

# Curso deploy automático e padrões de qualidade

---

GIT | JAVA JSF | MAVEN | ARTIFACTORY | JENKINS | SONARQUBE

INSTRUTOR RAJIV GEEVERGHESE (RAJIV@CNMP.MP.BR)

MARÇO/2019

# Cronograma

---

- Dia 1
  - Apresentação
  - Objetivos
  - Gitflow
  - Hello World
  - Deploy manual
- Dia 2
  - Aplicação 2.0.0
  - Estrutura padrão Maven
  - Project Object Model (POM)
  - Controle de dependências
  - Principais plug-ins
- Dia 3
  - Criação de profiles
  - Maven filtering
  - Deploy automatizado Jenkins
- Dia 4
  - Jenkins parametrizado
  - Artifactory como repositório de artefatos
  - Gerenciamento de repositórios
  - Publicação de libs locais
- Dia 5
  - Dashboard de qualidade com Sonarqube
  - Issues, rules, quality profile
  - Quality gate

# Curso deploy automático e padrões de qualidade

---

Dia 1

APRESENTAÇÃO | OBJETIVOS | GITLET | HELLO WORLD

Dia 2

APLICAÇÃO 2.0.0 | CONCEITOS MAVEN

Dia 3

MAVEN PROFILES | DEPLOY JENKINS

Dia 4

JENKINS PARAMETRIZADO | ARTIFACTORY

Dia 5

SONARQUBE | CONSIDERAÇÕES FINAIS | PRÓXIMOS PASSOS

# Dia 1

---

APRESENTAÇÃO | OBJETIVOS | GITFLOW | HELLO WORLD

# Quem sou eu?

---

- **Rajiv Geeverghese** - Analista de Desenvolvimento de Sistemas do CNMP
- Formado pelo Colégio Militar de Brasília (CMB)
- Graduado pela Universidade de Brasília (UnB) em Ciência da Computação
- Pós-graduado em Engenharia de Sistemas pela ESAB
- No serviço público desde 2009:
  - ANAC – 2009 a 2015
  - CNMP – 2015 a atualmente
- Linkedin: <https://www.linkedin.com/in/rajiv-geeverghese-83935bab/>
- Instagram: @rajiv.geeverghese

# E vocês?

---

- Nome
- Função na instituição
- Principais desafios a serem superados
- O que esperam do curso

# Cenário do ESMPU

---

- Todos os softwares estão no controle de versão? Existe algum sistema que está em produção e que a instituição não tem o código-fonte?
- Vocês conseguem saber as dependências dos projetos?
- Se o seu chefe pedisse agora para publicar uma nova versão do software com uma pequena alteração, quanto tempo levaria? Seria uma mudança simples?
- E se fosse para voltar à última versão estável?
- Vocês controlam a qualidade dos softwares e das entregas?

# Objetivos do curso

---

- Qual a necessidade de ter uma boa gestão de código-fonte?
  - Completude: todo o código-fonte está controlado
  - Rastreabilidade: gerenciar alterações e acompanhar a evolução do software
  - Isolamento: manter uma linha estável e isolar as mudanças
- Que outros ganhos vocês observam no controle de versão?
- Conhecem alguma história de um setor ou organização antes e depois do uso de controle de versão?

# Objetivos do curso

---

- Qual a necessidade de implementar deploy automático?
  - Autonomia: independência de outras áreas ou menos handoff
  - Agilidade: menor lead time
  - Processo: o deploy é automatizado e documentado
  - Confiabilidade: máquinas são bem mais consistentes do que nós, humanos!
- Que outros ganhos vocês observam na implementação do deploy automático?
- Conhecem alguma história de um setor ou organização antes e depois do uso de deploy automático e/ou outras implementações de devops?

# Objetivos do curso

---

- Qual a necessidade de controlar a qualidade das entregas?
  - Padrão: aderência às normas e aos padrões da instituição
  - Qualidade: software estável e menos sujeito a correções e alterações
  - Menor débito técnico: tratar o “saldo negativo” com a qualidade
- Que outros ganhos vocês observam na implementação do controle de qualidade?
- Conhecem alguma história de um setor ou organização antes e depois do uso de controles de qualidade?

# Estratégia do curso

---

- Projeto começando do zero
- A cada etapa/aula, o projeto se tornará mais completo
- Todas as configurações serão feitas durante o curso: nada “cairá do céu”
- Serão utilizados apenas softwares livres. Dessa forma, todo o “ecossistema” do curso poderá ser implantado na instituição de vocês.
- Serão abordados vários pontos considerados como **padrões de qualidade**. Muitos deles são utilizados atualmente no CNMP pela equipe de desenvolvimento.

# Estratégia do curso

---

- Principais etapas:
  - Criação do projeto maven
  - Implementação de uma aplicação web simples
  - Git: gestão do código-fonte e branches
  - Maven: gerenciamento do projeto Java, dependências e automação de build
  - Artifactory: repositório de artefatos
  - Jenkins: automação de build e deploy
  - Sonarqube: dashboard de qualidade com quality profile e quality gates

# Preparados?

---

MÃO NA MASSA!

# Devops

---

# Devops

---

- “How **development** and **operations** fits together and gets along... Actually work together and **aren't huge assholes to each other**” - John Allspaw (Flickr/Yahoo!) and Paul Hammond (Flickr) at “10+ Deploys Per Day: Dev and Ops Cooperation at Flickr”, 2009, Velocity O'Reilly Conference (<https://www.youtube.com/watch?v=LdOe18KhtT4>)



# Devops

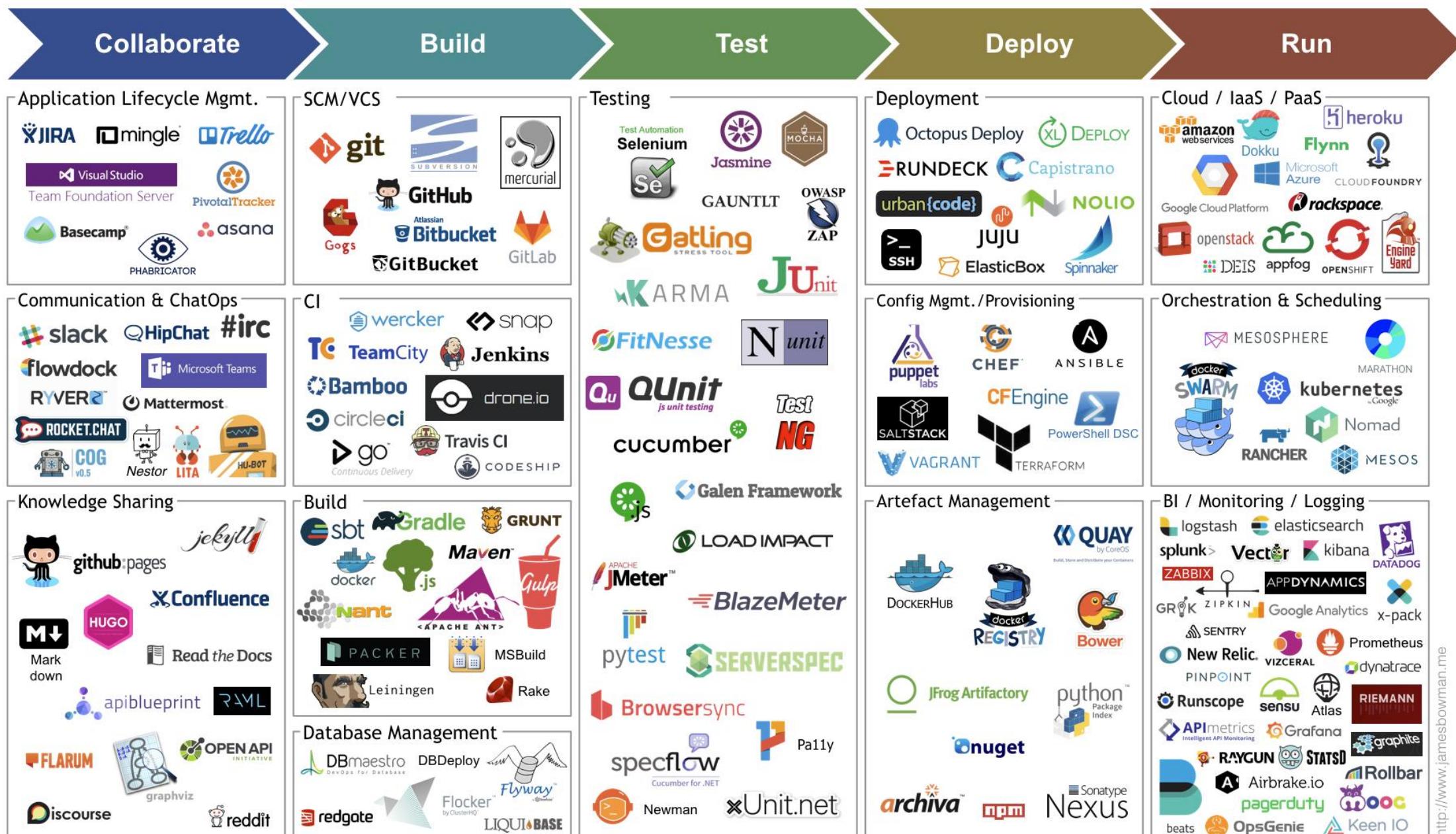
---

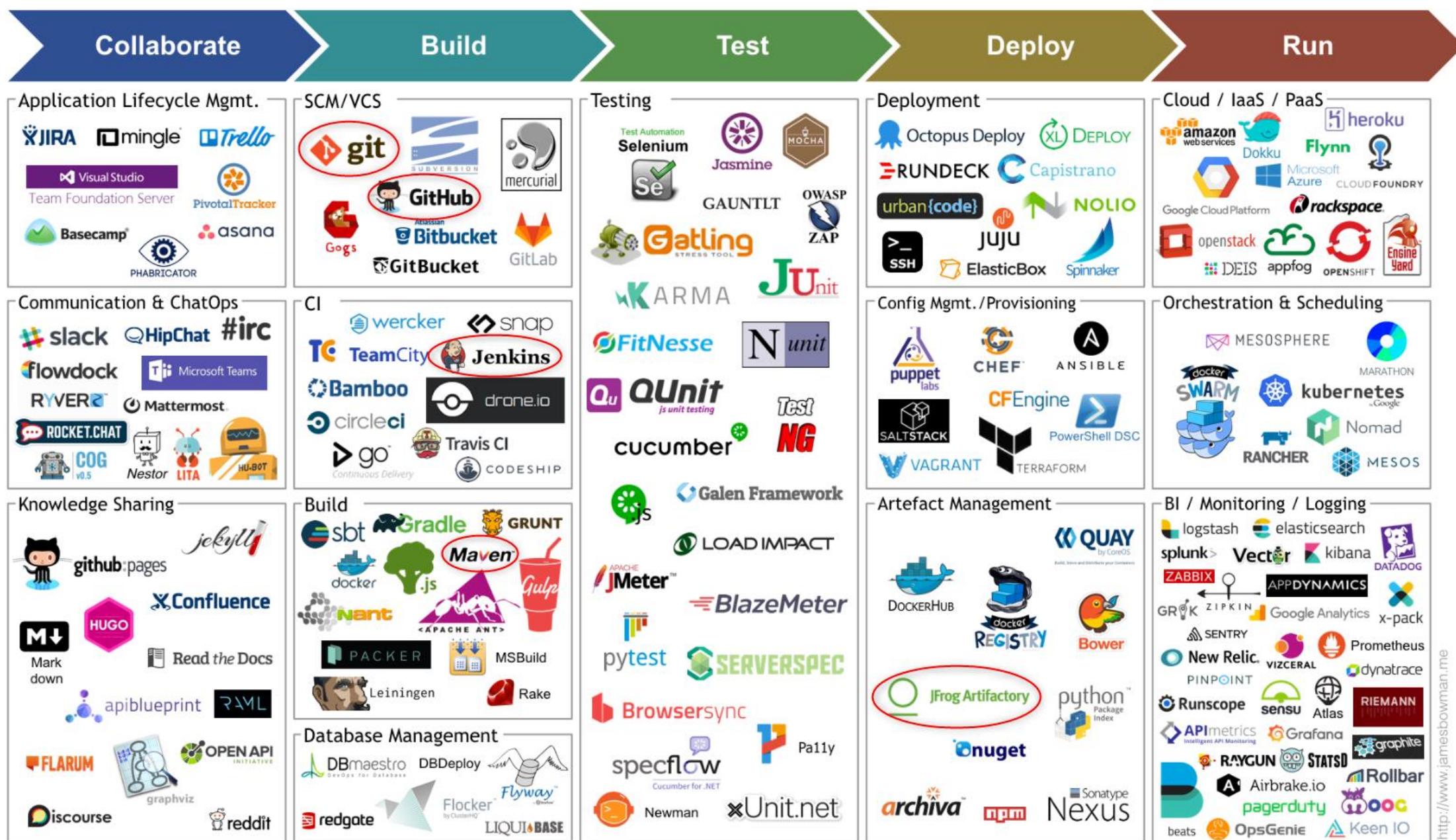
- O impacto dessa conferência resultou no primeiro DevOpsDay 2009, em Ghent, na Bélgica. Quando o evento terminou, a hashtag do twittter #DevOps se difundiu como o nome do próprio movimento.
  - <https://www.devopsdays.org/>
- Devops: termo criado para descrever um conjunto de práticas para integração entre as equipes de desenvolvimento de softwares, operações (infraestrutura ou sysadmin) e de apoio envolvidas (como controle de qualidade) e a adoção de processos automatizados para **produção rápida e segura** de aplicações e serviços.

# Devops

---







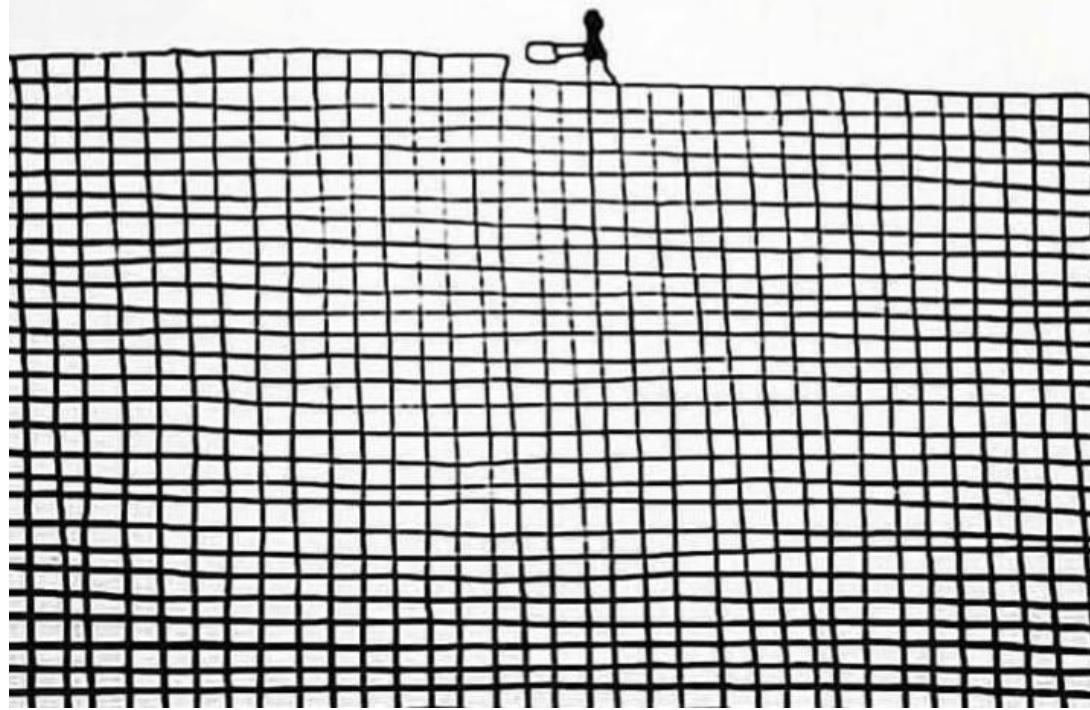
# Ferramentas x Mindset

---

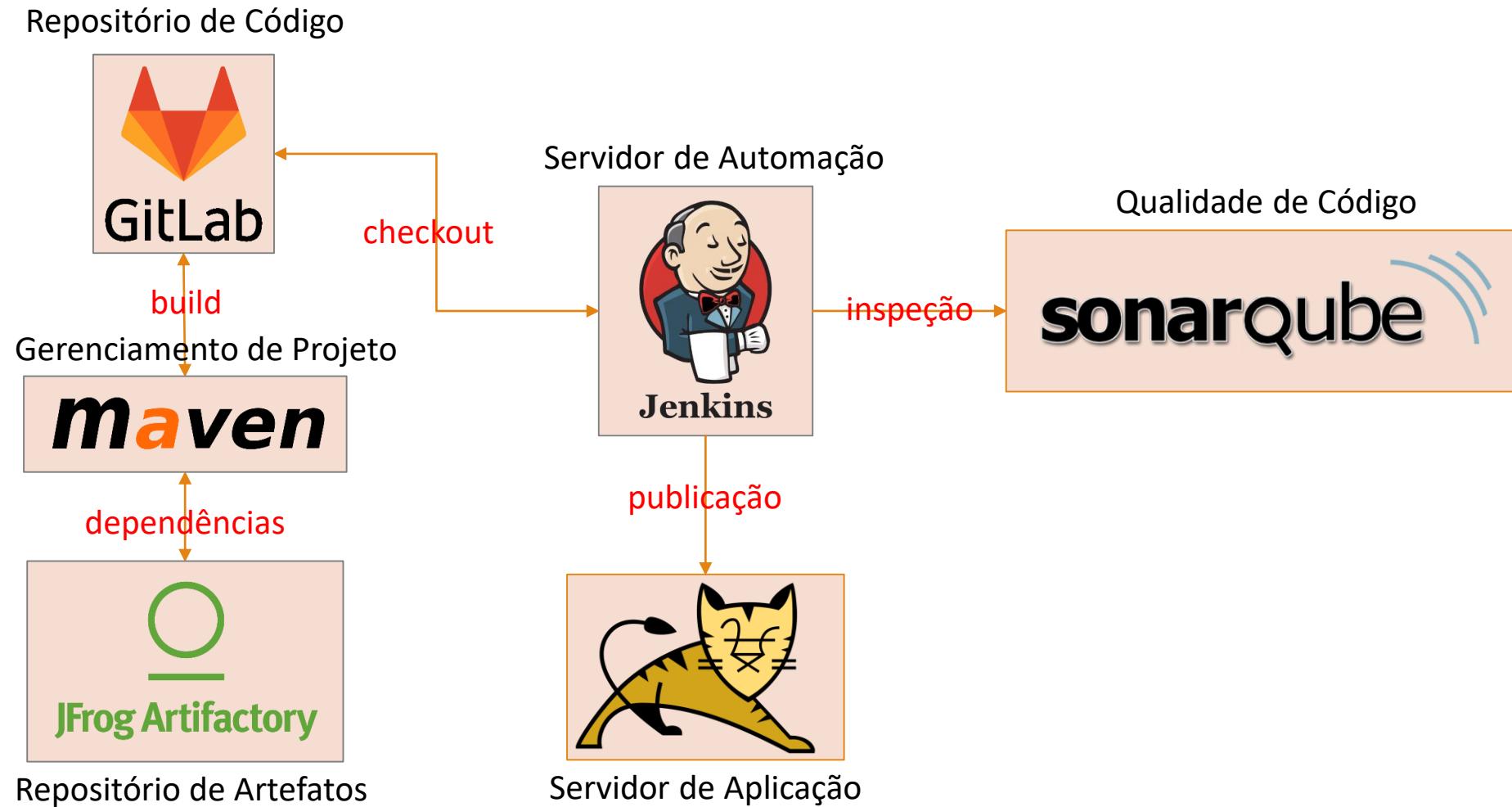
- As ferramentas são potencializadoras da implantação devops.
- Sem o mindset devops e um propósito muito claro, a instituição pode/vai criar um novo problema ao invés de aumentar a maturidade.
- Qual o ideal?



o superpoder de  
fazer um pouco  
todos os dias



# Arquitetura Devops Opensource



# Workspace

---

# Estrutura do workspace

- Estrutura padronizada que facilita a organização do projeto.
    - + workspace
    - |\_\_git repositório git e workspace eclipse
    - |\_\_ide pasta de IDEs
    - |\_\_server pasta dos servidores
    - |\_\_software pasta dos softwares originais (download)
    - |\_\_tool pasta de ferramentas e outros softwares com exceção de ide e server
  - @TODO:
    - Criar estrutura de diretórios
    - Descompactar: eclipse, tomcat, maven
    - Maven: criar variável de ambiente MAVEN\_HOME e adicionar bin no PATH
    - Java JDK 8: criar variável de ambiente JAVA\_HOME e adicionar bin no PATH

# Servidores

---

- Tomcat Desenvolvimento
  - <http://localhost:8080/curso-deploy>
- Tomcat Produção
  - <http://localhost:8180/curso-deploy>
- Jenkins
  - <http://localhost:8180/jenkins>
- Artifactory
  - <http://localhost:8081/artifactory>
- Sonarqube
  - <http://localhost:9000>

# Servidor de produção

---

- Vamos criar uma nova instância tomcat para simular um servidor de produção. Ele que rodará a nossa aplicação e também o jenkins.
  - Sugestão de nome: “apache-tomcat-8.5.38-producao”
- Editar o arquivo <TOMCAT\_HOME>\conf\server.xml e alterar as portas dos protocolos HTTP/1.1, AJP/1.3 e SHUTDOWN para não conflitar com a primeira instância do tomcat (desenvolvimento).

```
<Server port="8105" shutdown="SHUTDOWN">

<Connector port="8180" protocol="HTTP/1.1"

<Connector port="8109" protocol="AJP/1.3" redirectPort="8443" />
```

# Servidor de produção

---

- Aumentar o tamanho do cache do servidor de “produção” (arquivo \conf\context.xml) para 100MB:

```
<Context>
  (...)

  <Resources cacheMaxSize="104857600" />

</Context>
```

- Executar a instância tomcat:

- cd <TOMCAT\_HOME>\bin
- catalina.bat configtest
- catalina.bat start

# Github

---

## Create a new repository

A repository contains all project files, including the revision history.

Owner                  Repository name \*

 rajivunb / curso-deploy ✓

Great repository names are short and memorable. Need inspiration? How about [super-engine](#)?

Description (optional)

Curso de deploy automático e padrões de qualidade ministrado no ESMPU

 Public  
Anyone can see this repository. You choose who can commit.

 Private  
You choose who can see and commit to this repository.

Initialize this repository with a README  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: Java ▾ | Add a license: None ▾ 

**Create repository**

## Conta github

- <https://github.com/>
- Url-friendly
- README.md
- .gitignore Java

## Branch protection rule

Branch name pattern  
master

Applies to 1 branch  
master

Rule settings

**Protect matching branches**  
Disables force-pushes to all matching branches and prevents them from being deleted.

**Require pull request reviews before merging**  
When enabled, all commits must be made to a non-protected branch and submitted via a pull request with the required number of approving reviews and no changes requested before it can be merged into a branch that matches this rule.  
Required approving reviews: 1 ▾

**Dismiss stale pull request approvals when new commits are pushed**  
New reviewable commits pushed to a matching branch will dismiss pull request review approvals.

**Require review from Code Owners**  
Require an approved review in pull requests including files with a designated code owner.

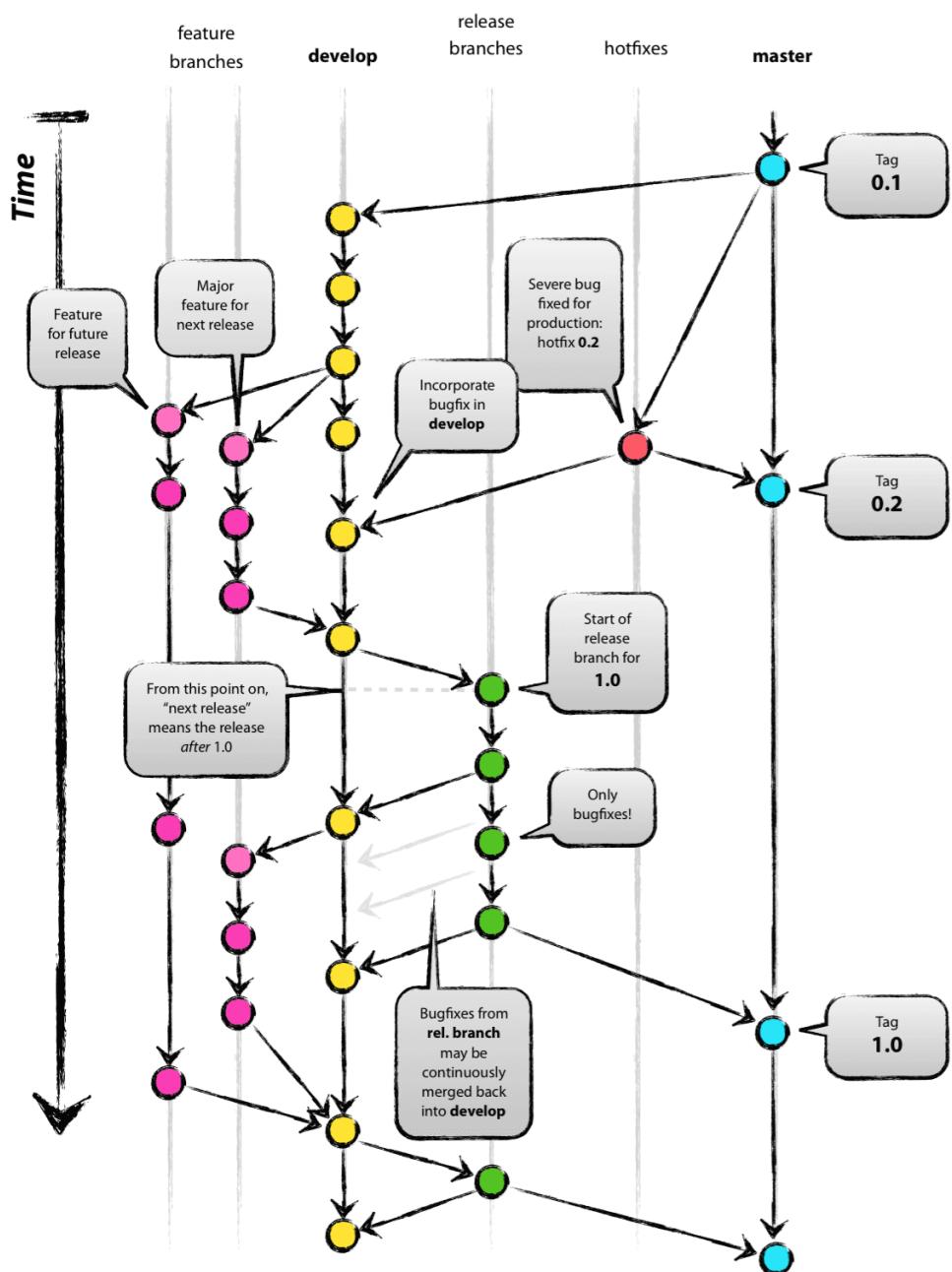
**Require status checks to pass before merging**  
Choose which [status checks](#) must pass before branches can be merged into a branch that matches this rule. When enabled, commits must first be pushed to another branch, then merged or pushed directly to a branch that matches this rule after status checks have passed.

**Require signed commits**  
Commits pushed to matching branches must have verified signatures.

**Include administrators**  
Enforce all configured restrictions for administrators.

# Branch Protection

- Proteger a branch master para evitar pushes diretos
- Settings > Branches > Branch protection rules > Add rule
- Marcar os itens:
  - Require pull request reviews before merging
  - Include administrators



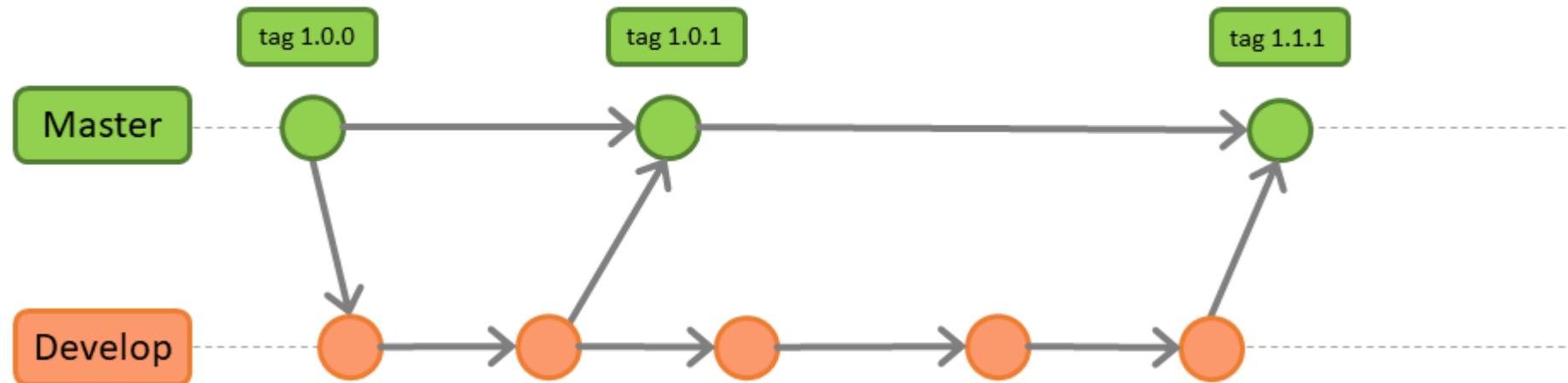
# Gitflow

- Criado em 2010 por Vincent Driessen, um engenheiro de software e consultor holandês.
- É um modelo de desenvolvimento que se popularizou por conta do uso de branches específicas que permitem isolamento e estabilidade.
  - <https://nvie.com/posts/a-successful-git-branching-model/>

# Gitflow mínimo

---

- Branch develop: commits de desenvolvimento
- Branch master: apenas código estável e pronto para produção



The screenshot shows a GitHub repository page for 'curso-deploy'. The top navigation bar includes links for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. The repository name is 'Curso de deploy automático e padrões de qualidade ministrado no ESMPU'. Below the repository name, there are summary statistics: 1 commit, 1 branch, 0 releases, and 1 contributor. A 'Manage topics' link is also present.

The main area displays the repository's history. It shows two 'Initial commit' entries, both made 34 minutes ago. A 'Create new file', 'Upload files', and 'Find file' button are available, along with a green 'Clone or download' button. On the left, a sidebar titled 'Switch branches/tags' lists 'develop' (selected) and 'master'. It also includes 'Create branch: develop from 'master'' and 'Branches' and 'Tags' tabs.

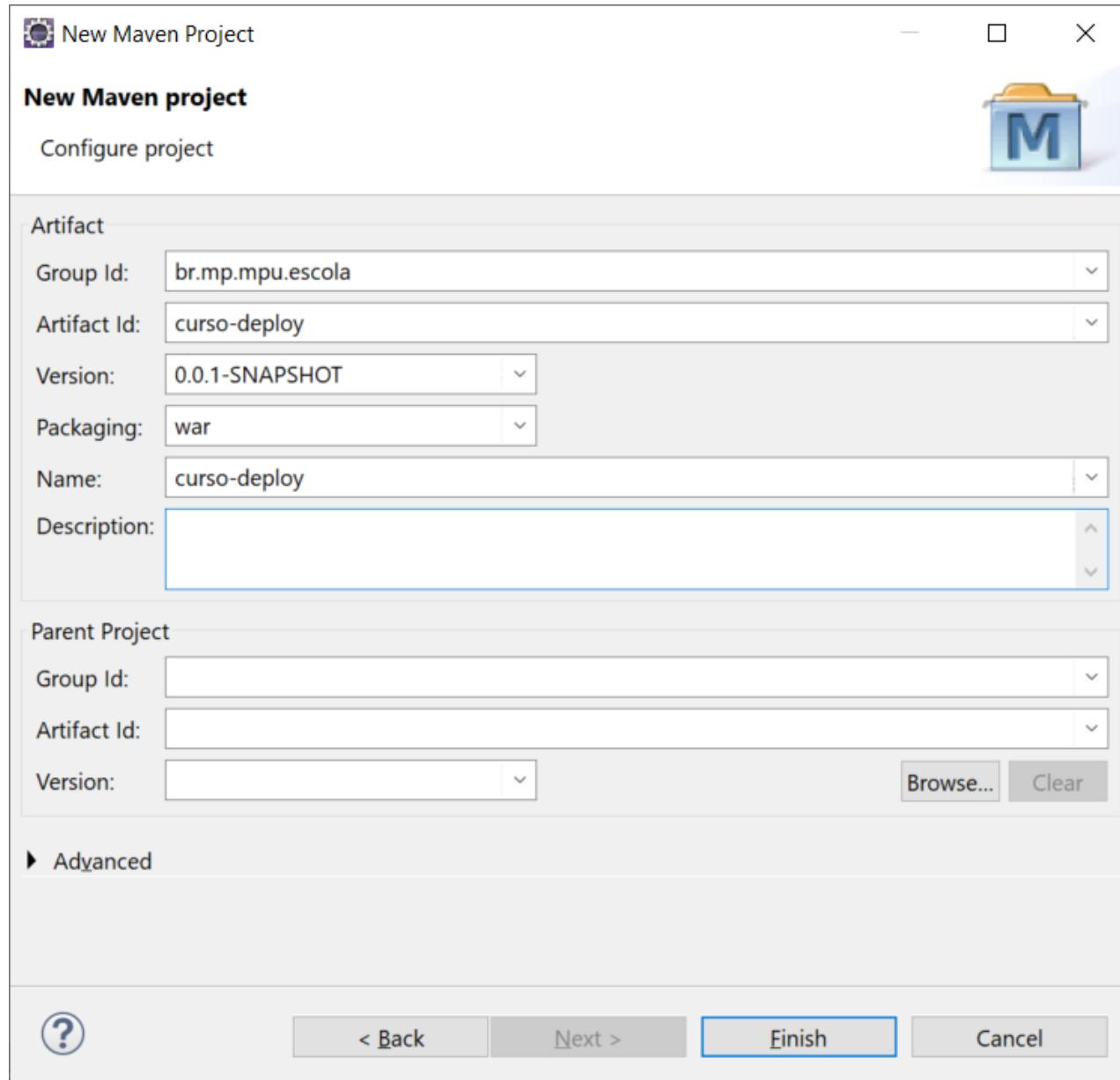
The repository title 'curso-deploy' is displayed prominently at the bottom of the page, along with the same repository information.

# Branch Develop

- Criar branch develop
- Clonar repositório git na pasta “git” e dar checkout da branch develop

# Aplicação 1.0.0

---



# New Maven Project

- Configurar Eclipse
  - Encoding
  - Java > Installed JREs
  - Add server | Targeted Runtime
- Criar novo projeto maven (skip archetype selection)
- Renomear pasta para “app”
- Deployment Descriptor > Generate Deployment Descriptor Stub

# Estrutura padrão GIT/Maven

---

```
projeto-padrão nomes url-based: minúsculos, sem caracteres especiais e separados por -
| README.md arquivo markdown padrão do GIT com um resumo do projeto

+---app
|   \---src
|       +---main
|           +---filters      uso de filters para propriedades por ambiente (profiles)
|               |   desenvolvimento.properties
|               |   homologacao.properties
|               |   local.properties
|               |   producao.properties
|
|           +---java        arquivos .java
|           +---resources   arquivos de configuração, tipicamente .properties e .xml
|               \---webapp    front-end
|           \---test        arquivos de teste, caso exista
|
|   \---test
|       +---filters
|       +---java
|       \---resources
|
\---doc
    +---modelo-de-dados
    |   nome-do-schema.pdm     modelo powerdesigner
    |
    \---script
        001-cria-tabelas.sql
        002-popula-tabelas-dominio.sql

```

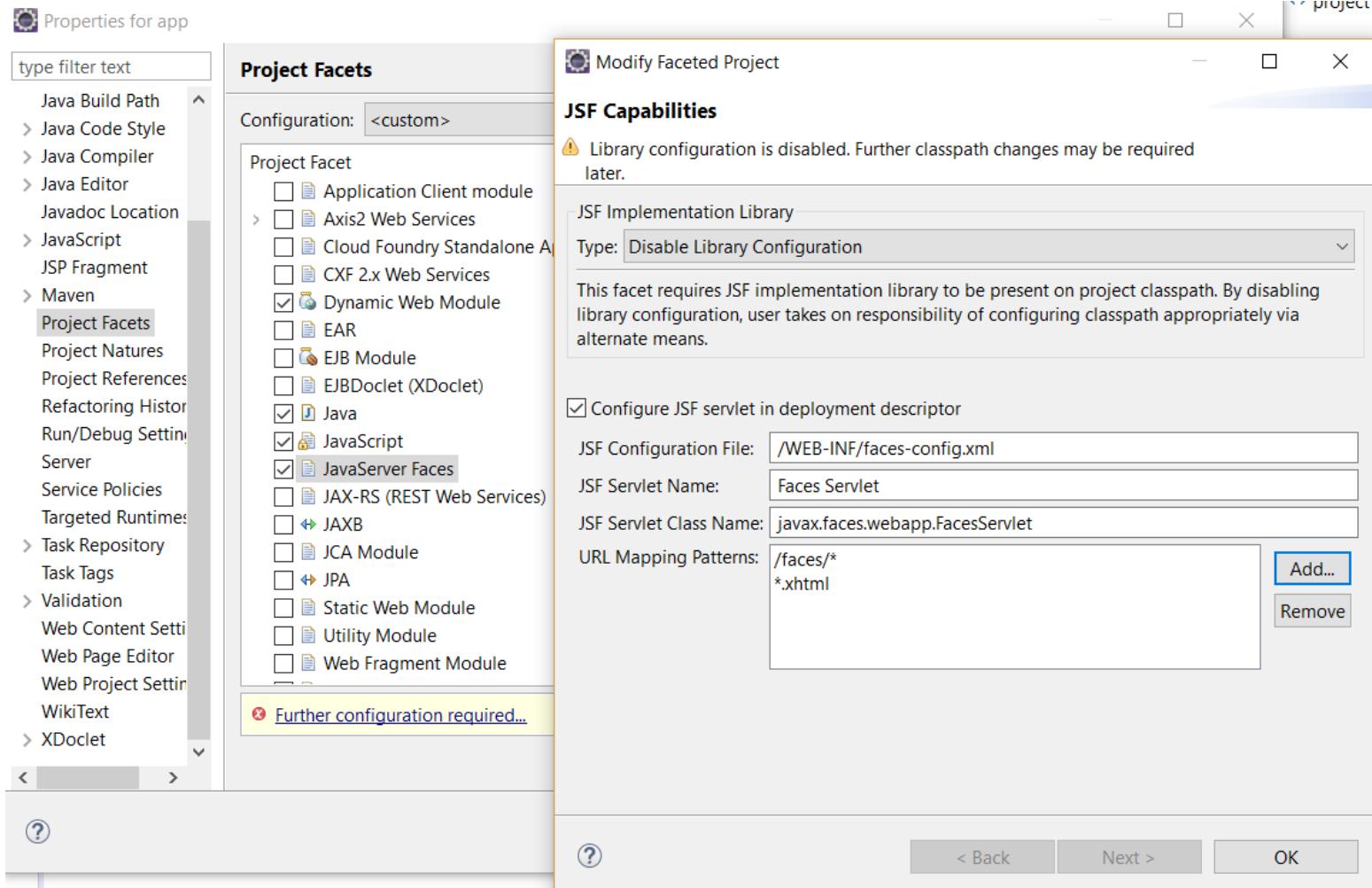
# Project Properties

---

- Configurar encoding, versão java no POM e nome do pacote build
  - Após a configuração, aperte Alt+F5 para chamar o “Update Maven Project”

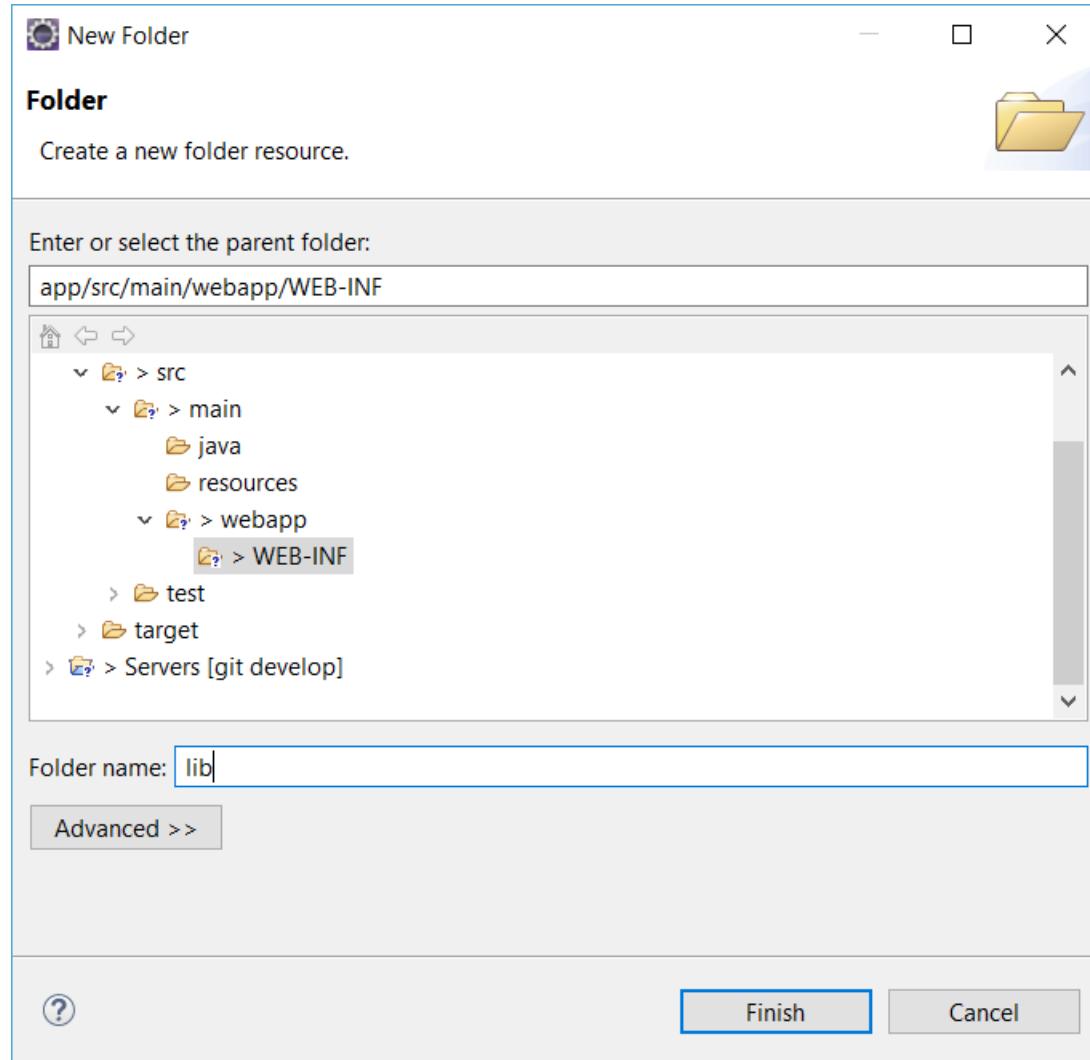
```
<properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
</properties>

<build>
    <finalName>${project.artifactId}</finalName>
</build>
```



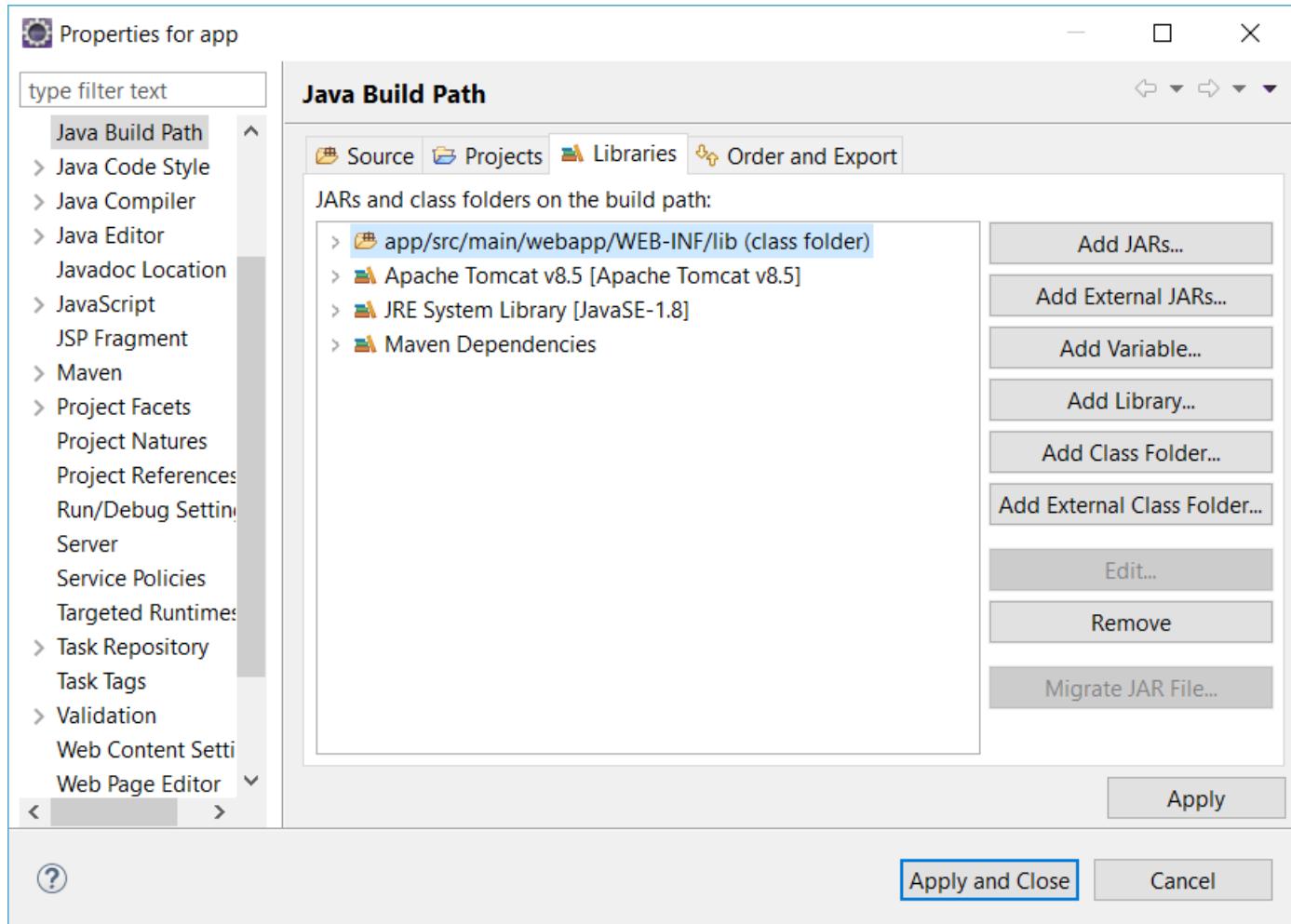
# Project Facets

- Habilitar facets do JSF 2.2
- Seleccionar “Disable Library Configuration”
- Adicionar \*.xhtml no URL Mapping Patterns



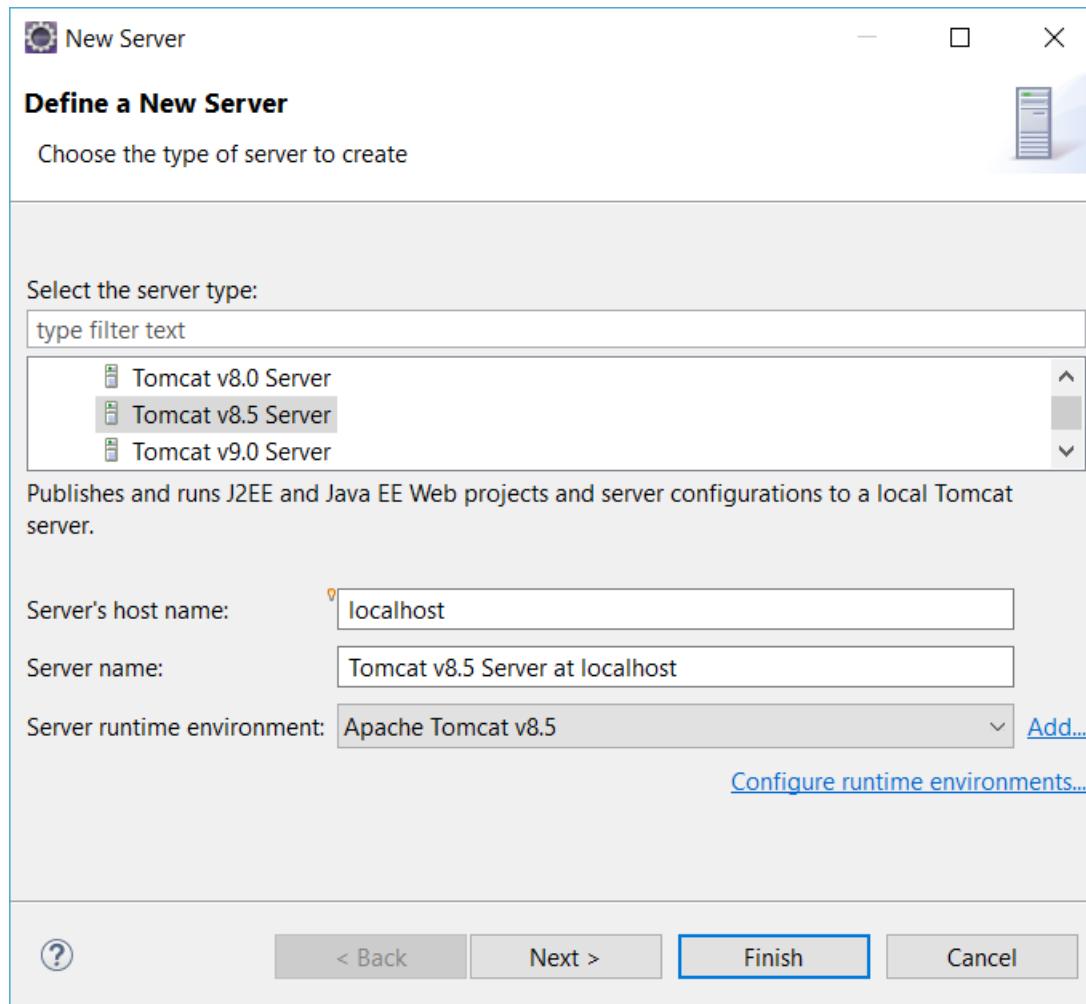
# Lib Folder

- Adicionar libs do JSF 2.2 no projeto
  - Criar pasta WEB-INF/lib
  - Copiar os arquivos jar jsf-api e jsf-impl



# Class Folder

- Adicionar a pasta lib como class folder



# New Server: Tomcat

- Adicionar o servidor Tomcat 8.5

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://java.sun.com/xml/ns/javaee"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
  http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
  version="2.5">

  <display-name>curso-deploy</display-name>

  <welcome-file-list>
    <welcome-file>index.xhtml</welcome-file>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>

  <servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>/faces/*</url-pattern>
    <url-pattern>*.xhtml</url-pattern>
  </servlet-mapping>

</web-app>
```

# Web.xml

---

- Configurar web.xml

# Index.xhtml

---

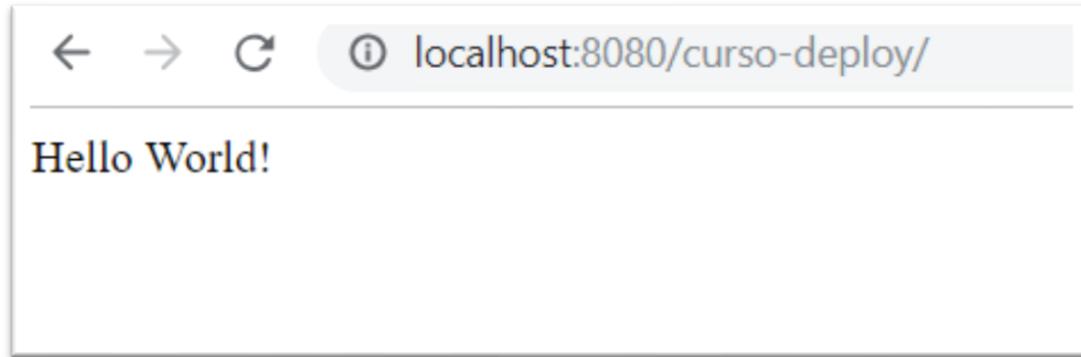
- Criar arquivo index.xhtml dentro da pasta webapp

```
<html>  
  <head>  
    <title>Curso de Deploy</title>  
  </head>  
  
  <body>  
Hello World!  
  </body>  
  
</html>
```

# Primeira aplicação

---

- Nossa super aplicação!



- Hora de commitar e gerar a versão 1.0.0!
  - Uso do **semantic version 2.0** (<https://semver.org/>)
  - Uso de **pull request**, pois a branch master está protegida

# Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

# Pull request

- Base: master
  - Compare: develop

## Projeto inicial hello world. #1

 Open rajivunb wants to merge 1 commit into `master` from `develop`

Conversation 0 Commits 1 Checks 0 Files changed 17

 rajivunb commented just now

Owner + ...

Primeira versão estável do projeto.

 Projeto inicial hello world. 31c80c5

Add more commits by pushing to the `develop` branch on [rajivunb/curso-deploy](#).

 Continuous integration has not been set up  
Several apps are available to automatically catch bugs and enforce style.

 This branch has no conflicts with the base branch  
Merging can be performed automatically.

**Merge pull request** ▾ You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

# Merge pull request

Releases

Tags

1.0.0

@ Target: master

Excellent! This tag will be created from the target when you publish this release.

Hello World

Write

Preview

Markdown supported

Primeira versão do projeto maven com facet JSF 2.2 e uma página inicial simples Hello World.

Attach files by dragging & dropping, selecting them, or pasting from the clipboard.

# Tag

- Primeira versão do projeto
- Uso do semver 2.0
- Versão 1.0.0:
  - 1: MAJOR version
  - 0: MINOR version
  - 0: PATCH
- Leitura facilitada da evolução do projeto e compatibilidade
- Permite fazer um rollback rapidamente

```
C:\WINDOWS\system32\cmd.exe
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ curso-deploy ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ curso-deploy ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ curso-deploy ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ curso-deploy ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ curso-deploy ---
[INFO]
[INFO] --- maven-war-plugin:2.2:war (default-war) @ curso-deploy ---
[INFO] Packaging webapp
[INFO] Assembling webapp [curso-deploy] in [E:\desenvolvimento\workspace\curso-deploy\git\app\target\curso-deploy]
[INFO] Processing war project
[INFO] Copying webapp resources [E:\desenvolvimento\workspace\curso-deploy\git\app\src\main\webapp]
[INFO] Webapp assembled in [329 msec]
[INFO] Building war: E:\desenvolvimento\workspace\curso-deploy\git\app\target\curso-deploy.war
[INFO] WEB-INF\web.xml already added, skipping
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 8.737 s
[INFO] Finished at: 2019-03-06T15:16:48-03:00
```

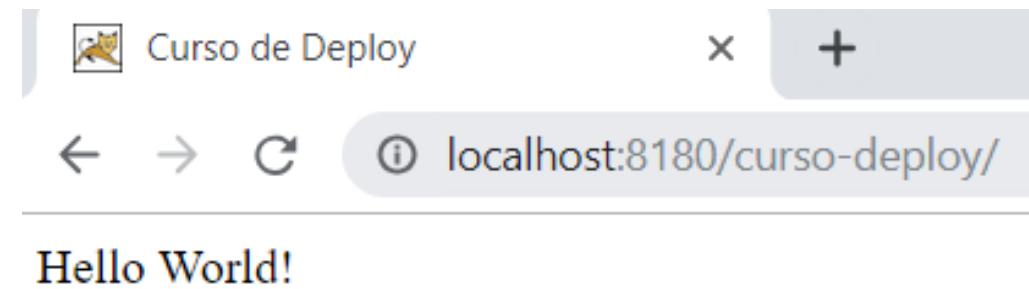
# Package

- No diretório raiz da aplicação (onde fica o pom.xml), rodar:
  - mvn clean package
- Na pasta target/ será gerado nosso pacote .war
  - curso-deploy.war

# Deploy manual

---

- Copiar o arquivo curso-deploy.war para a pasta “webapps” do servidor de produção.
- Se tudo deu certo, você terá feito o primeiro deploy manual da nossa aplicação no servidor de produção!



# Dia 2

---

APLICAÇÃO 2.0.0 | CONCEITOS MAVEN

# Aplicação-alvo

---

- NameInverter Tabajara! Aplicação que, ao receber um nome, apresenta ele ao contrário.
- Uma classe java e um arquivo de front-end.

The screenshot shows a web-based application with a light gray background and a white rounded rectangular form in the center. At the top of the form, the text "Bem-vindo ao curso de deploy automático e padrões de qualidade!" is displayed in bold black font. Below this, there is a label "Digite seu nome:" followed by a text input field containing the name "fulano". To the right of the input field are two buttons: a blue "Processar" button with a checkmark icon and a red "Limpar" button with a cross icon. At the bottom of the form, the text "Seu nome invertido é: onaluf" is displayed in red, indicating the result of the processed input.

**Bem-vindo ao curso de deploy automático e padrões de qualidade!**

Digite seu nome:

✓ Processar   ✗ Limpar

Seu nome invertido é: **onaluf**

```
package br.mp.esmpu.cursodeploy;

import javax.faces.bean.ManagedBean;
import javax.faces.bean.SessionScoped;

@ManagedBean(name = "helloWorld")
@SessionScoped
public class HelloWorld {

    private String name;
    private String nameReversed;

    public String greetings() {
        return "Bem-vindo ao curso de deploy automático e padrões de qualidade!";
    }

    public void clean() {
        this.name = this.nameReversed = "";
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
        this.nameReversed = /* POSSÍVEL CÓDIGO REUTILIZADO AQUI */;
    }

    public String getNameReversed() {
        return nameReversed;
    }
}
```

# Managed Bean

---

- Criar um bean gerenciado (JSF) chamado HelloWorld que recebe um nome e apresenta ele invertido
- Ele precisa inverter uma string... Eu acho que já deve ter uma lib que faz isso...

org.apache.commons.lang3

## Class StringUtils

java.lang.Object  
org.apache.commons.lang3.StringUtils

```
public class StringUtils  
extends Object
```

Operations on String that are null safe.

### reverse

```
public static String reverse(String str)
```

Reverses a String as per `StringBuilder.reverse()`.

A null String returns null.

```
StringUtils.reverse(null) = null  
StringUtils.reverse("") = ""  
StringUtils.reverse("bat") = "tab"
```

#### Parameters:

str - the String to reverse, may be null

#### Returns:

the reversed String, null if null string input

# Apache StringUtils

- <https://commons.apache.org/proper/commons-lang/apidocs/org/apache/commons/lang3/StringUtils.html>

# Mvnrepository

---

- Buscador de artefatos maven
- Possui os principais repositórios maven públicos
- Mais de 13 milhões de artefatos indexados
  - <https://mvnrepository.com/>

**Repository**

-  Central 3.9k
-  Sonatype 1.7k
-  Spring Plugins 1.0k
-  Spring Lib M 1.0k
-  IBiblio 220
-  JCenter 209
-  Spring Lib Release 171
-  WSO2 Releases 146

**Group**

-  org.apache 474
-  org.wso2 367
-  com.github 257
-  com.cedarssoft 147
-  org.xwiki 116
-  org.eclipse 95
-  org.jvnet 89
-  org.jresearch 64

**Category**

- Android Package 125
- Maven Plugins 33
- Eclipse Plugin 32
- Web App 25
- NetBeans Module 7

**Found 5502 results**Sort: [relevance](#) | [popular](#) | [newest](#)**1. Apache Commons Lang**[org.apache.commons](#) » commons-lang3

12,567 usages

Apache

Apache Commons Lang, a package of Java utility classes for the classes that are in java.lang's hierarchy, or are considered to be so standard as to justify existence in java.lang.

Last Release on Sep 19, 2018

**2. Apache Commons IO**[commons-io](#) » commons-io

15,940 usages

Apache

The Apache Commons IO library contains utility classes, stream implementations, file filters, file comparators, endian transformation classes, and much more.

Last Release on Oct 15, 2017

**3. Apache Commons Logging**[commons-logging](#) » commons-logging

8,515 usages

Apache

Apache Commons Logging is a thin adapter allowing configurable bridging to other, well known logging systems.

Last Release on Jul 5, 2014

**4. Commons Lang**[commons-lang](#) » commons-lang

9,949 usages

Apache

Commons Lang, a package of Java utility classes for the classes that are in java.lang's hierarchy, or are considered to be so standard as to justify existence in java.lang.

Last Release on Jan 16, 2011

# Apache Commons Lang 3.8.1

Home » org.apache.commons » commons-lang3 » 3.8.1



## Apache Commons Lang » 3.8.1

Apache Commons Lang, a package of Java utility classes for the classes that are in `java.lang`'s hierarchy, or are considered to be so standard as to justify existence in `java.lang`.

License	Apache 2.0
Categories	Core Utilities
HomePage	<a href="http://commons.apache.org/proper/commons-lang/">http://commons.apache.org/proper/commons-lang/</a>
Date	(Sep 19, 2018)
Files	<a href="#">jar (490 KB)</a> <a href="#">View All</a>
Repositories	<a href="#">Central</a> <a href="#">JCenter</a>
Used By	<a href="#">12,567 artifacts</a>

[Maven](#) [Gradle](#) [SBT](#) [Ivy](#) [Graal](#) [Leiningen](#) [Buildr](#)

```
<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-lang3 -->
<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-lang3</artifactId>
    <version>3.8.1</version>
</dependency>
```

Include comment with link to declaration

# Gerenciamento de dependências

---

- Adicionar no pom.xml, dentro da tag raiz project:

```
<dependencies>
    <!-- https://mvnrepository.com/artifact/org.apache.commons/commons-lang3 -->
    <dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-lang3</artifactId>
        <version>3.8.1</version>
    </dependency>
</dependencies>
```

- O que mudou nas libraries do projeto?

# Gerenciamento de dependências

---

- Maven dependencies
  - commons-lang3-3.8.1.jar
- Completando o HelloWorld.java...

```
import org.apache.commons.lang3.StringUtils;  
(...)  
this.nameReversed = StringUtils.reverse(name);
```

# Gerenciamento de dependências

---

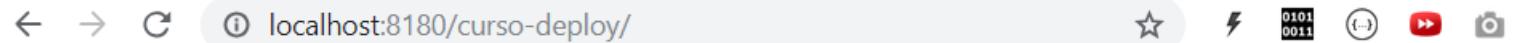
- Agora que já sabemos como procurar uma dependência (artefato) maven, vamos usar o maven para gerenciar as bibliotecas do JSF.
- TAREFA: usar o pom.xml para gerenciar as bibliotecas:
  - jsf-api
  - jsf-impl
- Após isso, excluir a pasta lib -> ela não é mais necessária!

# Index.xhtml

---

- Já podemos usar o managedBean na index.xhtml

```
<body>
    <h2 style="text-align: center;">#{helloWorld.greetings()}</h2>
</body>
```



**Bem-vindo ao curso de deploy automático e padrões de qualidade!**

```

<html xmlns="http://www.w3c.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html">

<head>
    <title>Curso de Deploy</title>

    <style type="text/css">
        .bold { font-weight: bold; }
        #titulo { text-align: center; margin-bottom: 40px; }
        #formulario { width: 500px; margin: auto; }
        #resultado { font-weight: bold; margin-top: 20px; }
        #resultado span { color: red; margin-left: 10px; }
    </style>
</head>

<body>
    <h2 id="titulo">#{helloWorld.greetings()}</h2>

    <h:form id="formulario">
        <h:panelGrid columns="4" cellpadding="5">
            <h:outputLabel styleClass="bold" for="nome" value="Digite seu nome:" />
            <h:inputText id="nome" value="#{helloWorld.name}" />
            <h:commandButton value="Processar" update="nome-invertido" />
            <h:commandButton value="Limpar" action="#{helloWorld.clean()}"
                             update="nome nome-invertido" />
        </h:panelGrid>

        <div id="resultado">
            <h:outputLabel for="nome-invertido" value="Seu nome invertido é:" />
            <h:outputText id="nome-invertido" value="#{helloWorld.nameReversed}" />
        </div>
    </h:form>
</body>
</html>

```

# Incrementando nossa aplicação

---

- Namespace JSF
- CommandButton que atualiza o managedBean e as seções da página via AJAX

# Quase lá... Versão 1.1.0

---

A screenshot of a web browser window. The address bar shows the URL `localhost:8080/curso-deploy/index.xhtml`. The page content is as follows:

**Bem-vindo ao curso de deploy automático e padrões de qualidade!**

Digite seu nome:

Seu nome invertido é: **onaluf**

- @TODO:
  - Testar o pacote localmente
  - Se validado, commitar e gerar a nova versão 1.1.0 (commit no develop, pull request, tag na branch master)
  - Publicar em produção (gerar pacote .war e fazer deploy manual)

# Melhorando a interface

---

- A equipe do projeto decidiu utilizar a biblioteca primefaces, popular no mundo JSF, para melhorar a interface do sistema.
- Roadmap:
  - Encontrar a dependência do primefaces no mvnrepository
  - Adicionar a dependência
  - Editar o web.xml para informar o theme do primefaces
  - Adaptar o código front-end para utilizar os componentes primefaces
  - Colocar um padrão de imagem no background

# Dependência primefaces

---

- No repositório central só existe até a versão 6.2 da biblioteca, mas eu gostaria de usar a mais atual (7.0.RC3).
- Solução: adicionar o repositório maven do projeto primefaces.

```
<repositories>
  <repository>
    <id>prime-repo</id>
    <name>PrimeFaces Maven Repository</name>
    <url>http://repository.primefaces.org</url>
  </repository>
</repositories>
```

```
<dependency>
  <groupId>org.primefaces</groupId>
  <artifactId>primefaces</artifactId>
  <version>7.0.RC3</version>
</dependency>
```

# Tema primefaces: nova-light

---

- Adicionar uma entrada no arquivo web.xml para utilizar o tema nova-light. É um tema famoso e nativo do universo primefaces que exibe uma interface clean.

```
<context-param>
    <param-name>primefaces.THEME</param-name>
    <param-value>nova-light</param-value>
</context-param>
```

```

<html xmlns="http://www.w3c.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:p="http://primefaces.org/ui">

<h:head>
    <title>Curso de Deploy</title>
    <style type="text/css">
        body { background-image: url("img/cream-pixels.png"); }
        .bold { font-weight: bold; }
        #content { margin: 0 auto; width: 960px; border: 1px solid black; padding: 40px;
                   border-radius: 20px; margin-top: 100px; }
        #titulo { text-align: center; margin-bottom: 40px; }
        #formulario { margin: 0 auto; width: 550px; }
        #resultado { font-weight: bold; margin-top: 20px; }
        #resultado span { color: red; margin-left: 10px; }
    </style>
</h:head>

<h:body>
    <div id="content">
        <h2 id="titulo">#{helloWorld.greetings()}</h2>
        <h:form id="formulario">
            <h:panelGrid columns="4" cellpadding="5">
                <h:outputLabel styleClass="bold" for="nome" value="Digite seu nome:" />
                <p:inputText id="nome" value="#{helloWorld.name}" />
                <p:commandButton value="Processar" update="nome-invertido" icon="pi pi-check" />
                <p:commandButton value="Limpar" action="#{helloWorld.clean()}" update="nome nome-invertido"
                                  icon="pi pi-times" style="background: red; border: 1px solid red" />
            </h:panelGrid>
            <div id="resultado">
                <h:outputLabel for="nome-invertido" value="Seu nome invertido é:" />
                <h:outputText id="nome-invertido" value="#{helloWorld.nameReversed}" />
            </div>
        </h:form>
    </div>
</h:body>
</html>

```

# Front-end

---

- Namespace JSF
- Tags InputText e CommandButton do primefaces
- Pasta img criada dentro de webapp
- Adequações ao CSS

# Resultado final

---

- Aplicação 2.0.0 com nova interface visual utilizando a biblioteca mais nova do primefaces.
- @TODO:
  - Gerar a versão 2.0.0 (commit, pull request, tag) e publicar em produção.

The screenshot shows a web application interface with a light gray background and a white rounded rectangular content area. Inside this area, the following text is displayed:

**Bem-vindo ao curso de deploy automático e padrões de qualidade!**

Digite seu nome:  ✓ Processar ✗ Limpar

Seu nome invertido é: **onaluf**

# Gerenciamento de dependências

---

# Gerenciamento de dependências

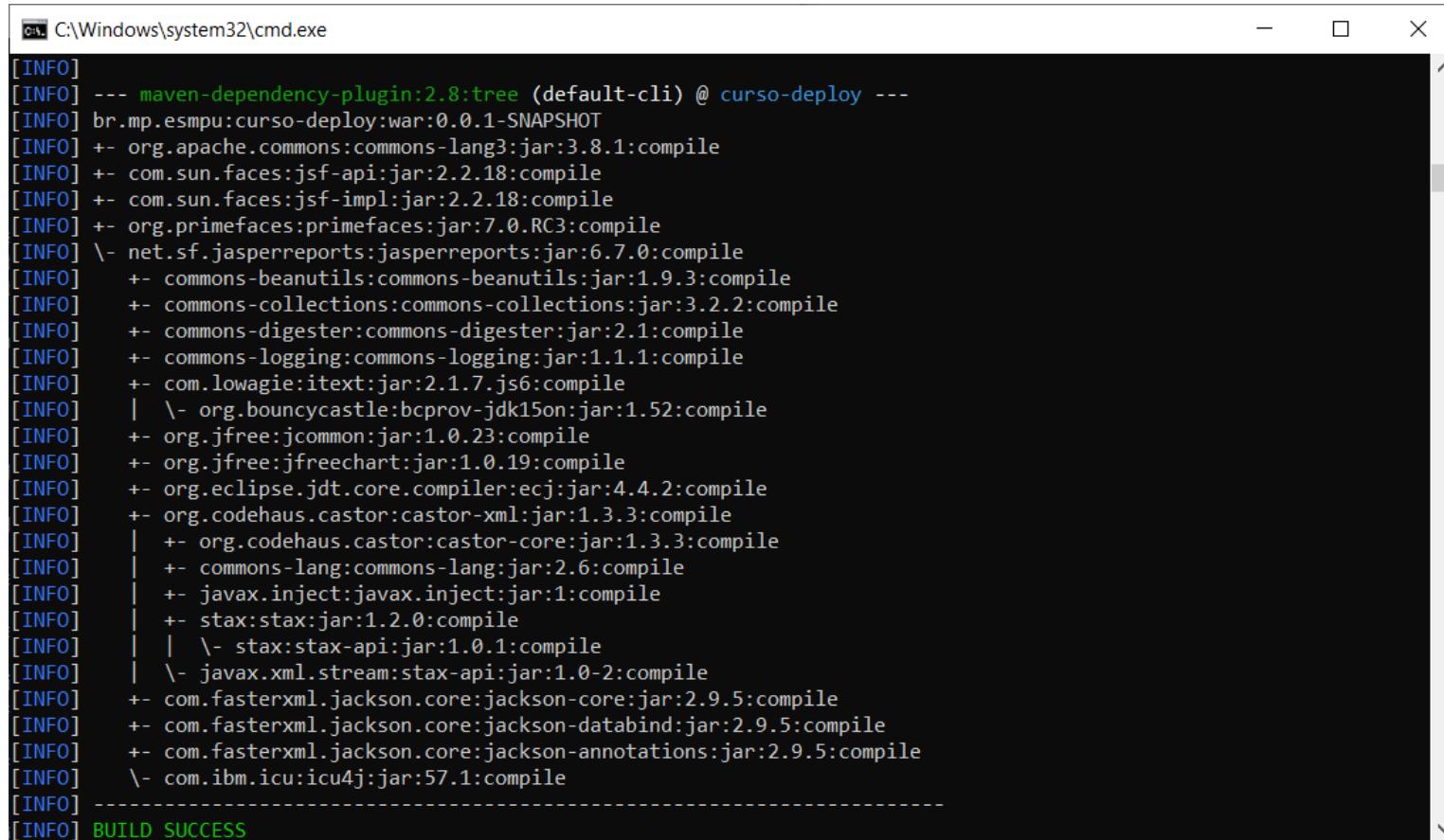
---

- A equipe do projeto escolheu utilizar o Jasper Reports, biblioteca famosa para a geração de relatórios PDF.
- Adicione a dependência no projeto:

```
<!-- https://mvnrepository.com/artifact/net.sf.jasperreports/jasperreports -->
<dependency>
    <groupId>net.sf.jasperreports</groupId>
    <artifactId>jasperreports</artifactId>
    <version>6.7.0</version>
</dependency>
```

# Gerenciamento de dependências

- Agora, vamos abrir um prompt de comando (cmd) e ir até a raiz do projeto, que é a pasta onde está o arquivo pom.xml
- Em seguida, execute o comando
  - mvn dependency:tree
- Qual o significado disso?



A screenshot of a Windows Command Prompt window titled 'C:\Windows\system32\cmd.exe'. The window displays the output of the 'mvn dependency:tree' command. The output shows a hierarchical tree of dependencies for the project 'br.mpeesmpu.curso-deploy'. The tree starts with the main project and branches down through various Java libraries like commons-lang, jsf-api, primefaces, jasperreports, bouncycastle, jfreechart, eclipse.jdt.core.compiler, codehaus.castor, stax, javax.inject, fasterxml.jackson.core, fasterxml.jackson.databind, fasterxml.jackson.annotations, and ibm.icu.icu4j. The output is color-coded with green for Maven commands and blue for project names. The message '[INFO] BUILD SUCCESS' is visible at the bottom.

```
[INFO]
[INFO] --- maven-dependency-plugin:2.8:tree (default-cli) @ curso-deploy ---
[INFO] br.mpeesmpu.curso-deploy:war:0.0.1-SNAPSHOT
[INFO] +- org.apache.commons:commons-lang3:jar:3.8.1:compile
[INFO] +- com.sun.faces:jsf-api:jar:2.2.18:compile
[INFO] +- com.sun.faces:jsf-impl:jar:2.2.18:compile
[INFO] +- org.primefaces:primefaces:jar:7.0.RC3:compile
[INFO] \- net.sf.jasperreports:jasperreports:jar:6.7.0:compile
    +- commons-beanutils:commons-beanutils:jar:1.9.3:compile
    +- commons-collections:commons-collections:jar:3.2.2:compile
    +- commons-digester:commons-digester:jar:2.1:compile
    +- commons-logging:commons-logging:jar:1.1.1:compile
    +- com.lowagie:itext:jar:2.1.7.js6:compile
        \- org.bouncycastle:bcprov-jdk15on:jar:1.52:compile
    +- org.jfree:jcommon:jar:1.0.23:compile
    +- org.jfree:jfreechart:jar:1.0.19:compile
    +- org.eclipse.jdt.core.compiler:ecj:jar:4.4.2:compile
    +- org.codehaus.castor:castor-xml:jar:1.3.3:compile
        +- org.codehaus.castor:castor-core:jar:1.3.3:compile
        +- commons-lang:commons-lang:jar:2.6:compile
        +- javax.inject:javax.inject:jar:1:compile
        +- stax:stax:jar:1.2.0:compile
            \- stax:stax-api:jar:1.0.1:compile
            \- javax.xml.stream:stax-api:jar:1.0-2:compile
    +- com.fasterxml.jackson.core:jackson-core:jar:2.9.5:compile
    +- com.fasterxml.jackson.core:jackson-databind:jar:2.9.5:compile
    +- com.fasterxml.jackson.core:jackson-annotations:jar:2.9.5:compile
    \- com.ibm.icu:icu4j:jar:57.1:compile
[INFO]
[INFO] BUILD SUCCESS
```

# Dependências transitivas

---

- Uma das funcionalidades mais poderosas do maven é a resolução de dependências.
  - <https://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html>
- Toda dependência no maven é transitiva: se alguma dependência precisa de uma outra não especificada no seu projeto, o seu projeto passará a depender dela também, e será incluída no artefato gerado.
- A transitividade de uma dependência pode ser limitada pelo escopo. Existem 6 escopos:
  - compile (padrão)
  - provided
  - runtime
  - test
  - system
  - import

# Maven lifecycle

---

# Maven lifecycle

---

- O processo de build e distribuição de um artefato/projeto é bem definido pelo ciclo de vida do maven.
- Um desenvolvedor, com apenas um comando, se beneficia da estrutura pré-definida do ciclo de vida e consegue construir qualquer projeto maven, baseado nas configurações do POM.
- Existem três ciclos de vida nativos, cada um constituído de várias fases:
  - default: gerencia o deploy do projeto
  - clean: gerencia a limpeza do projeto
  - site: gerencia a criação de um site de documentação do projeto

# Default lifecycle

---

- Por exemplo, o ciclo de vida default inclui as seguintes fases (ver [link](#)):
  - **validate**: valida se o projeto está correto e todas as informações necessárias estão disponíveis
  - **compile**: compila o código-fonte do projeto
  - **test**: teste o código-fonte compilado usando uma estrutura de teste unitário adequada.
  - **package**: pega o código compilado e o empacota em seu formato distribuível, por exemplo JAR ou WAR.
  - **verify**: verifica os resultados dos testes de integração para garantir que os critérios de qualidade foram atendidos
  - **install**: instala o pacote no repositório local, para uso como dependência em outros projetos localmente
  - **deploy**: feito no ambiente de build, copia o pacote final para o repositório remoto para compartilhar com outros desenvolvedores e projetos.
- Essas fases do ciclo de vida são executadas sequencialmente. Ou seja, caso o default lifecycle seja usado, o Maven primeiro validará o projeto, então tentará compilar os fontes, executar os testes, empacotar os binários (por exemplo jar), executar testes de integração e verificar os resultados, instalar o pacote verificado no repositório local e, em seguida, publicar o pacote instalado em um repositório remoto.

# Default lifecycle

---

- Ao rodar o maven, precisamos apenas indicar a fase final que gostaríamos de executar do ciclo de vida default.
- Por exemplo, o comando abaixo executará a fase package do ciclo de vida default e todas as fases anteriores, em ordem:
  - mvn package
- Podemos, inclusive, chamar outro lifecycle junto. É muito comum vermos a seguinte execução:
  - mvn clean package
- Mas, ao chamar um lifecycle, como eu sei o que será executado de fato?

# Plugin goals

---

- O Maven é, no fundo, um framework de execução de plugins. Quem, de fato, executa ou processa algo na arquitetura maven são os plugins.
- Os plugins são artefatos que disponibilizam goals que executam uma lógica associada.
- Vários plugins são próprios do maven:
  - <https://repo.maven.apache.org/maven2/org/apache/maven/plugins/>
- Para chamar um goal de um plugin, basta chamar:
  - mvn plugin:goal

# Plugin goals

---

- Por exemplo, o maven-compiler-plugin (compiler) possui apenas 3 goals:
  - compile: compila o código-fonte principal da aplicação
  - testCompile: compila o código-fonte de testes de aplicação
  - help: apresenta ajuda de uso do plugin
- Tente executar o seguinte comando:
  - mvn compiler:compile

# Packaging binding

## Default Lifecycle Bindings - Packaging `ejb` / `ejb3` / `jar` / `par` / `rar` / `war`

Phase	plugin:goal
<code>process-resources</code>	<code>resources:resources</code>
<code>compile</code>	<code>compiler:compile</code>
<code>process-test-resources</code>	<code>resources:testResources</code>
<code>test-compile</code>	<code>compiler:testCompile</code>
<code>test</code>	<code>surefire:test</code>
<code>package</code>	<code>ejb:ejb</code> or <code>ejb3:ejb3</code> or <code>jar:jar</code> or <code>par:par</code> or <code>rar:rar</code> or <code>war:war</code>
<code>install</code>	<code>install:install</code>
<code>deploy</code>	<code>deploy:deploy</code>

- O maven traz um [binding](#) padrão para cada packaging, o que facilita e muito nossa vida.
- O binding vincula a execução de um plugin:goal a uma fase do lifecycle default.
- Execute novamente a fase package e analise os goals que foram executados.

# Plugin binding

---

- Outra forma é declarar o uso de um plugin e vincular a uma fase do ciclo de vida.
- Por exemplo, caso exista um plugin display com um goal time, e você queira vincular sua execução à fase process-test-resources do ciclo de vida, precisamos configurar isso no POM:

```
...
<plugin>
    <groupId>com.mycompany.example</groupId>
    <artifactId>display-maven-plugin</artifactId>
    <version>1.0</version>
    <executions>
        <execution>
            <phase>process-test-resources</phase>
            <goals>
                <goal>time</goal>
            </goals>
        </execution>
    </executions>
</plugin>
...
```

# Plugins disponíveis

---

- <https://maven.apache.org/plugins/index.html>
- Vários deles são mantidos pelo “The Maven Project”, como por exemplo:
  - clean
  - compiler
  - war
  - ant
  - ...

# Maven POM

---

# POM

---

- O POM (Project Object Model) é a unidade fundamental de trabalho no Maven.
- É um arquivo XML que contém informações sobre o projeto e detalhes de configuração usados pelo Maven para construir o projeto. Ele contém valores padrão para a maioria dos projetos.
- Por exemplo, temos:
  - diretório de build: target/
  - diretório do código-fonte: src/main/java
  - diretório do código-fonte de testes: src/test/java
  - ...
- Ao executar uma tarefa ou goal, o Maven procura o POM no diretório atual. Ele lê o POM, obtém as informações de configuração necessárias e executa o goal.

# POM

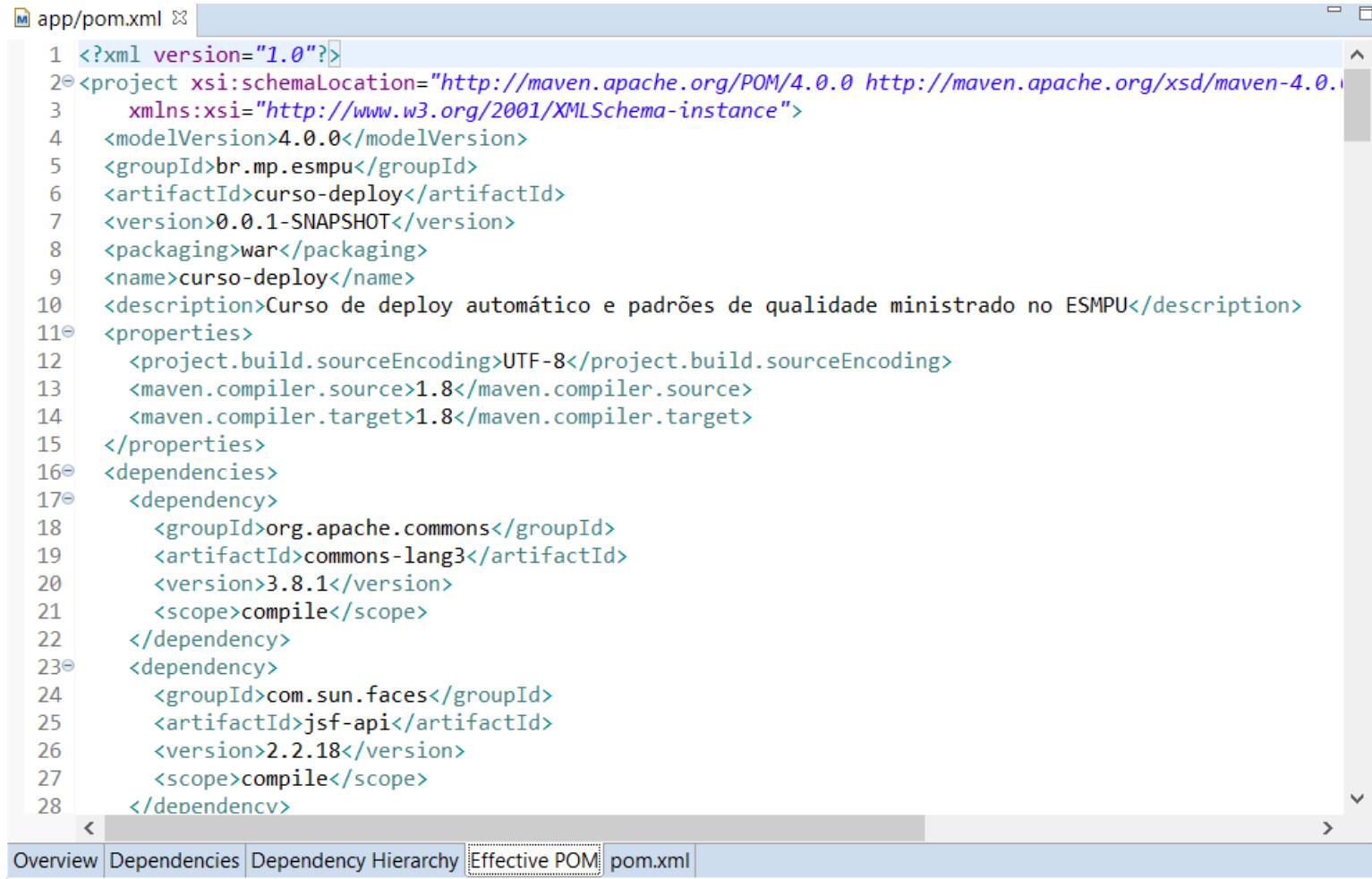
---

- Ao executar uma tarefa ou goal, o Maven procura o POM no diretório atual. Ele lê o POM, obtém as informações de configuração necessárias e executa o goal.
- Algumas das configurações que podem ser especificadas no POM são:
  - dependências do projeto
  - plugins ou metas (goals) que podem ser executados
  - perfis de construção (profiles)
  - outras informações, como a versão do projeto, descrição, desenvolvedores e listas de discussão.

# Super POM

---

- O Super POM é o POM padrão do Maven. Todos os POMs, de maneira geral, estendem o Super POM, o que significa que a configuração especificada no Super POM é herdada pelos POMs que você criou para seus projetos.
- Maven 3.6.0:
  - <https://maven.apache.org/ref/3.6.0/maven-model-builder/super-pom.html>



```
1 <?xml version="1.0"?>
2 <project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"
3   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
4   <modelVersion>4.0.0</modelVersion>
5   <groupId>br.mp.esmpu</groupId>
6   <artifactId>curso-deploy</artifactId>
7   <version>0.0.1-SNAPSHOT</version>
8   <packaging>war</packaging>
9   <name>curso-deploy</name>
10  <description>Curso de deploy automático e padrões de qualidade ministrado no ESMPU</description>
11  <properties>
12    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
13    <maven.compiler.source>1.8</maven.compiler.source>
14    <maven.compiler.target>1.8</maven.compiler.target>
15  </properties>
16  <dependencies>
17    <dependency>
18      <groupId>org.apache.commons</groupId>
19      <artifactId>commons-lang3</artifactId>
20      <version>3.8.1</version>
21      <scope>compile</scope>
22    </dependency>
23    <dependency>
24      <groupId>com.sun.faces</groupId>
25      <artifactId>jsf-api</artifactId>
26      <version>2.2.18</version>
27      <scope>compile</scope>
28    </dependency>

```

# Effective POM

- Merge entre o Super POM e o POM do projeto.
- É o POM que é utilizado, de fato, para o processamento maven do projeto.

# Standard Layout Directory

---

<code>src/main/java</code>	Application/Library sources
<code>src/main/resources</code>	Application/Library resources
<code>src/main/filters</code>	Resource filter files
<code>src/main/webapp</code>	Web application sources
<code>src/test/java</code>	Test sources
<code>src/test/resources</code>	Test resources
<code>src/test/filters</code>	Test resource filter files
<code>src/it</code>	Integration Tests (primarily for plugins)
<code>src/assembly</code>	Assembly descriptors
<code>src/site</code>	Site
<code>LICENSE.txt</code>	Project's license
<code>NOTICE.txt</code>	Notices and attributions required by libraries that the project depends on
<code>README.txt</code>	Project's readme

# Standard Layout Directory

---

- Estrutura de diretórios padrão esperada pelo Maven
- Pode ser configurada (sobrescrita)

# Dia 3

---

MAVEN PROFILES | DEPLOY JENKINS

# Jenkins

---

# Jenkins

---

- <https://jenkins.io/>

The Jenkins mascot is a cartoon character with a large, bulbous nose, wearing a blue suit jacket, a white shirt, and a red bow tie. He has a simple, friendly face with dark hair and is standing against a red circular background.

**Jenkins**

**Build great things at any scale**

The leading open source automation server, Jenkins provides hundreds of plugins to support building, deploying and automating any project.

[Documentation](#) [Download](#)

# Instalação jenkins

---

# Instalação jenkins

---

- Copiar o arquivo jenkins.war para o servidor de produção:
  - <TOMCAT\_HOME>\webapps
- Iniciar o servidor

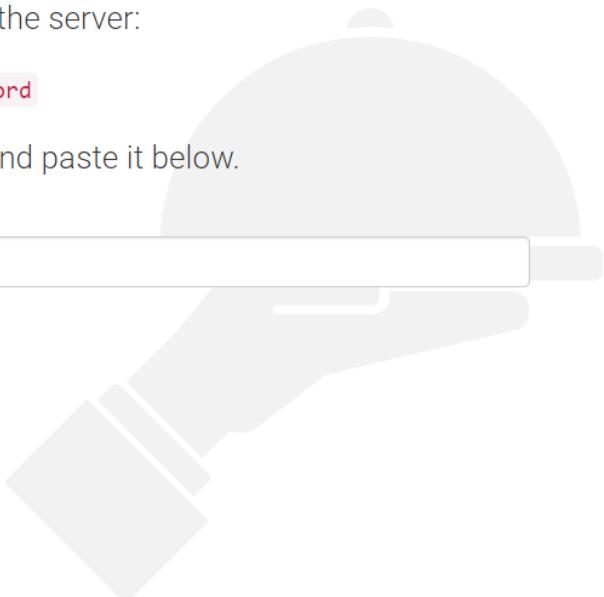
## Unlock Jenkins

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:

`C:\Users\rajiv\.jenkins\secrets\initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password



Continue

# Instalação jenkins

- Colocar o password de administrador

**Getting Started**

All | None | Suggested Selected (22/58)

- CVS** Integrates Jenkins with CVS version control system using a modified version of the Netbeans cvsclient.
- Git** This plugin integrates Git with Jenkins.
- Git Parameter** Adds ability to choose branches, tags or revisions from git repositories configured in project.
- GitHub** This plugin integrates GitHub to Jenkins.
- GitLab** This plugin allows GitLab to trigger Jenkins builds and display their results in the GitLab UI.
- P4** Perforce Client plugin for the Jenkins SCM provider. The plugin includes extension points for: Perforce Password and Ticket Credentials storeWorkspace management for static, manual, template and streamAction point for Review Builds
- REPO** This plugin allows use of `repo` as an SCM tool. A `repo` binary is required.
- Subversion**
- Team Concert**

Jenkins 2.166

Back Install

A screenshot of the Jenkins 'Getting Started' page. On the left is a sidebar with categories like Organization and Administration, Build Features, etc. The main area shows a list of available Jenkins plugins. Plugins listed include CVS, Git, Git Parameter, GitHub, GitLab, P4, REPO, Subversion, and Team Concert. The 'Git' plugin is selected, indicated by a checked checkbox. At the bottom right are 'Back' and 'Install' buttons.

# Plugins

- Adicionar os plugins:
  - Git Parameter
  - Locale
  - Build Timestamp

The screenshot shows the Jenkins dashboard. At the top, there's a navigation bar with the Jenkins logo, a search bar labeled "pesquisar", and user information for "admin" with a "sair" link. Below the bar, a green header bar has the text "HABILITAR ATUALIZAÇÃO AUTOMÁTICA". On the left, a sidebar lists various Jenkins management options: "Novo job", "Usuários", "Histórico de compilações", "Gerenciar Jenkins", "Minhas views", "Lockable Resources", "Credentials", and "New View". A collapsed section titled "Fila de builds" shows "Nenhum build na fila.". Another collapsed section titled "Estado do executor de builds" shows "1 Parado" and "2 Parado". The main content area features a large "Bem-vindo ao Jenkins!" heading and a message: "Por favor clique em: [Novo job](#) para iniciar." There's also a "Adicionar descrição" button. At the bottom, a footer bar displays the page generation time "Página gerada: 03/03/2019 17h39min58s BRT", links to "REST API" and "Jenkins ver. 2.166", and a small orange footer bar.

# Voilà!

- Jenkins v2.168 no ar!

# Locale en\_US

---

- Jenkins > Gerenciar Jenkins > Configurar o sistema
- Locale: en\_US
- Plugin para definir a língua do Jenkins. Por padrão, ele exibe de acordo com a definida para o sistema operacional.



## JDK installations

Add JDK

JDK	
Name	jdk1.8.0_201
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_201
<input type="checkbox"/> Install automatically	

Add JDK

List of JDK installations on this system

## Git

## Git installations

Git

Git	
Name	Default
Path to Git executable	C:\Program Files\Git\bin\git.exe
<input type="checkbox"/> Install automatically	

 Install automatically

## Maven

## Maven instalações

Adicionar Maven

Maven	
Nome	maven-3.6.0
MAVEN_HOME	E:\desenvolvimento\workspace\curso-deploy\tool\apache-maven-3.6.0
<input type="checkbox"/> Instalar automaticamente	

 Instalar automaticamente

Adicionar Maven

Lista de Maven instalações nesse sistema

# Configuração das ferramentas

- Jenkins > Manage Jenkins > Global Tool Configuration
- Ferramentas:
  - JDK
  - Git
  - Maven

# Configuração das ferramentas

---

- Jenkins > Manage Jenkins > Global Tool Configuration
- Ferramentas:
  - Build Timestamp Plugin

**Build Timestamp**

---

Enable BUILD\_TIMESTAMP

Timezone

Pattern

Using timezone: America/Sao\_Paulo; Sample timestamp: 18/03/2019 18:44:55 BRT

Export more variables

## Enter an item name

curso-deploy

» 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 build.



### Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.



### 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 multiple things of the same name 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.

# New job

- Criar um novo job jenkins como “Freestyle project”

## Source Code Management

- None
- Git

### Repositories

Repository URL

Credentials  Add

Advanced...

Add Repository

### Branches to build

Branch Specifier (blank for 'any')

x ?

Add Branch

### Repository browser

(Auto)

### Additional Behaviours

Add ▾

# Source code management

- Indicar a URL do repositório git
- Criar uma nova credencial (clicar em “Add”)
- Em “Branches to build”, deixar a configuração padrão, que busca a branch master

# Build environment

---

**Build Environment**

Delete workspace before build starts

[Advanced...](#)

Use secret text(s) or file(s)

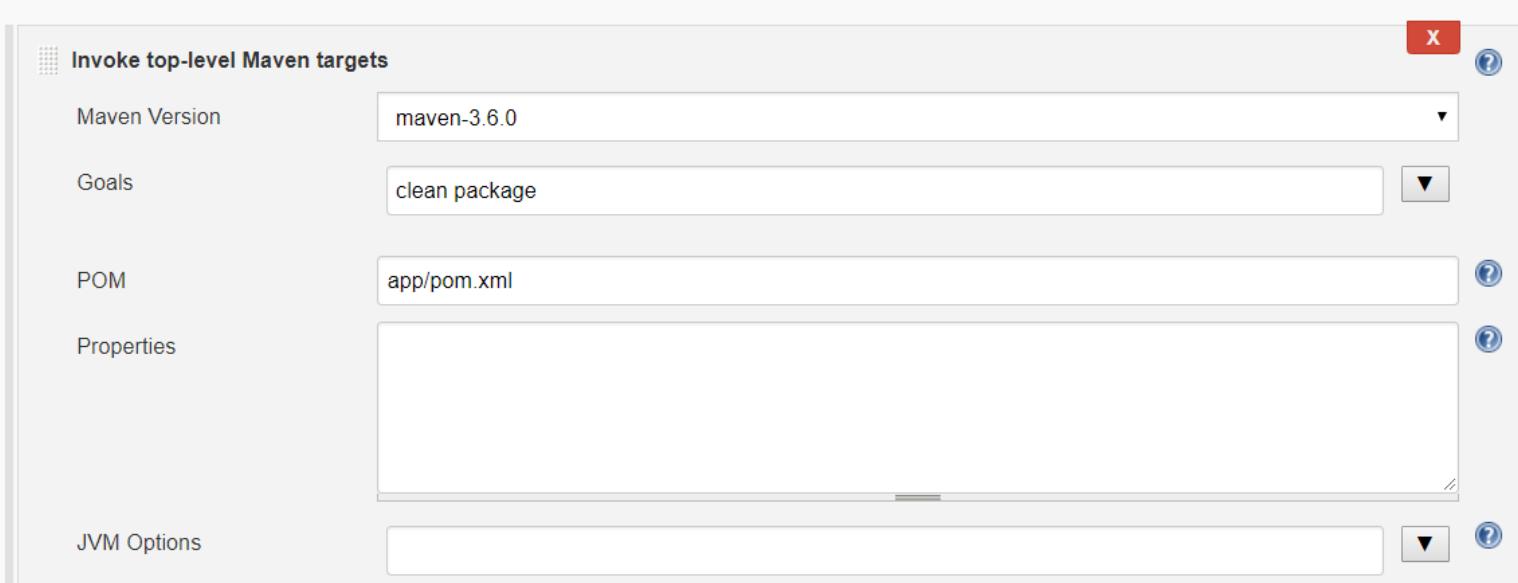
Abort the build if it's stuck

Add timestamps to the Console Output

With Ant

- Por precaução, marcamos sempre de deletar o workspace do projeto antes de iniciar uma build
- Para facilitar a visualização do log, é bom imprimir o timestamp na console

## Build



# Build

- Chamar o goal:
  - clean package
- Atenção! Indicar a localização do arquivo POM. No nosso caso, como colocamos a aplicação na pasta app/, temos que indicar o path correto.

# Post-build actions

---

**Post-build Actions**

Archive the artifacts

Files to archive    `**/*.war`

Add post-build action ▾



- Arquivar todos os pacotes .war

```
21:14:15 [INFO]
21:14:15 [INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ curso-deploy ---
21:14:16 [INFO] Using 'UTF-8' encoding to copy filtered resources.
21:14:16 [INFO] skip non existing resourceDirectory C:\Users\rajiv\.jenkins\workspace\curso-deploy\app\src\main\resources
21:14:16 [INFO]
21:14:16 [INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ curso-deploy ---
21:14:16 [INFO] No sources to compile
21:14:16 [INFO]
21:14:16 [INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ curso-deploy ---
21:14:16 [INFO] Using 'UTF-8' encoding to copy filtered resources.
21:14:16 [INFO] skip non existing resourceDirectory C:\Users\rajiv\.jenkins\workspace\curso-deploy\app\src\test\resources
21:14:16 [INFO]
21:14:16 [INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ curso-deploy ---
21:14:16 [INFO] No sources to compile
21:14:16 [INFO]
21:14:16 [INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ curso-deploy ---
21:14:16 [INFO] No tests to run.
21:14:16 [INFO]
21:14:16 [INFO] --- maven-war-plugin:2.2:war (default-war) @ curso-deploy ---
21:14:17 [INFO] Packaging webapp
21:14:17 [INFO] Assembling webapp [curso-deploy] in [C:\Users\rajiv\.jenkins\workspace\curso-deploy\app\target\curso-deploy]
21:14:17 [INFO] Processing war project
21:14:17 [INFO] Copying webapp resources [C:\Users\rajiv\.jenkins\workspace\curso-deploy\app\src\main\webapp]
21:14:17 [INFO] Webapp assembled in [82 mssecs]
21:14:17 [INFO] Building war: C:\Users\rajiv\.jenkins\workspace\curso-deploy\app\target\curso-deploy.war
21:14:17 [INFO] WEB-INF\web.xml already added, skipping
21:14:17 [INFO] -----
21:14:17 [INFO] BUILD SUCCESS
21:14:17 [INFO] -----
21:14:17 [INFO] Total time: 2.449 s
21:14:17 [INFO] Finished at: 2019-03-03T21:14:17-03:00
21:14:17 [INFO] -----
21:14:18 Archiving artifacts
21:14:18 Finished: SUCCESS
```

# Saída da console

---

- Finished: SUCCESS
- Estamos juntos até agora?

## Project curso-deploy



## Last successful artifacts

- Teste: pegar o arquivo .war gerado (curso-deploy.war) e jogar na pasta webapps do “servidor de produção”.
- Acessar a URL do nosso app em “produção”:
  - <http://localhost:8180/curso-deploy/>
- Funcionou? Parabéns pelo seu primeiro “deploy” Jenkins em produção!

# Deploy automatizado

---

- Ok, o nosso “deploy” poderia ser melhor. Precisamos automatizar o deploy, de maneira que ele publique automaticamente o .war gerado em produção.
- Como podemos fazer isso? Usando um plugin, é claro!
  - <http://tomcat.apache.org/maven-plugin-trunk/tomcat7-maven-plugin/plugin-info.html>
  - <https://www.baeldung.com/tomcat-deploy-war>
- Primeiramente, vamos criar um usuário admin em produção com as roles manager-gui e manager-script no arquivo %TOMCAT\_HOME%/conf/tomcat-users.xml

```
<role rolename="manager-gui"/>
<role rolename="manager-script"/>
<user username="admin" password="admin" roles="manager-gui,manager-script"/>
```

- Teste o acesso na página de administração do tomcat:
  - <http://localhost:8180/manager/html>

# Deploy automatizado

---

- Em seguida, vamos adicionar o plugin no pom.xml, dentro da tag <plugins>:

```
<build>
    <finalName>${project.artifactId}</finalName>
    <plugins>
        <plugin>
            <groupId>org.apache.tomcat.maven</groupId>
            <artifactId>tomcat7-maven-plugin</artifactId>
            <version>2.2</version>
            <configuration>
                <url>http://localhost:8180/manager/text</url>
                <path>/${project.build.finalName}</path>
                <username>admin</username>
                <password>admin</password>
                <update>true</update>
            </configuration>
        </plugin>
    </plugins>
</build>
```

# Tomcat maven plugin

---

- O plugin envia o pacote gerado pelo goal package para a URL de gerenciamento do Tomcat, passando o username e password informados:
  - <http://localhost:8180/manager/text>
- E publica o .war no contexto informado na tag <path>
- Não declaramos o caminho do pacote .war porque ele está já no caminho default esperado pela tag opcional <warFile>.
- @TODO:
  - Execute o comando “mvn tomcat7:redeploy” e veja o resultado
  - Pergunta: devemos vincular essa execução a alguma fase do default lifecycle?
  - Após as mudanças, fazer merge com a branch master e gerar nova tag

# Tomcat maven plugin

---

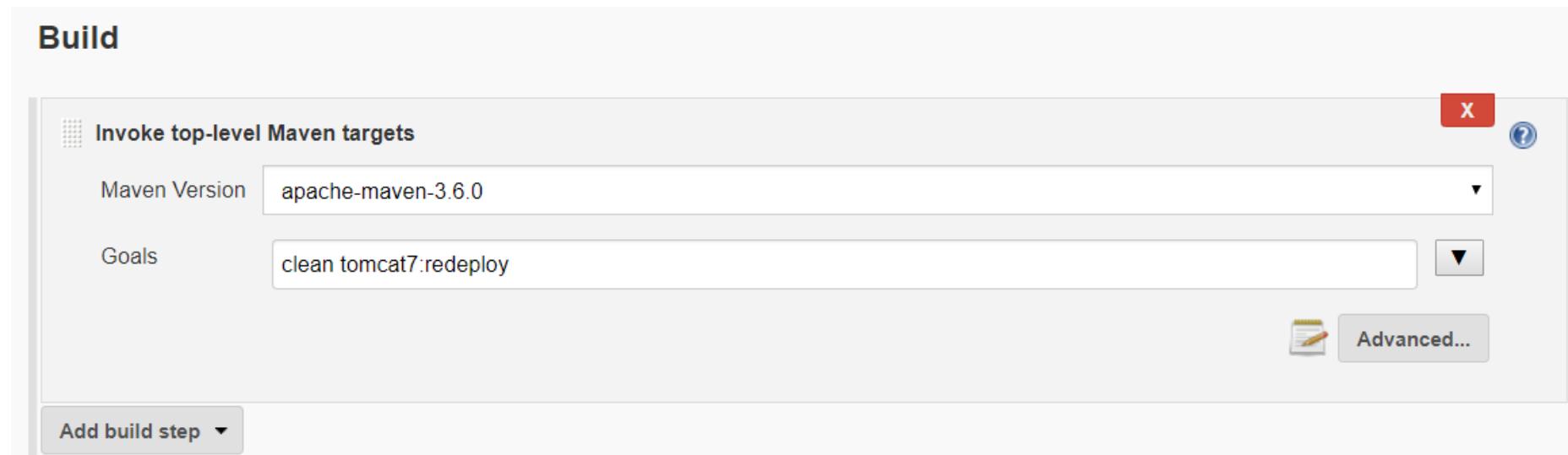
- TAREFA:
  - Execute o comando “mvn tomcat7:redeploy” e veja o resultado
  - Pergunta: devemos vincular essa execução a alguma fase do default lifecycle?
  - Após as mudanças, fazer merge com a branch master e gerar nova tag
- Observação:
  - Existem vários relatos que o tomcat 8.5, no windows, dá lock em arquivos de biblioteca e pode apresentar o erro:
    - [INFO] Undeploying application at http://localhost:8180/curso-deploy
    - [INFO] FAIL - Unable to delete [D:\desenvolvimento\workspace\curso-deploy\server\apache-tomcat-8.5.38-producao\webapps\curso-deploy]. The continued presence of this file may cause problems.
  - Caso aconteça isso, altere o arquivo %TOMCAT\_HOME%\conf\context.xml:

```
<Context antiResourceLocking="true">
```

# Deploy automatizado

---

- Agora, podemos editar o job Jenkins para realizar a etapa de deploy automático.
- Alterar a seção Build para chamar o plugin tomcat7:



# Deploy automatizado

---

- Finished: SUCCESS
- Agora sim, primeiro deploy automatizado realizado com sucesso!

```
34126/34136 KB
34128/34136 KB
34130/34136 KB
34132/34136 KB
34134/34136 KB
34136/34136 KB
Uploaded: http://localhost:8180/manager/text/deploy?path=%2Fcurso-deploy&update=true (34136 KB at 6399.7 KB/sec)
16:15:18
16:15:27 [INFO] tomcatManager status code:200, ReasonPhrase:
16:15:27 [INFO] OK - Deployed application at context path [/curso-deploy]
16:15:27 [INFO] -----
16:15:27 [INFO] BUILD SUCCESS
16:15:27 [INFO] -----
16:15:27 [INFO] Total time: 28.354 s
16:15:27 [INFO] Finished at: 2019-03-14T16:15:27-03:00
16:15:27 [INFO] -----
16:15:31 Archiving artifacts
16:15:32 Finished: SUCCESS
```

# Maven profile

---

# Maven profile

---

- No mundo real, as builds possuem peculiaridades dependendo do ambiente destino, como:
  - Conexão com o banco
  - Caminhos de pastas no filesystem
  - Arquivos de propriedades
  - Debugs ou níveis de log
  - Imagens diferenciadas
- O maven endereça isso com profiles, que modificam o POM em tempo de execução com os parâmetros e configurações informados.
- Por exemplo, eu poderia criar um profile prod (de produção) e chamar:
  - `mvn -P prod clean tomcat7:redeploy`

# Maven profile

---

- Objetivo: printar, na tela, o ambiente da aplicação, seja desenvolvimento, homologação ou produção.
  - O código-fonte é o mesmo e é baixado do GIT
  - Dependendo do profile escolhido, o maven saberá se deve imprimir na tela a string “Ambiente Local”, “Ambiente de Desenvolvimento”, “Ambiente de Homologação” ou não imprimir nada para o caso de estarmos em produção.

**Bem-vindo ao curso de deploy automático e padrões de qualidade!**  
**(Ambiente de Desenvolvimento)**

Digite seu nome:

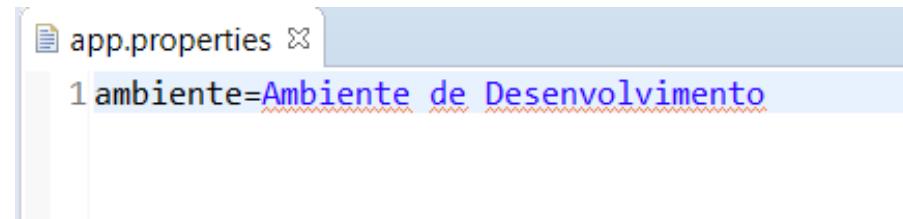
Processar     Limpar

Seu nome invertido é:

# Maven profile

---

- Primeiramente, vamos criar um arquivo de propriedades da nossa aplicação.
  - Crie o arquivo app.properties na pasta src/main/resources
  - Crie uma variável “ambiente” com valor inicial “Ambiente de Desenvolvimento”



# Maven profile

---

- Em seguida, edite o arquivo index.xhtml para carregar o arquivo app.properties na variável “propriedades” e exibir o valor propriedades.ambiente

```
<h:body>
    <f:loadBundle basename="app" var="properties" />

    <div id="content">
        <h2 id="titulo">
            #{helloWorld.greetings()}
            #{properties.ambiente}
        </h2>
        (...)
```

- Salve os arquivos e recarregue a nossa aplicação.

# Maven profile

---

- Agora, nossa aplicação consegue ler e imprimir uma propriedade do sistema. Porém, ela é diferente em cada ambiente.
  - **local**: ambiente=Ambiente Local
  - **des**: ambiente=Ambiente de Desenvolvimento
  - ~~**hom**: ambiente=Ambiente de Homologação (não usaremos)~~
  - **prod**: ambiente=
- Como fazer isso?

# Maven profile

---

- Usando um recurso nativo do Maven chamado filtering.
- Ele é previsto no Standard Directory Layout, sendo armazenado na pasta filters.

## Introduction to the Standard Directory Layout

Having a common directory layout would allow for users familiar with one Maven project to immediately feel at home in another Maven project. The advantages are analogous to adopting a site-wide look-and-feel.

The next section documents the directory layout expected by Maven and the directory layout created by Maven. Please try to conform to this structure as much as possible; however, if you can't these settings can be overridden via the project descriptor.

src/main/java	Application/Library sources
src/main/resources	Application/Library resources
src/main/filters	Resource filter files
src/main/webapp	Web application sources

# Maven profile

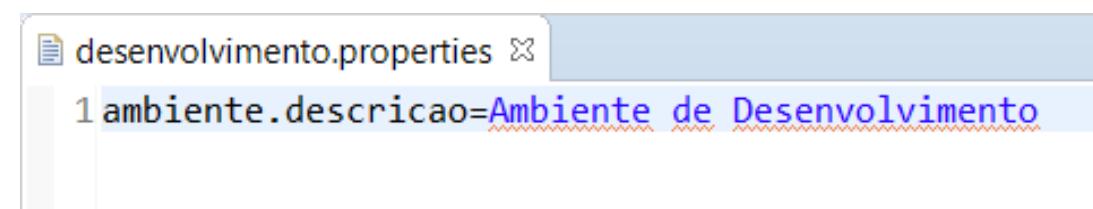
---

- Crie na pasta `src\main\filters` os arquivos abaixo, todos contendo uma variável `ambiente.descricao` com os respectivos valores:
  - `local.properties`
  - `desenvolvimento.properties`
  - `producao.properties`

```
local.properties
ambiente.descricao=Ambiente Local

desenvolvimento.properties
ambiente.descricao=Ambiente de Desenvolvimento

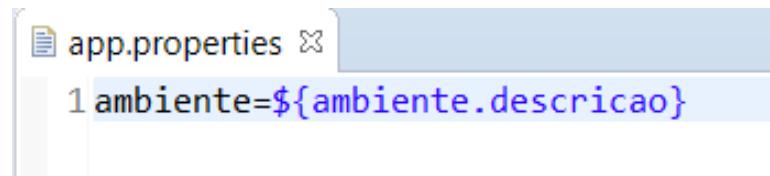
producao.properties
ambiente.descricao=
```



# Maven profile

---

- Altere o arquivo app.properties para usar uma notação especial do maven filtering que recupera a variável presente no arquivo filter que criamos no passo anterior.



A screenshot of a code editor showing the file "app.properties". The file contains the line "ambiente=\${ambiente.descricao}" with the variable "\${ambiente.descricao}" highlighted in blue, indicating it is being filtered.

```
app.properties
1 ambiente=${ambiente.descricao}
```

# Maven profile

---

- Agora, precisamos “ligar” o filtering para tipo de arquivo. O maven trata isso como um “resource”, que por sua vez é processado pelo maven-resources-plugin. Para isso, vamos configurar a tag <resources> no pom.xml:

```
<build>
    <finalName>${project.artifactId}</finalName>

    <resources>
        <resource>
            <directory>src/main/resources</directory>
            <filtering>true</filtering>
        </resource>
    </resources>

    (...)
```

# Maven profile

---

- Finalmente, podemos criar nossos profiles. Primeiro, vamos criar um profile “default”, que utiliza o arquivo de propriedades local.properties. A configuração é feita dentro da tag <profiles>. Repare que marcamos essa profile para ser ativa por padrão (activeByDefault = true).

```
<profiles>
  <profile>
    <id>default</id>
    <activation>
      <activeByDefault>true</activeByDefault>
    </activation>
    <build>
      <filters>
        <filter>src/main/filters/local.properties</filter>
      </filters>
    </build>
  </profile>
</profiles>
```

# Maven profile

---

- Todo configurado? Agora faça o teste:
  - mvn clean tomcat7:redeploy
- @TODO:
  - Renomar o profile “default” para “loc”
  - Criar os profiles “des” e “prod”, alterando para apontar para os respectivos arquivos de filter
  - Deixei apenas o profile “loc” como activeByDefault
- Em seguida, veja o resultado dos comandos:
  - mvn clean tomcat7:redeploy
  - mvn -P loc clean tomcat7:redeploy
  - mvn -P des clean tomcat7:redeploy
  - mvn -P prod clean tomcat7:redeploy

# Maven help plugin

---

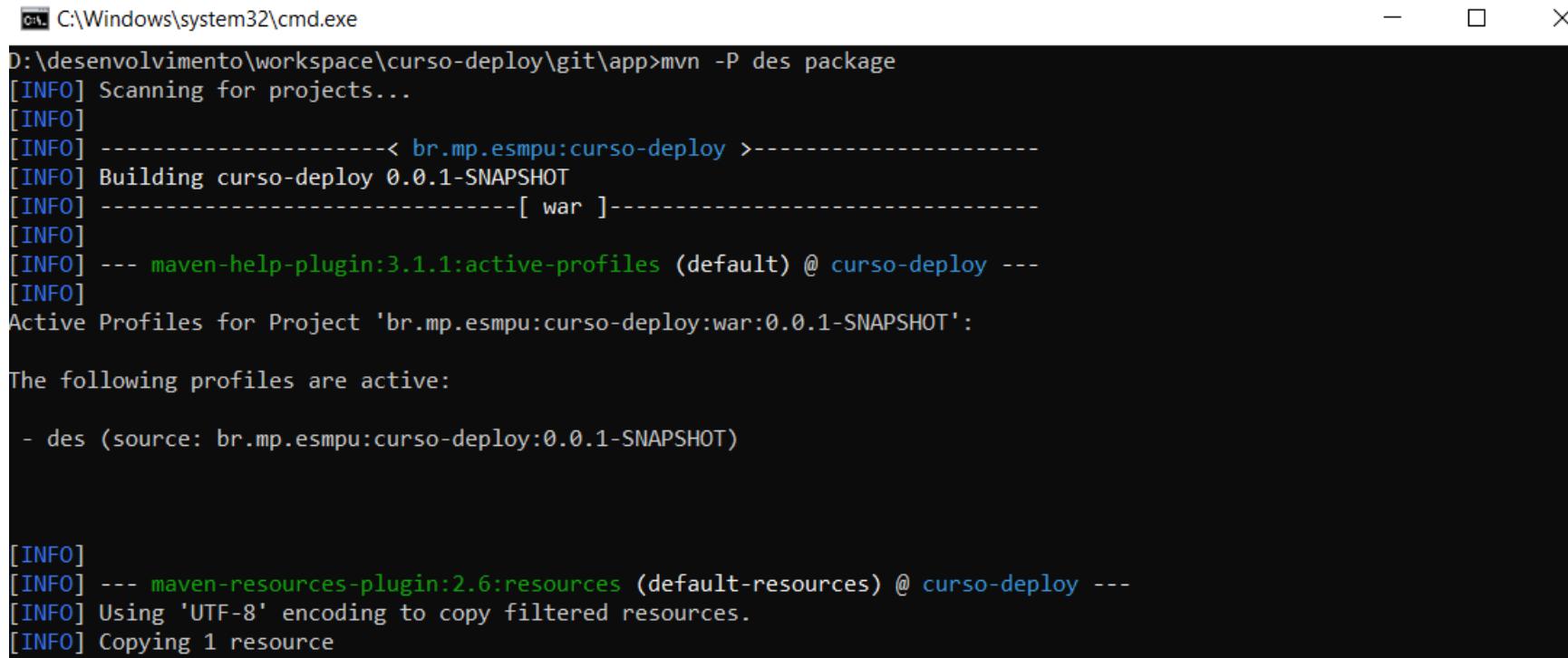
- É comum usarmos o maven-help-plugin para exibir os profiles ativos na execução da build. Ele possui um goal help:active-profiles.
- Vamos adicionar o plugin no pom.xml e vincular a uma fase do lifecycle.

```
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-help-plugin</artifactId>
  <version>3.1.1</version>
  <executions>
    <execution>
      <phase>initialize</phase>
      <goals>
        <goal>active-profiles</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

# Maven help plugin

---

- Agora, execute novamente algum comando do ciclo de vida, por exemplo:
  - mvn -P des package



```
C:\Windows\system32\cmd.exe
D:\desenvolvimento\workspace\curso-deploy\git\app>mvn -P des package
[INFO] Scanning for projects...
[INFO]
[INFO] -----< br.mp.esmpu:curso-deploy >-----
[INFO] Building curso-deploy 0.0.1-SNAPSHOT
[INFO] -----[ war ]-----
[INFO]
[INFO] --- maven-help-plugin:3.1.1:active-profiles (default) @ curso-deploy ---
[INFO]
Active Profiles for Project 'br.mp.esmpu:curso-deploy:war:0.0.1-SNAPSHOT':

The following profiles are active:

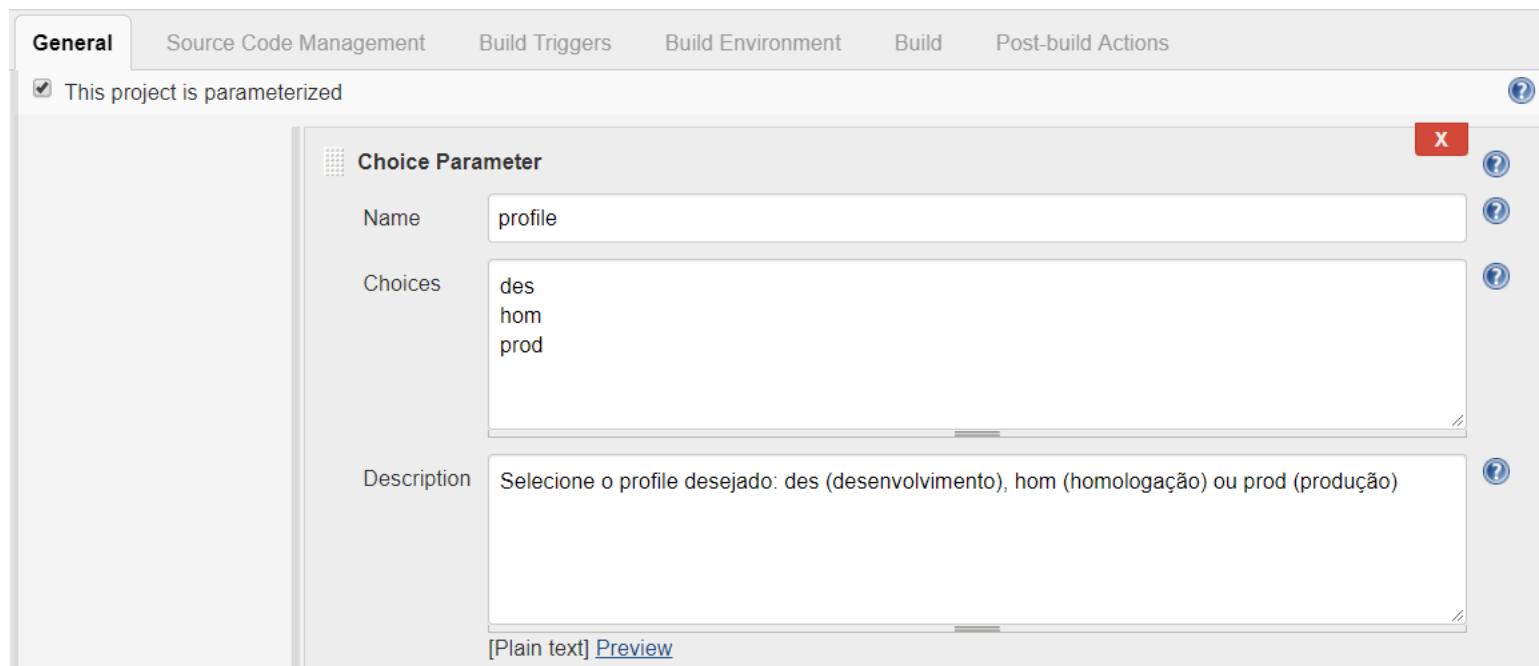
- des (source: br.mp.esmpu:curso-deploy:0.0.1-SNAPSHOT)

[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ curso-deploy ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] Copying 1 resource
```

# Dia 4

---

JENKINS PARAMETRIZADO | ARTIFACTORY

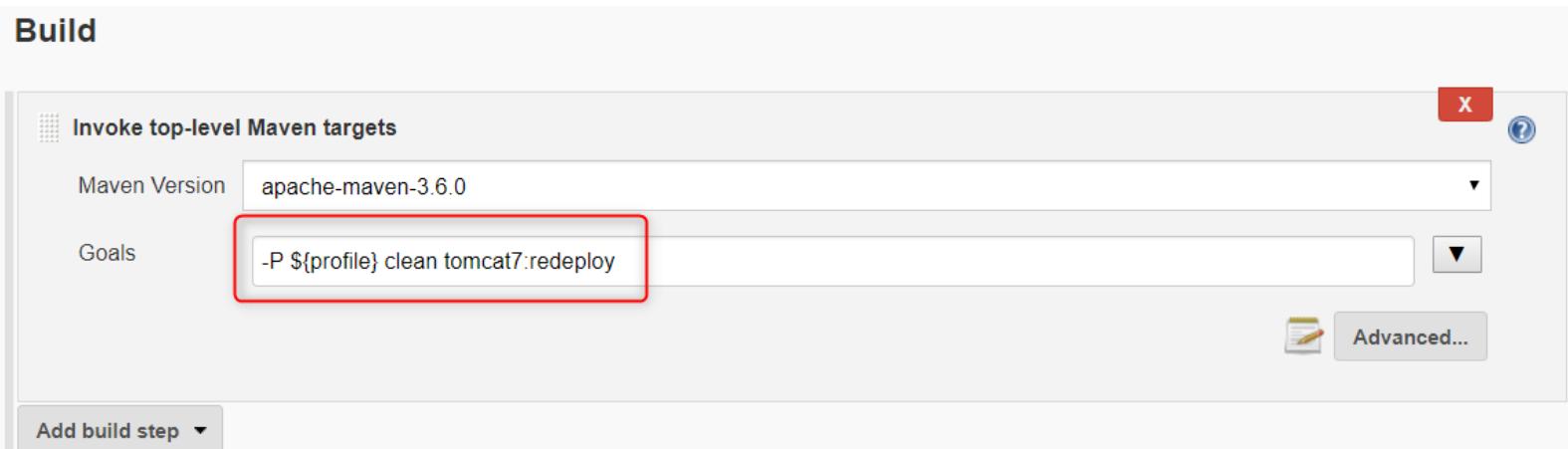


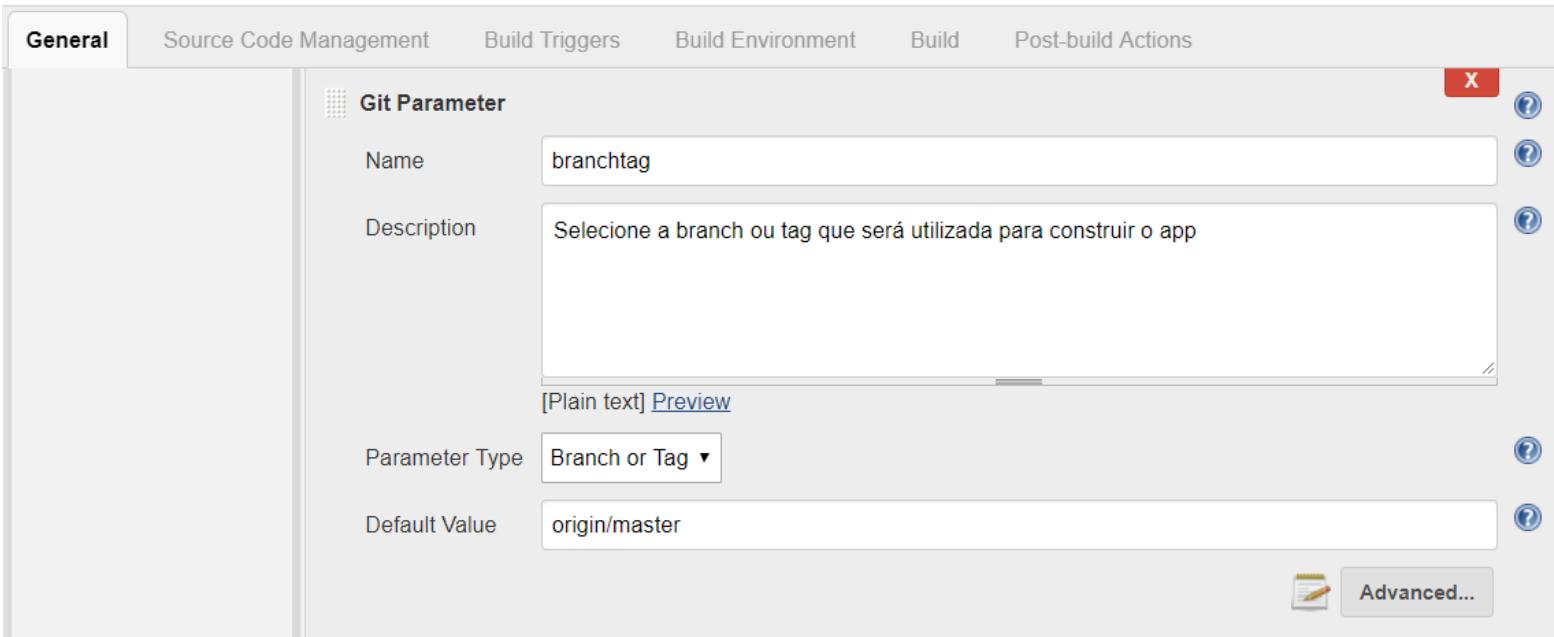
# Choice Parameter

- Exibe uma combo que permite escolher o profile desejado e armazena na variável \${profile}
- Valores possíveis:
  - des
  - hom
  - prod

# Choice Parameter

- Em seguida, altere o comando maven para utilizar o profile escolhido.





# Git Parameter

- Popula uma combo com todas as branches e tags do repositório
- Permite escolher qual branch ou tag será utilizada para construir a aplicação.

## Source Code Management

- None
- Git

### Repositories

Repository URL

Credentials

### Branches to build

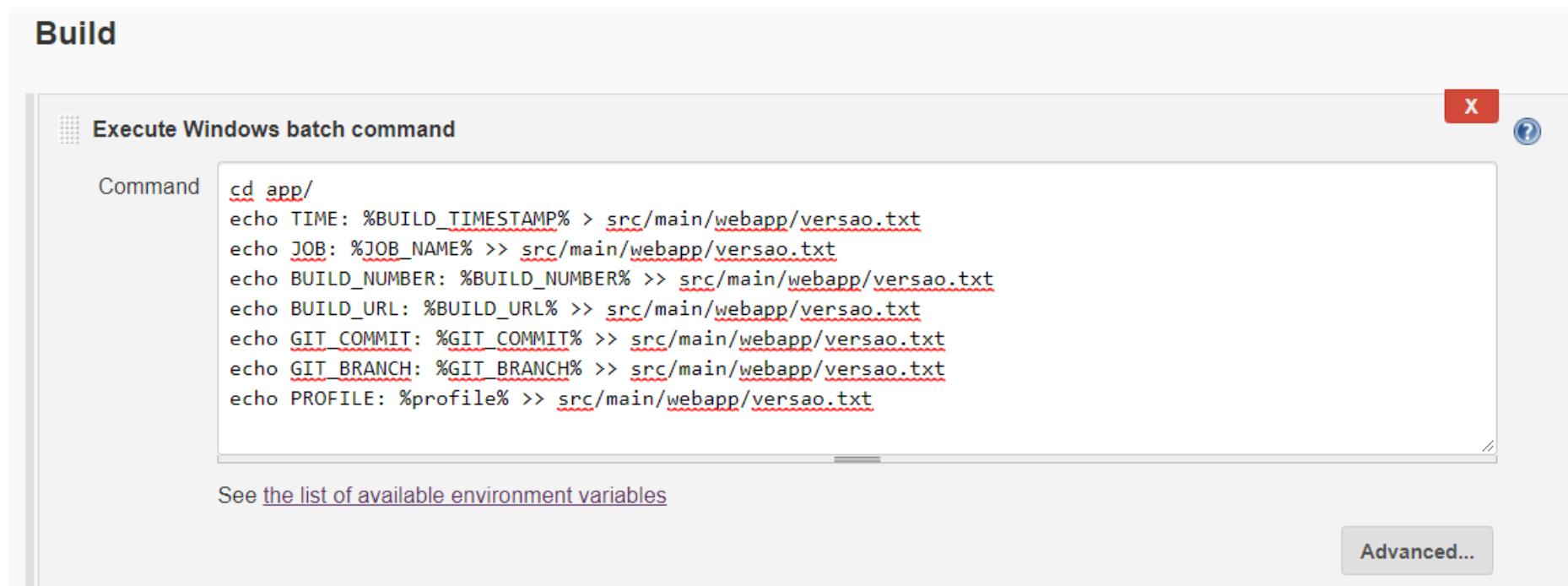
Branch Specifier (blank for 'any')

# Git Parameter

- Substitua o valor de “branches to build” com o valor que será selecionado na combo.

# Windows Batch Command

- Vamos disponibilizar um arquivo chamado versão.txt na raiz do contexto que contem informações úteis sobre o deploy:
  - <http://localhost:8180/curso-deploy/versao.txt>



# Windows Batch Command

---

- Fazemos um “echo” em algumas variáveis que o Jenkins disponibiliza.
- Esse build step deve ser executado antes do comando maven que realiza o package e deploy.

```
cd app/  
echo TIME: %BUILD_TIMESTAMP% > src/main/webapp/versao.txt  
echo JOB: %JOB_NAME% >> src/main/webapp/versao.txt  
echo BUILD_NUMBER: %BUILD_NUMBER% >> src/main/webapp/versao.txt  
echo BUILD_URL: %BUILD_URL% >> src/main/webapp/versao.txt  
echo GIT_COMMIT: %GIT_COMMIT% >> src/main/webapp/versao.txt  
echo GIT_BRANCH: %GIT_BRANCH% >> src/main/webapp/versao.txt  
echo PROFILE: %profile% >> src/main/webapp/versao.txt
```

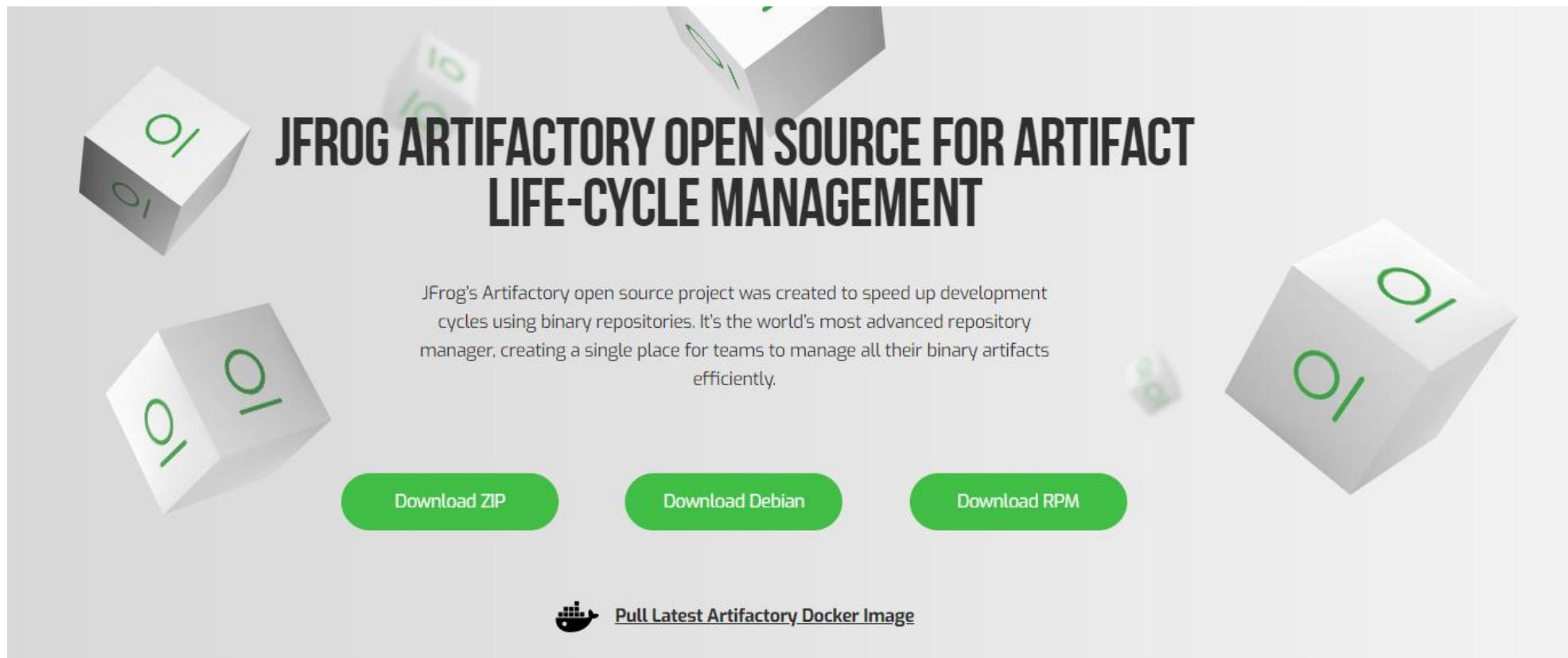
# Artifactory

---

# Artifactory

---

- <https://jfrog.com/open-source/>



The image shows the JFrog Artifactory Open Source landing page. The background features several white cubes with green 'O/' symbols floating in the air. The main title 'JFROG ARTIFACTORY OPEN SOURCE FOR ARTIFACT LIFE-CYCLE MANAGEMENT' is displayed in large, bold, black capital letters. Below the title is a descriptive paragraph: 'JFrog's Artifactory open source project was created to speed up development cycles using binary repositories. It's the world's most advanced repository manager, creating a single place for teams to manage all their binary artifacts efficiently.' At the bottom, there are three green buttons with white text: 'Download ZIP', 'Download Debian', and 'Download RPM'. To the left of the 'Download ZIP' button is a Docker logo icon. To the right of the 'Download ZIP' button is a link 'Pull Latest Artifactory Docker Image'.

JFROG ARTIFACTORY OPEN SOURCE FOR ARTIFACT  
LIFE-CYCLE MANAGEMENT

JFrog's Artifactory open source project was created to speed up development cycles using binary repositories. It's the world's most advanced repository manager, creating a single place for teams to manage all their binary artifacts efficiently.

Download ZIP   Download Debian   Download RPM

 Pull Latest Artifactory Docker Image

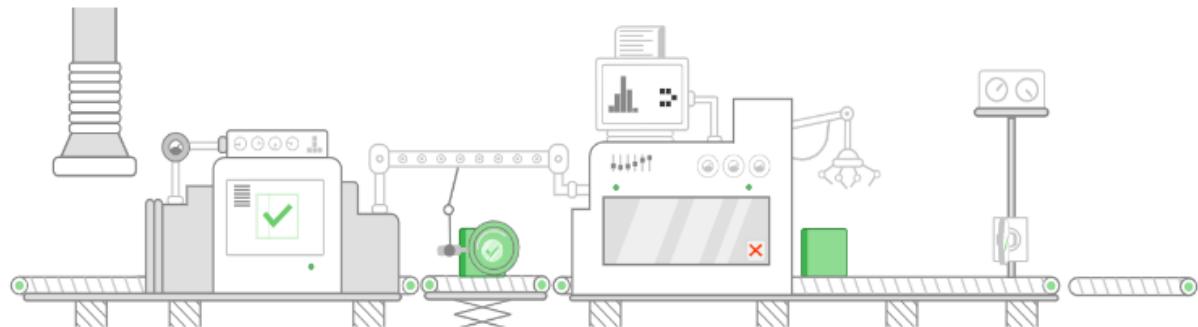
# Instalação artifactory

---



## Welcome to JFrog Artifactory!

This wizard will get your Artifactory set up in just a few short leaps.  
You can change any settings later – so let's begin!



Next

# Instalação artifactory

- Descompactar o arquivo “software\jfrog-artifactory-oss-6.8.4” (sugestão: pasta tool).
- Dentro da pasta bin, executar:
  - `artifactory.bat`
- Acessar:
  - <http://localhost:8081/artifactory>



# Set Admin Password

This new password is for the default admin user.  
Want to skip? Find the default admin credentials in the [JFrog Artifactory User Guide](#).

---

.....  
.....

Password Strength

███████████

Good

The screenshot shows a user interface for creating repositories. At the top left is a circular icon containing a stylized document with a green ribbon. The main title is "Create Repositories". Below it is a sub-instruction: "Select the package type(s) you want - we'll create the default repositories for you! Disabled package types already have default repositories configured." A horizontal line separates this from the repository list below.

Package Type	Status
Gradle	Available
Ivy	Available
Maven	Disabled (highlighted)
SBT	Available
Generic	Available
Bower	Available
Chef	Available
CocoaPods	Available
Conan	Available
CONDA	Available
CRAN	Available
debian	Available
docker	Available
Gems	Available
Git LFS	Available
GO	Available
HELM	Available
npm	Available
NuGet	Available
Opkg	Available
Composer	Available
PHP	Available
PyPI	Available
Puppet	Available
VAGRANT	Available
rpm	Available

# Instalação artifactory

- Senha do admin:
    - @dmin123
  - Tela “Configure a Proxy Server”: Skip
  - Tela “Create Repositories”: selecione Maven



## Artifactory on-boarding complete!

Congrats! These are the default repositories we created for you.

You're now ready to speed up your software releases!

Want to configure your client(s) and get started? Click the [Set Me Up](#) button for each repository.

Want to learn more about different repository types? Consult the [JFrog Artifactory User Guide](#).

**maven**

libs-snapshot-local libs-release-local jcenter libs-snapshot libs-release

# Instalação artifactory

---

- **Repositórios padrão criados:**
  - libs-snapshot-local
  - libs-release-local
  - jcenter
  - libs-snapshot
  - libs-release

# Remote repositories

---

- Repositório em cache que serve como um proxy a um repositório em determinada URL.
- Os artefatos remotos são baixados sob demanda, somente quando requisitados.
- O repositório do primefaces foi configurado como um repositório no pom.xml. Vamos centralizar tudo no artifactory e configurá-lo como um repositório remoto.

# Remote repositories

- Vá em Admin > Repositories > Remote e clique em “New”
- Selecione como Package Type o Maven

Edit primefaces Repository

Successfully connected to server

Basic Advanced Replications

Package Type \*

**maven**

Maven

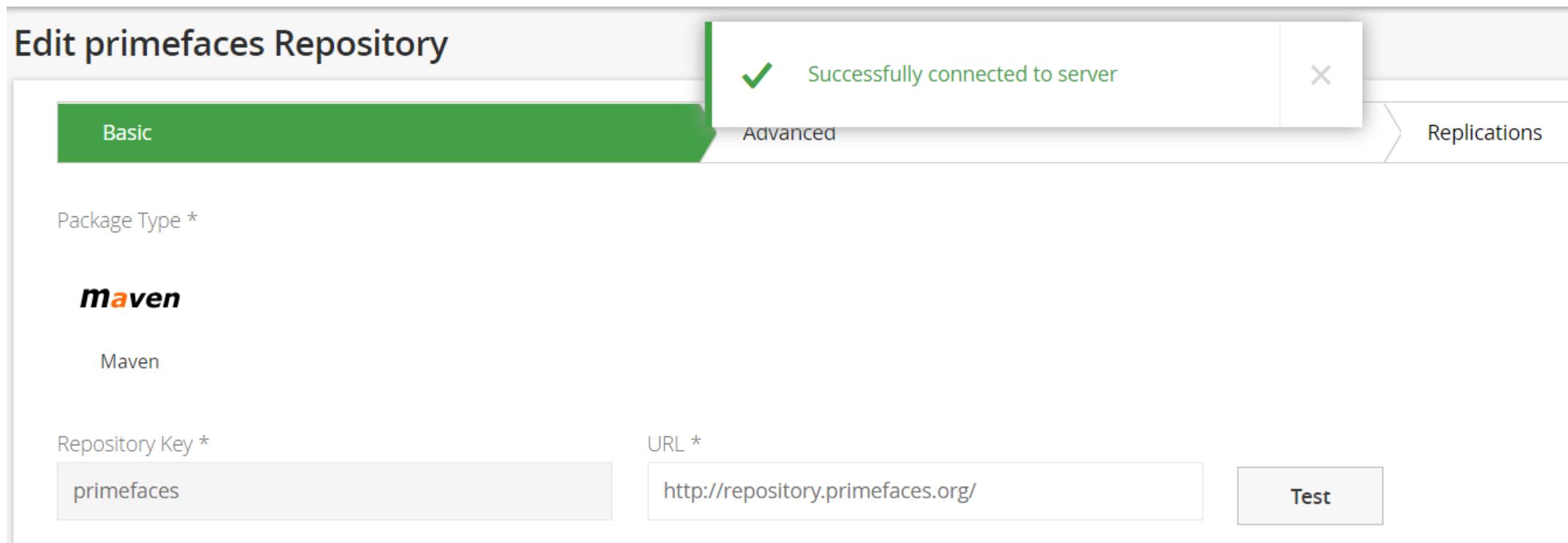
Repository Key \*

primefaces

URL \*

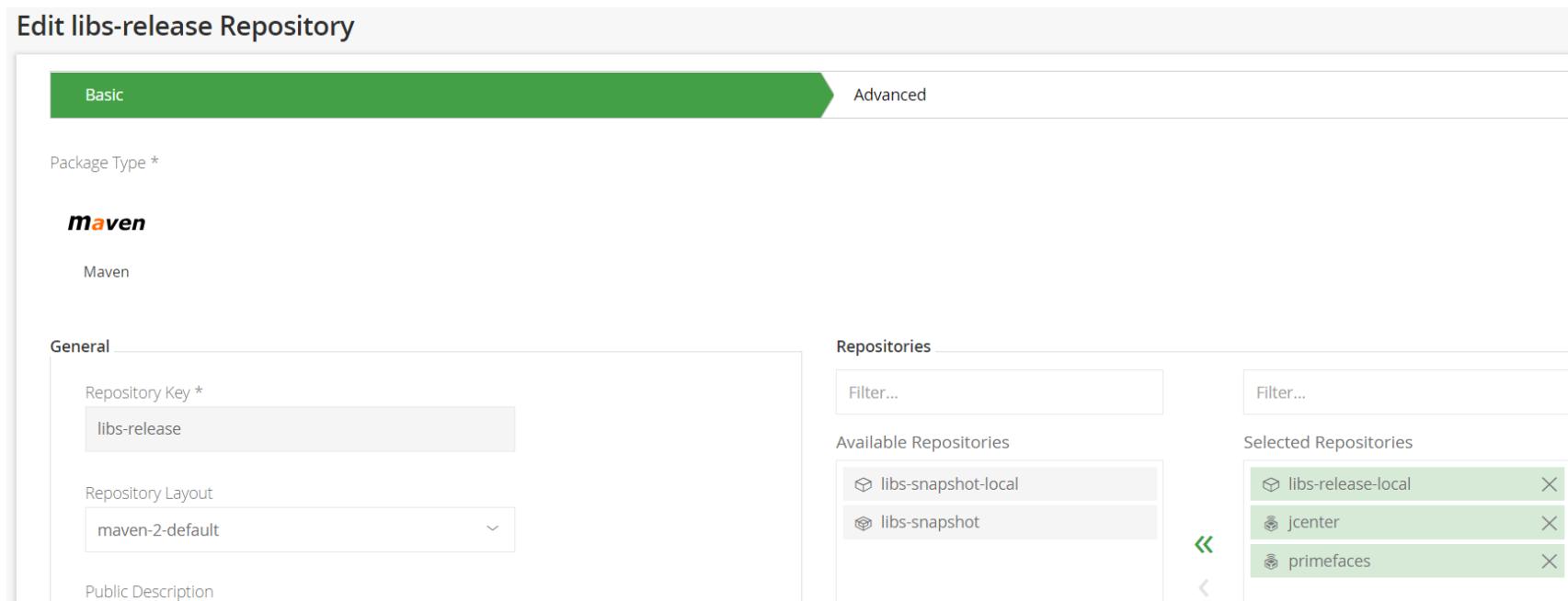
<http://repository.primefaces.org/>

Test



# Virtual repositories

- O Artifactory agrupa vários repositórios, locais ou remotos, em repositórios virtuais. Isso é feito por facilidade e organização.
- Dessa forma, precisamos adicionar o repositório remoto primefaces ao repositório virtual lib-releases, que utilizamos no pom.xml.



# Virtual repositories

---

- Pra finalizar, podemos agora remover a entrada do repositório primefaces do pom.xml, deixando apenas o repositório virtual lib-releases do artifactory.

```
<repositories>
  <repository>
    <id>central</id>
    <name>Local Artifactory Lib Releases</name>
    <url>http://localhost:8081/artifactory/libs-release</url>
  </repository>
</repositories>
```

- O <id> do repositório é central. Alguma ideia? Veja o Effective POM.
- Execute o comando mvn package novamente e confira o resultado.

# Cache repositories

- Quando um artefato é baixado, o artifactory cria um novo “repositório” com o sufixo cache, indicando que está fisicamente presente no servidor.
- No caso, a biblioteca primefaces 7.0.RC3 está armazenada no repositório primefaces-cache.

## Artifact Repository Browser

The screenshot shows the Artifactory Repository Browser interface. On the left, there's a tree view of repositories:

- libs-release
- libs-snapshot
- artifactory-build-info
- libs-release-local
- libs-snapshot-local
- jcenter
- jcenter-cache
- primefaces
- primefaces-cache
- org/primefaces/primefaces/7.0.RC3

The 'primefaces-cache' node is selected and highlighted in blue. At the bottom of the tree view are 'Trash Can' and a refresh icon.

The right panel displays the details for the 'primefaces-cache' repository. It has three tabs: General (selected), Effective Permissions, and Properties.

**General Tab (Info Section):**

Name:	primefaces-cache
Package Type:	Maven
Repository Path:	primefaces-cache/
Repository Layout:	maven-2-default
Artifact Count / Size:	Show
Created:	13-03-19 16:06:57 -03:00

**Virtual Repository Associations:**

- libs-release

# Cache repositories

---

- TAREFA:
  - Quais repositórios cache existem no artifactory? Quantas bibliotecas?
  - Delete algumas bibliotecas locais do maven (pasta .m2)
  - Execute o comando mvn package
  - Veja o que aconteceu com os repositórios cache
  - Quantos artefatos o Artifactory está servindo?



# Dia 5

---

SONARQUBE | CONSIDERAÇÕES FINAIS | PRÓXIMOS PASSOS

# Sonarqube

---

# Sonarqube

---

- <https://www.sonarqube.org/>

The screenshot shows the official SonarQube website. At the top left is the SonarQube logo. To its right are navigation links: FEATURES, DOWNLOADS, ROADMAP, COMMUNITY, and BLOG. A blue banner at the top center indicates "SonarQube 7.6 released on January 28th!" with a "See what's new" link. Below this, the text "The leading product for Continuous Code Quality" is displayed, followed by large, bold text: "Continuous" in dark grey and "Code Quality" in blue. To the right, three boxes represent "Code Smells" (radioactive symbol), "Bugs" (bug icon), and "Vulnerabilities" (padlock icon). At the bottom, there are download options ("DOWNLOAD" and "USE ONLINE"), a badge stating "Used by more than 85,000 organizations", and a footer menu listing supported languages: "On 25+ languages" (with a right arrow), Java, JavaScript, C#, C/C++, COBOL, and "AND MORE".

# Instalação sonarqube

---

# Sonarqube

---

- <https://docs.sonarqube.org/latest/setup/get-started-2-minutes/>
- Descompactar o arquivo “sonarqube-7.6” (sugestão: pasta tools)
- Entrar na pasta “tool\sonarqube-7.6\bin” e executar:
  - StartSonar.bat
- Acessar:
  - <http://localhost:9000/>
  - Usuário: admin
  - Senha: admin