

# Maven & Git & PRs

---

## Maven & Git & PRs

### Maven

What is Maven ?

Why do we need Maven?

Roles

Maven Project Architecture

Repository & Pom.xml

Pom.xml Decomposing

Maven UUIDs

Dependency

Dependency parent-inherit

Properties

Plugins

Example

Commands

IntelliJ run maven

Maven Build Life Cycle (important)

Maven Repositories

Maven Local Repository

Maven Central(Remote) Repository

Maven Remote(Private) Repository

Maven Dependency Search Sequence

Questions

Git(VCS - tool), Github(website)

Commands

Important Concepts

How to raise a PR?(Pull Request)

Git HW submission基本操作: (从git下载项目)

References

## Maven

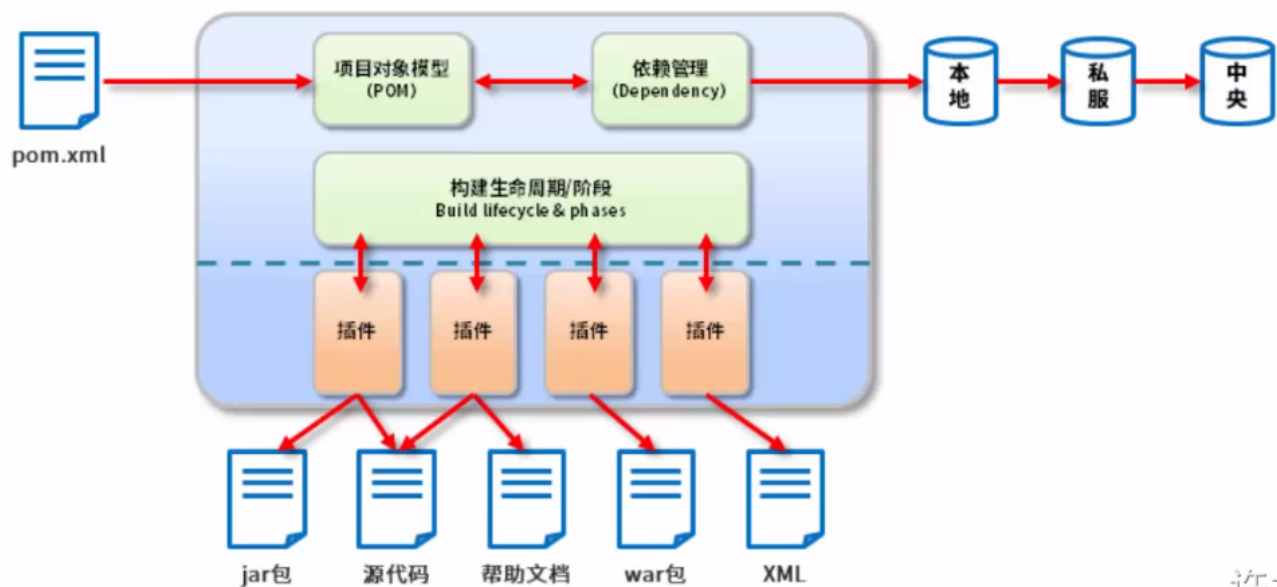
---

### What is Maven ?

- Maven 的本质是一个项目管理工具，将项目开发和管理过程抽象成一个项目对象模型（POM）。
- **POM.xml**（Project Object Model）：项目对象模型。

# Why do we need Maven?

Easy to download add and remove the dependencies/libraries



许大仙

- 项目构建：提供标准的、跨平台的自动化项目构建方式。
- 依赖管理：方便、快捷的管理项目依赖的资源（jar包），避免资源间的版本冲突问题。
- 统一开发结构：提供标准的、统一的项目结构。

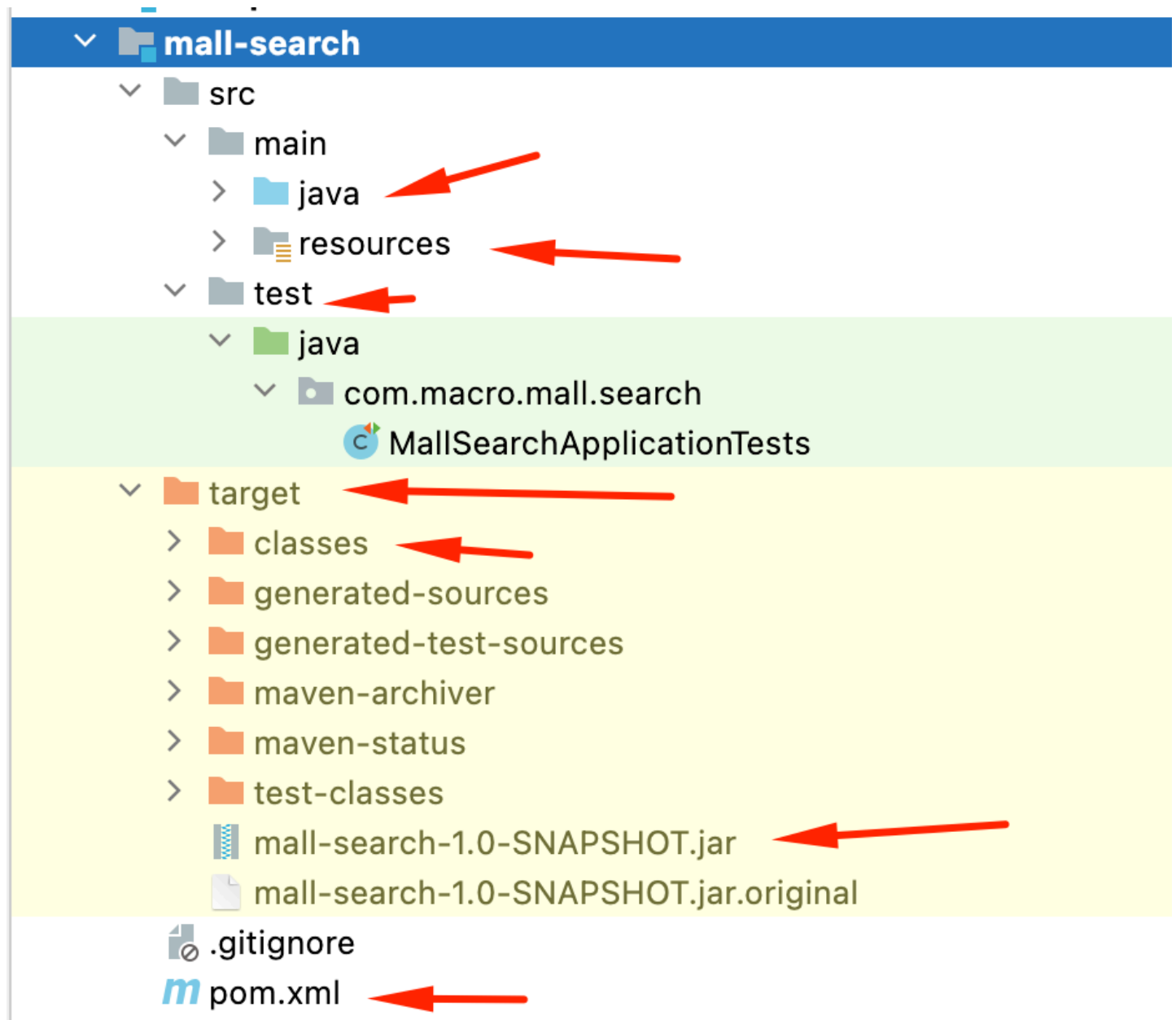
## External Libraries

- > < 1.8 > /Library/Java/JavaVirtualMachines/jdk1.8.0\_271.jdk/Contents/Home
- > Maven: ch.qos.logback:logback-classic:1.2.3
- > Maven: ch.qos.logback:logback-core:1.2.3
- > Maven: cn.hutool:hutool-all:5.4.0
- > Maven: com.alibaba:druid:1.1.23
- > Maven: com.alibaba:druid-spring-boot-starter:1.1.23
- > Maven: com.aliyun.oss:aliyun-sdk-oss:2.5.0
- > Maven: com.carrotsearch.thirdparty:simple-xml-safe:2.7.1
- > Maven: com.carrotsearch:hppc:0.8.1
- > Maven: com.fasterxml.jackson.core:jackson-annotations:2.11.0
- > Maven: com.fasterxml.jackson.core:jackson-core:2.11.0
- > Maven: com.fasterxml.jackson.core:jackson-databind:2.11.0
- > Maven: com.fasterxml.jackson.dataformat:jackson-dataformat-cbor:2.11.0
- > Maven: com.fasterxml.jackson.dataformat:jackson-dataformat-smile:2.11.0
- > Maven: com.fasterxml.jackson.dataformat:jackson-dataformat-yaml:2.11.0
- > Maven: com.fasterxml.jackson.datatype:jackson-datatype-jdk8:2.11.0
- > Maven: com.fasterxml.jackson.datatype:jackson-datatype-jsr310:2.11.0
- > Maven: com.fasterxml.jackson.module:jackson-module-parameter-names:2.11.0
- > Maven: com.fasterxml:classmate:1.5.1
- > Maven: com.github.jsqlparser:jsqlparser:3.2
- > Maven: com.github.pagehelper:pagehelper:5.2.0

## Roles

- Manage **Dependencies**(Package).
- **Build** Project (Cycle)
- Documentation
- Reporting
- Others

## Maven Project Architecture



Item	Default
source code	\${basedir}/src/main/java
Resources	\${basedir}/src/main/resources
Tests	\${basedir}/src/test
Compiled byte code	\${basedir}/target
distributable JAR	\${basedir}/target/classes

## Repository & Pom.xml

### Pom.xml Decomposing

#### Maven UIDs

```

1  <groupId>com.chuwa.mall</groupId>
2  <artifactId>oms</artifactId>
3  <version>1.0-SNAPSHOT</version>
4  <packaging>jar</packaging> #WAR

```

produce jar file: **artifactId - version . packaging**

```
oms-1.0-SNAPSHOT.jar
```

Example:

```

1  <dependency>
2    <groupId>junit</groupId>
3    <artifactId>junit</artifactId>
4    <version>3.8.1</version>
5    <scope>test</scope>
6  </dependency>

```

```

→ junit ls
3.8.1 3.8.2 4.11 4.12 4.13 4.13.1 4.13.2 4.8.2
→ junit cd 3.8.1
→ 3.8.1 ls
_remote.repositories      junit-3.8.1.jar.sha1
junit-3.8.1-sources.jar   junit-3.8.1.pom
junit-3.8.1-sources.jar.sha1 junit-3.8.1.pom.sha1
junit-3.8.1.jar          m2e-lastUpdated.properties

```

## Dependency

```
1      <dependencies>
2          <dependency>
3              <groupId>org.springframework.boot</groupId>
4              <artifactId>spring-boot-starter-actuator</artifactId>
5          </dependency>
6          <dependency>
7              <groupId>org.springframework.boot</groupId>
8              <artifactId>spring-boot-starter-aop</artifactId>
9          </dependency>
10         <dependency>
11             <groupId>org.springframework.boot</groupId>
12             <artifactId>spring-boot-starter-test</artifactId>
13             <scope>test</scope>
14         </dependency>
15         <dependency>
16             <groupId>org.projectlombok</groupId>
17             <artifactId>lombok</artifactId>
18         </dependency>
19         <dependency>
20             <groupId>org.springframework.boot</groupId>
21             <artifactId>spring-boot-configuration-processor</artifactId>
22             <optional>true</optional>
23         </dependency>
24     </dependencies>
```

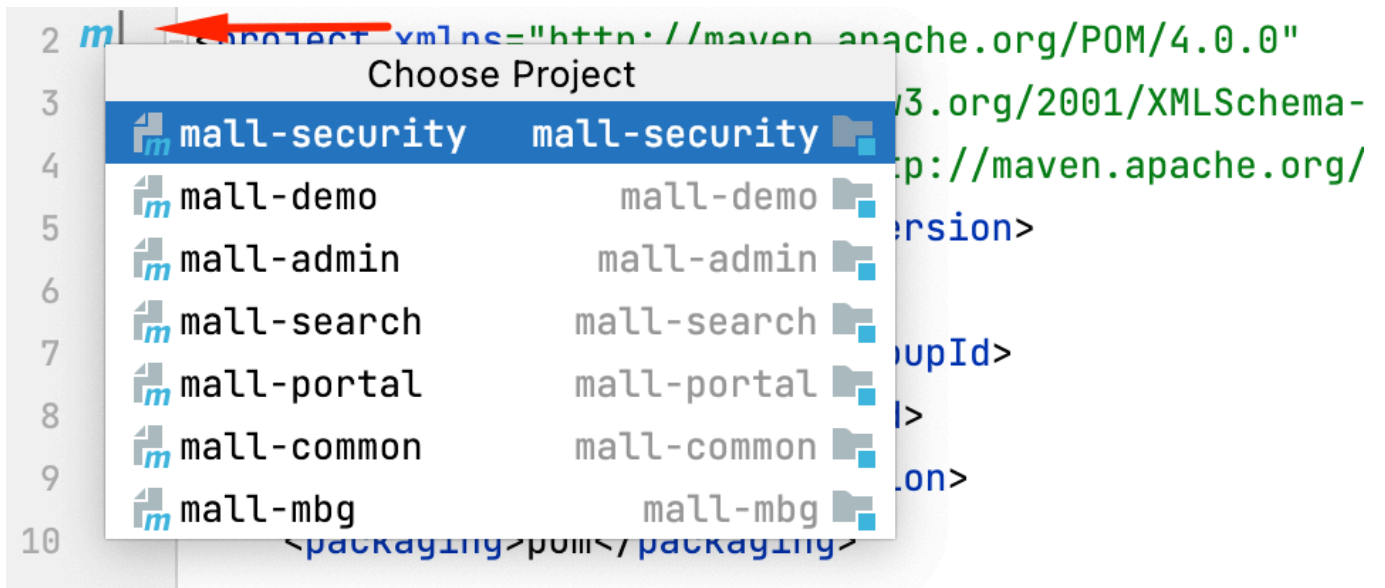
## Dependency parent-inherit

类似于Java OOP中的继承，当前pom.xml可继承来自于parent pom.xml中的配置。也可以override。

check the parent's pom.xml

```
1      <parent>
2          <groupId>org.springframework.boot</groupId>
3          <artifactId>spring-boot-starter-parent</artifactId>
4          <version>2.3.0.RELEASE</version>
5          <relativePath/> <!-- lookup parent from repository -->
6      </parent>
```

该pom.xml 被哪些继承了。



该pom.xml继承了谁

```

5      <modelVersion>4.0.0</modelVersion>
6
7      <parent>
8          <groupId>org.slf4j</groupId>
9          <artifactId>slf4j-parent</artifactId>
10         <version>1.7.30</version>
11     </parent>
12
13     <artifactId>jul-to-slf4j</artifactId>
14
15     <packaging>jar</packaging>
16     <name>JUL to SLF4J bridge</name>
17     <description>JUL to SLF4J bridge</description>
18
19     <url>http://www.slf4j.org</url>
20
21     <dependencies>
22         <dependency>
23             <groupId>org.slf4j</groupId>
24             <artifactId>slf4j-api</artifactId>
25         </dependency>

```

## Properties

`${mysql-connector.version}` use the value configured in properties.

```

1      <properties>
2          <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
3          <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
4          <java.version>1.8</java.version>

```

```

5      <mysql-connector.version>8.0.25</mysql-connector.version>
6  </properties>
7
8  <dependencies>
9      <!--Mysql数据库驱动-->
10     <dependency>
11         <groupId>mysql</groupId>
12         <artifactId>mysql-connector-java</artifactId>
13         <version>${mysql-connector.version}</version>
14     </dependency>
15
16     <dependency>
17         <groupId>mysql2</groupId>
18         <artifactId>mysql-connector-java</artifactId>
19         <version>${mysql-connector.version}</version>
20     </dependency>
21 </dependencies>

```

## Plugins

### Example

```

1  <build>
2      <plugins>
3          <plugin>
4              <groupId>org.apache.maven.plugins</groupId>
5              <artifactId>maven-antrun-plugin</artifactId>
6              <version>1.1</version>
7              <executions>
8                  <execution>
9                      <id>id.clean</id>
10                     <phase>clean</phase>
11                     <goals>
12                         <goal>run</goal>
13                     </goals>
14                     <configuration>
15                         <tasks>
16                             <echo>clean phase</echo>
17                         </tasks>
18                     </configuration>
19                 </execution>
20             </executions>
21         </plugin>
22     </plugins>
23 </build>

```

### Commands



## Commands

```
mvn [plugin-name]:[goal-name]
```

e.g. `mvn compiler:compile`

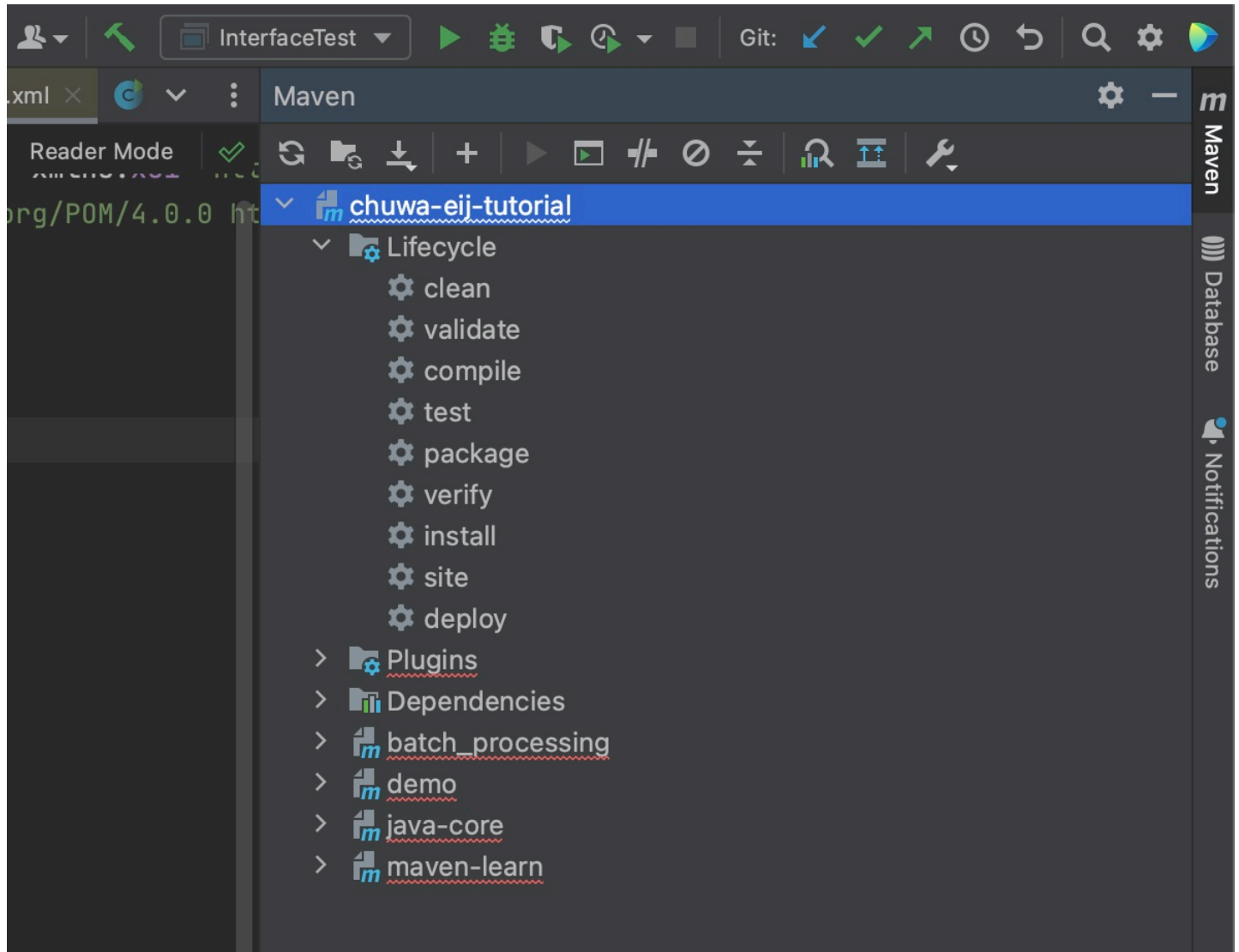
creating project in terminal:

```
1 $> mvn archetype:generate
2 -DgroupId = com.chuwa.app
3 -DartifactId = helloworld
4 -DarchetypeArtifactId = maven-archetype-quickstart
5 -DinteractiveMode = false
```

```
1 mvn compile                #Compiles source code and stores classes to target/classes
2 mvn validate               #Validates project
3 mvn test                   #Runs tests
4 mvn clean                  #Deletes target directory
5 mvn clean package          #Compiled code is packaged to WAR/JAR/deb etc
6 mvn clean install          #Install the artifact in local repository(.m2 Repo)
7 mvn clean deploy           #Copies the package(WAR/JAR/deb etc) to the remote
                             repository
8 mvn verify
9 mvn clean verify
10
11 # Options
12 mvn clean install -Dmaven.test.skip=true           #Skips compiling and running
                             tests
13 mvn clean install -DskipTests=true                 #Compiles but skips running
                             tests
14 mvn clean install -Dmaven.test.failure.ignore=true #Compiles and executes tests but
                             ignores if any tests failed
15 mvn verify -Dtest=TestName                         #Executes specified test
16 mvn clean install -T 4                             #-T is used to specify number
                             of threads used, default 2 i.e., 2 threads per CPU.
17 mvn clean install -X/--debug                       #Enables debug mode
18 mvn clean package -U/--update-snapshots            #Force check on dependency
                             updates
19 mvn dependency:purge-local-repository               #Removes local repository
20
21 # Dependency Plugin
22 mvn dependency:analyze                             #Analyzes dependencies of the
                             project
23 mvn dependency:tree                                #Prints dependency tree
24 mvn versions:display-dependency-updates            #Displays dependency updates
25 mvn dependency:analyze -DignoreNonCompile=true     #Shows unused dependencies
26
```

```
20
27 # Create java project (JAR):
28 mvn archetype:create -DgroupId=org.onecompiler.project -DartifactId=one-compiler -
    DarchetypeArtifactId=maven-archetype-quickstart
```

## IntelliJ run maven



## Maven Build Life Cycle (important)

每次build都是从起点到制定phase，只有Test可以跳过。

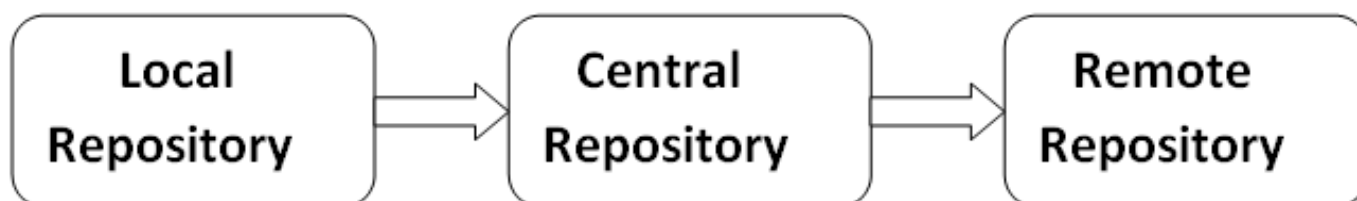
```
mvn clean install -Dmaven.test.skip=true
```

mvn Clean -> prepare-resources -> validate -> package -> install

Phase	Handles	Description
prepare-resources	resource copying	Resource copying can be customized in this phase.
validate	Validating the information	Validates if the project is correct and if all necessary information is available.
compile	compilation	Source code compilation is done in this phase.
Test	Testing	Tests the compiled source code suitable for testing framework.
package	packaging	This phase creates the JAR/WAR package as mentioned in the packaging in POM.xml.
install	installation	This phase installs the package in local/remote maven repository.
Deploy	Deploying	Copies the final package to the remote repository.

## Maven Repositories

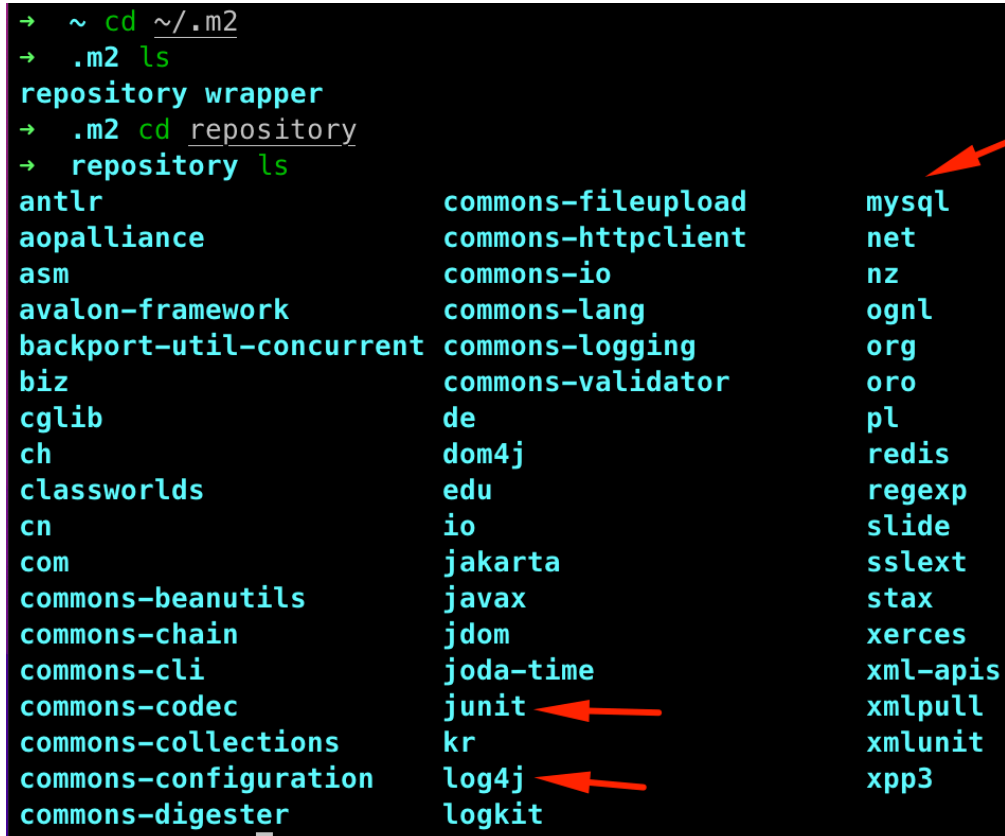
**Local repository** then **Central(Maven Public) repository** then **Remote repository**.



## Maven Local Repository

Cached the dependencies in your local machine.

```
1 cd ~/.m2
2 ls
3 cd repository
```



```
→ ~ cd ~/.m2
→ .m2 ls
repository wrapper
→ .m2 cd repository
→ repository ls
antlr                commons-fileupload  mysql
aopalliance          commons-httpclient  net
asm                  commons-io           nz
avalon-framework     commons-lang         ognl
backport-util-concurrent commons-logging      org
biz                   commons-validator   oro
cglib                 de                  pl
ch                    dom4j               redis
classworlds          edu                 regexp
cn                    io                  slide
com                   jakarta             sslex
commons-beanutils     javax               stax
commons-chain         jdom                xerces
commons-cli           joda-time           xml-apis
commons-codec         junit
commons-collections  kr
commons-configuration log4j
commons-digester     logkit
```

## Maven Central(Remote) Repository

Maven **central repository** is located on the web. It has been created by the **apache maven community** itself.

- The path of central repository is: <http://repo1.maven.org/maven2/>.
- The central repository contains a lot of common libraries that can be viewed by this url <https://central.sonatype.com/search>
- Get dependencies: <https://mvnrepository.com/>
  -



## JUnit » 4.13.2

JUnit is a unit testing framework for Java, created by Erich Gamma and Kent Beck.

License	EPL 1.0
Categories	Testing Frameworks
Organization	JUnit
HomePage	<a href="http://junit.org">http://junit.org</a>
Date	(Feb 13, 2021)
Files	<a href="#">jar (375 KB)</a> <a href="#">View All</a>
Repositories	<a href="#">Central</a> <a href="#">Minebench</a> <a href="#">Lutece Paris</a> <a href="#">Xceptance</a>
Used By	<b>117,509 artifacts</b>

[Maven](#)
[Gradle](#)
[Gradle \(Short\)](#)
[Gradle \(Kotlin\)](#)
[SBT](#)
[Ivy](#)
[Grape](#)
[Leiningen](#)
[Buildr](#)

```
<!-- https://mvnrepository.com/artifact/junit/junit -->
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.13.2</version>
  <scope>test</scope>
</dependency>
```

### Maven Remote(Private) Repository

Maven **remote repository** is located on the web. Most of libraries can be missing from the central repository such as JBoss library etc, so we need to define remote repository in pom.xml file.

Maven远程库也是位于网络上的存储库。例如一个公司可能有很多共享的jar包文件，就可以搭建一个公司内部的远程库，供众多开发人员使用；中央库可以认为是一个特殊的远程库。

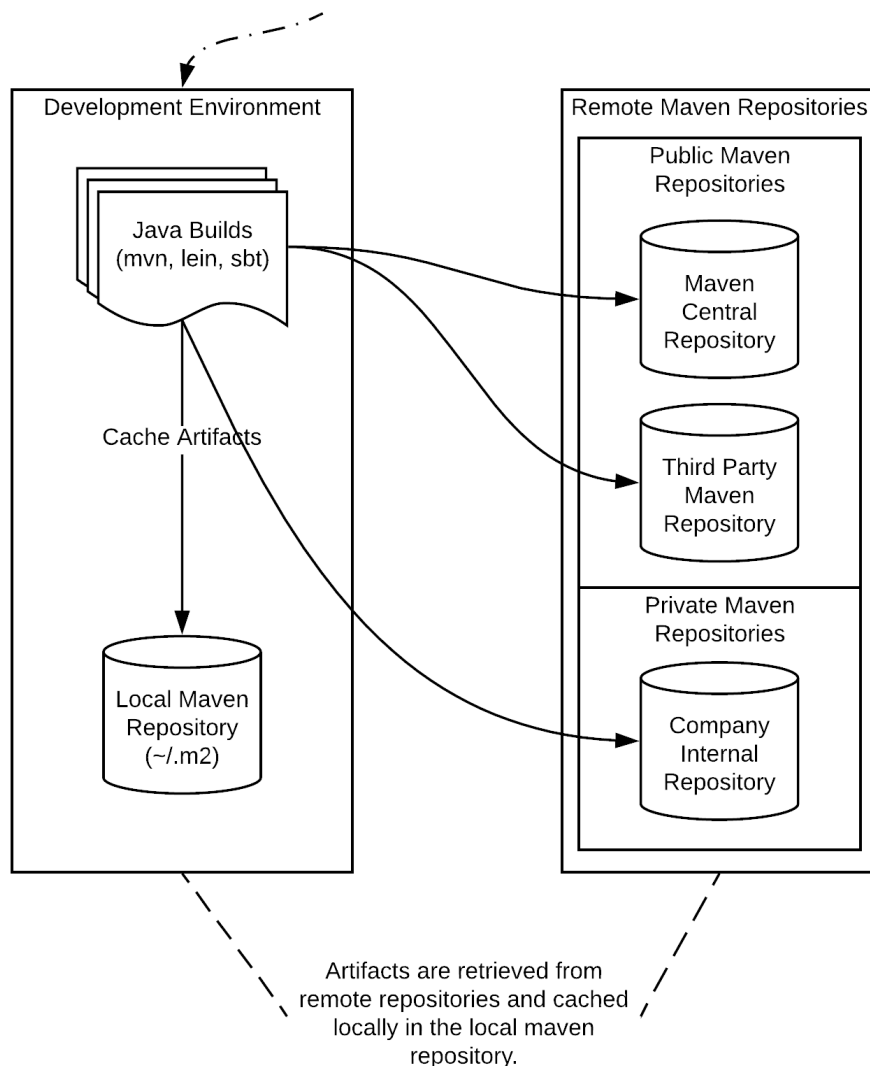
```
1  <repositories>
2    <repository>
3      <id>Redhat.Jboss.lib2</id>
4      <url>https://repository.jboss.org/</url>
5    </repository>
6
7    <repository>
8      <id>maven.companynamename.com.lib1</id>
9      <url>http://download.companynamename.org/maven2/lib1</url>
10   </repository>
11 </repositories>
```

### Maven Dependency Search Sequence

# Maven Repositories

This diagram illustrates how maven repositories fit within the software development lifecycle.

A Maven Development Environment can be on a local developer machine, or part of a continuous integration and deployment environment



When we execute Maven build commands, Maven starts looking for dependency libraries in the following sequence –

- **Step 1** – Search dependency in **local repository**, if not found, move to step 2 else perform the further processing.
- **Step 2** – Search dependency in **central repository**, if not found and remote repository/repositories is/are mentioned then move to step 4. Else it is downloaded to local repository for future reference.
- **Step 3** – If a **remote repository** has not been mentioned. Maven simply stops the processing and throws

Step 3 – If remote repository has not been mentioned, Maven simply stops the processing and throws error (Unable to find dependency). (公司内部自己开发的lib, 其它开源平台上的dependency, ie. Redhat)

- **Step 4** – Search dependency in remote repository or repositories, if found then it is downloaded to local repository for future reference. Otherwise, Maven stops processing and throws error (Unable to find dependency).

## Questions

- Which tool in **Node.js/Python** is similar to **Maven** ?

## Git(VCS - tool), Github(website)

### Commands

```
1  git init # init a git repo(repository-->source code)
2  git status
3  git add .
4  git commit -m "some commit message"
5  git push origin feature-branch
6
7  # 不commit当前的changes, 但是要切换到别的branch, 则stash changes,回来后再pop
8  # 演示IntelliJ上的操作
9  git stash
10 git stash pop
11
12 #Branches/Feature (Trunk-based development)
13 git branch branch_name
14 git checkout branch_name
15 git checkout -b branch_name # create and checkout to branch_name
16
17 # Git Remote
18 git remote -v
19 origin https://github.com/TAIsRich/chuwa-eij.git (fetch)
20 origin https://github.com/TAIsRich/chuwa-eij.git (push)
21 upstream https://github.com/BlgO/Everything-In-Java.git (fetch)
22 upstream https://github.com/BlgO/Everything-In-Java.git (push)
23
24 git remote add upstream https://github.com/BlgO/Everything-In-Java.git
25 git remote add upstream https://github.com/BlgO/Everything-In-Java11.git
26 git remote add upstreamWhateverName https://github.com/BlgO/Everything-In-Java11.git
27
28 git add "localfiles.java"
29 git commit -m 'something commit message'
30 git push
```

git branch + # branch 相关的命令

git remote + # remote(远程) 相关的命令

## Important Concepts

```
git push github-url branchName
```

我们用**origin/upstream** 来代替很长的url

**Origin** -> <https://github.com/TAlRich/chuwa-eij.git>

**Upstream** -> <https://github.com/sth/else.git>

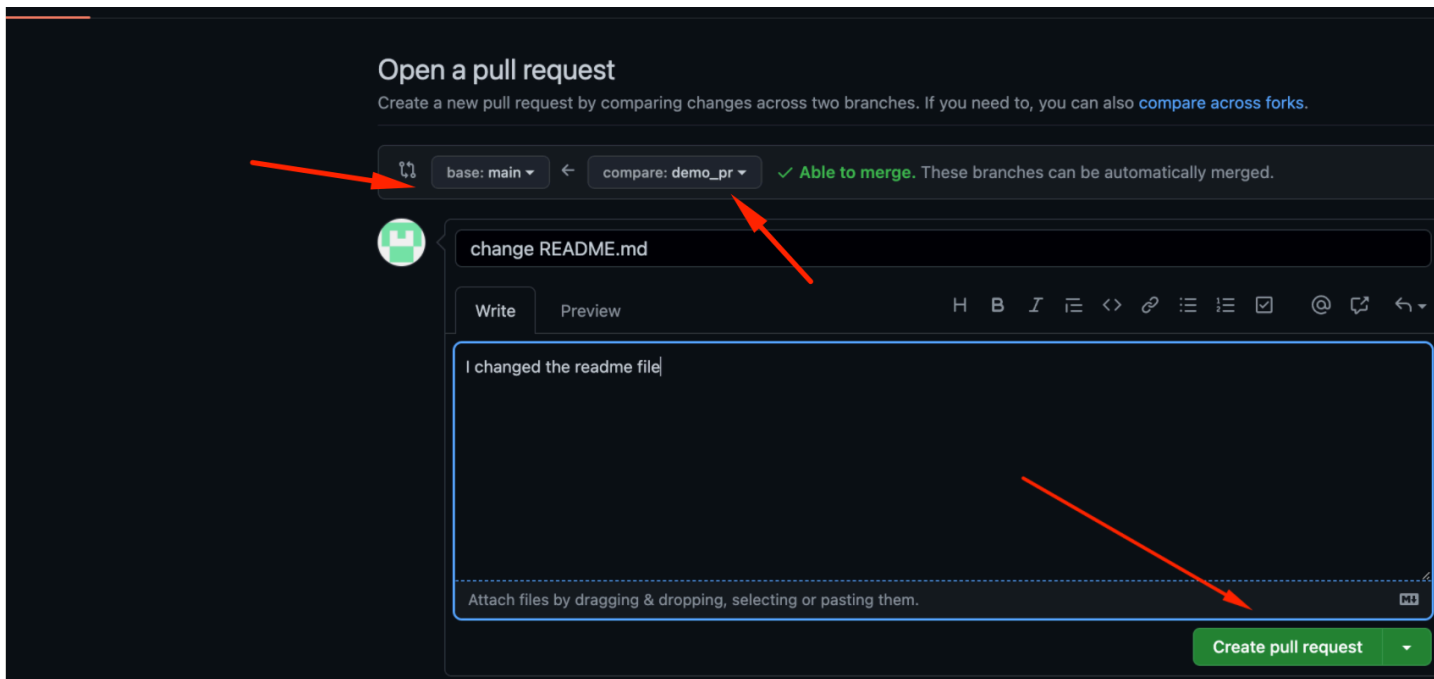
## How to raise a PR?(Pull Request)

PR是基于branch的

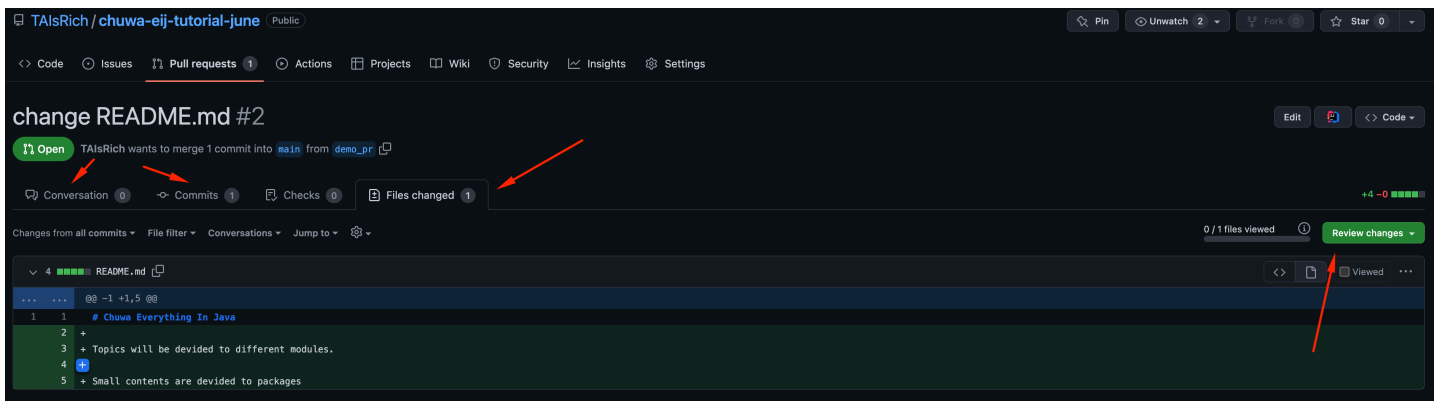
The screenshot shows the GitHub interface for the repository 'TAlRich / chuwa-eij-tutorial-june'. At the top, there's a navigation bar with links to Code, Issues, Pull requests, Actions, Projects, Wiki, Security, Insights, and Settings. Below this, a notification bar indicates that 'demo\_pr' had recent pushes less than a minute ago, with a green button labeled 'Compare & pull request' highlighted by a red arrow. Below the notification, there's a section for branches and tags, showing 'main' as the selected branch, 1 branch, and 0 tags. Below this, a list of recent commits is displayed, including 'TAlRich change jdk to 1.8' and several other commits related to 'init project' and 'Create README.md'. At the bottom, the 'README.md' file is visible, with the title 'Chuwa Everything In Java'.

Commit Message	Commit Hash	Time Ago	Commits
TAlRich change jdk to 1.8	3904248	6 minutes ago	4 commits
java-core change jdk to 1.8		6 minutes ago	
maven-learn init project		1 hour ago	
.gitignore init project		1 hour ago	
README.md Create README.md		11 minutes ago	
pom.xml init java-core		12 minutes ago	





Noticed that specify from **demo\_pr** to **main(master)**



## change README.md #2

 Open TAlsRich wants to merge 1 commit into `main` from `demo_pr` 

 Conversation 1  Commits 1  Checks 0  Files changed 1



TAlsRich commented 3 minutes ago

Owner   Tip ...

I changed the readme file

 change README.md

f631dd5



TAlsRich reviewed now

[View changes](#)

TAlsRich left a comment

Owner Author   Tip ...

please add more details

Add more commits by pushing to the `demo_pr` branch on TAlsRich/chuwa-eij-tutorial-june.



 Continuous integration has not been set up  
GitHub Actions and several other apps can be used 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.




Write Preview


H B I        

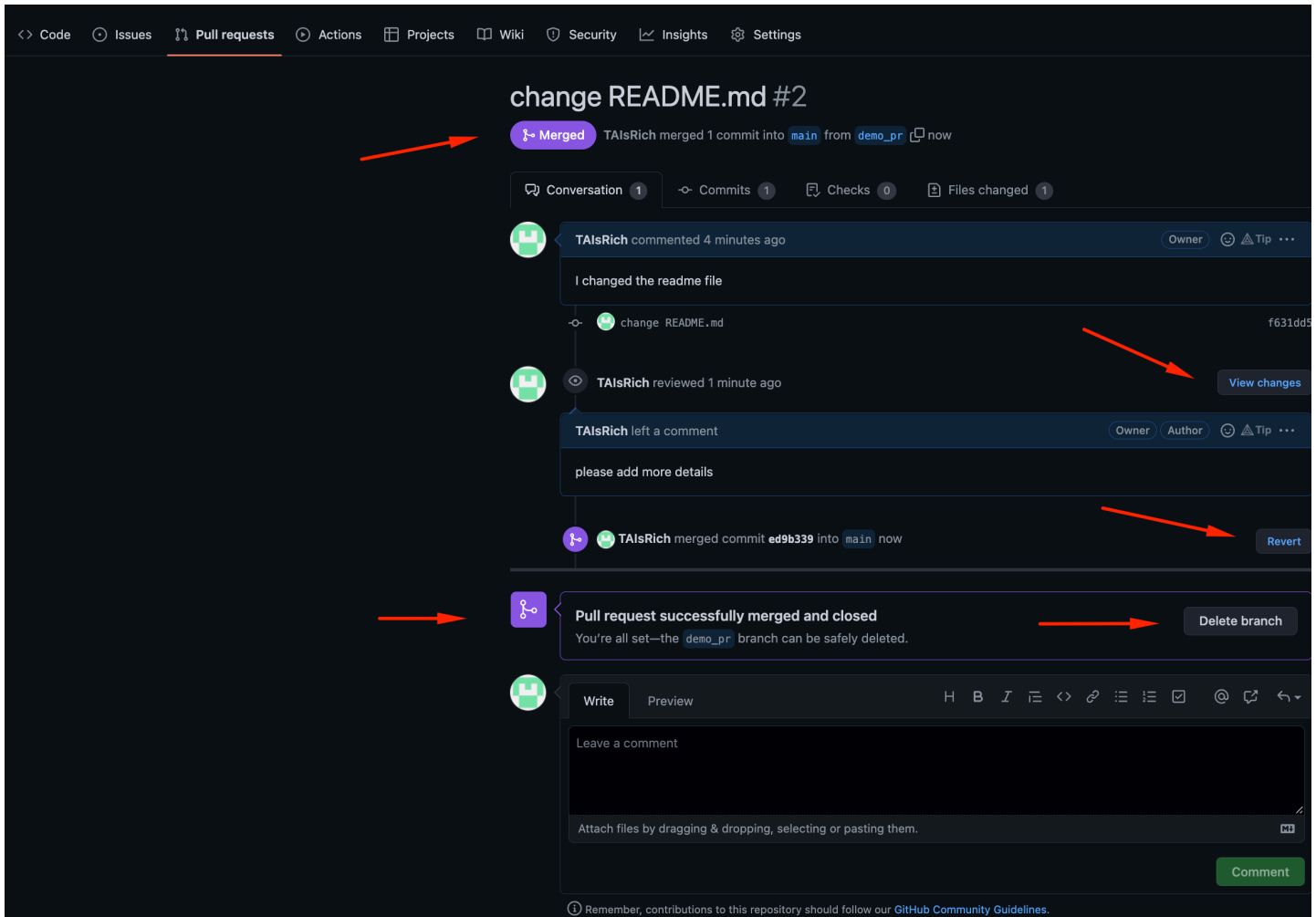
Leave a comment

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

 Close pull request

[Comment](#)

 Remember, contributions to this repository should follow our [GitHub Community Guidelines](#).



## Git HW submission基本操作：（从git下载项目）

1. 新建文件夹 命令是`mkdir`
2. 进入文件夹`cd`
3. 下载项目
  1. Git页面点code, 复制https链接
  2. 在下, **git clone** <https 链接>
4. 创建**notes** branch
  1. `cd` (比如chuwa888是名字, 先ls找到名字)
  2. `git branch <firstname_lastname/notes>` 创建一个notes branch 在本地
  3. `git checkout <firstname_lastname/notes>` 切换到新创建的notes branch
  4. `git push origin <firstname_lastname/notes>` 上传notes branch去远程
5. 创建**hw** branch并修改
  1. `git branch <firstname_lastnamehw1>` 创建hw branch在本地

2. git checkout <firstname\_lastname/hw1> 切换到hw branch
  3. 在hw branch上写作业
6. 上传修改
1. 打开IntelliJ 的commit tab
  2. 选上所有的changes, 右键add to VCS
  3. 打开 Terminal , git commit -m "" commit 当前的change
  4. git push origin <firstname\_lastname/hw1> 把当前本地hw branch的commit 上传到远程
7. 交pr
1. 打开git, 有个 compare & pull request, 点击进去
  2. Compare branch选 hw branch, base branch选 notes branch
  3. 点击create pull request

## Referencies

---

- basic introduction: <https://www.youtube.com/watch?v=KNGQ9JBQWhQ>
- Maven:
  - <https://www.yuque.com/fairy-era/yg511q/xdugky>
  - <https://www.tutorialspoint.com/maven/index.htm>
  - <https://www.javatpoint.com/maven-repository>