



# 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.....	2
Step 4: Update the Jenkinsfile .....	2
Step 5: Commit Changes and Push to Central Repository .....	2
Step 6: Setup a Jenkins pipeline .....	3
Step 7: Verify your docker image uploaded to the registry .....	6

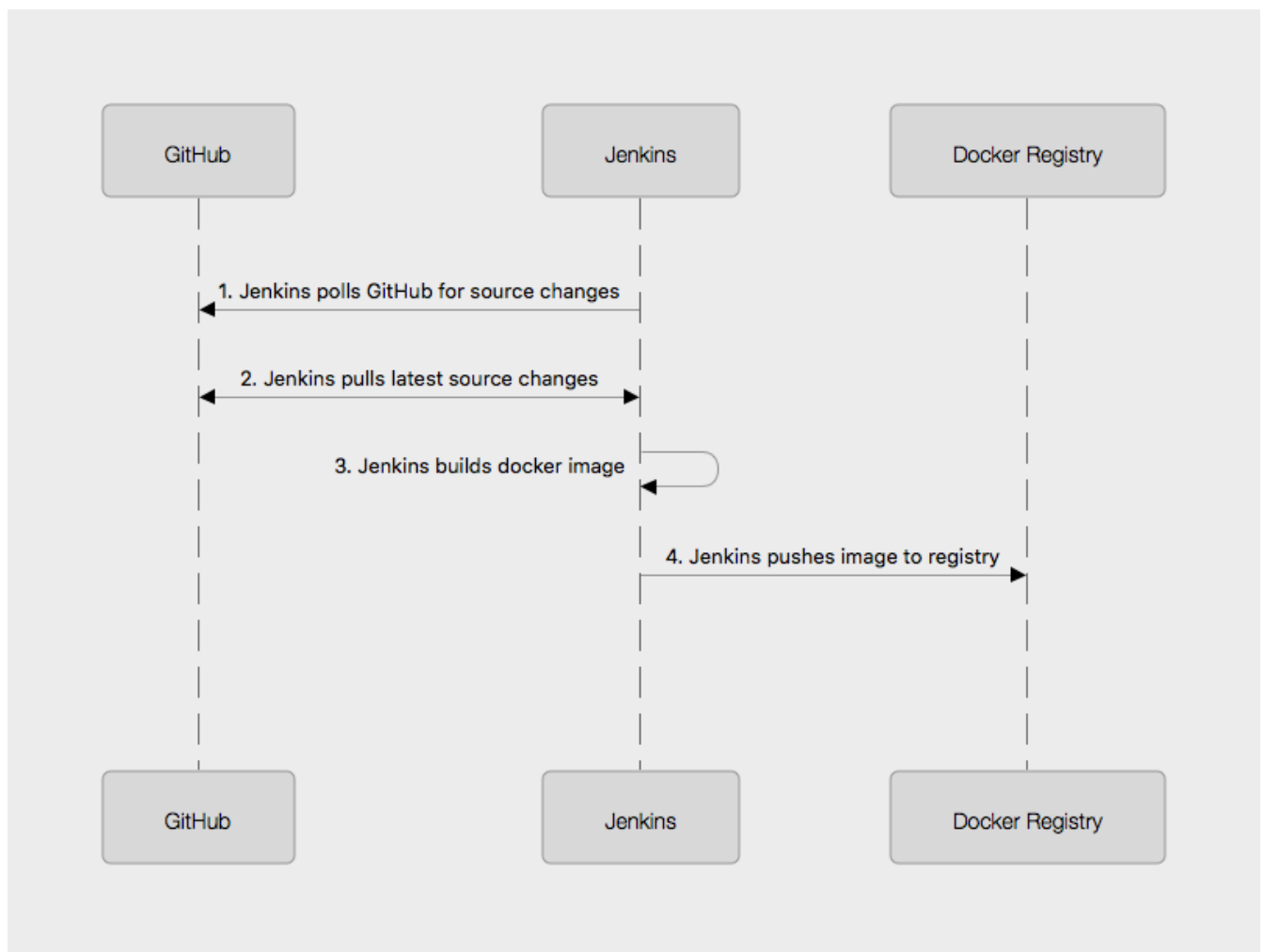
# Prerequisites

Students will need a computer with:

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

## Lab 1

Lab 1 will cause a new docker image to be built. Jenkins will poll GitHub (1 below). If there has been a source update (or if someone manually builds the pipeline), then Jenkins will pull the latest source (2 below) and build the image (3 below). Once built, the image is then pushed to the docker registry (4 below).



### 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 lab1
```

### Step 4: Update the Jenkinsfile

- Go to the home directory of your new repo

```
cd 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 ~/lab1
```

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

```
git add .
```

- Next, commit the changes.

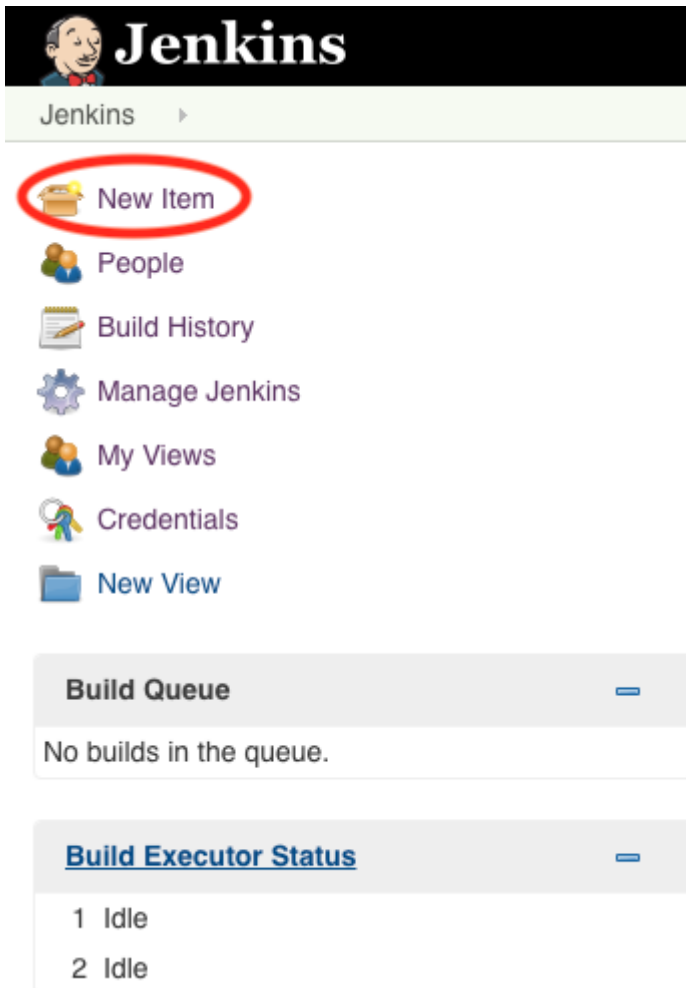
```
git commit -m "Updated Jenkinsfile"
```

- 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



- 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 (with 1 space between each) 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

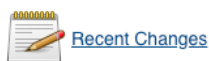
Lightweight checkout ☒

[Pipeline Syntax](#)

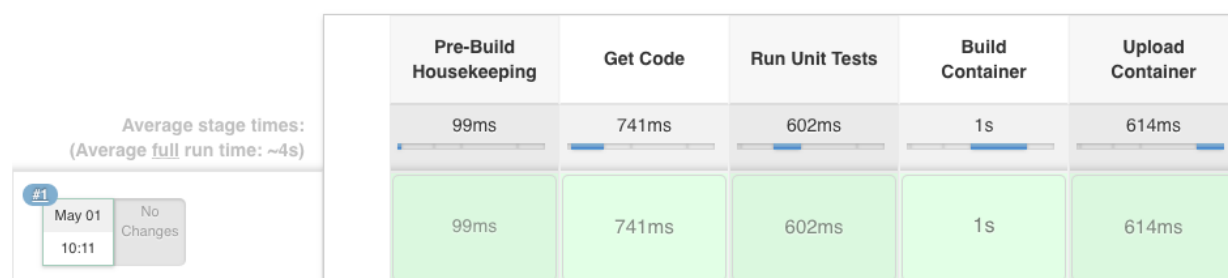
Save Apply

- At this point, you should see your pipeline run automatically. If not, click on **Build Now** on the upper left of the screen.
- After the build, click on your **trainingX\_lab1** link on the dashboard. You should see output something like this:

## Pipeline training1\_lab1



### Stage View



## Step 7: Verify your docker image uploaded to the registry

- Create a new pipeline by clicking on **New Item** on the Dashboard

TIP | You can always get to the Dashboard by clicking on the word **Jenkins** in the upper left of the web page

- Name your new pipeline **trainingX\_lab1\_check** (where 'X' is your training ID number)
- Select **Pipeline** option.
- Click **OK**
- Scroll to the bottom of the next screen until you get to the **Pipeline** section
- Leave the **Definition** field set to **Pipeline script**
- Paste (or write) the following in the **Script** field

```
node {  
  sh 'curl http://registry.roundtower.io:5000/v2/training99/lab1/tags/list'  
}
```

- Be sure to change **training99** to your training ID

**Pipeline**

Definition Pipeline script

Script

```
1 node {  
2   sh 'curl http://registry.roundtower.io:5000/v2/training99/lab1/tags/list'  
3 }
```

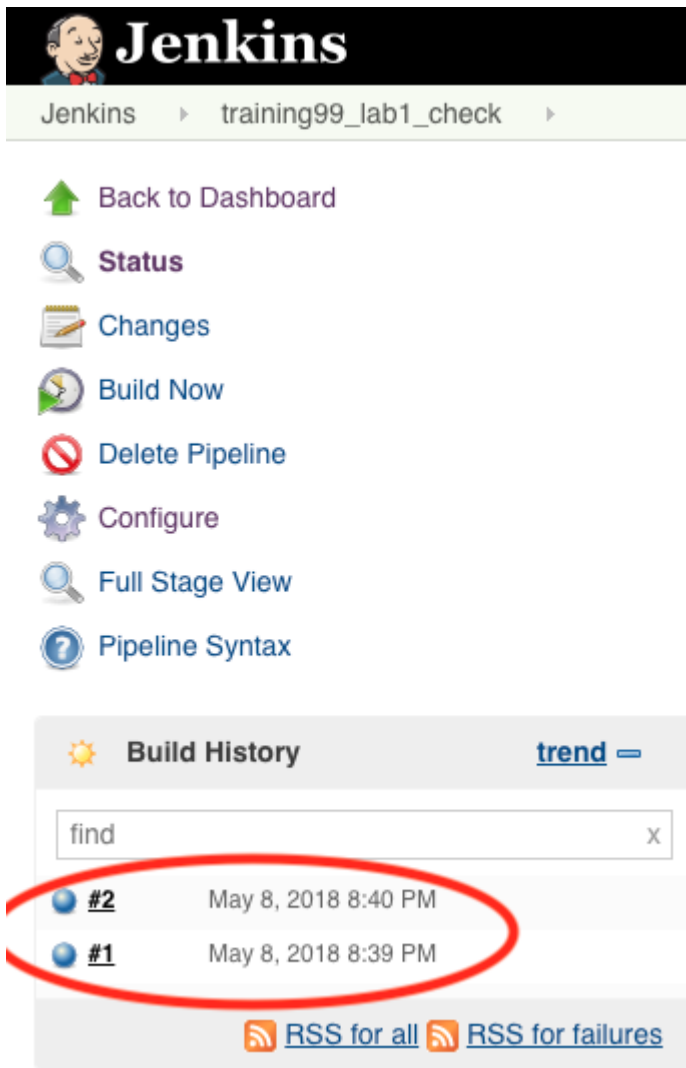
☒ Use Groovy Sandbox

[Pipeline Syntax](#)

Save Apply

- Click the **Save** button to exit
- Click on the **Build Now** button on the left.
- Once the build completes, click on the build number in the **Build History** on the lower left of the

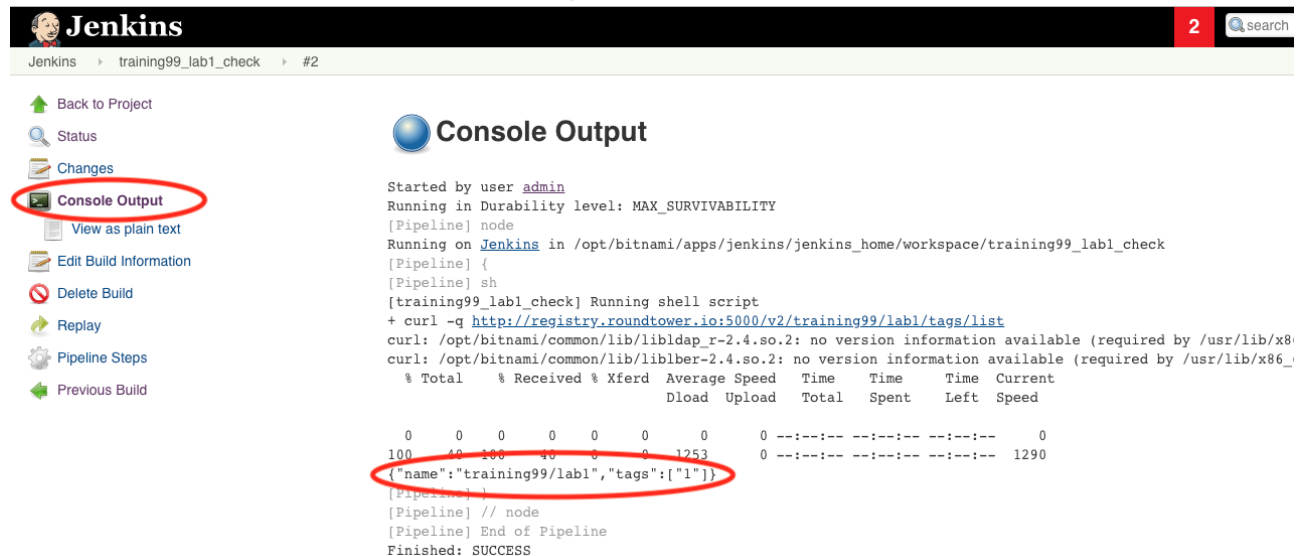




The Jenkins dashboard for the 'training99\_lab1\_check' pipeline shows a list of actions on the left: Back to Dashboard, Status, Changes, Build Now, Delete Pipeline, Configure, Full Stage View, and Pipeline Syntax. The main section displays the 'Build History' for this pipeline, with a search bar containing 'find'. Two builds are listed: #2 (May 8, 2018 8:40 PM) and #1 (May 8, 2018 8:39 PM). Both builds are circled in red. At the bottom of the build history, there are links for 'RSS for all' and 'RSS for failures'.

screen.

- Click on the **Console Output** on the left of the screen



The Jenkins console output for build #2 of the 'training99\_lab1\_check' pipeline is displayed. The left sidebar shows the 'Console Output' link circled in red. The main area shows the console output, which includes the following text:

```

Started by user admin
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] node
Running on Jenkins in /opt/bitnami/apps/jenkins/jenkins_home/workspace/training99_lab1_check
[Pipeline] {
[Pipeline] sh
[training99_lab1_check] Running shell script
+ curl -q http://registry.roundtower.io:5000/v2/training99/lab1/tags/list
curl: /opt/bitnami/common/lib/libldap_r-2.4.so.2: no version information available (required by /usr/lib/x86_64-linux-gnu/libcurl.so.4)
curl: /opt/bitnami/common/lib/liblber-2.4.so.2: no version information available (required by /usr/lib/x86_64-linux-gnu/libcurl.so.4)
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
  0     0    0     0    0     0     0      0  0 --:--:-- --:--:-- --:--:--    0
100  1253  100  1253    0     0    1253      0  0 --:--:-- --:--:-- --:--:--  1290
{"name": "training99/lab1", "tags": ["1"]}
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS

```

The output shows a successful curl command that retrieved a JSON response from the Roundtower registry. The JSON response is circled in red: `{"name": "training99/lab1", "tags": ["1"]}`.

- You should see in the output provided something like the following with your training ID in it.

```
Started by user admin
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] node
Running on Jenkins in
/opt/bitnami/apps/jenkins/jenkins_home/workspace/training99_lab1_check
[Pipeline] sh
[training99_lab1_check] Running shell script
...
{"name":"training99/lab1","tags":["1"]}
...
Finished: SUCCESS
```

- The line `{"name":"training99/lab1","tags":["1"]}` means that version 1 of training99/lab1 is in the registry. This verifies your docker image uploaded properly.