CI Tutorials

Complete each section with activity

1: Create Jenkins service users and Install Jenkins Locally (via Windows)

1.a Create a local user for Jenkins  
<https://support.microsoft.com/en-us/windows/create-a-local-user-or-administrator-account-in-windows-20de74e0-ac7f-3502-a866-32915af2a34d>

1.b Make user a “service user” by add permissions to run services

* Open Local Security Policy, windows + R, type secpol.msc, and then press ENTER.
* Security Settings of the console tree, Local Policies -> User Rights Assignment
* Double Click “Log on as a service”
* Add newly created user to permission  
    
  Graphical user interface, text, application

  Description automatically generated

1.c Download and install Jenkins locally (running as service user)

<https://www.jenkins.io/doc/book/installing/windows/>  
  
MAKE SURE YOU USE YOUR SERVICE USER, AND RUN ON PORT 8081  
(will prevent node and java apps complaining someone else is using 8080)

1.d Ensure Jenkins is running

go to website and <http://localhost:8081/>

you can find admin password at   
C:\Users\<SERVICE\_NAME>\AppData\Local\Jenkins\.jenkins\secrets\initialAdminPassword  
  
Graphical user interface, text, application, email

Description automatically generated

1.e Login as initial Admin, create a “admin user” using your riversafe email and complete setup 😊

Graphical user interface, application, Teams

Description automatically generated

**Trouble shooting:**

If you get stuck, don’t worry jump to tutorial 4  
lesson is windows is bad at servers

2: Create helloworld pipeline (via Windows)

Read through   
<https://www.jenkins.io/doc/book/pipeline/>   
  
Look at plugins registry  
<https://plugins.jenkins.io/>

Activities:

Create hello world pipeline with this Jenkinsfile

pipeline {

agent any

stages {

stage('Build') {

steps {

echo 'Hello Build Steps'

sh 'echo bash build steps'

}

}

stage('Test') {

steps {

echo 'Hello Testing steps'

sh 'echo bash deployment steps'

}

}

stage('Deploy') {

steps {

echo 'Hello Deployment steps'  
 sh 'echo bash deployment steps'

}

}

}

}

**Trouble shooting:**

If you get stuck, don’t worry jump to tutorial 4  
lesson is windows is bad at servers

3: Un-install Jenkins, delete Jenkins user 😊 (via Windows)

Read through   
<https://devrant.com/rants/336021/windows-servers-are-a-joke-a-bad-joke-i-feel-sorry-for-people-who-have-to-work-o>

Graphical user interface, text, application

Description automatically generated

As great as Jenkins is, and it can be installed, and code ran on it

Lesson : Window’s is useless at being a server, don’t listen to windows sysops lies  
  
Now lets do the same thing via Docker… which is also harder in windows

lesson is windows is bad at servers

4: Install Jenkins (via Docker)

Read through   
<https://www.jenkins.io/doc/book/installing/docker/>

<https://www.jenkins.io/doc/book/installing/docker/#on-windows>

DIND read through articles  
(aka Docker in Docker, aka only use for CI not normal use)  
<https://blog.nestybox.com/2019/09/14/dind.html>   
<https://shisho.dev/blog/posts/docker-in-docker/>

Activities:

1.a Go through windows installing docker in docker Jenkins steps  
(use **powershell** !)

<https://www.jenkins.io/doc/book/installing/docker/#on-windows>

1. Build network  
   $ docker network create jenkins
2. Go to dev folder and create Jenkins folder   
   $ cd C:\dev\   
    # aka **<INTO WHERE YOU WANT JENKINS FOLDER>**

mkdir jenkins  
 cd jenkins

mkdir jenkins-data

1. Start DIND Server

$ docker run --name jenkins-docker --rm --detach `

--privileged `

--network jenkins `

--network-alias docker `

--env DOCKER\_TLS\_CERTDIR=/certs `

--volume jenkins-docker-certs:/certs/client `

--volume ${PWD}/jenkins-data:/var/jenkins\_home `

--publish 2376:2376 `

docker:dind

1. Create Dockerfile inside dev folder  
   (as per <https://www.jenkins.io/doc/book/installing/docker/#on-windows>)

$ docker build -t local-jenkins .

1. Run Jenkins  
   $ docker run --name jenkins-local --detach `

--network jenkins `  
 --env DOCKER\_HOST=tcp://docker:2376 `

--env DOCKER\_CERT\_PATH=/certs/client `

--env DOCKER\_TLS\_VERIFY=1 `

--volume ${PWD}/jenkins-data:/var/jenkins\_home `

--volume jenkins-docker-certs:/certs/client:ro `

--volume "${HOMEDRIVE}${HOMEPATH}":/home `

--restart=on-failure `

--env JAVA\_OPTS="-Dhudson.plugins.git.GitSCM.ALLOW\_LOCAL\_CHECKOUT=true" `

--publish 8080:8080 --publish 50000:50000 `  
 local-jenkins

1.d Ensure Jenkins is running

go to website and <http://localhost:8080/>

you can find admin password at   
INTO WHERE YOU WANT JENKINS FOLDER\jenkins\_home\secrets\initialAdminPassword  
  
Graphical user interface, text, application, email

Description automatically generated

1.e Login as initial Admin, create a “admin user” using your riversafe email and complete setup 😊

(ps install recommended plugins)

Graphical user interface, application, Teams

Description automatically generated

EXTRA STEPS  
  
STOP JENKINS IN DOCKER

$ docker stop jenkins-docker  
docker stop jenkins-local

DEBUG JENKINS IN DOCKER

$ docker exec -it jenkins-local bash



RESTART JENKINS IN DOCKER

1. # goto directory

$ cd C:\dev\jenkins

# restart dind server

docker run --name jenkins-docker --rm --detach `

--privileged `

--network jenkins `

--network-alias docker `

--env DOCKER\_TLS\_CERTDIR=/certs `

--volume jenkins-docker-certs:/certs/client `

--volume ${PWD}/jenkins-data:/var/jenkins\_home `

--publish 2376:2376 `

docker:dind

# start jenkin server

docker start jenkins-local

5: Create simple Docker pipeline (via Docker)

Read through   
<https://www.jenkins.io/doc/book/using/>

<https://www.jenkins.io/doc/pipeline/tour/hello-world/>   
  
Read through  
<https://www.jenkins.io/doc/book/pipeline/docker/>

<https://plugins.jenkins.io/docker-workflow/>

**Activities:**

Create “Hello-python” Pipeline using docker agent

Jenkinsfile to use

pipeline {

agent {

docker { image 'python:latest' }

}

stages {

stage('Sanity') {

steps {

sh 'python --version'

}

}

}

}

Build Hello-python

Graphical user interface, text, application

Description automatically generated  
  
**Troubleshooting:**

Cannot connect to the Docker daemon at <tcp://docker:2376>. Is the docker daemon running?

**If locally in git-bash, this means check docker service is running, or just restart your computer**

**If inside Jenkins, this means the networked dind isn’t running, run docker ps and make sure it is**

6: Create Fully CI Docker pipeline (via Docker)

Read through   
<https://www.jenkins.io/doc/book/pipeline/docker/>

<https://www.jenkins.io/doc/book/pipeline/syntax/>

Activities:

Create a pipeline called “eze-cli” that:

* Build Trigger is Polling SCM with frequency once a day  
  aka “H 1 \* \* \*”
* Checks out repo  
  <https://github.com/RiverSafeUK/eze-cli>
* Install dependencies
* Runs unit tests

Pipeline Script

pipeline {

agent {

docker { image 'python:latest' }

}

stages {

stage("Set Up") {

steps {

git branch: 'develop', url: 'https://github.com/RiverSafeUK/eze-cli'

}

}

stage("Build") {

steps {

echo 'Build Steps'

sh '''

python -m venv .venv

. .venv/bin/activate

pip install -r requirements.txt

'''

}

}

stage("Test") {

steps {

echo 'Test Steps'

sh '''

python -m venv .venv

. .venv/bin/activate

pip install -r requirements-dev.txt

pytest tests -vv --cov=eze --cov-branch --cov-report=term-missing --cov-report=xml:reports/coverage/coverage.xml --junitxml=reports/xunit/test-results.xml -o junit\_family=xunit1 || true

'''

}

}

stage("Deploy") {

steps {

echo 'Placeholder Deploy Steps'

}

}

}

}

7: Display test results in Jenkins (via Docker)

Read through up on Jenkins Plugins

<https://plugins.jenkins.io/>   
<https://plugins.jenkins.io/xunit/>   
<https://plugins.jenkins.io/code-coverage-api/>  
<https://plugins.jenkins.io/cobertura>  
   
Activities:  
Install xunit and “Cobertura” locally on your Jenkins instance, then restart jenkins  
<http://localhost:8080/pluginManager/available>

Add stage after Testing to publish results  
  
 stage('Publish Tests') {

steps {

junit skipPublishingChecks: true, testResults: 'reports/xunit/test-results.xml'

cobertura autoUpdateHealth: false, autoUpdateStability: false, coberturaReportFile: 'reports/coverage/coverage.xml', conditionalCoverageTargets: '70, 0, 0', failUnhealthy: false, failUnstable: false, lineCoverageTargets: '80, 0, 0', maxNumberOfBuilds: 0, methodCoverageTargets: '80, 0, 0', onlyStable: false, sourceEncoding: 'ASCII', zoomCoverageChart: false

}

}

Look at results

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, application

Description automatically generated

8: CI Best Practices

Read best practice guides  
<https://www.jenkins.io/doc/book/pipeline/pipeline-best-practices/>

<https://github.com/jenkinsci/pipeline-examples/blob/master/docs/BEST_PRACTICES.md>

9: Fib pipeline (via Docker)

Activities:

Create a pipeline for your fib docker image:

* Make it scan for changes to git
* check out the git repo on pipeline start
* build docker image
* push docker image into dockerhub