# 创建Gitlab项目仓库（这里的helloworld也是我的firstproject）

# 1. 基于SpringBoot项目构建

## 1.方法：SPRING INITIALIZR(推荐)：

### 1.打开网址https://start.spring.io/

### （推荐使用的是，gradle.project,但是 项目中报错了，所以我使用的是marven）

1. **根据项目选型选择构建工具Gradle或Maven,这里我选择了Gradle**
2. **版本号一般就用其默认的最新发布版，这里默认是1.4.3**
3. **输入组织唯一标识Group,这里我输入了com.minixiao**
4. **输入项目唯一标识Artifact,这里我输入了firstprojct**
5. **在Search for dependencies可以输入需要依赖的项目，然后选择添加**
6. **推荐点击Switch to the full version，可以更方便的看到全部SpringBoot官方提供的Starter.**
7. 这里你点击Switch to the full version后可以看到，我们可以选择Packaging方式，和Java Version(我们采用JDK1.8),因为这些数据都会最后生成到build.gradle中去，推荐项目中Packaging采用Jar的打包方式，这样因为在打包是会自动将所需要的依赖一起打进jar包，最后项目构建产生的jar包只需要在装有JRE环境上的计算机上就能运行，无需再下载SpringBoot相关依赖或者是其他项目中引入的依赖，这也是SpringBoot官方推荐的fat jar打包方式，一次打包到处运行。
8. 后面您就能看见很多多选项，这就能看到很多SpringBoot官方提供的Starter,我们根据项目中用到的技术选择即可，这里简单介绍几个常用的选项。
   1. **Web** Full-stack web development with Tomcat and Spring MVC

其实这里的英文说明已经说明的很清楚了“全栈式的WEB开发，集成了Tomcat和Spring MVC”,我们在项目中直接可以使用SpringMVC相关技术，直接用就行，很多配置都不需要，Tomcat就自动打包进我们的jar包了，直接有了内置的Tomcat容器，运行的计算机上只需要JRE环境，而不需要再安装jetty或者tomcat这种web容器，这也是SpringBoot的一大特色，简化配置，更多的关注业务开发，快速构建和运行。

* 1. **Thymeleaf** Thymeleaf templating engine, including integration with Spring

模版引擎，自动和Spring集成。同类模版引擎还有Freemarker，Groovy Templates等等，我们团队内推荐用Thymeleaf，在前后端未完全分离的情况下，Thymeleaf模版技术可以很好的保留前端原样，前后端协作在某种程度上说是很方便的，更多用法查看[thymeleaf官网](http://thymeleaf.org/)。

* 1. **JPA** Java Persistence API including spring-data-jpa, spring-orm and Hibernate

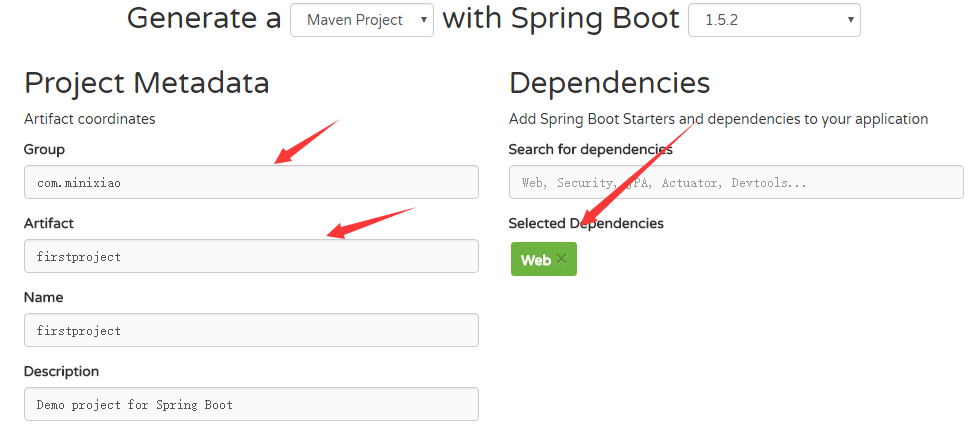
包含了spring-data-jpa, spring-orm and Hibernate，这位面向对象开发来说节省了不少成本，不过要高效率的使用，还需要多研读[spring-data-jpa官方文档](http://docs.spring.io/spring-data/jpa/docs/1.10.4.RELEASE/reference/html/#repositories.special-parameters)和[Hibernate文档](http://hibernate.org/orm/documentation/5.2/)

* 1. **PostgreSQL** PostgreSQL jdbc driver

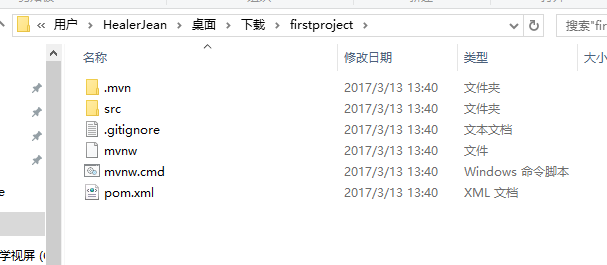
PostgreSQL 驱动是少不了的，根据项目需要可以选择Mysql等等其他数据库驱动。

* 1. **其他依赖**

根据项目需要，按需选择，具体到对应的依赖，可以看其注释说明就能知道其作用，按需引用即可。

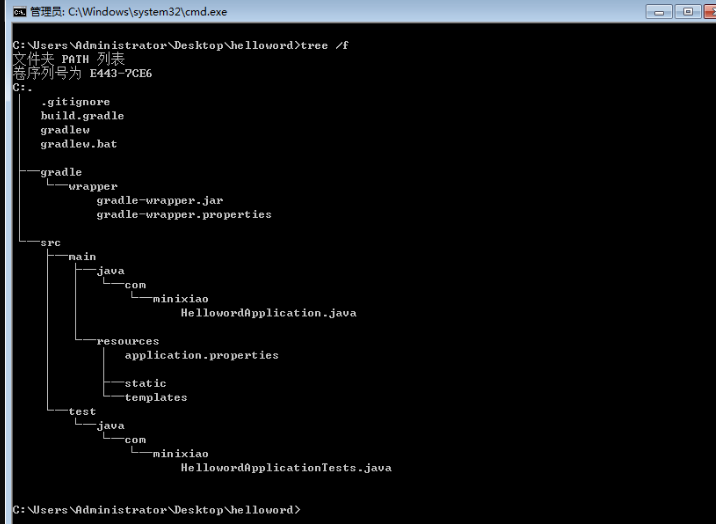


## 2.点击下载



## 3.目录解析

解压后用tree命令查看目录树结构如下：



**目录简单解析**：

\* `.gitignore` git版本控制忽略说明文件，更具项目可自行添加忽略目录，生成的gitignore已满足大多数场景

\* `build.gradle` 在整个构建工具中常用的文件，就像maven中pom.xml的地位一样，可以引入各种插件，各种依赖，设置依赖版本，介入构建过程等等的东西。

\* `gradlew` 这个文件很多时候我们很少关注，你用文本编辑其打开就能看到这样一样注释：`Gradle start up script for UN\*X`，这就是Gradle初始化脚本，设置一些环境等等。

\* `gradlew.bat` 这个文件很多时候我们很少关注，你用文本编辑其打开就能看到这样一样注释：`Gradle startup script for Windows`，这你就懂了吧？和gradlew对应。

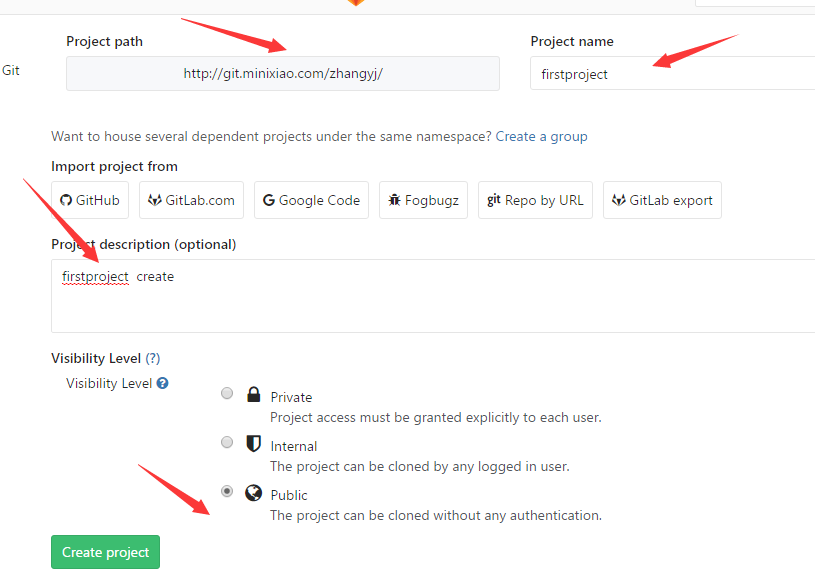
\* `gradle文件夹` 主要是wrapper文件夹下的gradle-wrapper.properties和gradle-wrapper.jar，作用是达到团队中其他人未安装会自动进行安装gradle等等，具体可以上官网查查。

\* `src文件夹` 主要代码放置地，包含main和test文件夹，这我猜你也懂了。还值得一说的是main文件夹下的resources文件夹，下面存在\*\*application.pro

## 4. **提示**：如果本地未使用过springboot 1.4.3.RELEASE初次构建需要等待下载依赖的jar包。

# 2.创建project项目

1. 创建帐号并登录 > 此次演示未在git.minixiao.com上创建帐号和而是在gitlab.com上创建helloworld项目进行演示，道理都一样。 创建账户登录不用过多说明，内部用git.minixiao.com帐号注册通道已经关闭，注册需联系管理员。
2. 创建firstproject项目
3. **输入Project name 项目名称**
4. **输入Project desciption，虽然是可选的，还是比较重要，让人第一眼了解你的项目。**
5. **根据项目需要选择项目Visibility Level项目可见等级，firstproject演示项目 ，这里直接公开即可。**
6. **点击Create project不出意外你可以看到下图的样子，恭喜你管理仓库已经建好了。**



# 3. 从远端仓库Clone克隆至本地

## 1.使用下面的命令即可克隆岛本地

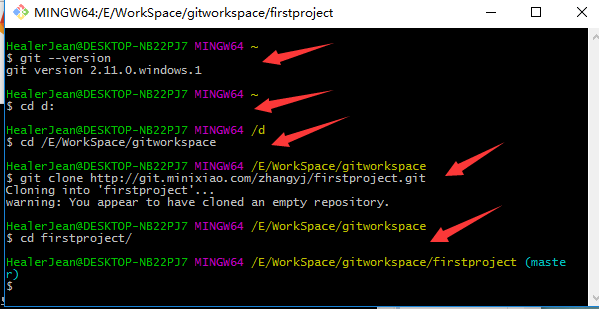
$ git --version

$ cd d:

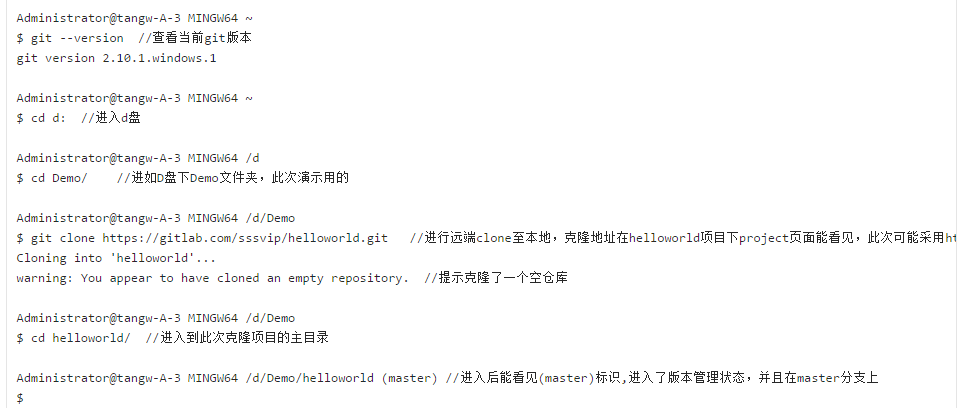
$ cd /E/WorkSpace/gitworkspace

$ git clone http://git.minixiao.com/zhangyj/firstproject.git

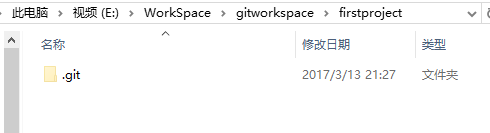
$ cd firstproject/



## 2.解释命令

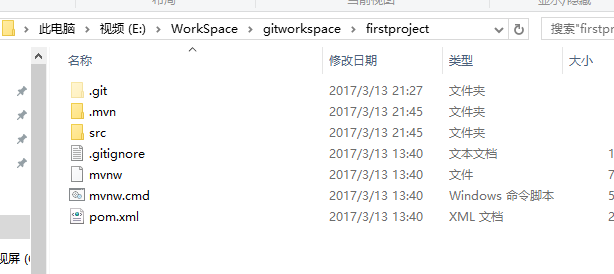


## 3.检测是不是成功的（.git隐藏起来的，上面的firstproject是自动创建的）

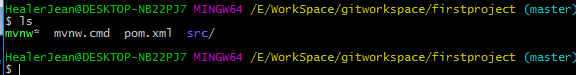


# 4. 从本地仓库push推至远端仓库

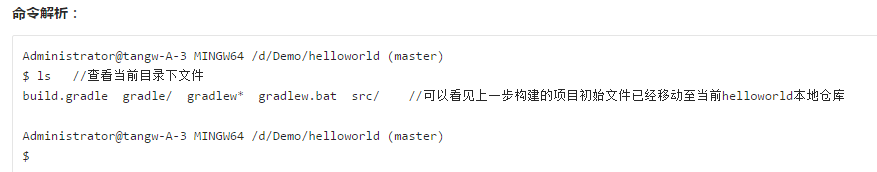
## 1.将刚刚添加的项目文件存储到本地仓库并push推到远端gitlab仓库（拷贝1中下载的文件到项目中去）



## 2.测试命令文件显示



## 3.命令解析



## 4.真正操作上传

**$ git status**

**$ git add -A$ git config user.name zhangyujin**

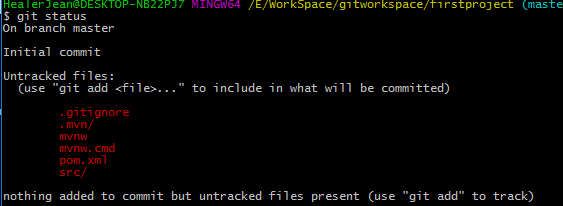
**$ git config user.email** [**zhangyj@minixiao.com**](mailto:zhangyj@minixiao.com)

**$ git commit -m "feat: init firstproject project file"**

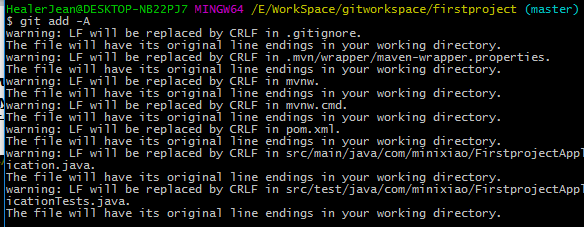
**$ git push origin**

### 1.$ git status 查看本地仓库状态--这里主要想看下本地仓库文件变化情况

Untracked files: //这里提示有未跟踪的文件



### 2.$ git add -A //提交当前变化的所有文件，git命令可上官方文档查一查

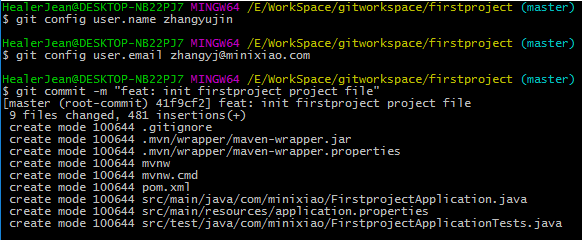


### 3.$ git config user.name zhangyujin

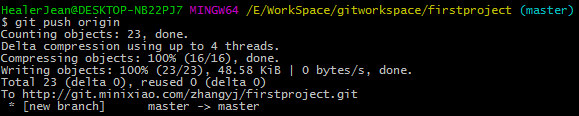
**$ git config user.email** [**zhangyj@minixiao.com**](mailto:zhangyj@minixiao.com)

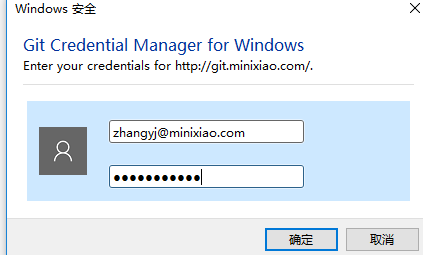
### 4.$ git commit -m "feat: init firstproject project file"

/提交刚add的文件到本地仓库，-m后面我加了注视初始化项目文件

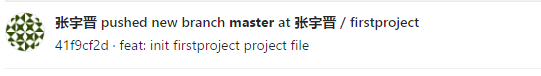


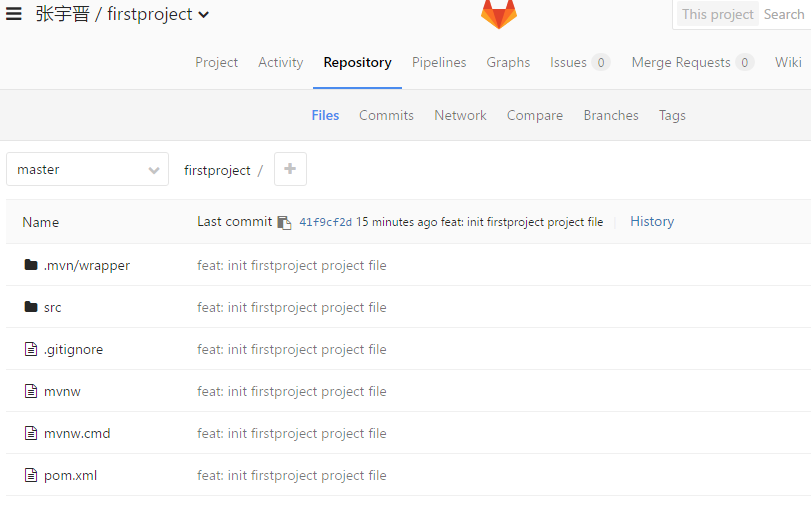
### 5.$ git push origin //push本地仓库文件到远端gitlab





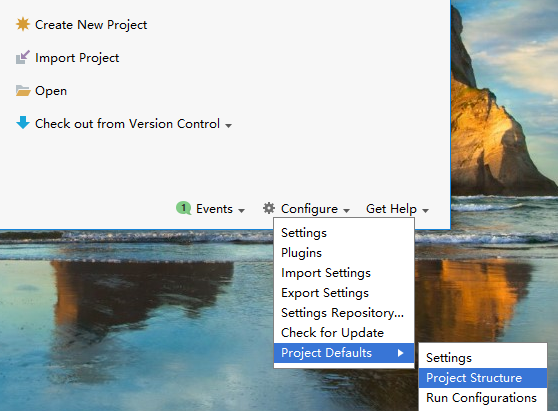
## 5. 检测成功

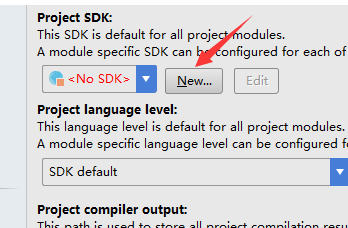


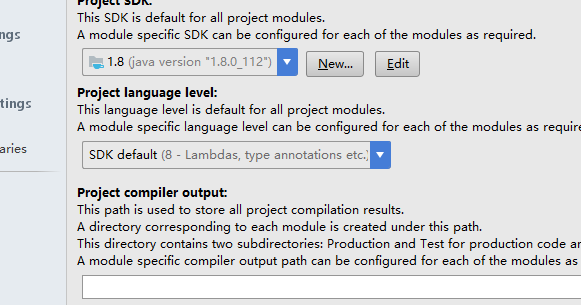


# 5. 导入IDE(IntelliJ IDEA)

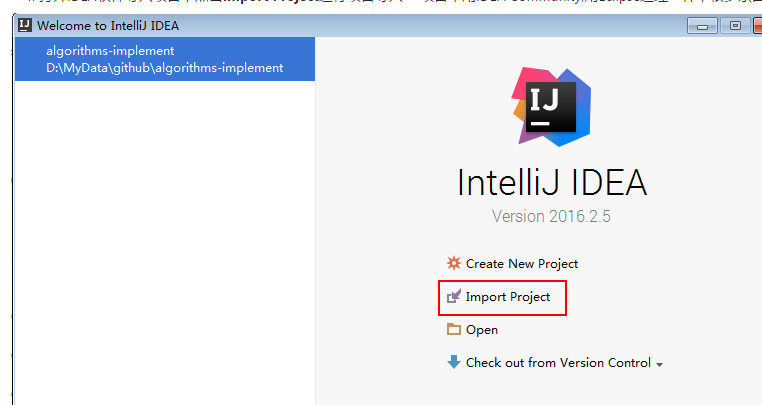
## 1.设置jdk JAVA\_HOME

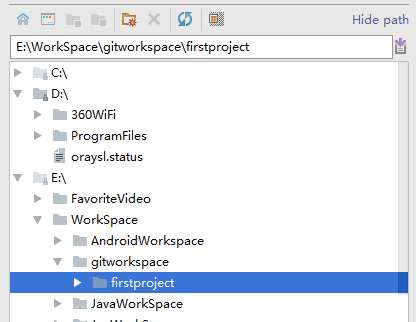


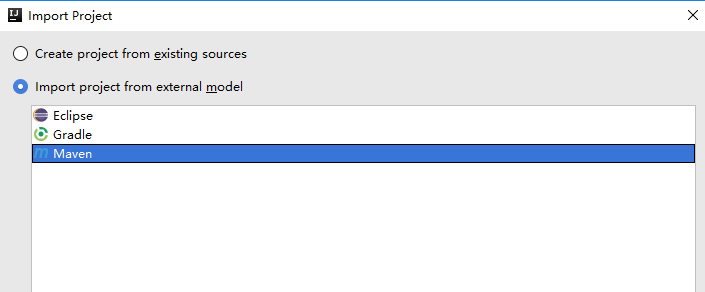


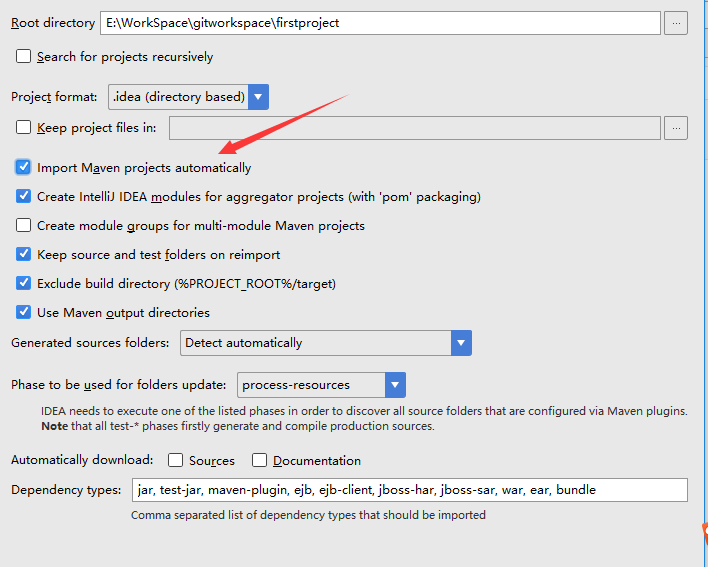


## 2. 打开IDEA软件导入项目，点击Import Project进行项目导入 > 项目中用IDEA Community,用Eclipse道理一样，很多东西都是共通的。





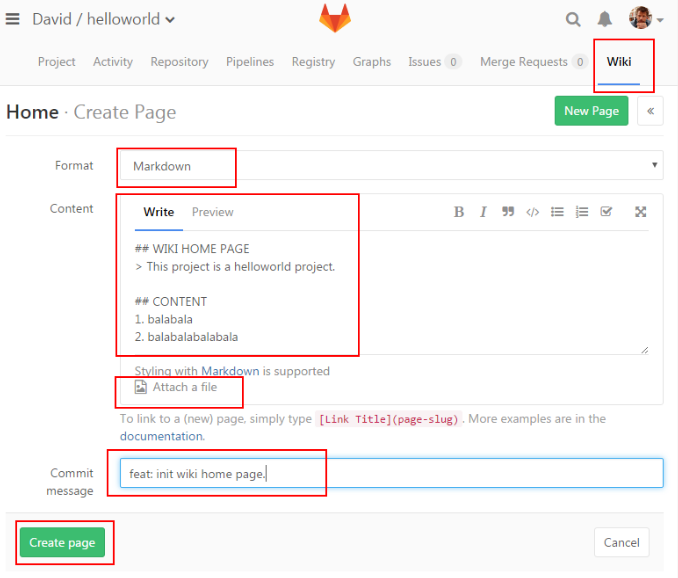




# 6. Gitlab中Wiki文档同项目空间管理

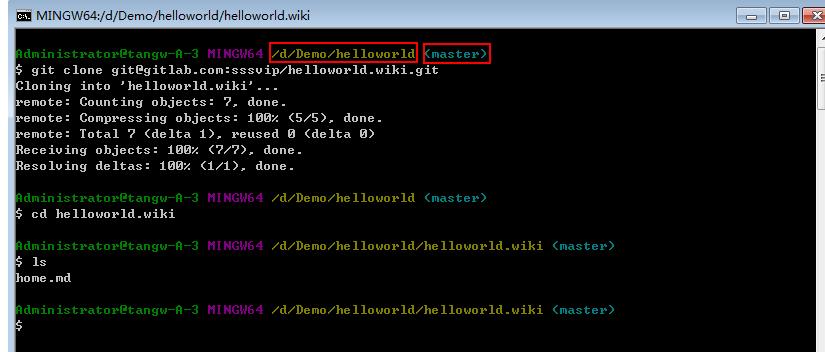
## 1.创建一个wiki

我们一般选择Markdown格式书写，也可个根据自己情况选择格式，这个会根据你选择的格式生成不同格式文件后缀文件。当然你可以看见可以添加附件的，Gitlab Wiki可以上传附件，其会根据文件hash产生唯一地址，不会担心重复上传什么的。做完初始填写后点击Create page,填错也不怕，后面可以修改。



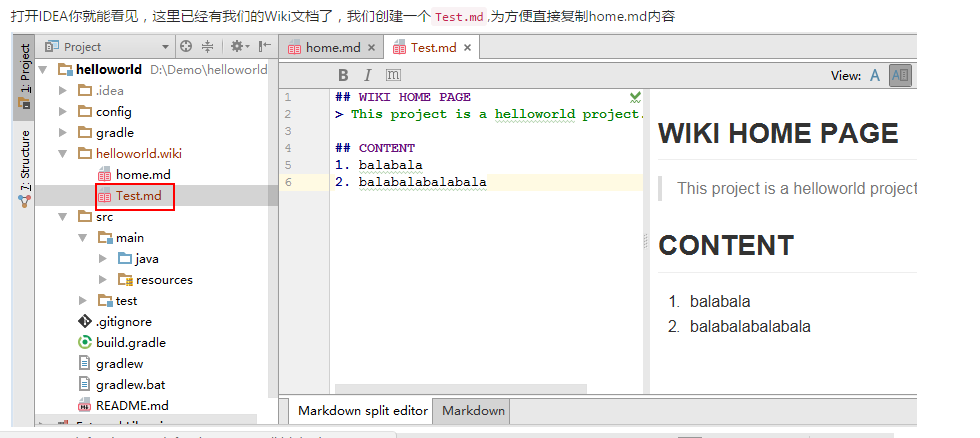
## 2. Wiki克隆至本地

### 1.Wiki页有右边"<<"按钮，点击后您可以看到Clone repository点击后你可以看到上面的页面，提示已经教会你如何进行管理它，clone后进行本地管理就行，我们这里要做的是同目录下用IDEA去管理，去编辑Wiki。（注意git的地址，下面应该写为我的项目中的git wiki的地址）



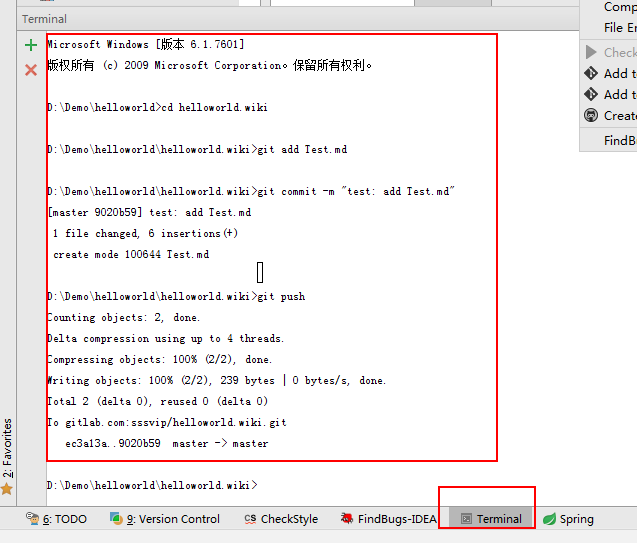
## 3. 添加Wiki并push至Gitlab的Wiki

打开IDEA你就能看见，这里已经有我们的Wiki文档了，我们创建一个Test.md,为方便直接复制home.md内容



## 4.下面写入上传的命令，这样就不需要黑框也可以了

Terminal中命令不再解释，为简单的版本管理流程，这里截图想说明的是，我们不但可以用IDEA提供GUI方式去版本控制，也可以用IDEA下方的Terminal进行用git命令方式进行版本控制，不用我们再开cmd窗口，这也是“一站式管理”的又一体现。



**d> cd firstproject.wiki**

**d> git add Test.md**

**d> git commit -m "test:add Test.md"**

**d> git push**

## 5.检测上传成功

