cloudbees UNIVERSITY

CLOUDBEES JENKINS PLATFORM: CERTIFICATION TRAINING

4 - MODERN JENKINS - PIPELINES

© 2018 CloudBees, Inc. All Rights Reserved

TOC

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINES CONCEPTS

© 2018 CloudBees, Inc. All Rights Reserved

WHY PIPELINE ?

- Freestyle jobs:
  - Provide only sequential steps
  - Does not "survive" upon restarts (even planned)
- Chaining job with upstream/downstream:
  - Is only GUI-based
  - Does not provide centralized configuration
- Implementing a complex CD pipeline is hard with those tools

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE GOALS

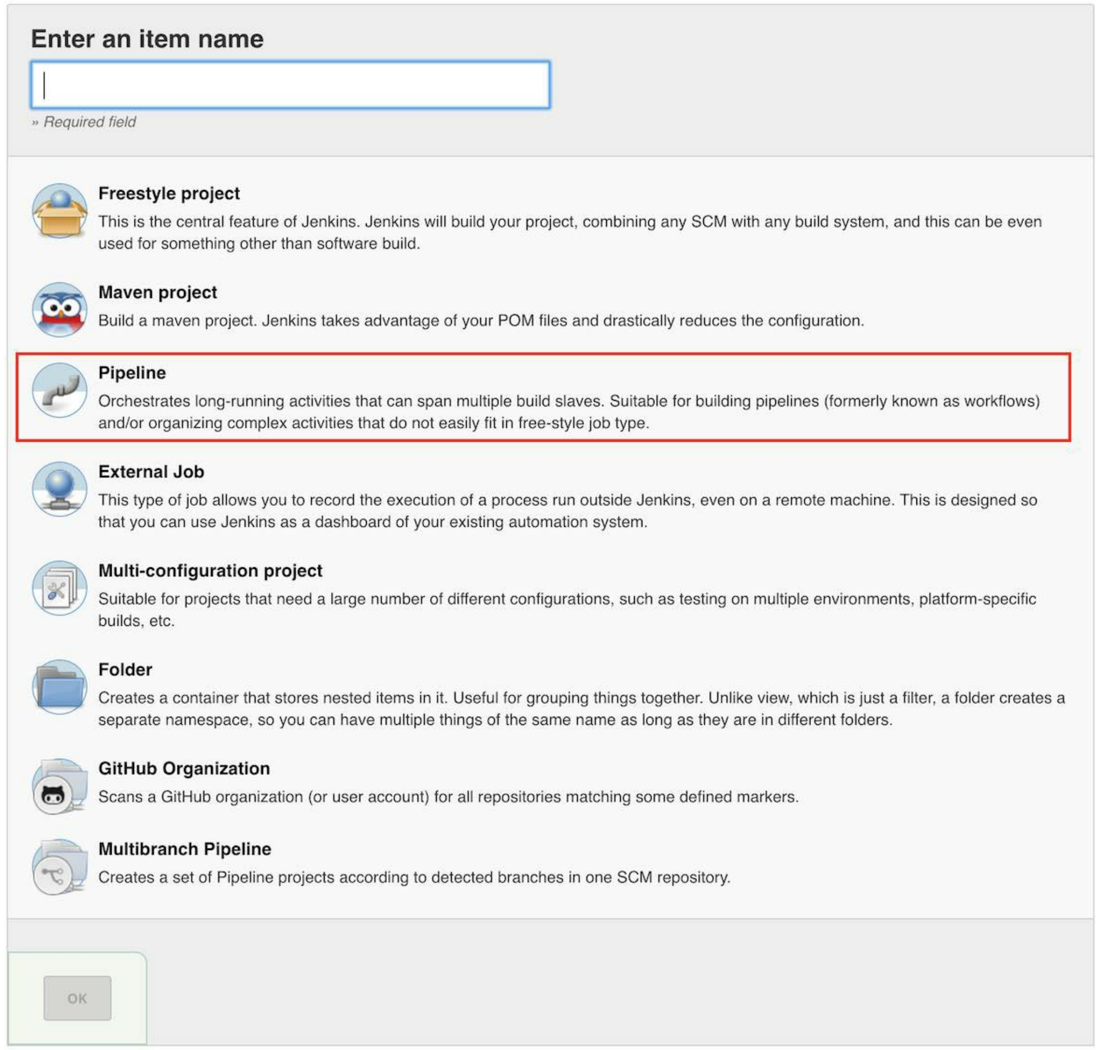
- The pipeline functionality is:
  - Durable: survives Jenkins master restarts
  - Pausable: can stop and wait for human input or approval
  - Versatile: supports complex real-world CD requirements (fork, join, loop, parallelize)
  - Extensible: supports custom extensions to its "DSL" (Domain Scripting Language)

WHAT IS A PIPELINE ?

- A Pipeline is:
  - A new type of Job
  - Script based: use Apache Groovy (JVM-based scripting language)
  - Uses the Pipeline DSL (Domain Scripting Language)
    - programmatically manipulate Jenkins Objects
  - An implementation of your CD Pipeline
    - Orchestrating your Steps

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE JOB TYPE



Enter an item name

» Required field

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.

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.

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

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.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM repository.

© 2018 CloudBees, Inc. All Rights Reserved

JENKINS VOCABULARY 1/2

- Master:
  - Where Jenkins is installed and run
  - Serves requests and handles build tasks
- Agent: (formerly "slave")
  - Computer set up to offload available projects from the master.
  - Has a number and scope of operations to perform.

© 2018 CloudBees, Inc. All Rights Reserved

JENKINS VOCABULARY 2/2

- Node:
  - Computer part of the Jenkins cluster
  - Can be master or agent
  - Generic name that we won’t use below !
- Executor:
  - Computational resource for running builds
  - Performs Operations
  - Can run on any master or agent Node
  - Can be parallelized on a specific Node

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE VOCABULARY: STEP

- Step:
  - A single task (also known as "Build Step")
  - Part of a sequence.
  - It tells Jenkins what to do.

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE VOCABULARY: NODE

- Node:
  - Type of step , NOT a Jenkins "Node"
  - Contains other steps
  - Schedule the contained steps across Jenkins agents and executors
  - Orchestrate ephemeral workspaces on remote agents (create and delete)

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE VOCABULARY: STAGE

- Stage:
  - Type of step
  - Logically distinct part of the task executions
  - Can have parameters for locking, labeling and ordering
  - Can have one or more build steps within it
- Best practice is to use it (visualization)

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE RUN ANATOMY

- Pipeline Groovy scripts are parsed and run on the master
  - node blocks allocate executors and workspaces from master
- agents still handle operations, by running executors, on scopes.
- Pipeline's operations run on master using flyweight executors
  - Uncounted executor: temporary slot
  - Use very little computing power.
  - Represent an idle Groovy script waiting for a step to complete

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE-AS-CODE

- Main Groovy script of a Pipeline is a **Jenkinsfile**
- The Jenkinsfile is stored on an SCM
  - Still, GUI configuration possible (bad practice)
  - Benefits of SCM: apply versioning, testing and merging against your CD Pipeline definition

© 2018 CloudBees, Inc. All Rights Reserved

WHAT DID WE LEARN ?

- Pipeline is a new kind of job
- It aims to be durable, pausable, versatile and extensible
- Based on centralized Groovy script that manipulates a DSL
  - Implements the Pipeline-as-Code concept
- Pipeline scripts are run on the master
  - flyweight executors
- It schedules their steps on executors
- with keyword node

© 2018 CloudBees, Inc. All Rights Reserved

GOING FURTHER

Some recommended readings on this subject:

- https://jenkins.io/projects/blueocean/
- https://jenkins.io/doc/book/pipeline/syntax/
- https://jenkins.io/doc/pipeline/steps/
- https://go.cloudbees.com/docs/cloudbees-documentation/use/automating-projects/
- https://go.cloudbees.com/docs/cloudbees-documentation/use/reference/pipeline/

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINES

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE REQUIREMENTS

- At least: Pipeline plugin: (formerly known as Workflow plugin)
- Works with a suite of related plugins that enhance functionality
- Related plugins add pipeline syntax or visualizations.
- Recommended: Start with
  - Pipeline workflow-aggregator: installs core plugins and dependencies
  - Pipeline Stage View
  - Multibranch Pipeline
  - Docker Pipeline

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE HELLO WORLD

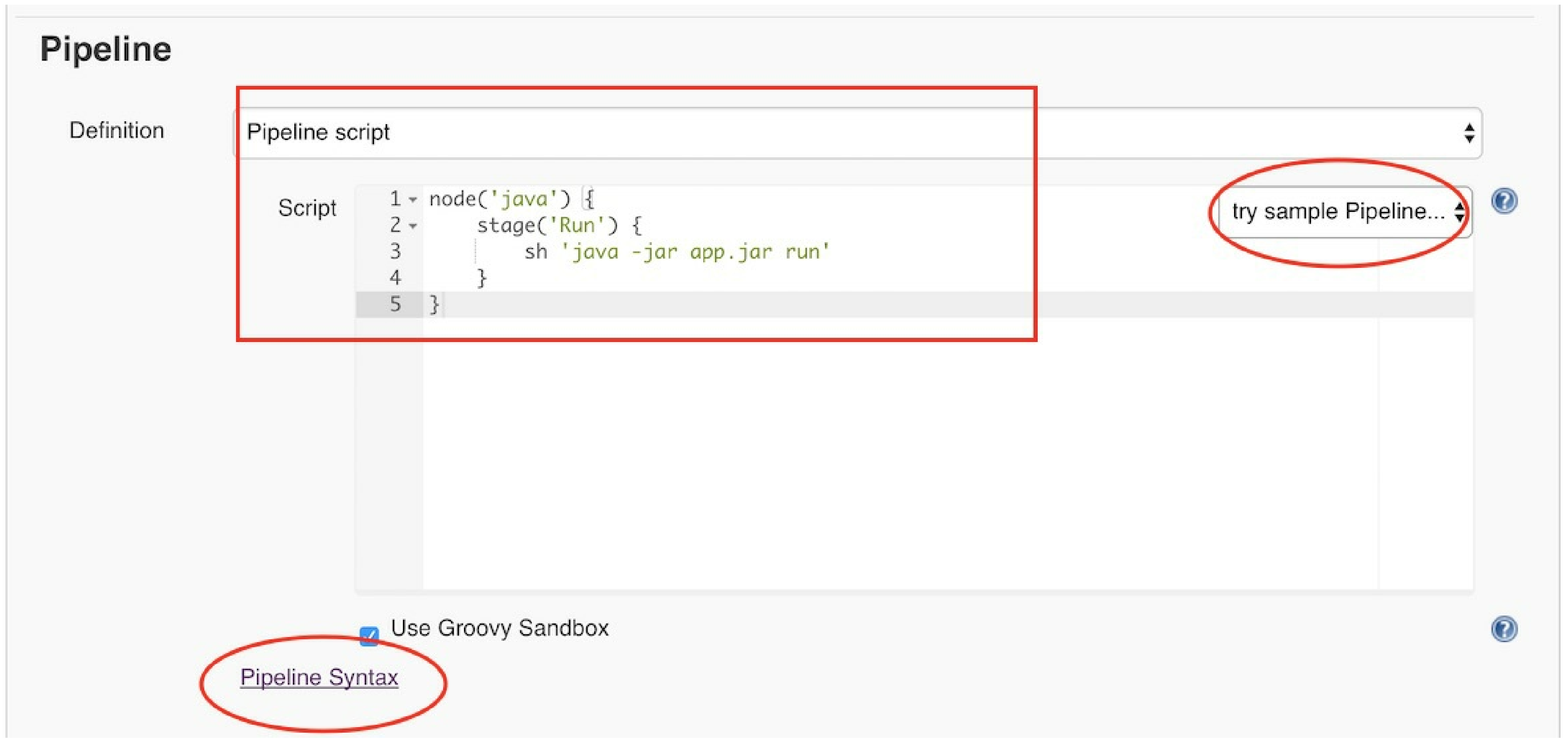
- Simple example:
  - Allocate an executor
  - Print the string

```
node {
  echo 'Hello from Pipeline'
}
```

© 2018 CloudBees, Inc. All Rights Reserved

WHERE TO START ? WRITING IN GUI

- SCM is recommended
  - However, Jenkins GUI helps a lot
- Create a new Pipeline Job configuration page:
  - Colored text editor + samples



Pipeline

Definition: Pipeline script

Script:

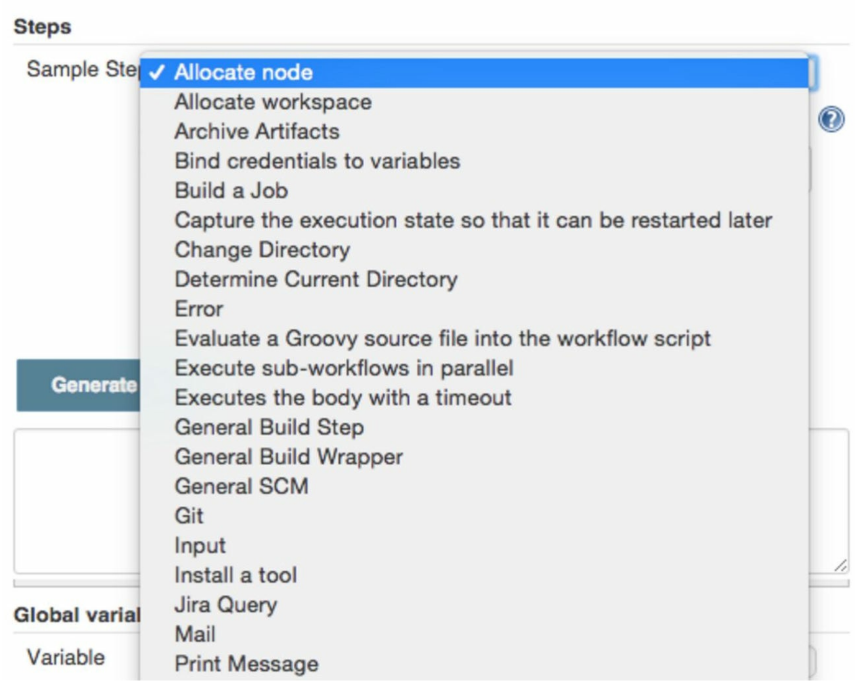
```
node('java') {
    stage('Run') {
        sh 'java -jar app.jar run'
    }
}
```

© 2018 CloudBees, Inc. All Rights Reserved

cloudbees UNIVERSITY

WHERE TO START ? SNIPPET GENERATOR

- Not familiar with an individual step ?
  1. Use the Snippet Generator to create syntax examples
  2. Copy and past generated snippets to your scripts
- Dedicated page of Jenkins GUI
- Dynamically populated with available steps
  - Depends on installed plugins


Steps

Sample Step

✓ Allocate node
Allocate workspace
Archive Artifacts
Bind credentials to variables
Build a Job
Capture the execution state so that it can be restarted later
Change Directory
Determine Current Directory
Error
Evaluate a Groovy source file into the workflow script
Execute sub-workflows in parallel
Executes the body with a timeout
General Build Step
General Build Wrapper
General SCM
Git
Input
Install a tool
Jira Query
Mail
Print Message

Generate

Global varial

Variable

PIPELINE: SIMPLE EXAMPLE

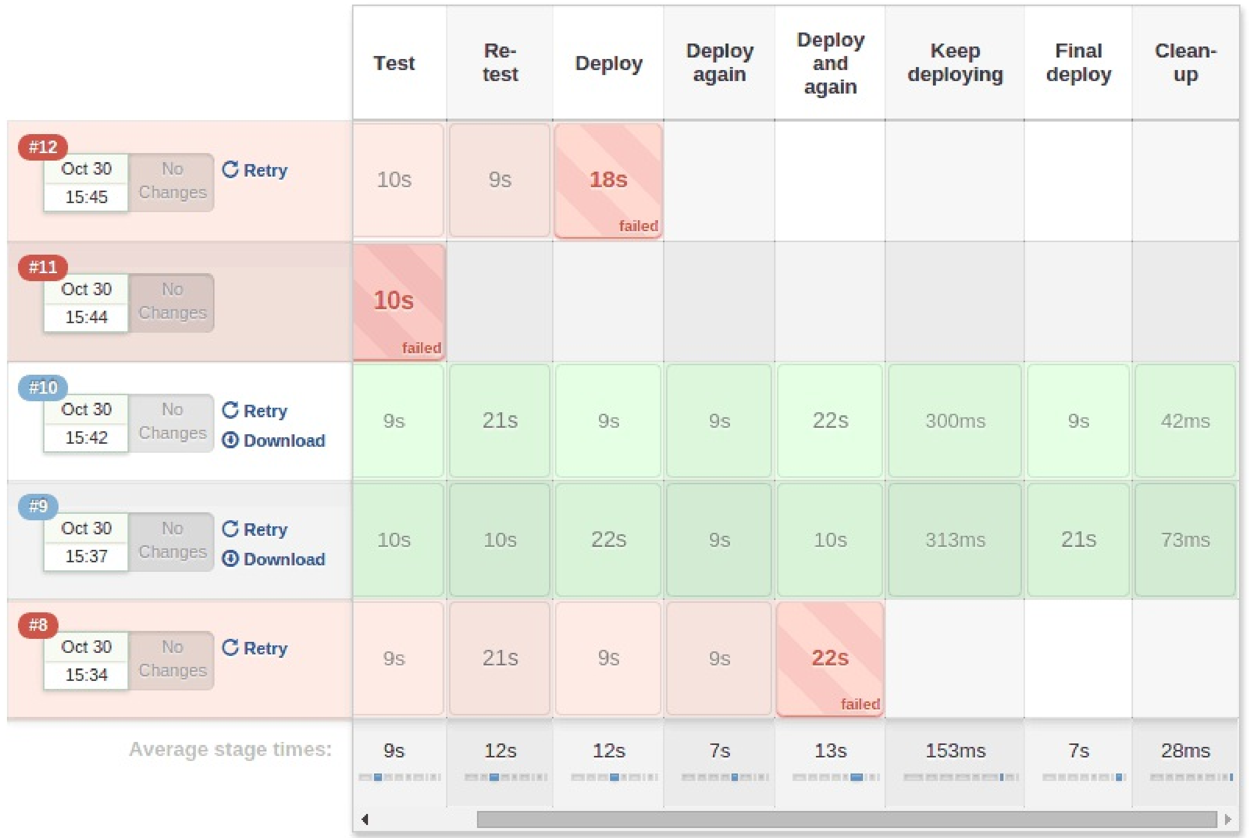
- Simple example:
  - Allocates a scoped executor (on an agent with label `cpp`)
  - Clones a git repository
  - Runs a shell command to build it

```
node('cpp') {
  git url: 'https://github.com/joe_user/cpp-app.git'
  sh "make build"
}
```

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE STAGE VIEW

- Plugin open sourced by CloudBees in 2016
  - Installed by the workflow-aggregator
- Provide Pipeline visualization relying on stages
- Not a Jenkins view: this is GUI on the Job Page
- Shows a matrix with build history and stages as dimensions



	Test	Re-test	Deploy	Deploy again	Deploy and again	Keep deploying	Final deploy	Clean-up
#12 Oct 30 15:45 No Changes Retry	10s	9s	18s failed					
#11 Oct 30 15:44 No Changes	10s failed							
#10 Oct 30 15:42 No Changes Retry Download	9s	21s	9s	9s	22s	300ms	9s	42ms
#9 Oct 30 15:37 No Changes Retry Download	10s	10s	22s	9s	10s	313ms	21s	73ms
#8 Oct 30 15:34 No Changes Retry	9s	21s	9s	9s	22s failed			
Average stage times:	9s	12s	12s	7s	13s	153ms	7s	28ms

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE STAGE VIEW: BENEFITS

- Get (another) feedback: a visual one
- Easier failure tracking:
  - Isolate failure to a specific stage
  - Output log is viewable "by stage"
- Visualization is related to a single Job and pipeline
- Displays pipeline data:
  - Date, time, changes (with changes links) per build
  - Execution time per build and per stage
  - Status and Output logs per stage

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE: APACHE GROOVY SYNTAX AND TOOLS

- Example with:
  - Using the MAVEN3 Jenkins Tool Installation
  - Groovy variables syntax (**def mvnHome =**)
  - Run a Maven build in the cloned repository

```
node() {
  git url: 'https://github.com/joe_user/simple-maven-project-with-te
  def mvnHome = tool 'MAVEN3'
  sh "${mvnHome}/bin/mvn -B verify"
  // Windows syntax instead of sh:
  // bat "${mvnHome}\\bin\\mvn -B verify"
}
```

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE: SCOPES

- Groovy syntax uses "Scope" syntax (~ anonymous functions).
- Example with:
  - Run a scripts inside a sub folder

```
node() {
  git url: 'https://github.com/joe_user/dockerized-app.git'
  dir('scripts') {
    sh 'bash ./admin-script.sh'
  }
}
```

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE: ENVIRONMENT

- Pipeline DSL provides the **env** variable
  - Its properties are environment variables on the current node.
  - Can override some environment variables; change seen in subsequent steps
- Example with:
  - Maven tools management for fungible agent

```
node {
  git url: 'https://github.com/jglick/simple-maven-project-with-tests.git'
  withEnv(["PATH+MAVEN=${tool 'MAVEN3'}/bin","M2_HOME=${tool 'MAVEN3'}"]) {
    sh 'mvn -B verify'
  }
 }
```

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE: STAGES

- Represent an abstract "stage"
- Expect a "label", a string provided as argument
- Stage is a scoped block

```
node {
  stage('Checkout SCM') {
    git url: 'https://github.com/jglick/simple-maven-project-with-tes
  }
  stage('Build') {
    sh 'mvn -B verify'
  }
 }
```

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE: MANUAL APPROVAL

- Blocking execution flow before a human validates
- You can "tune" message, buttons...
- Will fail the build if "NO" button pressed
- Good practices:
  - Run it outside a node so it does not block an executor
  - Use timeout to avoid waiting for an infinite amount of time
  - Use Groovy Control Structures (try/catch/finally)

```
stage('Waiting for Approval') {
  input message: "Does http://localhost:8888/staging/ look good?"
}
```

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE: PARALLELS STEPS

- Steps can be run in parallel:
  - Long running step to optimize Pipeline
  - Different independent use cases
- Each "parallelized branch" is a stage

```
parallel 'integration-tests':{
   node('mvn-3.3'){
     sh 'mvn verify'
    }
}, 'functional-tests':{
   node('selenium'){
     sh 'bash /run-selenium-tests.sh'
    }
}
```

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE: EXECUTION CONTROL

- Control execution of your pipeline:

```
// Try up N times
retry(10) { . . . }

// Pause the flow:
sleep time: 10, unit: 'MINUTES'

// Wait for event
waitUntil { . . . }

// Timeout an operation
timeout(time: 100, unit: 'SECONDS') { . . . }
```

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE: FILE SYSTEM

- Read file:

```
readFile file: 'some/file', encoding: 'UT
```

- Write file:

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

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE: SUPPORTED PLUGINS

- If a plugin is Pipeline-compliant, it provides more keywords
- Example with junit test report publisher:

```
stage('Build') {
  sh 'mvn clean install -fn'
  junit './target/**/*.xml'
}
```

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE: NON SUPPORTED PLUGINS

- Invocation syntax for plugins
  - Until they may evolve to offer native pipeline support:

```
step( // DSL keyword to invoke arbitrary step
  [ // groovy map
      $class: 'build_step_class_name', // Same as in job’s config.xml on c
      constructor_argument: 'value',
      constructor_argument_2: 'value'
  ]
)
// Exemple with the Archive Artifact step:
step([$class: 'ArtifactArchiver', artifacts: 'target/**/*.jar'])
```

© 2018 CloudBees, Inc. All Rights Reserved

WHAT DID WE LEARN ?

- Use pipeline by installing the workflow-aggregator plugin
  - Additional plugins may be required for your usage
- Start writing using the Snippet Generator and the GUI
- Move to SCM based **Jenkinsfile** when you are ready
- Use the visualization and stages for better feedback
- We browsed some common DSL keywords examples

© 2018 CloudBees, Inc. All Rights Reserved

GOING FURTHER

Some recommended readings on this subject:

- https://jenkins.io/doc/pipeline/
- https://jenkins.io/projects/blueocean/
- https://go.cloudbees.com/docs/cloudbees-documentation/cje-user-guide/index.html#workflow
- https://go.cloudbees.com/docs/cloudbees-documentation/cje-user-guide/index.html#workflow-sect-stage-view
- https://dzone.com/refcardz/continuous-delivery-with-jenkins-workflow
- https://jenkins.io/doc/pipeline/tour/hello-world/

© 2018 CloudBees, Inc. All Rights Reserved

ADVANCED PIPELINES

© 2018 CloudBees, Inc. All Rights Reserved

MULTI-BRANCH PIPELINES

WHY MULTI-BRANCH PIPELINES ?

- A Jenkins Pipeline Job has the following challenges:
  - It only maps a single branch of the SCM
  - No automatic discovery
  - No separation of concerns

© 2018 CloudBees, Inc. All Rights Reserved

WHAT IS A MULTI-BRANCH PIPELINE ?

- A kind of Jenkins Job
  - Basically: it is a folder
- Configured to point to an SCM
- Contains Pipeline Jobs
  - One Pipeline per SCM branch with a **Jenkinsfile**
  - Supports Pull Requests as well
  - Automatically created/deleted
- Will be the default and recommended way with future Jenkins versions

© 2018 CloudBees, Inc. All Rights Reserved

HOW TO START WITH MULTI-BRANCH PIPELINES ?

- Create the Multi-Branch
- Configure your SCM source
- (Optionnal) Configure a webhook from SCM
- Push a **Jenkinsfile** on any branch
  - Merge branch: jobs automatically managed
- Everything automated: no more Jenkins Admin nightmare

© 2018 CloudBees, Inc. All Rights Reserved

MULTI-BRANCH PIPELINES CONFIGURATIONS

- Customizable retention policy
  - "Orphaned Item Strategy" configuration section
- Secured: Run Pipeline in the Groovy Sandbox
  - Code considered "unsecure" needs admin validation
  - Avoid unknown code running without protection
- Provide additional variables for more complex pipelines
  - `BRANCH_NAME`
  - `CHANGE_ID`

© 2018 CloudBees, Inc. All Rights Reserved

ORGANIZATION SCANNING

- Using a hosted SCM with Jenkins (Github, Bitbucket, etc.) ?
  - Corresponding plugins must be installed
- Admin configures the organization for this kind of Jobs
  - One Credential (API token generally) needed
  - Maps to an "organization folder" as top level
- Each project maps to a Multi-Branch pipeline
  - Inside the "organization folder"
  - More automation
  - Automate webhooks creation

© 2018 CloudBees, Inc. All Rights Reserved

PIPELINE SHARED LIBRARIES

© 2018 CloudBees, Inc. All Rights Reserved

WHY PIPELINE SHARED LIBRARIES ?

- D.R.Y.: do not repeat yourself !
- Scale your Jenkins Pipeline usage
  - More projects
  - More teams
- Leverage maintenance overhead
  - Write once, propagate everywhere
  - Pipeline as code everywhere
- Use tooling to avoid silos
  - Collaborate instead of enforcing

© 2018 CloudBees, Inc. All Rights Reserved

WHAT IS A PIPELINE SHARED LIBRARY ?

- A set of SCM containing reusable Pipeline code
- Configured 1 time inside Jenkins
- Cloned at build time
- Loaded and used as code libraries on Jenkins Pipelines

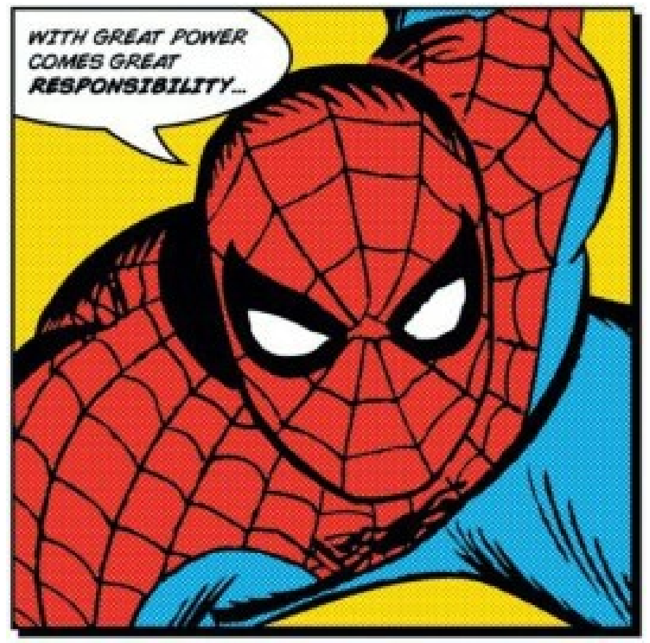
© 2018 CloudBees, Inc. All Rights Reserved

HOW TO USE PIPELINE SHARED LIBRARIES ?

- Configure 1 (or more) where you need it:
  - Trusted code: by admins at Jenkins level
  - Not trusted code: by developers at Multi-Branch/Folder levels
- Define policies:
  - Default branch / tag / changeset
  - Can developers override default version ?
- Load it from your pipeline:
  - By Annotation
  - By DSL keyword
  - Implicitly

© 2018 CloudBees, Inc. All Rights Reserved

NOTE ON SHARED LIBRARIES



- Extremely powerful, highly added value
- Learning curve: 1st step is not easy
- It is code so must be tested
  - Adds some overhead: time investment
- Many uses
  - Take time to read documentation

© 2018 CloudBees, Inc. All Rights Reserved

WHAT DID WE LEARN ?

- Default Pipeline usage should be Multi-Branched
  - Maps to a repository
  - Manage branches/Pull Requests for you
- Organization Folder
  - One more level of automation
  - Only for Github / Bitbucket (for now…)
- Shared Libraries:
  - Share and reuse your pipeline code
  - Help admin manage the code sprawl
  - Ease collaboration

© 2018 CloudBees, Inc. All Rights Reserved

GOING FURTHER

Some recommended readings on this subject:

- https://jenkins.io/doc/book/blueocean/getting-started/
- https://jenkins.io/doc/book/pipeline/multibranch/
- https://jenkins.io/blog/2015/12/03/pipeline-as-code-with-multibranch-workflows-in-jenkins/
- https://jenkins.io/doc/book/pipeline/shared-libraries/
- https://jenkins.io/blog/2017/02/15/declarative-notifications/
- https://plugins.jenkins.io/github-organization-folder
- https://plugins.jenkins.io/bitbucket

© 2018 CloudBees, Inc. All Rights Reserved

LAB EXERCISE

4 - Modern Jenkins: Pipelines

© 2018 CloudBees, Inc. All Rights Reserved