# Introduction to Git

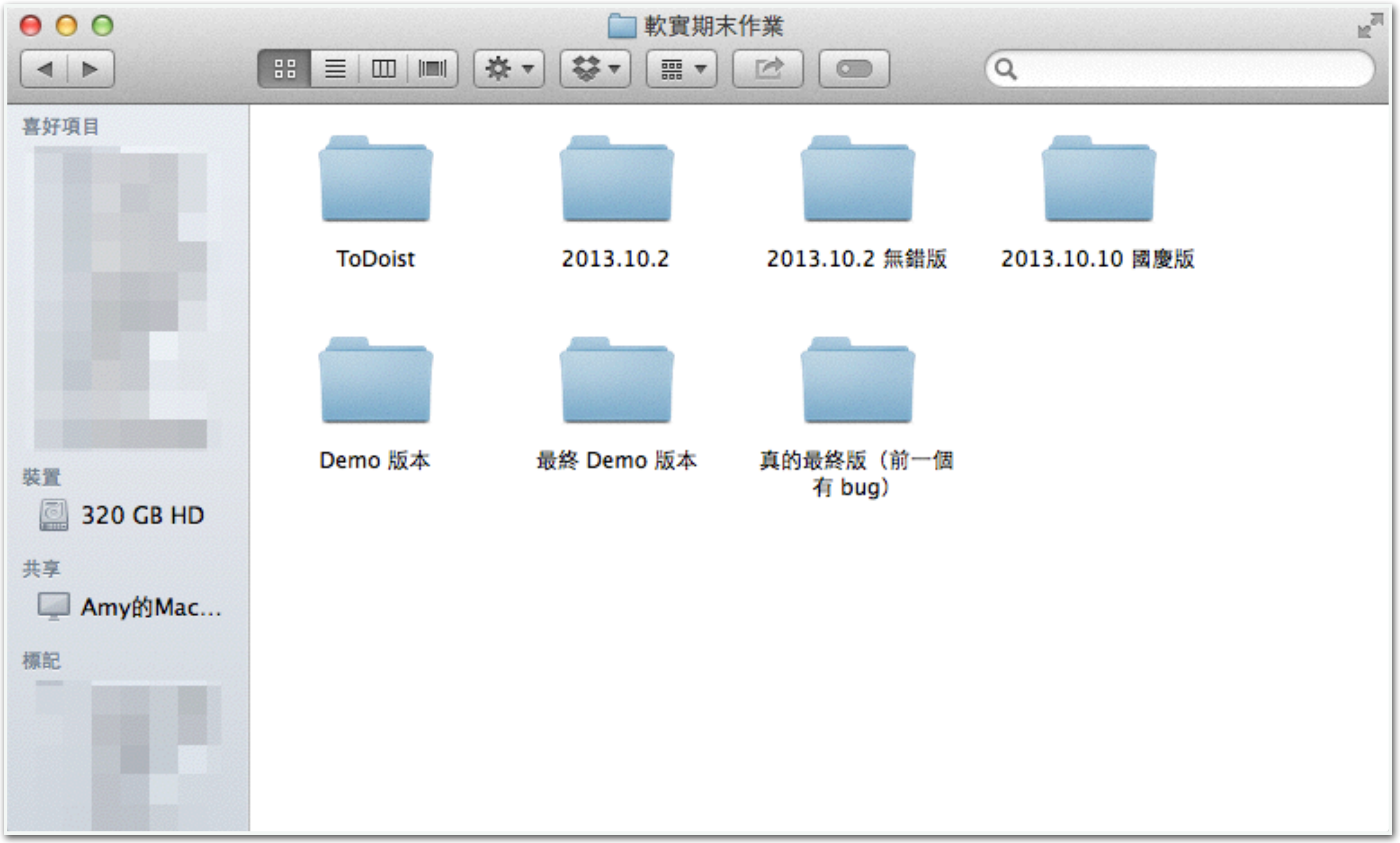
Cloud Databases
CS, NTHU
Spring, 2017

## What is Version Control ?

## Version Control System

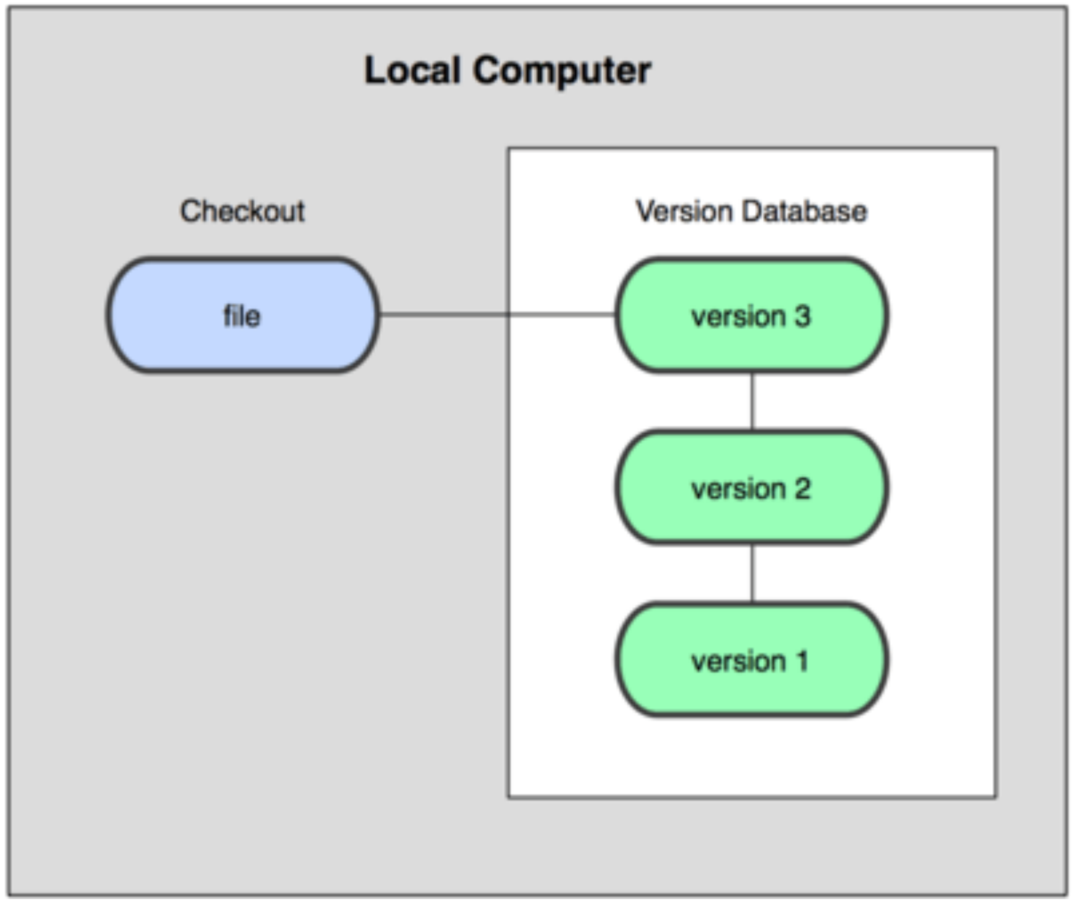
- Store the projects, keep your revision history
- Synchronization between modifications made by different developers

# Students' VCS



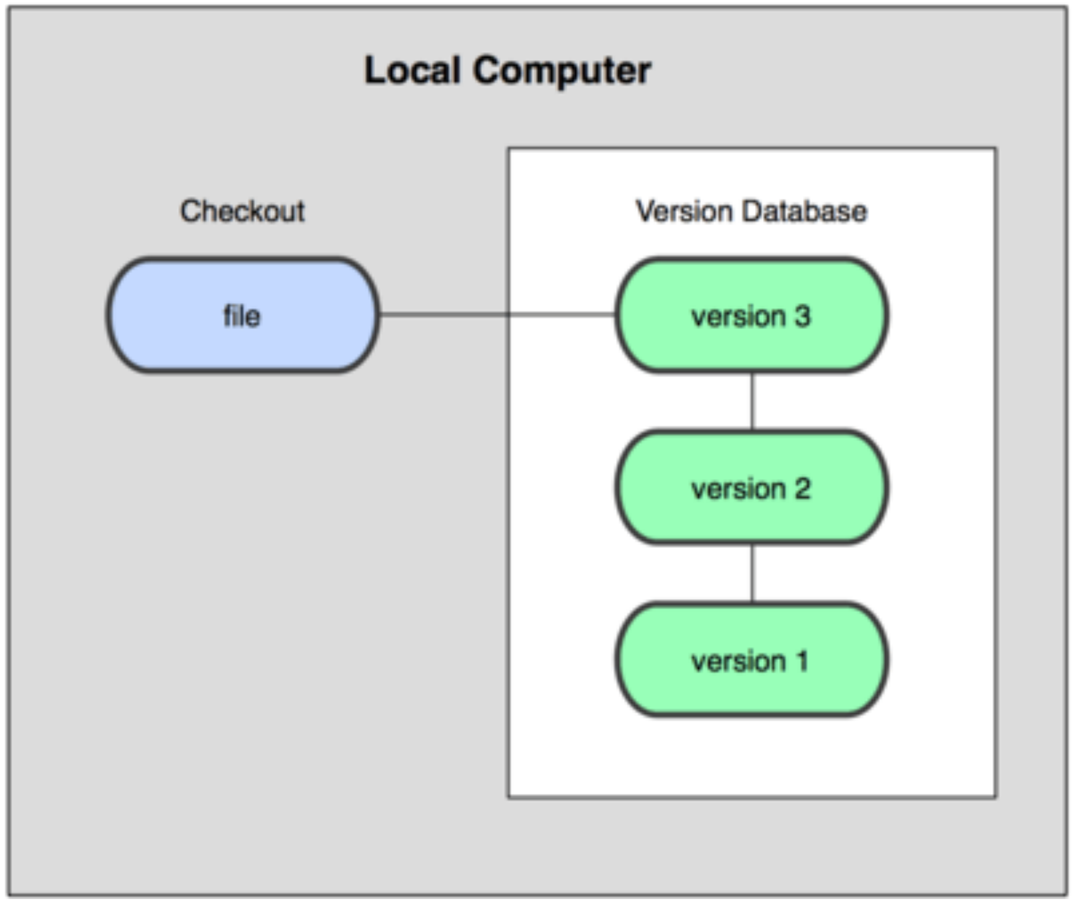
# Local VCS

- Build a simple version database in local
- rcs command



# Local VCS

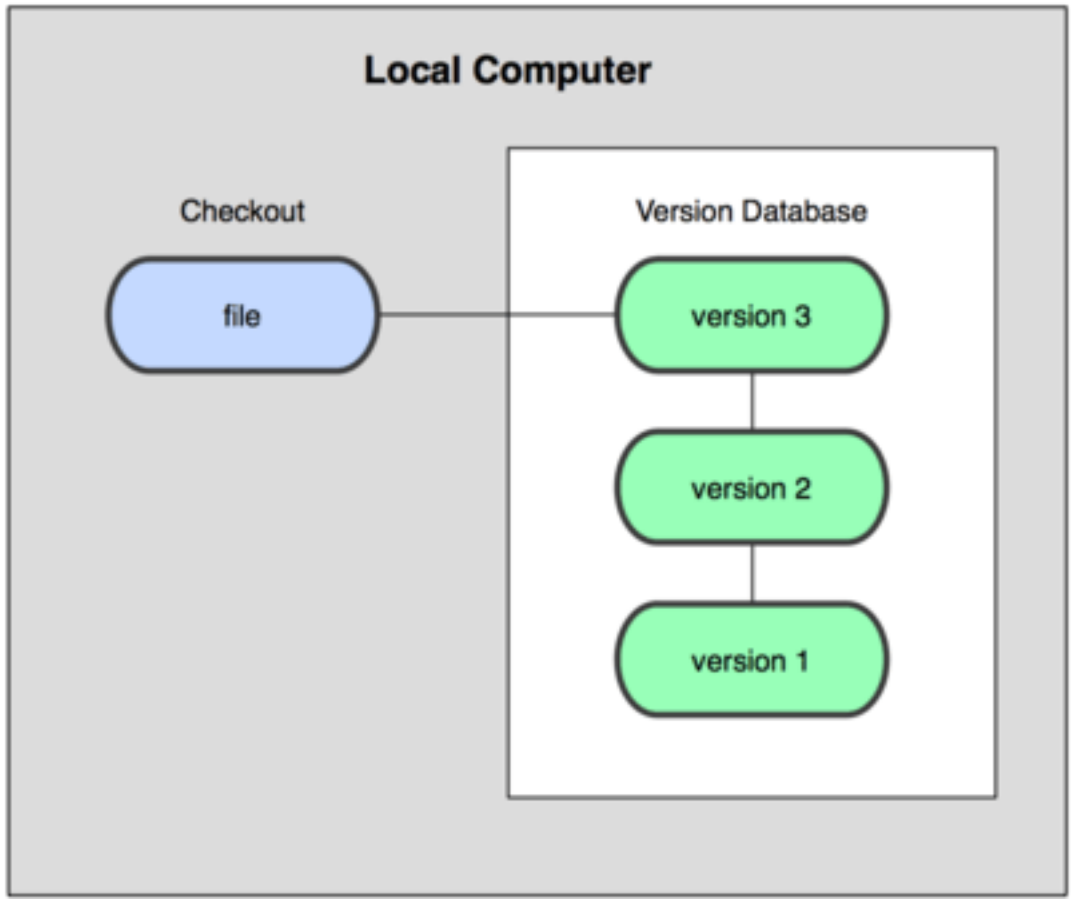
- Build a simple version database in local
- rcs command



**What if you got a partner ?**

# Local VCS

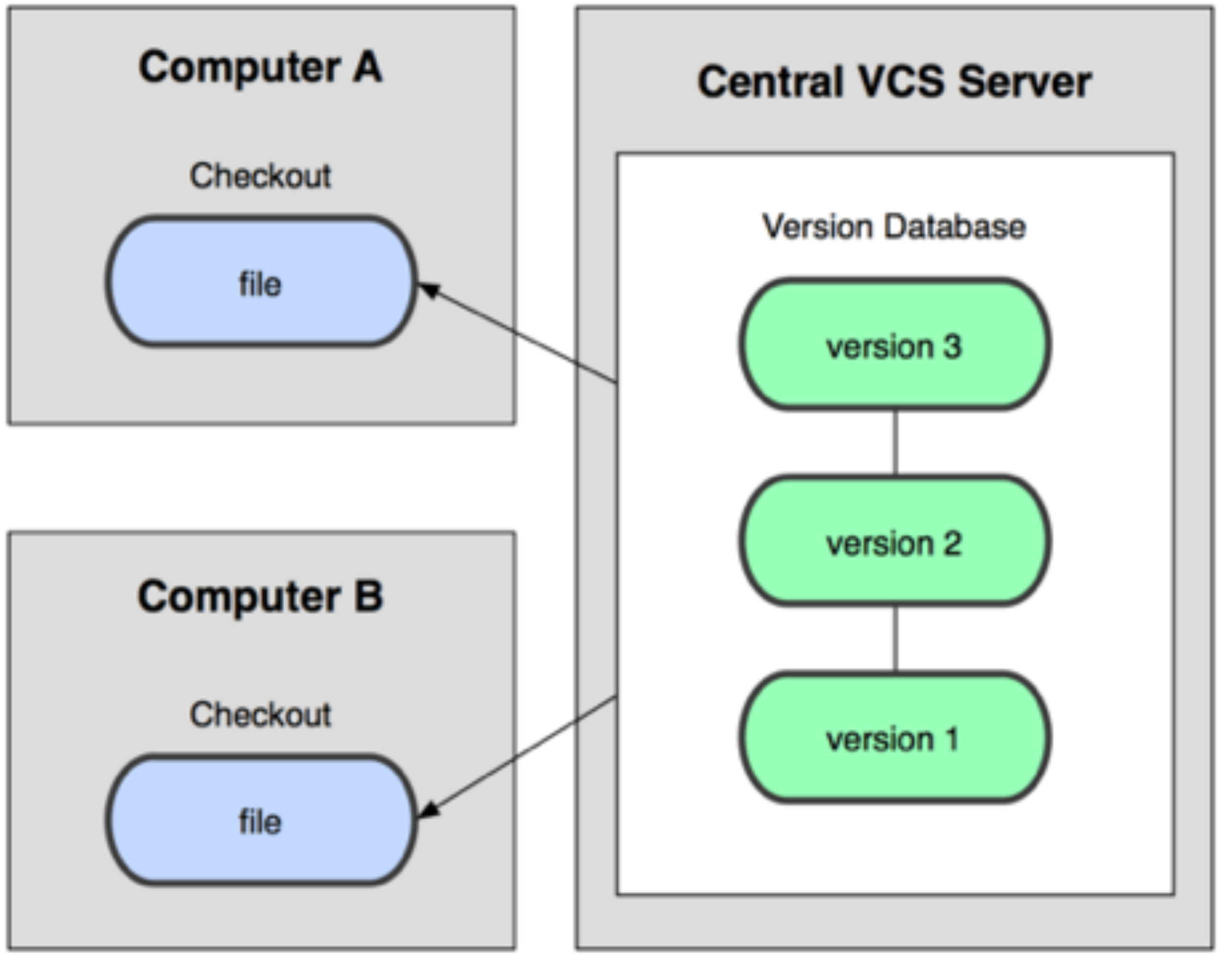
- Build a simple version database in local
- rcs command



**What if you got a partner ?** Dropbox **?**

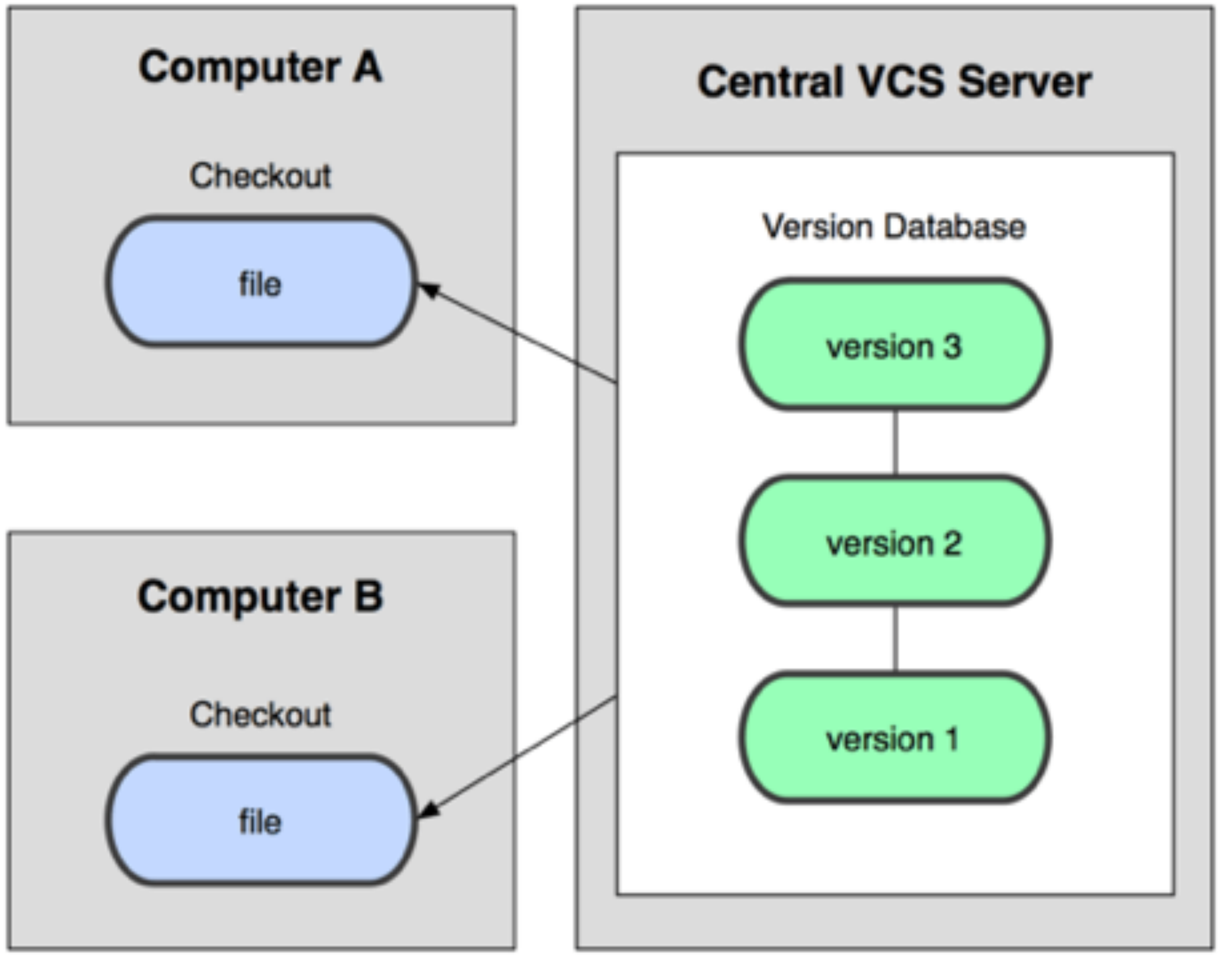
# Centralized VCS

- Single server that contains all the versioned files
- Subversion(SVN), CVS



# Centralized VCS

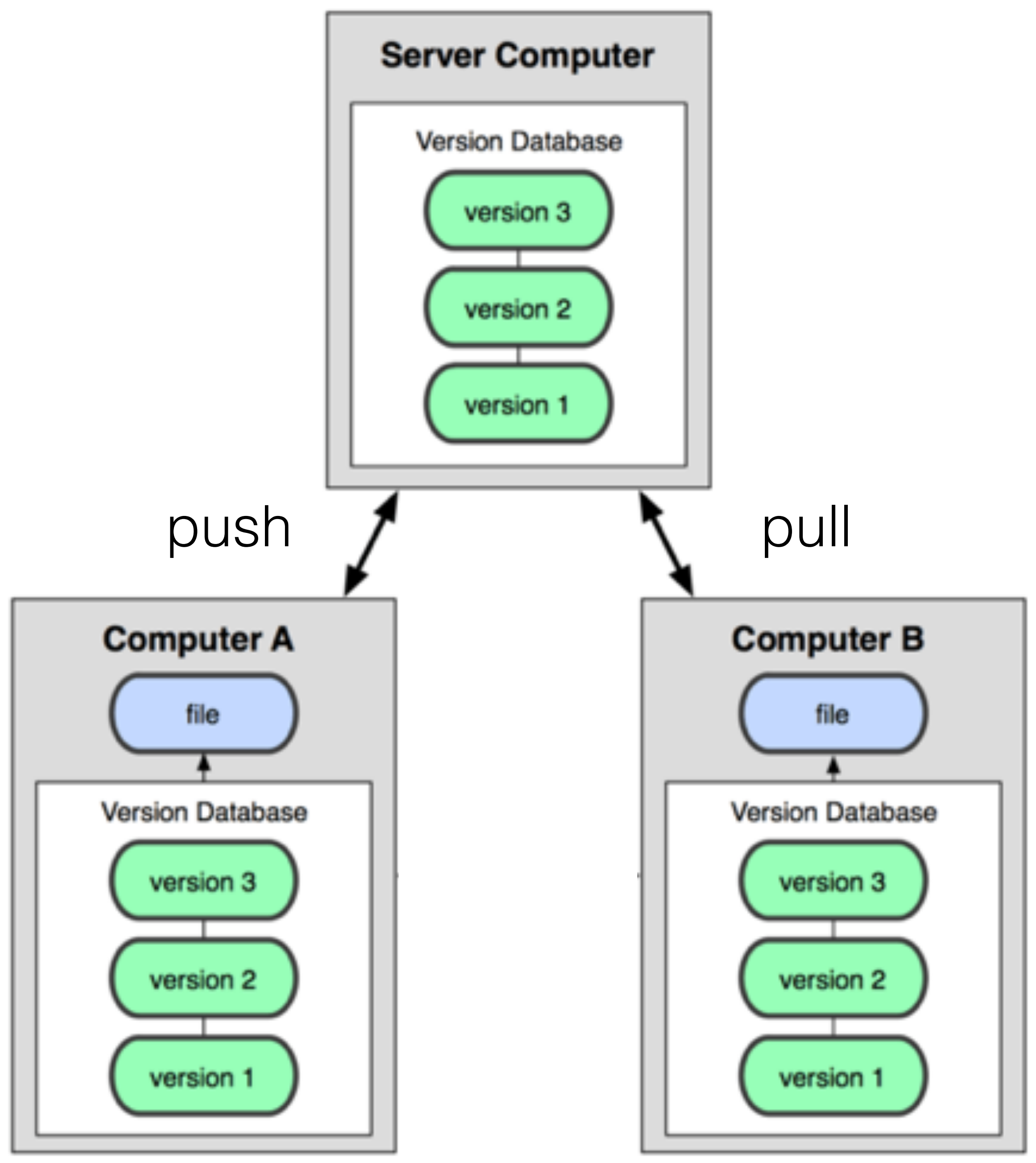
- Single server that contains all the versioned files
- Subversion(SVN), CVS



**Single point of failure ?**

# Distributed VCS

- Clients fully mirror the repository
- **Git**, Mercurial, Bazaar



## Why Git is Better ?

- Fast
- Simple
- Fully distributed
- Able to handle large projects (ex. Linux Kernel)

## Git Basics

## Installation

- You can either instal git
  - from source
  - from binary installer
- Please check this link
  - http://git-scm.com/book/en/Getting-Started-Installing-Git

## Configuration

- Modify ~/.gitconfig
- Or, type in following commands

```
git config --global user.name “your name”
git config --global user.email “your@email.com”
```

For more information, please refer this link

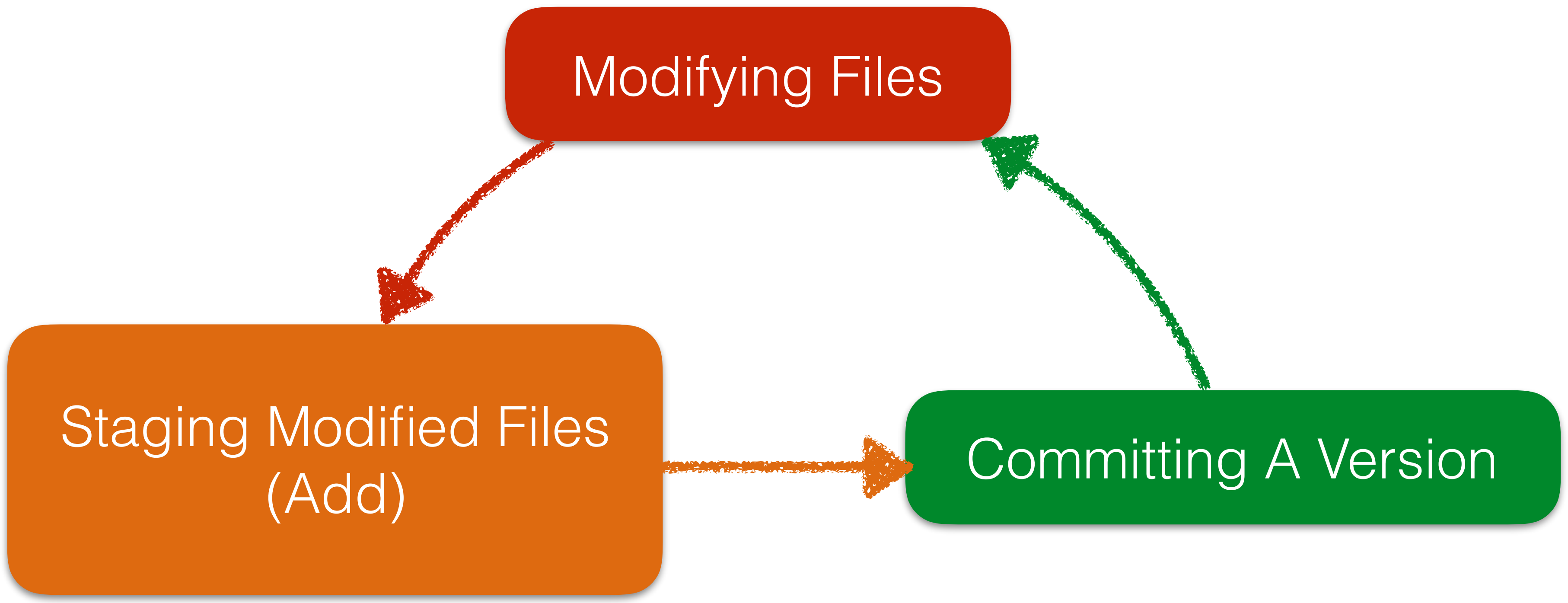
# Creating a new Repository

- Two ways to create a repository
  - Initializing a Repository in an Existing Directory

    ```
    git init
    ```

  - Cloning an Existing Repository
    - We will talk about it later
- The repository information will be stored in the .git directory

# Committing A Version



# Committing A Version

- Staging (adding) a file

```
git add [file name]
```

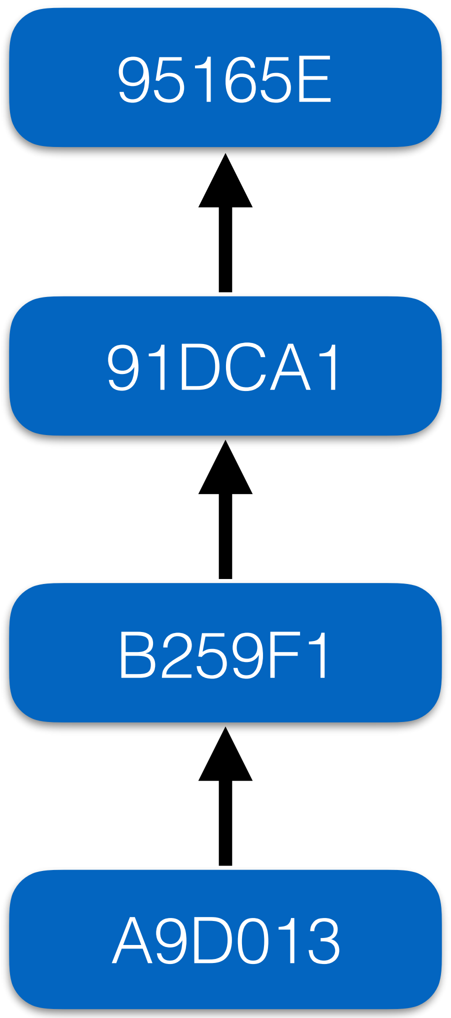
- Staging all files in the current directory

```
git add -A
```

- Committing

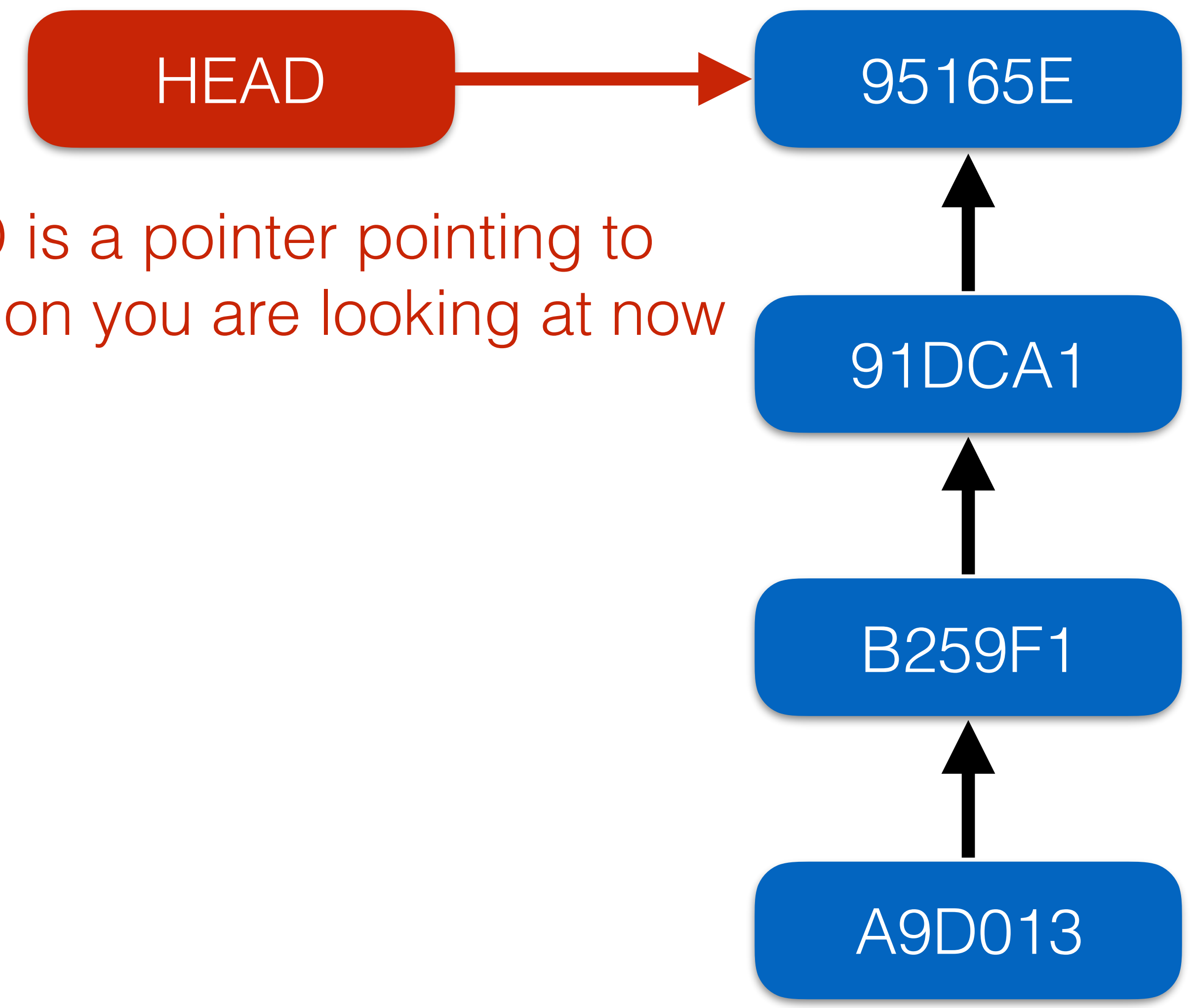
```
git commit -m “[message]”
```

# A History Tree



The versions you committed
form a history tree

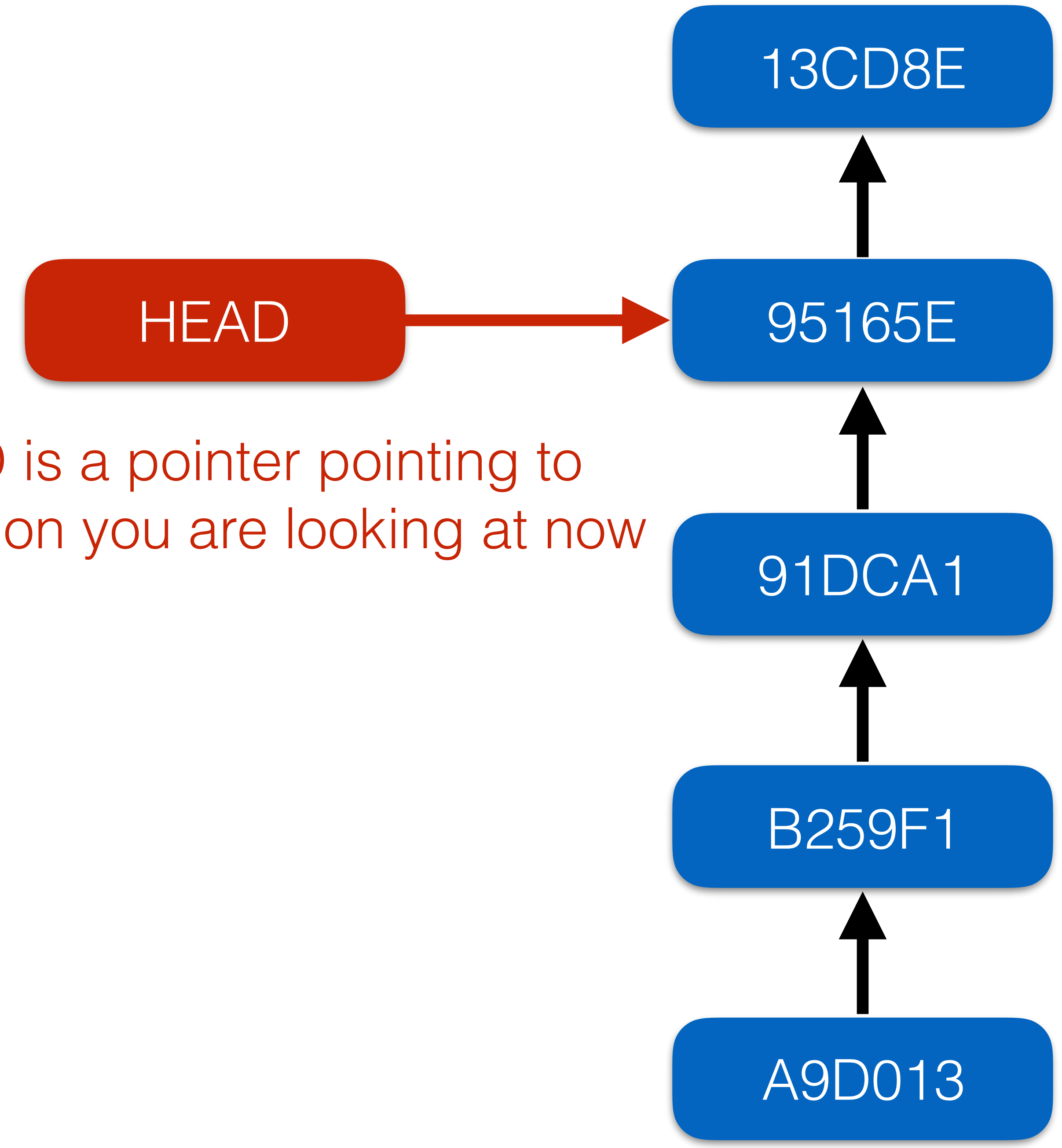
# A History Tree



HEAD is a pointer pointing to the version you are looking at now

The versions you committed form a history tree

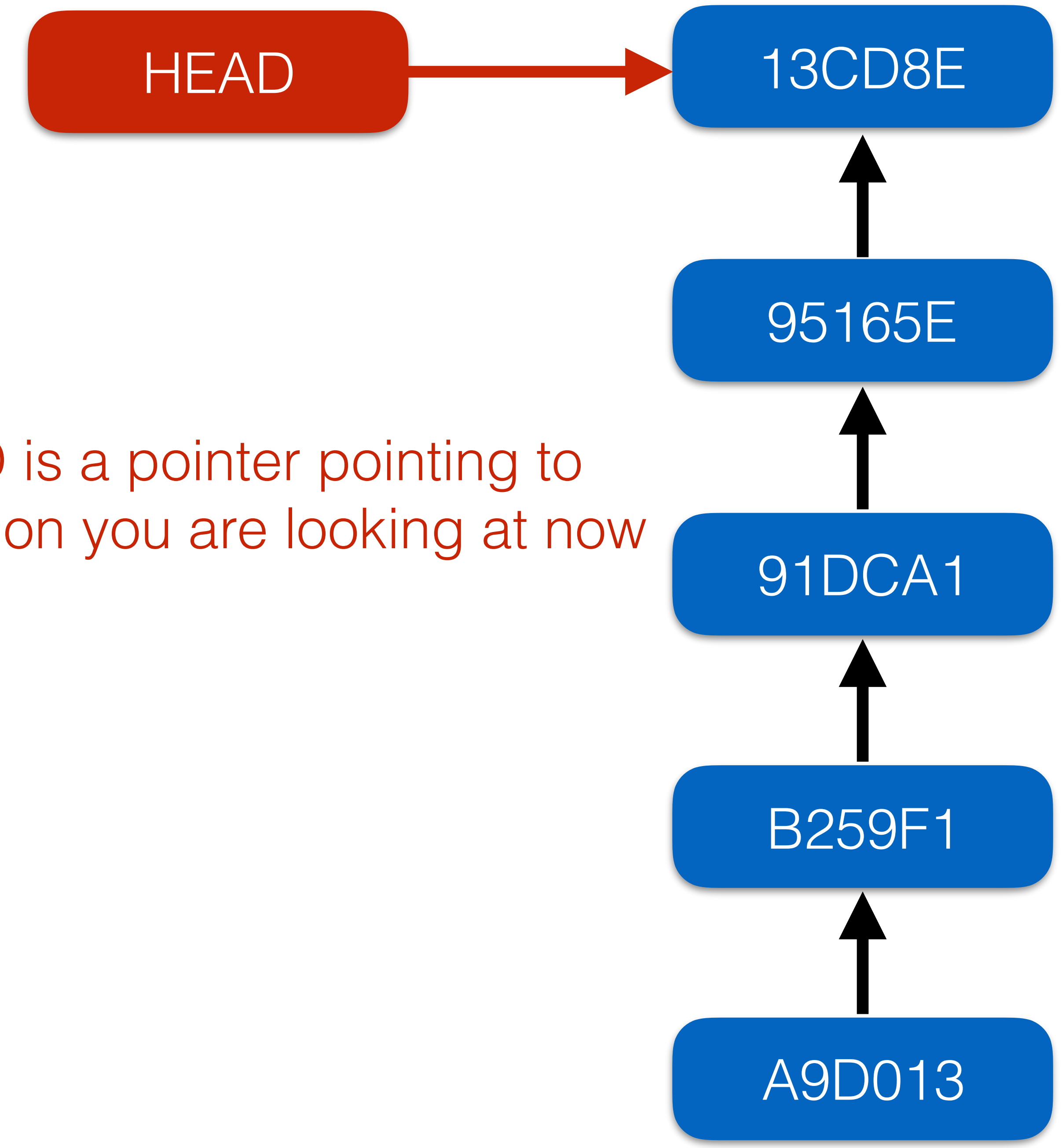
# A History Tree



HEAD is a pointer pointing to
the version you are looking at now

The versions you committed
form a history tree

# A History Tree



HEAD is a pointer pointing to
the version you are looking at now

The versions you committed
form a history tree

## Logs

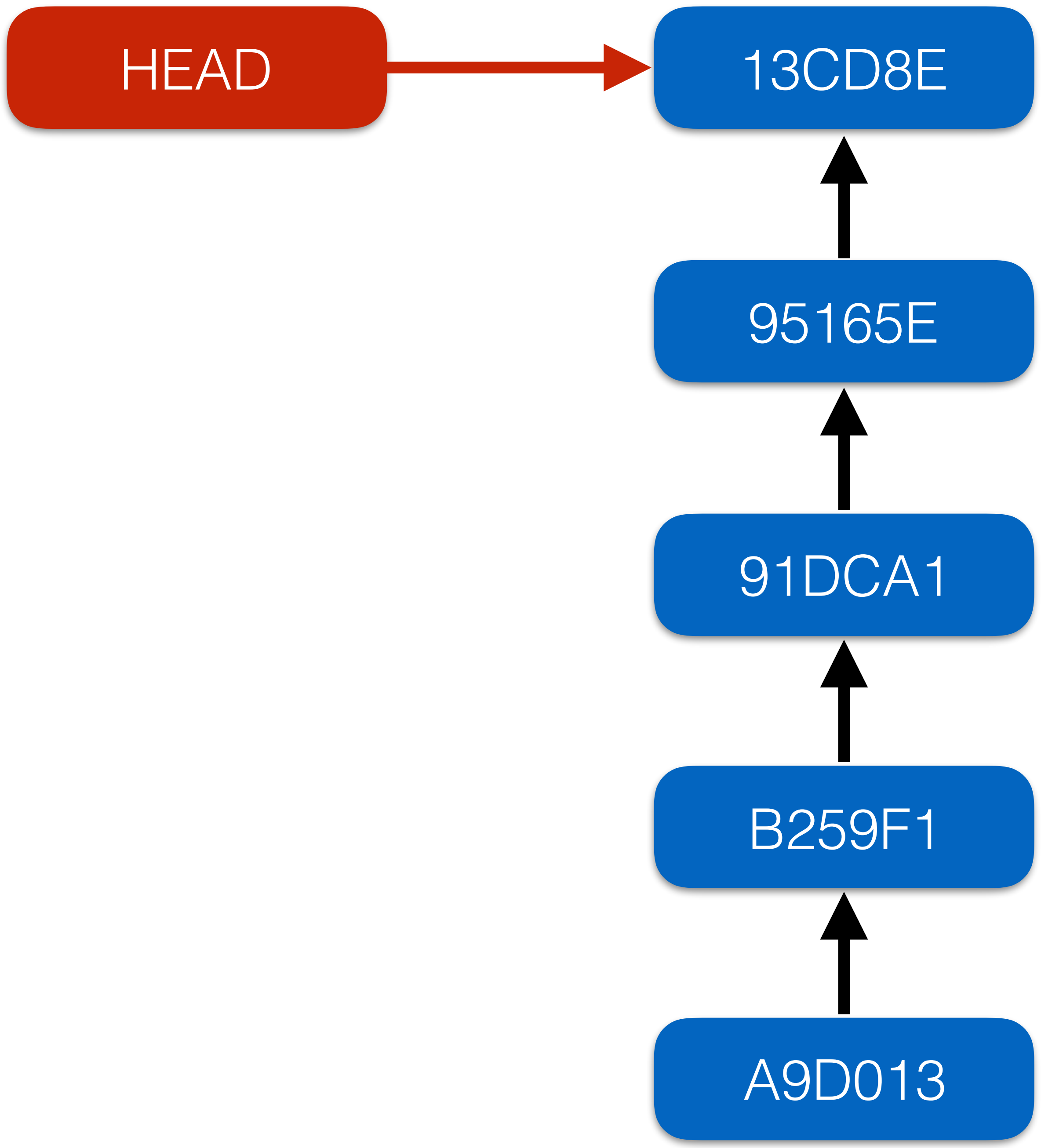
- Listing the log

```
git log
```

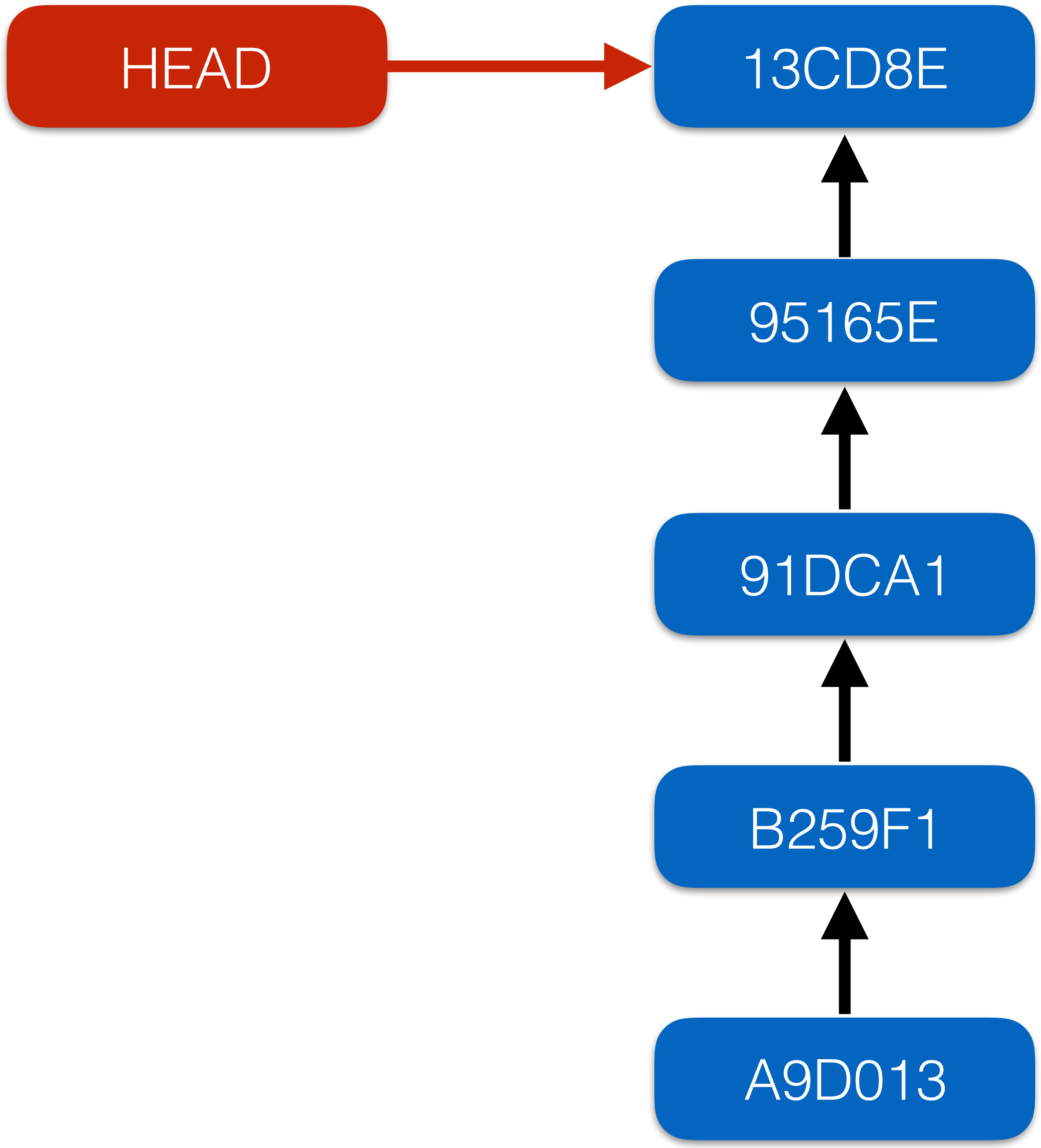
- Listing each log in one line

```
git log --oneline
```

# Checking Out A Version

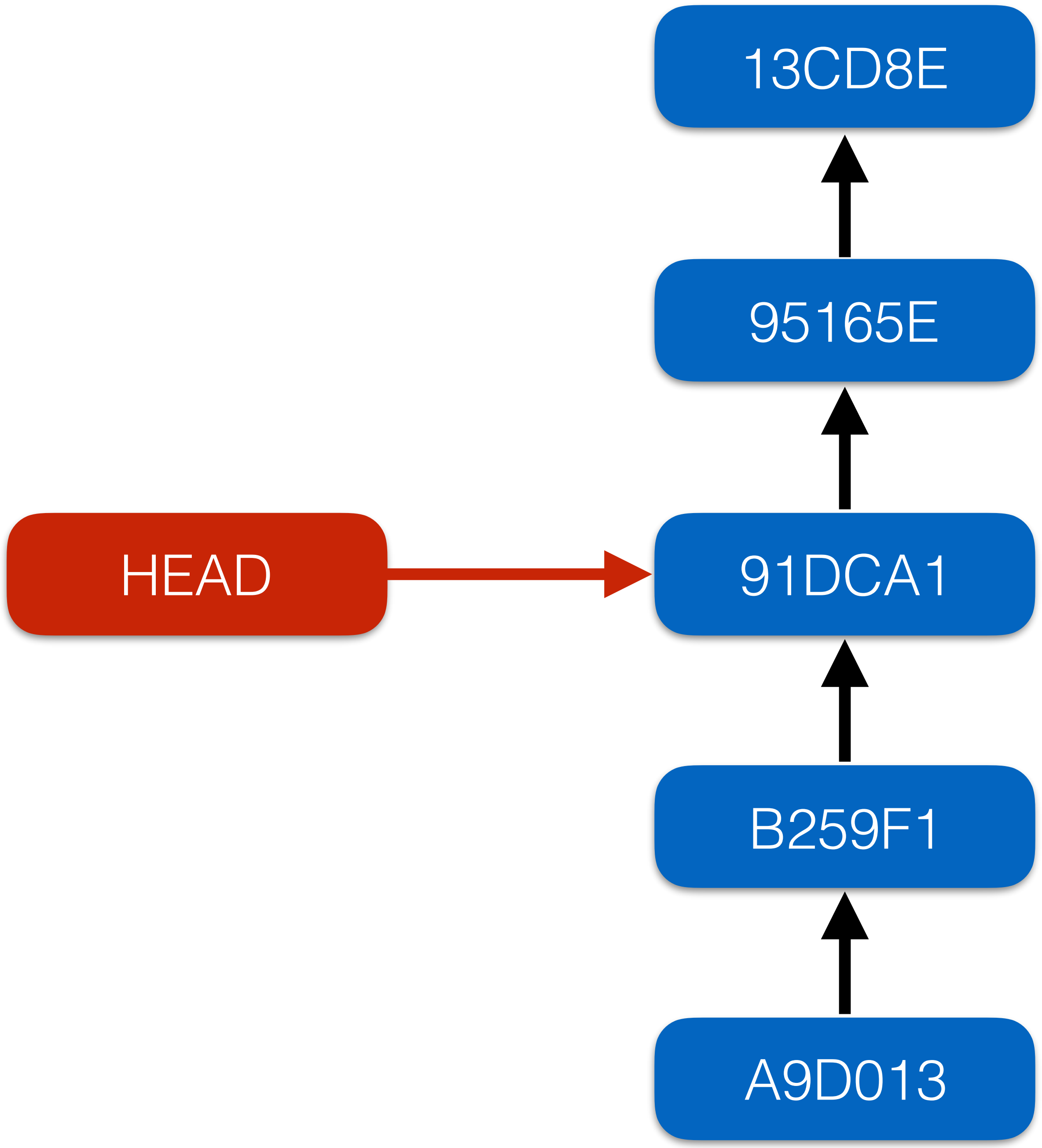


# Checking Out A Version



```
git checkout 91DCA1
```

# Checking Out A Version

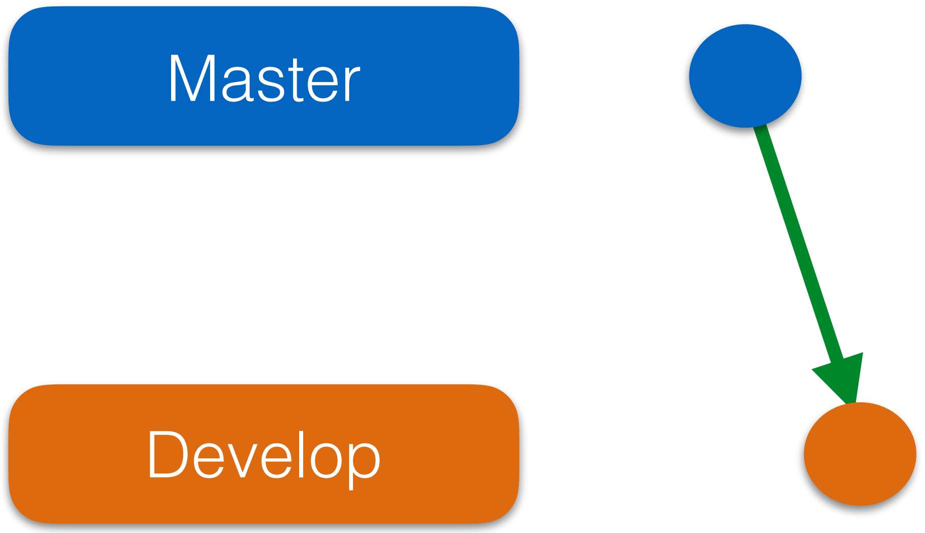


```
git checkout 91DCA1
```

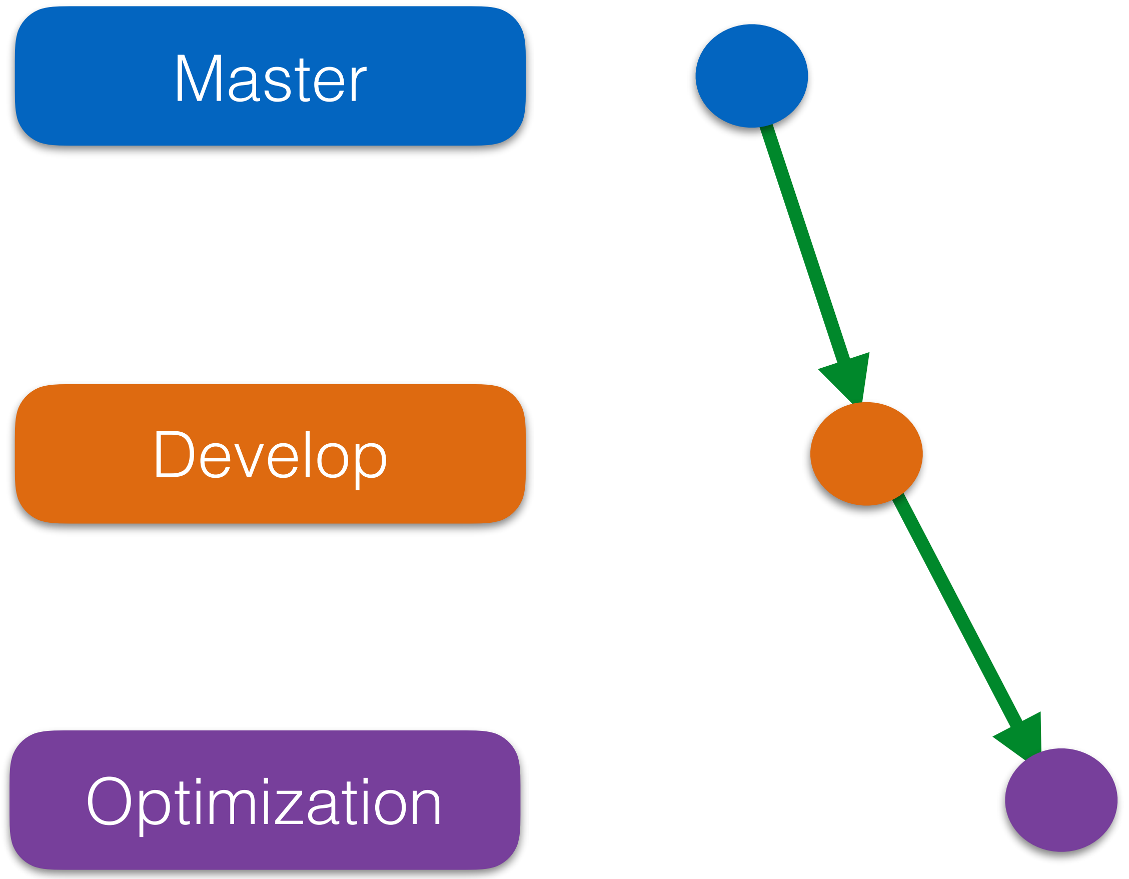
## Git Branching

## Branches

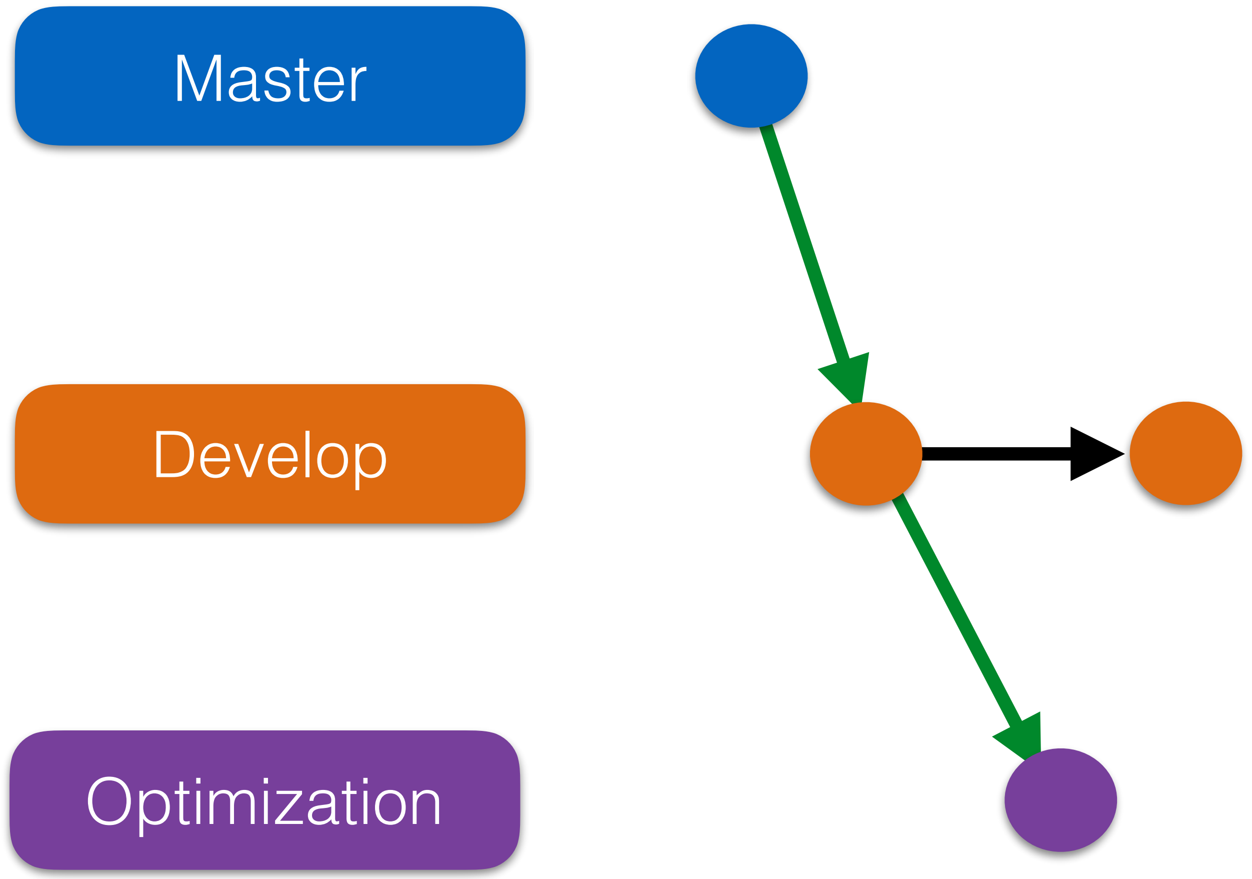
## Branches



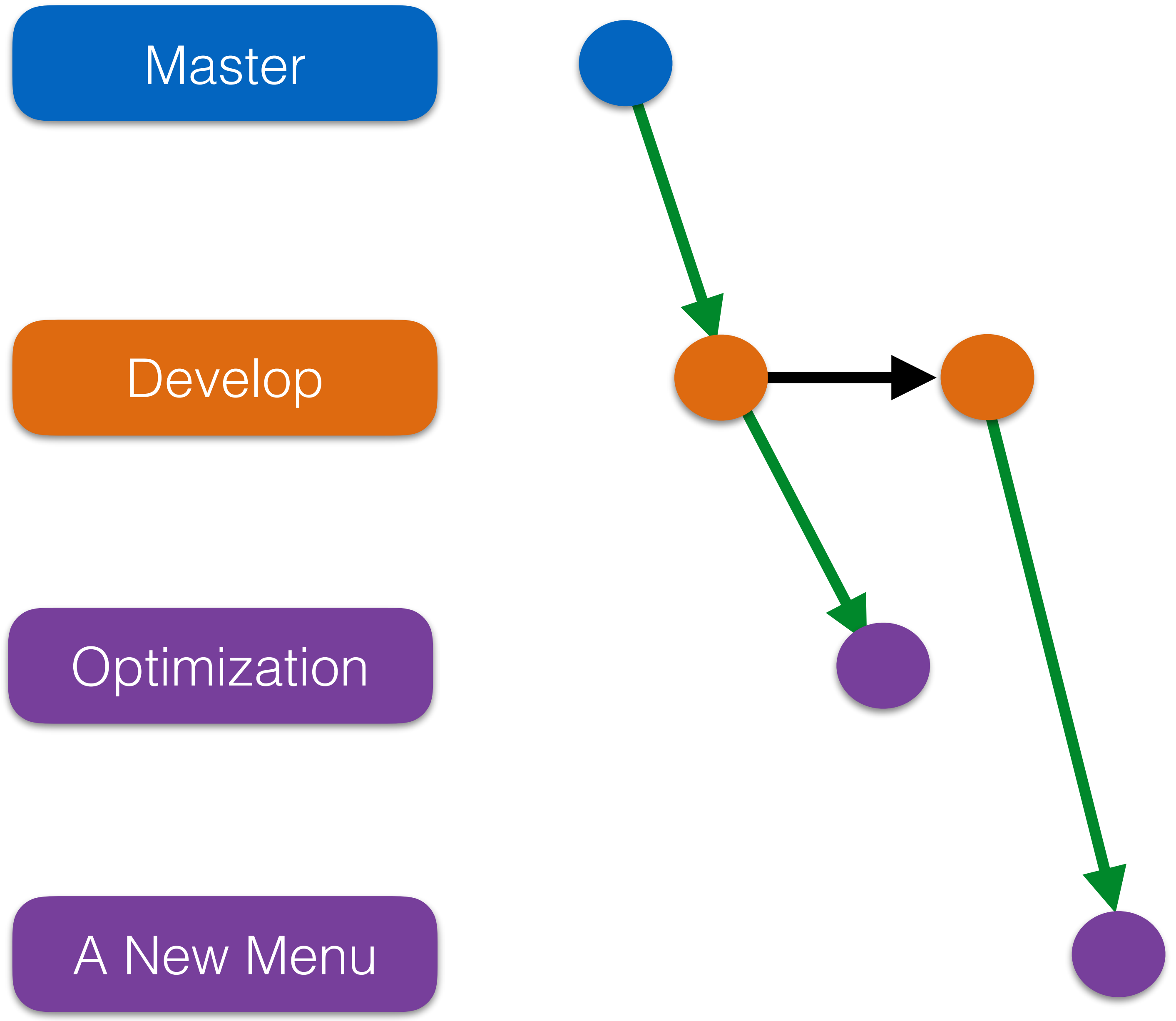
# Branches



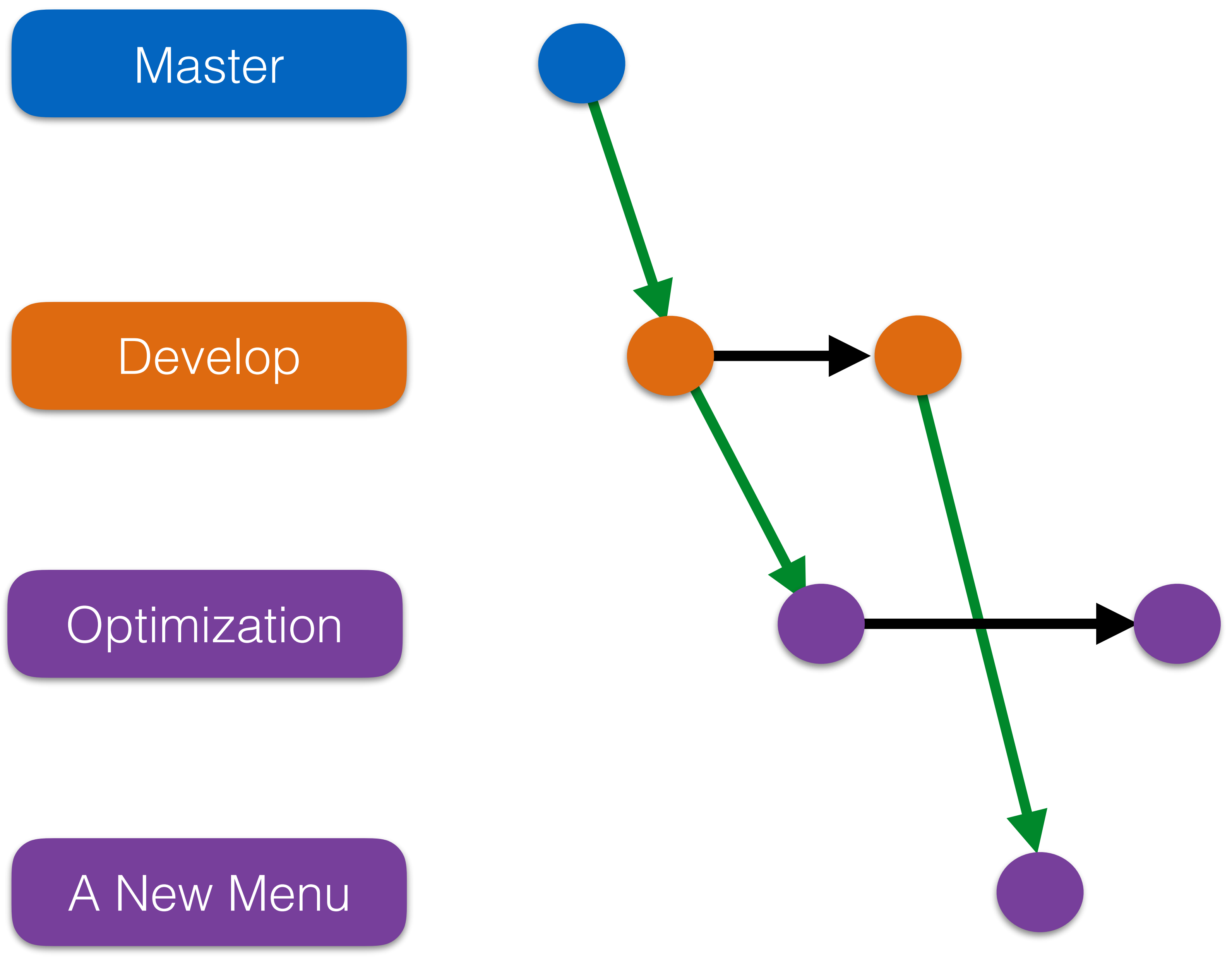
Branches



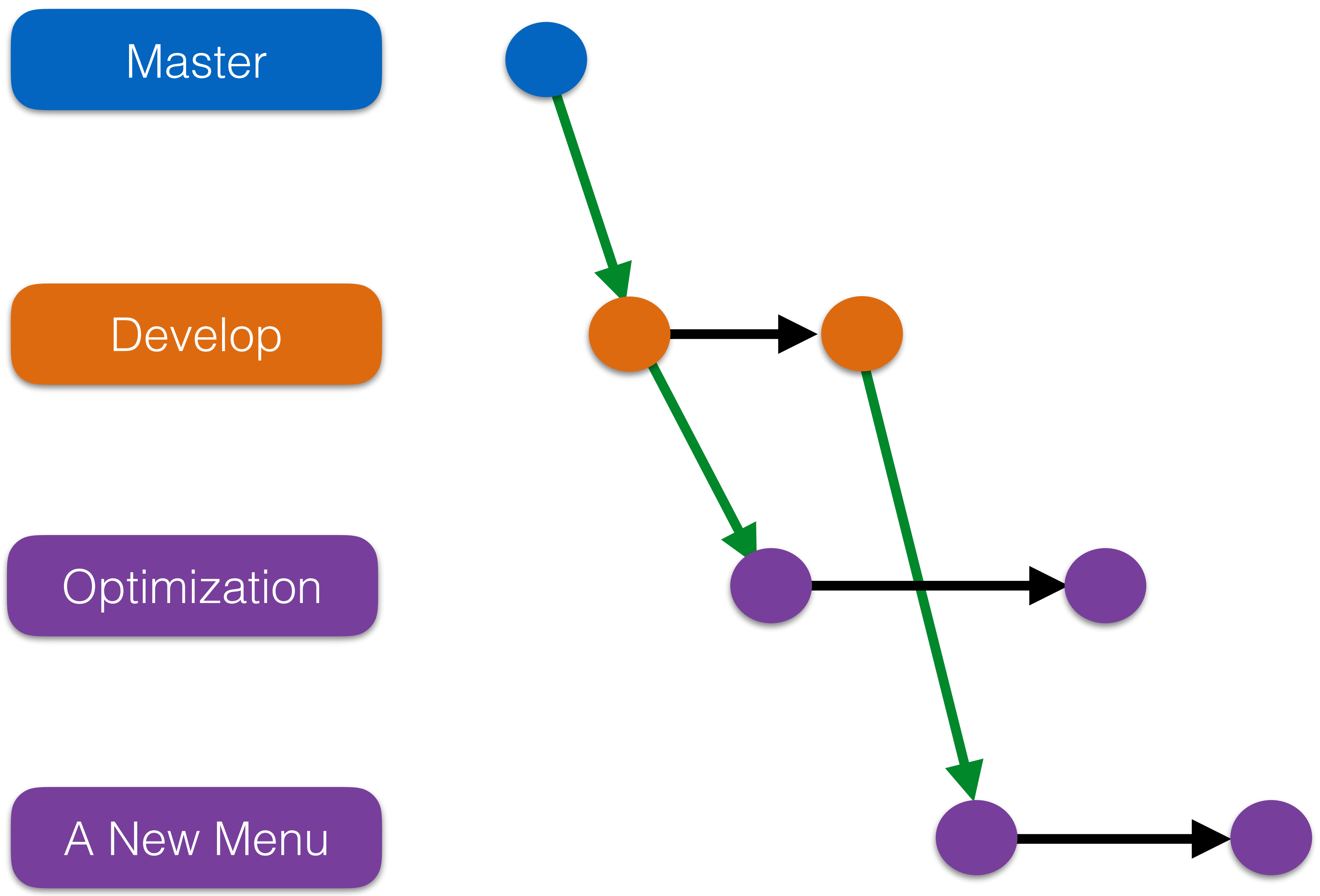
## Branches



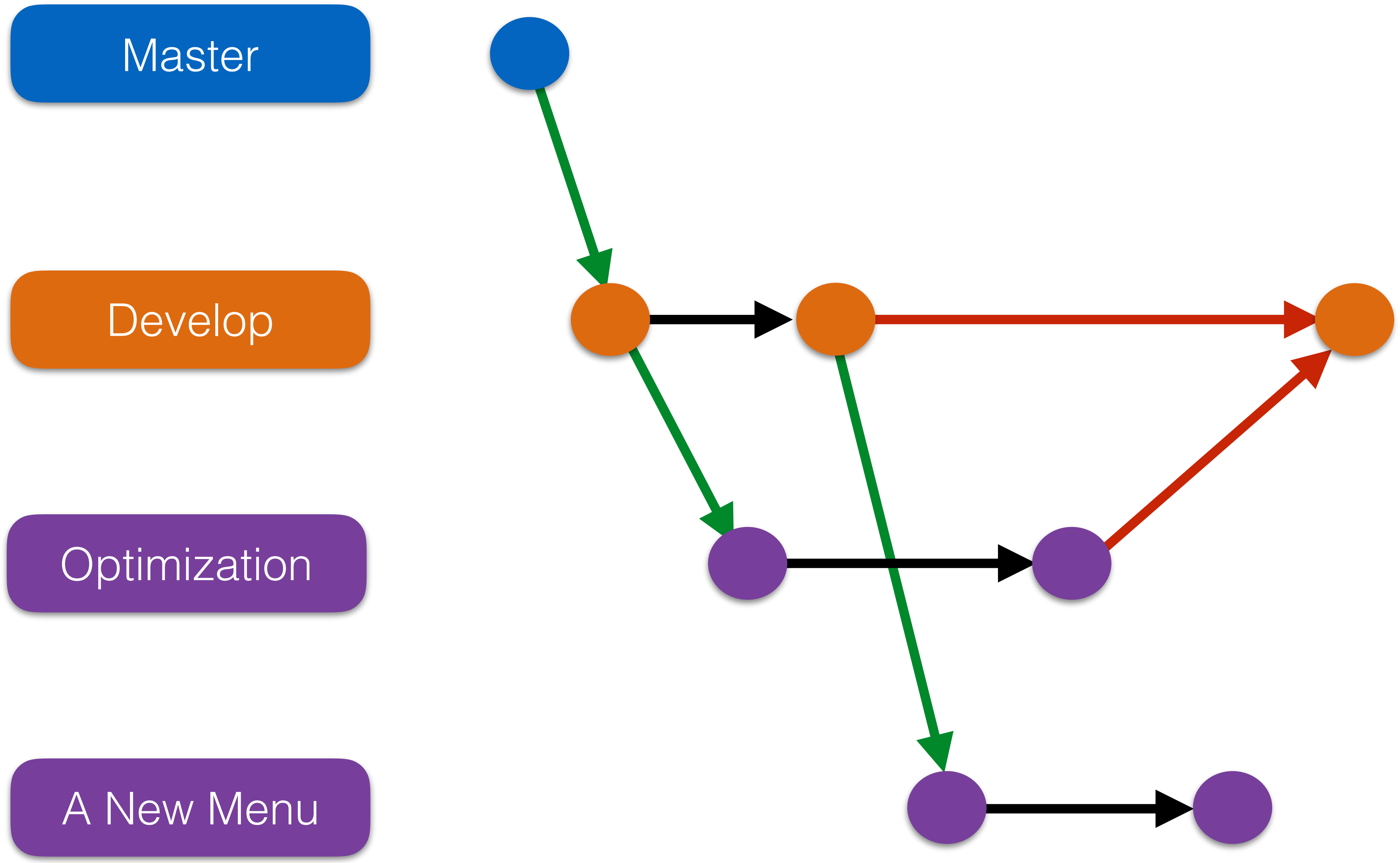
# Branches



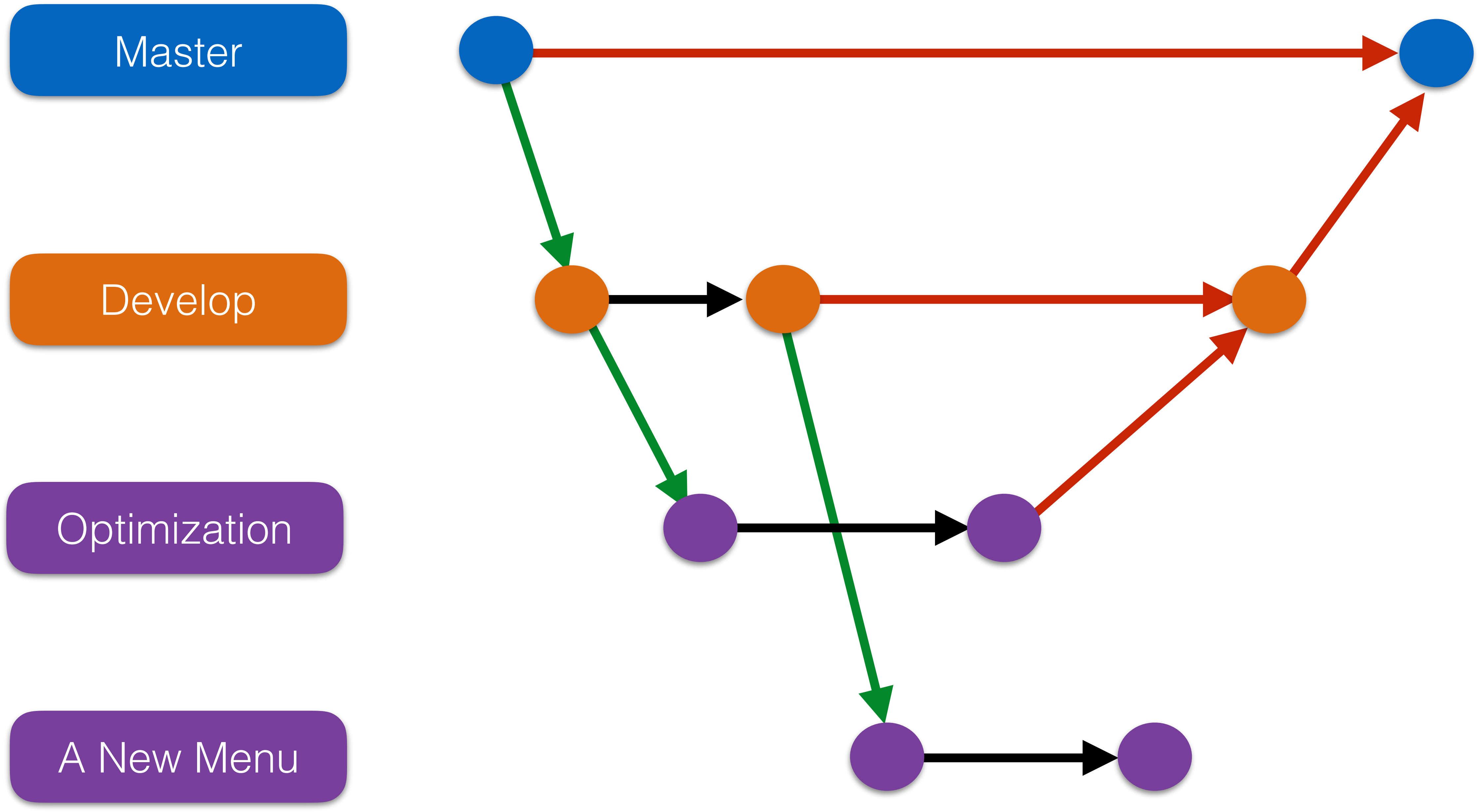
## Branches



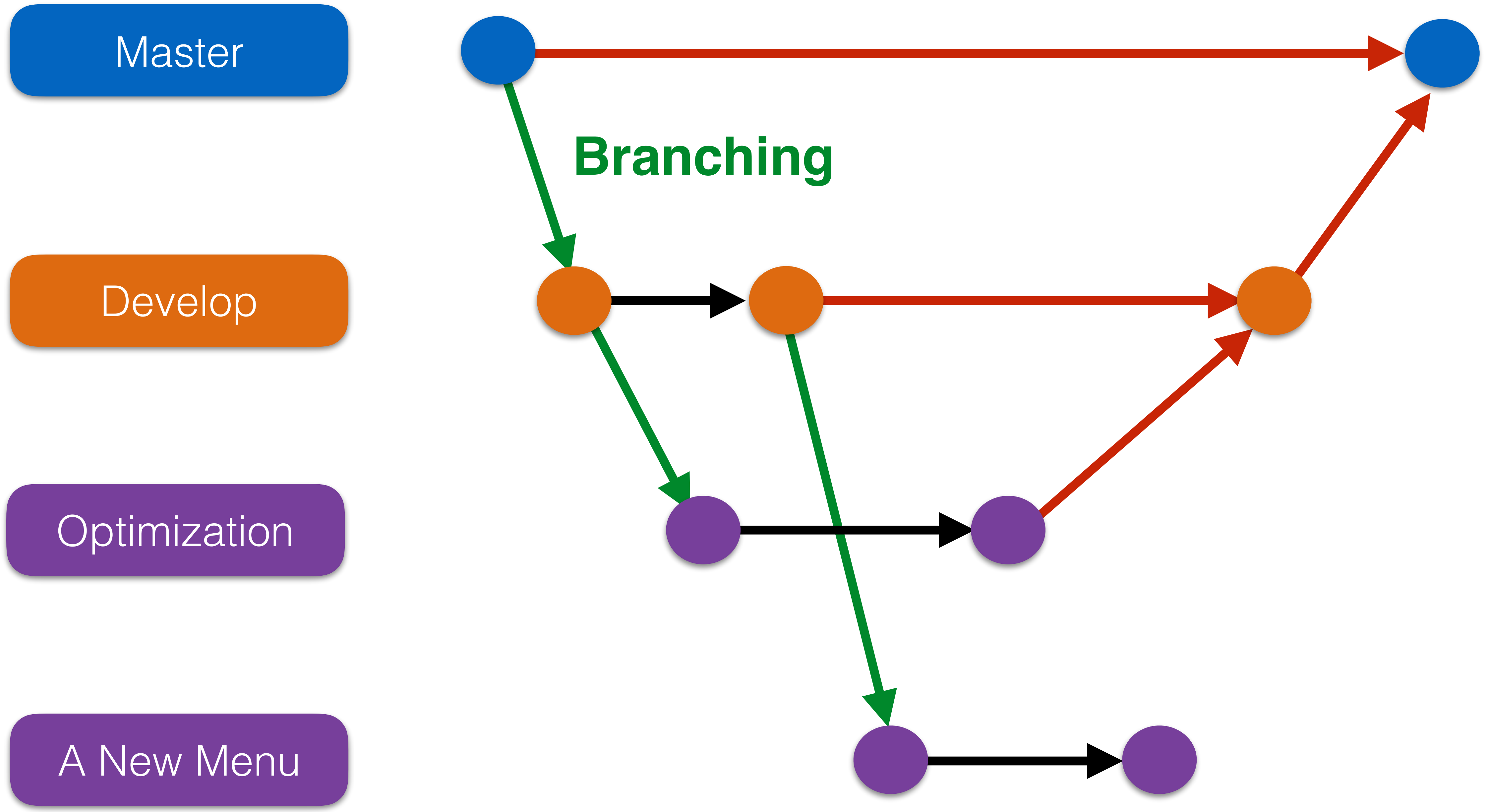
# Branches



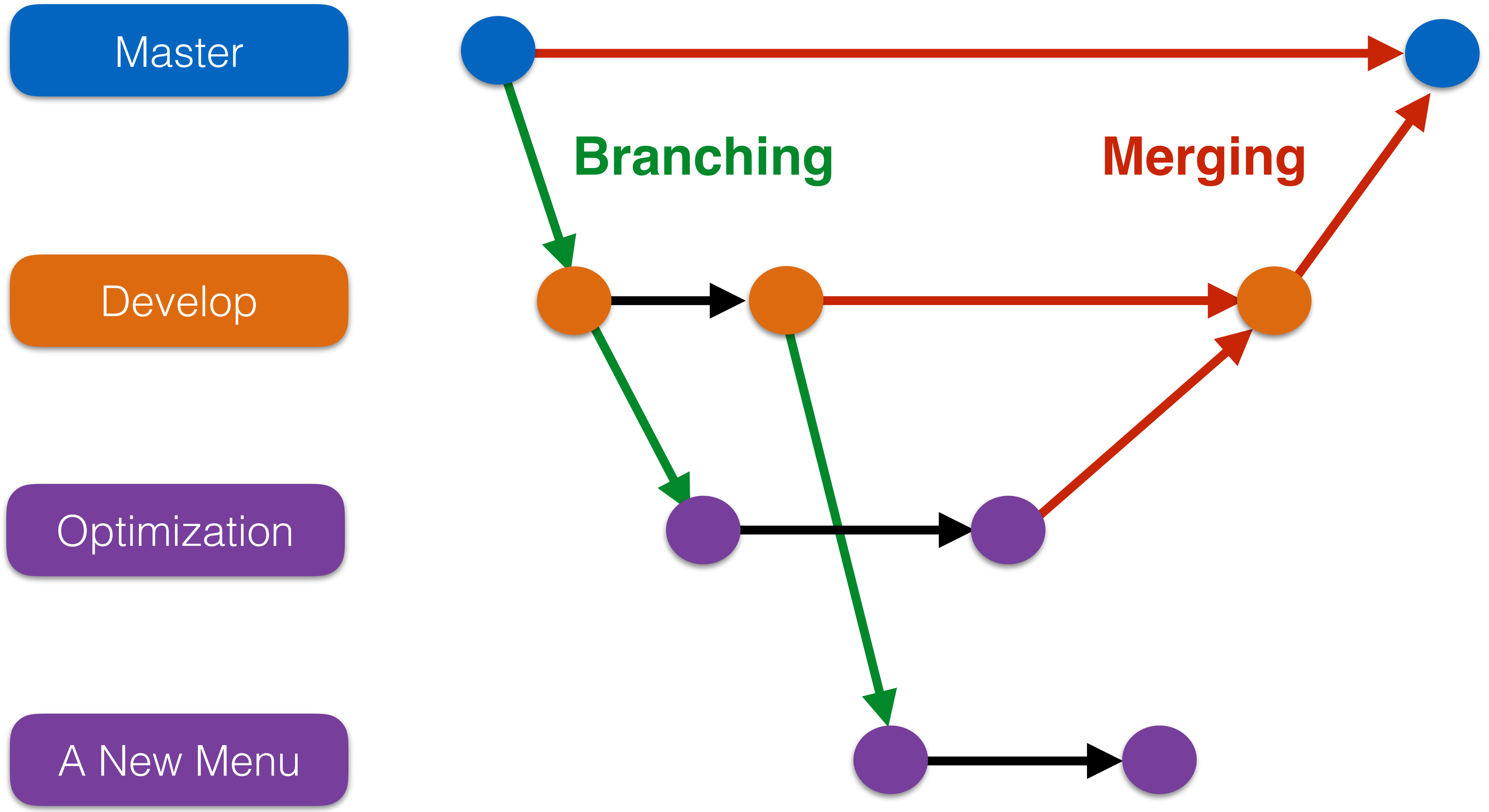
# Branches



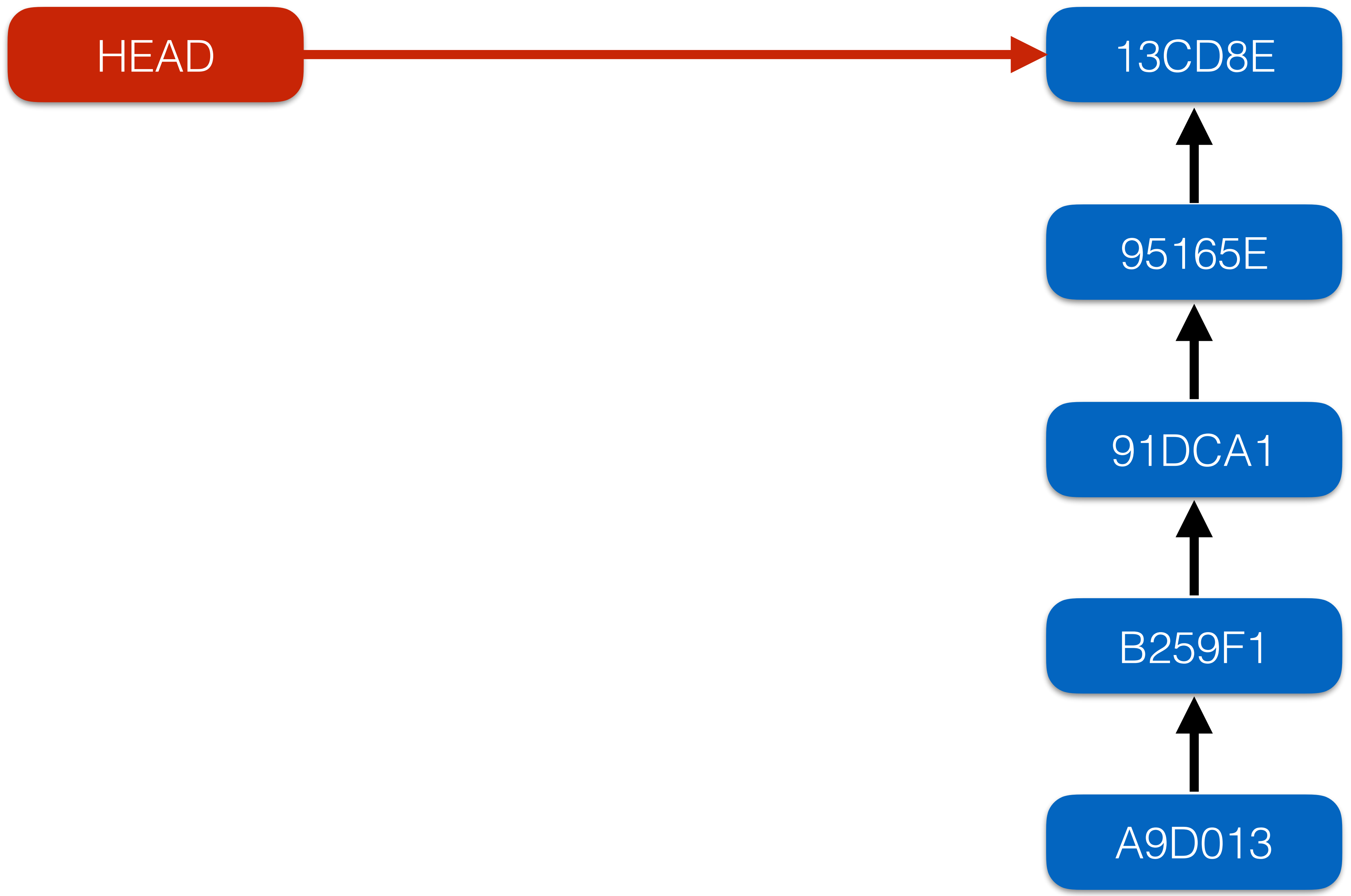
# Branches



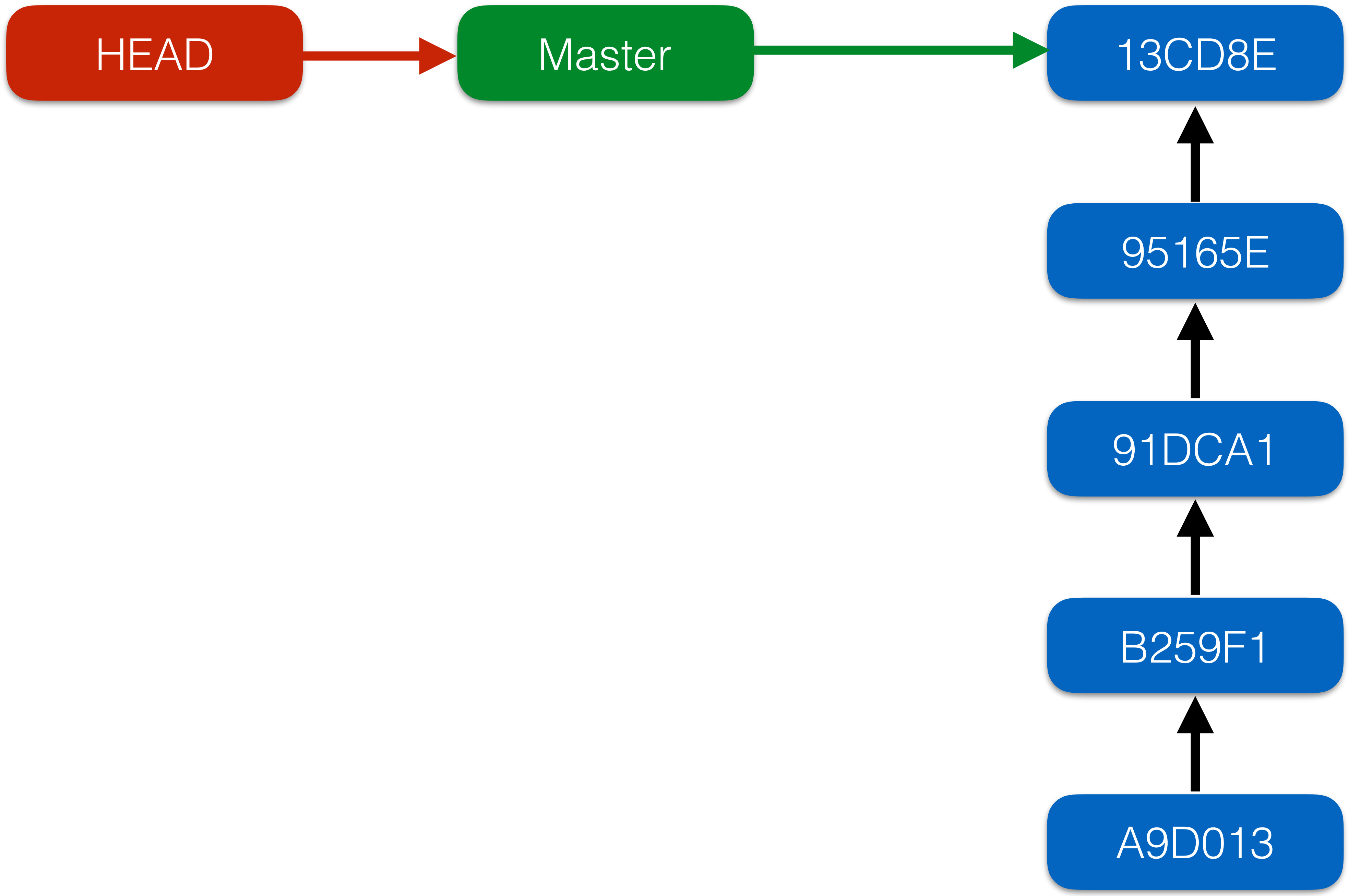
## Branches



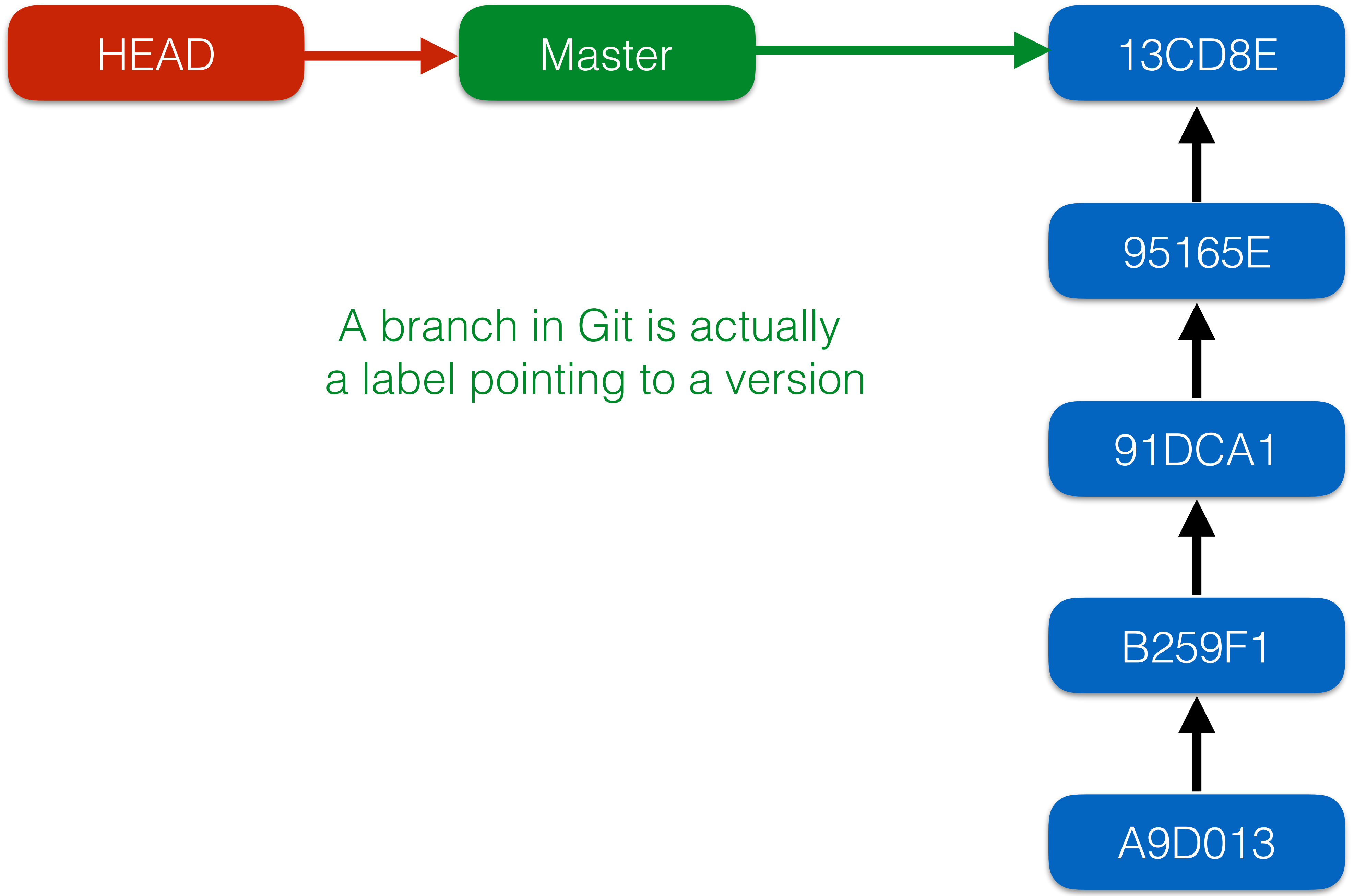
# The Master Branch



## The Master Branch

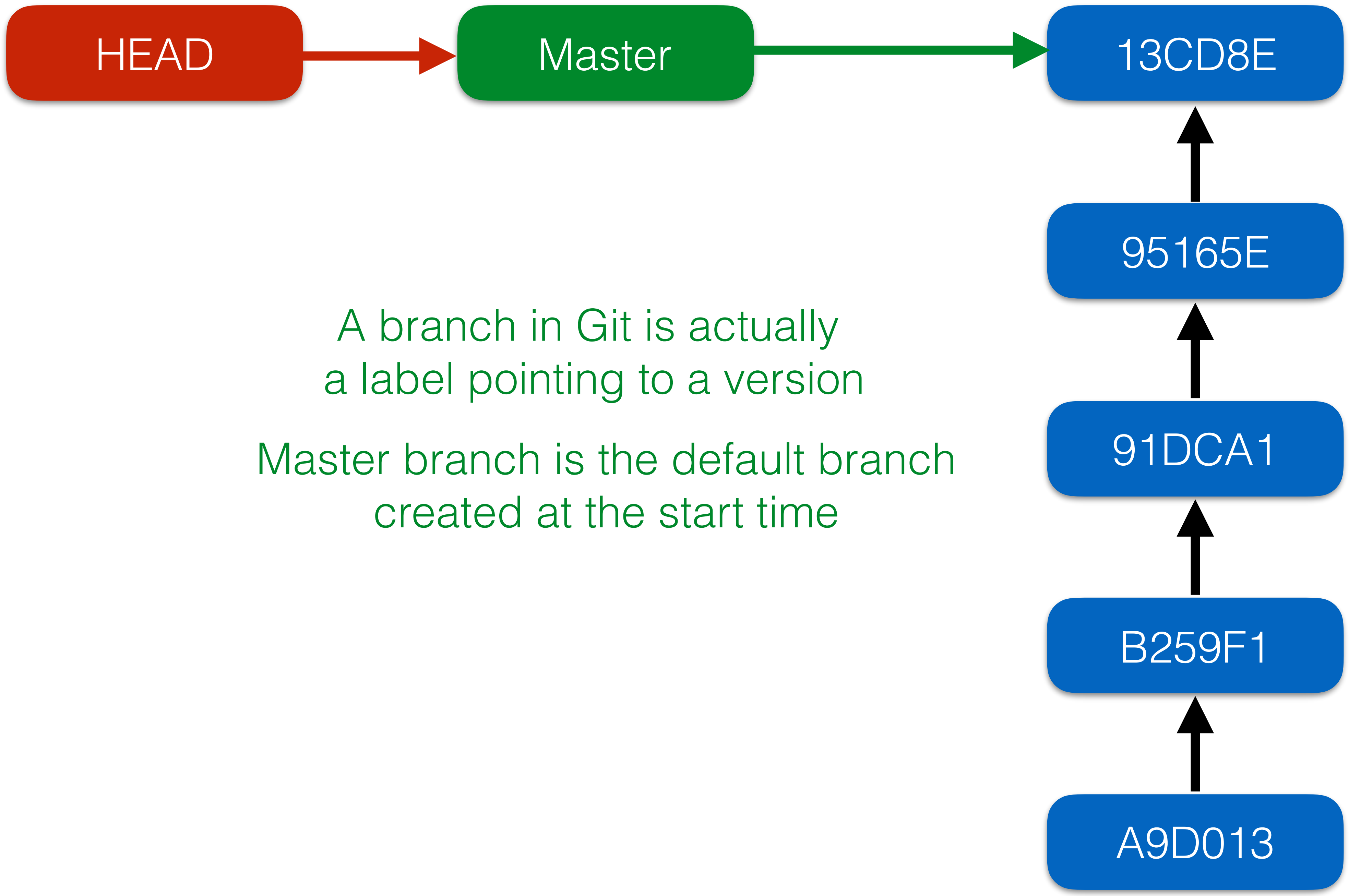


# The Master Branch



A branch in Git is actually
a label pointing to a version

# The Master Branch



A branch in Git is actually
a label pointing to a version

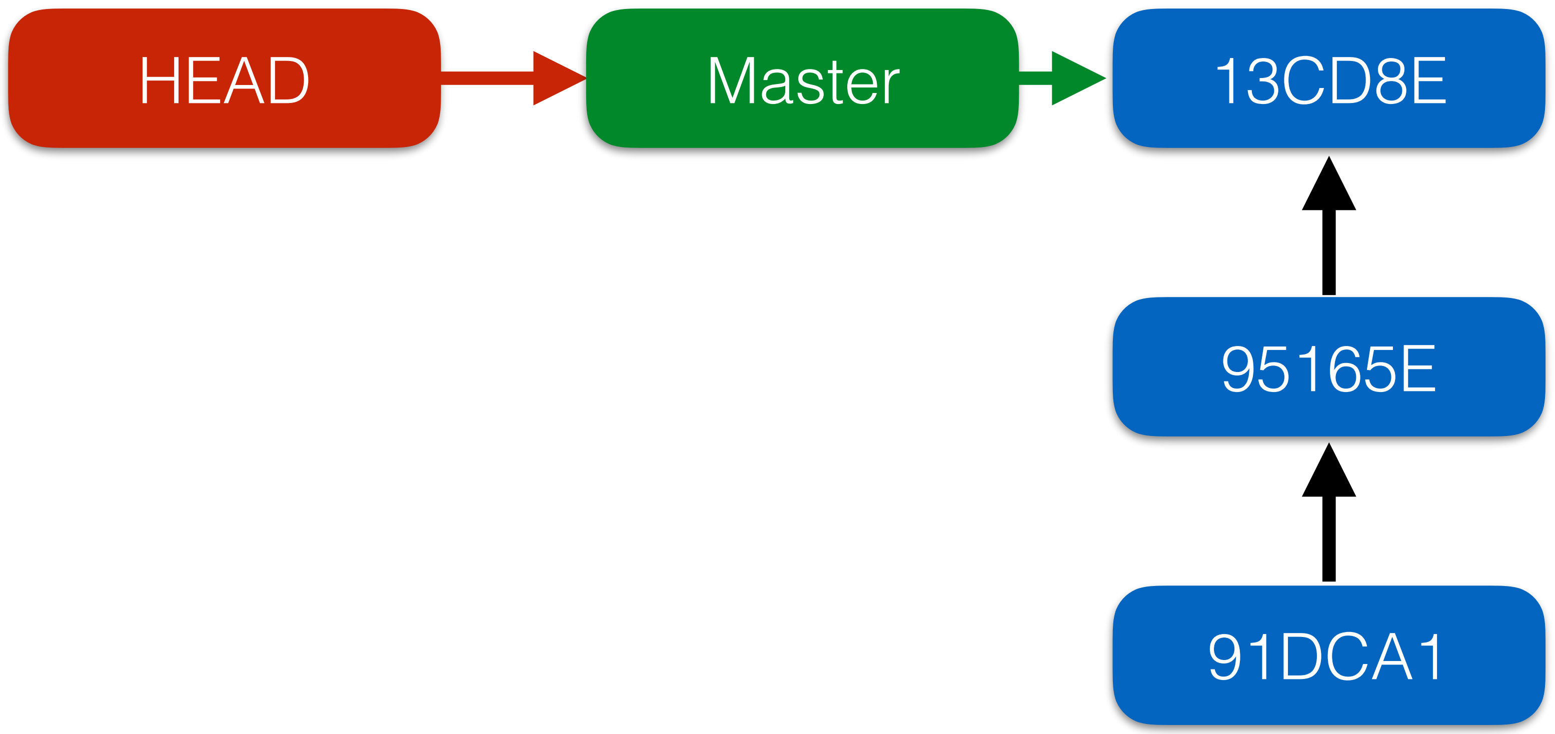
Master branch is the default branch
created at the start time

## Status

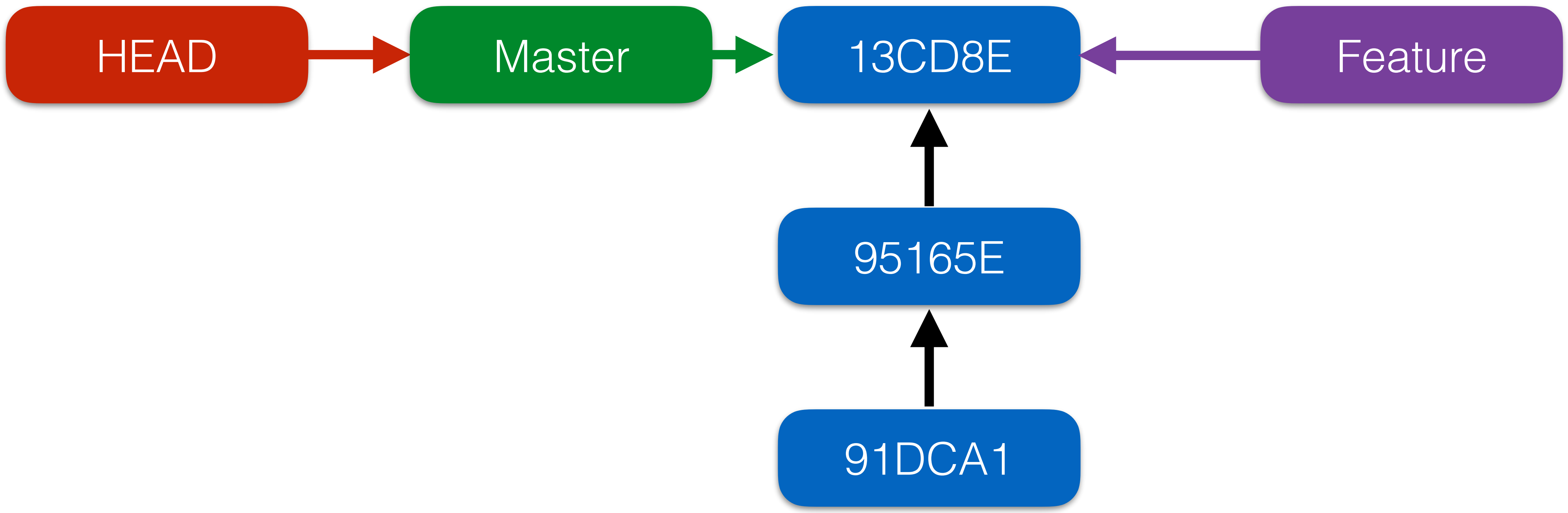
- Checking the current status and the current branch

```
git status
```

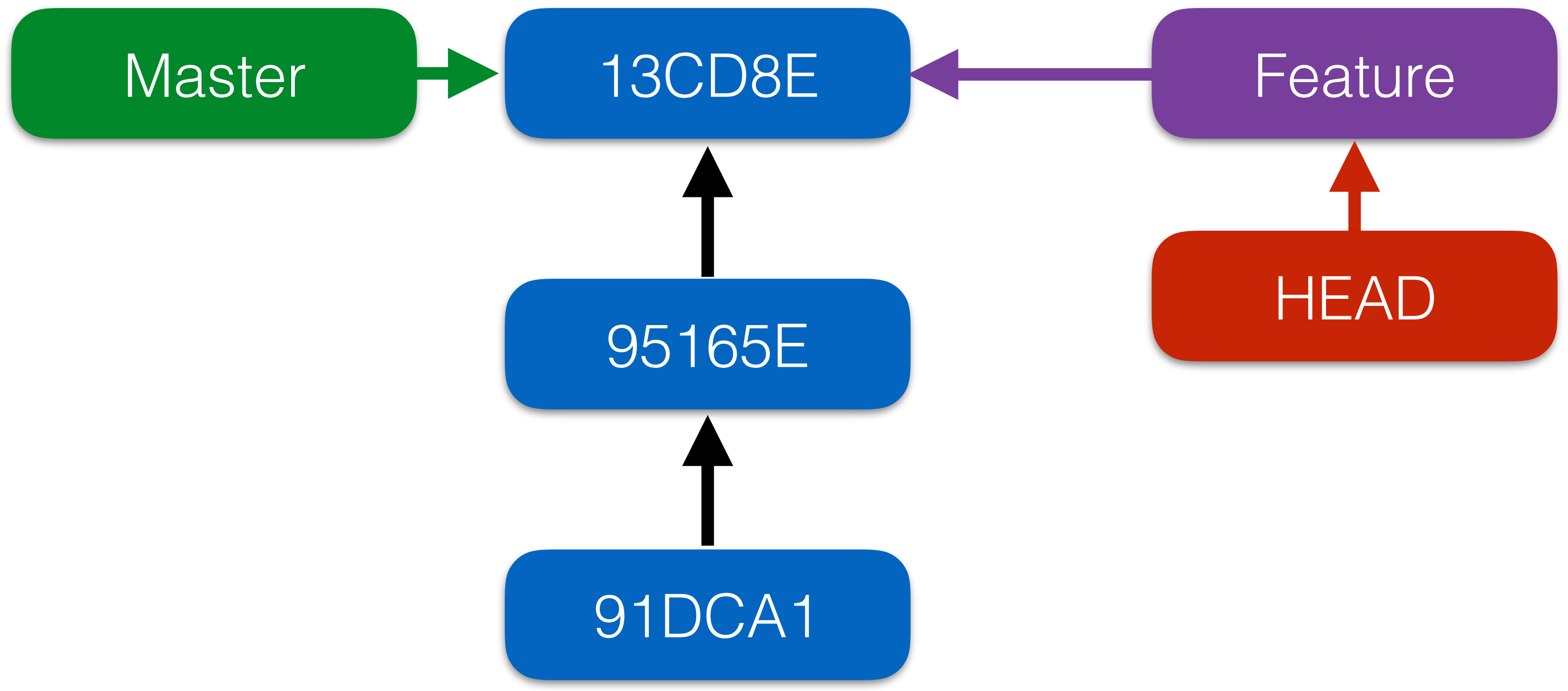
# Branching



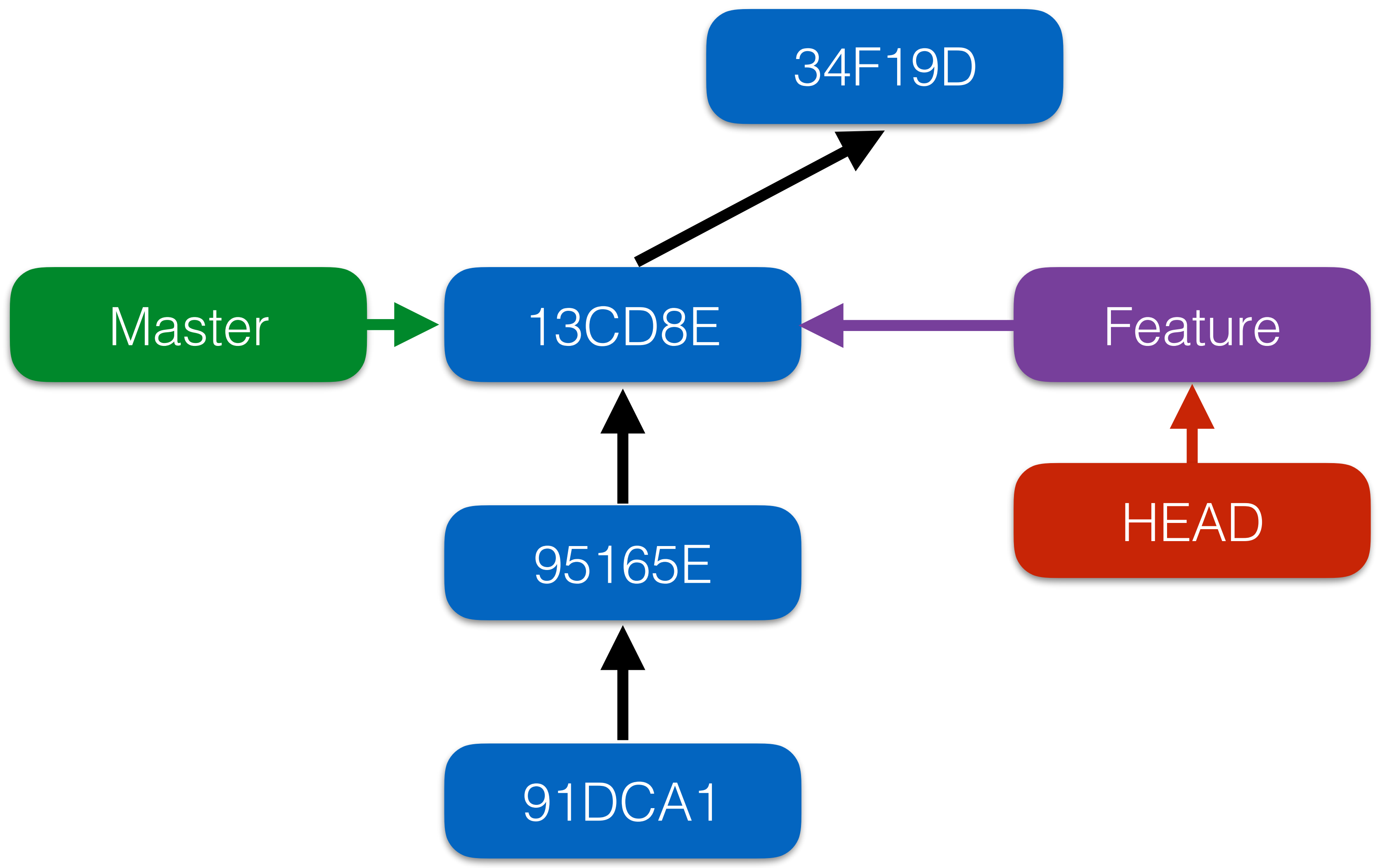
# Branching



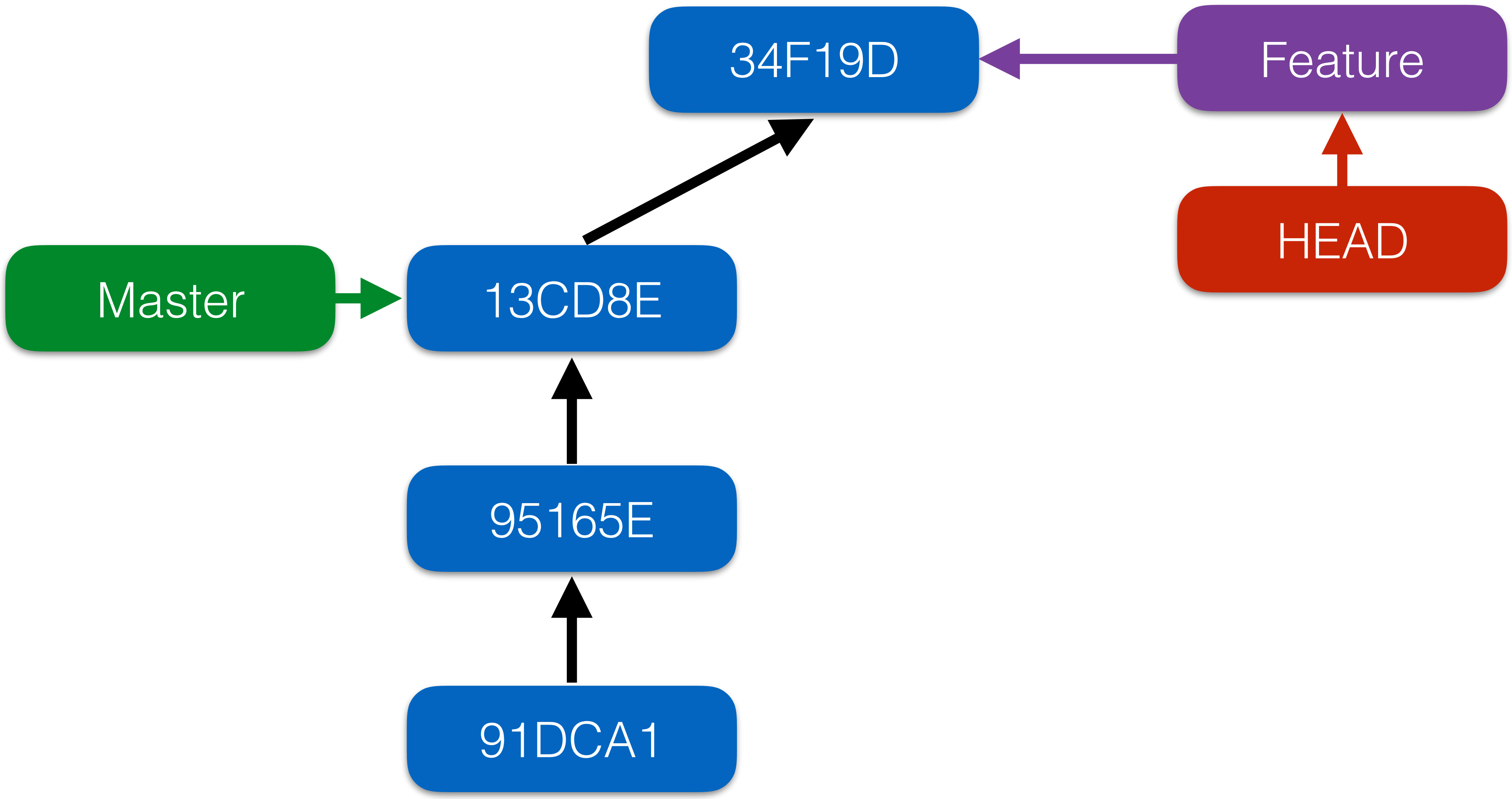
# Branching



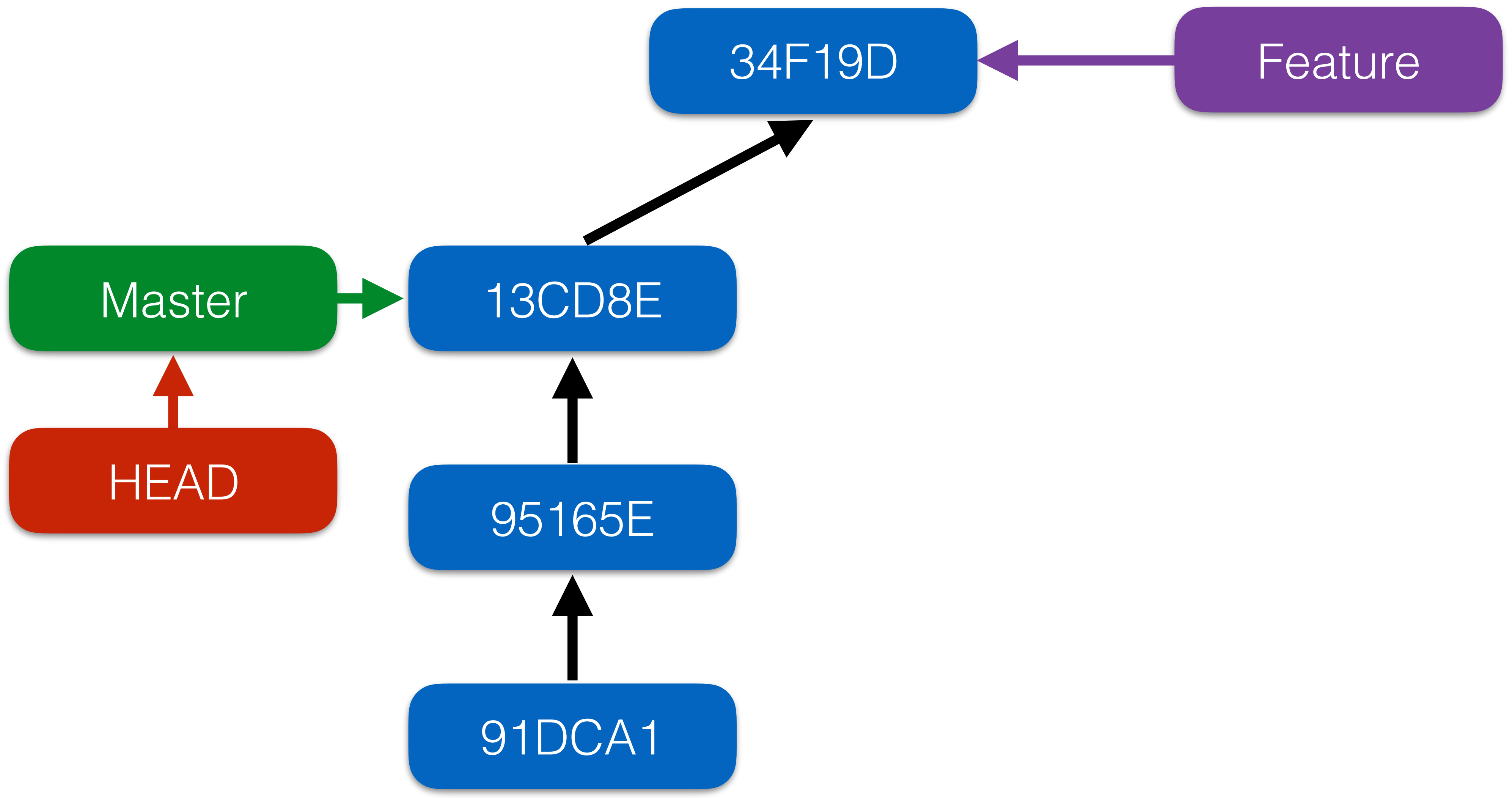
# Branching



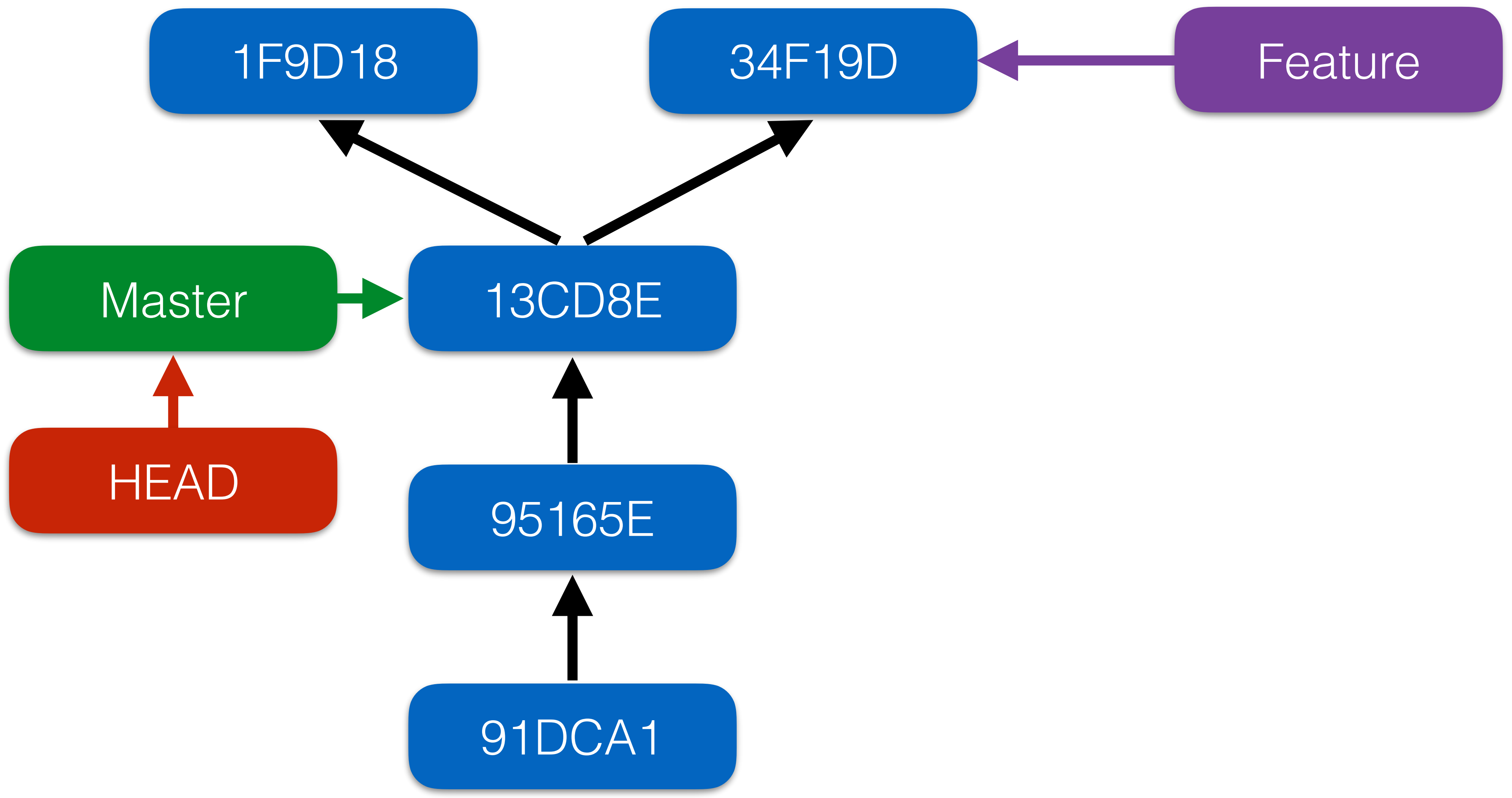
# Branching



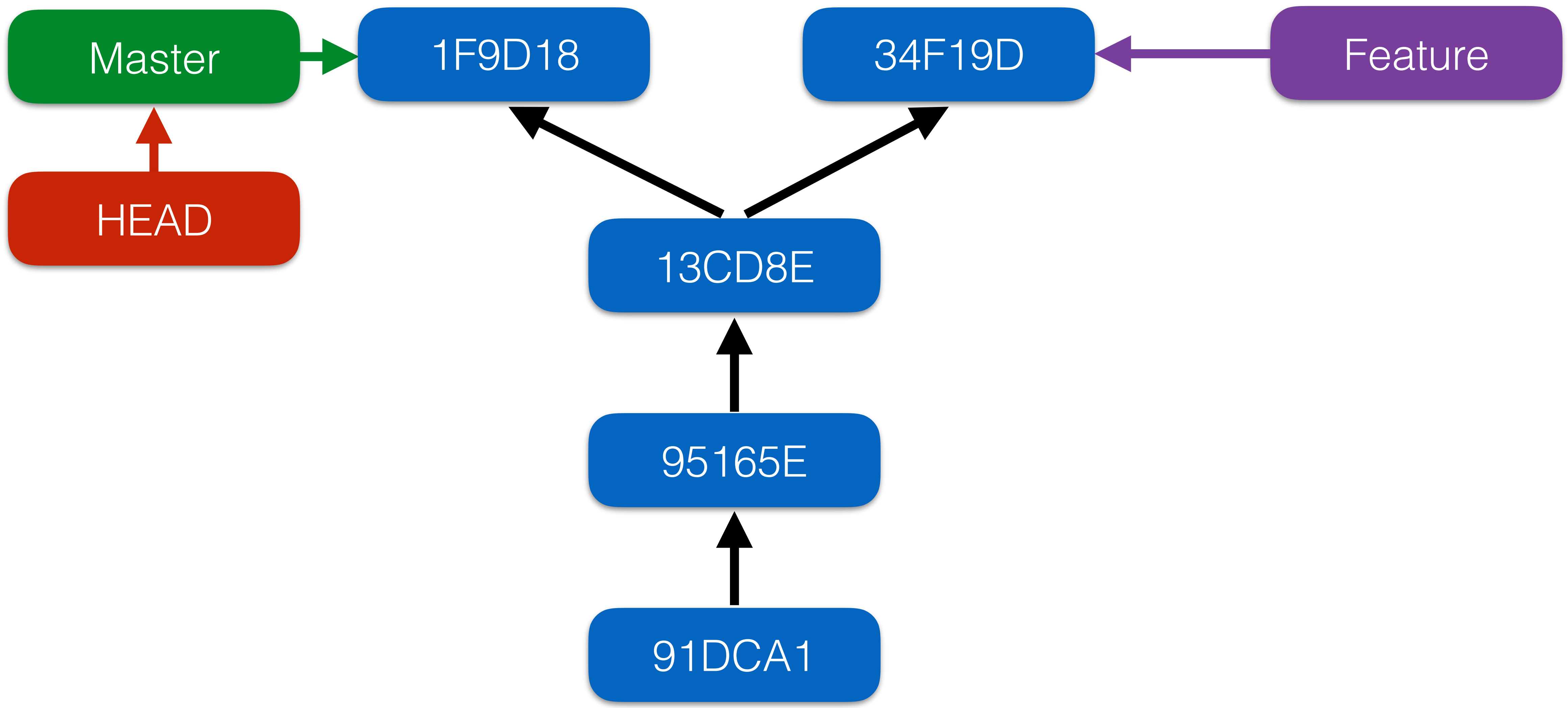
# Branching



# Branching



# Branching



# Git Branching

- Creating a new branch (label)

```
git branch [branch name]
```

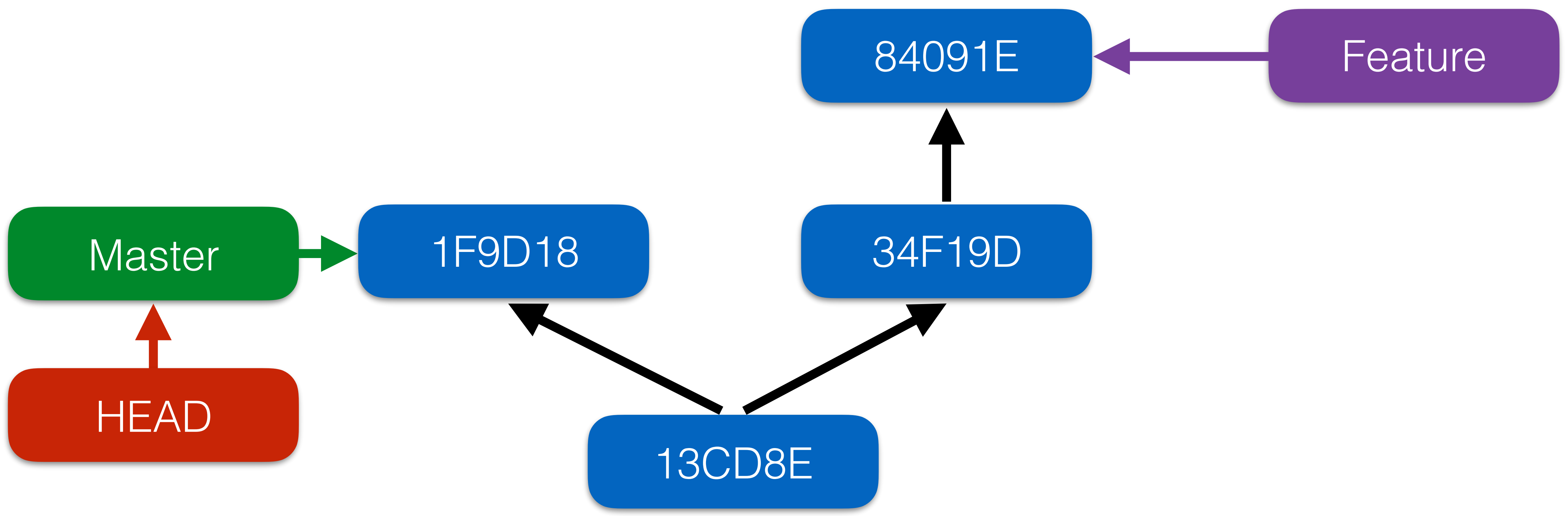
- Checking out the branch (move the HEAD)

```
git checkout [branch name]
```

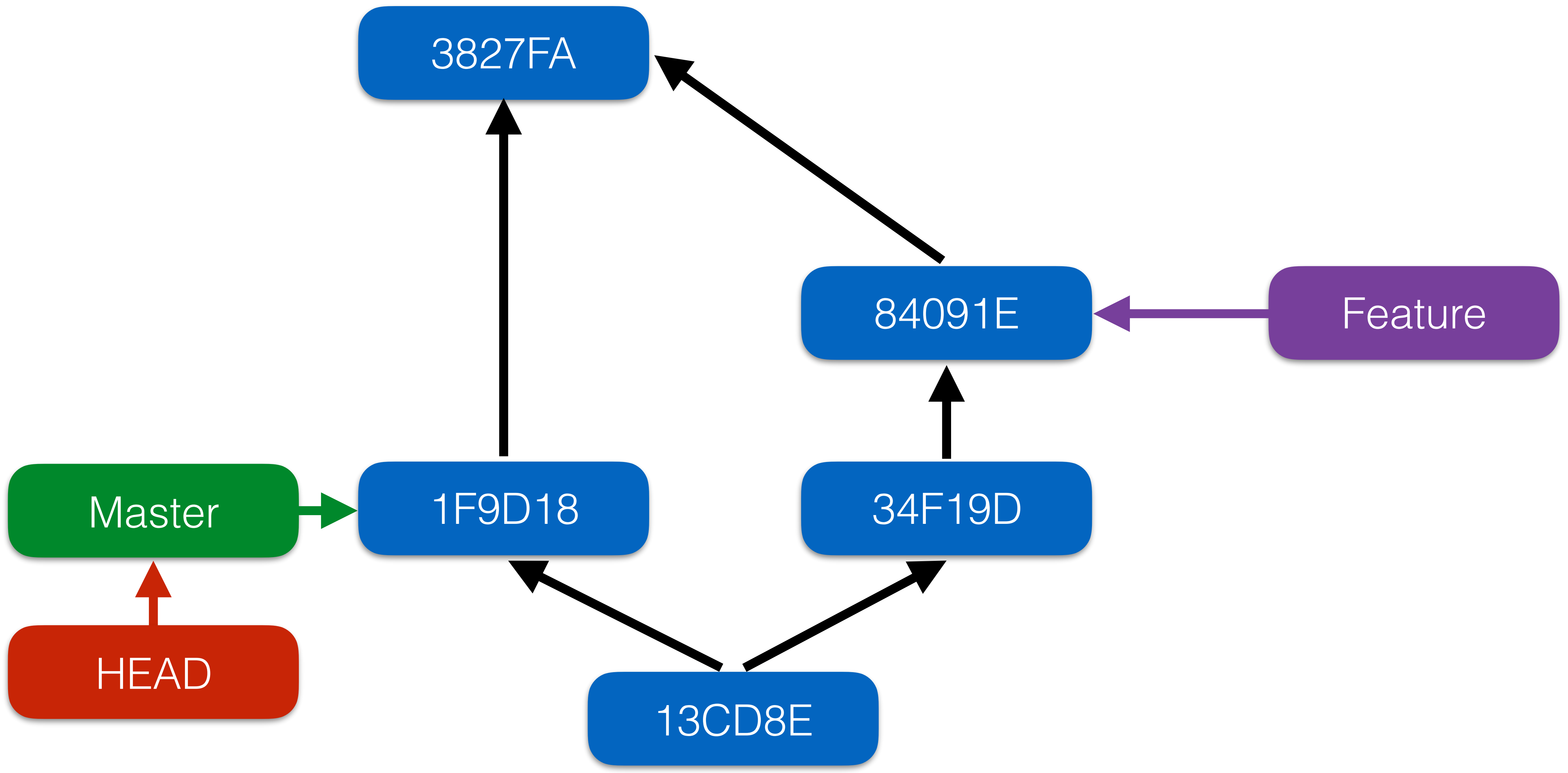
- Combining the above commands (create & checkout)

```
git checkout -b [branch name]
```

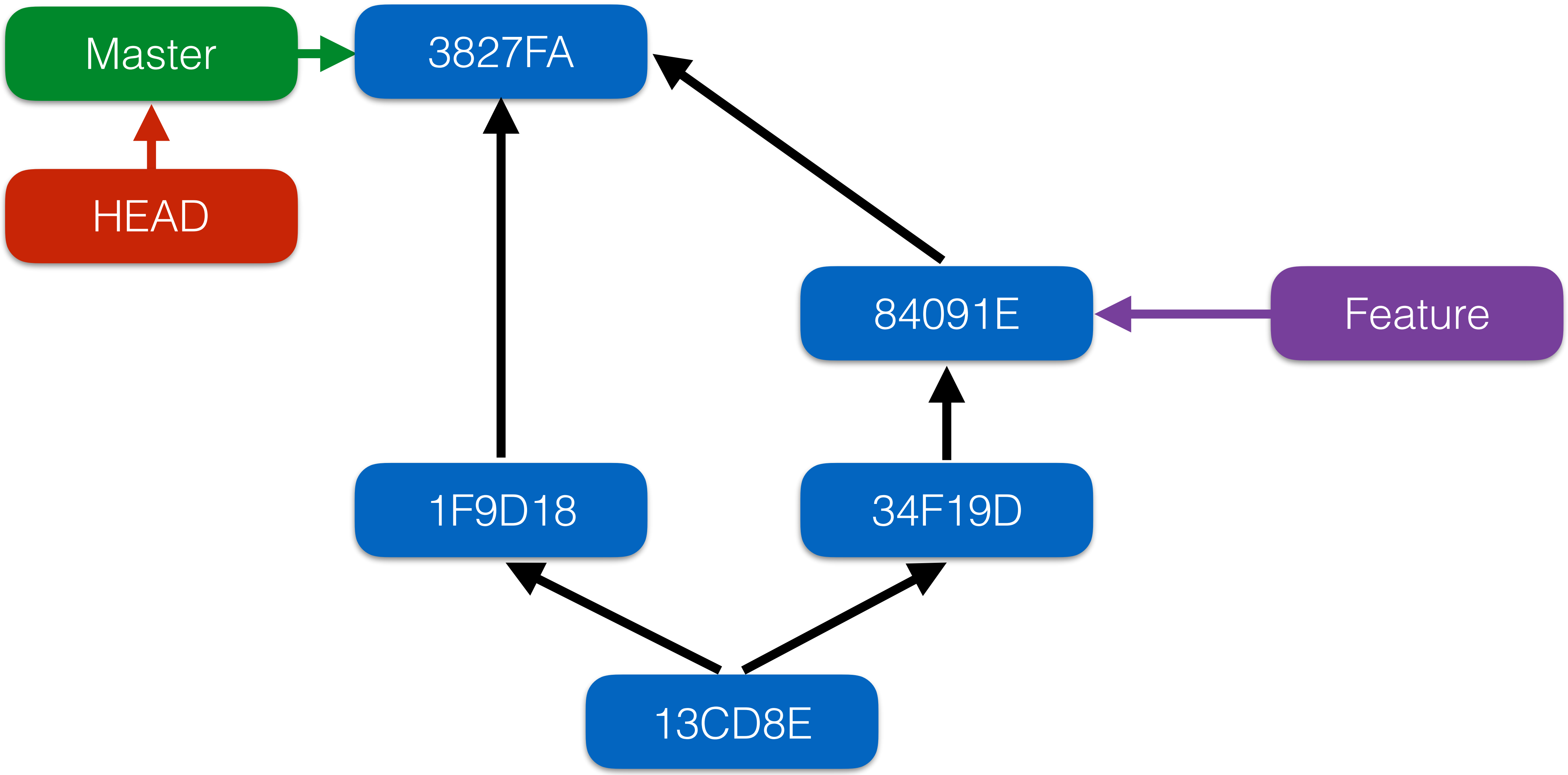
# Merging



# Merging



# Merging



# Git Merging

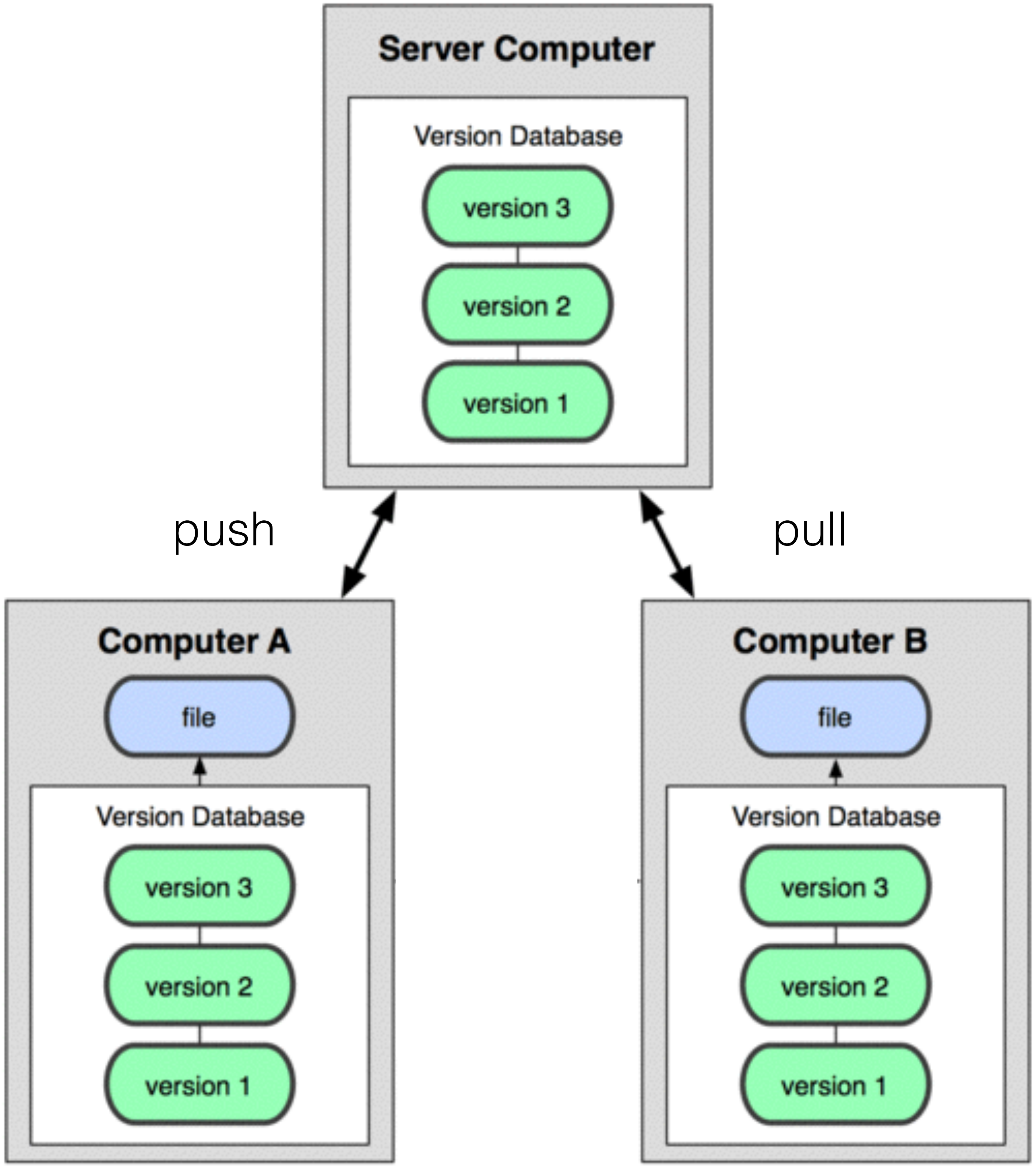
- Merging Steps
  - Checking out a branch to merge

    ```
    git checkout [branch 1 name]
    ```

  - Merging another branch

    ```
    git merge [branch 2 name]
    ```

## Remote Repositories


Server Computer
Version Database
version 3
version 2
version 1
push
pull
Computer A
file
Version Database
version 3
version 2
version 1
Computer B
file
Version Database
version 3
version 2
version 1

# Cloning & Pushing

- Cloning the remote repositories

```
git clone [Remote URL]
```

- The [Remote URL] is saved as **Origin**
  - After committing a few versions, you can push the branch back to **Origin**

  ```
  git push -u origin [Branch Name]
  ```

## Fetch & Pull

- Updating a branch from the remote repository
  - Fetching the remote repository to local

    ```
    git fetch origin
    ```

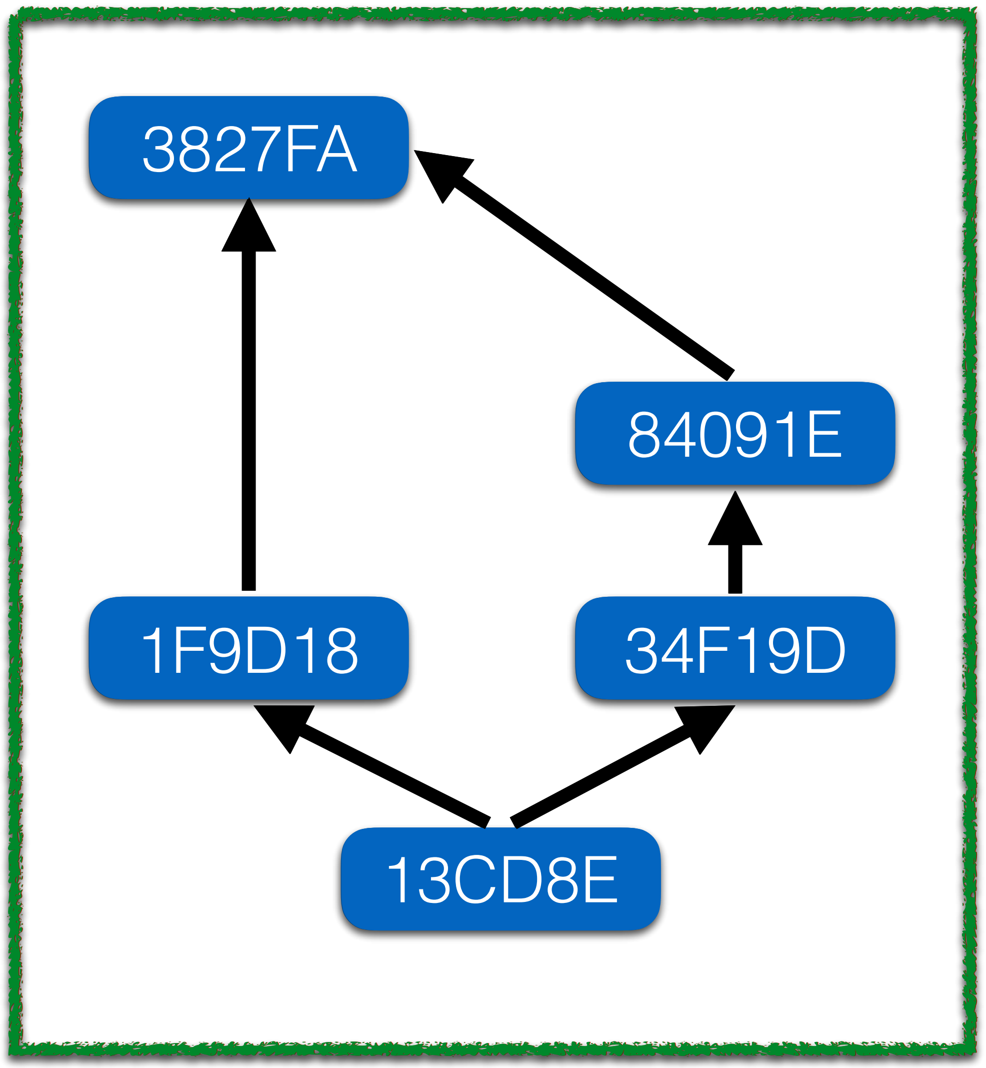
  - Merging the remote branch

    ```
    git merge origin/[Branch Name]
    ```

- Doing above commands in one command

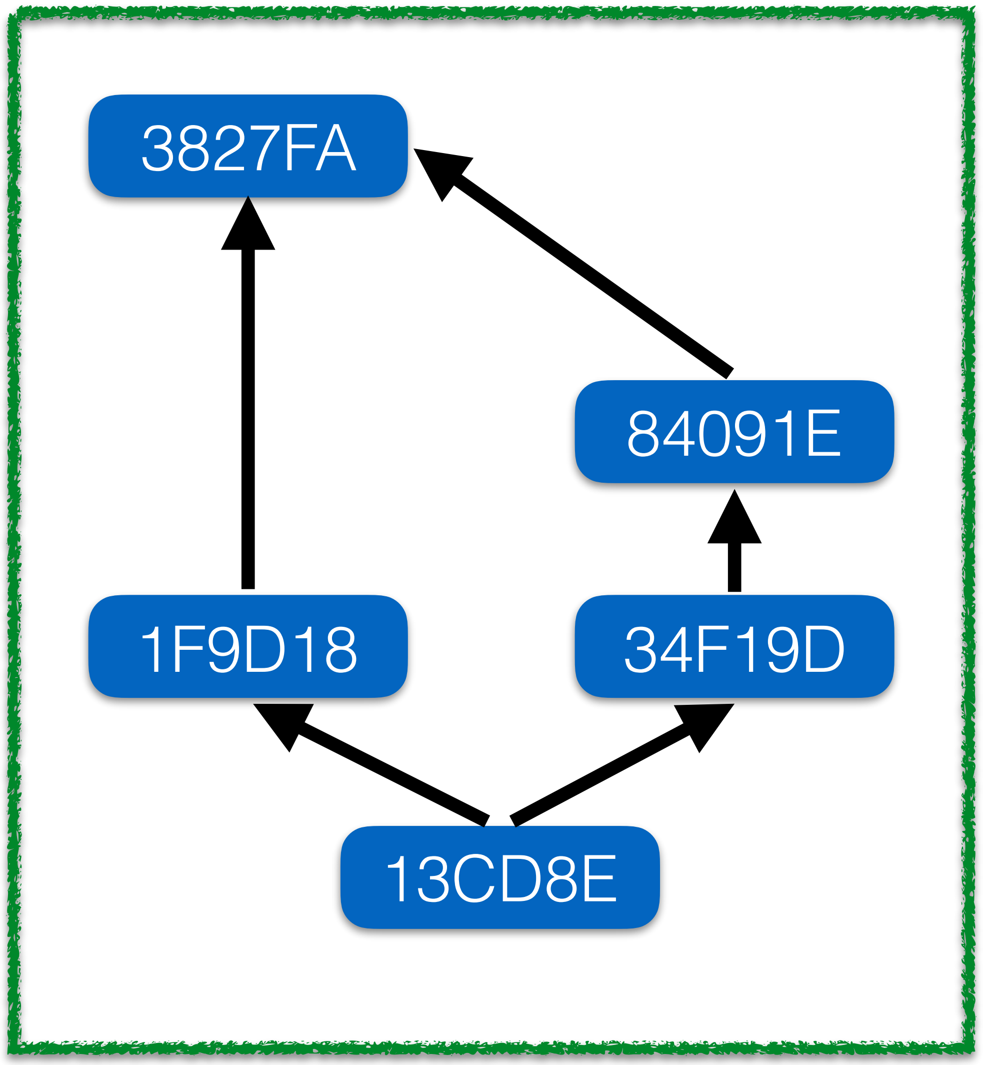
  ```
  git pull [Branch Name]
  ```

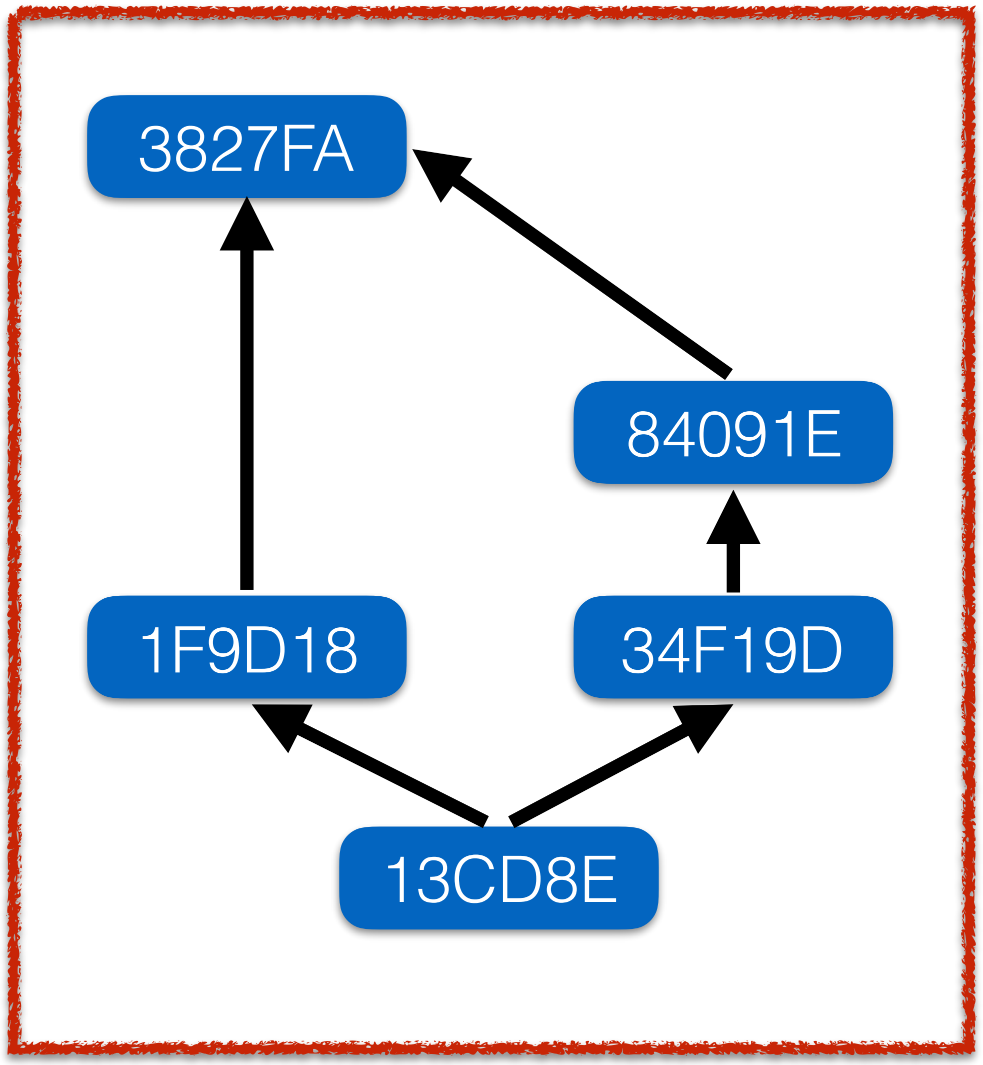
# Fork



The Repo. Under TA's Account

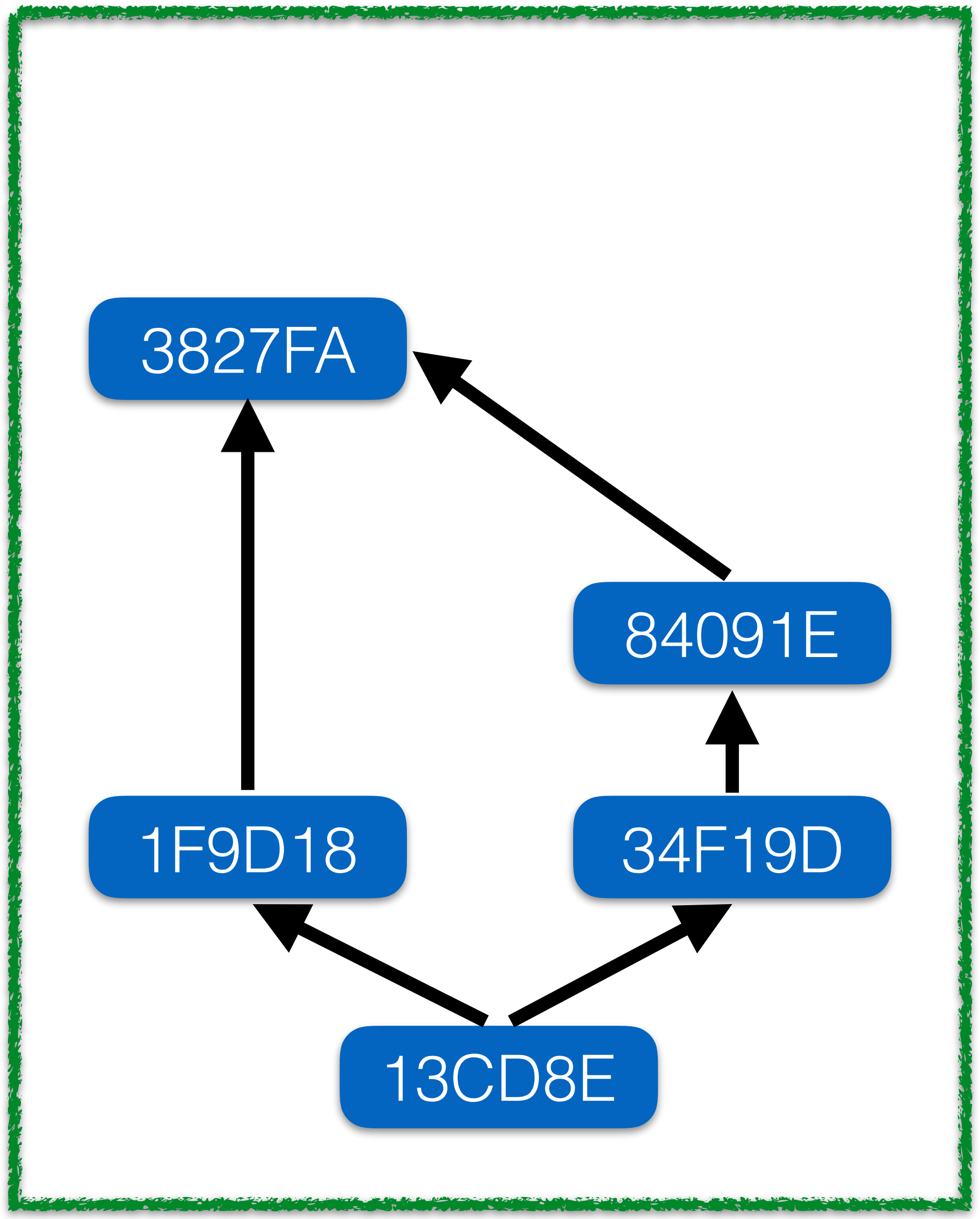
# Fork

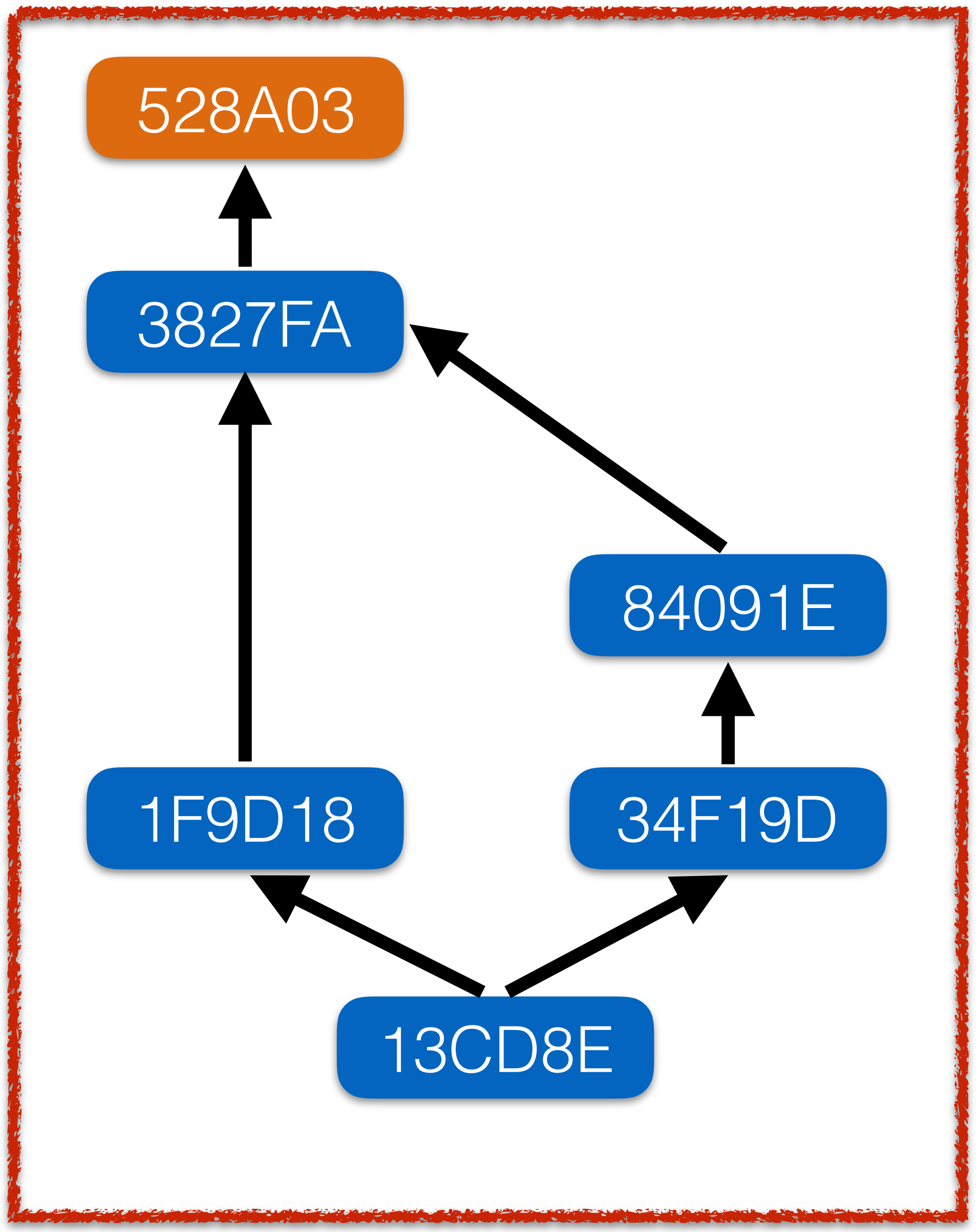




The Repo. Under TA's Account The Repo. Under Your Account

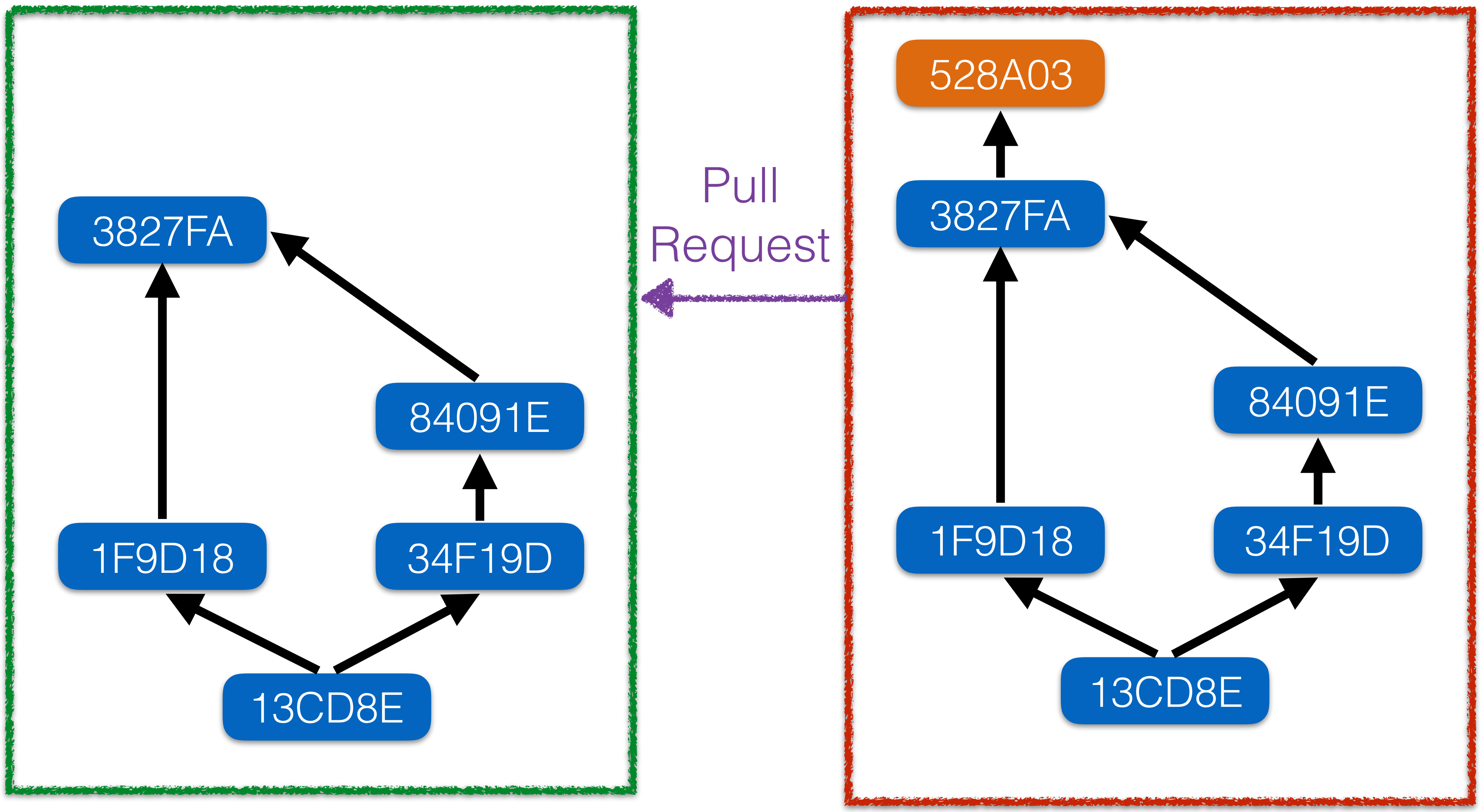
# Pull (Merge) Request





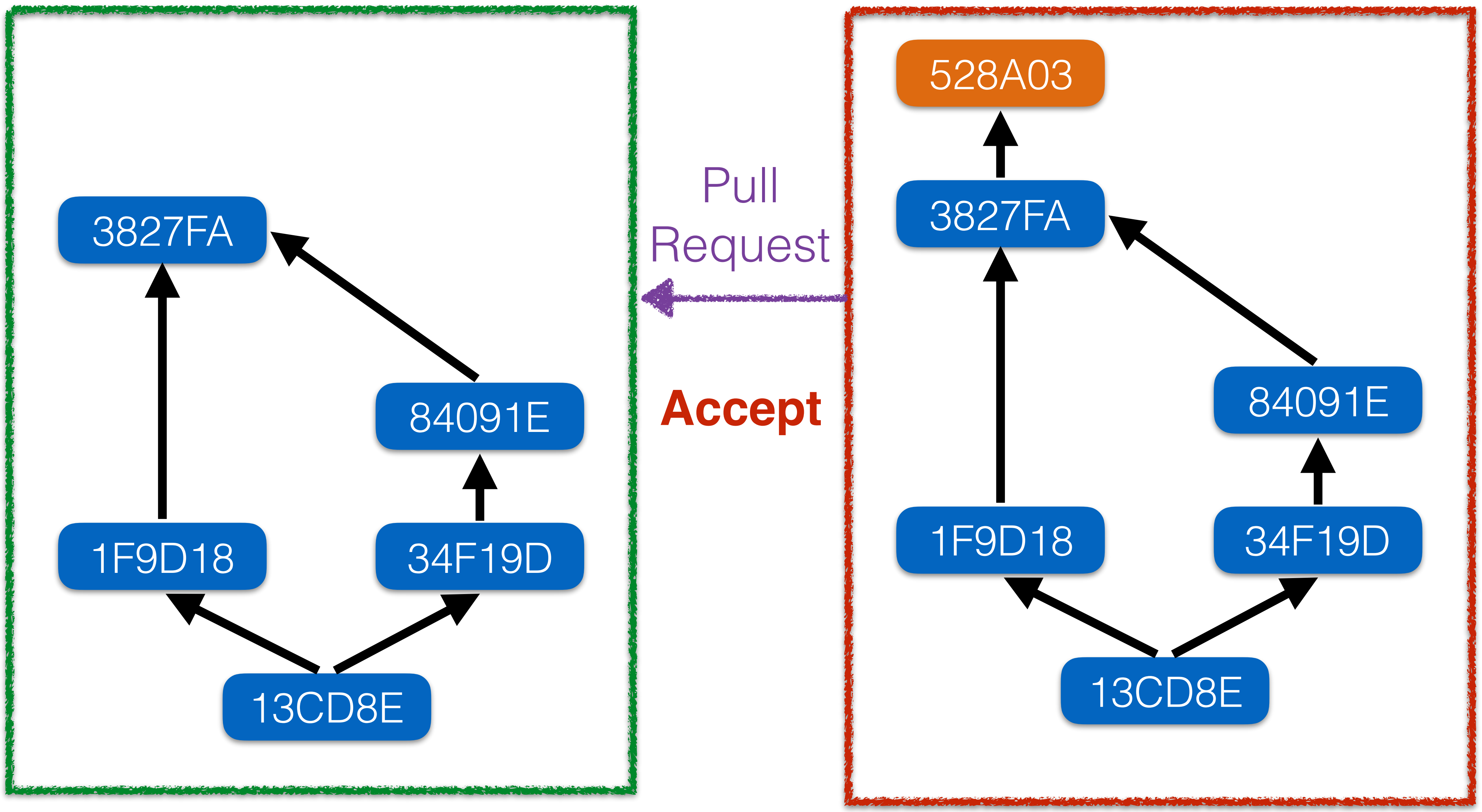
The Repo. Under TA's Account The Repo. Under Your Account

# Pull (Merge) Request



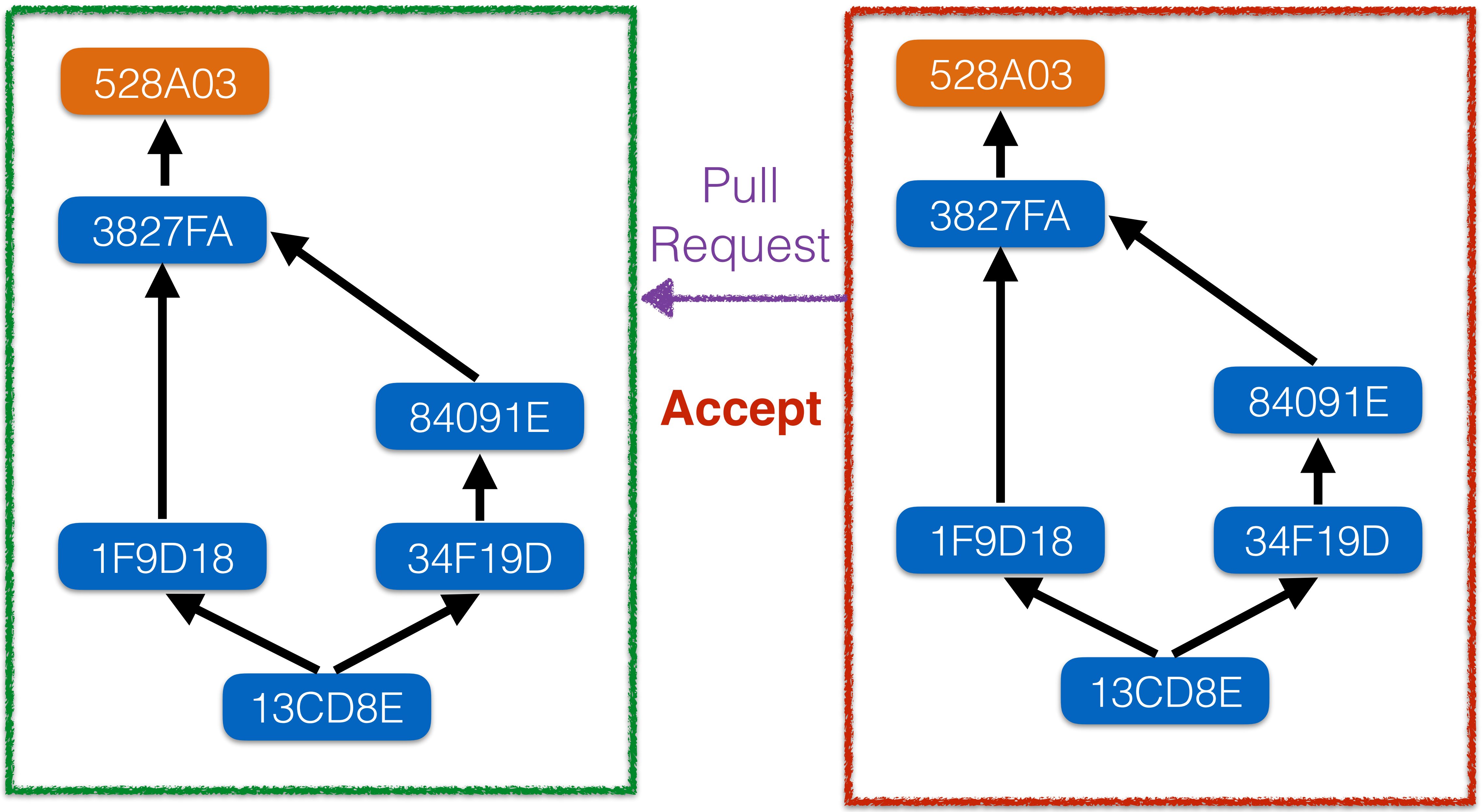
The Repo. Under TA's Account The Repo. Under Your Account

# Pull (Merge) Request



The Repo. Under TA's Account The Repo. Under Your Account

# Pull (Merge) Request



The Repo. Under TA's Account The Repo. Under Your Account

# .gitignore File

- You can ignore some files that you don’t want them to be tracked by editing the .gitignore file
- Remember to track and commit your .gitignore file
- Don’t know what should be in .gitignore ?
  - https://github.com/github/gitignore
  - https://www.gitignore.io/

axosoft

GitKraken

# Reference

- Learn Git branching (interactive)
    - http://pcottle.github.io/learnGitBranching/
- Pro Git
    - http://git-scm.com/book/
- 寫給大家的 Git 教學
    - http://www.slideshare.net/littlebtc/git-5528339