Vous trouvez sous ce lien la syntaxe du Pipeline : https://www.jenkins.io/doc/book/pipeline/syntax/

Exemple 1:

Créez un Pipeline line pour afficher le message «Bienvenu à Lille »

```
node {
    echo "Bienvenu à Lille"
}
```

Exemple 2 :

Modifiez le Pipeline précédent pour stocker le nom de la ville « Lille » dans une variable ville

```
node {
    def ville  = "Lille"
    echo "Bienvenu à ${ville}"
}
```

Exemple 3 :

Créez un Pipeline déclaratif pour afficher le message “Hello world”

```
pipeline{
    agent any
    stages{
        stage("Build"){
            steps{
                echo "Hello World"
            }
        }
    }
}
```

Exemple 4:

Changez l’exemple 3 et modifier agent à none, observez le résultat.

```
pipeline{
    agent none
    stages{
        stage("Build"){
            steps{
                echo "Hello World"
            }
        }
    }
}
```

⇨ Exécuter le job sur le nœud principal

Exemple 5:

Changez l'exemple 3 pour l'exécuter sur le nœud slave avec le label linux.

```
pipeline{
    agent{
        label "linux"
    }
    stages{
        stage("Build"){
            steps{
                echo "Hello World"
            }
        }
    }
}
```

Exemple 6

Modifier l'exemple 5 pour spécifier le `customWorkspace` utilisé lors de l'exécution du job.

```
pipeline{
    agent{
        node{
            label "linux"
            customWorkspace "/root/customeWorkspace"
        }

    }
    stages{
        stage("Build"){
            steps{
                echo "Hello World"
            }
        }
    }
}
```

Exemple 7

Créer un Pipeline permettant de déclarer un bloc steps et à l'intérieure, déclare un script permettant de :

- Initialiser une variable
- Selon le contenu du variable affiche un message adapté en utilisant la structure conditionnelle if
- Attend 5 seconds

```
pipeline {
  agent any
  stages {
    stage('Build') {
       steps {
           script{
              def cursus = "Devops"

              if(name == "Devops")
                  println("Nous allons voir plein d'outils d'automatisation ${name}")
              else
                  println("Noud allons coder plein de programme")

              sleep 2
              echo "Fin du script"
           }
        }
     }
  }
}
```

Exemple 8

Créer un Pipeline permettant de récupérer le Tools maven puis afficher la version. Utiliser l'aide Syntaxe Pipeline pout savoir comment utiliser le bloc tools.

```
pipeline {
    agent any

    stages {
        stage('Build') {
            tools {
                  maven 'Maven3'
            }
            steps {
                sh 'mvn --version'
            }
        }
    }
}
```

⇨ Le tools on peut l'utiliser dans le stage ou dans le step

Exemple 9

Dans des Pipelines scriptés, utilisez le bloc retry et la commande Error (pour générer une erreur) afin de vérifier le comportement du Pipline, testez les cas suivants :

- Retry dans le stage
- Le stage dans le retry
- Le node dans le retry

- Retry dans le stage

```
node{
	stage('Build'){
		retry(3){
			error "Error statement just got executed"
		}
	}
}
```

- Le stage dans le retry

```
node{
	retry(3){
		stage('Build'){
			error "Error statement just got executed"
		}
	}
}
```

- Le node dans le retry

```
retry(3){
	node{
		error "Error statement just got executed"
	}
}
```

Exemple 10

Dans un Pipeline scripté, testez la commande timeout pour arrêter l'exécution d'un script, utiliser sleep pour simuler l'exécution pendant une période.

```
node{
	stage('Build'){

		//values: NANOSECONDS, MICROSECONDS, MILLISECONDS, SECONDS, MINUTES,
HOURS, DAYS
		timeout(time: 1, unit: 'SECONDS')  {
			//sleeping for 2 seconds
			echo 'Avant Sleep'
			sleep 2
			echo 'Apres sleep'
		}
	}
}
```

Exemple 11

Dans un Pipeline déclaratif, utilisez le Timestamps, pour afficher le Timestamps de chaque commande, assurez-vous que le plugin Timestamper est installé.

```
pipeline{
    agent any
    stages{
        stage('Build'){
                  steps{
                      timestamps {
                              echo "Hello World in Build"
                              echo "Hello World in Build Again"
                        }
                  }
        }

        stage('Test'){
            steps{
                echo "Hello World in Test"
                echo "Hello World in Test Again"
            }
        }
    }
}
```

Exemple 12

Dans un Pipeline déclaratif, utilisez l'instruction conditionnelle when, pour paramétrer l'exécution selon une variable d'environnement spécifique.

```
pipeline {
   agent any
      environment{
            DEPLOY_TO='production'
      }
    stages {
        stage('Build') {
            when {
                environment name: 'DEPLOY_TO', value: 'production'
            }


            steps {
                echo 'Deploying'
            }
        }
    }
}
```

Utilisez les expressions et l'instruction not pour changer le comportement du pipeline