# Lab 1 – Jenkins intro

## Task – Install and setup Jenkins

Prerequisite – EC2 (t2.micro) with ports 22, 80, 8080 open to the internet.

1) SSH connect to your EC2 you have generated
2) Run the following commands within the EC2

```
sudo apt update
sudo apt install openjdk-11-jre
curl https://get.docker.com | sudo bash
sudo usermod -aG docker $(whoami) curl -fsSL https://pkg.jenkins.io/debian-
stable/jenkins.io.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

3) Using your web browser go to <public IP>:8080 which is the Jenkins GUI
4) Copy the initial admin password which is accessed via below and enter it in the GUI

```
sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

5) Install the suggested plugins and create an admin user (no need to use actual email, purpose is to recover account if you forget password)

# Lab 2 – Jenkins Builds

## Task – Basic Jenkins commands

Using a Jenkins freestyle project create a build which does the following:

- Prints out all files in location (including hidden ones)
- Creates a file called "coolScript.sh"
- Add an echo command to the coolScript.sh
- Get Jenkins to run the coolScript.sh
- Archive the coolScript.sh using a post build action

Find your artefact in the jobs directory within the EC2 Jenkins files.

# Lab 3 – Jenkins Pipeline Basic

## Task – Create Jenkins Pipeline with basic stages

Prerequisites – EC2 with Jenkins installed

Using a JenkinsFile create a Pipeline through the Jenkins GUI that simulates the standard Build, Test, Deploy stages using echo, ls, pwd, touch, mv commands.

A basic Jenkinsfile configuration can be found below:

```
pipeline {
    agent any
    stages {
        stage('Pipeline Stages'){
            steps {
                sh "ls"
            }
        }
        stage('second stage'){
            steps {
                sh "pwd"
            }
        }
```

The Jenkinsfile should be stored in a public repo connected to the pipeline.

Stretch goal – Use Jenkins to deploy the project from Task 1
https://gitlab.com/Reece-Elder/dockerfileexercise

# Lab 4 – Jenkins Project

## Task – Deploy a Node Project using Jenkins

Prerequisites – EC2 (at least of size t2.small) with Jenkins and Docker installed (or a 2nd EC2 that the Jenkins EC2 can ssh to that has Docker installed)

Create a pipeline that Builds and deploys the app at this repo https://gitlab.com/Reece-Elder/devops-m5-nodeproject. You will need to create the Dockerfile and use stages to build and deploy the app. You will likely need to research how to build a NodeJS application

# Lab 5 – Jenkins Credentials

## Task – Add credentials to Jenkins for security

Prerequisite – EC2 (t2.small) with ports 22 and 80 open to the internet with Jenkins installed.

1) Create a new repo on GitHub / GitLab that is private and keep note of the credentials to access the repo (username, password, SSH key, Personal Access Token etc.)
2) Create a JenkinsFile within the private repo that builds an nginx container with a custom nginx.conf that returns “Hello Jenkins!” and at some point prints out an env variable `echo $SECRET_VAR`
3) Through the Jenkins GUI go to ‘Credentials’ and add your Git Username and password as a ‘username with password’ credential.
4) Create a Secret Text credential with any value through the Jenkins GUI
5) Create a new Pipeline that pulls the project from the private repo and runs (Building the nginx container).
6) Configure the JenkinsFile to create the SECRET_VAR variable from the secret text credential uploaded to the GUI

```
environment {
    SECRET_VAR = credentials('secret_text')
}
```

Stretch goal – Push your docker image up to Docker Hub (Hint: will require Docker login credentials)

# Lab 6 – Jenkins Webhook

## Task – Use Webhook to connect Source Control to a pipeline

Prerequisite – Same machine as Jenkins installation

1) Create a new Git Repo (Public) with a basic JenkinsFile that echoes out “Hello World” via a shell command
2) Create a new Pipeline on Jenkins GUI adding the public Repo. On this configuration under ‘Polling SCM’ check Git Webhook
3) Go back to GitHub / <Your repo> / settings / webhooks / add webhook and set the PayLoad URL to `<jenkins-ip-address>/github-webhook/` ensuring the final / is added
4) Check the webhook is accepted (should show a green tick via the GitHub GUI) and make a change to the repo
5) Check the Jenkins pipeline has been triggered with a change made to the repo

Stretch goal – Create or use existing Git Repos for Task 1 and 2 of the repo https://gitlab.com/Reece-Elder/dockerfileexercise, use webhooks and make Jenkins pipeline builds to connect up. Add stages to build, deploy and push images to DockerHub, using the webhooks to test the pipeline