2.19.4

## Repo github

https://github.com/LearningCitizen/Jenkins-ece.git

## Plugins suggéré

* ant
* antisamy-markup-formatter
* build-timeout
* cloudbees-folder
* credentials-binding
* email-ext
* git
* github-organization-folder
* gradle
* ldap
* matrix-auth
* mailer
* pam-auth
* pipeline-stage-view
* ssh-slaves
* subversion
* timestamper
* workflow-aggregator
* ws-cleanup

## Key CI/CD/Jenkins concepts

### Continuous Delivery/Continuous Integration Concepts

**Continuous integration** : software development practice where members of a team integrate their work frequently. Then there is an automated build (including test) in an integration machine (in dev environment). (git push + automatic build). A build is self-testing if a test fail makes it fail.

**Continuous delivery** : continuous integration + the possibility of easily deploy any version to any environment (especially in a production-like env). All the steps of the delivery process should be automatated (deployment pipeline).

**Continuous deployment** : continuous delivery + every change goes throught the pipeline and automatically gets put into production.

**Differences between CI and CD** : Continuous delivery requires continuous integration. Continuous integration + the possibility of easily deploy any version to any environment (especially in a production-like env).

**Stages of CI and CD :   
CI :** integrate code with a VCS + automated the build (with self-testing code, unit test) + automated tests on the executable   
**CD :** CI + all the steps of the deployment are automated (roll back should be easy)

**Continuous delivery vs continuous deployment** : In continuous deployment every changes are automatically puts in production. Whereas in continuous delivery, you can do frequent deployment but can choose to not do it. Continous deployment requires continuous delivery.

### Jobs

**What are jobs (project) in Jenkins?** : The jobs are a particular steps in the build process or in the deployment process. It can be a way of compiling, testing, packaging and deploying the project, but also measuring code quality/code coverage, generate documentation.

Job is a deprecated term. A project is a user-configured description of work which Jenkins should perfom, such as building a piece of software.

**Types of jobs**

* Freestyle software project: general-purpose build jobs. Jenkins will build your project, combining any scm with any build system. It can be used for something other than software build.
* Pipeline : Jenkins Pipeline is a combination of plugins that support the integration and implementation of continuous delivery pipelines using Jenkins. running activities
* Multiconfiguration project : job with many different configurations. Can be useful to test the application in many different environments. It runs the job with all the different. combinations of axis parameterers.
* Gitub Organization : Scans a Github organization for all matching some defined markers.

Multibranch pipeline : Create sets of pipelines according to detected branches in one scm repository.

* (Maven project : build a maven project taking advantage of POM files.)
* (External job : this type of job record the execution of a process run outside of Jenkins, even on a remote Jenkins.)

**Scope of jobs** : you can have jobs with same name in different folders.

### Builds

**What are builds in Jenkins ?**

Result of a single execution of a Project/job.

**What are build steps, triggers, artifacts, and repositories ?**

A build step is a single task during a build.

A build trigger is a criteria for triggering a new pipeline run or Build (time, scm polling, etc..).

An artifact is an immutable file generated during a build or pipeline run which is **archived** onto the Jenkins master for later retrieval by users.

A git repository is a data structure (folder .git in a project) used by git to store the set of files as well as history of changes made to those.

An artifact repository is a source for artifacts needed for build and a target to deploy artifacts generated in the build process.

**Build tools configuration**

In manage Jenkins -> systems configuration -> set the location of the tools used in build as ant, jdk, maven, node js python, etc..

### Source

**What are source code management systems and how are they used ?**

It is a software wich allows user to keep track of the changes in a project and enable them to collaborate.

**Cloud-based SCMs**

Cloud-based or distributed scms are a type of scms in which all the developer’s computer contains the complete code base and the full history.

Ex: Github.

**Jenkins changelogs**

It is a functionality of Jenkins which shows changes tracked by the scm between each revision.

Lists commits since last build.

**Incremental updates v clean check out**

Incremental update -> faster

Clean check out -> guarantees no extra or changed local files

**Checking in code**

Means when code is uploaded in the main branch repository, to review the code before update the project version. At least daily with CI.

**Infrastructure-as-Code**

Storingin config files everything need to build our environment instead of interactive operations.

**Branch and Merge Strategies**

* Branch by release
* Branch by feature – by user story
* Branch by abstraction – one branch, but turn features on/off by release
* Merge regularly
* Minimize number of branches

### Testing

**Benefits of testing with Jenkins**

* Fast feedback
* Easier to find bugs

**Define unit test, smoke test, acceptance test, automated verification/functional tests**

### Unit test – test one class, often involves test doubles

### Integration/functional test – test components together.

### Smoke test – sanity check to reject a release. Simple test that looking for major errors.

### Acceptance test – user level test for feature. Similar to the integration test but focused on use case.

### Notifications

**Types of notifications in Jenkins**

* Email notification
* Hipchat
* Slack Notification
* Skype Notification
* Sms Notification
* Extreme Notification (Webhook -> requete http + parameter)

**Importance of notifications**

* Fixing a build is high priority so need to know it is broken
* Communicating the status to all parties

### Distributed Builds

**What are distributed builds?**

* Running builds on a different machine than master

**Functions of masters and agents**

* Master – basic Jenkins install, co-ordination + GUI + API enpoints
* Slaves – just for running jobs

### Plugins

**What are plugins?**

Extension of Jenkins functionality that is not in Jenkins core.

**What is the plugin manager?**

UI for install, upload update or delete plugins. You can also define a proxy.

### Jenkins Rest API

**How to interact with it**

* Xml api
* Json api
* Python api
* (Python Api wrappers, Ruby Api wrappers, Java Api wrappers)

**Why use it?**

* Information for Programmatic acces
* Trigger new build
* Create/copy jobs

### Security

**Authentication versus authorization**

* Authentication – identify a user
* Authorization – what user can do

**Matrix security**

* Maps roles to permissions
* Major categories: overall, slave, job, run, view and SCM

**Definition of auditing, credentials, and other key security concepts**

* Auditing – logging user operations and changes
* Credentials – username/password or the like for access

### Fingerprints

**What are fingerprints?**

The fingerprint of a file is its MD5 checksum (a 32 character hexadecimal number computed). It’s for any types of files. It is useful to track dependencies (artifacts).

**How do fingerprints work?**

* The first time you run a job with a post build step to generate a fingerprint, a new left navigation option shows up to check a file’s fingerprint.
* You can upload a file you have to see if any file Jenkins knows the fingerprint of matches.
* Jenkins maintains a database md5sum and for each md5sum, Jenkins records which build of which project used. Jenkins stores only the md5sum and their usages.

### Artifacts

**How to use artifacts in Jenkins**

* Download
* Upload in Nexus/Artifactory,
* deploy,
* use in an other etc…

**Storing artifacts**

* can archive artifact
* discard old build by age or number

### Using 3rd party tools

**How to use 3rd party tools**

* Setup in Manage System the location on disk or download from there
* Ex: JDK, Maven, Git, Gradle, Docker, Ant
* Can install automatically or from file system

### Installation Wizard

**What is the Jenkins Installation Wizard?**

It’s a setup wizard which helps to finish “one-off” steps.

**How to use the Wizard?**

You need to donwnload, install and run Jenkins.

**Which configurations are covered by the Installation Wizard?**

* Unlock Jenkins
* Customize Jenkins with plugins
* Define the first administrator user

## Jenkins Usage

### Jobs

**Organizing jobs in Jenkins**

* Jobs are organized in folders (new Item -> Folder)

**Parameterized jobs**

* Parameters allow you to prompt user for imputs that will be passed into a build.
* Check “This build is parameterized” and enter parameters/default values
* Run directly with “Build with Parameters” or call from upstream job with “trigger parameterized build” post build action and passing parameters

**Usage of Freestyle/Pipeline/Matrix jobs**

* Freestyle – most flexible job
* Pipeline – enter code in DSL. There is a snippet generator which generates the Groovy for common operations and lists the available environment variables. Allow to create complex build that is not easily fit in free-style job.
* Matrix (multi-config) – Specify a configuration matrix with one or more dimensions. Runs all combinations when build.
  + Axis: slave, label (for slave) or user defined (string)
  + Combination filter: if don’t want cross product of all axis to run
  + Can execution “touchstone” builds first to specify which job(s) should run first and if this should skip the others

* New item -> select the job

### Builds

**Setting up build steps and triggers**

* Job -> configure -> Build -> add build step
* Windows command batch
* Shell
* Ant
* Gradle
* Maven
* Run with timeout
* Set build status to “pending” on github

**Configuring build tools**

* In Manage Jenkins > Global Tool Configuration
* Install automatically (define at least 1 installer: command) or define the path location

**Running scripts as part of build steps**

* Job -> Configure -> Execute Shel/Windows command/ant/graddle/maven target
* Can run OS script or Groovy script
* Groovy scripts can run as system or user level. System has access to Jenkins object model

### Source Code Management

**Polling source code management**

* Set schedule using cron format
  + minute hour dayOfMonth month dayOfWeek
  + For dayofWeek, 0 and 7 is Sunday
* If no schedule is set the build will be triggered by a post-commit hook

**Creating hooks**

* Hook script in repository triggers job
* Ex: Github plugin provides hook

**Including version control tags and version information**

* We can create a tag for everybuild : Job -> Configure -> Source Code Management -> Git -> add -> create a tag for everybuild + job-> Configure -> PostBuild Actions -> Git Publisher -> Tags
* Version Number plugin lets you include info in build name

## Testing

**Testing for code coverage**

* Plugin Cobertura, Jacoco, Clover,
* In build, configure to generate the code coverage report(.cml, or .exec : Jacoco)
* Enable the Code coverage Publisher
* Specify the directory where the publisher can find the reports
* Configure the metrics target
* In Cobertura, can set thresholds for weather icons:
  + Sunny - % higher than threshold
  + Stormy - % lower than threshold
  + Unstable - % lower than threshold
* In Jacoco, can set thresholds for sunny and stormy

**Test reports in Jenkins**

* You can configure your build to produces some test reports
* Some plugin can use test reports to provide graphical visualization of test like Publish JUnit or TestNG reports
* In JUnit, can set amplification factor - 1.0 means 10% failure rate scores 90% health. .1 means 10% failure rate scores 99% health.

**Displaying test results**

* With Junit publisher
* Configure the location of the xml files test reports
* set amplification factor - 1.0 means 10% failure rate scores 90% health. .1 means 10% failure rate scores 99% health

**Integrating with test automation tools**

* TestComplete est un outil qui permet de gérer automatiquement les tests. TestComplete plugin est un plugin qui permet d’exécuter des projets TestComplete.
* You can run acceptance test in pipeline.

**Breaking builds**

* JUnit allows choosing whether to fail builds on test failures - default is “unstable” not failure (if reports are empty)
* testComplete : actions on errors/ actions on warnings
* Dans pipeline, tu peux faire des tests et renvoyer une erreur , catcher l’erreur et fal le build :

catchError(message: 'the test has failed', stageResult: 'FAILURE') {

if (env.TEST\_STATUS == 'failed') {

error 'the test has failed'

}

### Notifications

**Setup and usage**

Setup in post build action section

**Email notifications, instant messaging**

* Email
  + Same recipient for each one (except can add committers since passed)
* Email ext
  + lets you customize the message and tailor the recipients per trigger
  + can send on failing, still failing, unstable, still unstable, successful, etc
* Jabber and IRC for instant messaging
* Since build radiators are full screen, the only way to edit is to add /configure to the URL

**Alarming on notifications**

Extreme notifications can have a video or audio cue in the real world

### Distributed Builds

**Setting up and running builds in parallel**

* Builds run on different executors
* Multi-configuration jobs run the pieces in parallel

**Setting up and using SSH agents, JNLP agents, cloud agents**

Can launch local slaves with SSH (blocking or non-blocking IO), Java Web Start, command line on master or Windows service

◦ Monitoring nodes

• Plugins

◦ Setting up and using Plugin Manager

◦ Finding and configuring required plugins

## Building Continuous Delivery (CD) Pipelines

## CD-as-code best practices