Details

----- BEGIN CONNECTION DETAILS -----
H4sIAAAAAAAA2KSQ7DIAwA/+JzDTaKi81vnAYI6YKqlJyq/r1c5jAzX1gPbx0KTM
aelG4osmSc
nGd0y3lAqsisktngAvsyXsJ7bY+9fbBXf2Ea/jyel2y9v0uMbCnwVYNZYKKipBTh9wfF
XyeybAAA

Operations Center cluster: <http://192.168.99.100:8080/>



CONCEPTS

- Shared agent:

The screenshot shows the Jenkins Operations Center interface for managing slaves. On the left, there's a sidebar with links: Back to Dashboard, Status, Delete Slave, Configure, Move/Copy/Promote, Groups, and Roles. The main area is titled 'shared-slave-1'. It contains the following fields:

- Name:** shared-slave-1
- Description:** A built-in, example shared slave based on Alpine Linux that includes, Java (jdk8) and Git 2.8
- Details:**
 - # of executors: 2
 - Remote FS root: /home/jenkins
 - Labels: shared alpine jdk8 git
 - Usage: Utilize this node as much as possible
 - Launch method: Launch slave agents on Unix machines via SSH (Non-blocking I/O)
- Host:** shared-slave-1
- Credentials:** jenkins/******** (built-in slave credentials (cjp-trial))
- Availability:** Keep this slave on-line as much as possible
- Properties:**
 - Inject Node Properties (unchecked)

At the bottom, there are 'Save' and 'Apply' buttons, and a checked checkbox for 'Take on-line after saving'.





CONCEPTS

- Shared cloud:

The screenshot shows the 'CloudBees Jenkins Operations Center' interface. On the left, there's a sidebar with links: Back to Dashboard, Status, Delete Cloud, Configure, Move/Copy/Promote, Groups, and Roles. The main area is titled 'Slaves Cloud' and shows a configuration for a 'VCenter Cloud for US slaves'. The 'Cloud' section is set to 'Pooled VMWare Virtual Machines'. Other settings include: Machine Center, Machine Pool, Usage (set to 'Utilize this node as much as possible'), Labels, Remote FS root, # of Executors (set to 1), Only one build per VM (unchecked), Shutdown Timeout (set to 5), and Connector (set to 'Launch slave agents on Unix machines via SSH'). A 'Credentials' dropdown shows 'jenkins/******** (built-in slave credentials (cjp-trial))' with an 'Add' button. There are also 'Advanced...' buttons at the bottom and right.

TOC

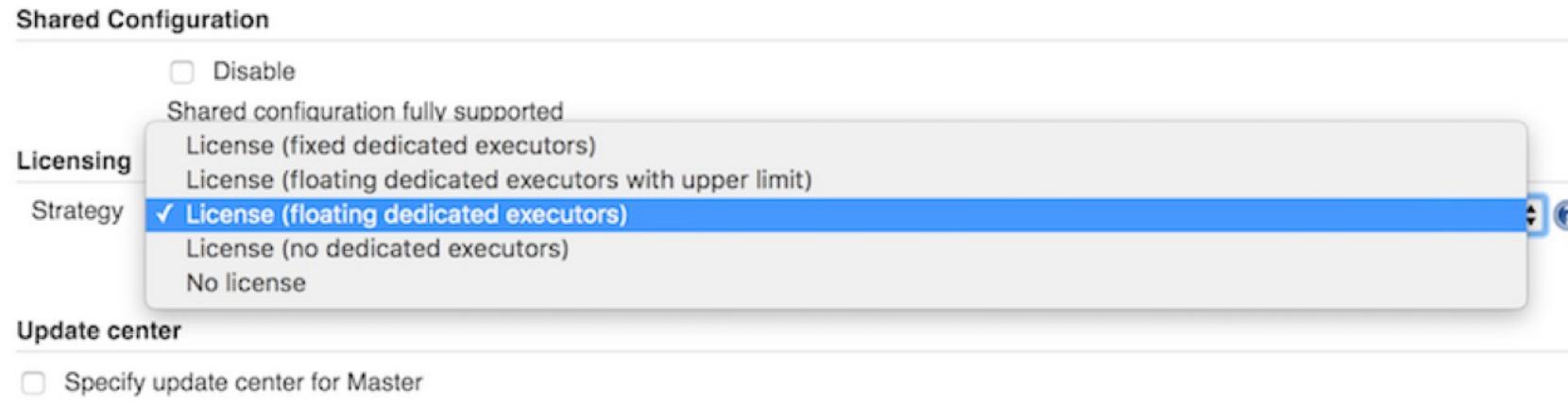
CONCEPTS

- Folders:

All	Build Analytics	Performance Analytics	+			
S ↑	W	Name		Last Success	Last Failure	Last Duration
		global-nightly-backup		N/A	N/A	N/A
		EU Jenkins resources		N/A	N/A	N/A

CONCEPTS

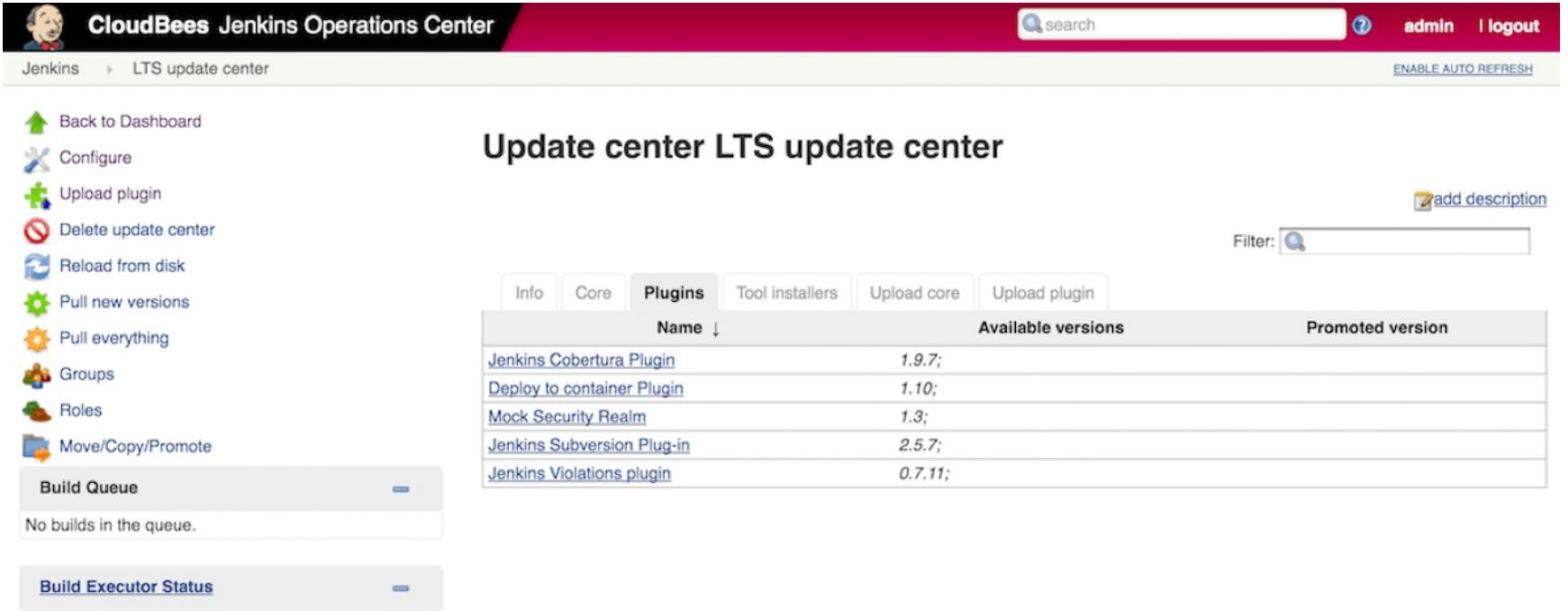
- Sub-licensing:



The screenshot shows the 'Shared Configuration' section of the Jenkins configuration interface. Under the 'Licensing' heading, a dropdown menu is open, displaying several options: 'Disable', 'Shared configuration fully supported', 'License (fixed dedicated executors)', 'License (floating dedicated executors with upper limit)', '**License (floating dedicated executors)**' (which is selected and highlighted in blue), 'License (no dedicated executors)', and 'No license'. The 'Strategy' and 'Update center' sections are also visible below.

CONCEPTS

- Update Center:



The screenshot shows the 'Update center LTS update center' page in the CloudBees Jenkins Operations Center. The left sidebar contains links for Back to Dashboard, Configure, Upload plugin, Delete update center, Reload from disk, Pull new versions, Pull everything, Groups, Roles, and Move/Copy/Promote. Below these are sections for Build Queue (No builds in the queue) and Build Executor Status. The main content area has a search bar, a 'Plugins' tab (selected), and a table showing available plugin versions. The table has columns for Name, Available versions, and Promoted version.

Name ↓	Available versions	Promoted version
Jenkins Cobertura Plugin	1.9.7;	
Deploy to container Plugin	1.10;	
Mock Security Realm	1.3;	
Jenkins Subversion Plug-in	2.5.7;	
Jenkins Violations plugin	0.7.11;	

BENEFIT: MAINTAINABILITY

- While some organizations seek to ease the pain of maintenance by offloading Jenkins to a managed service like CloudBees' DEV@cloud or PSE, CJOC offers an alternative solution.
- CJOC's Cluster Operations can **automate maintenance tasks**, like plugin and core updates, rollbacks, and master back ups.
 - Cluster Operations can be scheduled, triggered, etc. like regular Jenkins jobs.

BENEFIT: AUDIT TRAIL

- **Track:**
 - who performed an update or plugin install,
 - which master or update center was affected,
 - and what the outcome was.
- CJOC's Analytics allows **metrics** on master/agent **usage** to be tracked and stored in an Elasticsearch instance.

BENEFIT: SCALABILITY

- CJOC allows **masters to be onboarded quickly**, with new masters receiving pre-configured security authorization and authentication settings as well as the ability to “lease” agents from a shared pool.
- Shared agents allow **build resources** to be used more **efficiently** between Jenkins masters.

BENEFIT: STABILITY

- **Test masters** for production instances can be set up and managed using CJOC.
 - Test instances are the best way to test for plugin/core conflicts **before** running updates on production masters
- CJOC and client masters are compatible with CloudBees' **High Availability** plugin.

WHERE DO WE START?

- CJOC will need to be installed on a server
 - Run as .war, RPM, etc
 - At least 1 GB for CJOC's JVM

