哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

哈尔滨工业威海校区2024年秋季学期

开源软件开发实践

徐汉川
xhc@hit.edu.cn

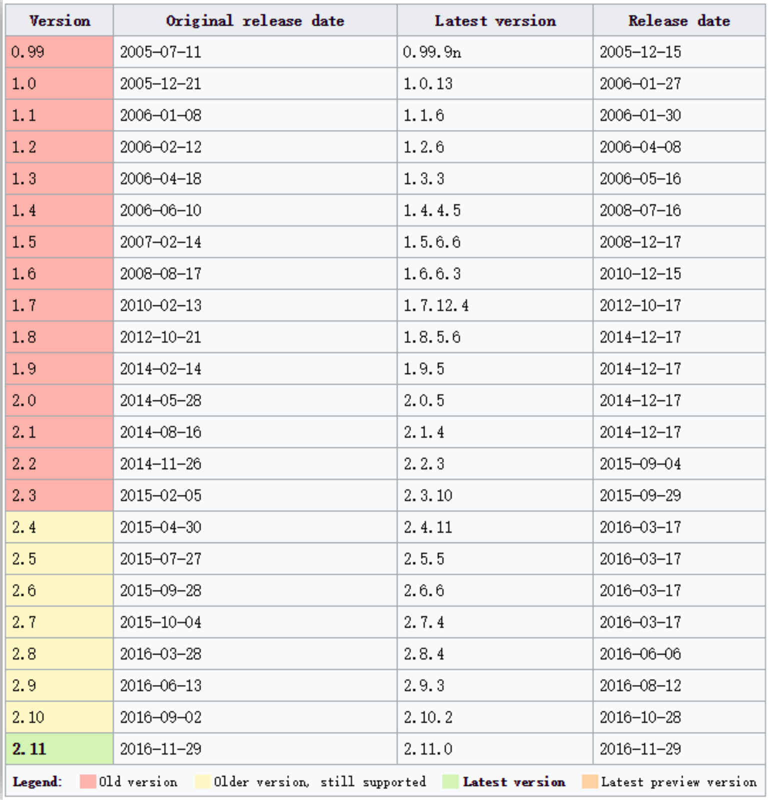
September 27, 2024

Git 回顾

What is Git?

- Initial release: 2005
- Initial Author: Linus Torvalds
- For development of the Linux kernel.

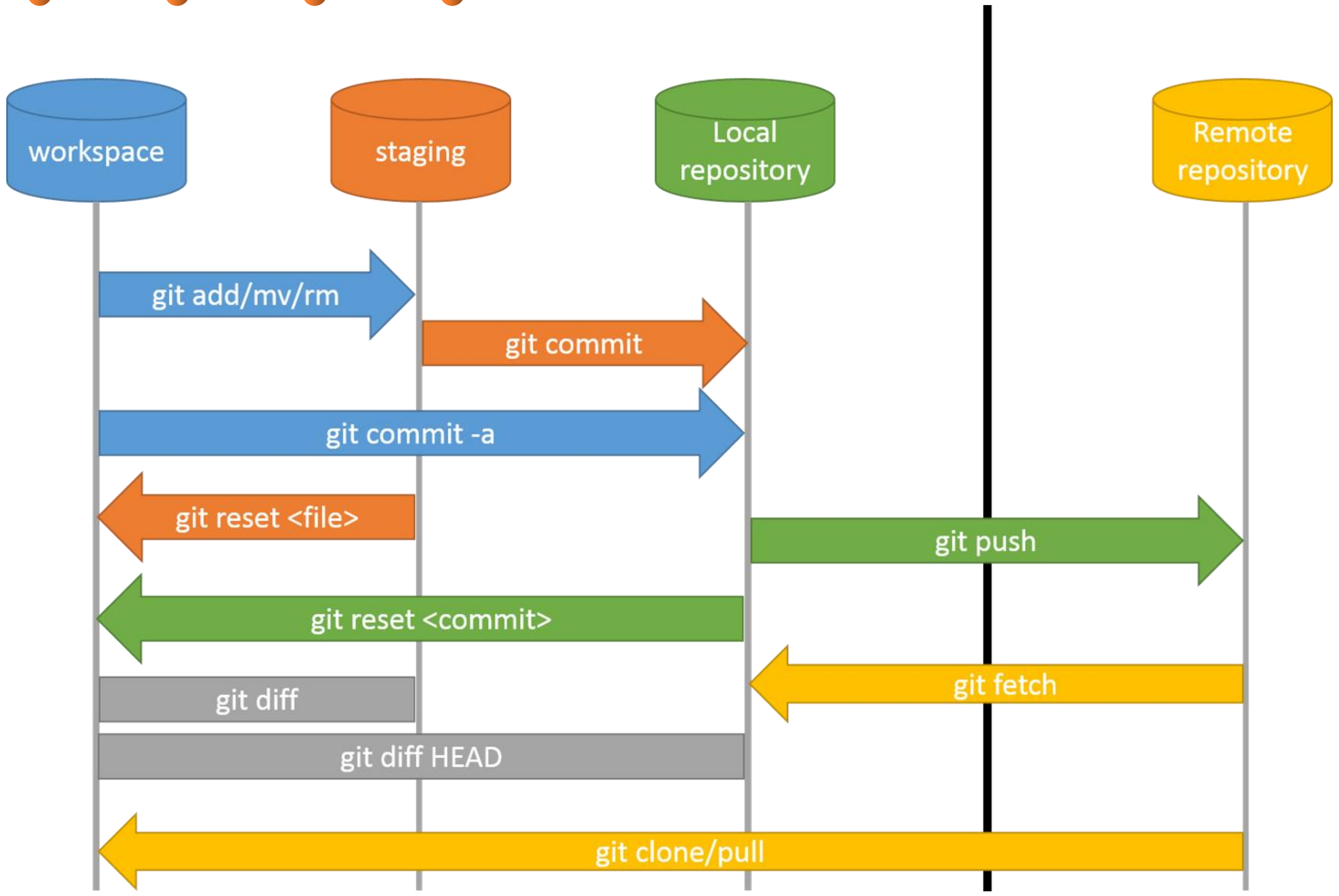
Linus Torvalds (1969-)



Version	Original release date	Latest version	Release date
0.99	2005-07-11	0.99.9n	2005-12-15
1.0	2005-12-21	1.0.13	2006-01-27
1.1	2006-01-08	1.1.6	2006-01-30
1.2	2006-02-12	1.2.6	2006-04-08
1.3	2006-04-18	1.3.3	2006-05-16
1.4	2006-06-10	1.4.4.5	2008-07-16
1.5	2007-02-14	1.5.6.6	2008-12-17
1.6	2008-08-17	1.6.6.3	2010-12-15
1.7	2010-02-13	1.7.12.4	2012-10-17
1.8	2012-10-21	1.8.5.6	2014-12-17
1.9	2014-02-14	1.9.5	2014-12-17
2.0	2014-05-28	2.0.5	2014-12-17
2.1	2014-08-16	2.1.4	2014-12-17
2.2	2014-11-26	2.2.3	2015-09-04
2.3	2015-02-05	2.3.10	2015-09-29
2.4	2015-04-30	2.4.11	2016-03-17
2.5	2015-07-27	2.5.5	2016-03-17
2.6	2015-09-28	2.6.6	2016-03-17
2.7	2015-10-04	2.7.4	2016-03-17
2.8	2016-03-28	2.8.4	2016-06-06
2.9	2016-06-13	2.9.3	2016-08-12
2.10	2016-09-02	2.10.2	2016-10-28
2.11	2016-11-29	2.11.0	2016-11-29

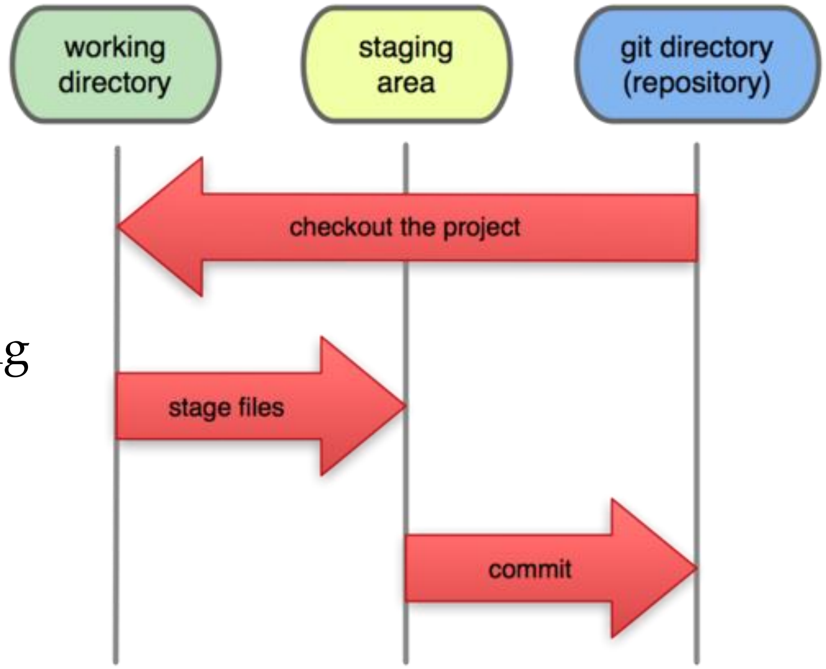
Legend: 🟥 Old version 🟨 Older version, still supported 🟩 **Latest version** 🟧 Latest preview version

Managing changes in software evolution process



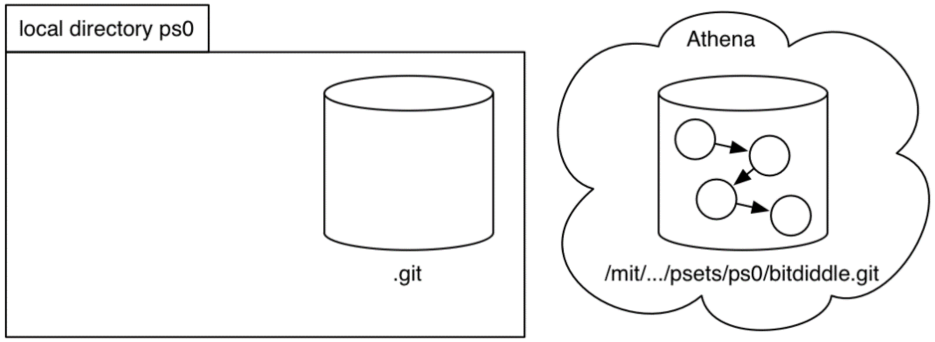
Git repository

- **A Git repository has three parts:**
  - .git directory (a repository storing all version control data) 本地的CMDB
  - Working directory (local file system) 工作目录：本地文件系统
  - Staging area (in memory) 暂存区：隔离工作目录和Git仓库
- **Each file belongs to one of the following three states:**
  - Modified (the file in working directory is different from the one in git repository, but is not in staging area) 已修改
  - Staged (the file is modified and has been added into the staging area) 已暂存
  - Committed (the file keeps same in working directory and git directory) 已提交



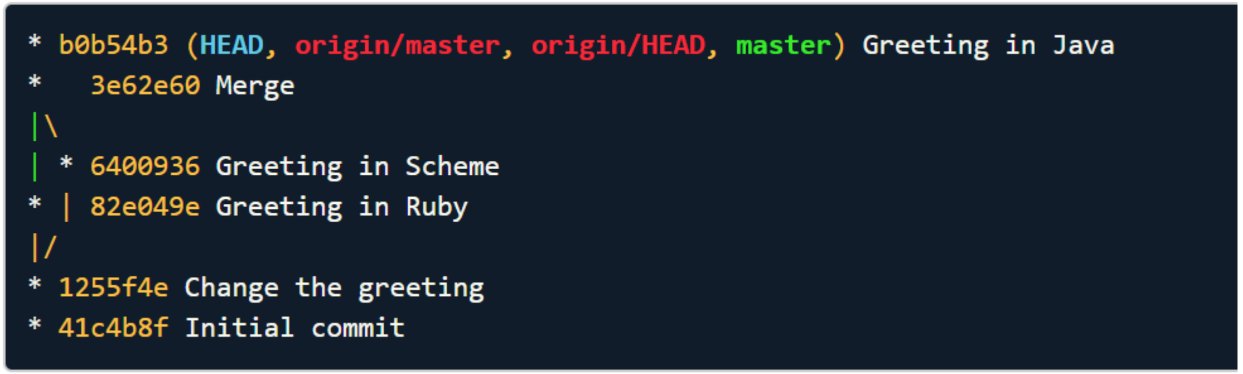
Object graph in Git

- All of the operations we do with Git — **clone, add, commit, push, log, merge, …** — are operations on a graph data structure that stores all of the versions of files in the project, and all the log entries describing those changes. Git的所有操作都是在一个图数据结构(对象图)上进行
- The **Git object graph** is stored in the .git directory of the repository.
- Copying a git project from another machine/server means copying the whole object graph. 从另一台机器/服务器复制git项目意味着复制整个对象图
  - `git clone URL local_repository`

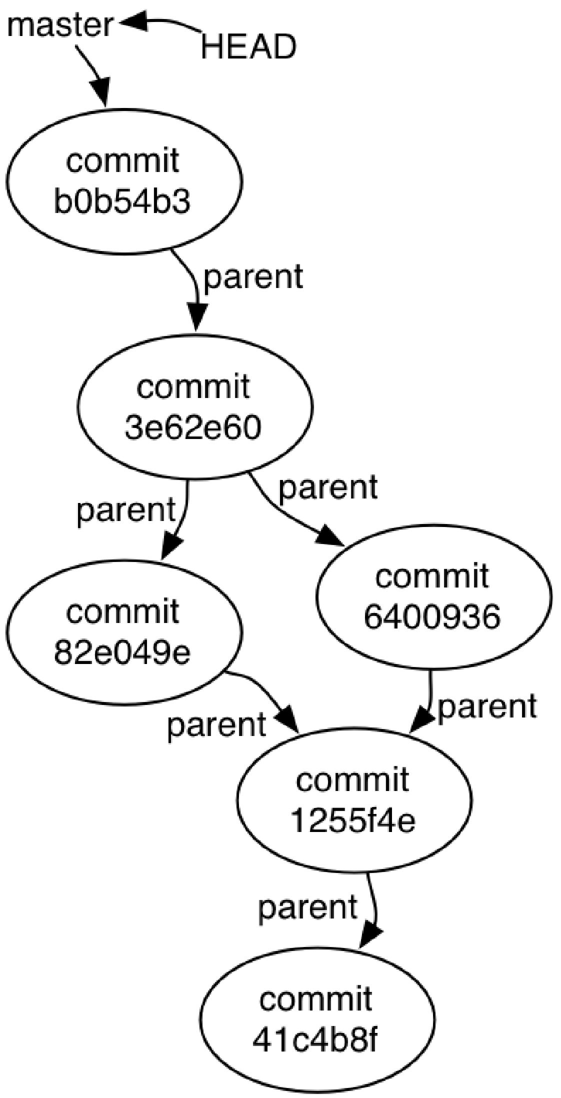


What an Object Graph looks like?

- **Object graph, being the history of a Git project, is a directed acyclic graph (DAG).**

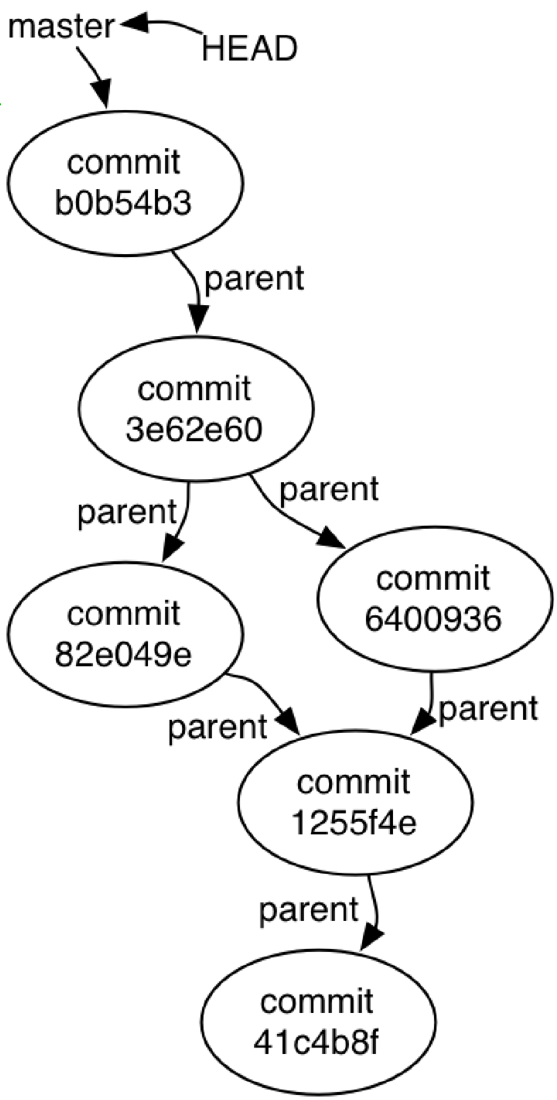


```
* b0b54b3 (HEAD, origin/master, origin/HEAD, master) Greeting in Java
*   3e62e60 Merge
|\
| * 6400936 Greeting in Scheme
* | 82e049e Greeting in Ruby
|/
* 1255f4e Change the greeting
* 41c4b8f Initial commit
```



- **Object Graph：版本之间的演化关系图，一条边A→B表征了“在版本B的基础上作出变化，形成了版本A”**

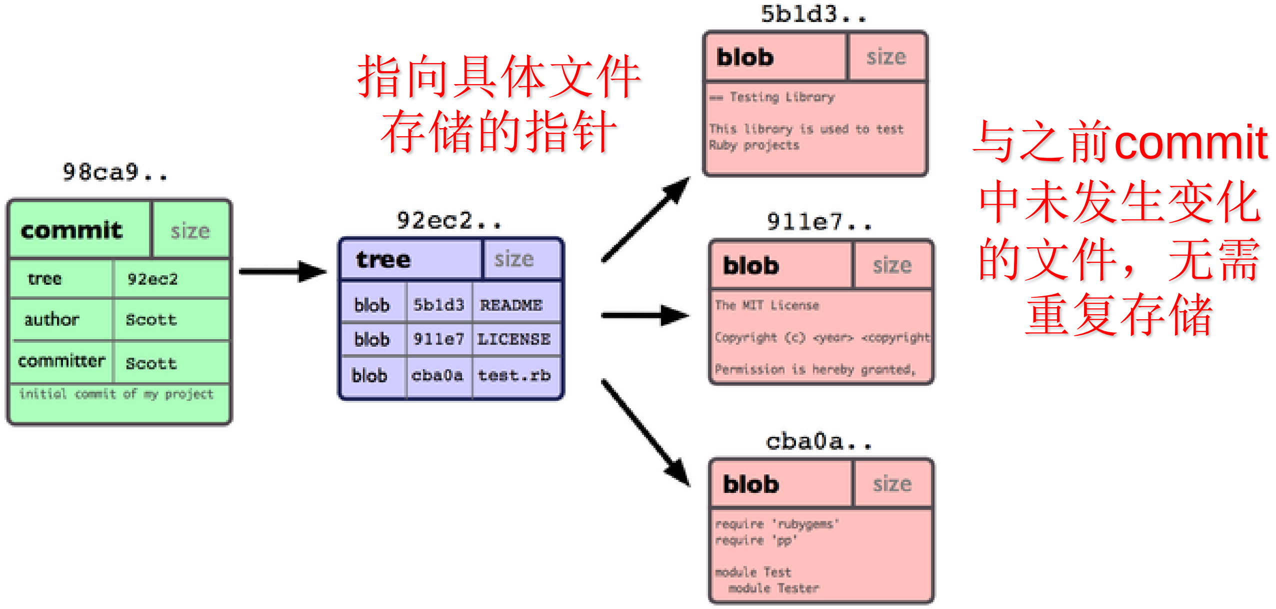
Commits: nodes in Object Graph



- **Each node in the history graph is a commit a.k.a. version a.k.a. revision of the project: a complete snapshot of all the files at that point in time.**
  - Except for the initial commit, each commit has a pointer to a parent commit. 每个commit指向一个父亲
  - Some commits have the same parent: they are versions that diverged from a common previous version. 多个commit指向同一个父亲：分支
  - Some commits have two parents: they are versions that tie divergent histories back together. 一个commit指向两个父亲：合并
- **A `branch` is just a name that points to a commit.**
- **HEAD points to the current commit.**
  - We need to remember which branch we're working on. So HEAD points to the current branch, which points to the current commit.

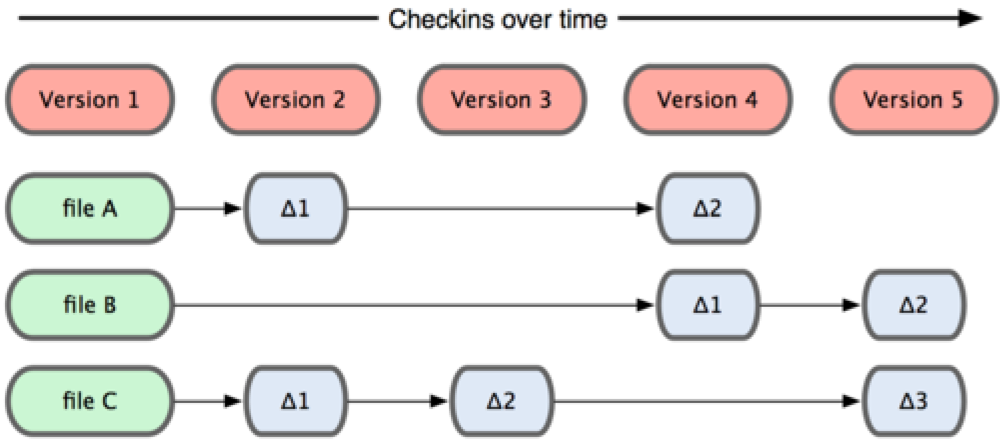
Commits: nodes in Object Graph

- Git represents a commit with a **tree** node.
  - For a project of any reasonable size, most of the files *won't* change in any given revision. Storing redundant copies of the files would be wasteful, so Git doesn't do that.
  - Instead, Git object graph stores each version of an individual file *once*, and allows multiple commits to *share* that one copy.
  - Each commit also has log data — who, when, short log message, etc.



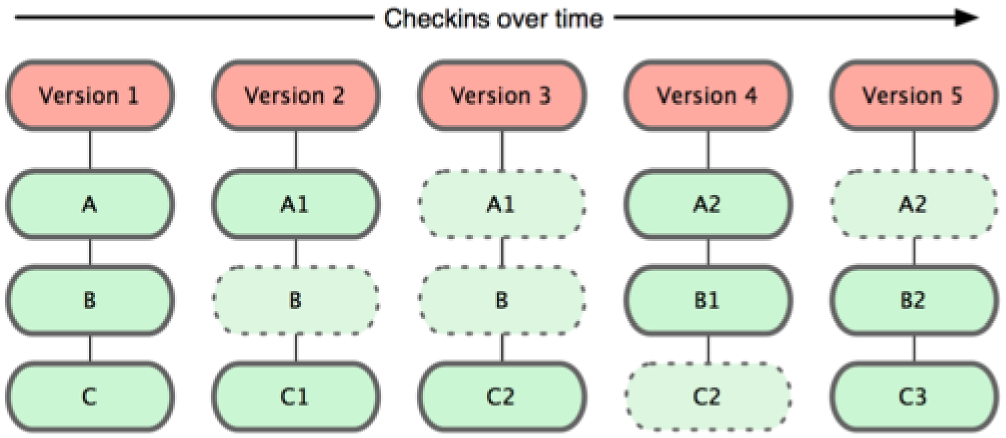
Managing changes in Git

- **Traditional VCS:**



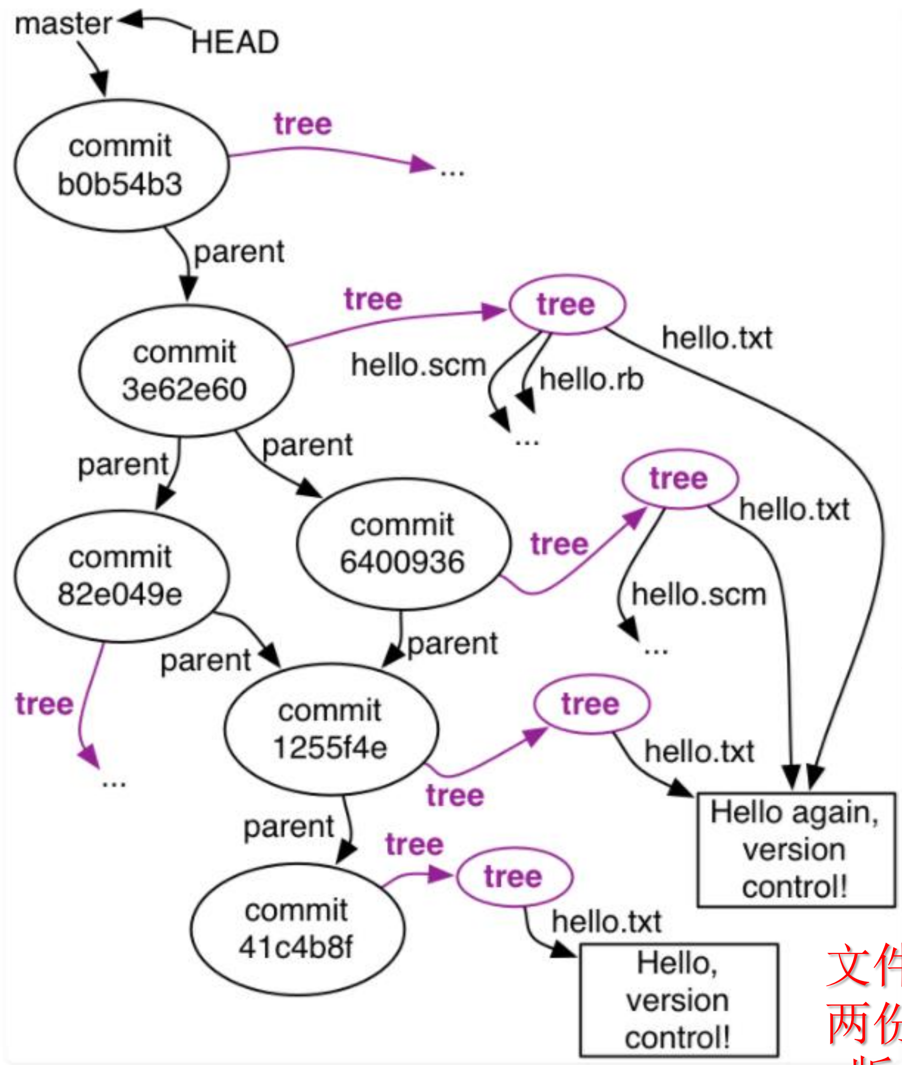
传统VCS存储版本之间的变化（行）

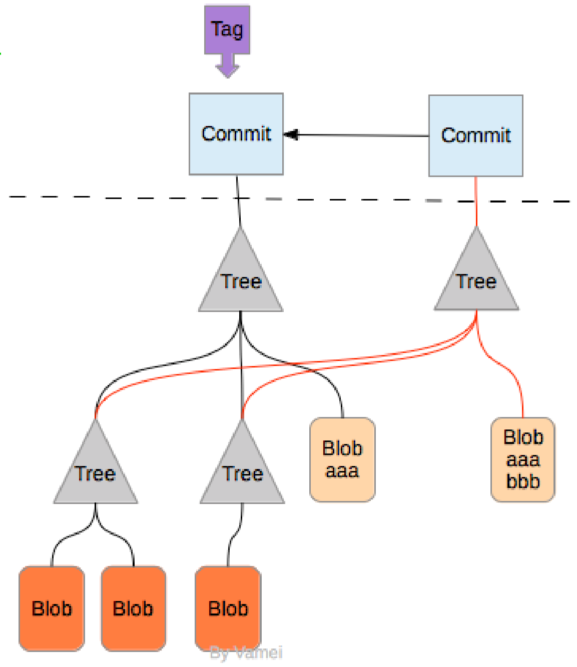
- **In Git:**



Git存储发生变化的文件（而非代码行），不变化的文件不重复存储

Commits: nodes in Object Graph

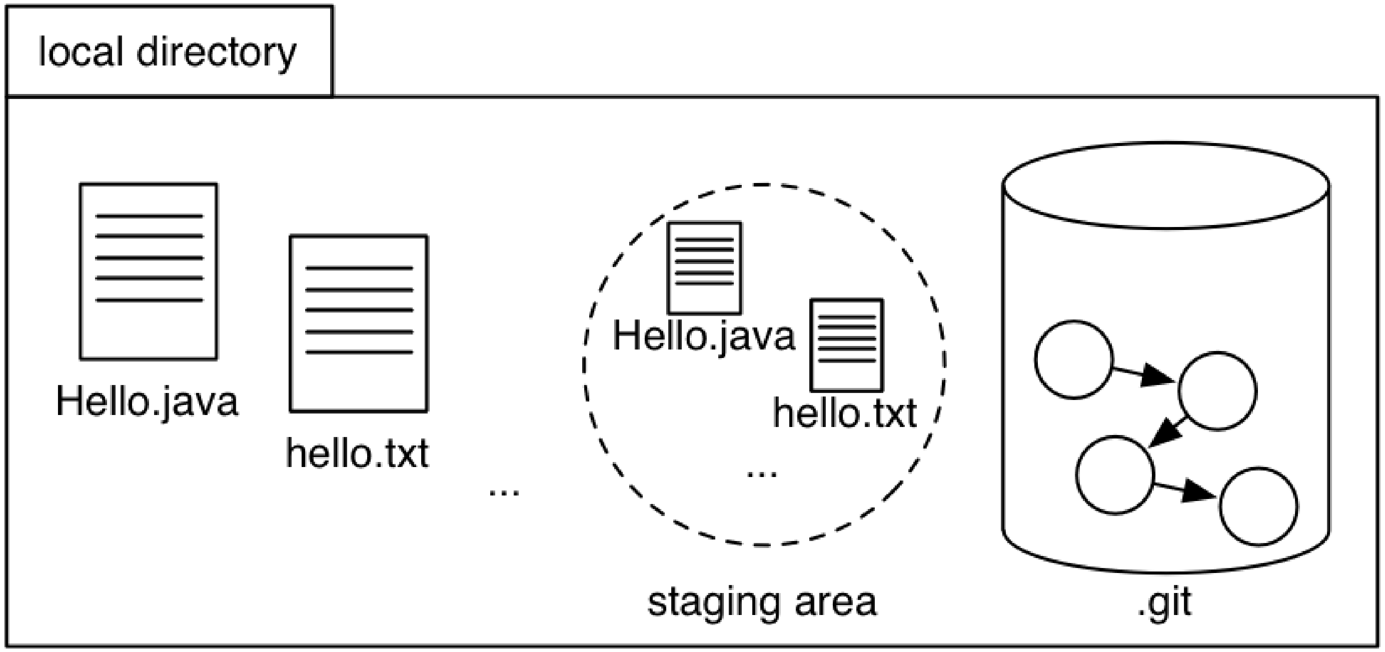




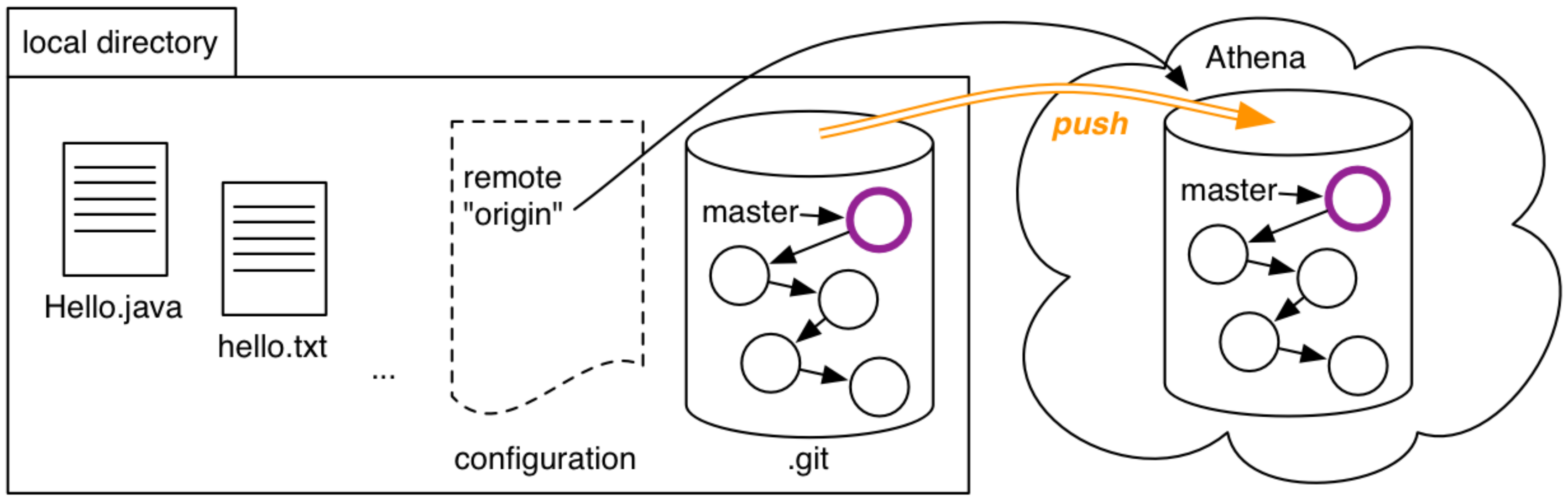
文件未发生变化，则后续多个版本始终指向同一个文件

文件发生变化了，存储两份不同的文件，两个版本指向不同的文件

Add to the object graph with `git commit`



Send & receive object graphs with git push & git pull



本地仓库

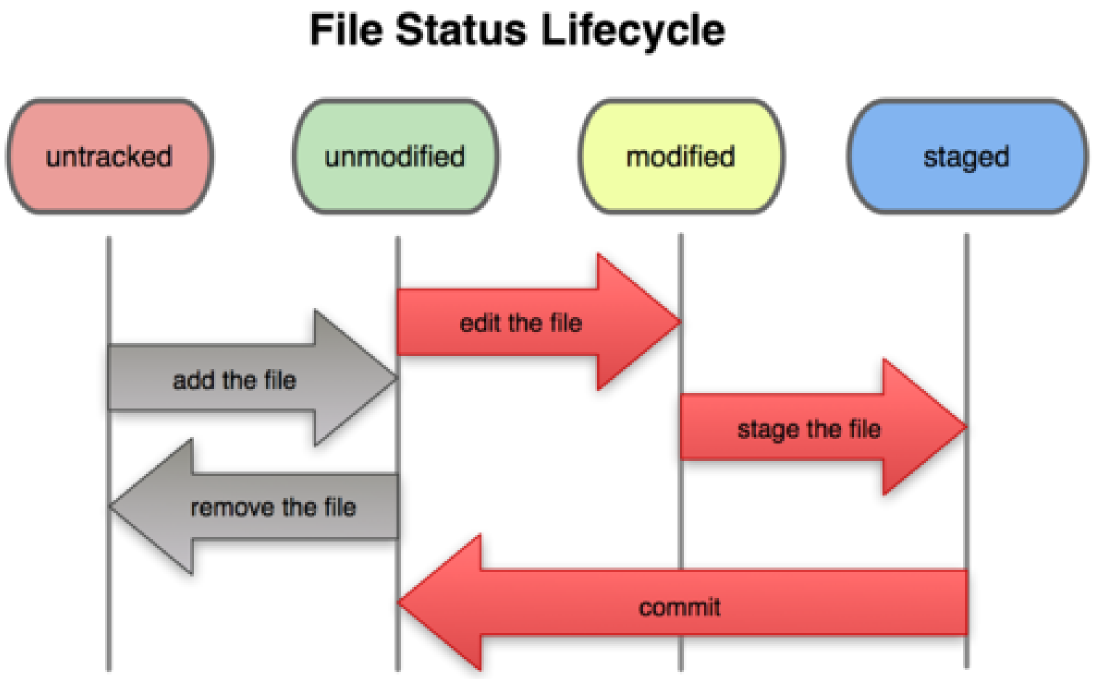
远程仓库

基本的Git命令：取得项目的 Git 仓库

- **下载并在本机安装设置Git环境** https://git-scm.com/
- **Git官方参考资料：** https://git-scm.com/book/zh/v2
- 在工作目录中初始化新仓库
  - 要对现有的某个项目开始用Git管理，只需到此项目所在的目录，执行**git init**命令，用 **git add** 命令告诉Git开始对这些文件进行跟踪，然后提交：
    - git init
    - git add *.c
    - git add readme.txt
    - git commit -m 'initial project version'
- 从现有仓库克隆：复制服务器上项目的所有历史信息到本地
  - git clone [url]

基本的Git命令：记录每次更新到仓库

- 在工作目录对某些文件作了修改之后，Git将这些文件标为“已修改”，可提交本次更新到仓库。
- 逐步把这些修改过的文件放到暂存区域，直到最后一次性提交所有这些暂存起来的文件，如此重复。



基本的Git命令：跟踪新文件、暂存已修改文件

- 使用git add开始跟踪一个新文件（使某个文件纳入到git中管理）。
- 一个修改过的且被跟踪的文件，处于暂存状态。

- git add后面可以指明要跟踪的文件或目录路径。如果是目录的话，就说明要递归跟踪该目录下的所有文件。
- git add的潜台词：把目标文件快照放入暂存区域，也就是 add file into staged area，同时之前未曾跟踪过的文件标记为需要跟踪。

- 若对已跟踪的文件进行了修改，使用git add命令将其放入暂存区；
- 运行了git add之后又对相应文件做了修改，要重新git add。

基本的Git命令：检查当前文件状态

- **要确定哪些文件当前处于什么状态，用git status命令：**
  - # On branch master nothing to commit (working directory clean)
    当前没有任何跟踪着的文件，也没有任何文件在上次提交后更改过
  - # On branch master # Untracked files: …
    有未跟踪的文件，使用git add开始跟踪一个新文件
  - # On branch master # Changes to be committed:
    有处于已暂存状态的文件

基本的Git命令：查看已暂存和未暂存的更新

- **git status回答：当前做的哪些更新还没有暂存？有哪些更新已经暂存起来准备好了下次提交？**
- **如果要查看具体修改了什么地方，可以用git diff命令，使用文件补丁的格式显示具体添加和删除的行。**
- **要查看尚未暂存的文件更新了哪些部分，不加参数直接输入git diff：**
  - 比较的是工作目录中当前文件和暂存区域快照之间的差异，也就是修改之后还没有暂存起来的变化内容。
- **若要查看已暂存起来的文件和上次提交时的快照之间的差异，可以用 git diff --cached 命令。**

基本的Git命令：提交更新

- **在使用git commit命令进行提交之前，要确认是否还有修改过的或新建的文件没有git add过，否则提交的时候不会记录这些还没暂存起来的变化。**
  - 每次准备提交前，先用git status进行检查，然后再运行提交命令git commit。
- **提交后返回结果：**
  - 当前是在哪个分支(master)提交的
  - 本次提交的完整 SHA-1 校验和是什么
  - 在本次提交中，有多少文件修订过、多少行添改和删改过。
- **提交时记录的是放在暂存区域的快照，任何还未暂存的仍然保持已修改状态，可以在下次提交时纳入版本管理。每一次运行提交操作，都是对项目做一次快照，以后可以回到这个状态，或者进行比较。**

基本的Git命令：跳过使用暂存区域、移除文件

- **Git提供了一个跳过使用暂存区域的方式，只要在提交的时候，给 git commit 加上-a 选项，Git 就会自动把所有已经跟踪过的文件暂存起来一并提交，从而跳过git add步骤；**

- **使用git rm命令从Git中移除某个文件，把它从已跟踪文件清单(暂存区域)中移除，并连带从工作目录中删除指定的文件。**

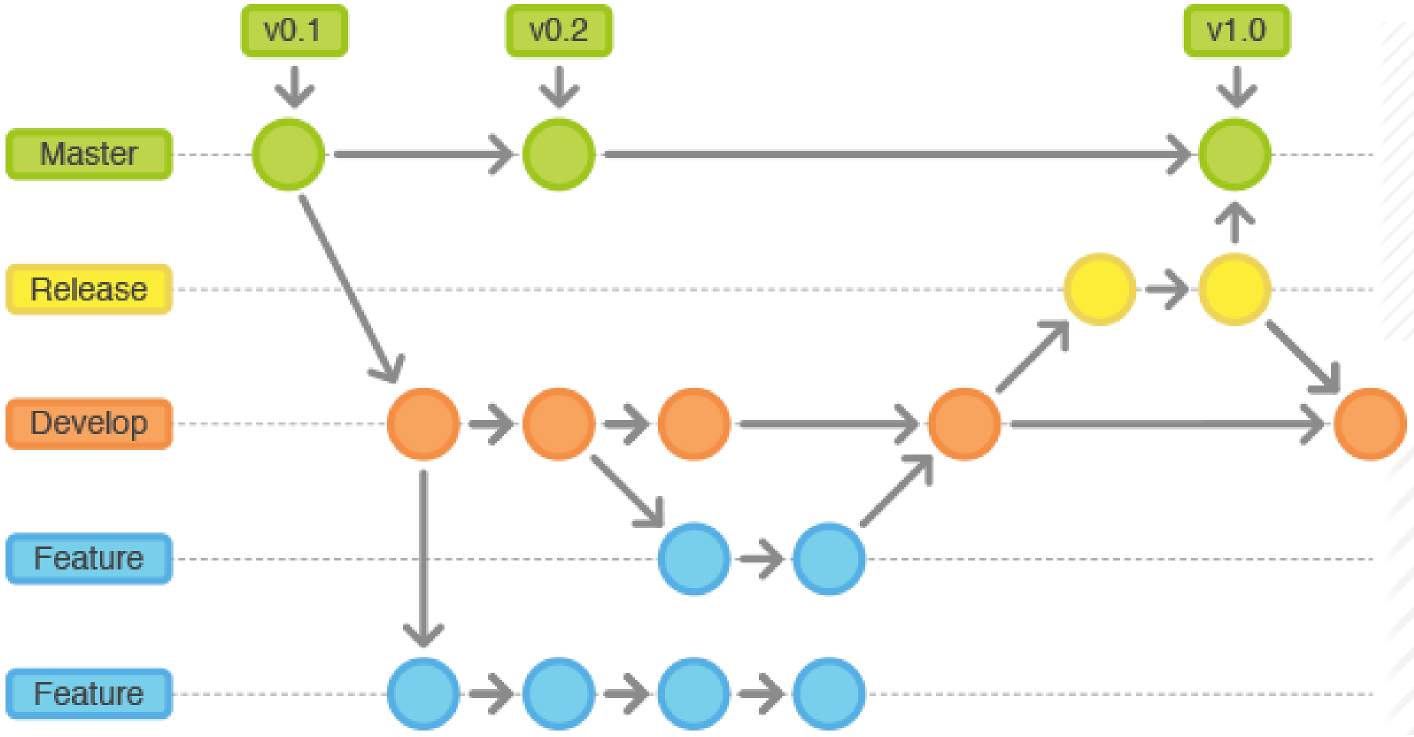
基本的Git命令：对远程仓库的操作

- **远程仓库：托管在网络上的项目仓库；**
- **多人协作开发某个项目时，需要管理这些远程仓库，以便推送或拉取数据，分享各自的工作进展。**
- **管理远程仓库：添加远程库、移除废弃的远程库、管理各式远程库分支、定义是否跟踪这些分支。**
  - git remote：获取当前配置的所有远程仓库；
  - git remote add [shortname] [url]：添加一个远程仓库；
  - git fetch：从远程仓库抓取数据到本地；
  - git pull： 从一个仓库或者本地的分支拉取并且整合代码；
  - git push [remote-name] [branch-name]：将本地仓库中的数据推送到远程仓库；
  - git remote show [remote-name]：查看某个远程仓库的详细信息；
  - git remote rm：从本地移除远程仓库；

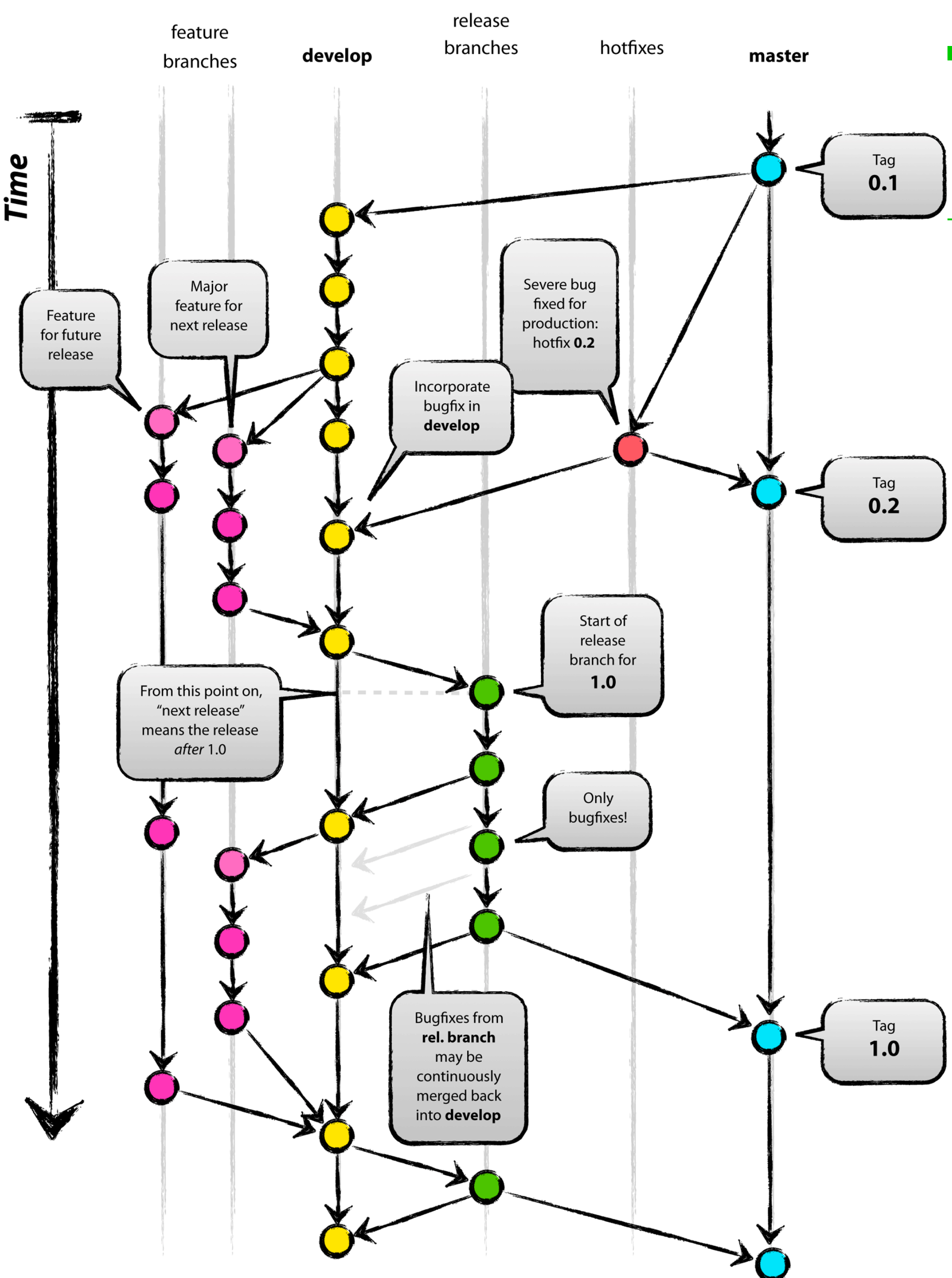
Git supports Branch and Merge 分支/合并

- A branch is the duplication of an object under revision control so that modifications can happen in parallel along both branches. 分支是在版本控制下对对象的复制，以便可以沿两个分支平行进行修改。

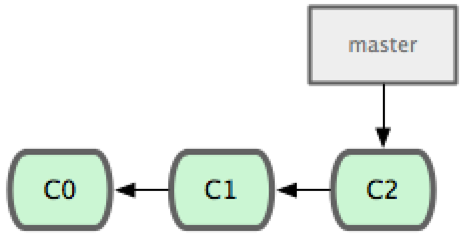
- Merging two branches together.

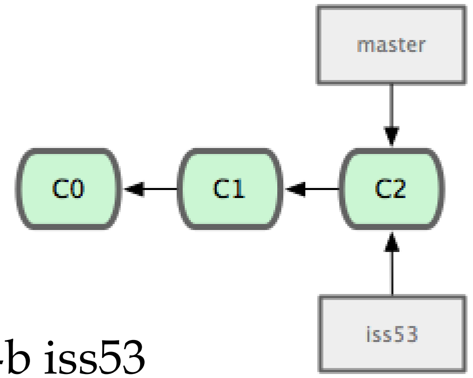


Branching

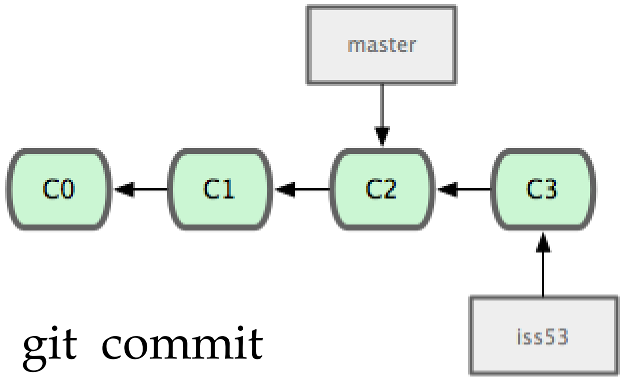

feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Major feature for next release
Feature for future release
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Creating and merging branches in Git

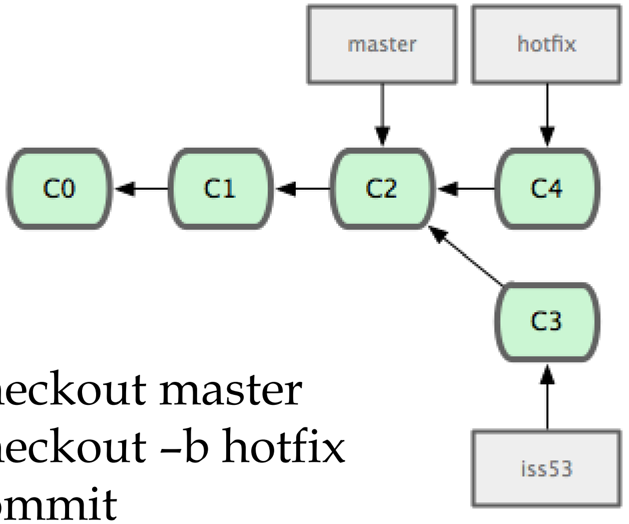




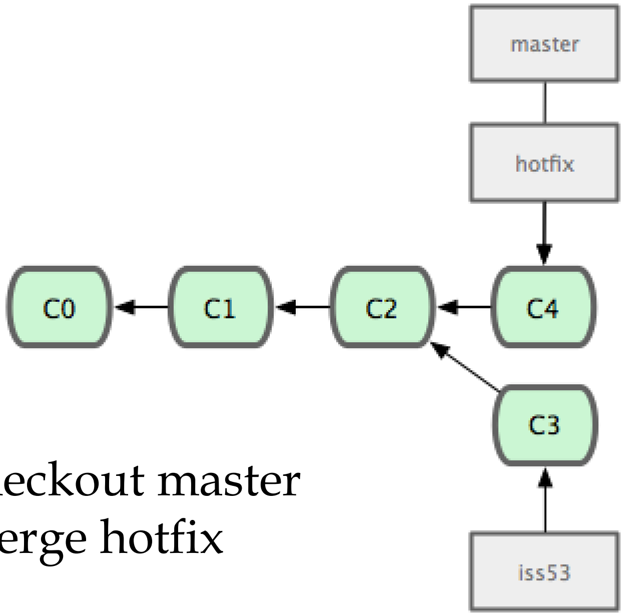
git checkout –b iss53



git commit

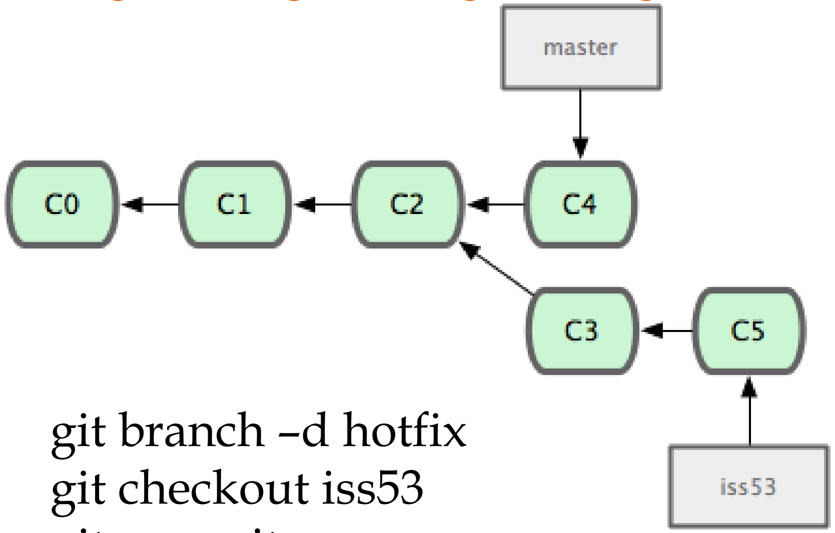


git checkout master
git checkout –b hotfix
git commit

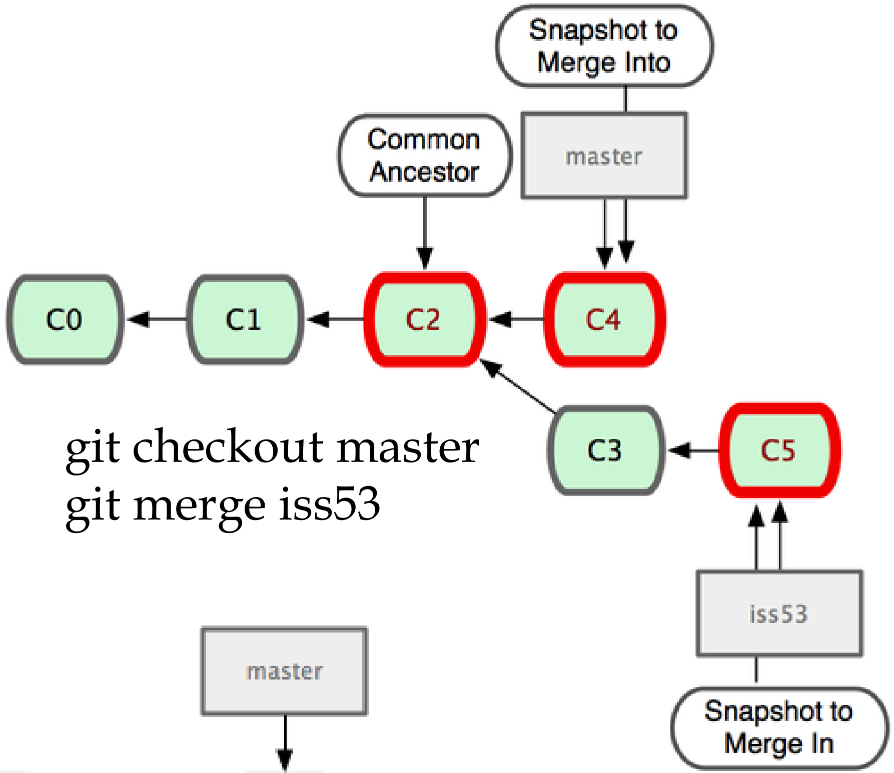


git checkout master
git merge hotfix

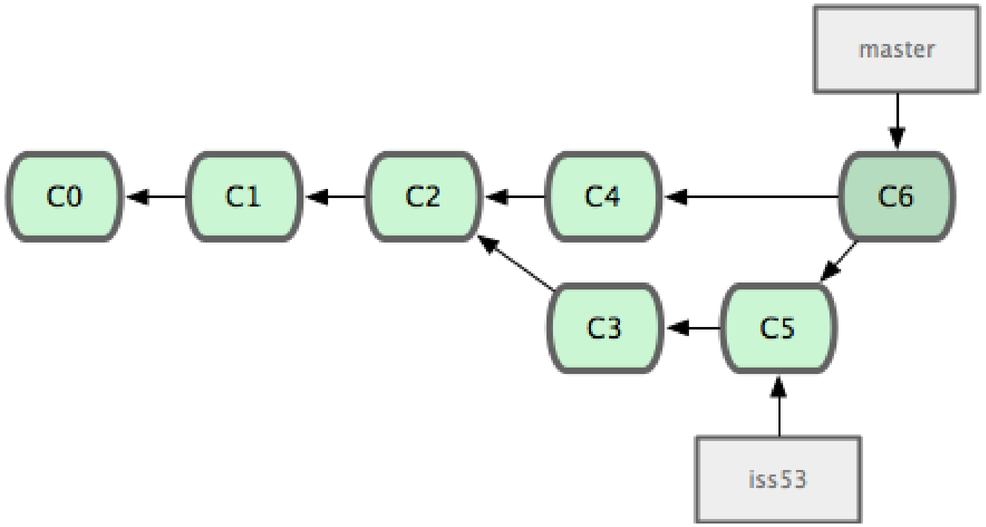
Creating and merging branches in Git



```
git branch –d hotfix
git checkout iss53
git commit
```

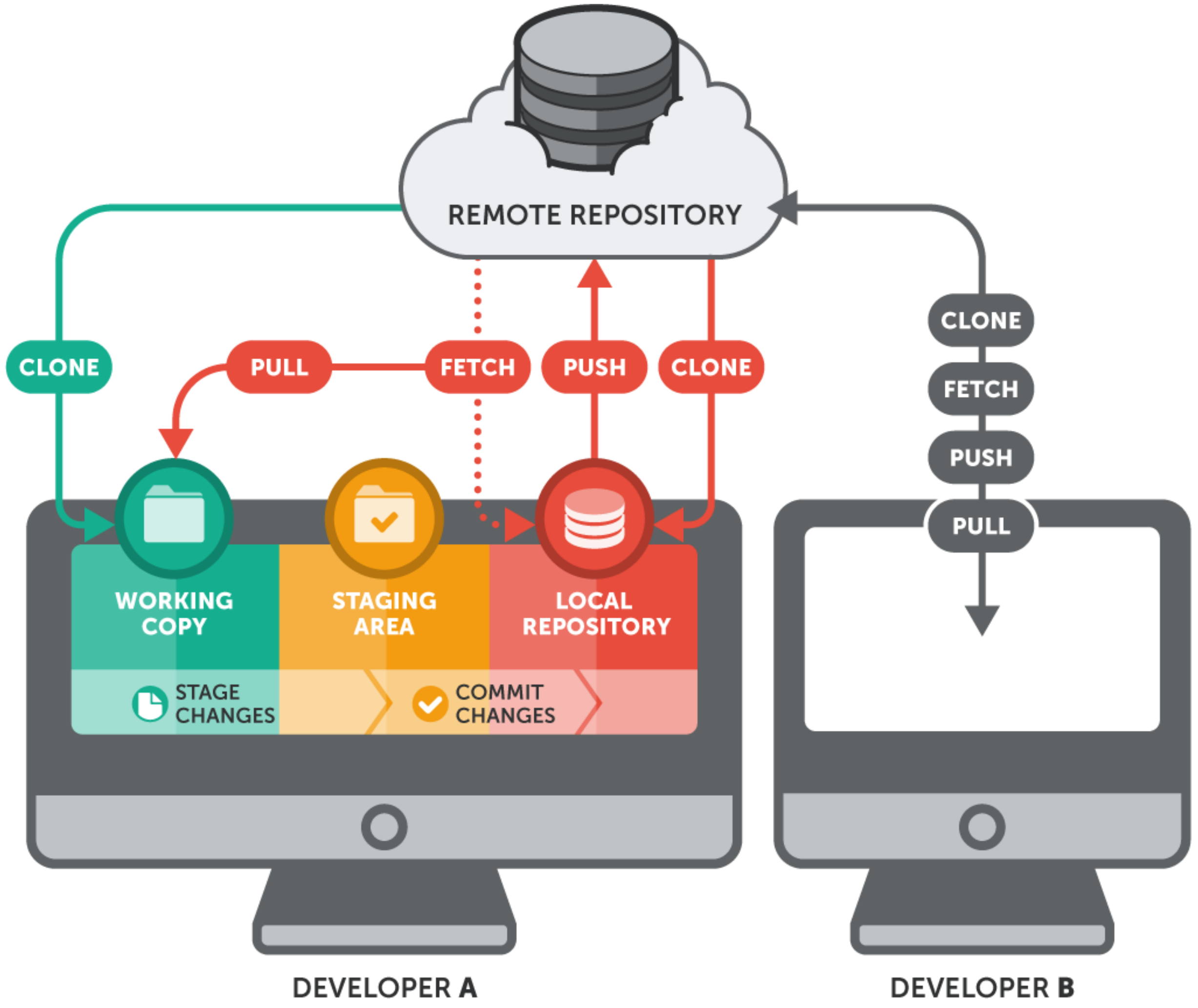


```
git checkout master
git merge iss53
```

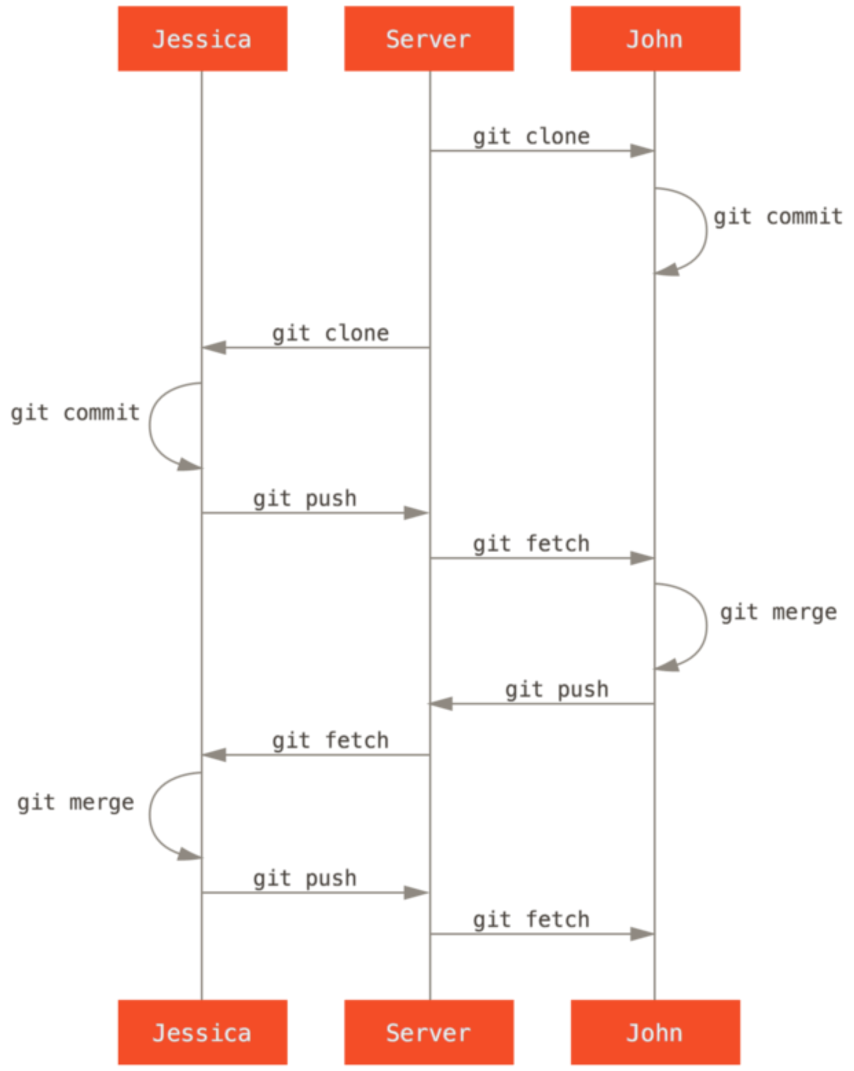


Git supports collaboration

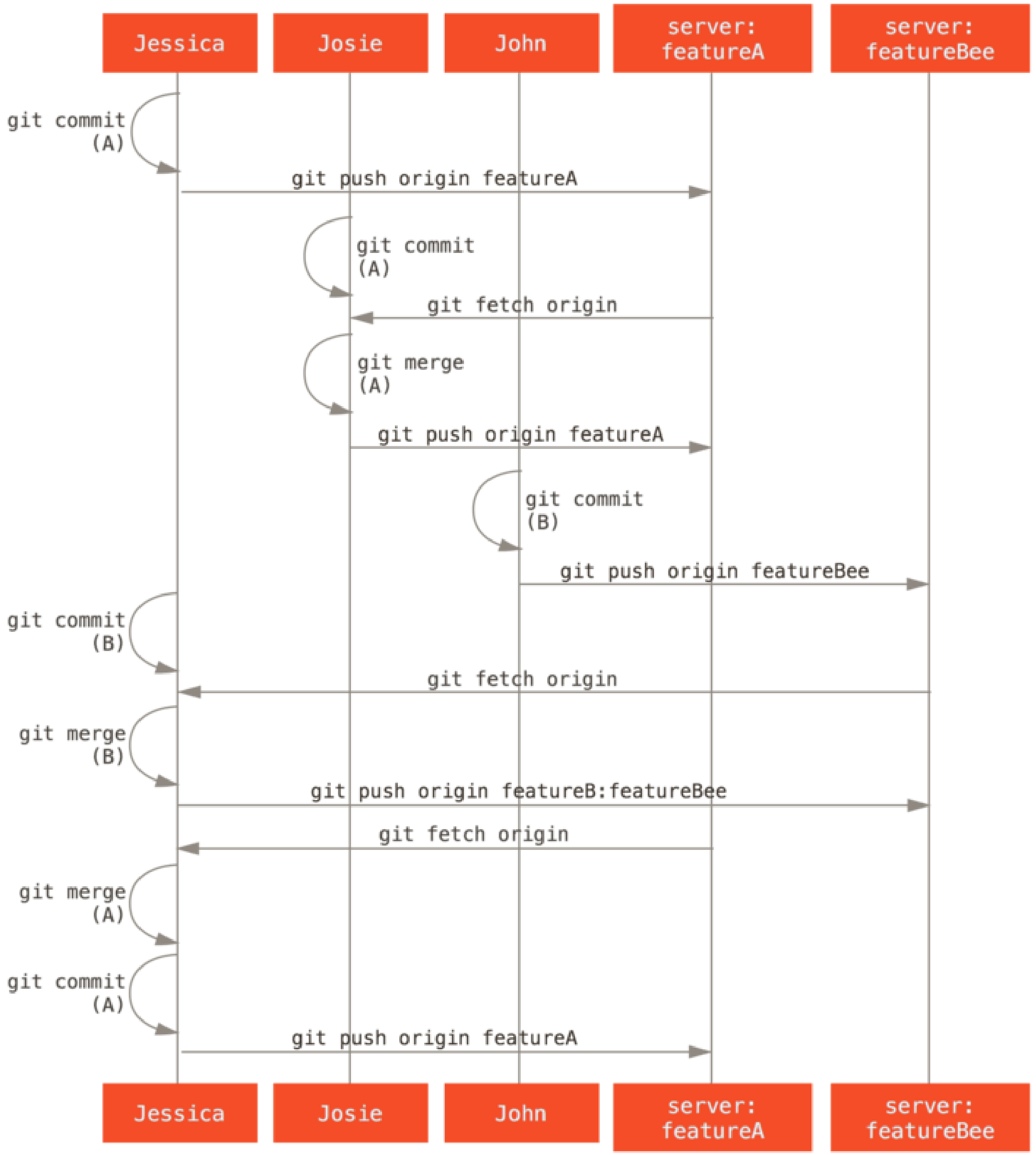
- **Local repository and Remote Repository**



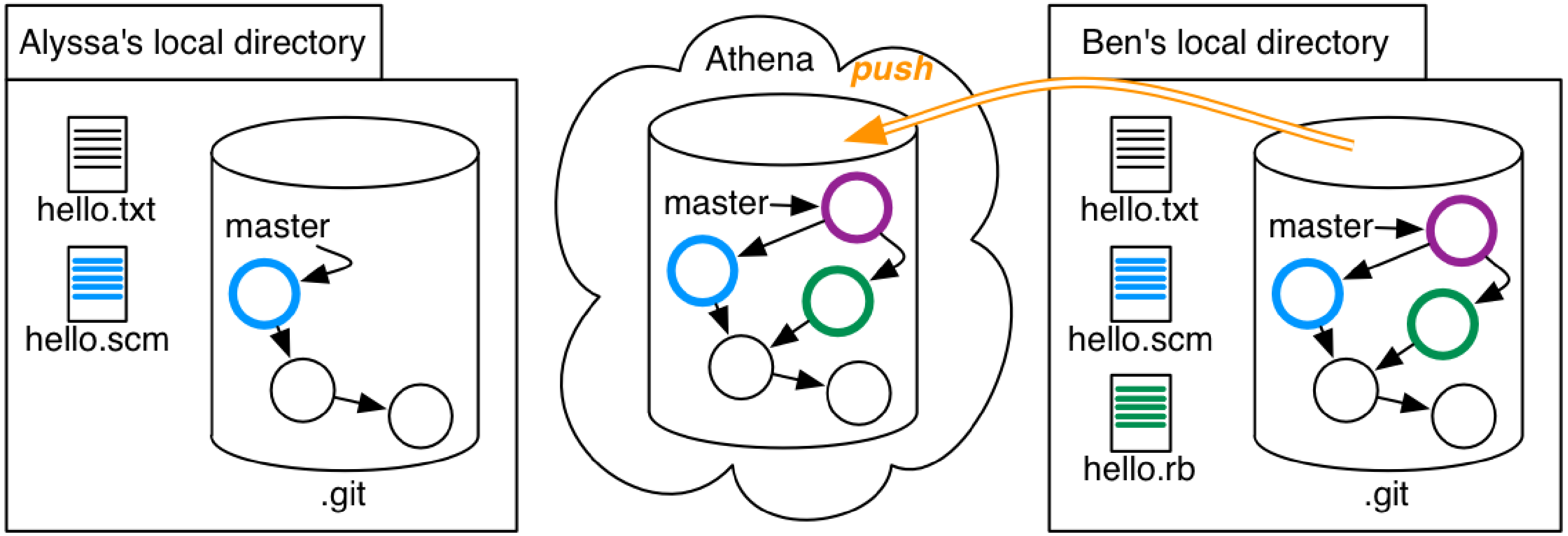
Git supports collaboration



Git supports collaboration



Git supports collaboration



结束!

September 27, 2024