Jenkins CI Pipeline 생성 실습

💡 **실습 목표**
1. Jenkinsfile 의 기본적인 구조를 알아보고 생성한다.
2. Jenkins 에서 Pipeline Job 을 생성하고 빌드한다.
3. post 를 이용해 모든 stage 가 실행된 후의 명령을 정의한다.
4. when 을 이용해 stage 가 실행되는 조건을 추가해 본다.
5. Jenkinsfile 환경변수를 설정해 본다.
6. parameters 를 이용하는 방법을 알아 본다.
7. 외부 groovy scripts 를 만들어 사용해 본다.
8. Replay 에서 수정 후 빌드 다시 실행해 보기

Jenkinsfile 의 기본적인 구조를 알아보고 생성한다.

- 기본 코드 구조(Section)
  - pipeline : 반드시 맨 위에 있어야 한다.
    일종의
    작업명세서.

    CI/CD flow를 젠킨스에 구현하기 위한 일련의 플러그인의 집합이자 구성이며 젠킨스의 핵심 플러그인.
    젠킨스는 여러 플러그인들을 pipeline에서 용도에 맞게 사용하고 정의함으로써 CI/CD를 할 수 있게 해주는 도구.
  - agent : 어디에서 실행할 것인지 정의한다.

    여러 slave node 를 두고 일을 시킬 수 있는데 어떤 젠킨스가 일을
    하게 할 것 인지를 지정한다.
    젠킨스 노드 관리에서 새로 노드를 띄우거나 혹은 docker 이미지 등을 통해
    서 처리할 수 있음
    - any, none, label, node, docker, dockerfile, kubernetes
    - agent 가 none 인 경우 stage 에 포함시켜야 함

```
pipeline {
    agent none
    stages {
        stage('Example Build') {
            agent { docker 'maven:3-alpine' }
            steps {
                echo 'Hello, Maven'
                sh 'mvn --version'
            }
        }
        stage('Example Test') {
            agent { docker 'openjdk:8-jre' }
            steps {
                echo 'Hello, JDK'
                sh 'java -version'
            }
        }
```

```
    }
}
```

- stages : 어떤 일들을 처리할 건지 일련의 stage를 정의 (카테고리)
  - pipeline 블록 안에서 한 번만 실행 가능함

```
pipeline {
    agent any
    stages {
        stage("build") {
            steps {
                echo 'building the applicaiton...'
            }
        }
        stage("test") {
            steps {
                echo 'testing the applicaiton...'
            }
        }
        stage("deploy") {
            steps {
                echo 'deploying the applicaiton...'
            }
        }
    }
}
```

- 본인 Github 에 새 Repository [js-pipeline-project] 생성
- Local(**윈도우PC**) 에서 'Jenkinsfile' 파일 생성하여 위 stages 코드 복사한 후 Github 업로드
- 우분투에서 진행시 git을 설치하고 이름과 이메일 설정 해야되며 master가 기본입니다.
  (git 자동로그인 방법
  https://daechu.tistory.com/33)

```
mkdir jenkins
cd jenkins

#vi Jenkinsfile
#위 stages코드 복사

#git install 필요
git init
git remote add origin <github주소>

#Jenkinsfile을 위 코드로 생성

git add .
git commit -m "first commit”
#이메일, 이름 입력 필요
```

```
git push origin master
#github login(password는 github 키값. 없으면 다시 발급)
```

Jenkins 에서 Pipeline Job 을 생성하고 빌드 한다.

- Pipeline 생성
  - Create a job → pipeline 선택
  - 이름은 'jenkins-pipeline' 으로 입력
- Git 추가
  - Pipeline
    - Definition → Pipeline sctipt from SCM
    - SCM → Git
    - Repository URL 입력
    - Credentials - 현재 public으로 생성했으므로 생략
    - [Save] - Log 출력
- Pipeline Status 확인
  - 각 stage log 확인
  - Build Now
  - Build History → 해당 빌드 번호 선택
  - Console Output

post 를 이용해 모든 stage 가 실행된 후의 명령을 정의한다.

- post 조건
  - always, changed, fixed, regression, aborted, success, unsuccessful, unstable, failure, notBuilt, cleanup

```
pipeline {
    agent any
    stages {
        stage("build") {
            steps {
                echo 'building the applicaiton...'
            }
        }
        stage("test") {
            steps {
                echo 'testing the applicaiton...'
            }
        }
        stage("deploy") {
            steps {
                echo 'deploying the applicaiton...'
            }
        }
```

```
    }
    post {
            always {
                echo 'building..'
            }
            success {
                echo 'success'
            }
            failure {
                echo 'failure'
            }
        }
    }
```

git push

```
git add .
git commit -m "post test commit”
git push origin master
```

젠킨스 → Build Now

when 을 이용해 stage 가 실행되는 조건을 추가 해본다.

- when 조건 추가
  - test stage 에서 Branch 이름에 따른 조건 추가
  - build stage 에서 Branch 이름에 따른 조건 추가
  - deploy stage의 steps에 echo "${env.GIT_BRANCH}” 추가 후 현재 branch 이름 확인

```
pipeline {
    agent any
    stages {
        stage("build") {
            when {
                expression {
                    env.GIT_BRANCH == 'origin/master'
                }
            }
            steps {
                echo 'building the applicaiton...'
            }
        }
        stage("test") {
            when {
                expression {
                    env.GIT_BRANCH == 'origin/test' || env.GIT_BRANCH == ''
                }
            }
```

```
            steps {
                echo 'testing the applicaiton...'
            }
        }
        stage("deploy") {
            steps {
                echo 'deploying the applicaiton...'
            }
        }
    }
}
```

Jenkinsfile 환경변수를 설정 해본다.

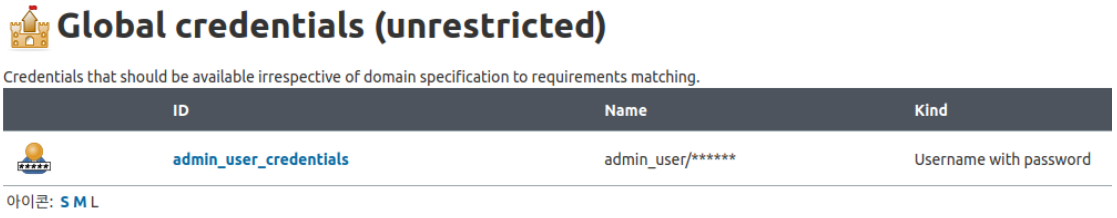
- Jenkinsfile 자체 환경변수 목록 보기
  - http://localhost:8080/env-vars.html/ 접속 / 또는 Jenkins pipeline-syntax 참고
- Custom 환경변수 사용하기
  - **echo 사용 시 큰 따옴표 주의**

```
pipeline {
    agent any
    environment {
        NEW_VERSION = '1.0.0'
    }
    stages {
        stage("build") {
            steps {
                echo 'building the applicaiton...'
                echo "building version ${NEW_VERSION}"
            }
        }
        stage("test") {
            steps {
                echo 'testing the applicaiton...'
            }
        }
        stage("deploy") {
            steps {
                echo 'deploying the applicaiton...'
            }
        }
    }
}
```

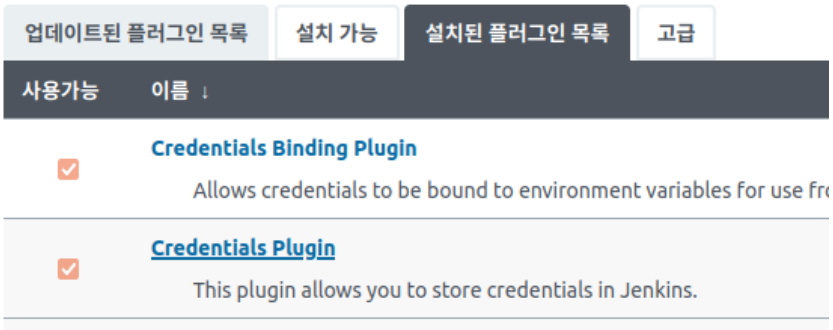
- Credentials 자격 증명 환경 변수로 사용하기
  - Jenkins credential 추가
    - [Jenkins 관리]-[Manage Credentials]-[Global credentials]-[Add credentials]
    - Username : admin_user / Password : 1234 / ID : admin_user_credentials

- Jenkinsfile 에서 환경변수로 사용

```
pipeline {
    agent any
    environment {
        NEW_VERSION = '1.0.0'
        ADMIN_CREDENTIALS = credentials('admin_user_credentials')
    }
    stages {
        stage("build") {
            steps {
                echo 'building the applicaiton...'
                echo "building version ${NEW_VERSION}"
            }
        }
        stage("test") {
            steps {
                echo 'testing the applicaiton...'
            }
        }
        stage("deploy") {
            steps {
                echo 'deploying the applicaiton...'
                echo "deploying with ${ADMIN_CREDENTIALS}"
                sh 'printf ${ADMIN_CREDENTIALS}'
            }
        }
    }
}
```



- Jenkins 플러그인 중 Credentials Plugin 확인



- UsernamePasswordMultiBinding Class

https://javadoc.jenkins-ci.org/plugin/credentials-binding/index.html?org/jenkinsci/plugins/credentialsbinding/impl/UsernamePasswordMultiBinding.DescriptorImpl.html

```
pipeline {
    agent any
    environment {
        NEW_VERSION = '1.0.0'
    }
    stages {
        stage("build") {
            steps {
                echo 'building the applicaiton...'
                echo "building version ${NEW_VERSION}"
            }
        }
        stage("test") {
            steps {
                echo 'testing the applicaiton...'
            }
        }
        stage("deploy") {
            steps {
                echo 'deploying the applicaiton...'
                withCredentials([[$class: 'UsernamePasswordMultiBinding',
                    credentialsId: 'admin_user_credentials',
                    usernameVariable: 'USER',
                    passwordVariable: 'PWD'
                ]]) {
                    sh 'printf ${USER}'
                }
            }
        }
    }
}
```

parameters 를 이용하는 방법을 알아본다.

- Jenkinsfile 에 parameter 추가
- **Build with Parameters 메뉴로 확인**

```
pipeline {
    agent any
    parameters {
        string(name: 'VERSION', defaultValue: '', description: 'deployment vers
        choice(name: 'VERSION', choices: ['1.1.0','1.2.0','1.3.0'], description
        booleanParam(name: 'executeTests', defaultValue: true, description: '')
    }
    stages {
```

```
        stage("build") {
            steps {
                echo 'building the applicaiton...'
            }
        }
        stage("test") {
            steps {
                echo 'testing the applicaiton...'
            }
        }
        stage("deploy") {
            steps {
                echo 'deploying the applicaiton...'
            }
        }
    }
}
```

- executeTests 가 true 인 경우의 조건 추가해보기
- 실제 Jenkinsfile 에 적용해 본다.
  - Build with Parameters 실행
    - 1.2.0 버전 선택
    - executeTests 선택 해제
  - test stage 를 건너뛰고 실행되는지 확인

```
pipeline {
    agent any
    parameters {
        choice(name: 'VERSION', choices: ['1.1.0','1.2.0','1.3.0'], description
        booleanParam(name: 'executeTests', defaultValue: true, description: '')
    }
    stages {
        stage("build") {
            steps {
                echo 'building the applicaiton...'
            }
        }
        stage("test") {
            when {
                expression {
                    params.executeTests
                }
            }
            steps {
                echo 'testing the applicaiton...'
            }
        }
        stage("deploy") {
            steps {
```

```
                echo 'deploying the applicaiton...'
                echo "deploying version ${params.VERSION}"
            }
        }
    }
}
```

외부 groovy scripts 를 만들어 사용해 본다.

- groovy script 추가

  [script.groovy]

```
def buildApp() {
    echo 'building the applications...'
}

def testApp() {
    echo 'testing the applications...'
}

def deployApp() {
    echo 'deploying the applicaiton...'
    echo "deploying version ${params.VERSION}"
}
return this
```

  [Jenkinsfile]

```
pipeline {
    agent any
    parameters {
        choice(name: 'VERSION', choices: ['1.1.0','1.2.0','1.3.0'], description
        booleanParam(name: 'executeTests', defaultValue: true, description: '')
    }
    stages {
        stage("init") {
            steps {
                script {
                    gv = load "script.groovy"
                }
            }
        }
        stage("build") {
            steps {
                script {
                    gv.buildApp()
                }
            }
        }
```

```
        stage("test") {
            when {
                expression {
                    params.executeTests
                }
            }
            steps {
                script {
                    gv.testApp()
                }
            }
        }
        stage("deploy") {
            steps {
                script {
                    gv.deployApp()
                }
            }
        }
    }
}
```

- Jenkinsfile 의 모든 환경변수는 groovy script 에서 사용 가능하다.
- Github Repo 에 반영하고 실행/로그 확인
- 빌드 결과 확인

Replay 에서 수정 후 빌드 다시 실행해 보기

- testApp 에 echo 'Replay' 를 추가 후 다시 빌드