

# Jenkins utilisation : Cahiers de TPs

# TP1

---

# Objectifs

---

- Créer son premier job ;
- Lancer un job ;
- Consulter le log d'un build ;

# Sujet

- Créer dans Jenkins un job de type Freestyle ;
- Ajouter le code suivant dans le partie build :

```
echo "Formation ADEO"
```

- Lancer le job manuellement et consulter le log

# Correction


---

Créer le job en cliquant sur *Nouveau item*, sélectionner le premier type *Construire un projet free-style* :

Configurer le job comme suit : dans la partie *build*, sélectionner dans la liste *Exécuter un script shell* :

Résultat attendu dans la console :

Jenkins ▸ tp1 ▸ #1


 Retour au projet

 État

 Modifications


 **Console Output**

 View as plain text

 Informations de la construction

 Supprimer le build

 Open Blue Ocean



## Sortie de la console

Started by user [Jenkins Admin](#)  
Building in workspace /var/lib/jenkins/jobs/tp1/workspace  
[workspace] \$ /bin/sh -xe /tmp/jenkins8585542782959276384.sh  
+ echo 'Formation ADEO'  
Formation ADEO  
Finished: SUCCESS

# TP2

---

# Objectifs

---

- créer un secret Git



# Sujet

---

- Créer un secret Git **global** de type utilisateur/mot de passe
- Renseigner :
  - Utilisateur : nom d'utilisateur GitHub
  - Mot de passe : Token GitHub

# Correction

Jenkins ▸ Identifiants ▸ System ▸ Identifiants globaux (illimité) ▸ sandrineperrin/\*\*\*\*\* (test-g-pass)

Portée	Global (Jenkins, esclaves, Items, etc...)	?
Nom d'utilisateur	sandrineperrin	P ?
Mot de passe	.....	? ?
ID	test-g-pass	?
Description	github-access-with-token	?

[Retour à la liste](#) [Ajouter un nouveau](#)

**Sauvegarder**

# TP3

---

# Objectifs

---

- Créer un job pipeline

# Sujet

---

# Correction

---

Jenkins > first-pipeline-job >

General Build Triggers Advanced Project Options **Pipeline**

### Pipeline

Definition

Pipeline script

Script

```
1 node{
2   stage('Hello'){
3     echo "Hello ADEO"
4   }
5 }
```

try sample Pipeline...

☒ Use Groovy Sandbox

[Pipeline Syntax](#)

Sauver Apply

The screenshot displays the Jenkins web interface. A modal window titled "Stage Logs (Hello)" is open, showing a log entry: "Print Message -- Hello ADEO -- (self time 24ms)" followed by the message "Hello ADEO".

The background interface shows the "Stage View" for a pipeline named "Hello". The "Average stage times" section indicates an average full run time of approximately 1s. The "Recent Changes" section shows a change on Sep 28 at 11:54, labeled "No Changes". The "Stage View" section shows a bar chart with a single bar for "Hello" with a value of 129ms.

The "Historique des builds" section shows a list of builds, with the first build (#1) dated 28 sept. 2018 09:54. The "Liens permanents" section provides links for "RSS des builds" and "RSS des échecs".



# TP4

---

# Objectifs

---

- Utiliser Git comme source pour un job pipeline

# Sujet

- Créer un projet sur GitHub, et pousser un fichier `tp4.groovy` contenant :

```
node{
  stage('Hello'){
    echo "Hello ADEO"
  }
}
```

- Créer un job pipeline qui utilise le projet git (*Pipeline from SCM*) et renseigner le script `tp4.groovy`, ainsi que le secret Git créé au *TP 2*
- Exécuter le job manuellement

# Correction

---

# TP5

---

# Objectifs

---

- Créer un scripted pipeline avec plusieurs stages

# Sujet

---

- Créer un job tp5 de type pipeline en lien avec le projet git ;
- Créer un fichier tp5.groovy :
  - reprendre le stage Hello du tp4 ;
  - ajouter un nouveau stage pour afficher les variables d'environnement du node (fonction shell env) ;
- Pusher le code sur Github et lancer le job ;

# Correction

Fichier pipeline.groovy

```
node{
  stage('Hello'){
    echo "Hello ADEO"
  }

  stage ('Env variable'){
    sh 'echo Affiche toutes les variables environnement disponibles :'
    sh 'env'
  }
}
```



Résultat dans l'interface Jenkins :

The screenshot displays the Jenkins interface for a pipeline named 'tp5'. On the left, a sidebar contains navigation links: 'Back to Dashboard', 'Status', 'Changes', 'Lancer un build', 'Supprimer Pipeline', 'Configurer', 'Full Stage View', 'Open Blue Ocean', 'Rename', and 'Pipeline Syntax'. The main content area is titled 'Pipeline tp5' and includes a 'Recent Changes' section with a notepad icon. Below this is the 'Stage View' section, which shows 'Average stage times: (Average full run time: ~1s)' and a table of stage durations. The table has two columns: 'hello' and 'Env variable'. The first row shows '49ms' and '147ms' respectively, with progress bars. The second row shows '49ms' and '147ms' in green boxes. To the left of the table is a build history section titled 'Historique des builds' with a search bar and a table showing build #1 on Oct 07 at 15:41 with 'No Changes'. At the bottom, there are 'Liens permanents' for 'RSS des builds' and 'RSS des échecs'.

Jenkins > tp5 >

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Lancer un build](#)

[Supprimer Pipeline](#)

[Configurer](#)


[Full Stage View](#)

[Open Blue Ocean](#)

[Rename](#)

[Pipeline Syntax](#)

## Pipeline tp5

 [Recent Changes](#)

### Stage View

Average stage times:  
(Average full run time: ~1s)

	hello	Env variable
	49ms	147ms
#1 Oct 07 15:41 No Changes	49ms	147ms

### Historique des builds

[tendance](#)

find x

#1 7 oct. 2018 13:41

[RSS des builds](#) [RSS des échecs](#)

### Liens permanents

# TP6

---

# Objectifs

---

- Apprendre à identifier des erreurs
- Utiliser des paramètres
- Utiliser des variables d'environnement
- Utiliser des secrets

# Sujet

---

Télécharger le tp6 :

- Créer un job pipeline pour le tp6
- Exécuter le job, et corriger les erreurs
- Modifier la section *1 - Variables d'environnement* pour afficher le numéro de build du job.
- Modifier la section *3 - Récupération des secrets* pour afficher :
  - L'utilisateur et le mot de passe
  - Le chemin du fichier de secret

# Correction

---

Création de variable globale:

Création d'un paramètre dans le job:

Création d'un credential pour login/password:

Jenkins ▸ Identifiants ▸ System ▸ Identifiants globaux (illimité) ▸ username\_demo/\*\*\*\*\* (use credentials tp6)

[↑ Revenir à Identifiants globaux \(illimité\)](#)  
[🔧 Mettre à jour](#)  
[🚫 Supprimer](#)  
[📁 Déplacer](#)

Portée

Global (Jenkins, esclaves, items, etc...)

?

Nom d'utilisateur

username\_demo

P ?

Mot de passe

.....

P ?

ID

missing\_credential

?

Description

use credentials tp6

?

Sauvegarder

Création d'un credential pour un fichier secret:

Jenkins ▸ Identifiants ▸ System ▸ Identifiants globaux (Illimité) ▸

[Revenir aux identifiants de domaines](#)

[Ajouter des identifiants](#)

Type **Secret file**

Portée **Global (Jenkins, esclaves, items, etc...)**

File **Parcourir...** **tp6\_secret\_file**

ID **missing\_secret\_file**

Description **use tp6**

**OK**



Version corrigé du script tp6 :

[Télécharger tp6-stage1](#)

```
//  
// Variable d'environnement  
//  
node {  
  
    stage('0- clean'){  
        deleteDir()  
        checkout scm  
    }  
  
    stage ('1- Print Jenkins variables'){  
        echo "$env.GLOBAL_JENKINS_VARIABLE"  
    }  
    stage ('1- Print all env'){  
        echo 'Affiche toutes les variables environnement disponibles :'  
        sh 'env'  
    }  
  
    stage ('1- BuildNumber'){  
        echo 'Affichage du numero de build:' + BUILD_NUMBER  
        sh ' echo $BUILD_NUMBER'  
    }  
}
```

Télécharger tp6-stage2

```
//  
// Paramètres utilisateurs du script  
//  
node {  
  stage('2- Print parameter'){  
    sh '''  
      echo "Affichage du paramètre saisie par l'utilisateur"  
      echo "  valeur du paramètre : $script_param"  
    '''  
  
    // Pour récupérer la valeur dans le script  
    def value = params.script_param  
    def value_upper = params.script_param.toUpperCase()  
  
    println "Print default => " + params.missing_param  
    println "Print default => " + value  
    println "Print upper case value => " + value_upper  
  
  }  
}
```

Télécharger tp6-stage3

```
//
// Credentials
//

node {
  stage('3- Récupération des credentials'){
    withCredentials([
      usernamePassword(
        credentialsId: 'log_demo',
        usernameVariable: "DEMO_USERNAME",
        passwordVariable: "DEMO_PASS"
      ),
      file(
        credentialsId: 'secret_file_demo',
        variable: 'SECRET_FILE')
    ]){

      // Espace où sont accessibles les variables définies pour les credentials

      sh '''
        echo -e "Affichage du username $DEMO_USERNAME \t doit être crypté"
        echo -e "Affichage du password $DEMO_PASS \t doit être crypté"

        echo -e "Affichage du path du fichier secret : $SECRET_FILE"
      '''
    } // end stage
  } // end withCredential, les variables ne sont plus accessible après
}
```

# TP 7-1

---

# Objectifs

---

- Création et portée des variables
- Utilisation de fonctions

# Sujet

---

Télécharger le script tp7-1

1. Créer un job pipeline pour le tp7-1;
2. Lancer le job plusieurs fois en changeant les valeurs des paramètres ;

# TP 7-2

---

# Objectifs

---

- Gestion des erreurs
- Création d'une fonction



# Sujet

---

Télécharger le script tp7-2

1. Créer un job pipeline pour le tp7-2;
2. Exécuter le job tp7 plusieurs fois, il génère une erreur si le numéro de build est impair ;
3. Décommenter les codes pour capturer l'erreur et relancer le job plusieurs fois ;
4. Modifier le stage « Gestion erreurs », remplacer les appels à la fonction groovy *println* par une fonction locale :
  - pour afficher le résultat du *if* ;
  - pour afficher un message d'erreur ;

# Correction

Version final du script tp7-2 corrigé :

```
node {  
  
  stage('Gestion erreur'){  
  
    println "Commande avant le try/catch"  
    try {  
  
      println "Commande pouvant générer une erreur"  
      def number = env.BUILD_NUMBER as Integer  
  
      if (number%2) {  
        //println "Number " + number + " is impair"  
        print_result(number, "impair")  
        // levee une exception  
        throw new Exception()  
      }  
      else {  
        // println "Number "+ number +" is PAIR"  
        print_result(number, "pair")  
      }  
  
    } catch (Exception e) {  
      // println "FAIL : la commande échoue, traiter ce cas"  
      fail_message()  
    } finally {  
      println "Commande toujours exécutée"  
    }  
  }  
}  
  
stage('End'){ println 'END'}
```

# TP 8

---

# Objectifs

---

- Utilisation de fonction partagée

# Sujet

---

Télécharger le script tp7-2

1. Créer un job pipeline pour le tp8 ;
2. Dupliquer tp7.groovy en tp8.groovy. Au lieu de créer des fonctions locales pour afficher les messages,
  - créer une fonction partagée **notify** pour afficher les messages ;
  - utiliser le même projet git.

# Correction

Utiliser le projet git comme source de code pour les fonctions partagées.

Ajouter un script groovy dans un dossier vars :

```
# à la racine du projet
mkdir vars

# placer le fichier notify.groovy
```

```
#!/usr/bin/env groovy

def call(status){

    println status + ": Job '${env.JOB_NAME} [${env.BUILD_NUMBER}]': "
    println "Check console output at ${env.BUILD_URL} ${env.JOB_NAME} [${env.BUILD_NUMBER}] "
}
```

Configuration des fonctions partagées :

Version final du script tp8 corrigé :

[Télécharger le corrigé du tp8](#)