Software Engineering and Architecture
Lecture 3: Continuous Delivery (1)

Olivier Liechti
olivier.liechti@heig-vd.ch

MSE | MASTER OF SCIENCE IN ENGINEERING

Today.

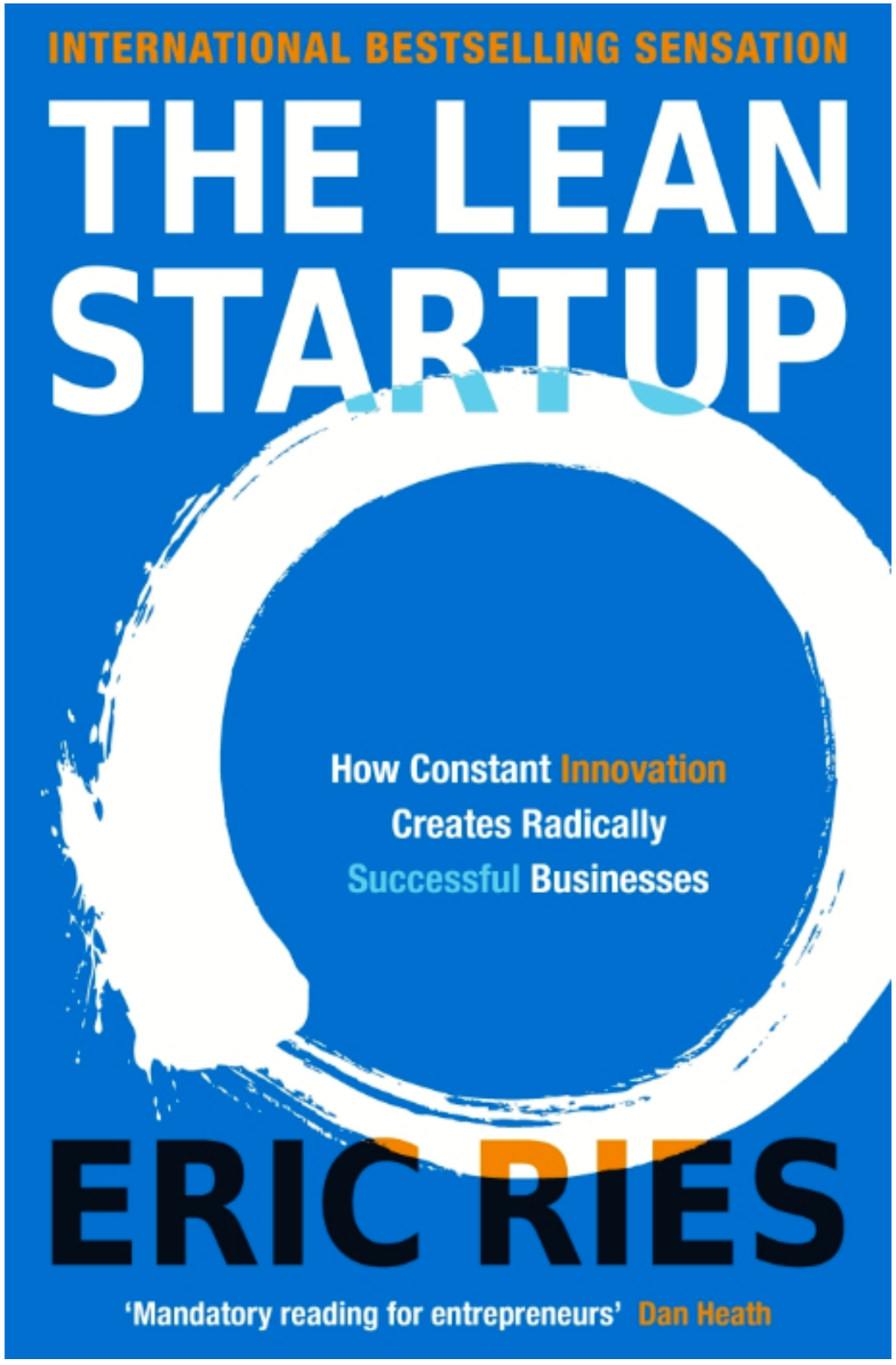
Continuous Delivery (1)
15h00 - 15h45

OO Reengineering Patterns (7 x 10’)
16h00 - 17h10

First Jenkins demo
17h10 - 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*	

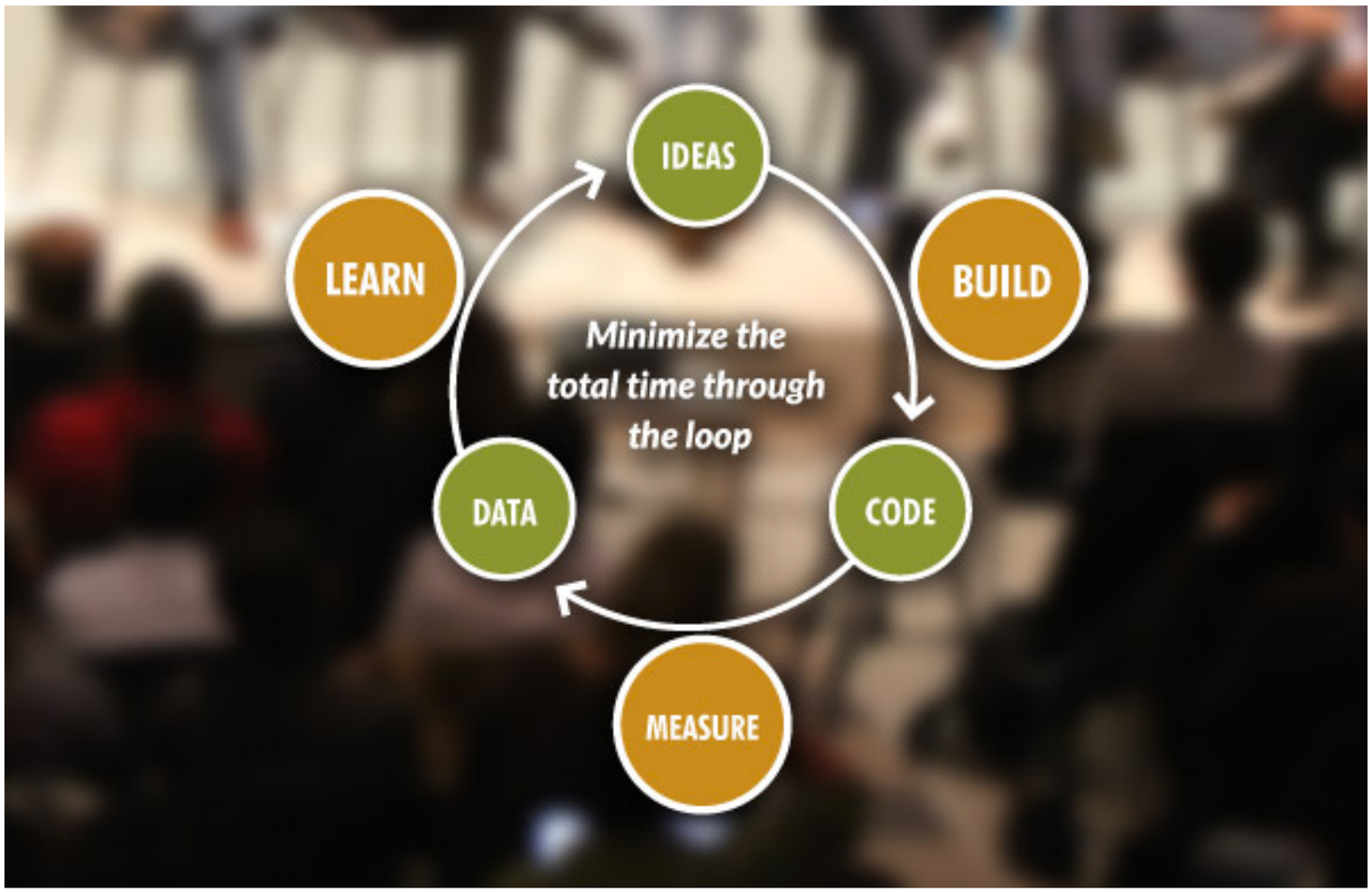
The Business Need



MSE | MASTER OF SCIENCE IN ENGINEERING

How to build the right product?

Apply the scientific method to business ideas



http://theleanstartup.com/principles

Problem statement

How do we ensure that the business can quickly run a lot of experiments, to assess the value of features or to compare alternatives?

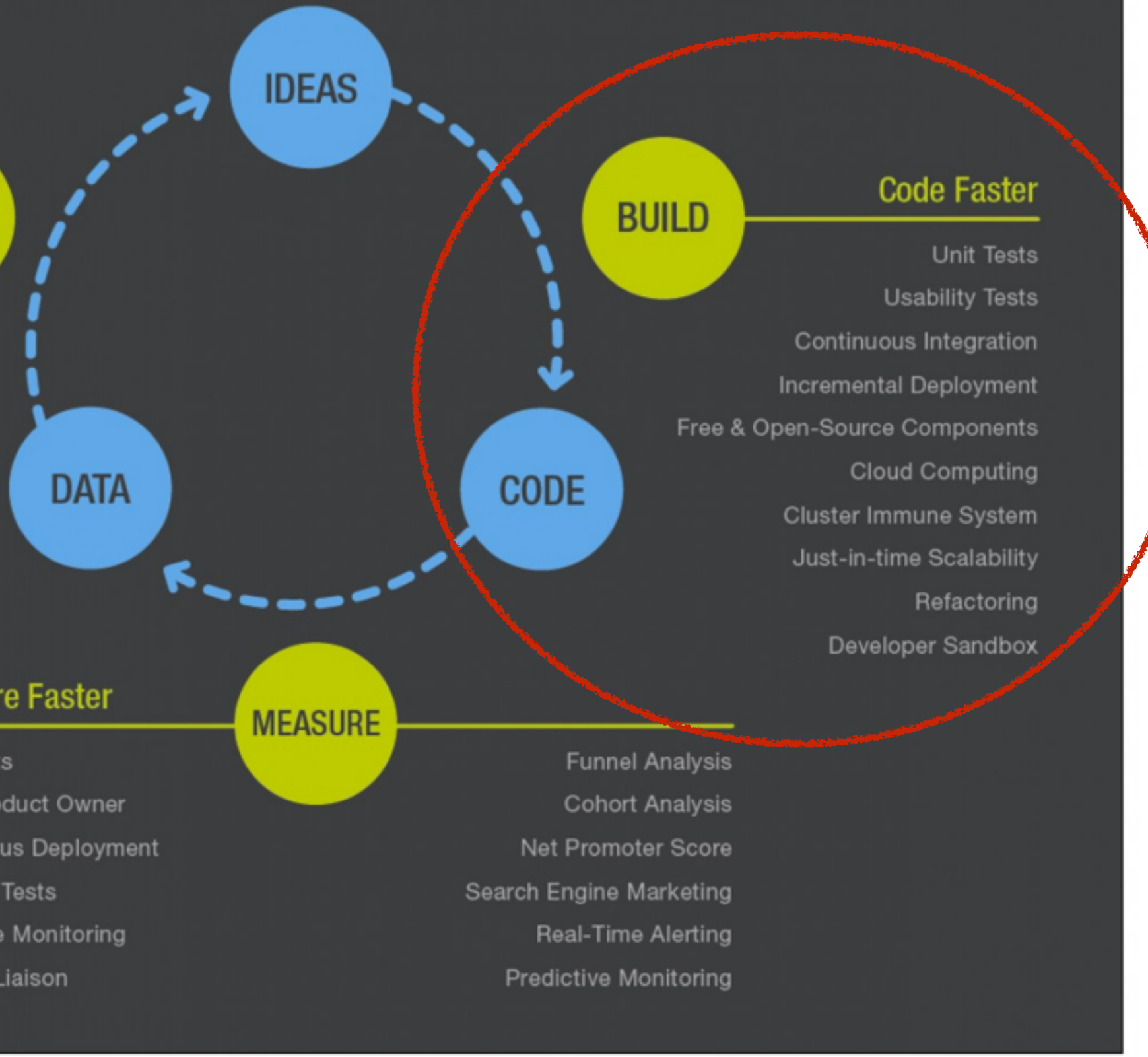
“If we added a feature to receive Telegram notifications from our online shop, how many customers would use it?”

How much time do we need to 1) implement a minimal version of this feature, 2) to release it to a portion of the audience and 3) to collect data?

THE LEAN STARTUP

Created by Eric Ries - startuplessonslearned.blogspot.com

Designed by KISSmetrics



Learn Faster

- Split Tests
- Customer Interviews
- Customer Development
- Five Whys Root Cause Analysis
- Customer Advisory Board
- Falsifiable Hypotheses
- Product Owner Accountability
- Custom Archetypes
- Cross-functional Teams
- Smoke Tests

Code Faster

- Unit Tests
- Usability Tests
- Continuous Integration
- Incremental Deployment
- Free & Open-Source Components
- Cloud Computing
- Cluster Immune System
- Just-in-time Scalability
- Refactoring
- Developer Sandbox

Measure Faster

- Split Tests
- Clear Product Owner
- Continuous Deployment
- Usability Tests
- Real-time Monitoring
- Custom Liaison
- Funnel Analysis
- Cohort Analysis
- Net Promoter Score
- Search Engine Marketing
- Real-Time Alerting
- Predictive Monitoring

The Problem with Silos

Ops guy

Dev guy

“Here is a USB key with the .war file that you need to deploy on Tomcat. I have written down the configuration steps on this sheet of paper...”

Pissed-off Ops guy

Pissed-off Dev guy

"WTF... why do I get a database connection error when I deploy this damn application in the Glassfish cluster...???"

Worried Ops guy

Worried Dev guy

*“We have deployed the new release... are we **sure** that we are ok?*
By the way, anyone has a clue about what is in the release?”

*“I hate these production rollouts... it’s always so **stressful**.*
What could we do to avoid bad surprises and not be so
***afraid to touch the prod**...?”*

*"Releasing **small chunks** of changes is less scary than releasing big chunks of changes."*

*"Something that you do **often** is less scary than something you do rarely."*

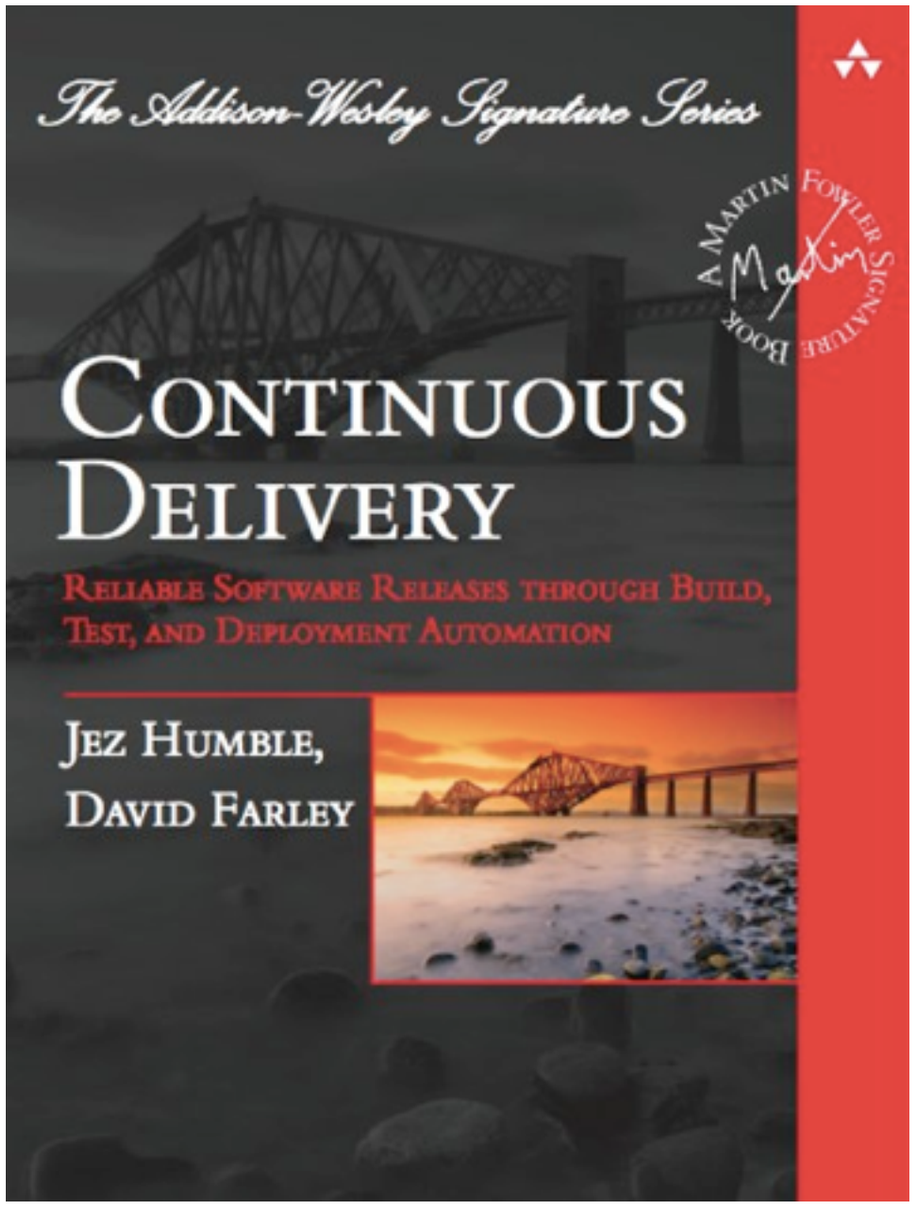
Agile
Robert Redford

*"If something is **automated**, there is now worry that I can make a wrong manipulation."*

*"If I have **evidence** that the system has been **fully tested**, at different levels, I feel secure."*

The Solution

Continuous Delivery



The slides are based on this book (ISBN:)

Free chapter: http://www.informit.com/articles/article.aspx?p=1621865

Foreword, by Martin Fowler (1)

*"**Continuous Integration** is just the first step.*

Software that's been successfully integrated into a mainline code stream still isn't software that's out in production doing its job.

*Dave and Jez's book pick up the story from CI to **deal with that "last mile"** describing **how to build the deployment pipeline that turns integrated code into production software.**"*

Foreword, by Martin Fowler (2)

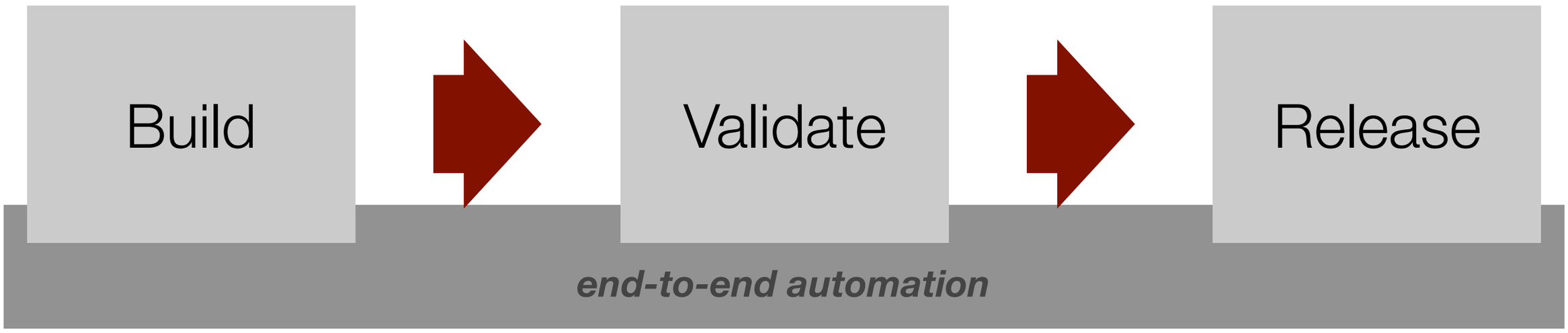
"This kind of delivery thinking has long been a forgotten corner of software development, falling into ***a hole between developers and operations teams****.*

So it's no surprise that the techniques in this book rest upon bringing these teams together—a harbinger of the nascent but growing ***DevOps*** *movement.*

This process also involves ***testers****, as testing is a key element of ensuring error-free releases."*

The Problem of Delivering Software

- How do we transform an **idea** into **something usable by end-users** as **quickly** as possible?
- We need an efficient **software development process**, but that is not enough.
- We need to cover the **complete delivery life-cycle**, until the release into the production environment.



Release Anti-Patterns

- **Anti-Pattern**: Deploying Software Manually
- **Anti-Pattern**: Deploying to a Production-like Environment Only after Development Is Complete
- **Anti-Pattern**: Manual Configuration Management of Production Environments

Principles of Software Delivery

1. Create a **Repeatable, Reliable Process** for Releasing Software
2. **Automate** Almost Everything
3. Keep Everything in **Version Control**
4. **If It Hurts, Do It More Frequently**, and Bring the Pain Forward
5. Build **Quality** In
6. "**Done**" Means Released
7. **Everybody Is Responsible for the Delivery Process**
8. **Continuous Improvement**

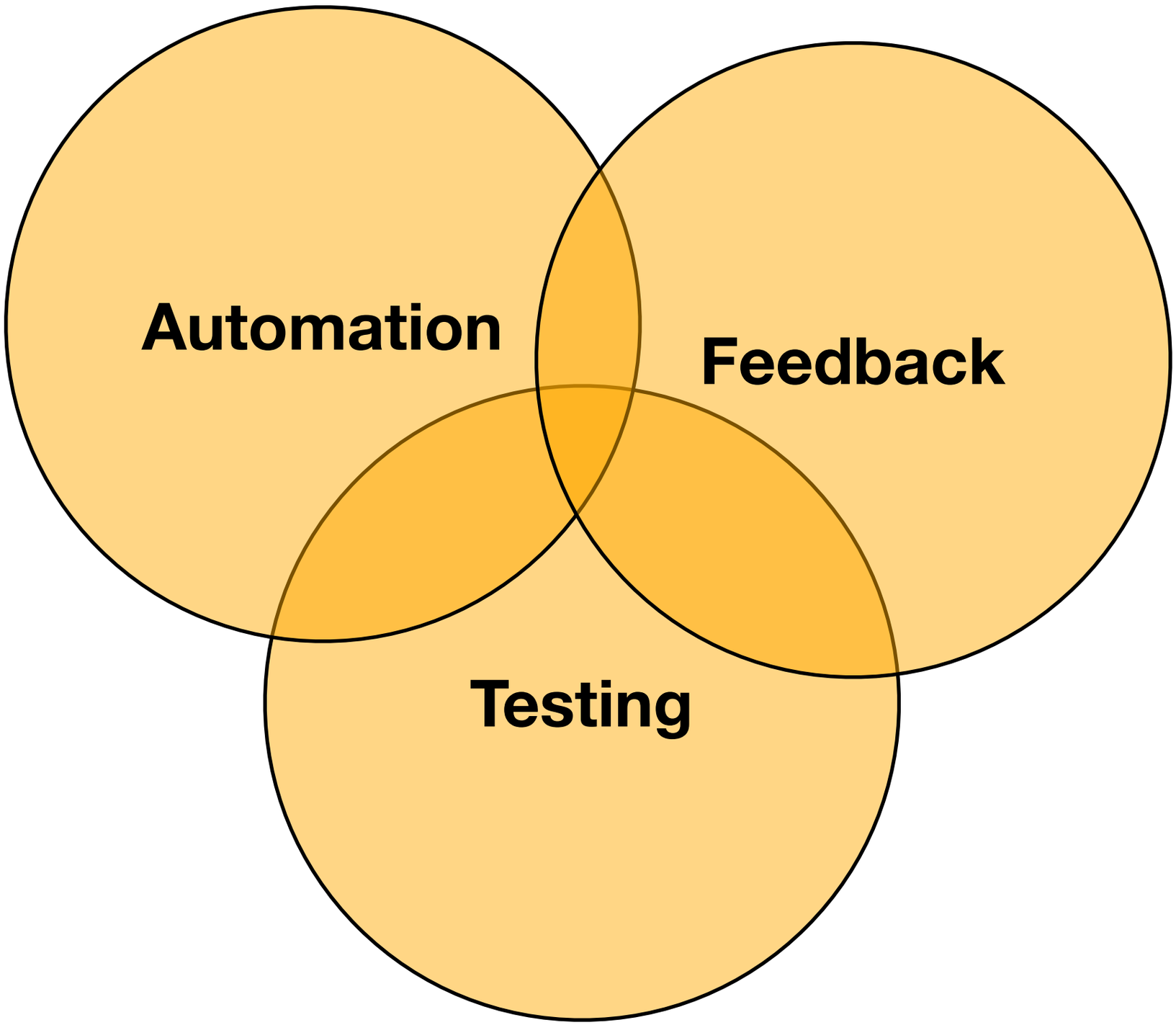
Frequent and Automated Releases

- We, and our fellow practitioners, have discovered that in order to achieve these goals—low cycle time and high quality—**we need to make frequent, automated releases of our software.** Why is this?

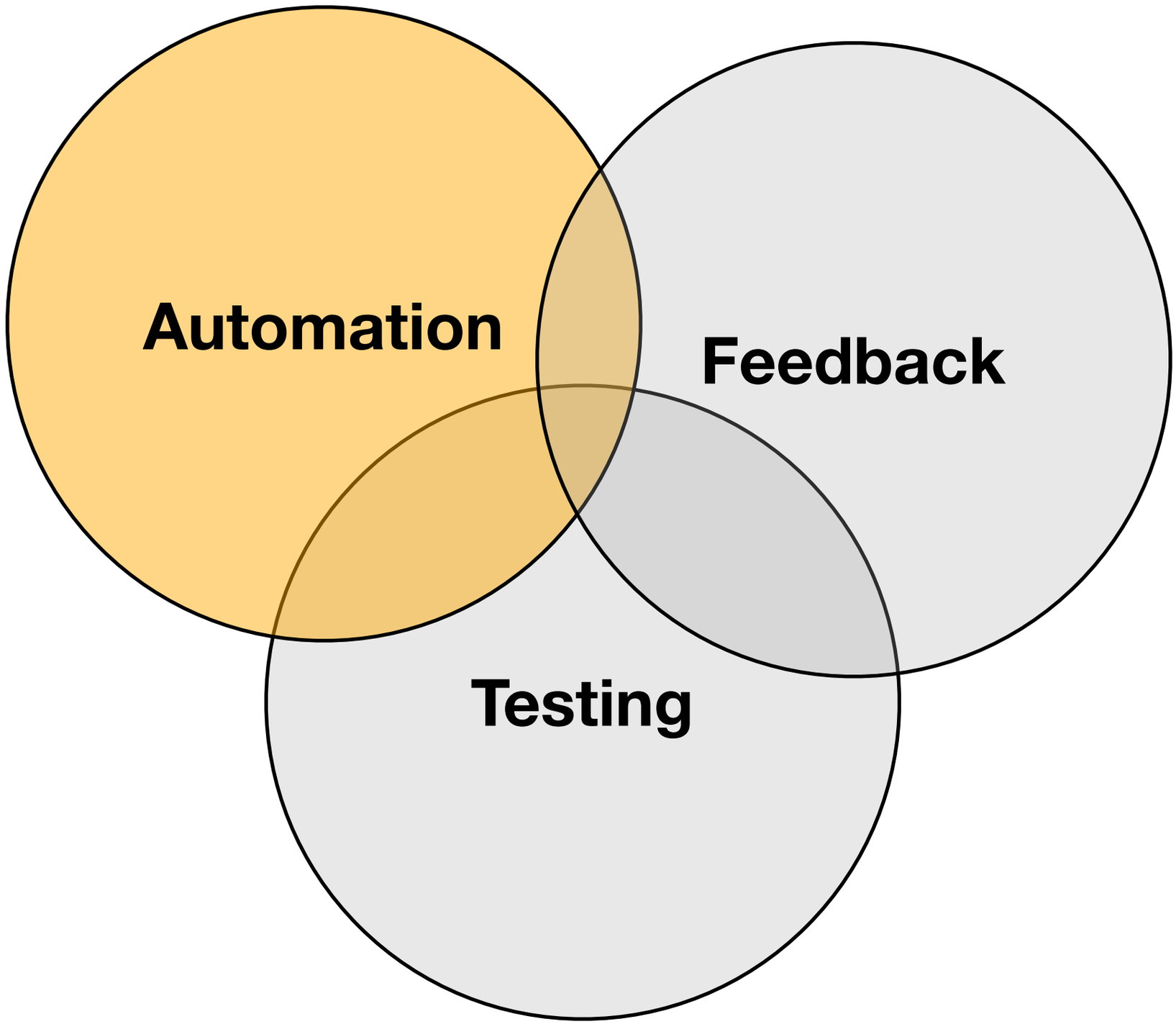
- **Automated.** If the build, deploy, test, and release process is not automated, it is not repeatable. Every time it is done, it will be different, because of changes in the software, the configuration of the system, the environments, and the release process. Since the steps are manual, they are error-prone, and there is no way to review exactly what was done. This means there is no way to gain control over the release process, and hence to ensure high quality. **Releasing software is too often an art; it should be an engineering discipline.**

- **Frequent.** If releases are frequent, the delta between releases will be small. This significantly reduces the risk associated with releasing and makes it much easier to roll back. Frequent releases also lead to faster feedback—indeed, they require it. Much of this book concentrates on getting feedback on changes to your application and its associated configuration (including its environment, deployment process, and data) as quickly as possible.

The Pillars of Continuous Delivery



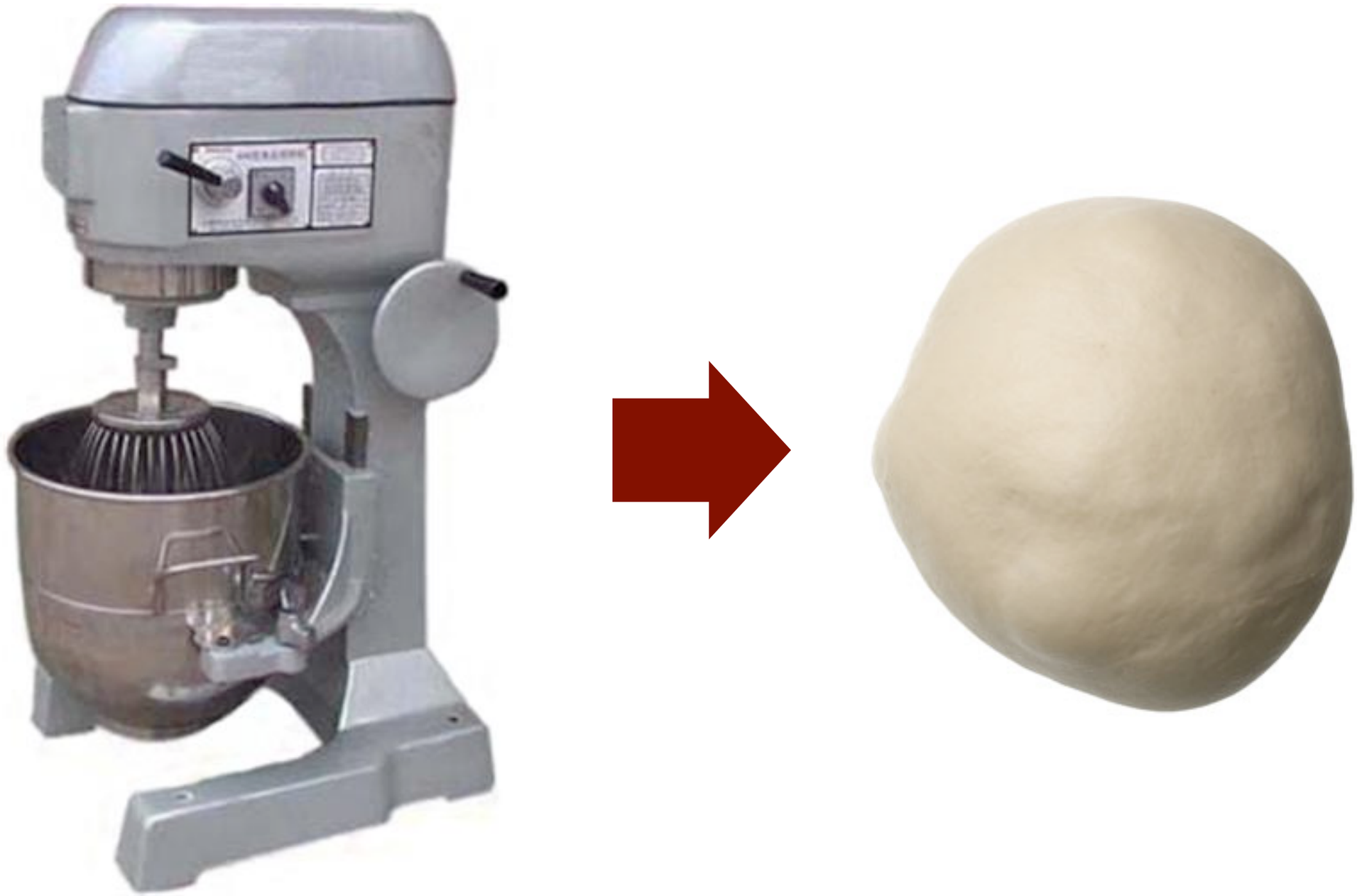
The Pillars of Continuous Delivery



The Deployment Pipeline

"The deployment pipeline has its foundations in the process of continuous integration and is in essence ***the principle of continuous integration taken to its logical conclusion****."*





Continuous Integration

Continuous Delivery

The Deployment Pipeline

Figure 1.1 *The deployment pipeline*

The Deployment Pipeline

"Every change that is made to an application's configuration, source code, environment, or data, ***triggers the creation of a new instance of the pipeline****.*

One of the first steps in the pipeline is to ***create binaries and installers****.*

The rest of the pipeline ***runs a series of tests*** *on the binaries to prove that they can be released. Each test that the release candidate passes gives us more confidence that this particular combination of binary code, configuration information, environment, and data will work.*

If the ***release candidate*** *passes all the tests, it can be* ***released****."*

The Deployment Pipeline

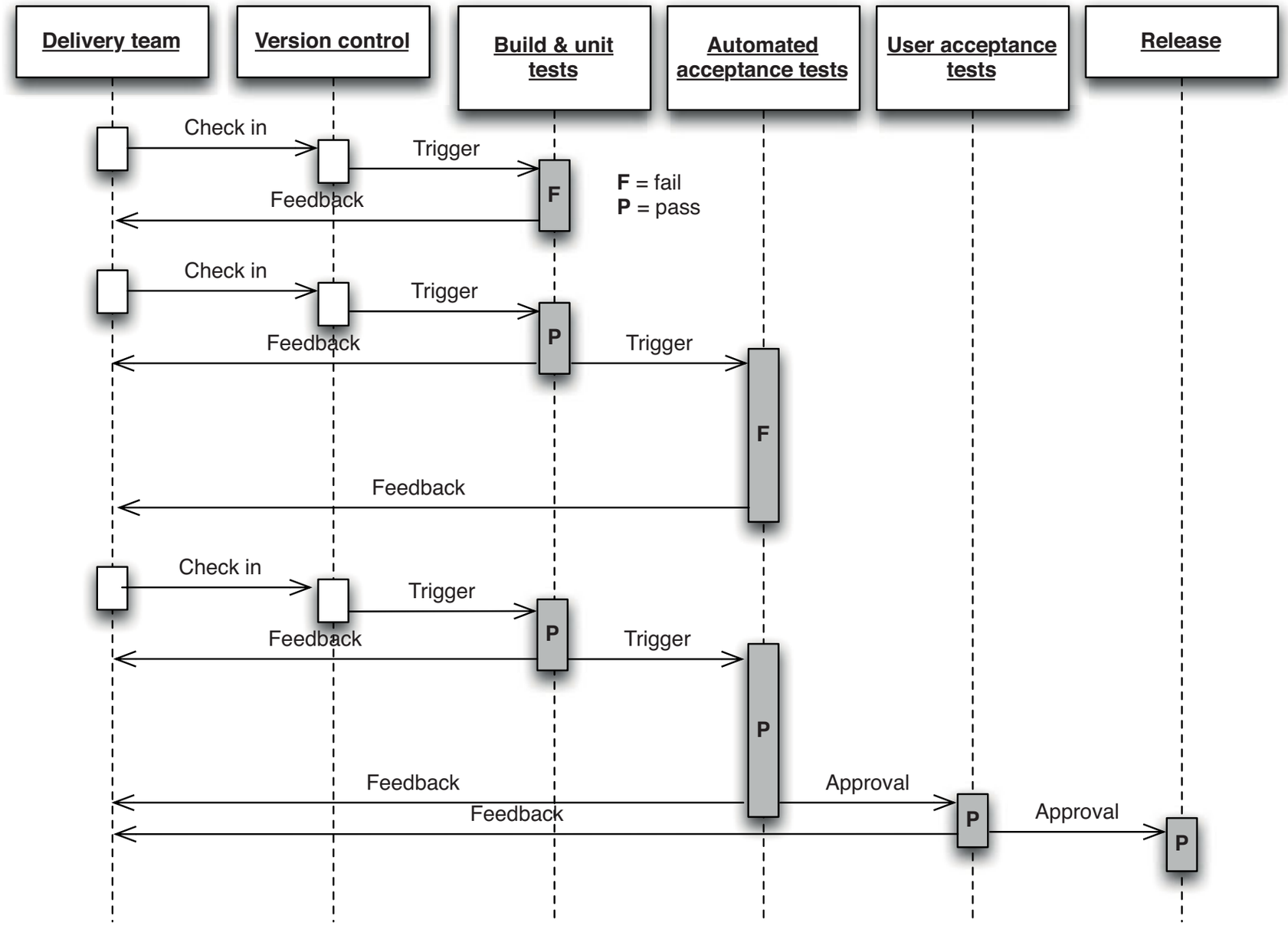


Figure 5.2 *Changes moving through the deployment pipeline*

The Deployment Pipeline

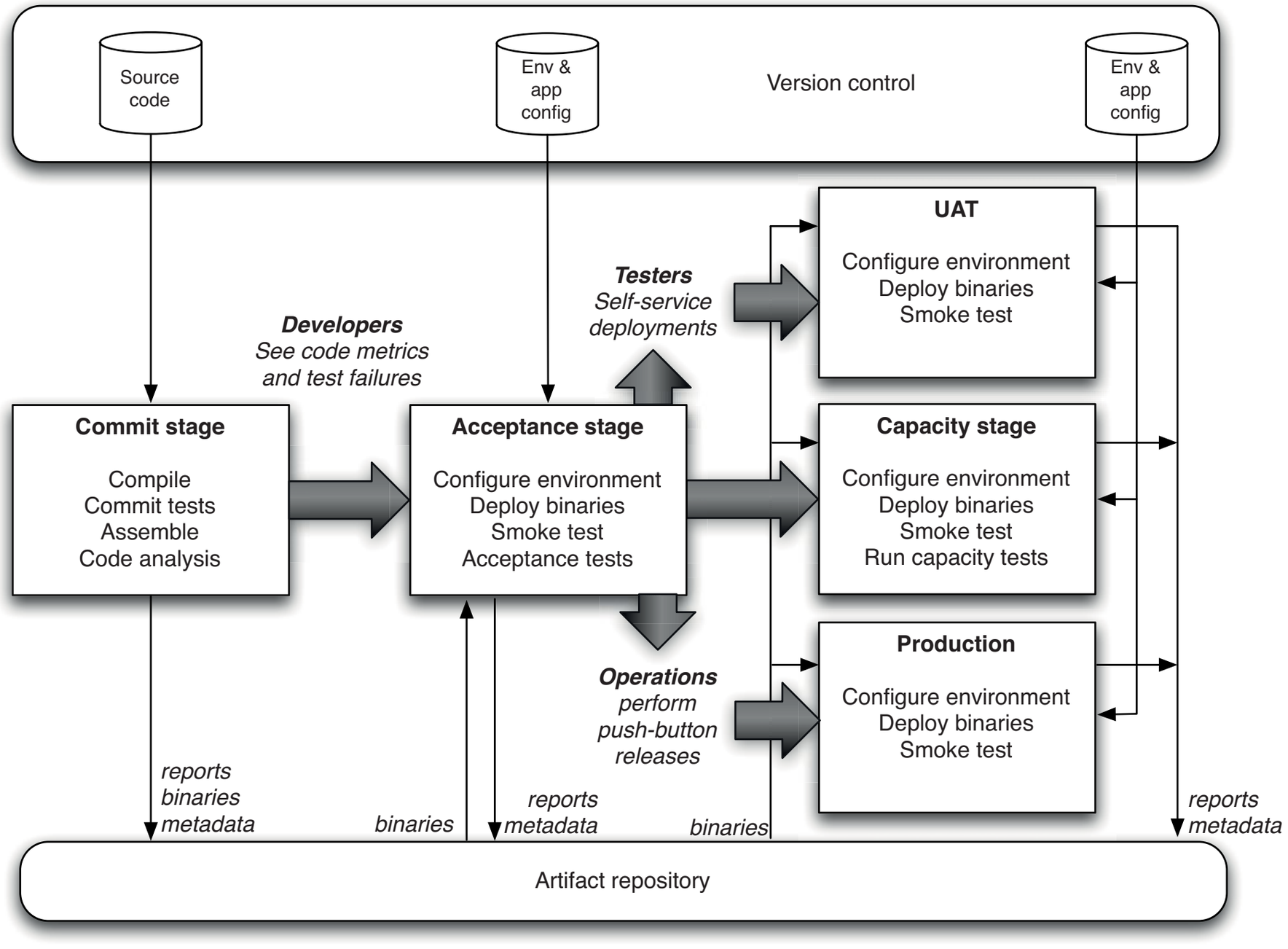


Figure 5.4 *Basic deployment pipeline*

The Commit Stage

- The commit stage begins with a change to the state of the project—that is, a **commit to the version control system**.
- It **ends with either a report of failure or**, if successful, **a collection of binary artifacts and deployable assemblies** to be used in subsequent test and release stages, as well as reports on the state of the application.
- Ideally, a commit stage should **take less than five minutes to run**, and certainly no more than ten.
- The commit stage represents, in more ways than one, **the entrance into the deployment pipeline**. Not only is it the point at which a new release candidate is created; it is also where many teams start when they begin to implement a deployment pipeline. **When a team implements the practice of continuous integration, it creates a commit stage in the process of doing so.**

Implementing a Deployment Pipeline

- **Take an agile approach**. Start simple, iterate and improve on a continuous basis.
- Consider the following elements:
  - Automate the **build** and **deployment** process.
  - Automate **unit tests** and **code analysis**.
  - Automate **acceptance tests**.
  - Automate **releases**.

Tools

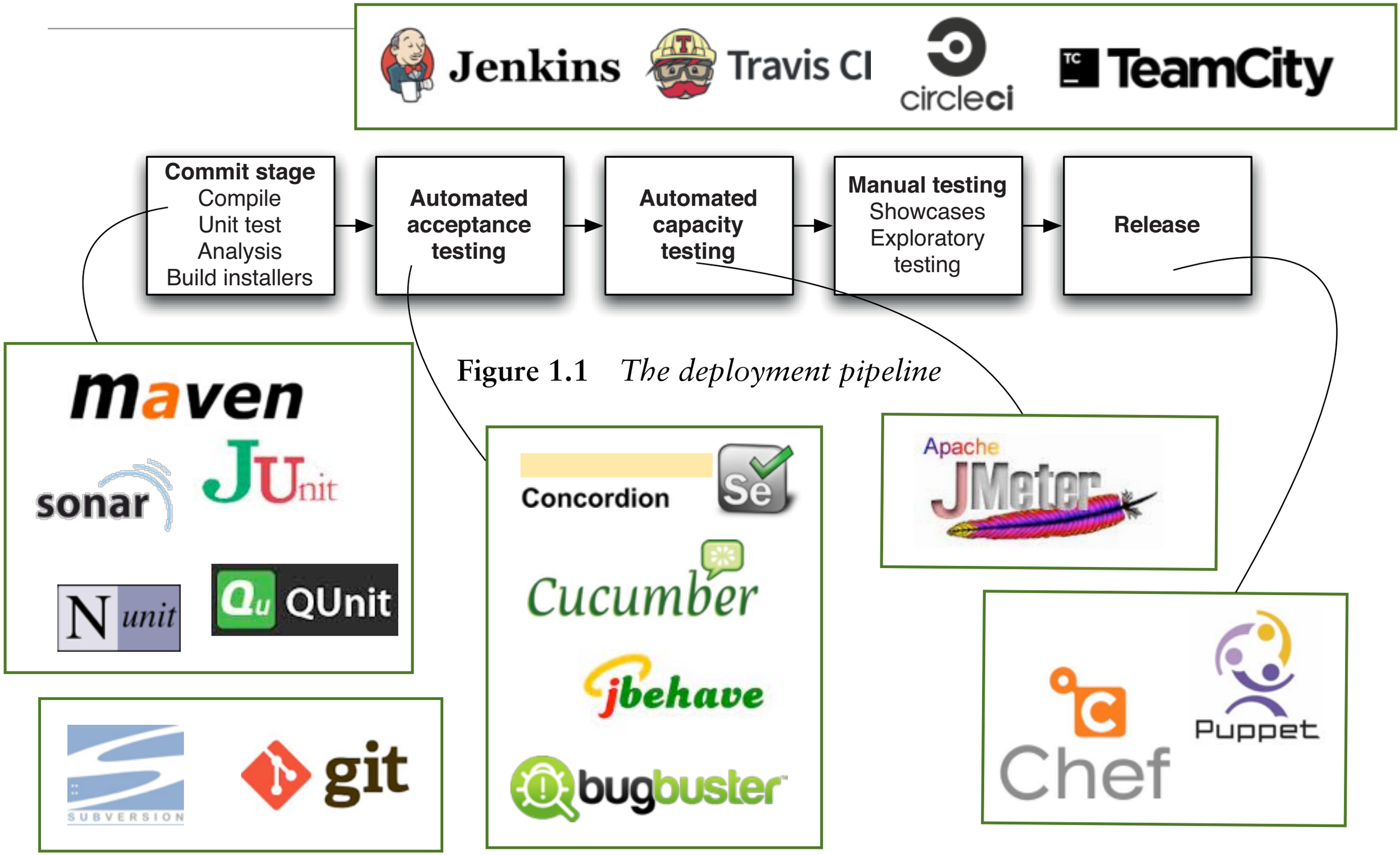


Figure 1.1 *The deployment pipeline*

Summary

*"There is **no one-size-fits-all solution** to the complex problem of implementing a deployment pipeline.*

*The crucial point is to create a system of record that **manages each change from check-in to release**, providing the **information** you need to discover problems as early as possible in the process."*

Summary

*"Having an implementation of the deployment pipeline can then be used to **drive out inefficiencies** in your process so you can make your feedback cycle faster and more powerful, perhaps by adding more automated acceptance tests and parallelizing them more aggressively, or by making your testing environments more production-like, or by implementing better configuration management processes."*

Summary

"A deployment pipeline, in turn, depends on having some ***foundations*** *in place:* ***good configuration management, automated scripts for building and deploying your application, and automated tests*** *to prove that your application will deliver value to its users.*

It also requires ***discipline****, such as ensuring that only changes that have passed through the automated build, test, and deployment system get released."*

Resources

- http://continuousdelivery.com

First experiments with Jenkins

- **Last week**
  - When you have made the 2 Spring Boot tutorials, you have run "**mvn clean install**" on your local machine. As a result, you have generated a .jar file in your "target" folder.
- **Today**
  - **Let's configure Jenkins to do the exact same procedure**. This would allow us to do a daily build of the micro-service (get the last version of the code and build it with maven).
  - This is going to be a very, **very basic build pipeline with only the commit stage**.

Overview

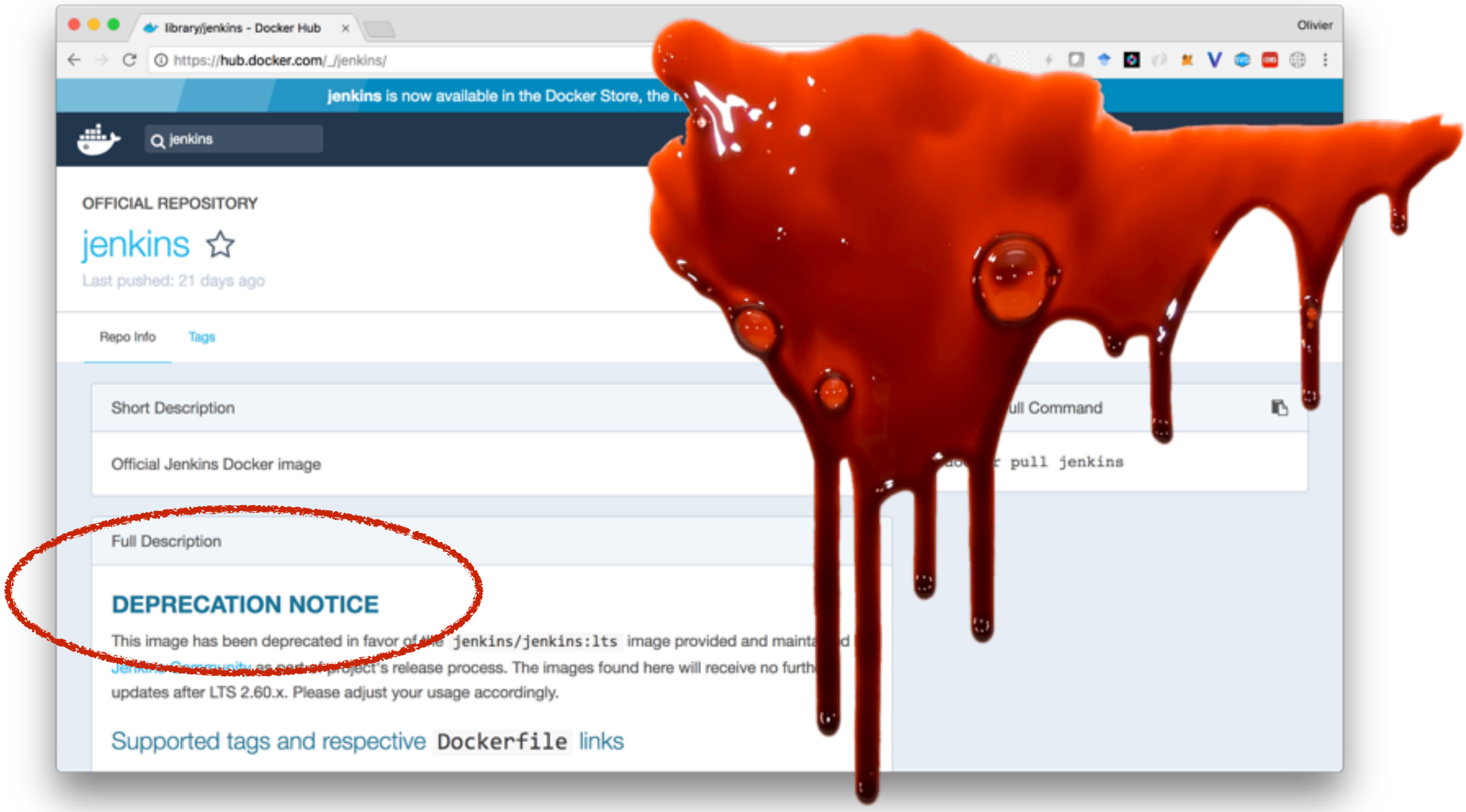
- **Jenkins** is one of the most popular continuous integration / continuous delivery severs.
- **There are many others**. Some are “cloud-first” services (e.g. Travis CI, CircleCI, etc.), others are “on-premise” products (e.g. GoCD, TeamCity, Bamboo, etc.).
- It is an **open source project** with a large community (lots of plugins) and the support of a commercial company (Cloudbees).
- It was initially created with a Java / maven focus but has a wider scope today.
- A **big transition** is underway, from being a continuous integration to becoming a full-fledged continuous delivery infrastructure. See the “pipeline” and “stage view” plugins released by Cloudbees.

First steps

- **The first thing that we want to do is to get a Jenkins server running on our machine and have a look at its web UI.**
- How do we do that?
  - We could make sure that we have a JVM on our machine, download the Jenkins binaries, follow the instructions, install it on our machine. We could then play with it. If we don’t like it, there are certainly instructions for uninstalling it.
  - Or we can take advantage of Docker!

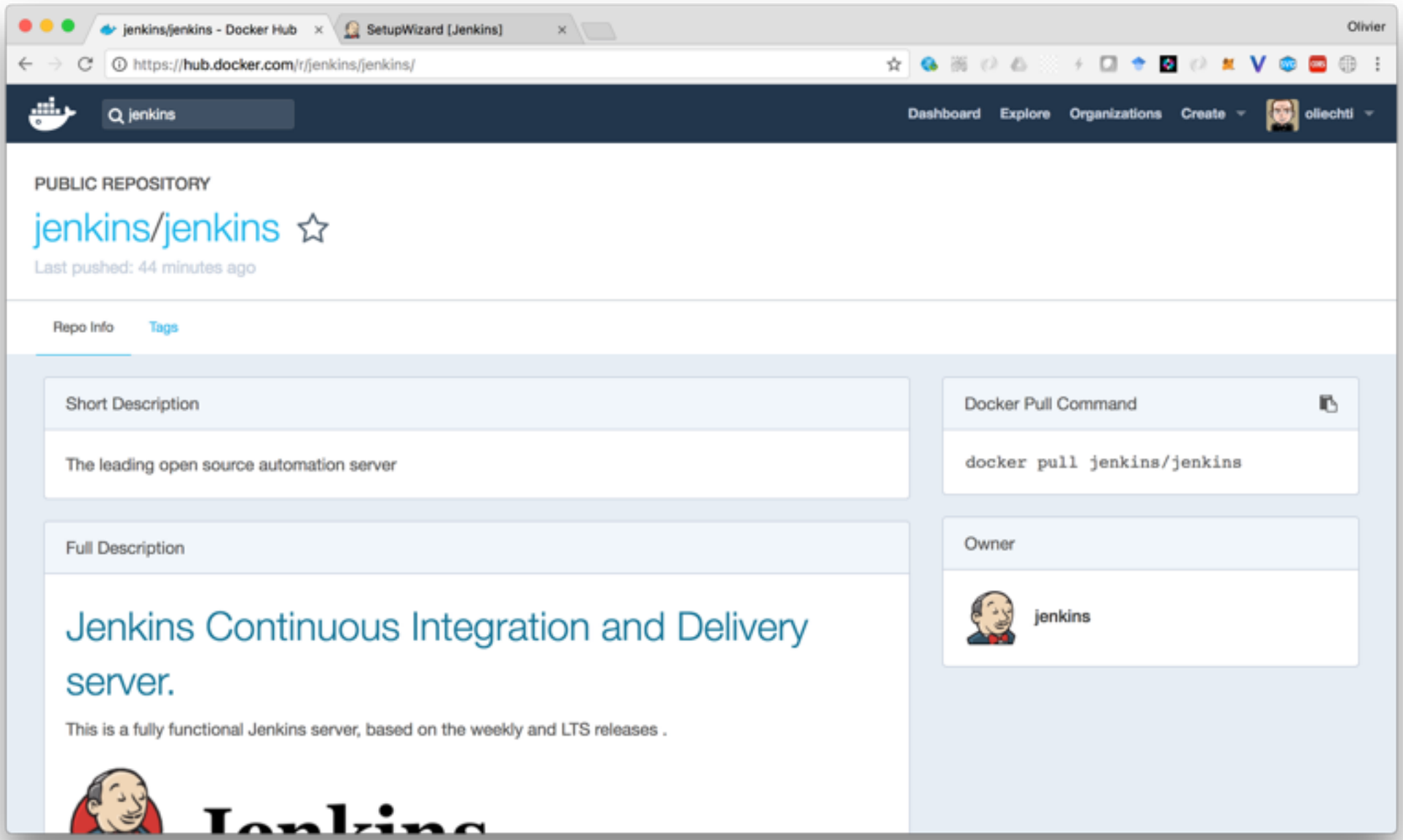
Ready to use Jenkins images

- **Warning: there are many Jenkins images on Dockerhub...**



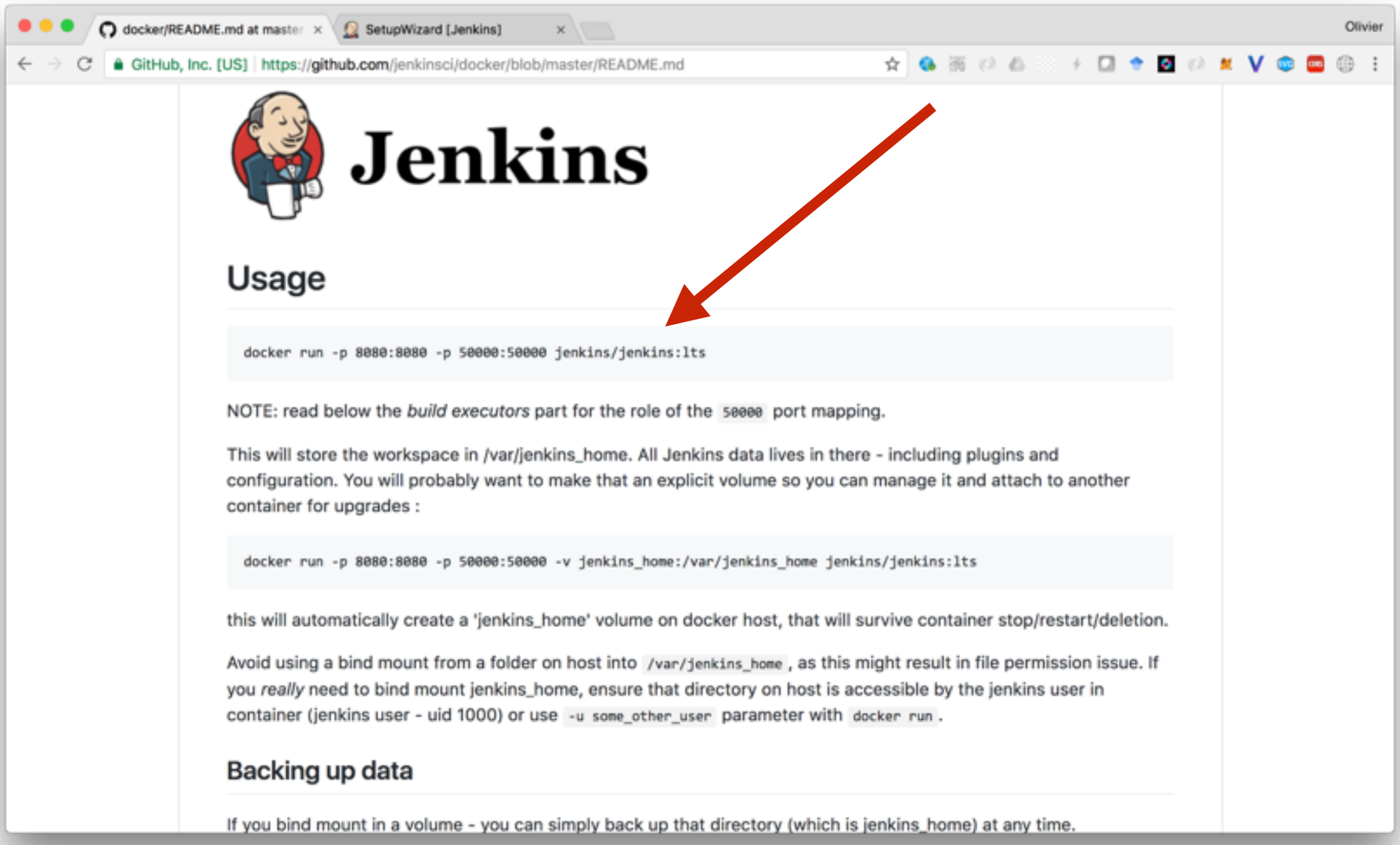
Ready to use Jenkins images

- **Use the jenkins/jenkins image**



The official Jenkins Docker image

- **It only takes one command to run it (notice the port mapping)!**

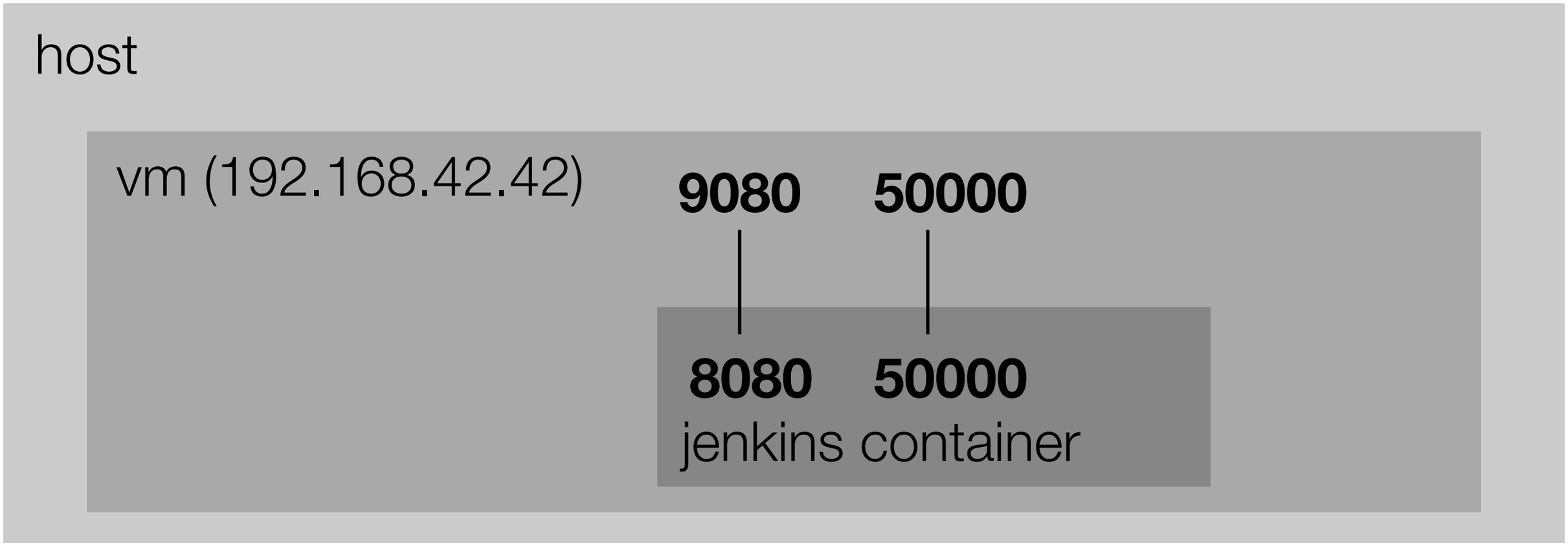


The official Jenkins Docker image

- **It only takes one command to run it (notice the port mapping)!**

```
$ docker run -p 9080:8080 -p 50000:50000 jenkins
```

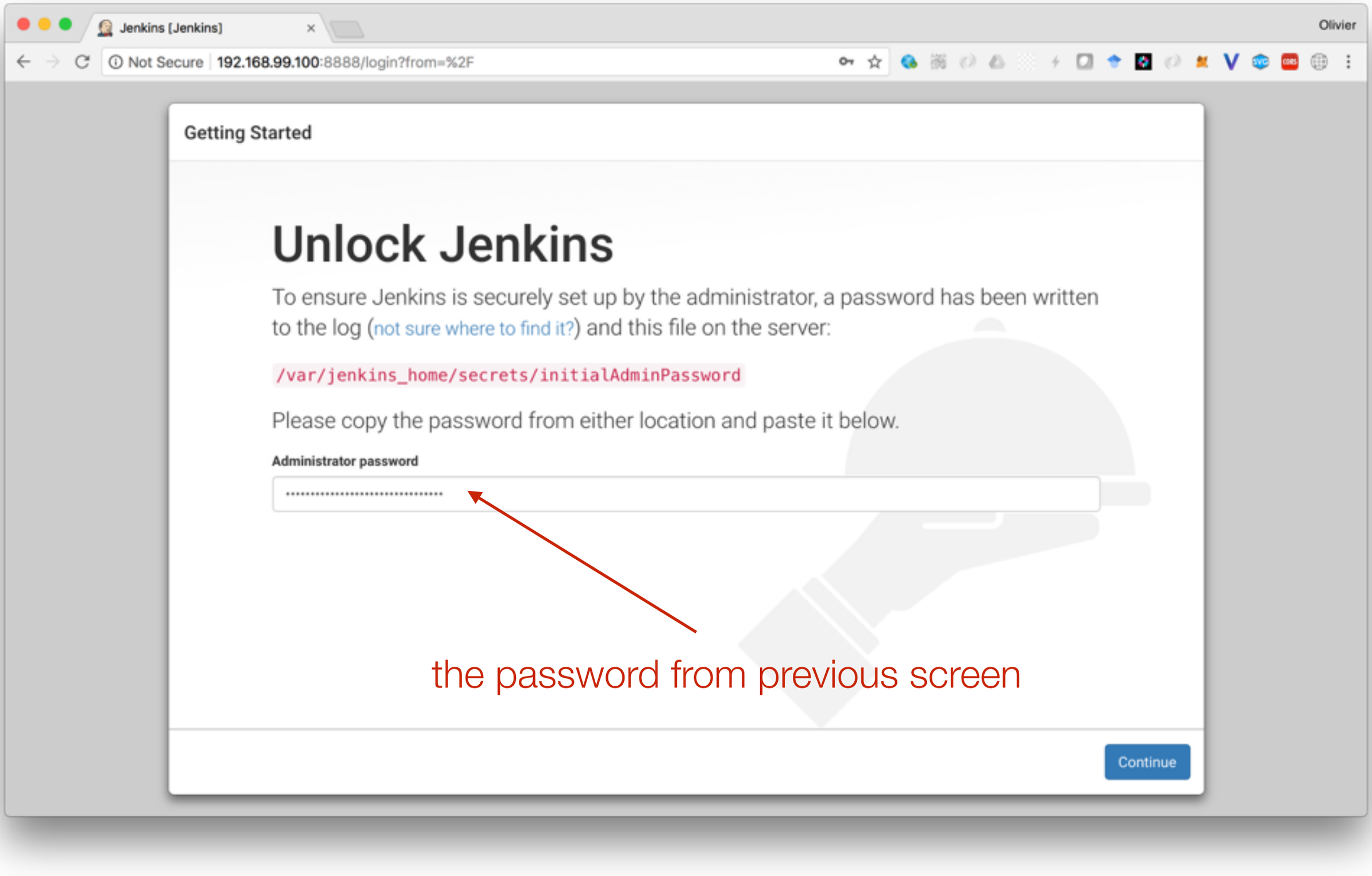
host port : container port

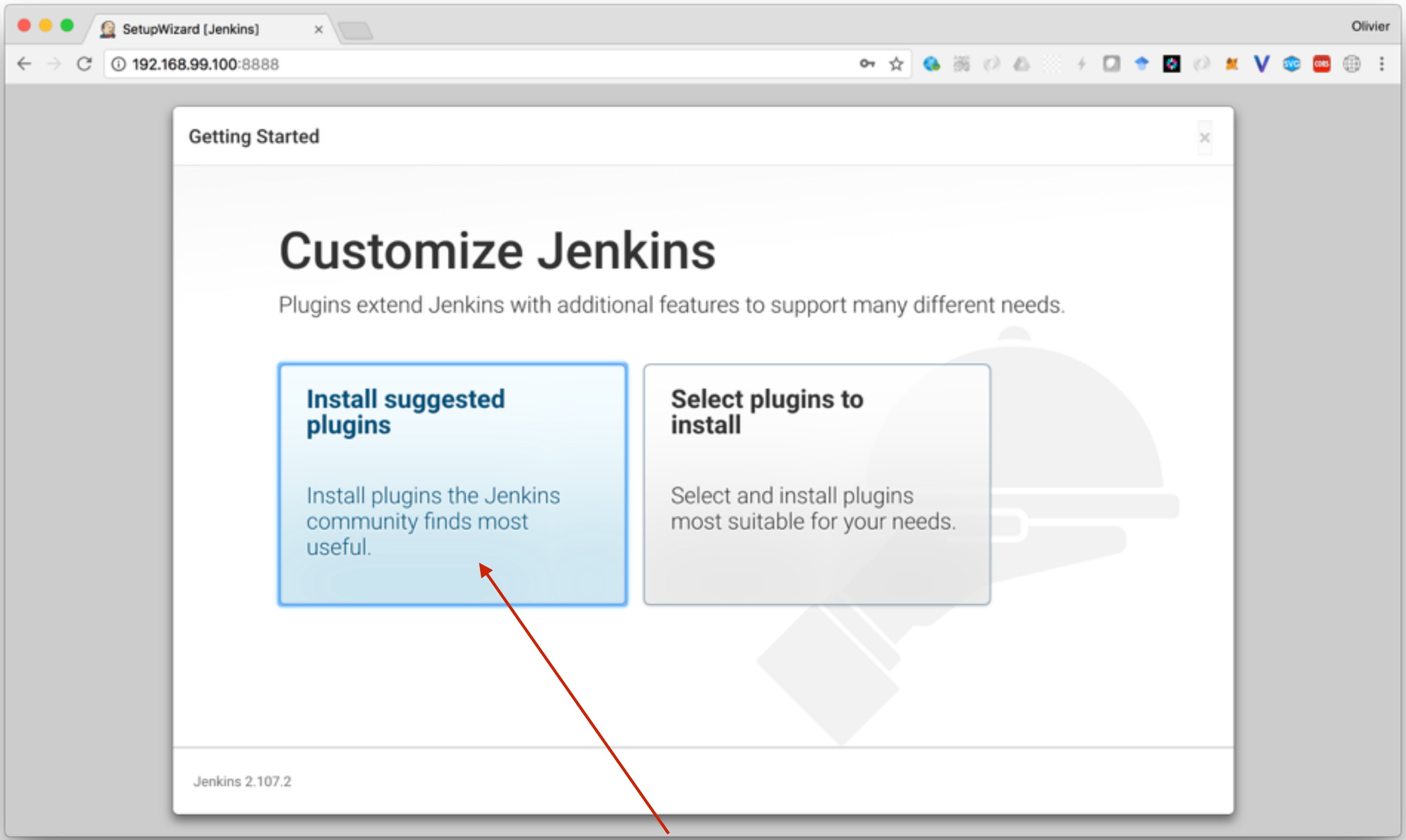


The official Jenkins Docker image

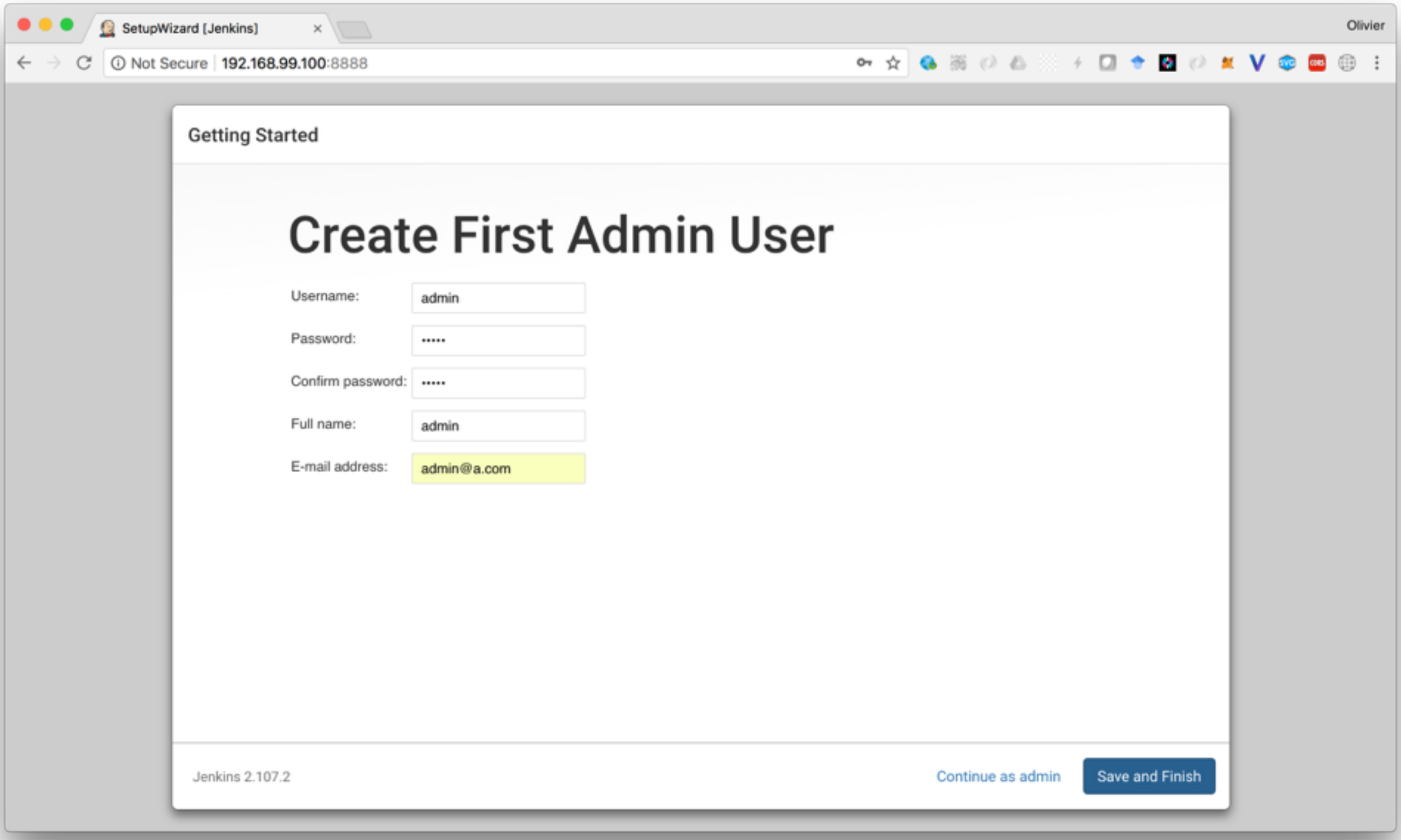
- **Read carefully what appears in the docker log. You will find the default admin password (for THIS container).**
- **If you are new to Docker, try to ‘docker run -it XXX /bin/sh’ (where XXX is the name of your container**

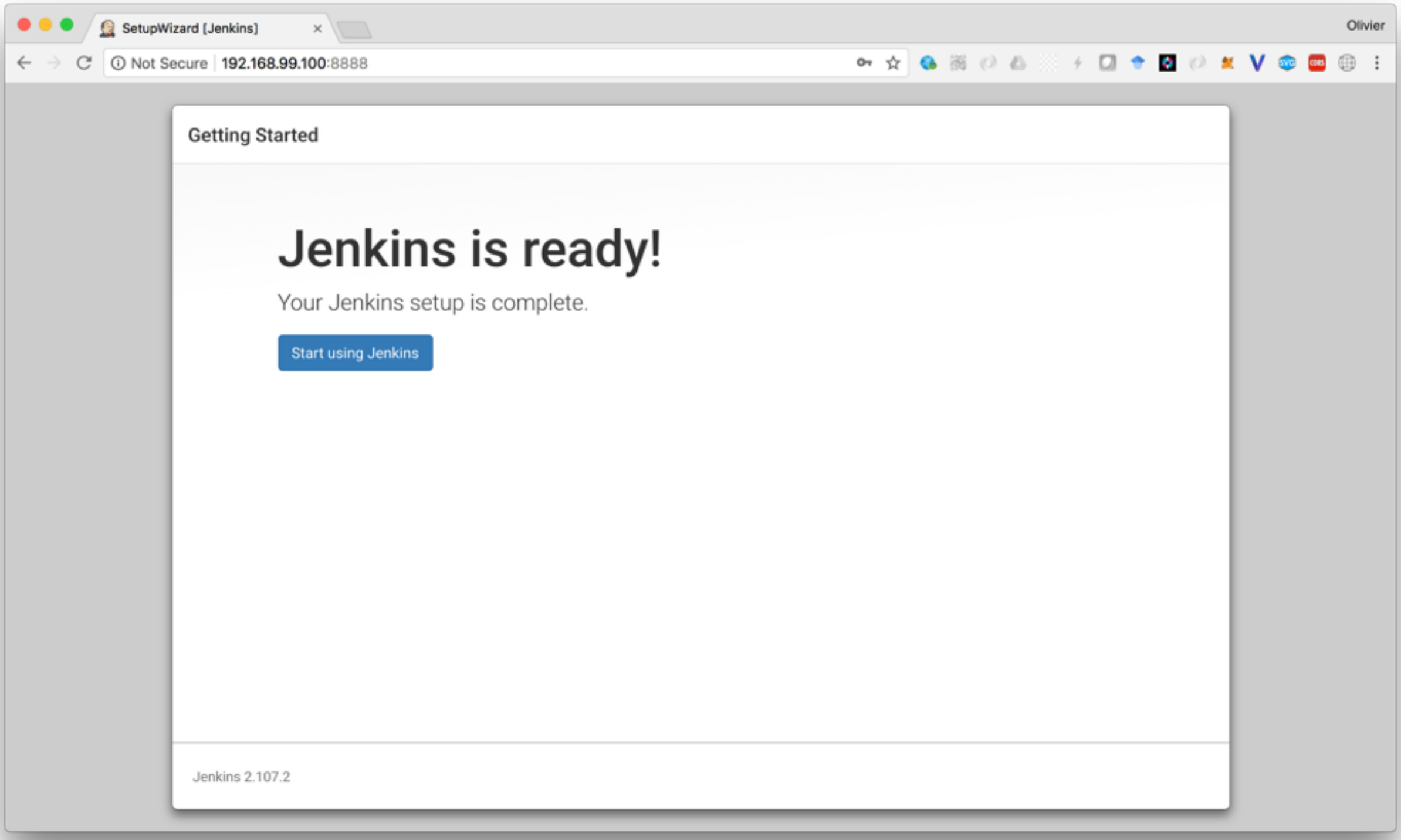
```
*************************************************************
*************************************************************
*************************************************************

Jenkins initial setup is required. An admin user has been created and a password generated.
Please use the following password to proceed to installation:

9f87ada80d464782b2249cad4fd1b688

This may also be found at: /var/jenkins_home/secrets/initialAdminPassword

*************************************************************
*************************************************************
*************************************************************
```

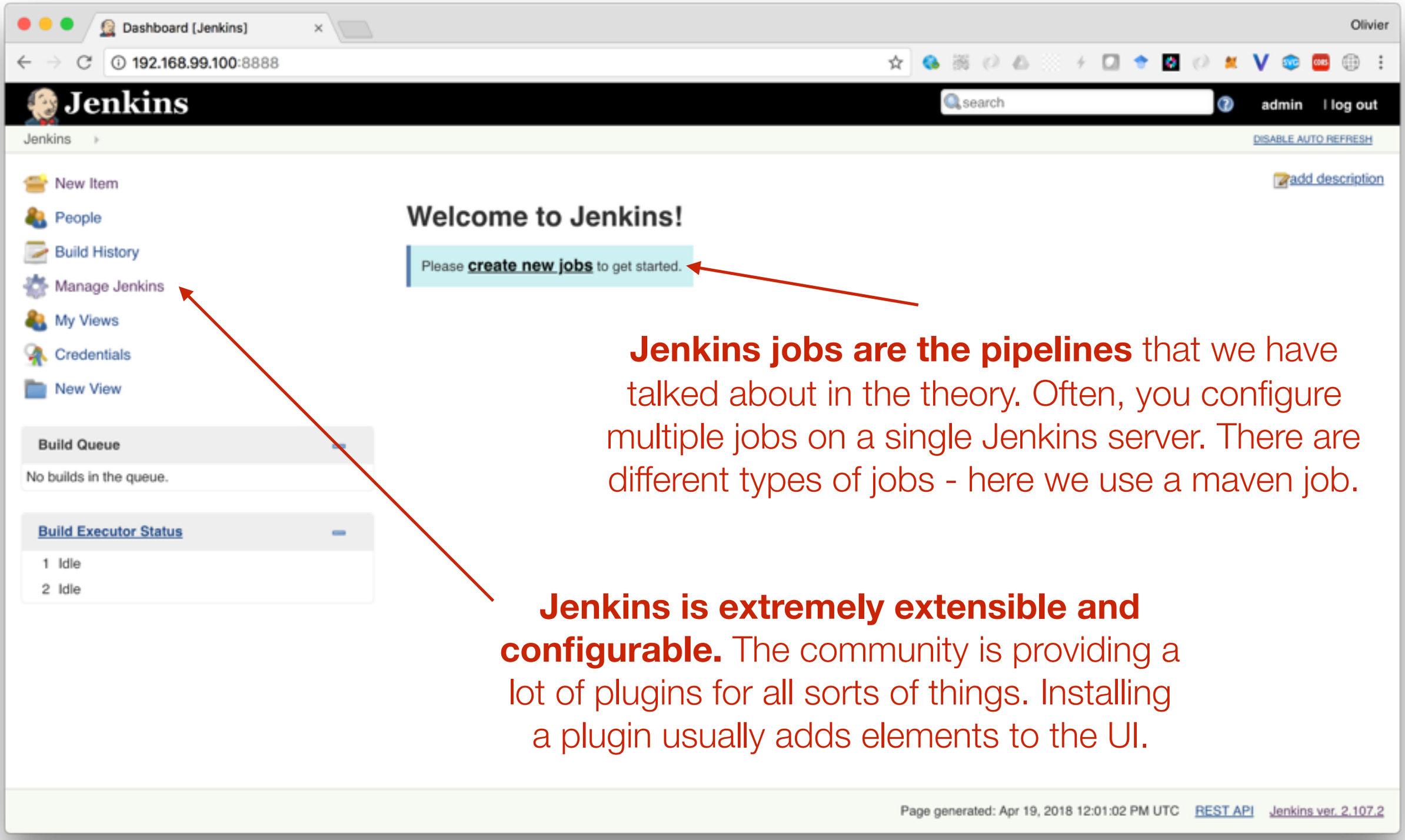

Jenkins [Jenkins]
Olivier
Not Secure 192.168.99.100:8888/login?from=%2F
Getting Started
Unlock Jenkins
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:
/var/jenkins_home/secrets/initialAdminPassword
Please copy the password from either location and paste it below.
Administrator password
the password from previous screen
Continue



Doing that in Provence... can be a challenge
(check the UI and the log on your Docker console)

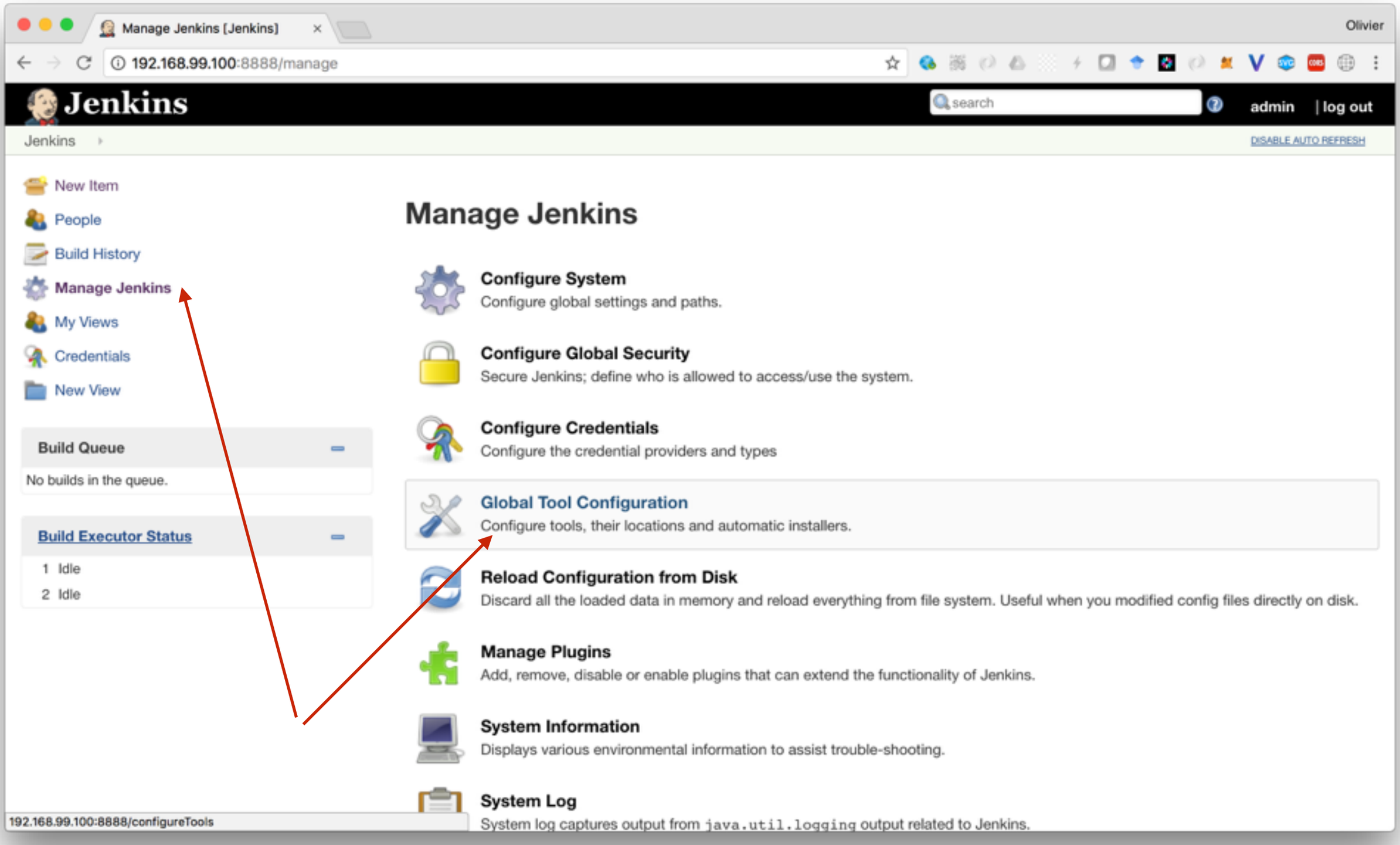

SetupWizard [Jenkins]
Olivier
Not Secure 192.168.99.100:8888
Getting Started
Create First Admin User
Username: admin
Password: •••••
Confirm password: •••••
Full name: admin
E-mail address: admin@a.com
Jenkins 2.107.2
Continue as admin
Save and Finish


SetupWizard [Jenkins]
Olivier
Not Secure
192.168.99.100:8888
Getting Started
Jenkins is ready!
Your Jenkins setup is complete.
Start using Jenkins
Jenkins 2.107.2

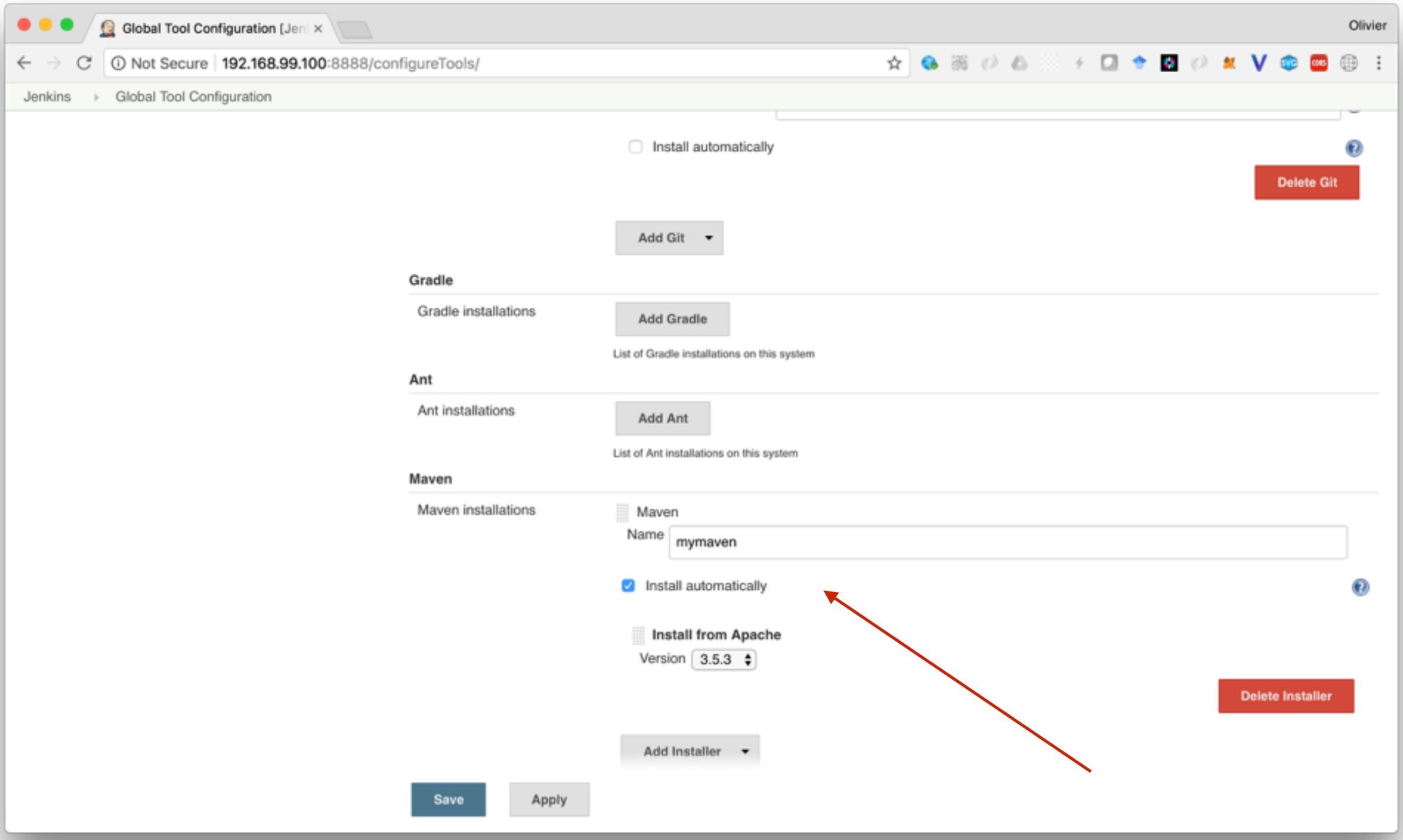


Jenkins jobs are the pipelines that we have talked about in the theory. Often, you configure multiple jobs on a single Jenkins server. There are different types of jobs - here we use a maven job.

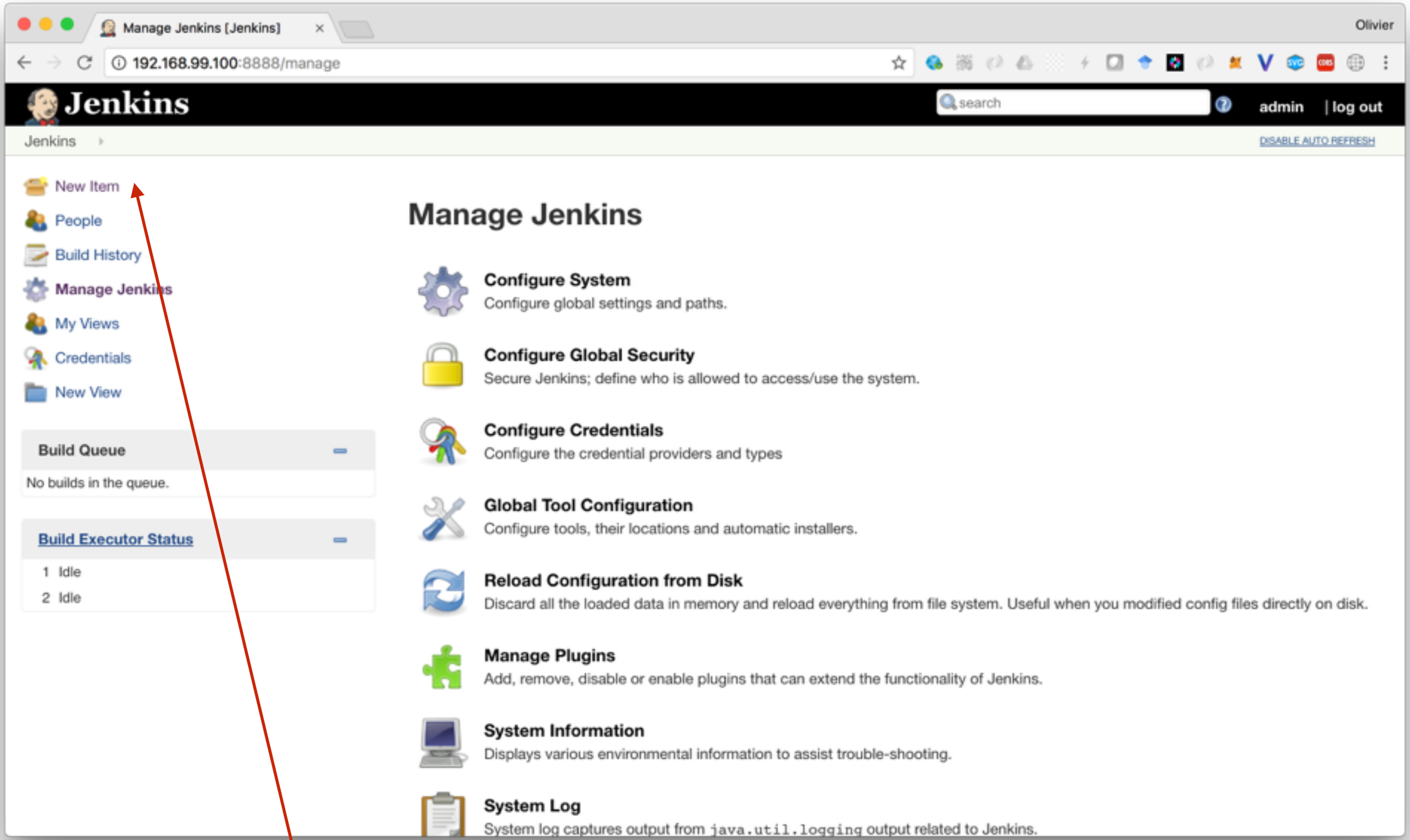
Jenkins is extremely extensible and configurable. The community is providing a lot of plugins for all sorts of things. Installing a plugin usually adds elements to the UI.



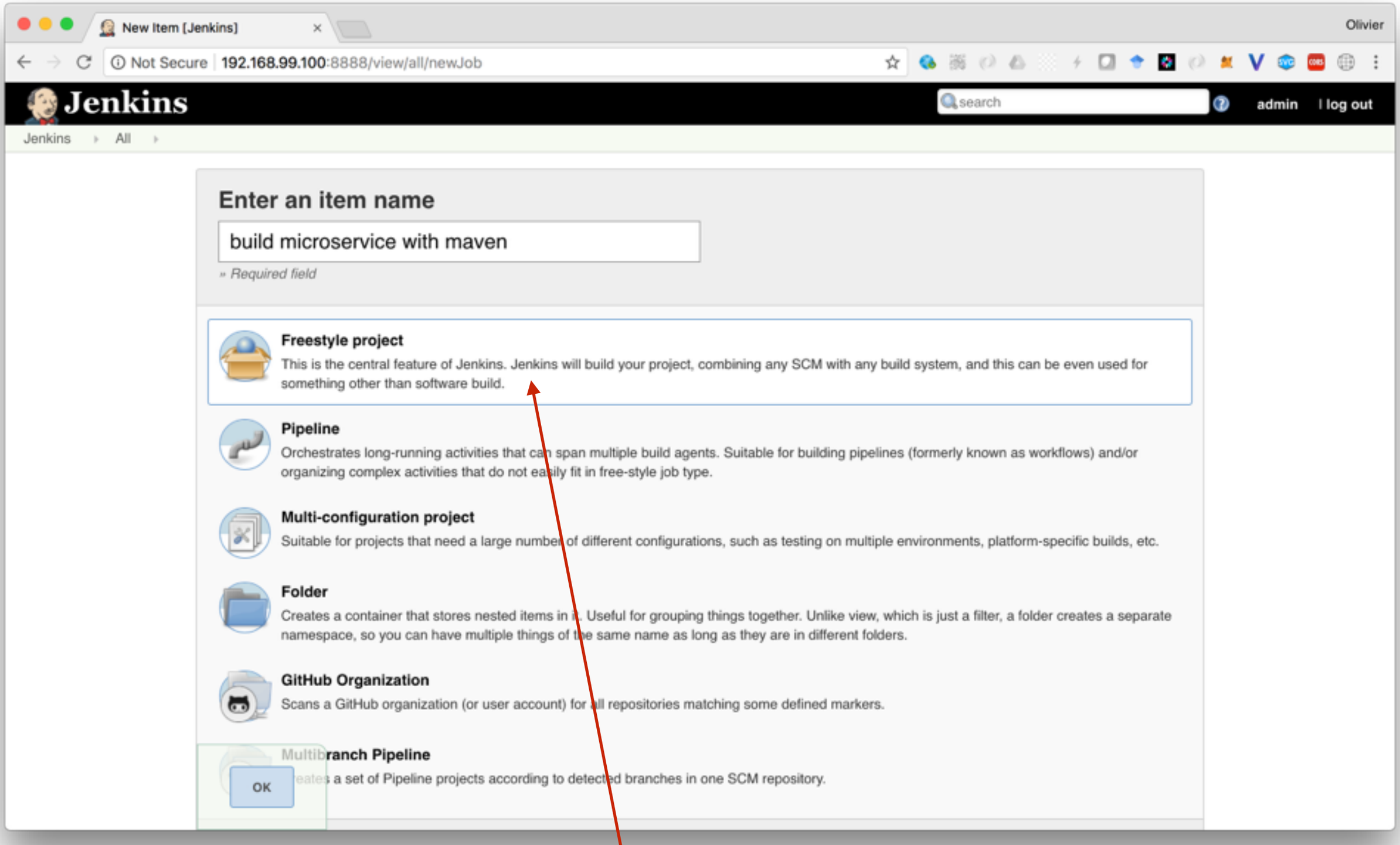
Jenkins will need to delegate (most of the) work to maven. So, we need to install maven in our Docker container. We could do that in the Dockerfile, but we can ask Jenkins to do that for us (via the Global Tool Configuration)



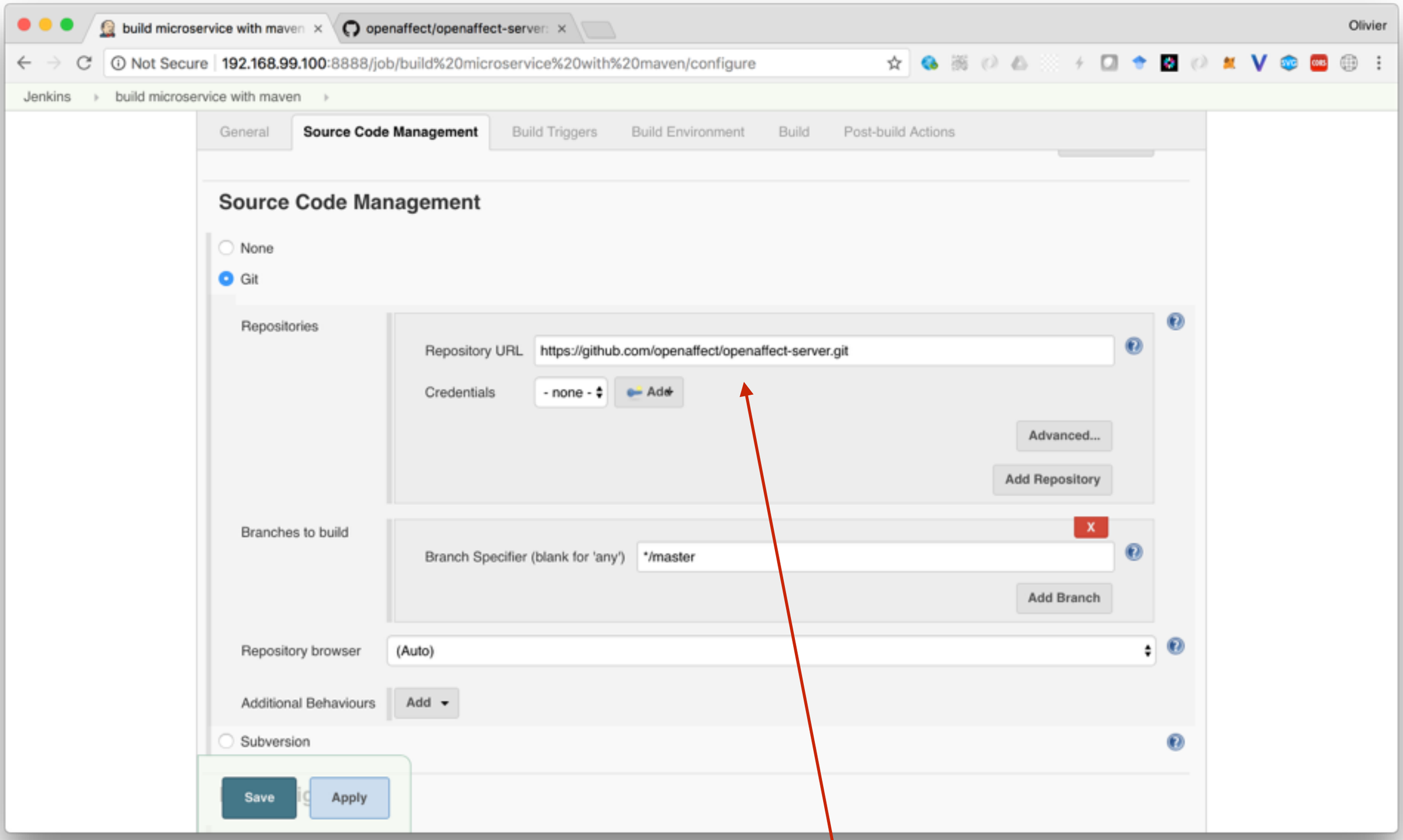
Jenkins will need to delegate (most of the) work to maven. So, we need to install maven in our Docker container. We could do that in the Dockerfile, but we can ask Jenkins to do that for us (via the Global Tool Configuration)



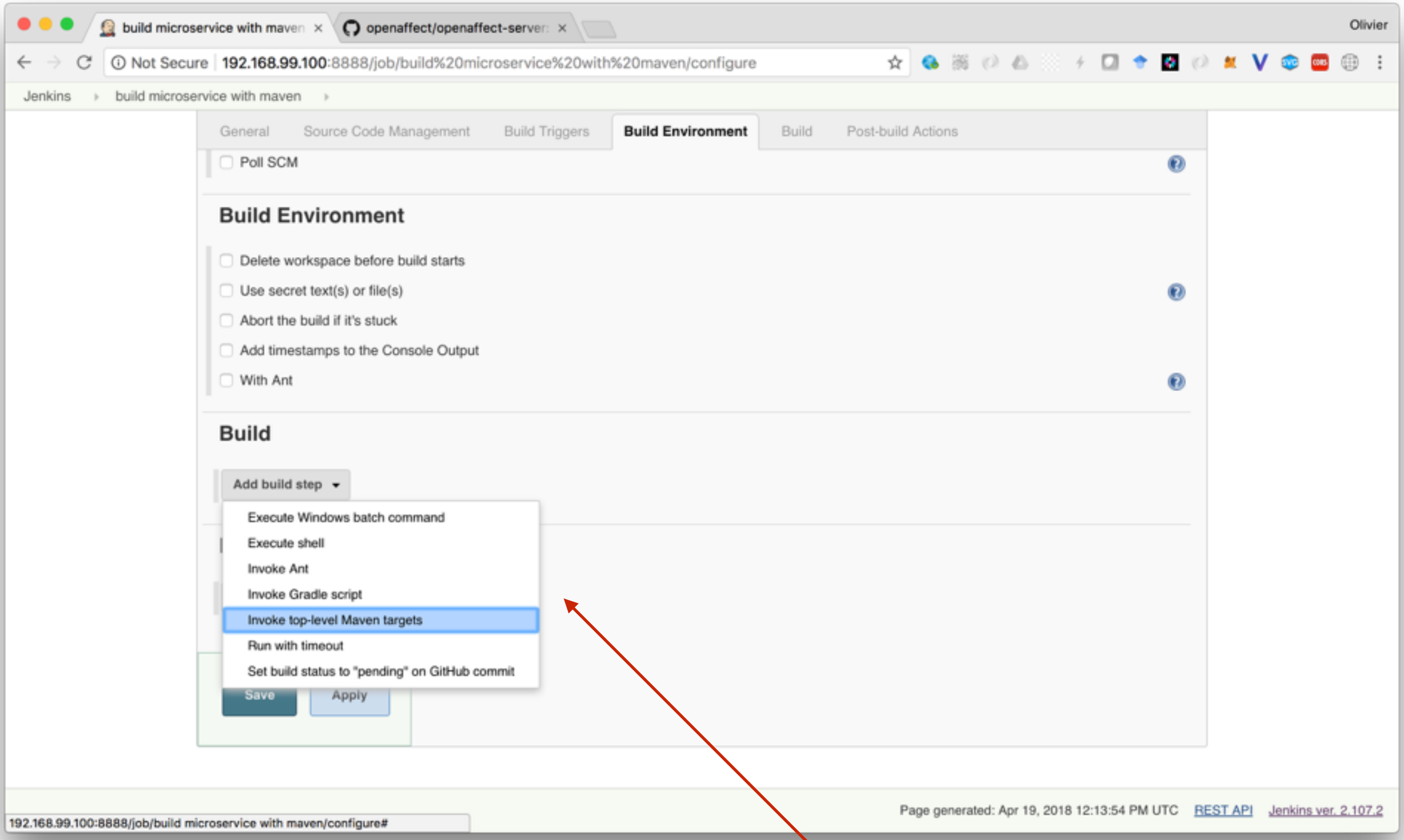
Let's create a new job (pipeline).



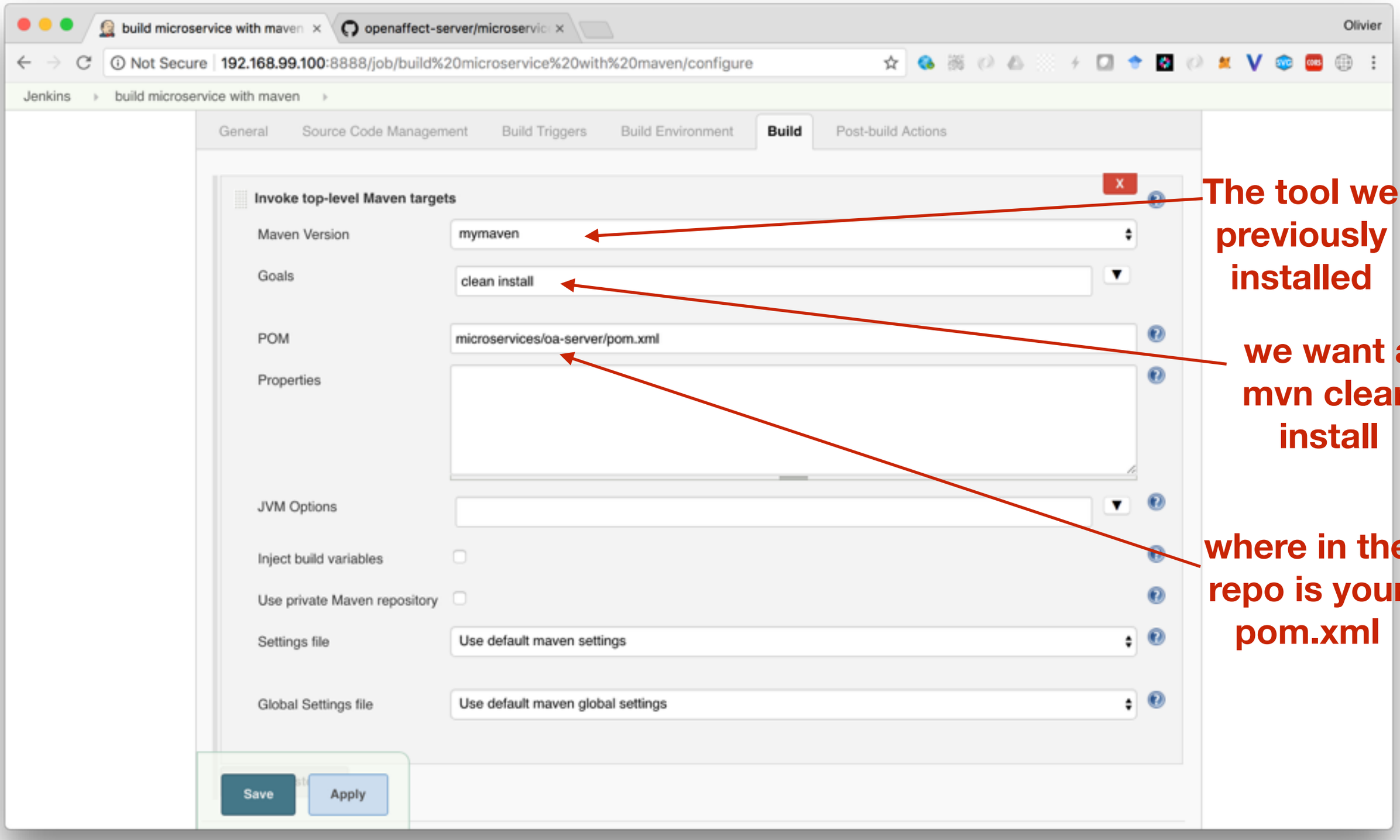
For simple maven builds, we use the “freestyle” job type



Where is the git repo where you have the code of your micro-service?



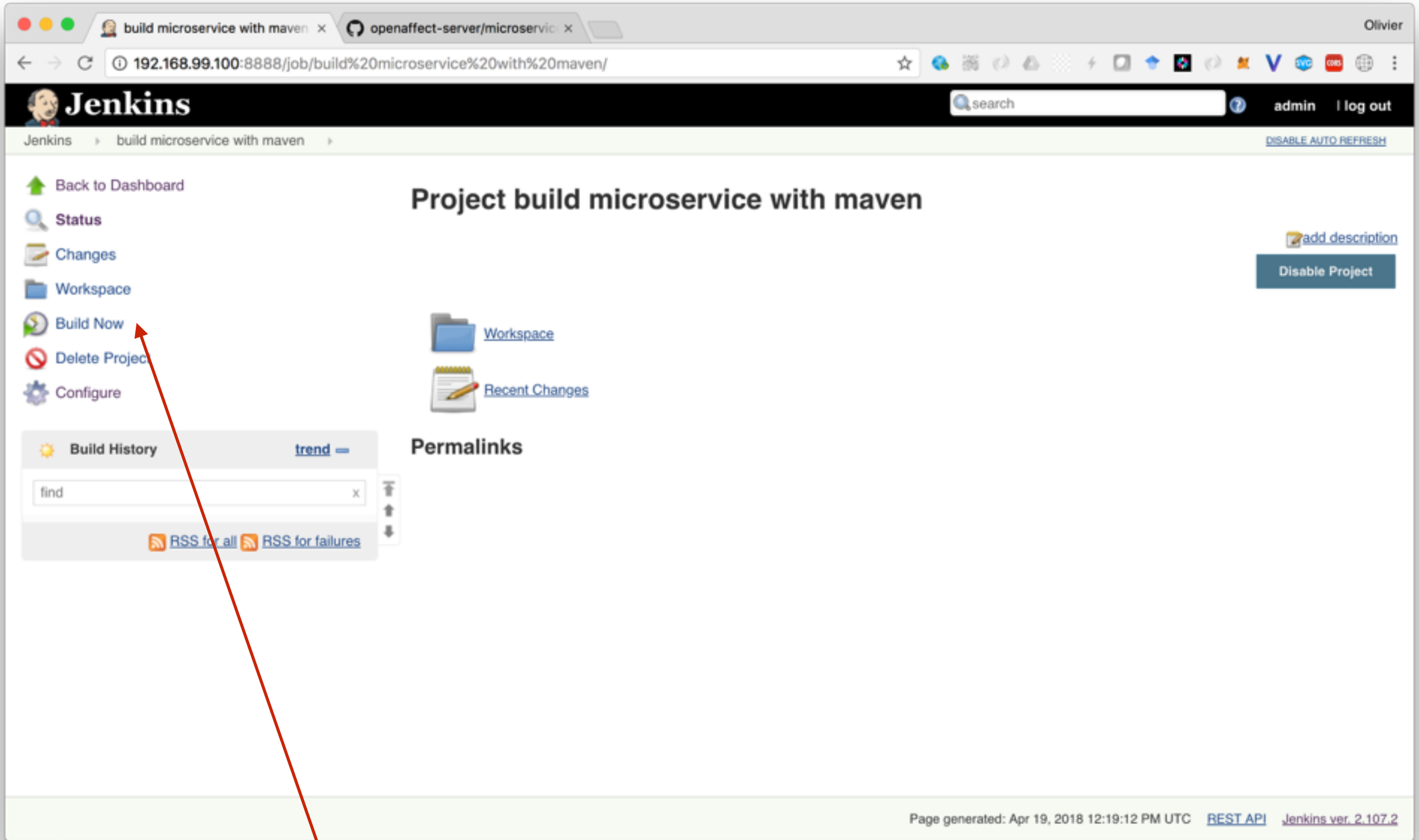
We have only one thing to do: launch maven (with a proper reference to the pom.xml file)



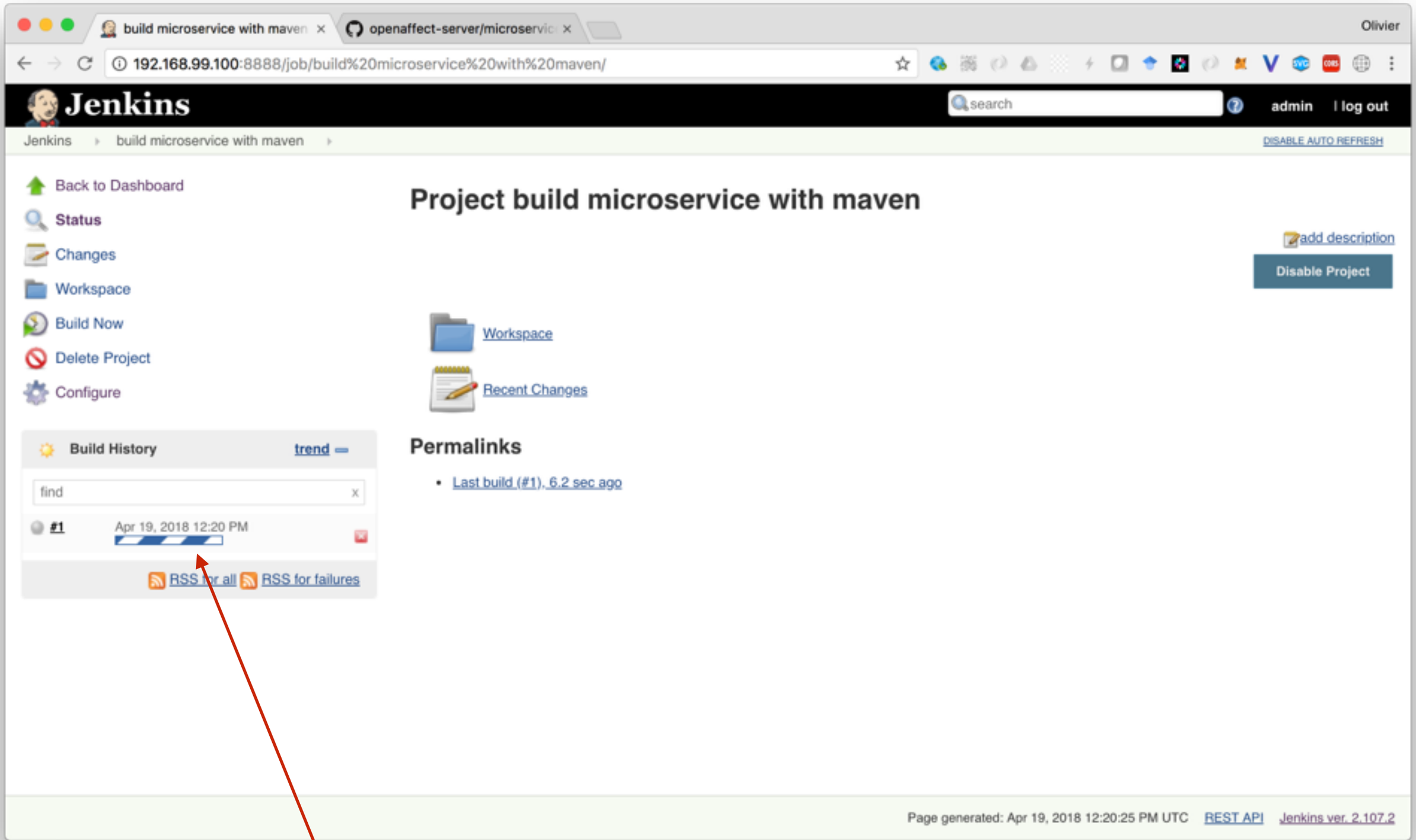
The tool we previously installed

we want a mvn clean install

where in the repo is your pom.xml



In this first example, we trigger the pipeline execution manually



The pipeline execution has started. We can click on that link to see the details (what is displayed in the console)

build microservice with maven × openaffect-server/microservic × Olivier

192.168.99.100:8888/job/build%20microservice%20with%20maven/1/console

Jenkins search admin | log out

Jenkins › build microservice with maven › #1

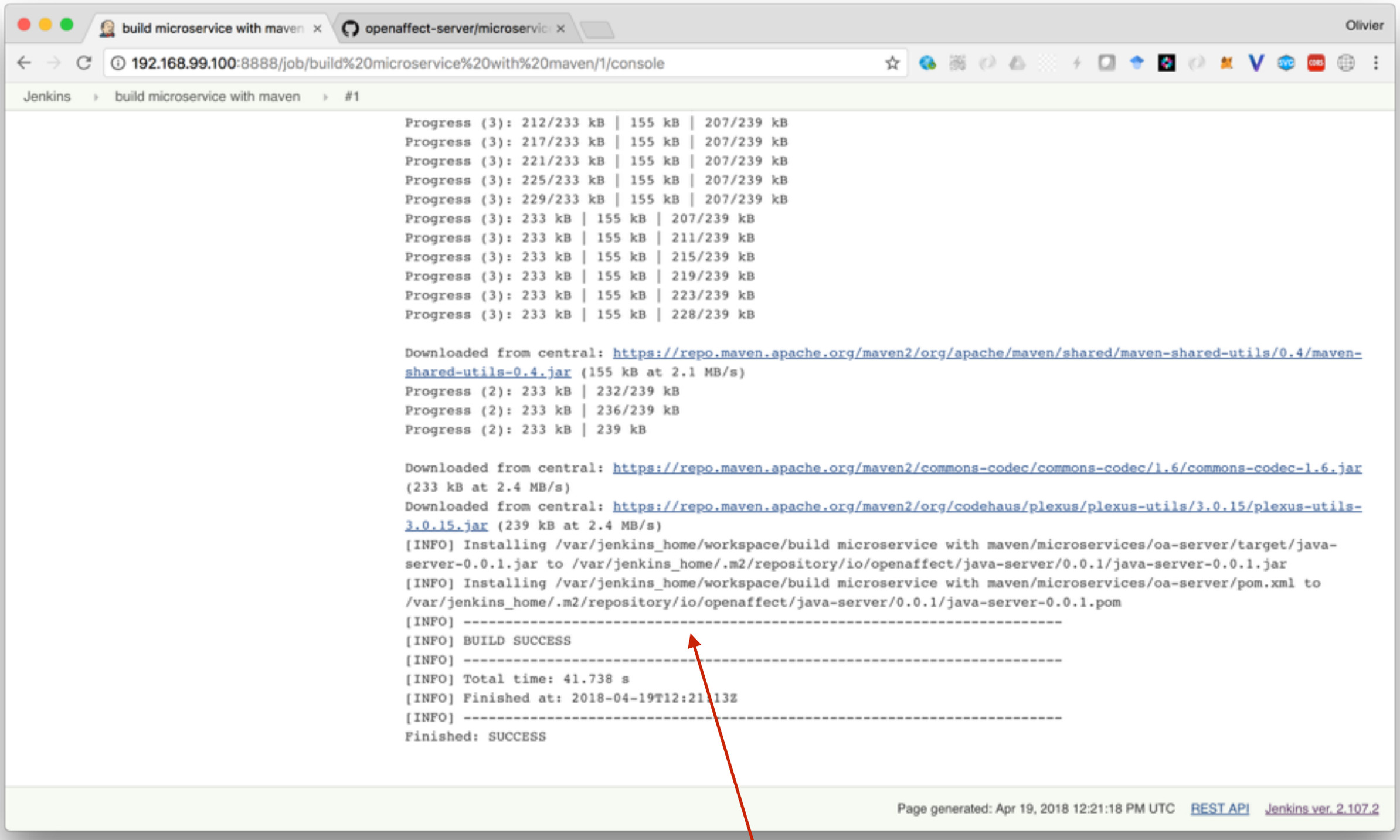
- Back to Project
- Status
- Changes
- Console Output
- View as plain text
- Edit Build Information
- Delete Build
- Git Build Data
- No Tags

Console Output

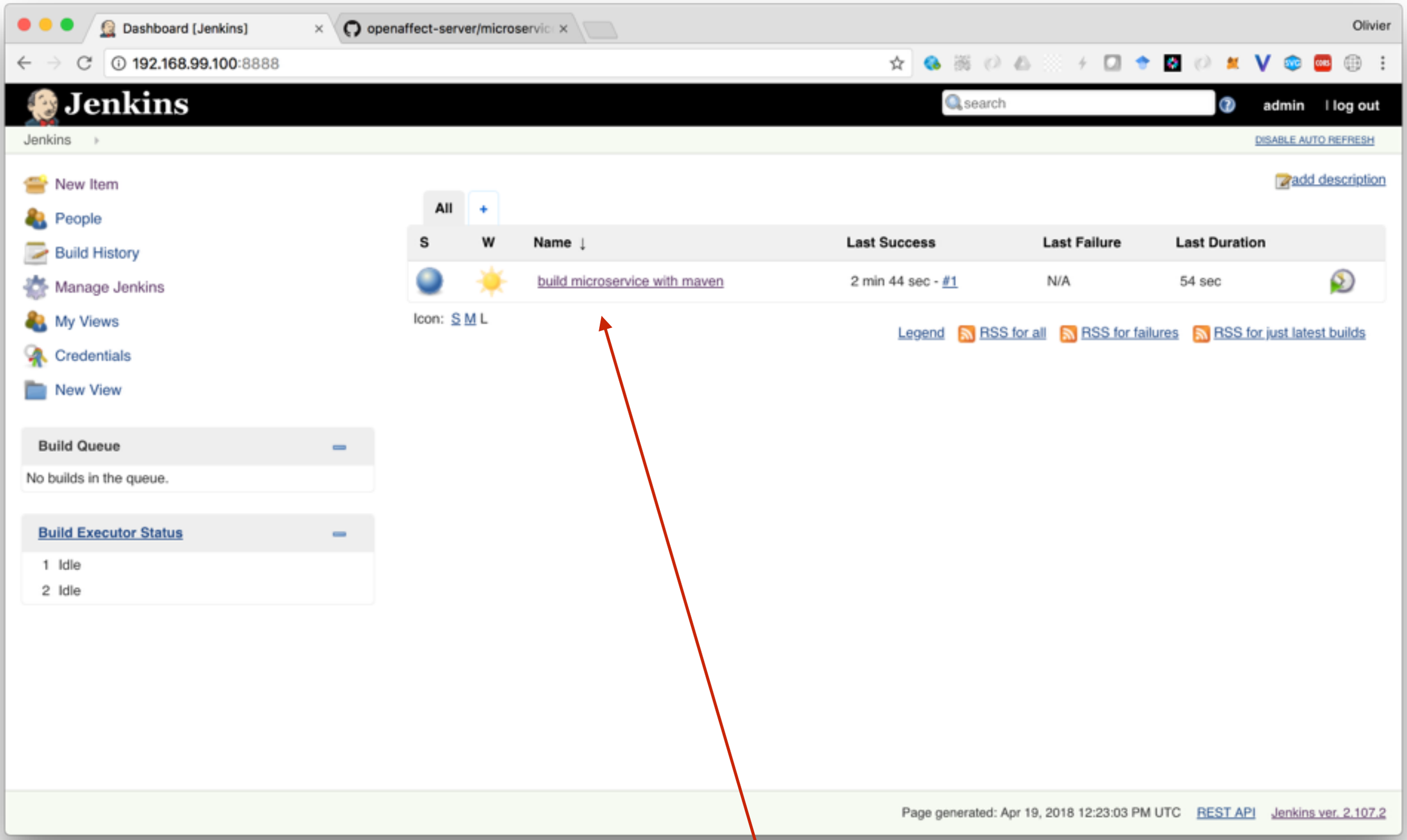
Skipping 861 KB.. Full Log

```
Downloaded from central: https://repo.maven.apache.org/maven2/org/slf4j/jcl-over-slf4j/1.5.6/jcl-over-slf4j-1.5.6.jar
(17 kB at 63 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/commons/commons-lang3/3.1/commons-lang3-
3.1.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/reporting/maven-reporting-
api/3.0/maven-reporting-api-3.0.jar (11 kB at 41 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/surefire/surefire-
api/2.18.1/surefire-api-2.18.1.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-repository-metadata/2.2.1/maven-
repository-metadata-2.2.1.jar (26 kB at 91 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-toolchain/2.2.1/maven-
toolchain-2.2.1.jar
Progress (2): 13 kB | 2.2/10 kB
Progress (2): 13 kB | 5.0/10 kB
Progress (2): 13 kB | 7.7/10 kB
Progress (2): 13 kB | 10 kB

Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-error-diagnostics/2.2.1/maven-
error-diagnostics-2.2.1.jar (13 kB at 46 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/plugin-tools/maven-plugin-
annotations/3.3/maven-plugin-annotations-3.3.jar
Progress (2): 10 kB | 4.1/316 kB
Progress (2): 10 kB | 7.7/316 kB
Progress (2): 10 kB | 12/316 kB
Progress (2): 10 kB | 16/316 kB
Progress (2): 10 kB | 20/316 kB
Progress (2): 10 kB | 24/316 kB
Progress (2): 10 kB | 28/316 kB
```

The pipeline execution has started. We can click on that link to see the details (what is displayed in the console)



You have seen last week on your terminal, when doing the Spring Boot tutorials



Back to the main page, we see that our last build was successful. Try to introduce a syntax error in your code. Commit and Push. Rebuild and you will see that Jenkins is unhappy.