Continuous Deployment with Pipeline as Code – Quick start Series – #1 Jenkins

Monday April 30th, 2018 (https://www.gologic.ca/en/english-continuous-deployment-with-pipeline-as-code-quick-start-series-jenkins/)

Goals Series

This article series aims to explore the different tools for doing pipeline-as-code and deployment.

The goal for each article remains the same: Checkout GIT source code, compile a JAVA project with Maven, run the tests then deploy the application on AWS BeanStalk.

Those steps will be written as code in a Pipeline and executed with a CI/CD tools.

Each article is divided into several parts:

- Installation and startup of a CI/CD tool
- Configuration CI/CD tool (if needed)
- Code continuous deployment pipeline
- Check deployment
- A simple Conclusion

If you want to run pipeline, you will need:

- Docker runtime to execute pipeline steps

- an AWS BeanStalk environment with access key and secret to deploy application

Before starting, let's define two key concepts: Continuous Deployment and Pipeline-As-Code.

What does "Continuous Deployment" mean ?

Continuous Deployment is closely related to Continuous Integration and refers to the release into production of software that passes the automated tests.

"Essentially, it is the practice of releasing every good build to users", explains Jez Humble, author of Continuous Delivery.

By adopting both Continuous Integration and Continuous Deployment, you not only reduce risks and catch bugs quickly, but also move rapidly to working software.

With low-risk releases, you can quickly adapt to business requirements and user needs. This allows for greater collaboration between ops and delivery, fueling real change in your organization, and turning your release process into a business advantage.

What does "Pipeline as Code" mean ?

Teams are pushing for automation across their environments(testing), including their development infrastructure.

Pipelines as code is defining the deployment pipeline through code instead of configuring a running CI/CD tool.

Source code

GitHub Demo Reference is there: Continuous Deployment Demo (https://github.com/gologic-ben/continuous-deployment)

Jenkins

Goal

For this first article, Jenkins will be the guinea pig.

Jenkins is an open source automation server written in Java. Jenkins helps to automate the non-human part of software development process, with continuous integration and facilitating technical aspects of continuous delivery. It is a server-based system that runs in servlet containers such as Apache Tomcat.

Jobs can be triggered by various means, for example by commit in a version control system, by scheduling via a cron-like mechanism and by requesting a specific build URL. It can also be triggered after the other builds in the queue have completed. Jenkins functionality can be extended with plugins.

Install and Run Jenkins with Docker

Run Jenkins docker container

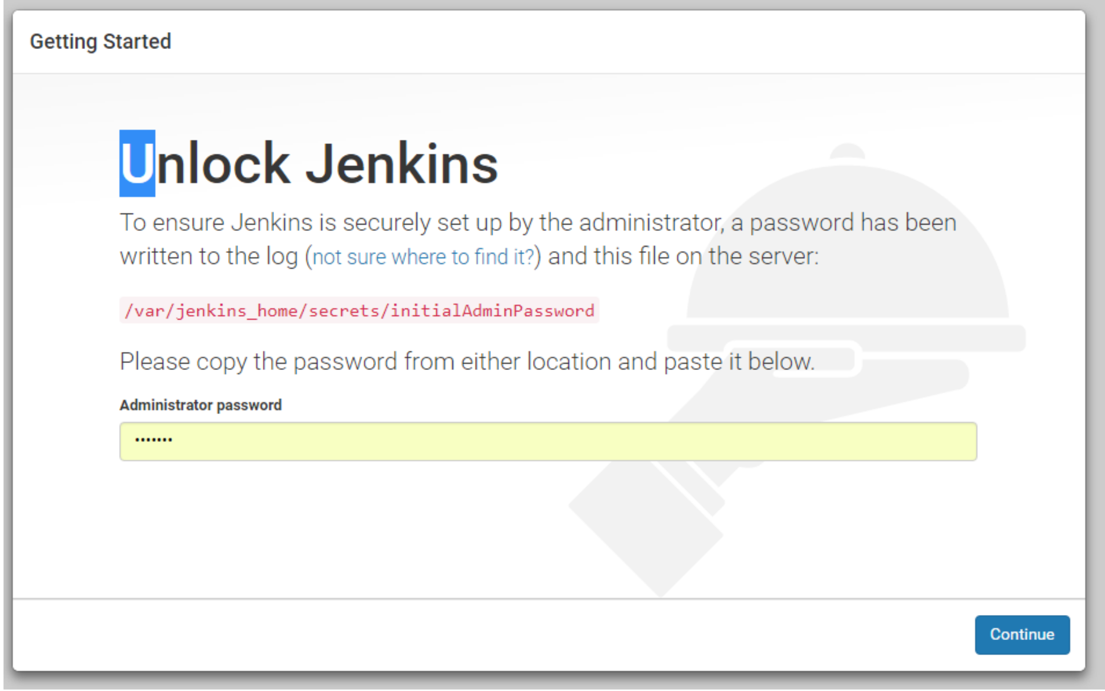
Execute following docker command to run Jenkins:

```
docker run -p 8080:8080 -p 50000:50000 jenkins/jenkins:lts
```

More details on wiki.jenkins.io (https://wiki.jenkins.io/display/JENKINS/Installing+Jenkins+with+Docker)

Then connect to Jenkins at http://localhost:8080 (http://localhost:8080/) using password generated in Jenkins starting logs.

```
ex:
Jenkins initial setup is required. An admin user has been create
d and a password generated.
Please use the following password to proceed to installation:
f421195b70e74f718bd2cc2377debf51
```



Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

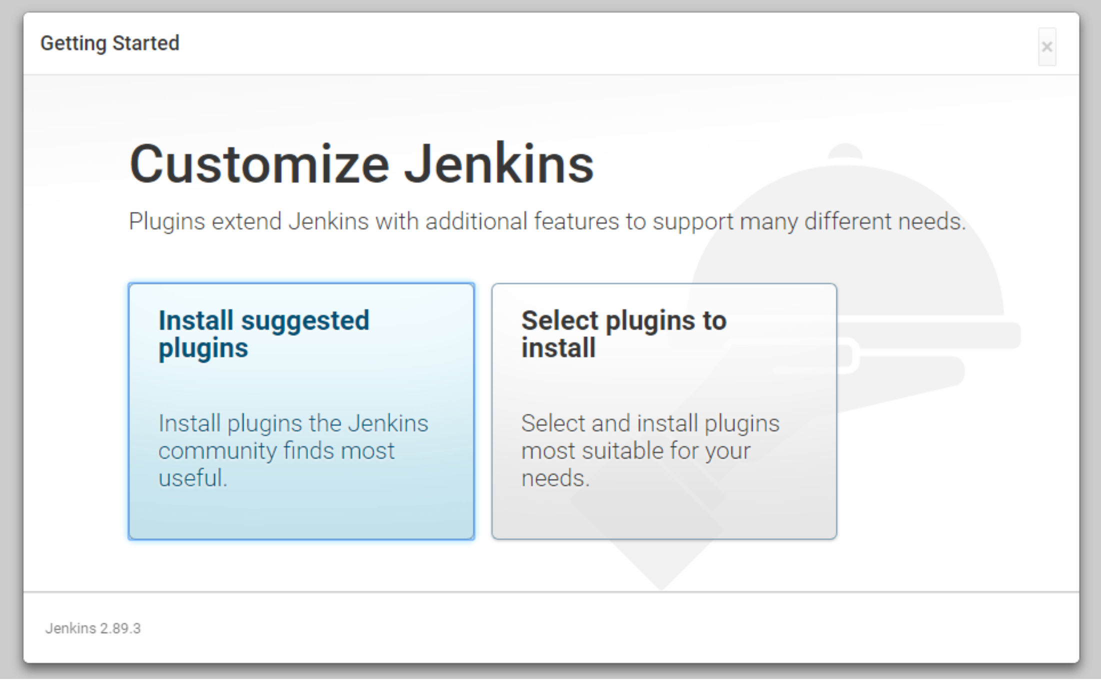
`/var/jenkins_home/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password: •••••••

Install latest plugins

First installation, Jenkins proposes to install a series of suggested plugin. Let’s do it !



Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

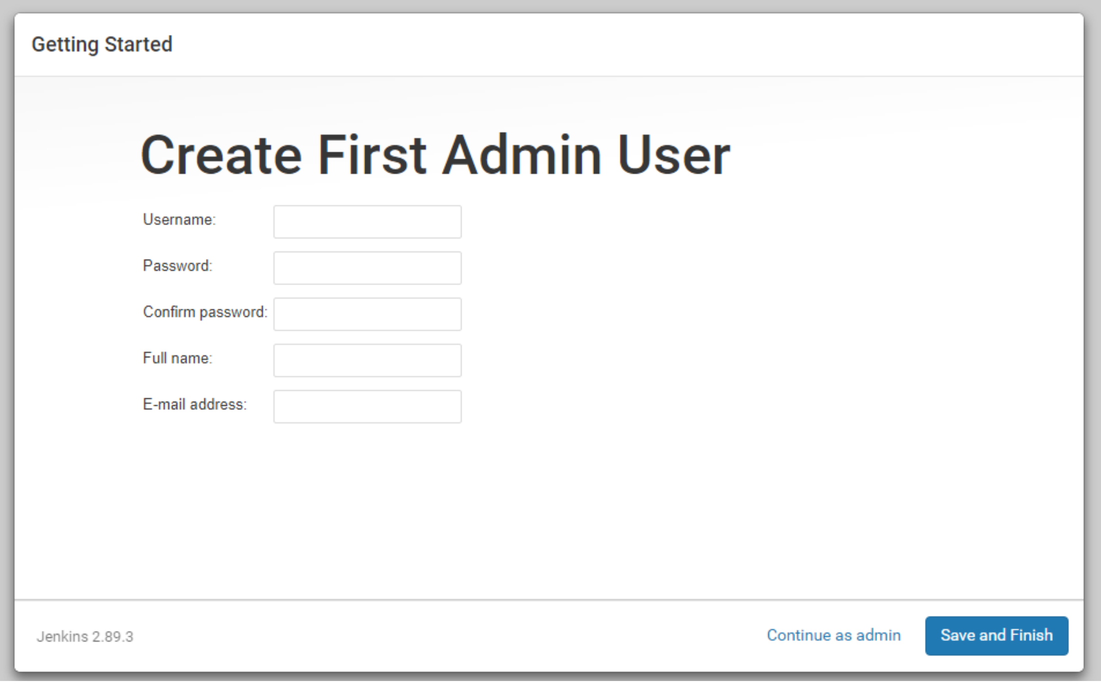
Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.89.3

Create first user



Getting Started

Create First Admin User

Username:

Password:

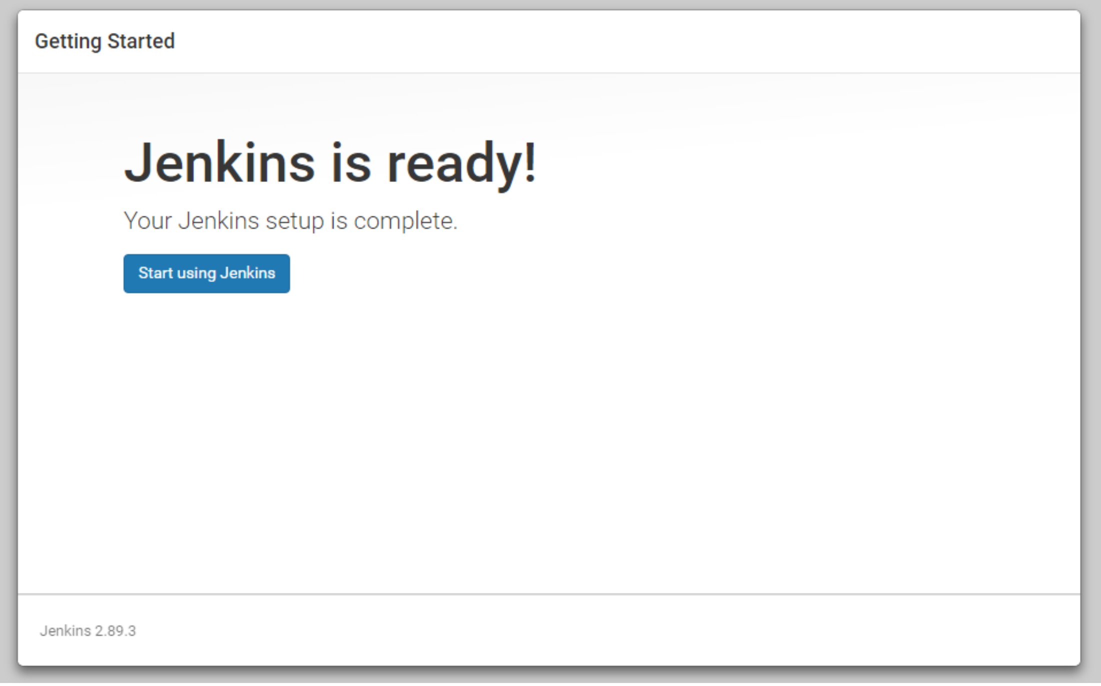
Confirm password:

Full name:

E-mail address:

Jenkins 2.89.3

Ready to rock !



Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

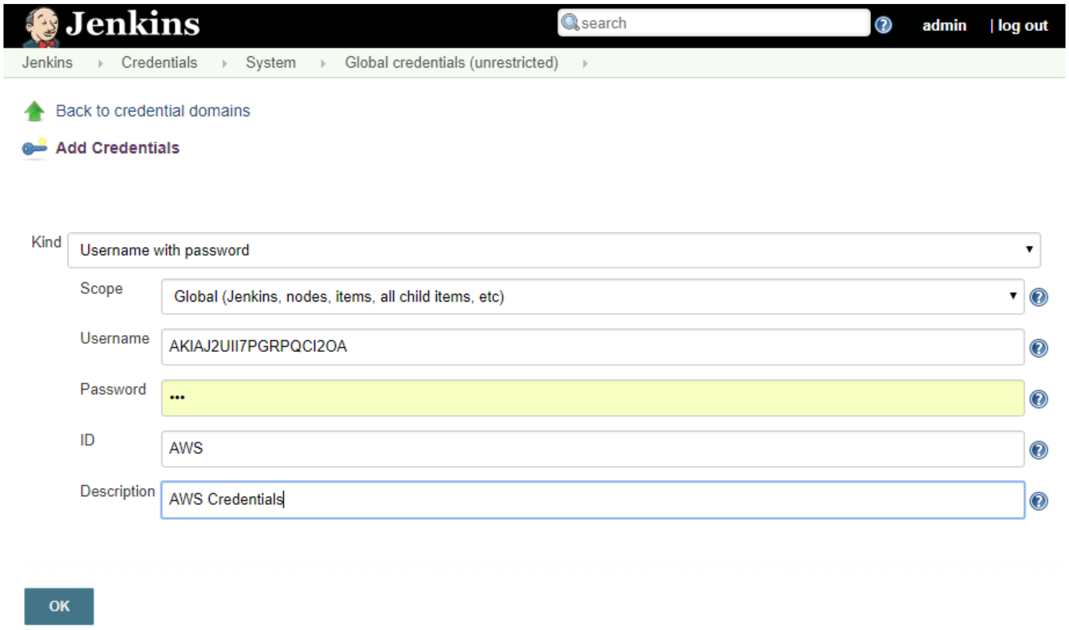
Jenkins 2.89.3

Jenkins Security configuration

To be able to checkout projet source code and deploy to AWS, Jenkins needs to access credentials. Jenkins has its own vault to store credentials.

Navigate to http://localhost:8080/credentials/store/system/domain/_/newCredentials (http://localhost:8080/credentials/store/system/domain/_/newCredentials)

- Create GIT credentials to access your repository (username/password or ssh key)
- Create AWS Username/Password credentials (id: AWS) to login to your AWS account (`AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`). More details on AWS Documentation (https://docs.aws.amazon.com/general/latest/gr/managing-aws-access-keys.html)



Jenkins
search
admin
| log out
Jenkins ▸ Credentials ▸ System ▸ Global credentials (unrestricted) ▸
Back to credential domains
Add Credentials

Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: AKIAJ2UII7PGRPQCI2OA

Password: •••

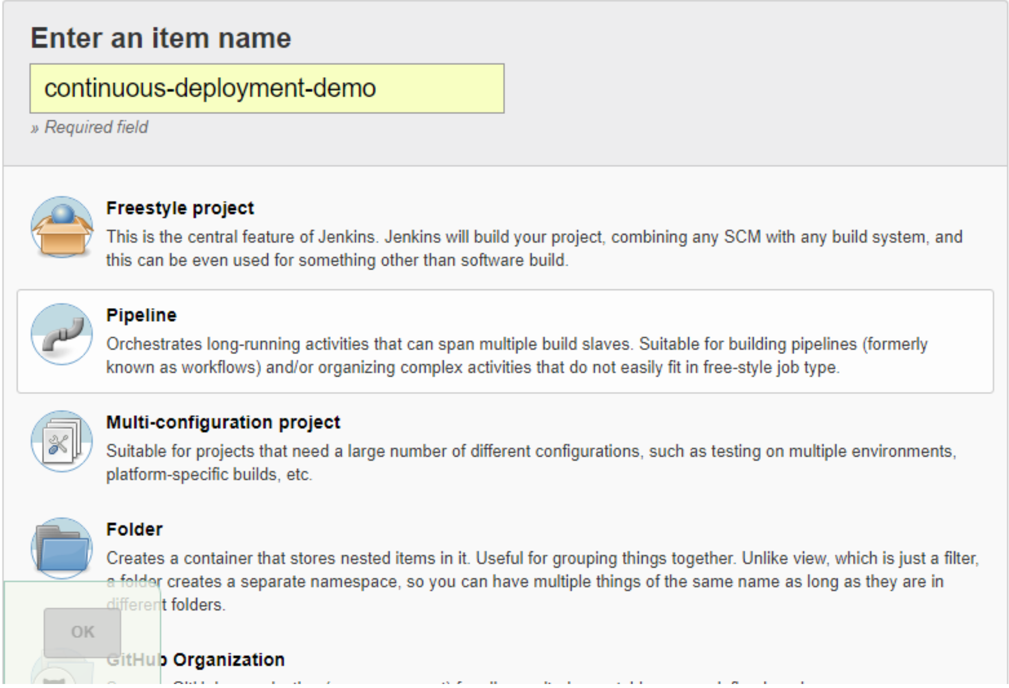
ID: AWS

Description: AWS Credentials

Jenkins vault has sub-level of credentials, in this article credentials are added as global credentials so all jobs can use them.

Create first Jenkins Pipeline Job

Create first pipeline job



Enter an item name

continuous-deployment-demo

» 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.

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.

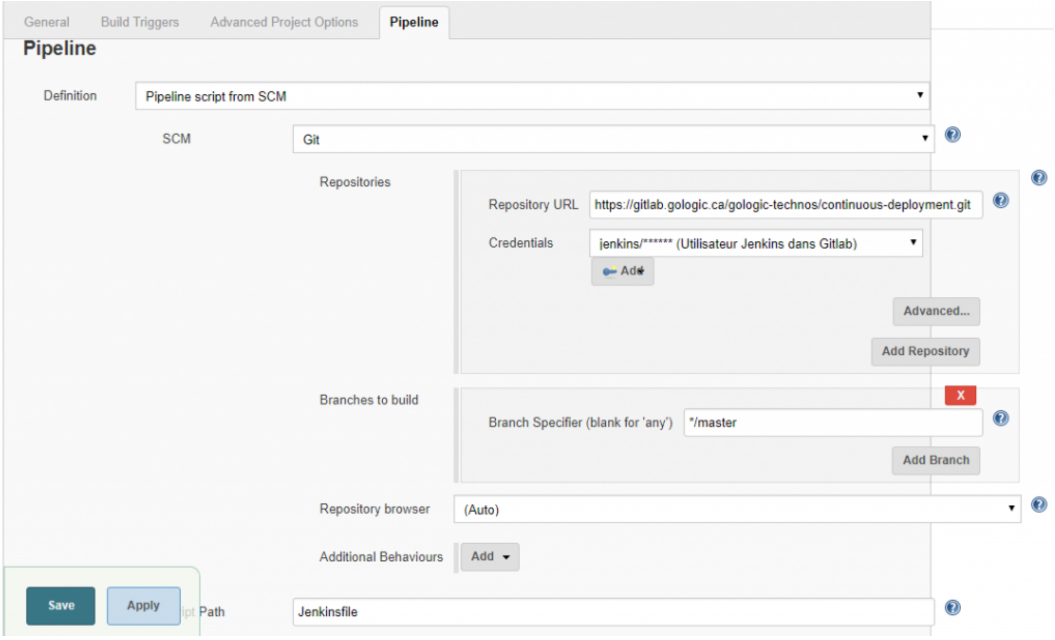
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

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

On tab “Pipeline”:

- As Definition: Pipeline script from SCM
- SCM: GIT
- Repository URL: [your git repository]
- Credentials: [GIT Credentials]
- Let the other default parameters (branch: Master, Script path: Jenkinsfile)
- Then save it and let’s look as pipeline inside project.

- 

Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories

Repository URL: https://gitlab.gologic.ca/gologic-technos/continuous-deployment.git

Credentials: jenkins/****** (Utilisateur Jenkins dans Gitlab)

Branches to build

Branch Specifier (blank for 'any'): */master

Repository browser: (Auto)

Additional Behaviours

Script Path: Jenkinsfile

pipeline-as-code: Let’s get started

In project, open Jenkinsfile and let’s take a look !

```
node {
  // 1. Pipeline options: check for code change in GIT' and keep only two days of history.
  properties([pipelineTriggers([pollSCM('5 * * * *')]),[$class: 'BuildDiscarderProperty', strategy: [$class: 'LogRotator', daysToKeepStr: '2', numToKeepStr: '5']]]);

  // 2. Clean workspace and checkout code from job SCM configurations
  stage("Checkout") {
    deleteDir()
    checkout scm
  }

  // 3. Let's build application within a Maven docker image
  stage("Build") {
      // Compile, Test and Package
      docker.image('maven').inside {
          sh "mvn package"
      }
  }

  // 4. Add Job Junit reporting inside Jenkins
  stage("Report tests") {
      junit allowEmptyResults: true, testResults: '**/TEST-*.xml'
  }

  // 5. Deploy to AWS using Security credentials: ACCESS_KEY and SECRET from custom IAM Jenkins user
  docker.image('chriscamicas/awscli-awsebcli').inside {
    withCredentials([[$class: 'UsernamePasswordMultiBinding', credentialsId: 'AWS', usernameVariable: 'AWS_ACCESS_KEY_ID', passwordVariable: 'AWS_SECRET_ACCESS_KEY']]) {
      // Prepare environment by creating and prepare environments
      stage('Prepare environment') {
        sh 'eb init continuous-deployment-demo -p "64bit Amazon Linux 2017.09 v2.6.4 running Java 8" --region "ca-central-1" '
        // Since AWS failed on create if environment already exists, try/catch block allow to continue deploy without failing
    try {
     sh 'eb create jenkins-env --single'
    } catch(e) {
     echo "Error while creating environment, continu
e..., cause: " + e
    }
    sh 'eb use jenkins-env'
    sh 'eb setenv SERVER_PORT=5000'
   }
   // Ready to deploy our new version !
   stage('Deploy') {
    sh 'eb deploy'
    sh 'eb status'
   }
  }
 }

}
```

To execute Pipeline, Return to Jenkins and click "Build Now" in pipeline job.

The pipeline script (Jenkinsfile) is in the application, it evolves at the same time as the application and is now under version control. Each code change will trigger a new pipeline job and redeploy to AWS !

Conclusion

⊕ Jenkins has a tons of plugins to orchestrate with multiple tools and platforms.

⊕ Jenkins has built-in credentials system, storage, configuration files. Jenkins works like a standalone system.

⊕ Jenkins pipeline-as-code (Jenkinsfile) allows code sharing through shared library mechanics, convenient for common operations.

⊖ Jenkins-cli is not powerfull as other CI-Engine so WebInterface is required in almost *every* case.

⊖ Jenkins pipeline-as-code (Jenkinsfile) is based on Groovy engine but with specific syntax which requires a learning curve.

Suivez-nous et partagez

POSTED IN AFFAIRE (HTTPS://WWW.GOLOGIC.CA/EN/CATEGORY/AFFAIRE/), APPS (HTTPS://WWW.GOLOGIC.CA/EN/CATEGORY/APPS/), EMPLOI (HTTPS://WWW.GOLOGIC.CA/EN/CATEGORY/EMPLOI/), FORMATION (HTTPS://WWW.GOLOGIC.CA/EN/CATEGORY/FORMATION/)

TAGGED #AWSBEANSTALK (HTTPS://WWW.GOLOGIC.CA/EN/TAG/AWSBEANSTALK/), #CI/CD (HTTPS://WWW.GOLOGIC.CA/EN/TAG/CI-CD/), #DEPLOIEMENTCONTINU (HTTPS://WWW.GOLOGIC.CA/EN/TAG/DEPLOIEMENTCONTINU/), #DEVOPS (HTTPS://WWW.GOLOGIC.CA/EN/TAG/DEVOPS/), #DOCKER (HTTPS://WWW.GOLOGIC.CA/EN/TAG/DOCKER/), #GIT (HTTPS://WWW.GOLOGIC.CA/EN/TAG/GIT/), #GITHUB (HTTPS://WWW.GOLOGIC.CA/EN/TAG/GITHUB/), #GROOVY (HTTPS://WWW.GOLOGIC.CA/EN/TAG/GROOVY/), #JAVASPRINGBOOT (HTTPS://WWW.GOLOGIC.CA/EN/TAG/JAVASPRINGBOOT/), #JENKINS (HTTPS://WWW.GOLOGIC.CA/EN/TAG/JENKINS/), #LIVRAISONCONTINUE (HTTPS://WWW.GOLOGIC.CA/EN/TAG/LIVRAISONCONTINUE/), #MAVEN (HTTPS://WWW.GOLOGIC.CA/EN/TAG/MAVEN/), #PIPELINEASCODE (HTTPS://WWW.GOLOGIC.CA/EN/TAG/PIPELINEASCODE/), #SCM (HTTPS://WWW.GOLOGIC.CA/EN/TAG/SCM/)

LEAVE A REPLY

You must be logged in (https://www.gologic.ca/wp-login.php?redirect_to=https%3A%2F%2Fwww.gologic.ca%2Fen%2Fenglish-continuous-deployment-with-pipeline-as-code-quick-start-series-jenkins%2F) to post a comment.

RECENT POSTS

Continuous Deployment with Pipeline as Code – Quick start Series – #3 GitLab (https://www.gologic.ca/en/explore-et-comprends-facilement-les-mecaniques-des-differents-outils-de-pipeline-as-code-3-gitlab/)

Continuous Deployment with Pipeline as Code – Quick start Series – #2 Concourse (https://www.gologic.ca/en/deploiement-continu-en-pipeline-as-code-series-de-demarrage-rapide-1-concourse/)

Continuous Deployment with Pipeline as Code – Quick start Series – #1 Jenkins (https://www.gologic.ca/en/english-continuous-deployment-with-pipeline-as-code-quick-start-series-jenkins/)

Boot Camp DevOps at Desjardins (https://www.gologic.ca/en/boot-camp-devops-chez-desjardins/)

How to Maximize Traceability with Gitlab (https://www.gologic.ca/en/how-to-maximize-traceability-with-gitlab/)

Montreal collective made up of technology enthusiasts

(HTTP://WWW.GOLOGIC.CA/FEED) DERNIÈRES NOUVELLES GOLOGIC (HTTPS://WWW.GOLOGIC.CA/)

Gologic présent au Forum TI DevOps Les Affaires! (https://www.gologic.ca/gologic-present-au-forum-ti-devops-les-affaires/) June 27, 2018

Explore et comprends facilement les mécaniques des différents outils de pipeline as code – #3 GitLab (https://www.gologic.ca/explore-et-comprends-facilement-les-mecaniques-des-differents-outils-de-pipeline-as-code-3-gitlab/) May 29, 2018

Découvre 4 moyens efficaces et faciles pour créer et gérer des tâches dans l'outil Gitlab (https://www.gologic.ca/decouvrez-4-moyens-efficaces-et-faciles-pour-creer-et-gerer-des-taches-dans-loutil-gitlab/) May 22, 2018

WITH US

(HTTPS://GOO.GL/MAPS/
460 Sainte-Catherine St W, suite 942
Montreal, QC
H3B 1A7
(https://goo.gl/maps/zVzKGXe5Aps)

☎ (+1) 438 385 7188 (tel:+14383857188)

✉ contact@gologic.ca (mailto:contact@gologic.ca)

FAX: no way !

ZVZ
KGX
E5A
PS)

BUSINESS HOURS

Monday to Friday: 9 AM to 5 PM