Software Engineering and Architecture
Lecture 4: Continuous Delivery (2)

Olivier Liechti
olivier.liechti@heig-vd.ch

MASTER OF SCIENCE
IN ENGINEERING

Today.

OO Reengineering Patterns (5 x 10’)
15h - 16h

Second Jenkins demo & time to implement
16h - 17h20

Week	Theory	OO Reengineering	Practice
#1	Agile, Scrum		Intro to Docker
#2	Software evolution	Introduction	Specify and implement a micro-service
#3	Continuous delivery (1)	*Setting Directions*	Intro to Jenkins & Travis
#4	Continuous delivery (2)	*First Contact*	Our first build pipeline
#5	Agile testing (1)	*Initial Understanding*	Intro to Cucumber
#6	Agile testing (2)	*Detailed Model Capture*	Add tests to pipeline
#7	Agile metrics	*Tests: Your Life Insurance!*	Add Sonar to pipeline
#8	Continuous improvement	*Migration Strategies I*	Add non-functional tests
#9	Wrap-up	*Migration Strategies II*	

Back to the CI/CD pipeline

The Deployment Pipeline

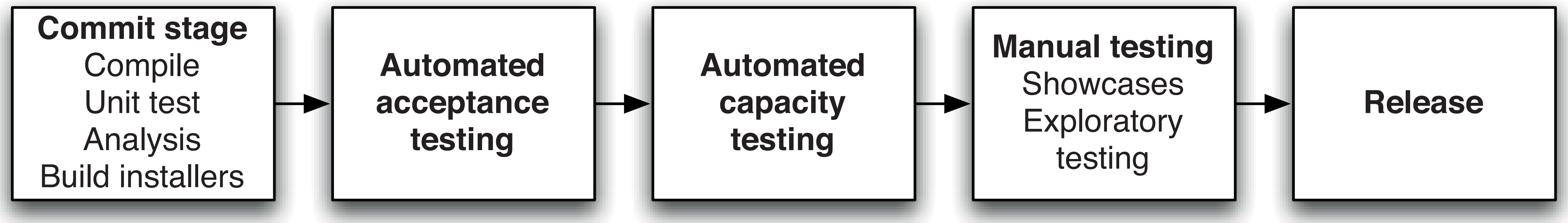


Figure 1.1 *The deployment pipeline*

Tools

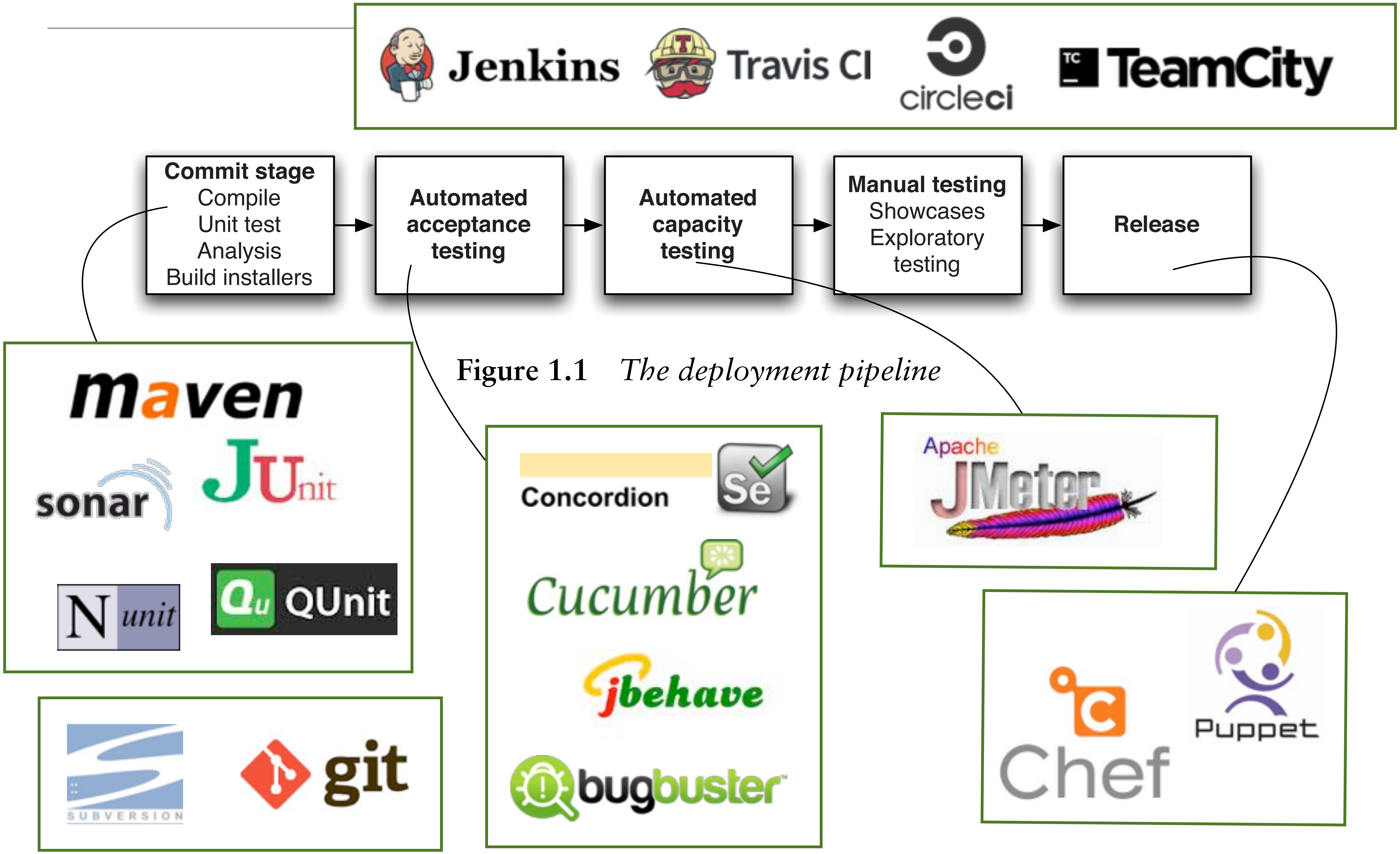


Figure 1.1 *The deployment pipeline*

What have we done so far?

- ✓ Use Spring Boot to implement a micro-service
  - ✓ Use Spring MVC to implement a REST API
  - ✓ Use Spring Data to access a database
  - ✓ Use maven to build the micro-service and generate an executable .jar file
- ✓ Use Docker to deploy the micro-service “in prod”
  - ✓ Build a Docker image
  - ✓ Define a service topology with Docker Compose

What have we done so far?

- ✓ Make basic experiments with Jenkins
  - ✓ Use a Docker image (not a deprecated one!)
  - ✓ Understand that there are a lot of plugins
  - ✓ Understand that Jenkins can automatically install some of the tools during the build, including maven
  - ✓ Define a first “pipeline” (which Jenkins used to call “Job” or “Item” in the UI), by applying the free-form job type.
  - ✓ Get source code out of git, ask maven to build it and find executable .jar in the Jenkins workspace

Next steps

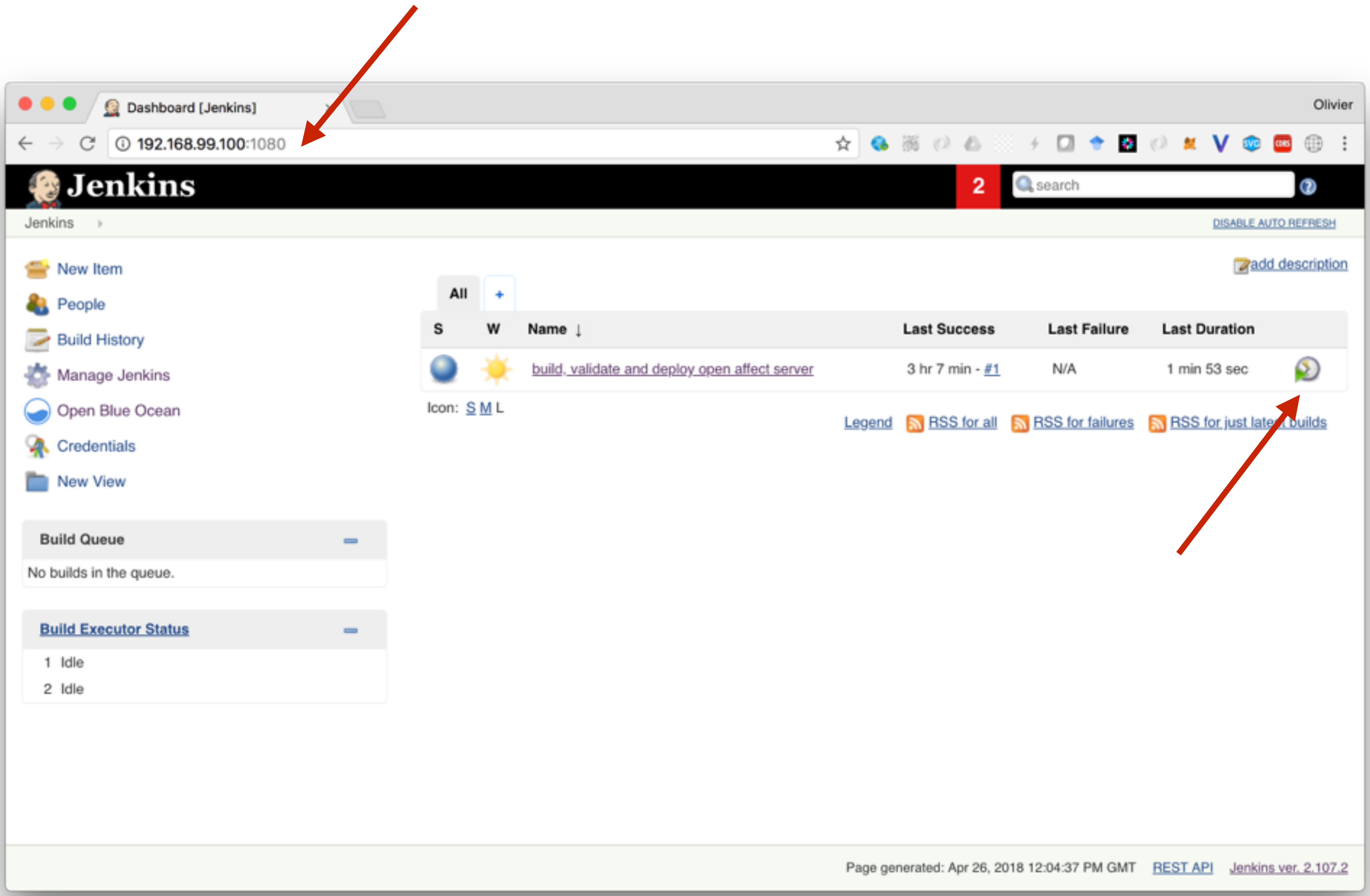
- ☐ Understand that Jenkins has introduced a new type of Jobs, which are those called “Pipelines” in the UI
- ☐ Understand that in this case, we can describe the pipelines in special files (Jenkinsfile), which can be easily put under version control (e.g. in git).
- ☐ Create our first “Pipeline” pipeline
  - ☐ See that we can use Docker in the pipeline
  - ☐ Understand that there are different ways to achieve the same result. Logic can be written in Jenkins and shell scripts (I prefer to keep the Jenkins part simple)

Let’s start with a demo…

```
git clone https://github.com/openaffect/openaffect-server.git
```

```
cd /openaffect-server/docker-topologies/cdpipeline
```

```
docker-compose up -d
docker ps
```



Dashboard [Jenkins]

192.168.99.100:1080

Jenkins

Jenkins › DISABLE AUTO REFRESH

- New Item
- People
- Build History
- Manage Jenkins
- Open Blue Ocean
- Credentials
- New View

add description

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
●	☀	build, validate and deploy open affect server	3 hr 7 min - #1	N/A	1 min 53 sec

Icon: S M L

Legend RSS for all RSS for failures RSS for just latest builds

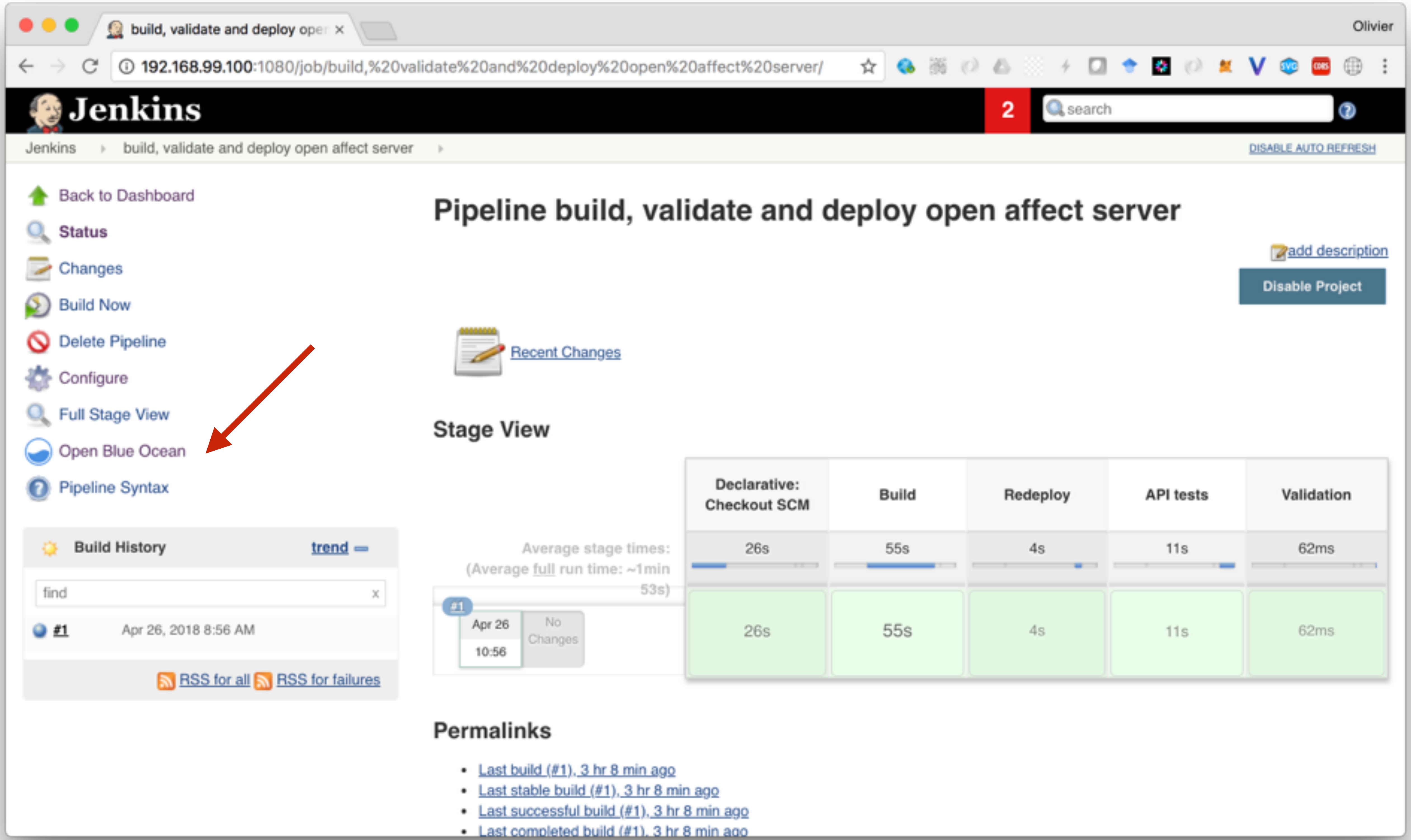
Build Queue

No builds in the queue.

Build Executor Status

1 Idle
2 Idle

Page generated: Apr 26, 2018 12:04:37 PM GMT REST API Jenkins ver. 2.107.2



Pipeline build, validate and deploy open affect server

Stage View

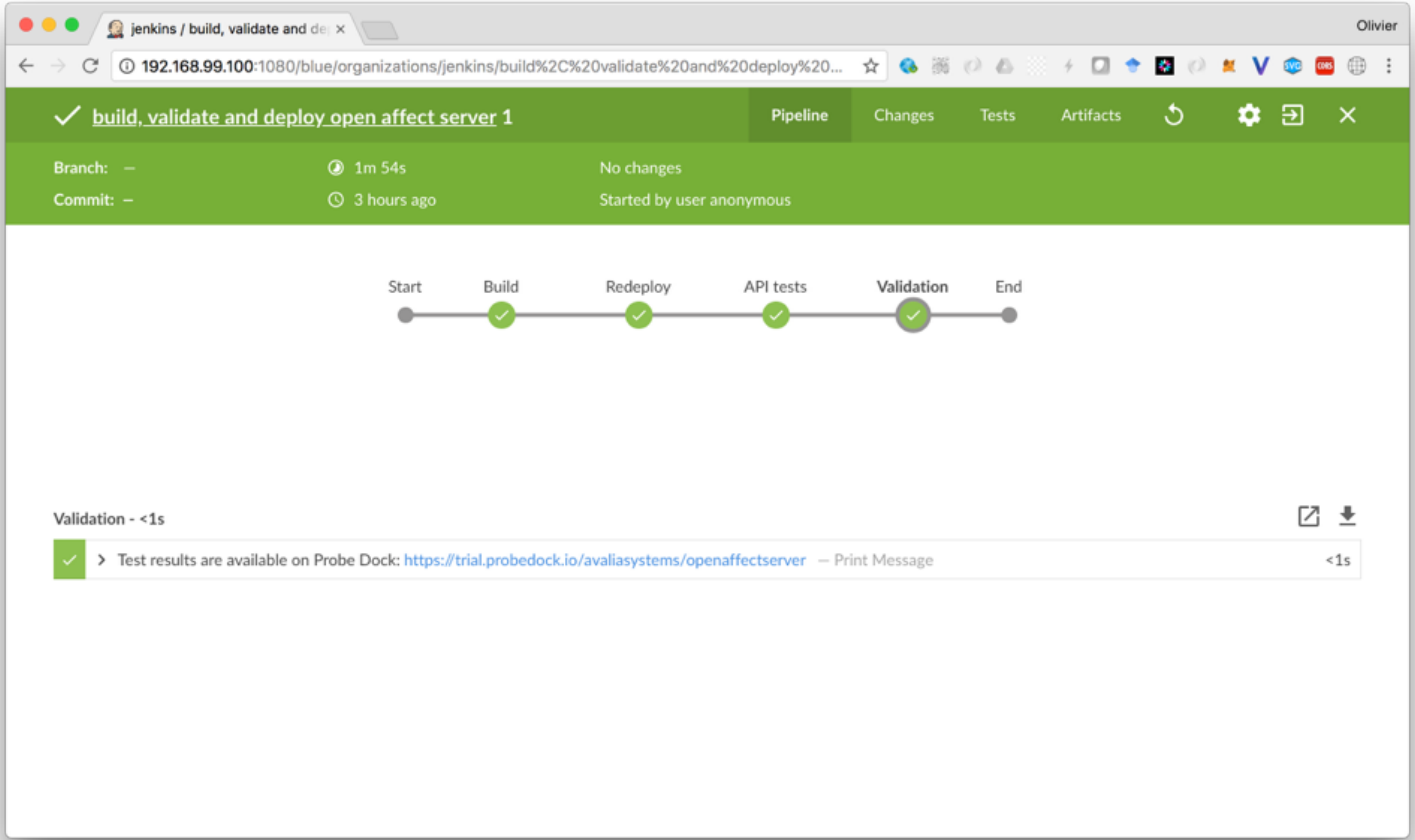
	Declarative: Checkout SCM	Build	Redeploy	API tests	Validation
Average stage times: (Average full run time: ~1min 53s)	26s	55s	4s	11s	62ms
#1 Apr 26 10:56 No Changes	26s	55s	4s	11s	62ms

Build History: #1 Apr 26, 2018 8:56 AM

Permalinks

- Last build (#1), 3 hr 8 min ago
- Last stable build (#1), 3 hr 8 min ago
- Last successful build (#1), 3 hr 8 min ago
- Last completed build (#1), 3 hr 8 min ago

MSE MASTER OF SCIENCE IN ENGINEERING

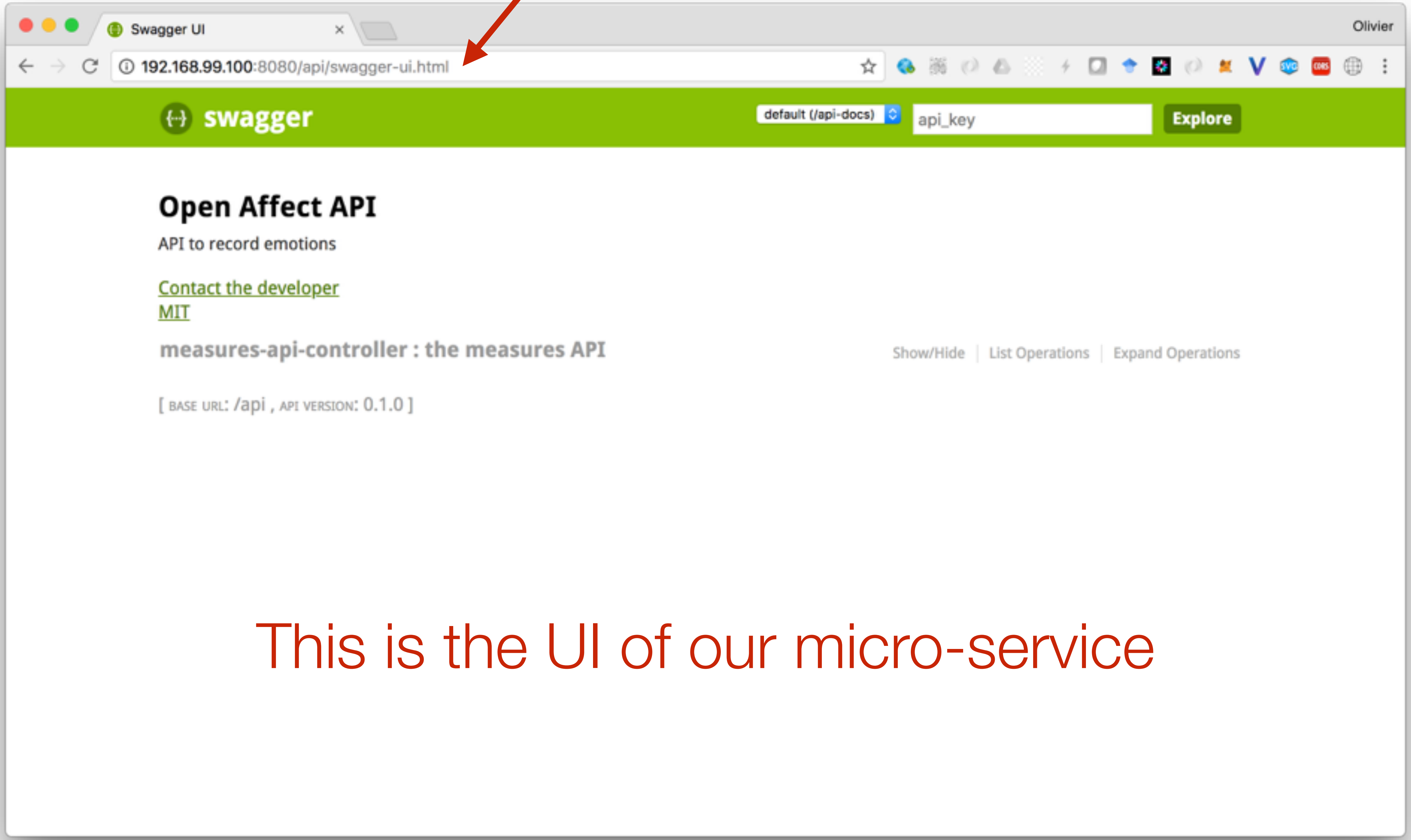

jenkins / build, validate and de
Olivier
192.168.99.100:1080/blue/organizations/jenkins/build%2C%20validate%20and%20deploy%20...
Pipeline
Changes
Tests
Artifacts
Start
Build
Redeploy
API tests
Validation
End

build, validate and deploy open affect server 1

Branch: –
Commit: –
1m 54s
3 hours ago
No changes
Started by user anonymous

Validation - <1s

✓ Test results are available on Probe Dock: https://trial.probedock.io/avaliasystems/openaffectserver — Print Message <1s



192.168.99.100:8080/api/swagger-ui.html

Open Affect API

API to record emotions

Contact the developer

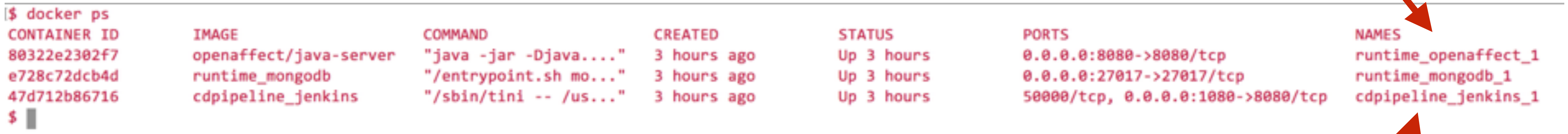
MIT

measures-api-controller : the measures API

[BASE URL: /api , API VERSION: 0.1.0]

This is the UI of our micro-service

The execution of the pipeline has "docker-compose upped" a topology with our micro-service and its database



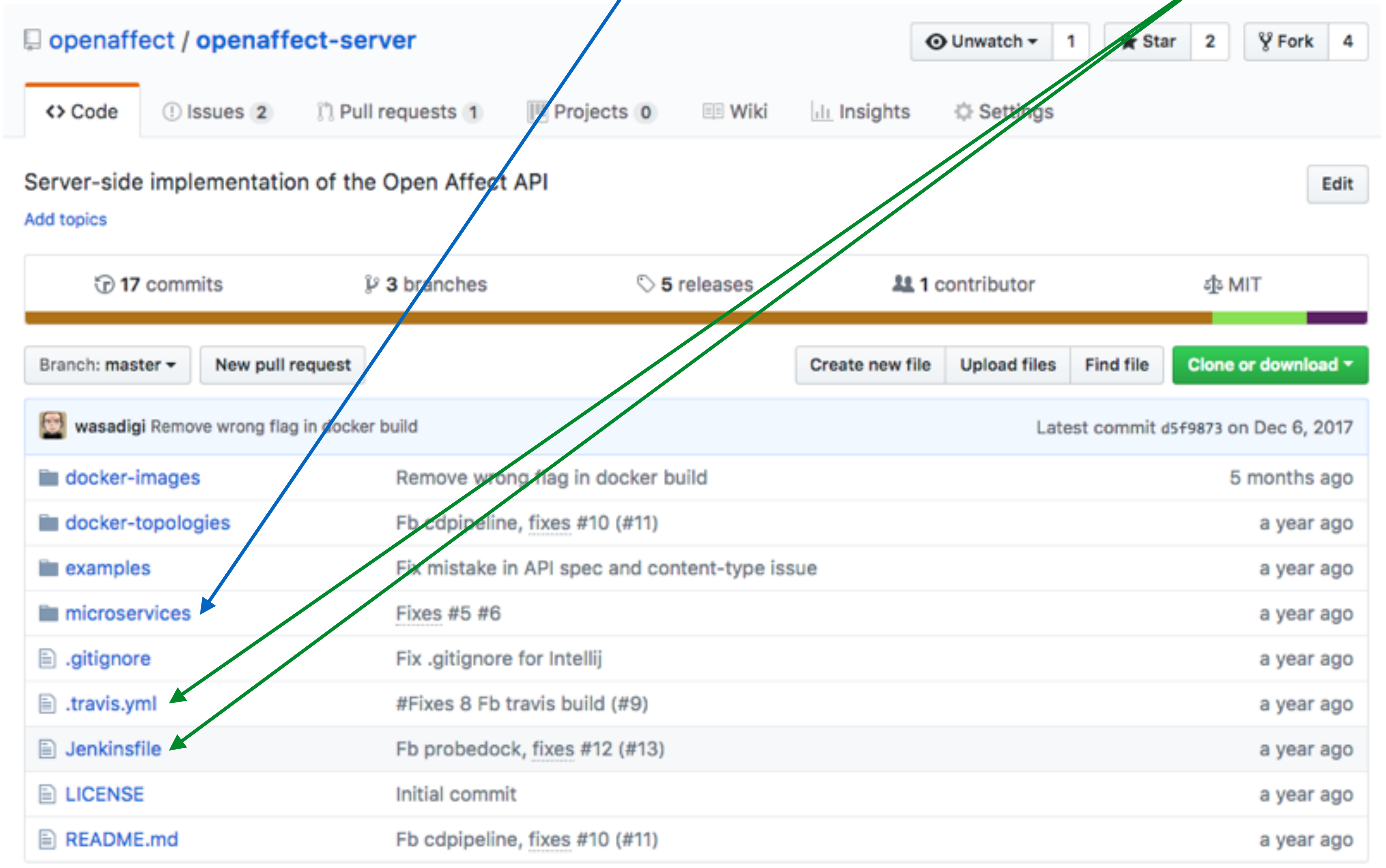
```
$ docker ps
CONTAINER ID        IMAGE                    COMMAND                  CREATED             STATUS              PORTS                                 NAMES
80322e2302f7        openaffect/java-server   "java -jar -Djava...."   3 hours ago         Up 3 hours          0.0.0.0:8080->8080/tcp                runtime_openaffect_1
e728c72dcb4d        runtime_mongodb          "/entrypoint.sh mo..."   3 hours ago         Up 3 hours          0.0.0.0:27017->27017/tcp              runtime_mongodb_1
47d712b86716        cdpipeline_jenkins       "/sbin/tini -- /us..."   3 hours ago         Up 3 hours          50000/tcp, 0.0.0.0:1080->8080/tcp     cdpipeline_jenkins_1
$ 
```

We have a "cdpipeline" topology with one Jenkins container

MASTER OF SCIENCE
IN ENGINEERING

Let’s see how it is built...

This repo contains both 1) our codebase and 2) our build pipelines



openaffect / openaffect-server

Unwatch 1 | Star 2 | Fork 4

Code | Issues 2 | Pull requests 1 | Projects 0 | Wiki | Insights | Settings

Server-side implementation of the Open Affect API

Add topics

17 commits | 3 branches | 5 releases | 1 contributor | MIT

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

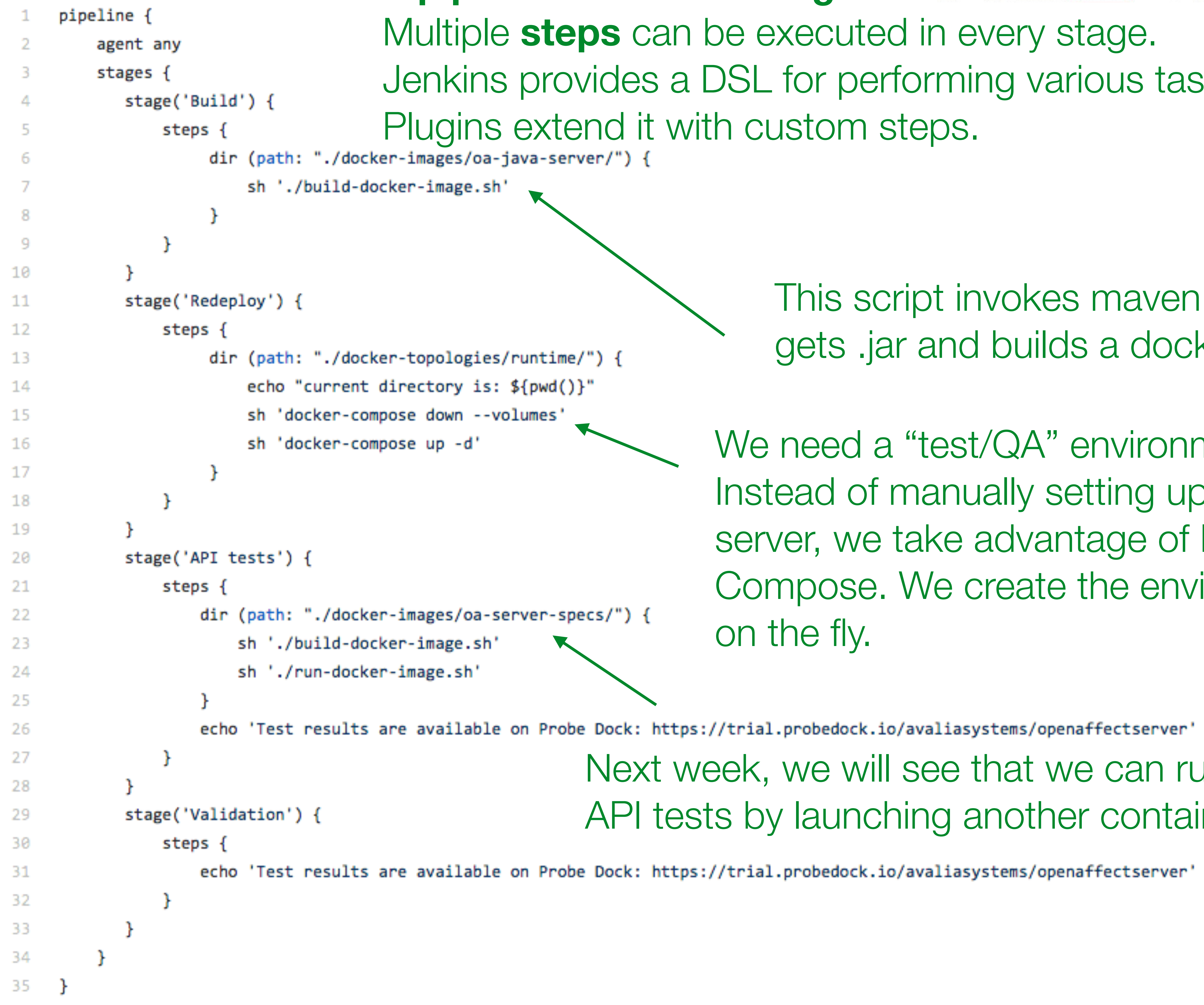
wasadigi Remove wrong flag in docker build — Latest commit d5f9873 on Dec 6, 2017

docker-images	Remove wrong flag in docker build	5 months ago
docker-topologies	Fb cdpipeline, fixes #10 (#11)	a year ago
examples	Fix mistake in API spec and content-type issue	a year ago
microservices	Fixes #5 #6	a year ago
.gitignore	Fix .gitignore for Intellij	a year ago
.travis.yml	#Fixes 8 Fb travis build (#9)	a year ago
Jenkinsfile	Fb probedock, fixes #12 (#13)	a year ago
LICENSE	Initial commit	a year ago
README.md	Fb cdpipeline, fixes #10 (#11)	a year ago

When we ask Jenkins to execute our pipeline, it will 1) clone/pull a git repo, 2) find a Jenkinsfile, 3) interpret it.

A **pipeline** is made of **stages**.
Multiple **steps** can be executed in every stage.
Jenkins provides a DSL for performing various tasks.
Plugins extend it with custom steps.

ASTER OF SCIENCE
ENGINEERING

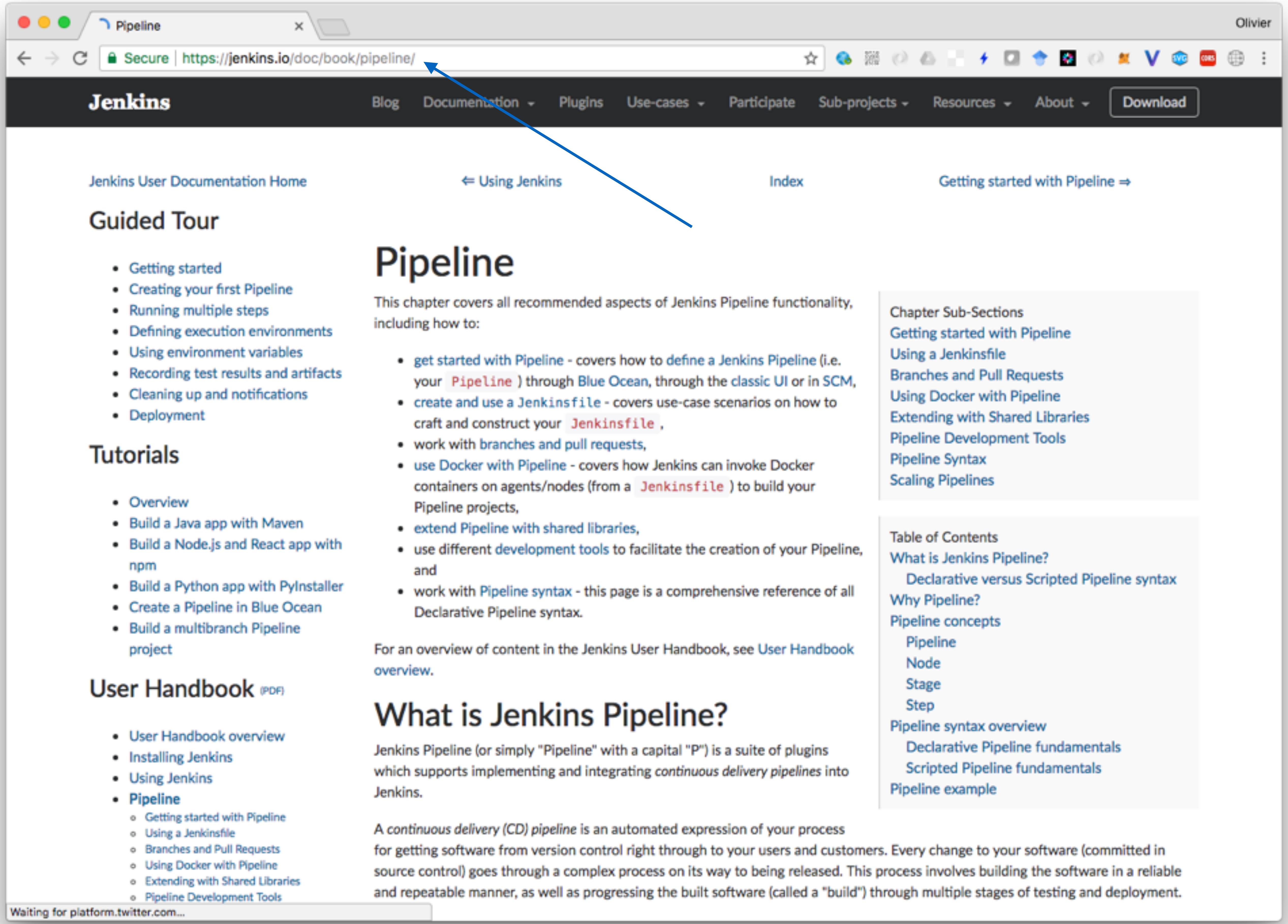


```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                dir (path: "./docker-images/oa-java-server/") {
                    sh './build-docker-image.sh'
                }
            }
        }
        stage('Redeploy') {
            steps {
                dir (path: "./docker-topologies/runtime/") {
                    echo "current directory is: ${pwd()}"
                    sh 'docker-compose down --volumes'
                    sh 'docker-compose up -d'
                }
            }
        }
        stage('API tests') {
            steps {
                dir (path: "./docker-images/oa-server-specs/") {
                    sh './build-docker-image.sh'
                    sh './run-docker-image.sh'
                }
                echo 'Test results are available on Probe Dock: https://trial.probedock.io/avaliasystems/openaffectserver'
            }
        }
        stage('Validation') {
            steps {
                echo 'Test results are available on Probe Dock: https://trial.probedock.io/avaliasystems/openaffectserver'
            }
        }
    }
}
```

This script invokes maven,
gets .jar and builds a docker image

We need a “test/QA” environment. Instead of manually setting up a server, we take advantage of Docker Compose. We create the environment on the fly.

Next week, we will see that we can run API tests by launching another container


Pipeline

Olivier

Secure https://jenkins.io/doc/book/pipeline/

Jenkins Blog Documentation Plugins Use-cases Participate Sub-projects Resources About Download

Jenkins User Documentation Home ⇐ Using Jenkins Index Getting started with Pipeline ⇒

Guided Tour

- Getting started
- Creating your first Pipeline
- Running multiple steps
- Defining execution environments
- Using environment variables
- Recording test results and artifacts
- Cleaning up and notifications
- Deployment

Tutorials

- Overview
- Build a Java app with Maven
- Build a Node.js and React app with npm
- Build a Python app with PyInstaller
- Create a Pipeline in Blue Ocean
- Build a multibranch Pipeline project

User Handbook (PDF)

- User Handbook overview
- Installing Jenkins
- Using Jenkins
- Pipeline
  - Getting started with Pipeline
  - Using a Jenkinsfile
  - Branches and Pull Requests
  - Using Docker with Pipeline
  - Extending with Shared Libraries
  - Pipeline Development Tools

Pipeline

This chapter covers all recommended aspects of Jenkins Pipeline functionality, including how to:

- get started with Pipeline - covers how to define a Jenkins Pipeline (i.e. your `Pipeline`) through Blue Ocean, through the classic UI or in SCM,
- create and use a `Jenkinsfile` - covers use-case scenarios on how to craft and construct your `Jenkinsfile`,
- work with branches and pull requests,
- use Docker with Pipeline - covers how Jenkins can invoke Docker containers on agents/nodes (from a `Jenkinsfile`) to build your Pipeline projects,
- extend Pipeline with shared libraries,
- use different development tools to facilitate the creation of your Pipeline, and
- work with Pipeline syntax - this page is a comprehensive reference of all Declarative Pipeline syntax.

For an overview of content in the Jenkins User Handbook, see User Handbook overview.

What is Jenkins Pipeline?

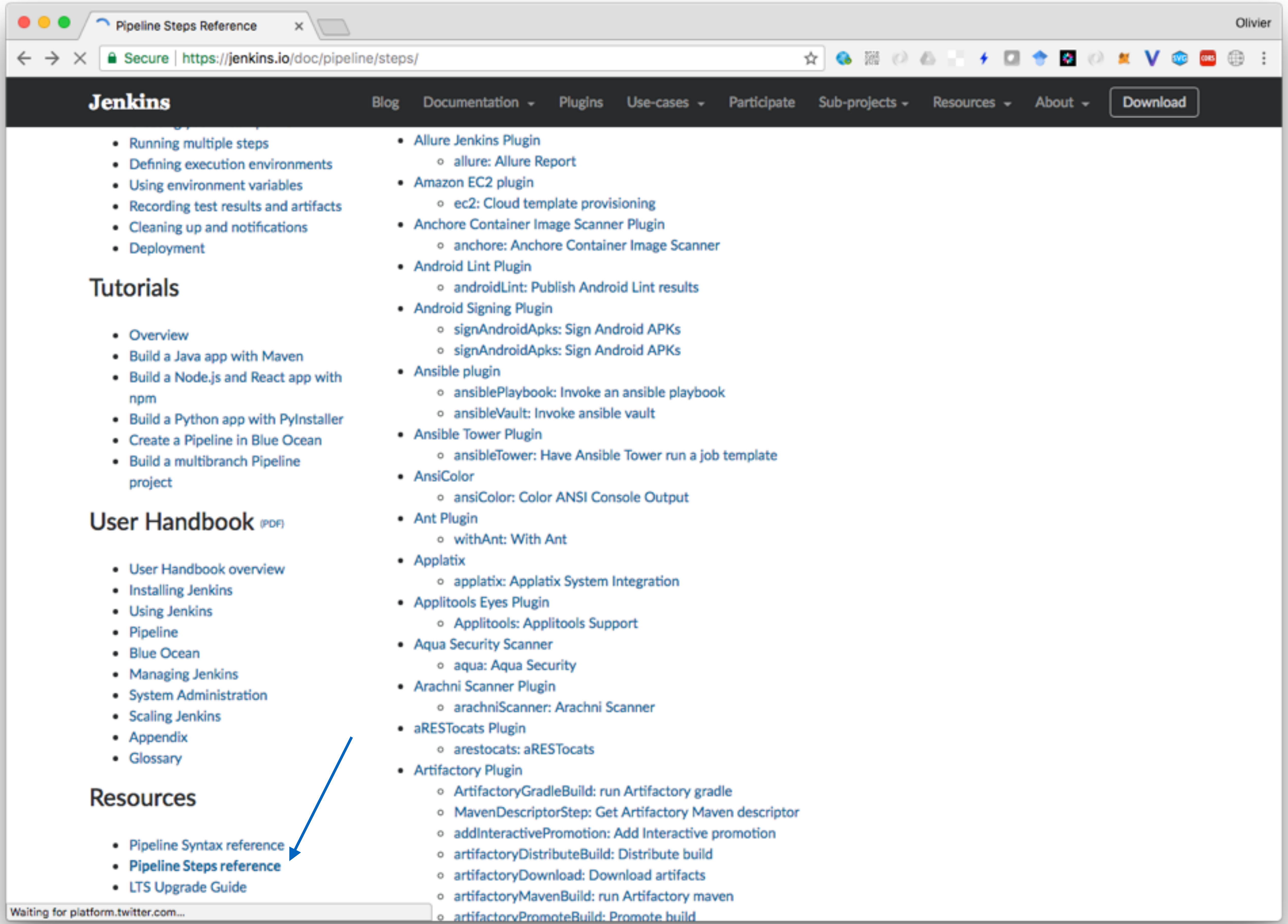
Jenkins Pipeline (or simply "Pipeline" with a capital "P") is a suite of plugins which supports implementing and integrating *continuous delivery pipelines* into Jenkins.

A *continuous delivery (CD) pipeline* is an automated expression of your process for getting software from version control right through to your users and customers. Every change to your software (committed in source control) goes through a complex process on its way to being released. This process involves building the software in a reliable and repeatable manner, as well as progressing the built software (called a "build") through multiple stages of testing and deployment.

Chapter Sub-Sections
- Getting started with Pipeline
- Using a Jenkinsfile
- Branches and Pull Requests
- Using Docker with Pipeline
- Extending with Shared Libraries
- Pipeline Development Tools
- Pipeline Syntax
- Scaling Pipelines

Table of Contents
- What is Jenkins Pipeline?
  - Declarative versus Scripted Pipeline syntax
- Why Pipeline?
- Pipeline concepts
  - Pipeline
  - Node
  - Stage
  - Step
- Pipeline syntax overview
  - Declarative Pipeline fundamentals
  - Scripted Pipeline fundamentals
- Pipeline example

Waiting for platform.twitter.com...


Pipeline Steps Reference

https://jenkins.io/doc/pipeline/steps/

Jenkins

Blog Documentation Plugins Use-cases Participate Sub-projects Resources About Download

- Running multiple steps
- Defining execution environments
- Using environment variables
- Recording test results and artifacts
- Cleaning up and notifications
- Deployment

Tutorials

- Overview
- Build a Java app with Maven
- Build a Node.js and React app with npm
- Build a Python app with PyInstaller
- Create a Pipeline in Blue Ocean
- Build a multibranch Pipeline project

User Handbook (PDF)

- User Handbook overview
- Installing Jenkins
- Using Jenkins
- Pipeline
- Blue Ocean
- Managing Jenkins
- System Administration
- Scaling Jenkins
- Appendix
- Glossary

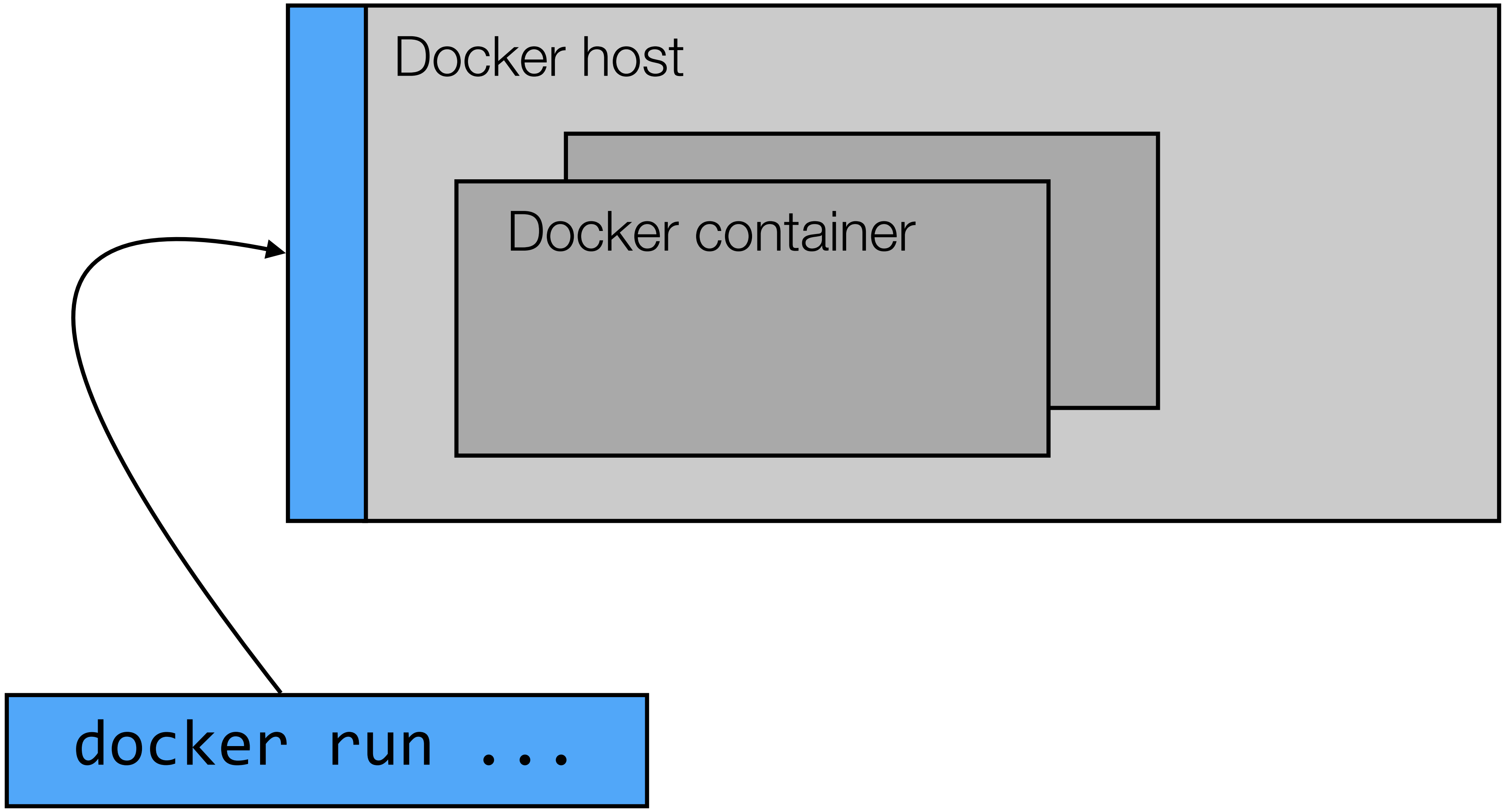
Resources

- Pipeline Syntax reference
- **Pipeline Steps reference**
- LTS Upgrade Guide

- Allure Jenkins Plugin
  - allure: Allure Report
- Amazon EC2 plugin
  - ec2: Cloud template provisioning
- Anchore Container Image Scanner Plugin
  - anchore: Anchore Container Image Scanner
- Android Lint Plugin
  - androidLint: Publish Android Lint results
- Android Signing Plugin
  - signAndroidApks: Sign Android APKs
  - signAndroidApks: Sign Android APKs
- Ansible plugin
  - ansiblePlaybook: Invoke an ansible playbook
  - ansibleVault: Invoke ansible vault
- Ansible Tower Plugin
  - ansibleTower: Have Ansible Tower run a job template
- AnsiColor
  - ansiColor: Color ANSI Console Output
- Ant Plugin
  - withAnt: With Ant
- Applatix
  - applatix: Applatix System Integration
- Applitools Eyes Plugin
  - Applitools: Applitools Support
- Aqua Security Scanner
  - aqua: Aqua Security
- Arachni Scanner Plugin
  - arachniScanner: Arachni Scanner
- aRESTocats Plugin
  - arestocats: aRESTocats
- Artifactory Plugin
  - ArtifactoryGradleBuild: run Artifactory gradle
  - MavenDescriptorStep: Get Artifactory Maven descriptor
  - addInteractivePromotion: Add Interactive promotion
  - artifactoryDistributeBuild: Distribute build
  - artifactoryDownload: Download artifacts
  - artifactoryMavenBuild: run Artifactory maven
  - artifactoryPromoteBuild: Promote build

Waiting for platform.twitter.com...

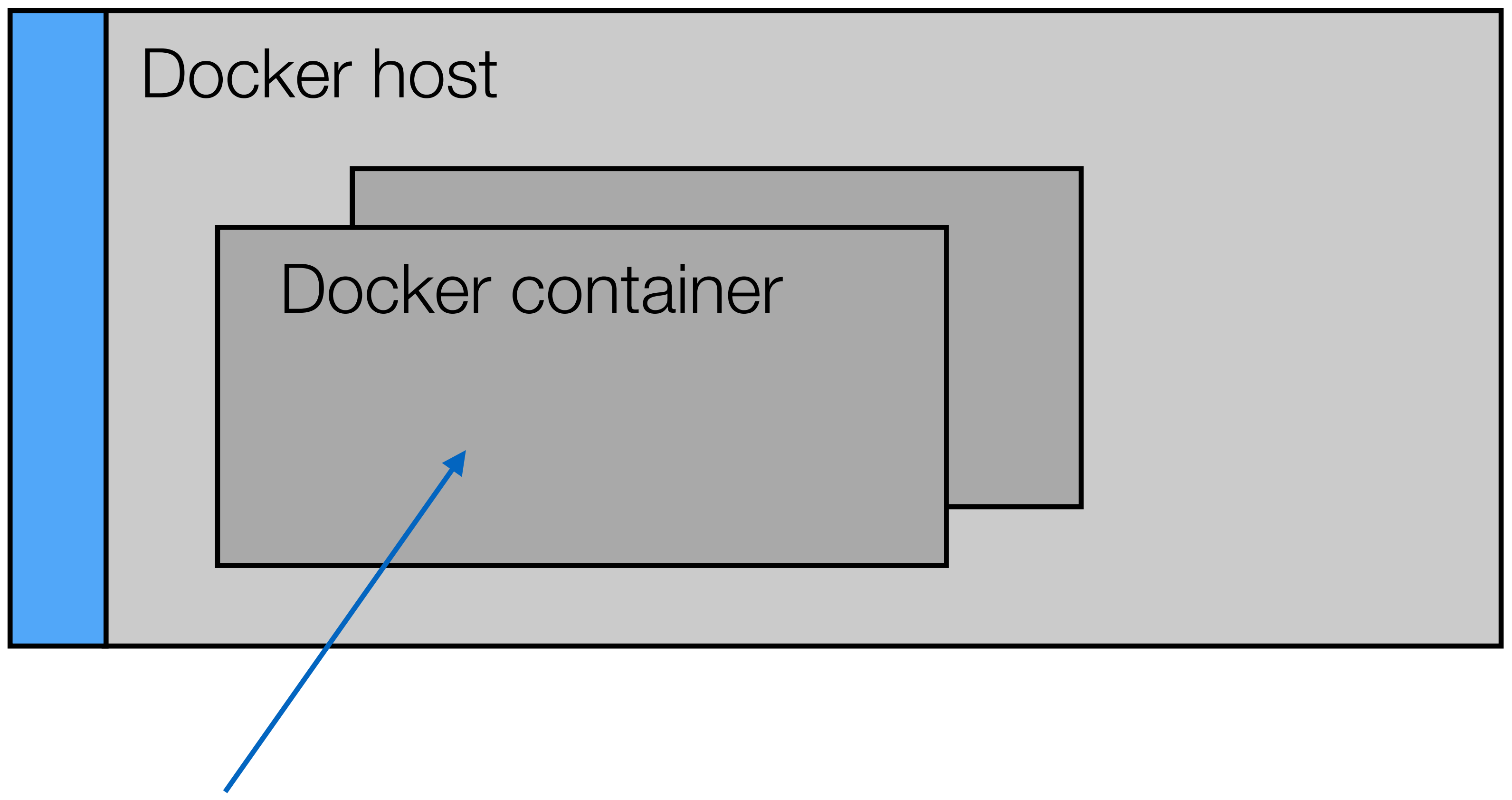
How can we use Docker in a Jenkins container...?



Remember that the CLI is a a Docker client talking to the Docker host.
Communication can happen over a TCP or Unix socket.

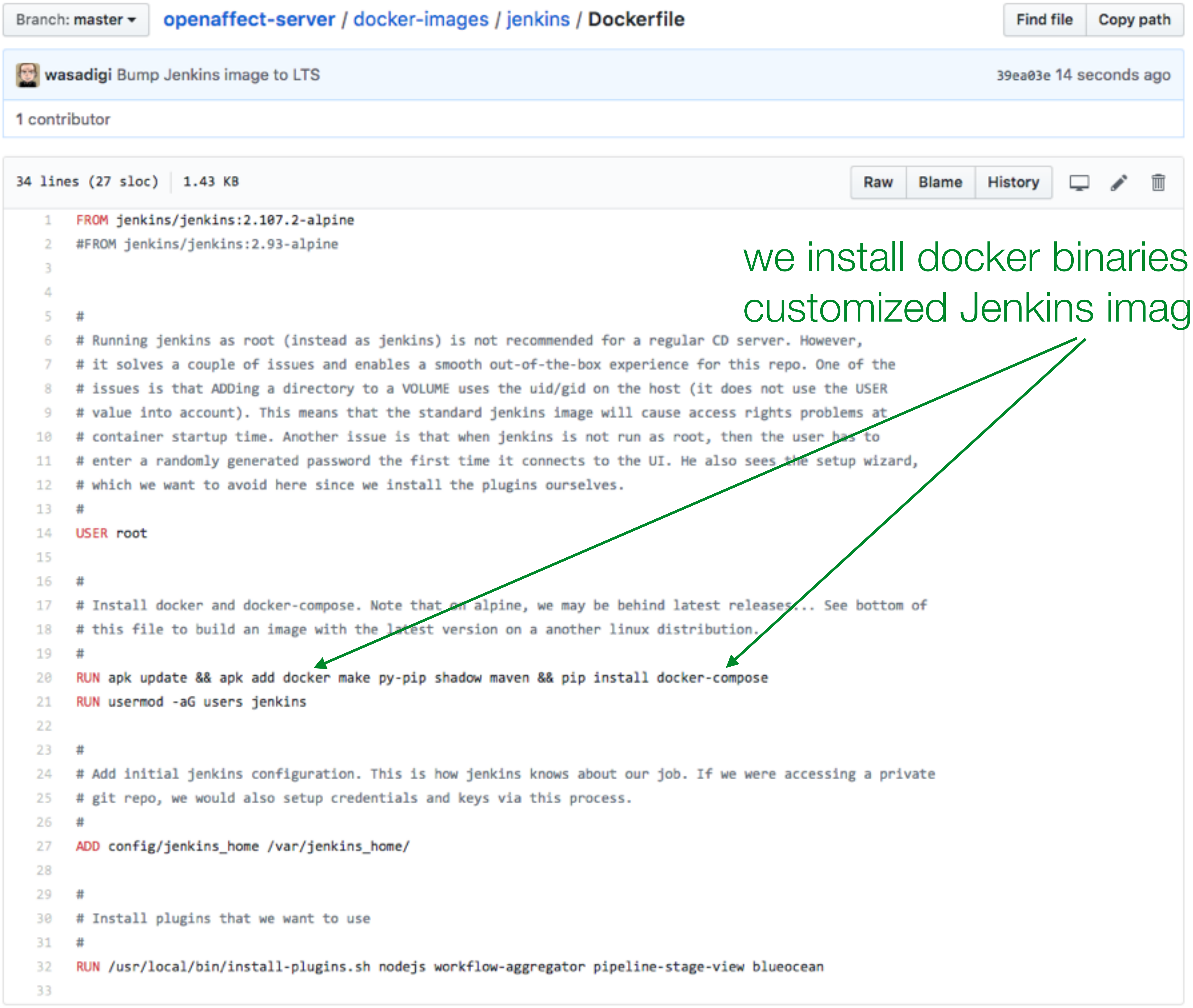
MASTER OF SCIENCE
IN ENGINEERING

How can we use Docker in a Jenkins container...?



To be able to start a container from here, we need 1) to have the docker client binaries (and later docker-compose) and 2) to have a way to reach a docker host.

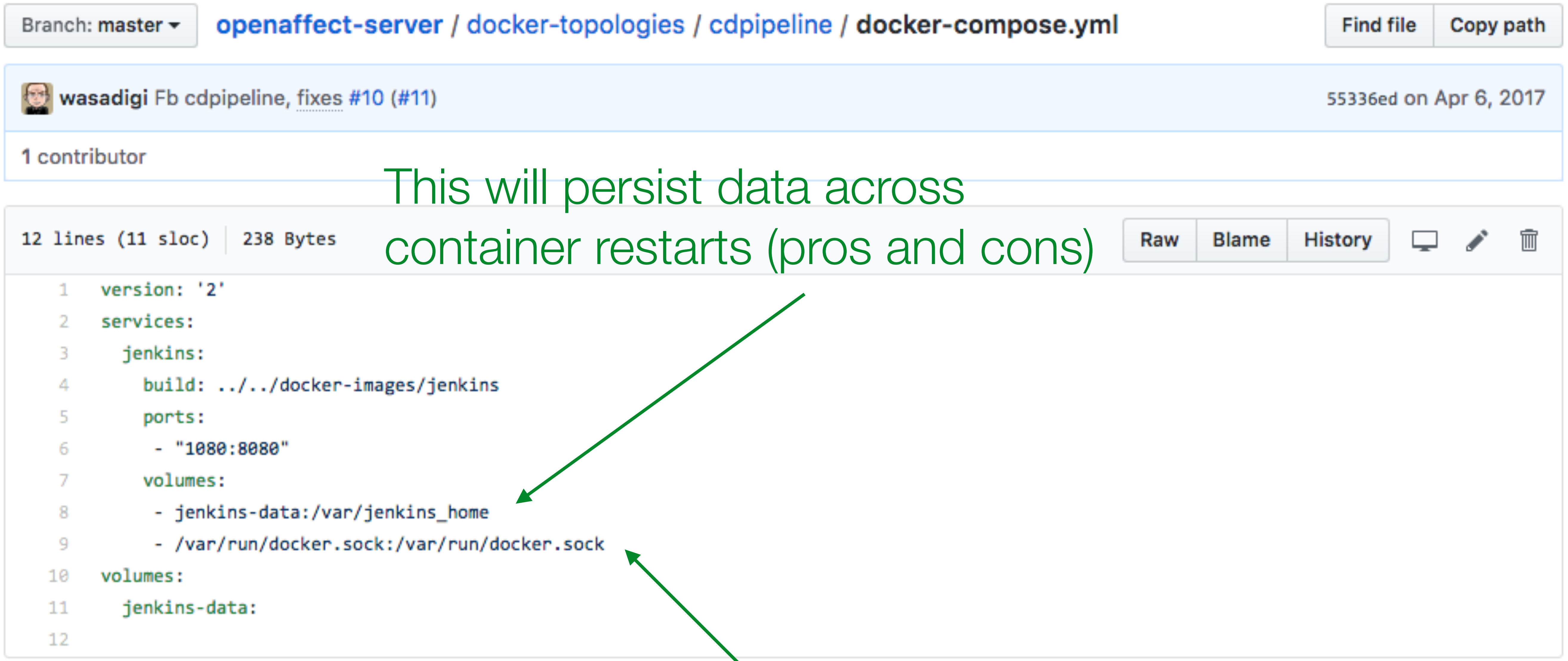
We will use the same docker host. In other words, our container will start a "brother" container (not a child).


Branch: master
openaffect-server / docker-images / jenkins / Dockerfile
Find file
Copy path
wasadigi Bump Jenkins image to LTS
39ea03e 14 seconds ago
1 contributor
34 lines (27 sloc) | 1.43 KB
Raw
Blame
History

```
FROM jenkins/jenkins:2.107.2-alpine
#FROM jenkins/jenkins:2.93-alpine


#
# Running jenkins as root (instead as jenkins) is not recommended for a regular CD server. However,
# it solves a couple of issues and enables a smooth out-of-the-box experience for this repo. One of the
# issues is that ADDing a directory to a VOLUME uses the uid/gid on the host (it does not use the USER
# value into account). This means that the standard jenkins image will cause access rights problems at
# container startup time. Another issue is that when jenkins is not run as root, then the user has to
# enter a randomly generated password the first time it connects to the UI. He also sees the setup wizard,
# which we want to avoid here since we install the plugins ourselves.
#
USER root

#
# Install docker and docker-compose. Note that on alpine, we may be behind latest releases... See bottom of
# this file to build an image with the latest version on a another linux distribution.
#
RUN apk update && apk add docker make py-pip shadow maven && pip install docker-compose
RUN usermod -aG users jenkins

#
# Add initial jenkins configuration. This is how jenkins knows about our job. If we were accessing a private
# git repo, we would also setup credentials and keys via this process.
#
ADD config/jenkins_home /var/jenkins_home/

#
# Install plugins that we want to use
#
RUN /usr/local/bin/install-plugins.sh nodejs workflow-aggregator pipeline-stage-view blueocean

```

we install docker binaries in our customized Jenkins image

F SCIENCE
ERING



openaffect-server / docker-topologies / cdpipeline / docker-compose.yml

wasadigi Fb cdpipeline, fixes #10 (#11) 55336ed on Apr 6, 2017

1 contributor

12 lines (11 sloc) | 238 Bytes

```yaml
version: '2'
services:
  jenkins:
    build: ../../docker-images/jenkins
    ports:
     - "1080:8080"
    volumes:
     - jenkins-data:/var/jenkins_home
     - /var/run/docker.sock:/var/run/docker.sock
volumes:
  jenkins-data:

```

This will persist data across container restarts (pros and cons)

This is the trick: we mount the unix socket from the host file system on the container file system. As a result, running docker commands from the host or from this container will talk to the same docker engine.

Your turn!

- **If you have followed all the tasks until today, you should have:**
  - a git repo with the code of your micro-service
  - Jenkins running in a Docker container (official image)
  - A “freestyle” job configured in your Jenkins, which uses maven to build the codebase and produce a .jar
- **Let’s move closer to the openaffect template:**
  - Make sure that you build your custom image of Jenkins, so that the plugins are pre-installed and that the default pipeline is ready to use.
  - Make sure that you have a Jenkinsfile in your repo.
  - Don’t worry about the automated tests now (remove the last 2 stages from the Jenkinsfile.
  - After running your pipeline, make sure that your micro-service and the database are running and that you can connect.