# Hands-on Jenkins Day-2 : Jenkins Jobs and SCM

Purpose of the this hands-on training is to learn how to trigger Jenkins jobs with different ways.

## Learning Outcomes

At the end of the this hands-on training, students will be able to;

- integrate your Jenkins server with Github
- trigger Jenkins jobs with webhook
- trigger Jenkins jobs with Poll SCM

## Outline

- Part 1 - Integrating Jenkins with GitHub using Webhook
- Part 2 - Integrating Jenkins Pipeline with GitHub Webhook
- Part 3 - Configuring Jenkins Pipeline with GitHub Webhook to Run the Python Code
- Part 4 - Creating a Pipeline with Poll SCM

## Part 1 - Jenkins with GitHub using Webhook

- Create a public project repository `webhook-project1` on your GitHub account.
- Clone the `webhook-project1` repository on local computer.

```
git clone <your-repo-url>
```

- Go to your local repository.

```
cd webhook-project1
```

- Write a simple python code which prints `Hello World` and save it as `hello-world.py`.

```
print('Hello World')
```

- Commit and push the local changes to update the remote repo on GitHub.

```
git add .
git commit -m 'added code'
```

```
git push
```

- Go back to Jenkins dashboard and click on `New Item` to create a new job item.
- Enter `webhook-project1` then select `free style project` and click `OK`.
- Enter `My first job webhook from GitHub` in the description field.
- Put a checkmark on `Git` under `Source Code Management` section, enter URL of the project repository.

```
https://github.com/<your-github-account-name>/webhook-project1/
```

- Put a checkmark on `GitHub hook trigger for GITScm polling` under `Build Triggers` section,
- Check `Branch Specifier`. It must be same branch name with your `webhook-project1` Github repository.
- Go to `Build` section and choose "Execute Shell Command" step from `Add build step` dropdown menu.
- Write down `python hello-world.py` to execute shell command, in a box shown.
- Click `apply` and `save`.
- Go to the Jenkins project page and click `Build Now`. The job has to be executed manually one time in order for the push trigger and the git repo to be registered.
- Go to your Github `webhook-project1` repository page and click on `Settings`.
- Click on the `Webhooks` on the left hand menu, and then click on `Add webhook`.
- Copy the Jenkins URL from the AWS Management Console, paste it into `Payload URL` field, add `/github-webhook/` at the end of URL, and click on `Add webhook`.

```
http://ec2-ip-adress-.compute-1.amazonaws.com:8080/github-webhook/
```

- Change the python code on your local repository to print `Hello World 2nd update` and save.

```
print('Hello World 2nd update')
```

- Commit and push the local changes to update the remote repo on GitHub.

```
git add .
git commit -m 'code updated'
git push
```

- Observe the new built under `Build History` on the Jenkins project page.
- Explain the details of the built on the Build page.
- Go back to the project page and explain the GitHub Hook log.

## Part 2 - Jenkins Pipeline with GitHub Webhook

- Go to your Github `pipeline-project` repository page and click on `Settings`.
- Click on the `Webhooks` on the left hand menu, and then click on `Add webhook`.
- Copy the Jenkins URL from the AWS Management Console, paste it into `Payload URL` field, add `/github-webhook/` at the end of URL, and click on `Add webhook`.

```
http://ec2-ip-address.compute-1.amazonaws.com:8080/github-webhook/
```

- Go to the Jenkins dashboard and click on `New Item` to create a pipeline.
- Enter `pipeline-project-webhook` then select `Pipeline` and click `OK`.
- Enter `Simple pipeline and GitHub Webhook` in the description field.
- Put a checkmark on `GitHub Project` under `General` section, enter URL of the project repository.

```
https://github.com/<your-github-account-name>/pipeline-project/
```

- Put a checkmark on `GitHub hook trigger for GITScm polling` under `Build Triggers` section.
- Go to the `Pipeline` section, and select `Pipeline script from SCM` in the `Definition` field.
- Select `Git` in the `SCM` field.
- Enter URL of the project repository, and let others be default.

```
https://github.com/<your-github-account-name>/pipeline-project.git
```

- Click `apply` and `save`. Note that the script `Jenkinsfile` should be placed under root folder of repo.
- Go to the Jenkins project page and click `Build Now`.The job has to be executed manually one time in order for the push trigger and the git repo to be registered.
- Now, to trigger an automated build on Jenkins Server, we need to change any file it the repo, then commit and push the change into the GitHub repository. So, update the `Jenkinsfile`on your local repository with the following pipeline script.

```
pipeline {
    agent any
    stages {
        stage('build') {
            steps {
                echo 'Welcome to Jenkins World'
                sh 'echo Integrating Jenkins Pipeline with GitHub Webhook using
Jenkinsfile'
            }
        }
    }
}
```

- Commit and push the change to the remote repo on GitHub.

```
git add .
git commit -m 'updated Jenkinsfile'
git push
```

- Observe the new built triggered with `git push` command under `Build History` on the Jenkins project page.
- Explain the built results, and show the `Integrating Jenkins Pipeline with GitHub Webhook using Jenkinsfile` output from the shell.
- Explain the role of `Jenkinsfile` and GitHub Webhook in this automation.

## Part 3 - Configuring Jenkins Pipeline with GitHub Webhook to Run the Python Code

- To build the `python` code with Jenkins pipeline using the `Jenkinsfile` and `GitHub Webhook`, we will leverage from the same job created in Part 2.
- To accomplish this task, we need;
    - a python code to build
    - a python environment to run the pipeline stages on the python code
    - a Jenkinsfile configured for an automated build on our repo
- Create a python file on the `pipeline-project` local repository, name it as `pipeline.py`, add coding to print `My first python job within Jenkinsfile.` and save.

```
print('My first python job  within Jenkinsfile.')
```

- Update the `Jenkinsfile` with the following pipeline script, and explain the changes.

```
pipeline {
    agent any
    stages {
        stage('run') {
            steps {
                echo 'Welcome to Jenkins World'
                sh 'python --version'
                sh 'python pipeline.py'
            }
        }
    }
}
```

- Commit and push the changes to the remote repo on GitHub.

```
git add .
git commit -m 'updated jenkinsfile and added pipeline.py'
git push
```

- Observe the new built triggered with `git push` command on the Jenkins project page.

## Part 4 - Creating a Pipeline with Poll SCM

- Go to the Jenkins dashboard and click on `New Item` to create a pipeline.
- Enter `pipeline-pollSCM` then select `Pipeline` and click `OK`.
- Enter `This is a pipeline project with pollSCM` in the description field.
- We will use same github repo project in Part 2 (named as `pipeline-project`).
- Go to the `Pipeline` section.
    - for definition, select `Pipeline script from SCM`
    - for SCM, select `Git`
        - for `Repository URL`, select `https://github.com/<your-github-account-name>/pipeline-project/`, show the `Jenkinsfile` here.
        - approve that the `Script Path` is `Jenkinsfile`
- `Save` and `Build Now` and observe the behavior.
- Go to the the `Configure` and skip to the `Build Triggers` section
    - Select Poll SCM, and enter `* * * * *` (5 stars)
- `Save` the configuration.
- Go to the GitHub repo and modify some part in the `Jenkinsfile` and commit.

- Observe the auto build action at Jenkins job.