



# Continuous Delivery

## Lab 1

Tennis Smith

1.0

# Table of Contents

Prerequisites .....	1
Lab 1 .....	1
Step 1: Login to GitHub.....	1
Step 2: Fork the lab1 repo .....	1
Step 3: Clone a copy of the lab1 repo.....	1
Step 4: Update the Jenkinsfile .....	2
Step 5: Commit Changes and Push to Central Repository .....	2
Step 6: Setup a Jenkins pipeline .....	2

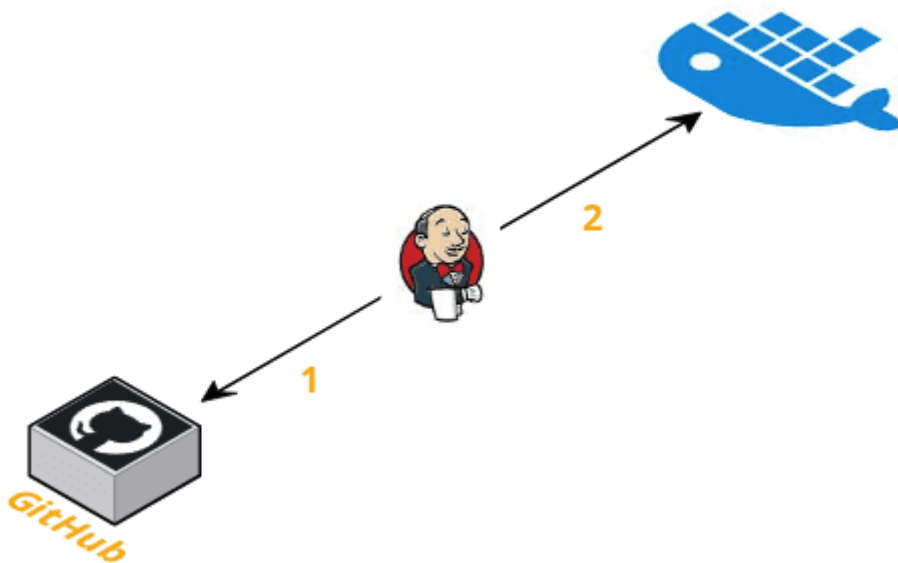
# Prerequisites

Students will need a computer with:

- An Amazon Workspaces login (see your instructor for details)

## Lab 1

The goal of this lab is very simple. We will setup a pipeline on Jenkins that detects source changes. When changes are detected (1), we will build a Docker image and upload it to a Docker registry (2)



### Step 1: Login to GitHub

- In your workspaces session, open a web browser and login to github: <http://github.com>

**TIP** If you don't have a GitHub account, go to <http://github.com> and create a free account

### Step 2: Fork the lab1 repo

- Go to this url: [https://github.com/RoundTower-io/cd\\_workshop\\_lab1](https://github.com/RoundTower-io/cd_workshop_lab1)
- Fork the repo by clicking on the "Fork" button in the upper right of the screen.
- This will create a copy of the lab1 repo under your own GitHub id

### Step 3: Clone a copy of the lab1 repo

- In your workspace session, open a new terminal window by clicking on the Powershell icon.



- Make a local copy of the repo by cloning it with the following command

```
git clone https://github.com/<your user name>/cd_workshop_lab1.git
```

## Step 4: Update the Jenkinsfile

- Go to the home directory of your new repo

```
cd cd_workshop_lab1
```

- Now edit the file **Jenkinsfile**

```
atom Jenkinsfile
```

- Change every occurrence of **training99** to your training ID (assigned by the instructor).
- Save and exit the file

## Step 5: Commit Changes and Push to Central Repository

- First, change the working directory.

```
cd ~/cd_workshop_lab1
```

- Next, add all altered files to the change set.

```
git add .
```

- Next, commit the changes.

```
git commit -m "Updated Jenkinsfile"
```

- Last, push the change to GitHub.

```
git push
```

## Step 6: Setup a Jenkins pipeline

- Login to Jenkins at <http://jenkins.roundtower.io>
- Login using your assigned training id (get it from your instructor).
- Click on the "New Item" option on the main menu



People

Build History

Manage Jenkins

My Views

Credentials

New View

## Build Queue

No builds in the queue.

## Build Executor Status

1 Idle

2 Idle

- Name your new pipeline `<your training id>_lab1` and select `pipeline` as the type. Then click `ok` to save it.

**Enter an item name**

training1\_lab1

» 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 development.
- 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 span multiple build slaves.
- External Job**  
This type of job allows you to record the execution of a process run outside Jenkins, even on a remote machine. This is designed so that you can use Jenkins as a dashboard for monitoring the execution of a process.
- 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 as many items as you want as long as they are in different folders.
- GitHub Organization**  
Scans a GitHub organization (or user account) for all repositories matching some defined markers.
- Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

- Next, click on the **Poll SCM** option (about halfway down the page) and enter 5 asterisks in the field. This will cause Jenkins to look at GitHub once per minute for changes. If changes are found, then the pipeline will run.

## Build Triggers

- ☐ Build after other projects are built
- ☐ Build periodically
- ☐ GitHub hook trigger for GITScm polling
- ☒ **Poll SCM**

**Schedule**

\*\*\*\*\*

- At the bottom of the page, set the **Definition** field to **Pipeline script from SCM**, then set the **SCM** field to **Git**. Put your lab1 url in the **Repository URL** field. Finally, click on **Save** to save all your work.

## Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL `https://github.com/gamename/cd_workshop_lab1.git`

Credentials `- none -` [Add](#)

Branches to build

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

Repository browser `(Auto)`

Additional Behaviours [Add](#)

Script Path `Jenkinsfile`

Lightweight checkout ☒

[Pipeline Syntax](#)

Save

Apply

- At this point, you should see your pipeline run automatically. If so, you will see output something like this

