Java01 进阶任务：

什么是.git 目录？

.git 目录是 Git 版本控制系统的核心目录，包含了 Git 的所有配置信息，版本历史记录，分支信息等。

.git 目录包含了什么？

1. config 文件，包含 Git 项目的配置信息；
2. HEAD 文件，指向当前工作分支的最新提交；
3. hooks 目录，可以在 Git 操作中被调用的脚本文件，如 pre-commit 钩子；
4. objects 目录，存储 Git 对象，包括提交，树等；
5. refs 目录，包含分支和标签的引用信息；
6. index 文件，保存暂存区的数据；

   相应的 Git 命令：

   1. 添加：git add；

      将指定的文件添加到暂存区；
   2. 提交，git commit；

      可以将暂存区的文件，添加到本地仓库；
   3. 回滚：

      首先用 git log 查看历史记录；

      用 git revert +版本号，可以回到指定的提交；

      用 git revert +HEAD 可以撤销最新的提交并且创建一个新的提交；

      用 git revert - -hard +版本号，可以强制回滚到较早的版本；

      用 git cherry-pick 可以将提交恢复到分支上；
   4. 签出：

      第一类：可以用来切换分支；git checkout branch；

      第二类：回退到历史版本；git checkout commit-ID；
   5. 删除：

      第一类：删除本地分支；用 git branch -d branch-name；（-D）表示强制删除；（应对有未合并的情况）

      第二类：删除远程分支：git push origin -- delete branch-name；
   6. 合并（git merge）

      第一步，切换到目标分支，点击顶部的“Branches“选项卡，然后选择源分支，并且请求合并，按“Merge into Current”选项。最后将合并结果上传到本地仓库。上传到远程仓库时候要先解决分支。可以保存完整的提交历史。
   7. 变基（git rebase）

      可以将一系列的提交移动到另一系列提交上，还是在“Branches”选择想要变基的分支；

方式：git rebase [options] <切换分支> <当前分支>
--continue 继续未完成的变基操作;
--abort 结束变基操作;

8. 克隆操作（clone）

在 git 弹出的窗口中粘贴远程仓库地址实现克隆操作；如 Https：/***;

9. 提取：
   使用 git fetch 操作从远程仓库提取到最新的提交和分支信息，但不会将这些分支合并到本地仓库中，适合在合并前查看仓库的状态。在选择要更新的远程仓库地址再点击“Fetch”。
10. 更新：
    在 IDEA 的 git 工具栏选择“Update project“，在选择了项目和分支后，IDEA 会自动执行更新操作;
11. 将传入更改合并：
    与合并区别就在对于提交历史和分支的处理方式不同。
12. 在传入更改上变基当前分支：（git pull –rebase）
    在拉取远程远程分支的最新更改的同时，应用到当前分支的顶部，保持一个干净，线性的提交历史。
13. 推送：git push：
    通过 git push 操作可以将本地仓库中的代码推送到远程仓库，还可以推送到多个远程仓库，（前提是提前关联了远程仓库 ->git remote add origin Https：//---

Fork 和 Clone 有什么区别?

Fork 是在代码托管平台上进行的，将一个仓库复制过来的同时生成一个新的仓库，可以通过 Pull Request 向原始仓库提交合并需求，不需要写权限，只需要读权限

Clone 是在本地计算机上进行的，将远程仓库的内容复制到本地计算机上，无需连接远程仓库，就可以在本地进行开发和修改，通常需要写权限，以便将本地的修改推送到远程仓库，常用于本地开发和测试。

Pull Request 和 Push 有什么区别?

Pull Request 是在 GitHub 等托管平台上，常常是从 fork 中拉取代码，是开发者将他们的更改提交给另一个仓库的所有者或有合并权限的人审查。相当于请求合并，无需权限。

Push 是将本地仓库提交到远程仓库，是单向的，但推送通常需要权限。

Git 管理的四个区及其功能：

工作区（Workspace）：进行代码编写和修改的地方，但这种更改并未被 Git 跟踪或者记录

暂存区（Index）：是一个临时存储区域，用于存放已经修改并且准备提交到版本库中的文件，通过 git add 可以将工作区的修改添加到暂存区。

本地仓库（Local Repository）：是存储在本地计算机上的版本库，通过 git commit 命令可以将暂存区的内容提交到本地仓库，并且生成一次新的提交记录，本地仓库记录了项目的所有历史版本和提交记录。

远程仓库（Remote Repository）：位于代码托管平台上的仓库，用于协作和备份，多个开发者可以共享一个远程仓库，都可以将自己的改动推送到远程仓库，实现协同操作。

移动过程：在工作区进行开发，改动，通过 git add 将工作区的文件添加到暂存区。

通过 git commit 将暂存区提交到本地仓库。

通过 git push 将本地仓库内容推送到远程仓库。

Git 冲突是什么？以及产生原因：

多人协作开发过程中，两个或者多个分支修改了同一个文件，Git 无法自动合并这些更改，就会产生冲突。

原因可能有：

1. 多人同时修改同一文件；
2. 多个分支的修改合并到同一分支；
3. 文件重命名或移动；
4. 没有及时同步代码，本地和远程代码不同步；
5. 同时修改同一行代码；

解决措施：

使用 git status 查看冲突文件，用 git add 添加到暂存区，用 git commit 修改提交。

手动解决冲突：识别冲突标记，根据实际需要保留需要的代码，删除冲突标记，使用文本编辑器保存文件；

如果在某个分支进行修改，但并没有保存，此时切换分支，工作区的文件会发生什

么?

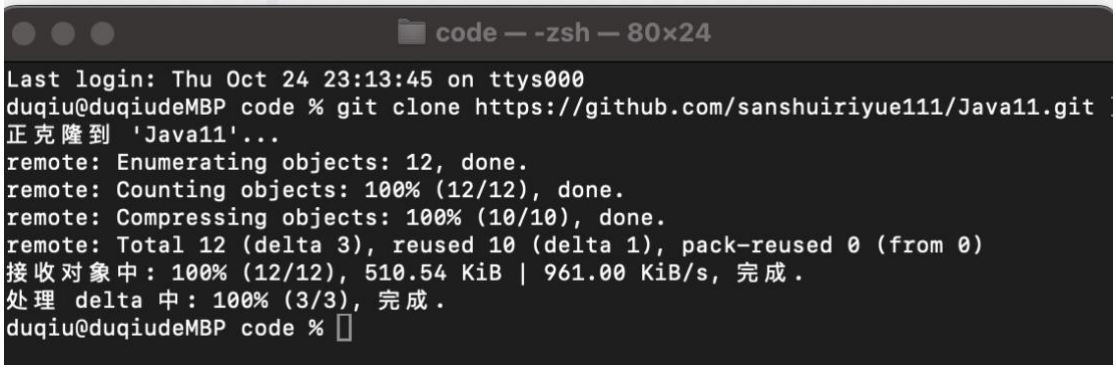
会丢失或者冲突。

因为 Git 在切换分支会重置工作区的内容，未提交的修改不会被保存;

如果新，旧分支有相同的文件并且内容不同，切换后可能会出现文件冲突;

GitHub Flow 工作流实践:

安装配置 git 已经完成，克隆了一个远程仓库到本地仓库，如下所示:



然后相应地进行 git add，git commit ，git push -u origin,

最后推送:

```
duqiu@duqiudeMBP code % git push -u origin develop
枚举对象中：3，完成.
对象计数中：100% (3/3)，完成.
使用 11 个线程进行压缩
压缩对象中：100% (2/2)，完成.
写入对象中：100% (3/3)，346 字节 | 346.00 KiB/s，完成.
总共 3（差异 0），复用 0（差异 0），包复用 0（来自 0 个包）
To https://github.com/sanshuiriyue111/Java1pro.git
 * [new branch]      develop -> develop
分支 'develop' 设置为跟踪 'origin/develop'。
duqiu@duqiudeMBP code %
```

下一步，切换到 feature 分支的操作以及最后推送到远程仓库，再删除分支:

```
duqiu@duqiudeMBP code % git checkout -b feature1
切换到一个新分支 'feature1'
duqiu@duqiudeMBP code % git add abc.txt
duqiu@duqiudeMBP code % git commit -m "Hello"
[feature1 e2e4276] Hello
 1 file changed, 1 insertion(+)
 create mode 100644 abc.txt
duqiu@duqiudeMBP code % git merge develop
已经是最新的。
duqiu@duqiudeMBP code % git push -u origin feature1
已删除分支 feature（曾为 8c9e50d）。
duqiu@duqiudeMBP code %
```

Bug 修复;

如下所示:

```
[duqiu@duqiudeMBP code % git checkout -b hotfix/bug
切换到一个新分支 'hotfix/bug'
[duqiu@duqiudeMBP code % git add abc.txt
[duqiu@duqiudeMBP code % git commit -m "bug解决了"
[hotfix/bug b3725ca] bug解决了
 1 file changed, 2 insertions(+), 1 deletion(-)
```

```
[duqiu@duqiudeMBP code % git checkout master
切换到分支 'master'
[duqiu@duqiudeMBP code % git merge hotfix/bug
更新 e2e4276..b3725ca
Fast-forward
 abc.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
duqiu@duqiudeMBP code %
```

```
[duqiu@duqiudeMBP code % git branch -d hotfix/bug
已删除分支 hotfix/bug (曾为 b3725ca) 。
duqiu@duqiudeMBP code %
```

什么叫流程引擎执行打包部署:

GitHub Actions:

可以通过 GitHub Actions 来实现自动化构建，打包和部署的过程;

分 Workflow（工作流程），Job（任务），Step（步骤，多个步骤组成一个 Job），Action（每个 Step 可以执行一个或者多个命令）

接下来模拟多人操作:

首先克隆一个远程仓库（已经克隆完毕）

然后进行相关操作:

```
Last login: Fri Oct 25 12:24:15 on ttys000
[duqiu@duqiudeMBP code % cd Java11
[duqiu@duqiudeMBP Java11 % git checkout -b develop
切换到一个新分支 'develop'
[duqiu@duqiudeMBP Java11 % git add 1.java
[duqiu@duqiudeMBP Java11 % git commit -m "develop"
位于分支 develop
无文件要提交，干净的工作区
duqiu@duqiudeMBP Java11 %
```

为什么在远程仓库进行修改后，本地仓库修改再推送 git 会拒绝推送?

因为远程仓库的 develop 分支有更新，而本地仓库没有同步这些更新就会产生冲突，从而 git 拒绝推送，需要先将远程仓库的更新拉取到本地。

在多人开发同一个项目时，应进行什么 git 操作?

Step1.初始化本地仓库，通过 git init 来实现;

Step2.关联远程仓库：git remote add origin <URL>

Step3.多人合作是在自己的分支上进行：这就需要创建一个新的分支：git checkout -b 分支名;

Step4:将自己的分支推送到远程仓库：通过 git push -u origin 分支名;

Step5:在远程仓库上建立 Pull Request 请求将该分支合并到主分支，需要人来审核代码是否通过再合并;

Step6.实时检查更新本地仓库，通过 git pull origin 主分支名来实现。