



# Introduction to Git for Version Control

This tutorial is aimed at Data Scientists and Statisticians

Presented by: Pavan Datta



# Outline of Topics

- 1. What is Git? Why use Git?**
- 2. Definition of Terms**
- 3. Common and Useful Commands**
- 4. Step-by-Step Walkthrough with Git**
- 5. Live Tutorial + Setup Git/GitHub**
- 6. References**

# 1. What is Git? Why use Git?

- A open source distributed version control system (DVCS)
  - Source code is available on a **remote repository** (GitHub / GitLab)
  - Source code can be **cloned** onto numerous **local repositories**
- Lightweight, fast, and flexible
  - Most operations are performed **locally**
  - Git only stores **differences** between **commits**
  - Easily context switch between **branches**
  - Easily manage changes to files across large teams
  - Git keeps track of **every** change ever made
- Multiple backups are supported
- Many different workflow styles supported

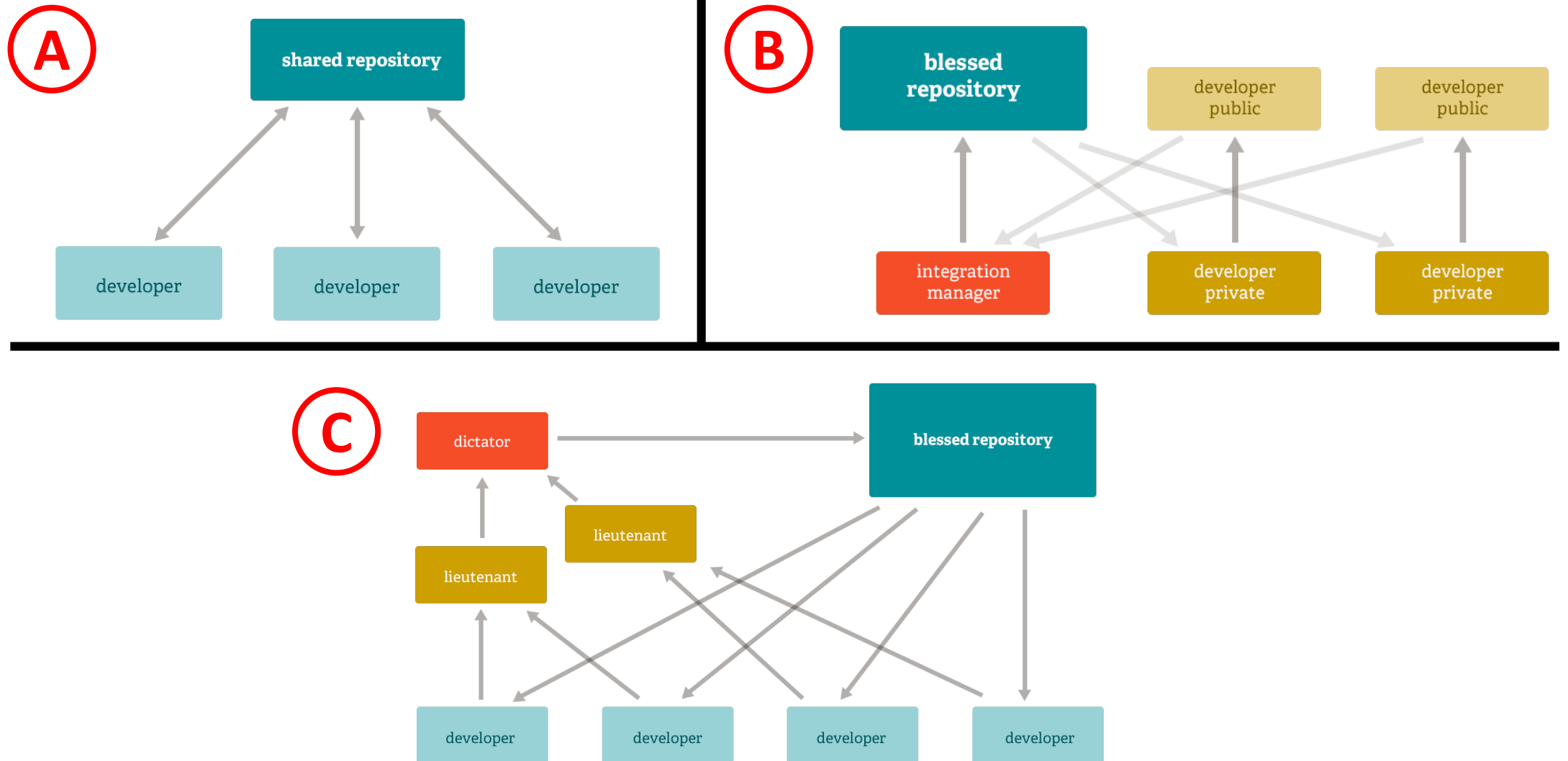
# 1. What is GitHub? Why use GitHub?

- GitHub is like Facebook, but for hosting code
  - With the right permissions, code sharing and collaboration are possible
  - Many R-packages are now available via GitHub
- GitHub makes code collaborating easier than before
- Supports many different workflow styles
- Let's check it out!

# 1. Flexible Workflow Styles Supported by Git

- A** Subversion (SVN) style workflow
- B** Integration Manager Workflow
- C** Dictator and Lieutenants Workflow

# 1. Flexible Workflow Styles Supported by Git



## 2. Definition of Terms

- **Repository** – directory structure under which all files are stored for a project
  - **Local Repository** – the repository hosted locally on your computer system
    - All project contributors make their changes (**commits**) in their own **local repositories**
  - **Remote Repository** – the repository hosted remotely on GitHub / GitLab
    - All project contributors **push** their **local changes** (**commits**) to the **remote repository**
    - All project contributors **pull** changes (**commits**) from the **remote repository** into their **local repositories**
- **Branches** – the different workflows or paths in a **repository**
  - Each **repository** has *at least* one **branch** (the default branch is called **master**)
  - Branches get sync'd between **remote** and **local repositories**
  - Branches can be **created**, **deleted**, and **merged**
  - Branches can be used as a *playground* for your changes so that they do not interfere with the **master** branch
- **Clone** – the process of creating a **local copy** of a **remote repository**

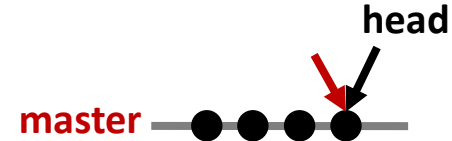
## 2. Definition of Terms

- **Commit** – a change (or snapshot in time)
  - Each **branch** consists of **commits**
  - Each **commit** has a unique SHA or hash (40 character checksum) by which it is identified
  - A commit considers only **diffs** or **deltas** from the previous commit
- **Checkout** – the process of selecting a specific **branch** or **commit** (snapshot)
- **Pull** – the process of **fetching** commits from the **remote repository** and **merging** the differences into the **local repository**
  - **Fetch** – get/copy commits from the remote repository, into the local repository
  - **Merge** – account for all differences between the local repository and remote repository
- **Push** – send changes from the **local repository** to the **remote repository**
  - This is frequently preceded by a **pull**
  - Permissions in the remote repository may prevent you from performing a **push** and may instead require a formal **pull-request** (e.g. merge-request) through GitHub / GitLab



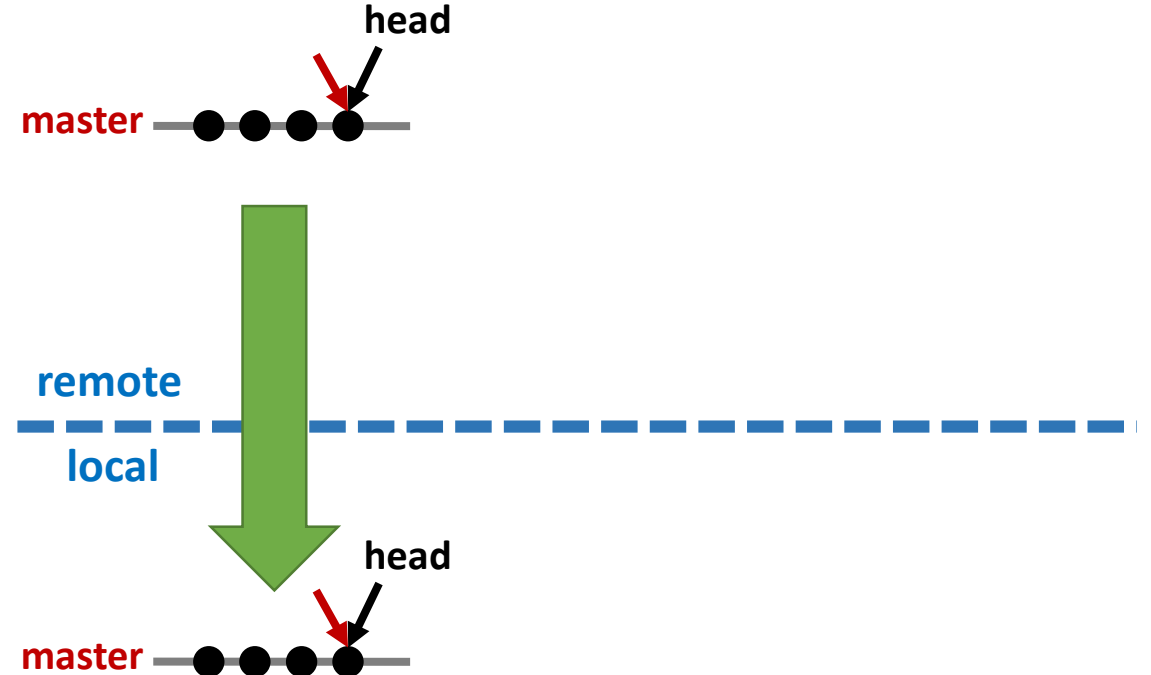
## 2. Definition of Terms (Visually)

- Consider a **remote repository** with one **branch**
  - Dots represent **commits**
  - Small arrows represent **pointers to commits**



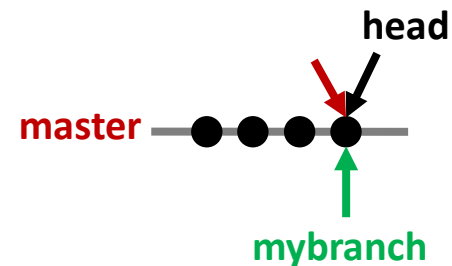
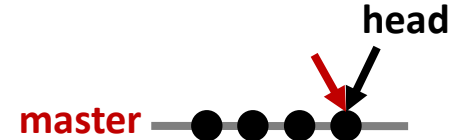
## 2. Definition of Terms (Visually)

- Consider a **remote repository** with one **branch**
  - Dots represent **commits**
  - Small arrows represent **pointers to commits**
- **Clone** the remote repository locally



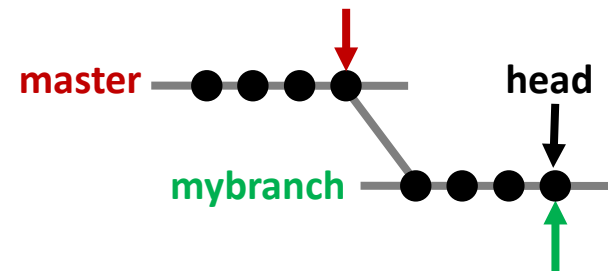
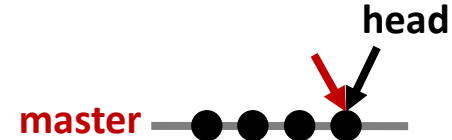
## 2. Definition of Terms (Visually)

- Consider a **remote repository** with one **branch**
  - Dots represent **commits**
  - Small arrows represent **pointers to commits**
- **Clone** the remote repository locally
- Create a new **branch** in the **local repository**



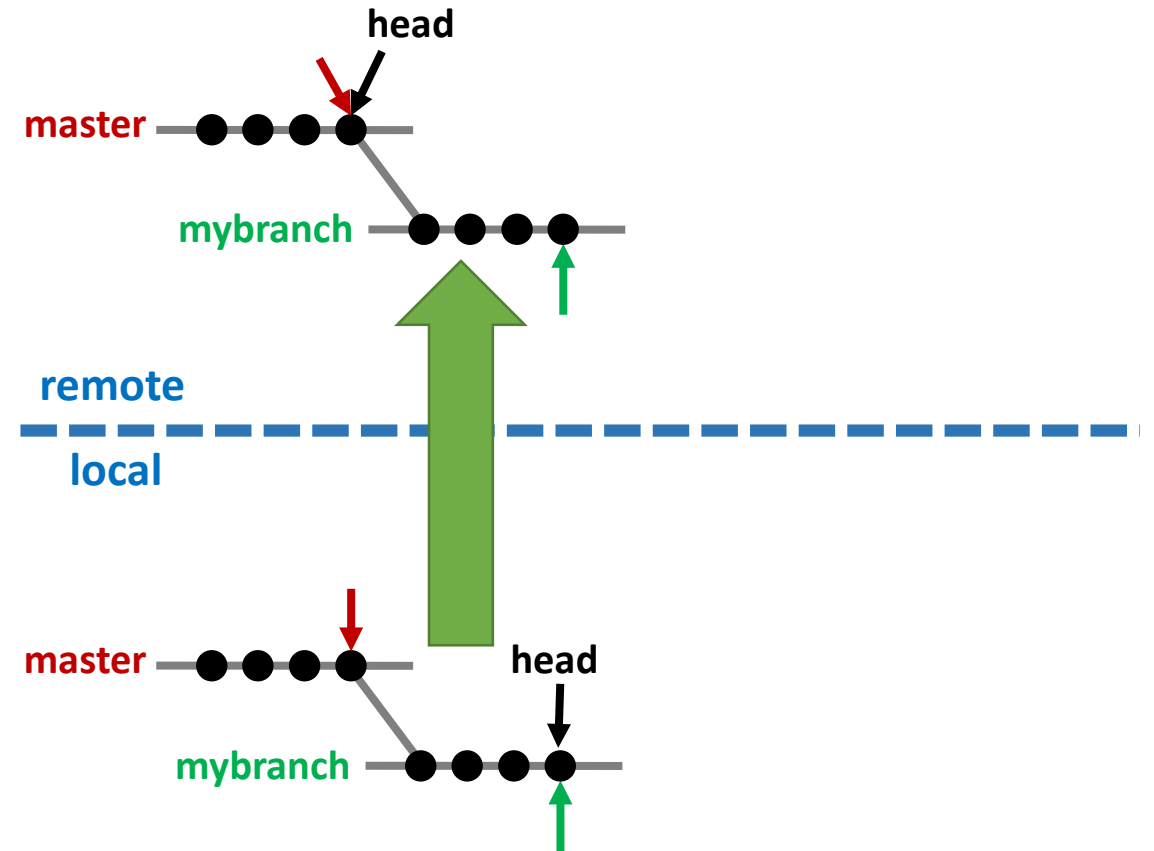
## 2. Definition of Terms (Visually)

- **Clone** the remote repository locally
- Create a new **branch** in the **local repository**
- Make **commits** to the **new branch**
  - **Head** is a pointer the current commit in your workspace



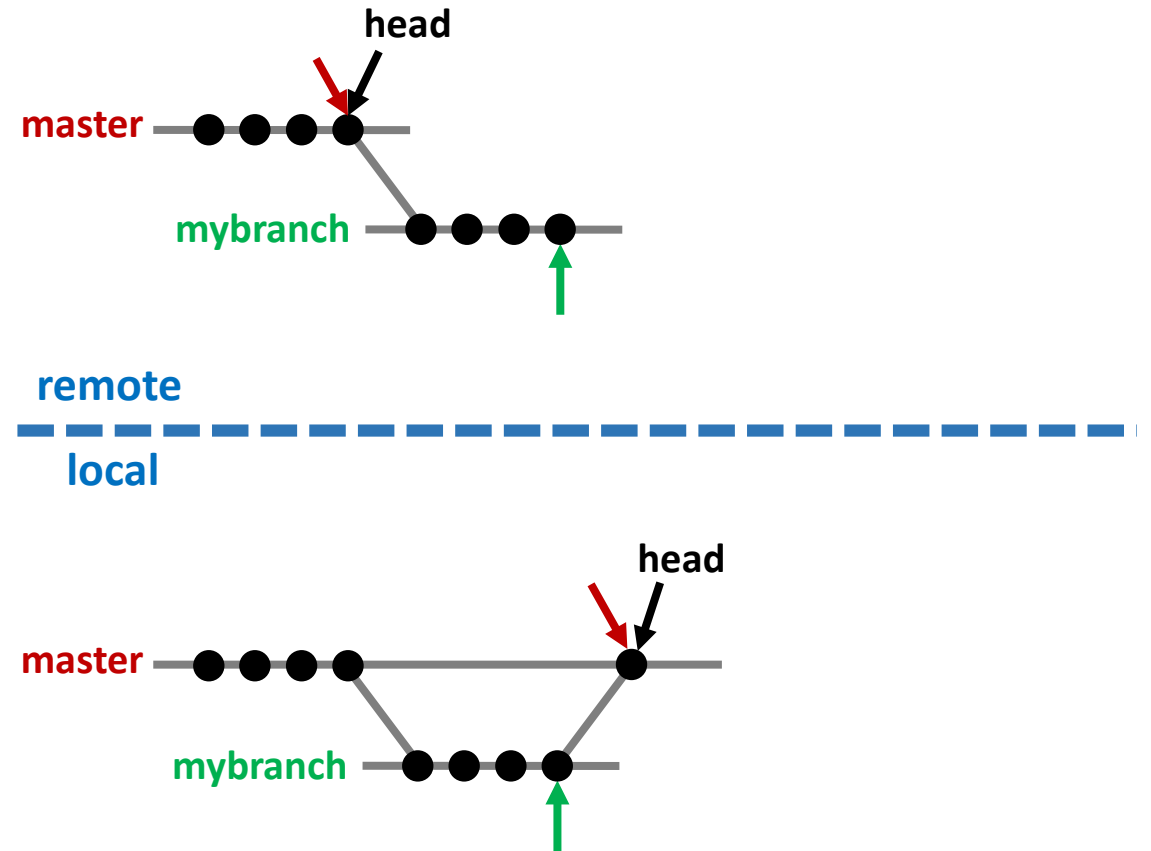
## 2. Definition of Terms (Visually)

- Create a new **branch** in the **local repository**
- Make **commits** to the **new branch**
  - **Head** is a pointer the current commit in your workspace
- **Push** all commits associated with the **new branch** to the **remote repository**



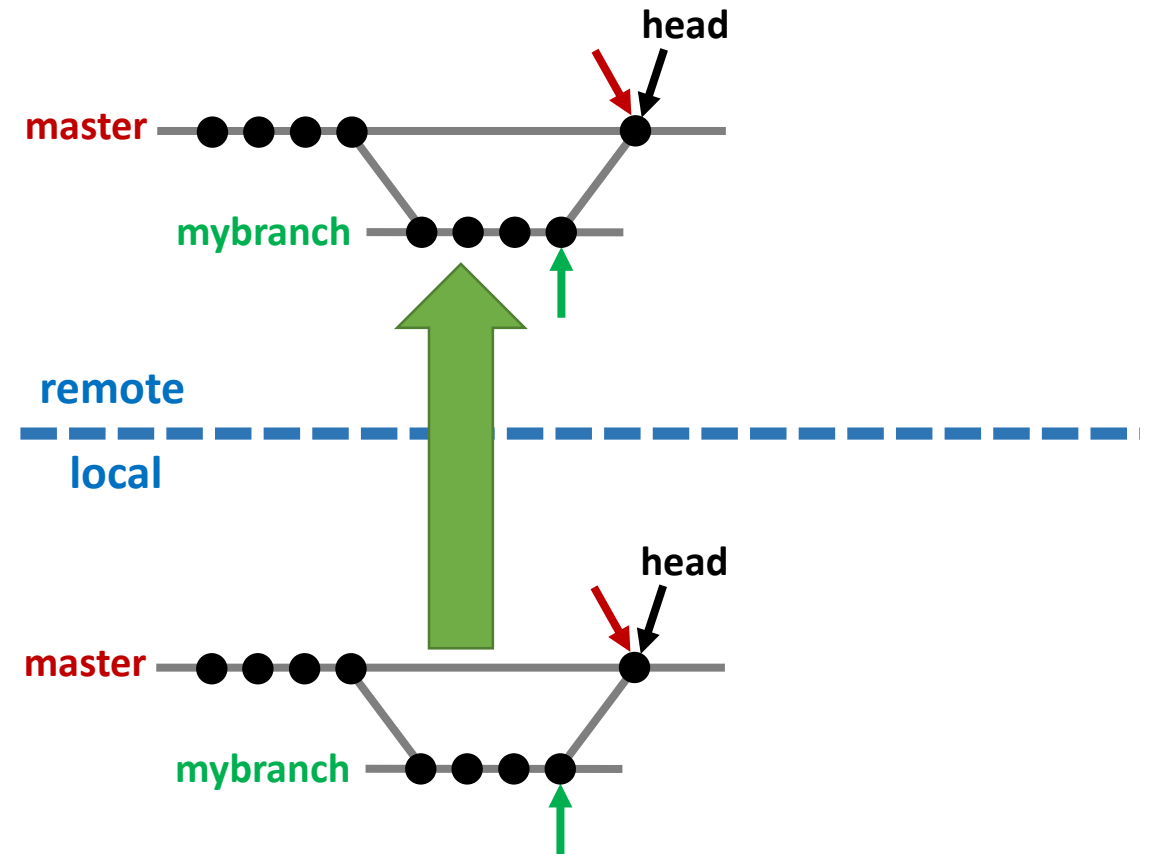
## 2. Definition of Terms (Visually)

- Make **commits** to the **new branch**
  - **Head** is a pointer the current commit in your workspace
- **Push** all commits associated with the **new branch** to the **remote repository**
- **Merge** the **new** branch into the **master** branch



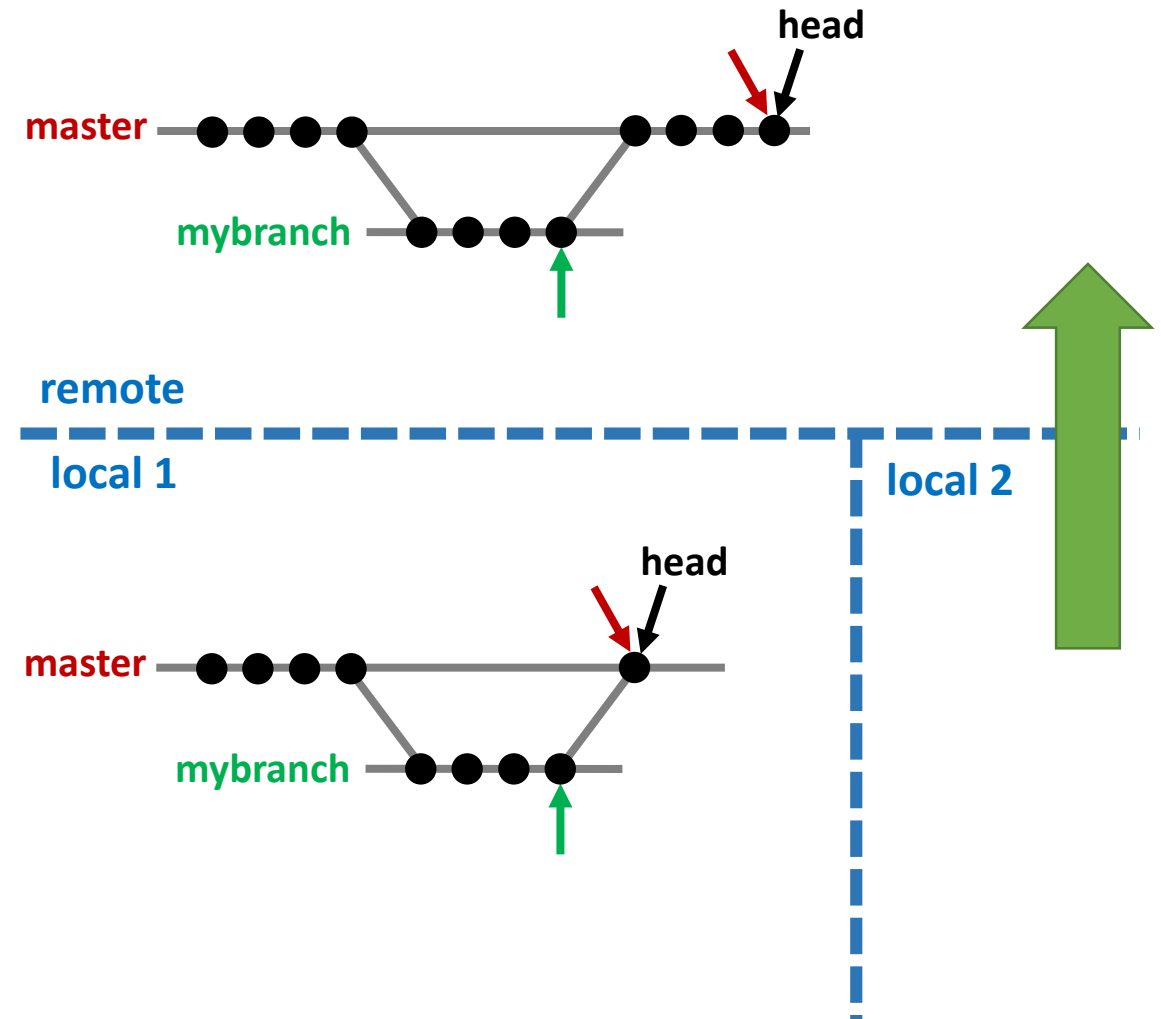
## 2. Definition of Terms (Visually)

- **Push** all commits associated with the **new branch** to the **remote repository**
- **Merge** the **new** branch into the **master** branch
- **Push** all changes to the remote repository



## 2. Definition of Terms (Visually)

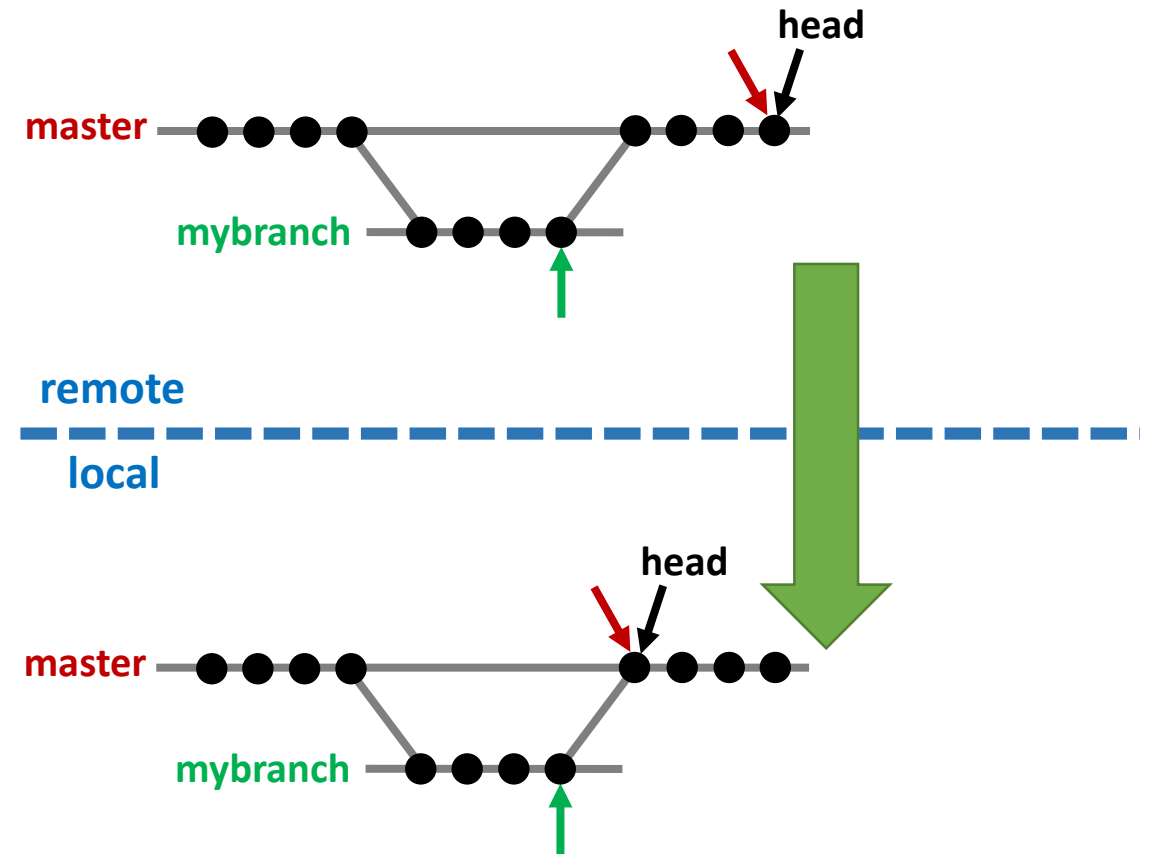
- Merge the **new** branch into the **master** branch
- **Push** all changes to the remote repository
- Suppose another developer **pushes** changes to the **remote repository**





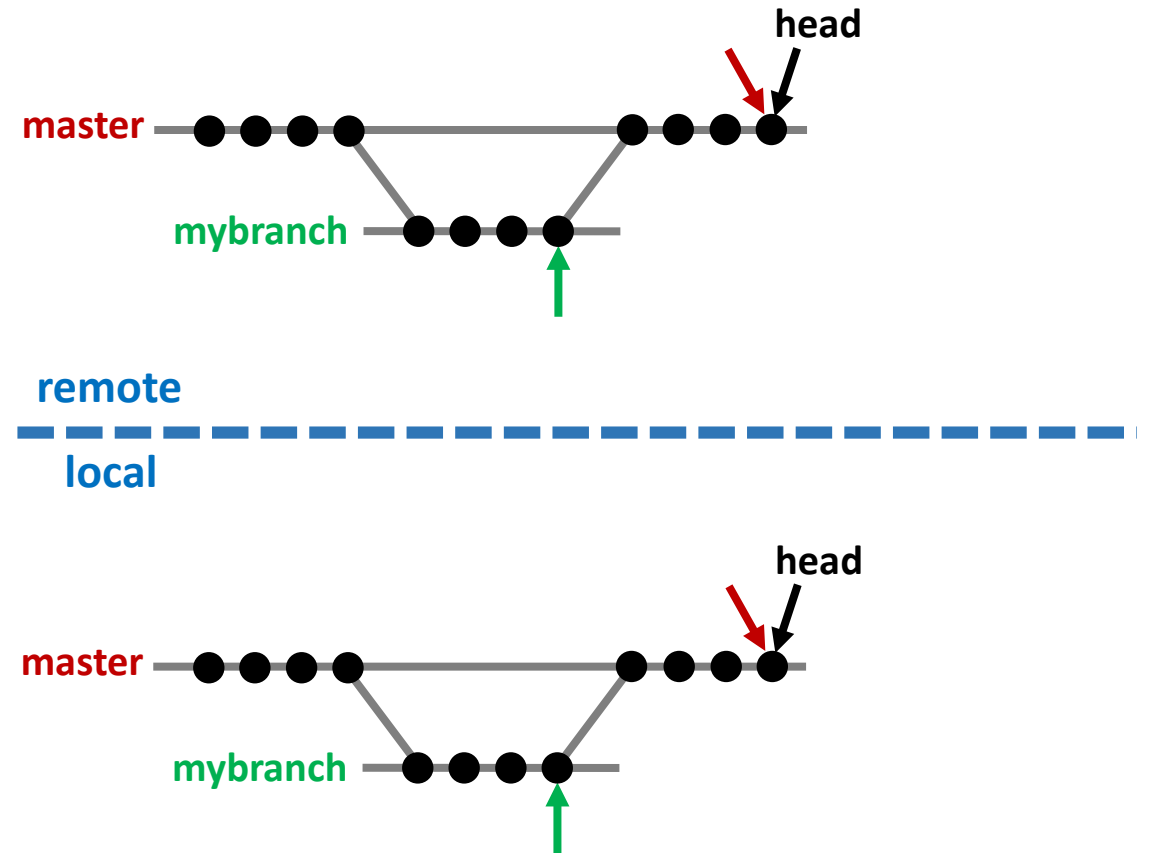
## 2. Definition of Terms (Visually)

- **Push** all changes to the remote repository
- Suppose another developer **pushes** changes to the **remote repository**
- **Fetch** the new commits from the **remote repository**

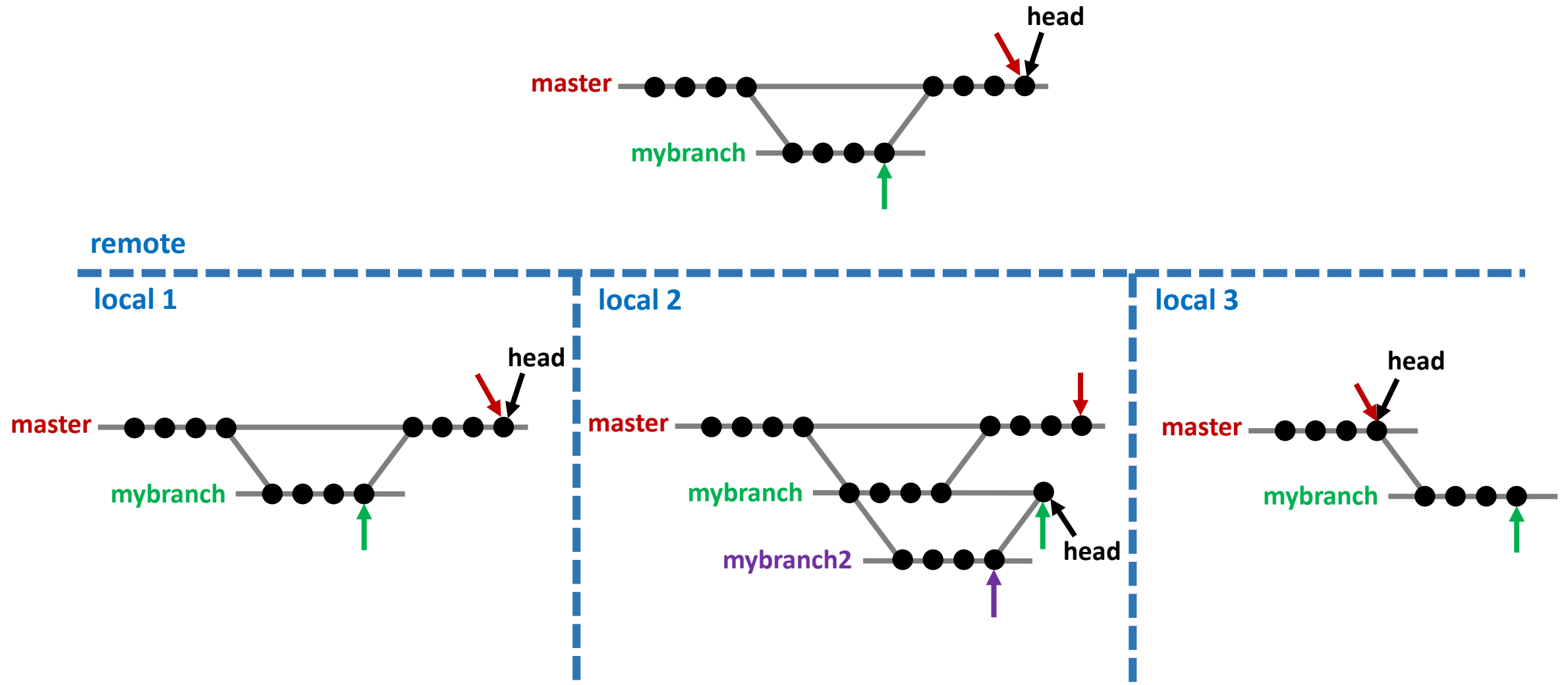


## 2. Definition of Terms (Visually)

- Suppose another developer **pushes** changes to the **remote repository**
- **Fetch** the new commits from the **remote repository**
- **Merge** the differences in your local repository



## 2. Snapshot of Git Repositories



# 3. Navigating with Git

- The process of promoting code/changes is as follows:
  1. Source code changes made locally in **workspace** (Git is unaware)
  2. **Stage** the files that changed (e.g. tell Git which files you want to commit)
  3. **Commit** the files that changed to your **local repository** (Git is aware of changes)
  4. **Push** changes from **local repository** to **remote repository**
- All Git operations may be performed by **command-line** or using conveniently designed **user-interfaces** for your OS
- This tutorial relies exclusively on the **command-line** approach
  - Used by most people in the user community
  - Well documented and easy to get help

### 3. Common and Useful Commands

> `git clone <repo-location>:<user>/<repo-name>.git`

- Create a **clone** of the **remote repository** locally

> `git remote -v`

- Check the **remote repository** and its **alias**

> `git checkout <branch-name | commit-SHA>`

- Look at a particular branch or commit (sets the HEAD pointer)

> `git checkout -b <branch-name>`

- Create a new branch and switch to it (sets the HEAD pointer to the new branch)

### 3. Common and Useful Commands

- > `git pull <remote-repo> <branch>`
  - **Fetch** commits and **merge** differences from a **branch** in the **remote repository**
  - **NOTE:** Only state the **remote repository** and **branch name** if it isn't already set
- > `git push <remote-repo> <branch>`
  - **Push** commits to the **remote repository** and its respective branch
  - **NOTE:** Only state the **remote repository** and **branch name** if it isn't already set
- > `git push -u <remote-repo> <branch>`
  - **Push** a newly created branch to the **remote repository**
- > `git log --oneline -n10`
  - View the last 10 commits, with each commit taking up a single line

# 3. Common and Useful Commands

## > `git status`

- Check the status of tracked files, staged files, and files ready for commit

## > `git add <file(s)>`

- **Stages** files for **commit**

## > `git commit -m "<Commit message>"`

- **Commits** all **staged** files with the specified commit message
- Preceded by the **add** command

## > `git revert <commit-hash>`

- **Revert** or undo a **commit**
- **NOTE:** This adds a new commit to undo a previous commit

### 3. Common and Useful Commands

> `git diff <commit-hash-1> <commit-hash-2>`

- Visually see the differences between two commits
- If only **one** commit-hash is provided, then run **diff** against the HEAD commit
- If no commit-hash is provided, then run diff against the

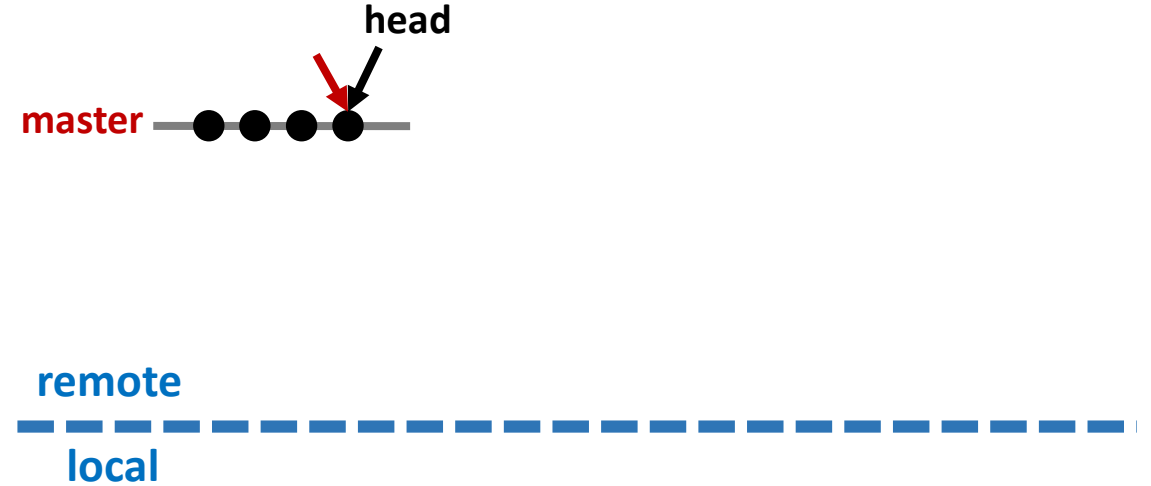
> `git diff --stat <commit-hash-1> <commit-hash-2>`

- List the files that differ between two commits



# 3. Common and Useful Commands (Visually)

- Consider a **remote repository** with one **branch**
  - Dots represent **commits**
  - Small arrows represent **pointers to commits**



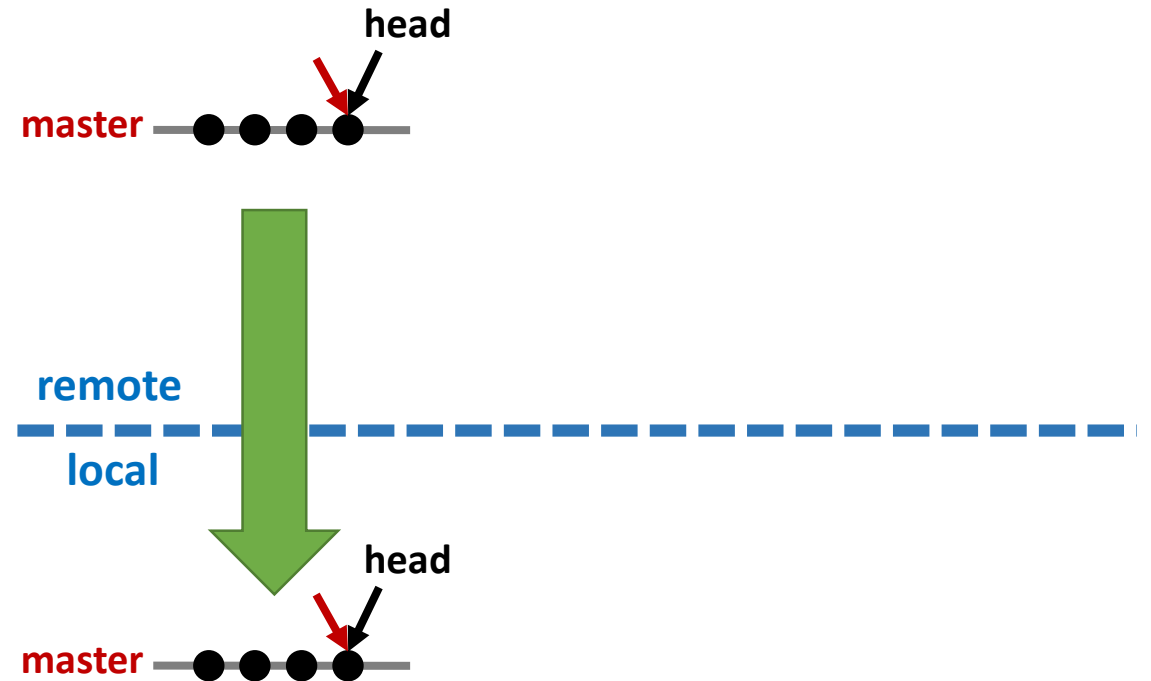
# 3. Common and Useful Commands (Visually)

- Consider a **remote repository** with one **branch**

- Dots represent **commits**
- Small arrows represent **pointers to commits**

- **Clone** the remote repository locally

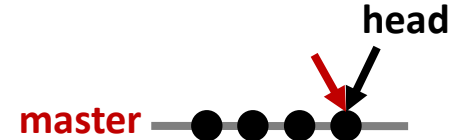
```
> git clone  
git@github.com:user1/my_repo
```



# 3. Common and Useful Commands (Visually)

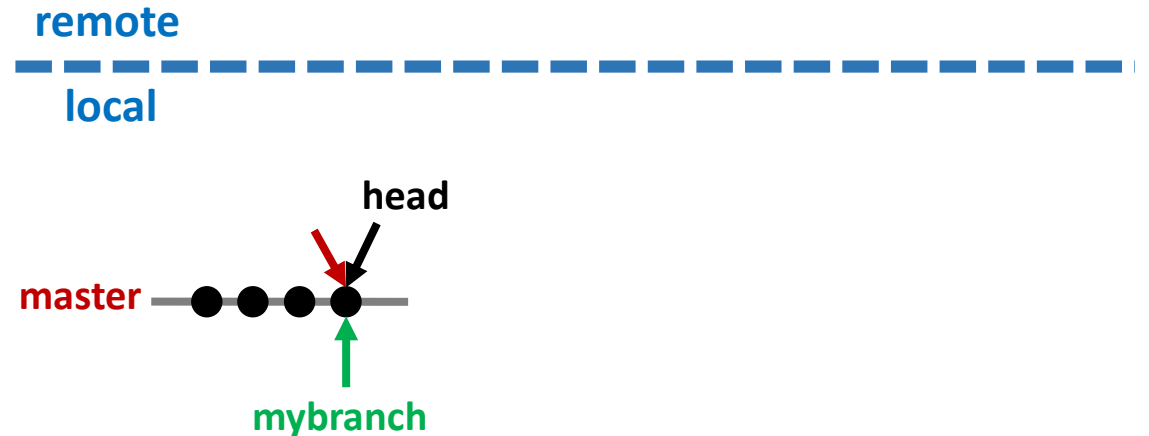
- **Clone** the remote repository locally

```
> git clone  
git@github.com:user1/my_repo
```



- Create a new **branch** in the **local repository**

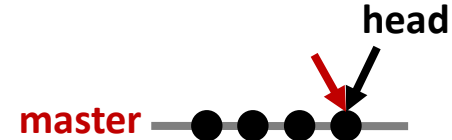
```
> git checkout -b mybranch
```



### 3. Common and Useful Commands (Visually)

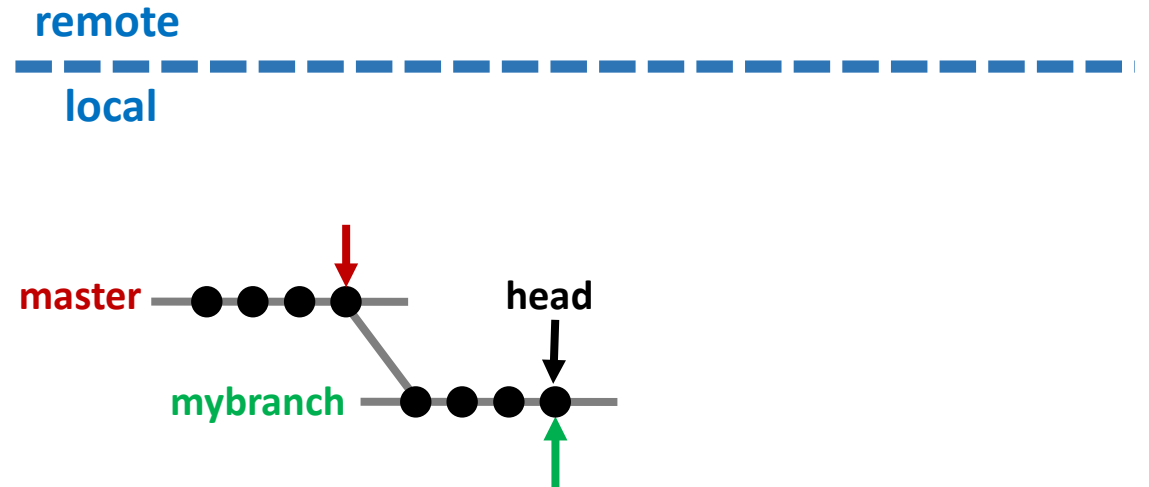
- Create a new **branch** in the **local repository**

```
> git checkout -b mybranch
```



- Make **commits** to the **new branch**

```
> # Make changes to files...  
> git add <file(s) that changed>  
> git commit -m "Initial commit"
```



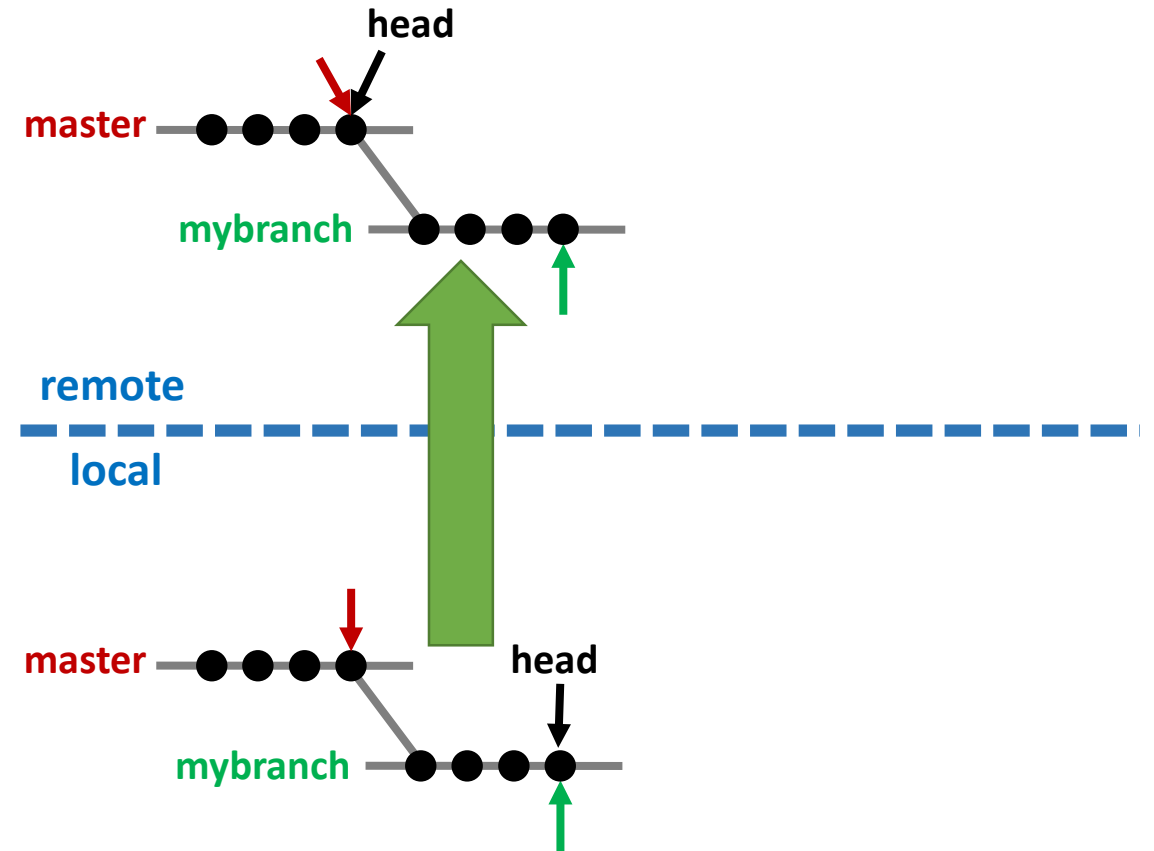
### 3. Common and Useful Commands (Visually)

- Make **commits** to the **new branch**

```
> # Make changes to files...  
> git add <file(s) that changed>  
> git commit -m "Initial commit"
```

- **Push** all commits associated with the **new branch** to the **remote repository**

```
> git push -u origin mybranch
```



### 3. Common and Useful Commands (Visually)

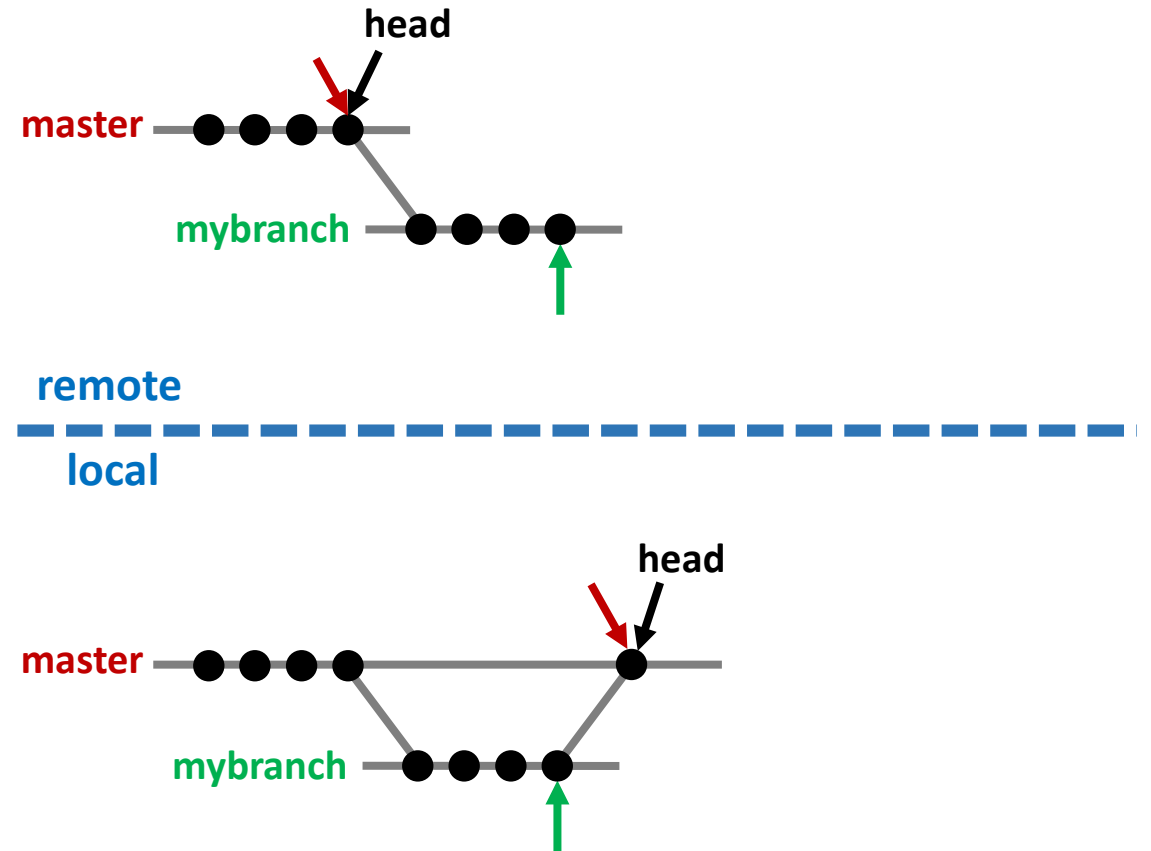
- **Push** all commits associated with the **new branch** to the **remote repository**

```
> git push -u origin mybranch
```

- **Merge** the **new** branch into the **master** branch

```
> git checkout master
```

```
> git merge mybranch
```



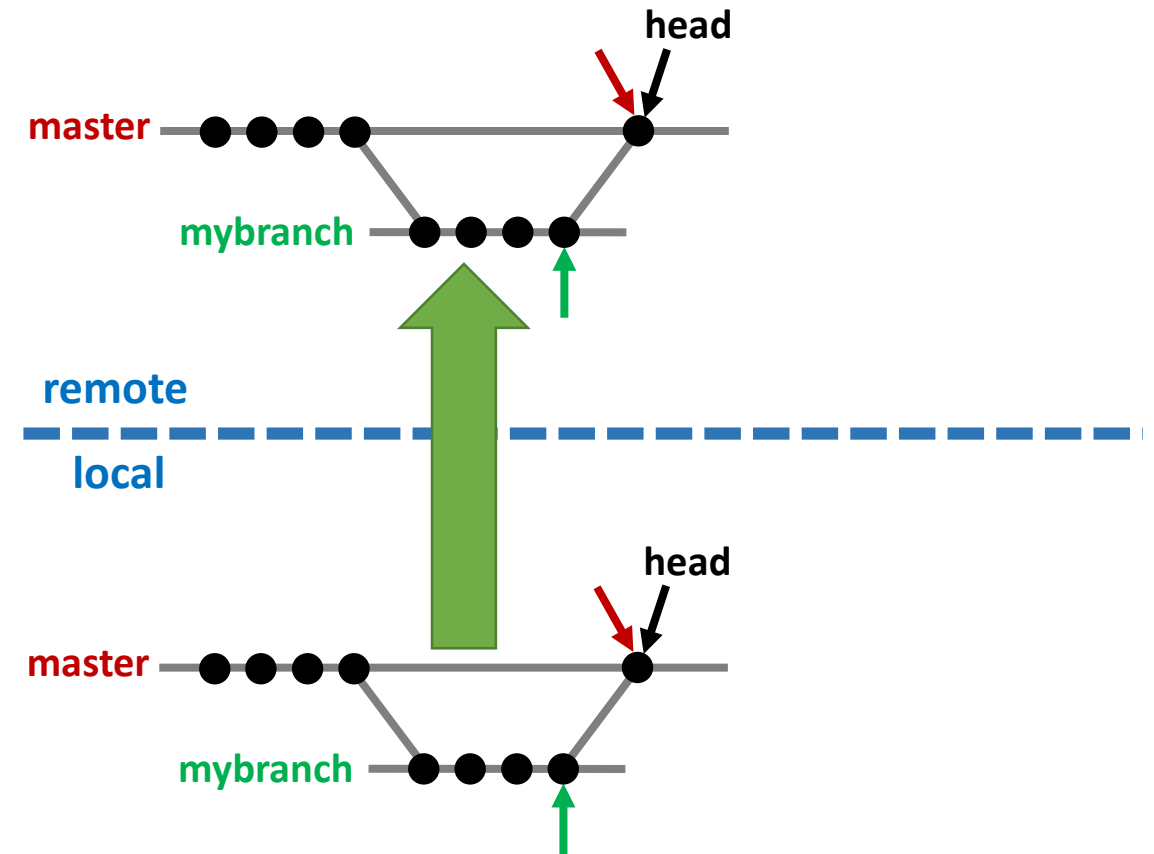
### 3. Common and Useful Commands (Visually)

- **Merge** the **new** branch into the **master** branch

```
> git checkout master  
> git merge mybranch
```

- **Push** all changes to the remote repository

```
> git push  
> git push origin mybranch
```



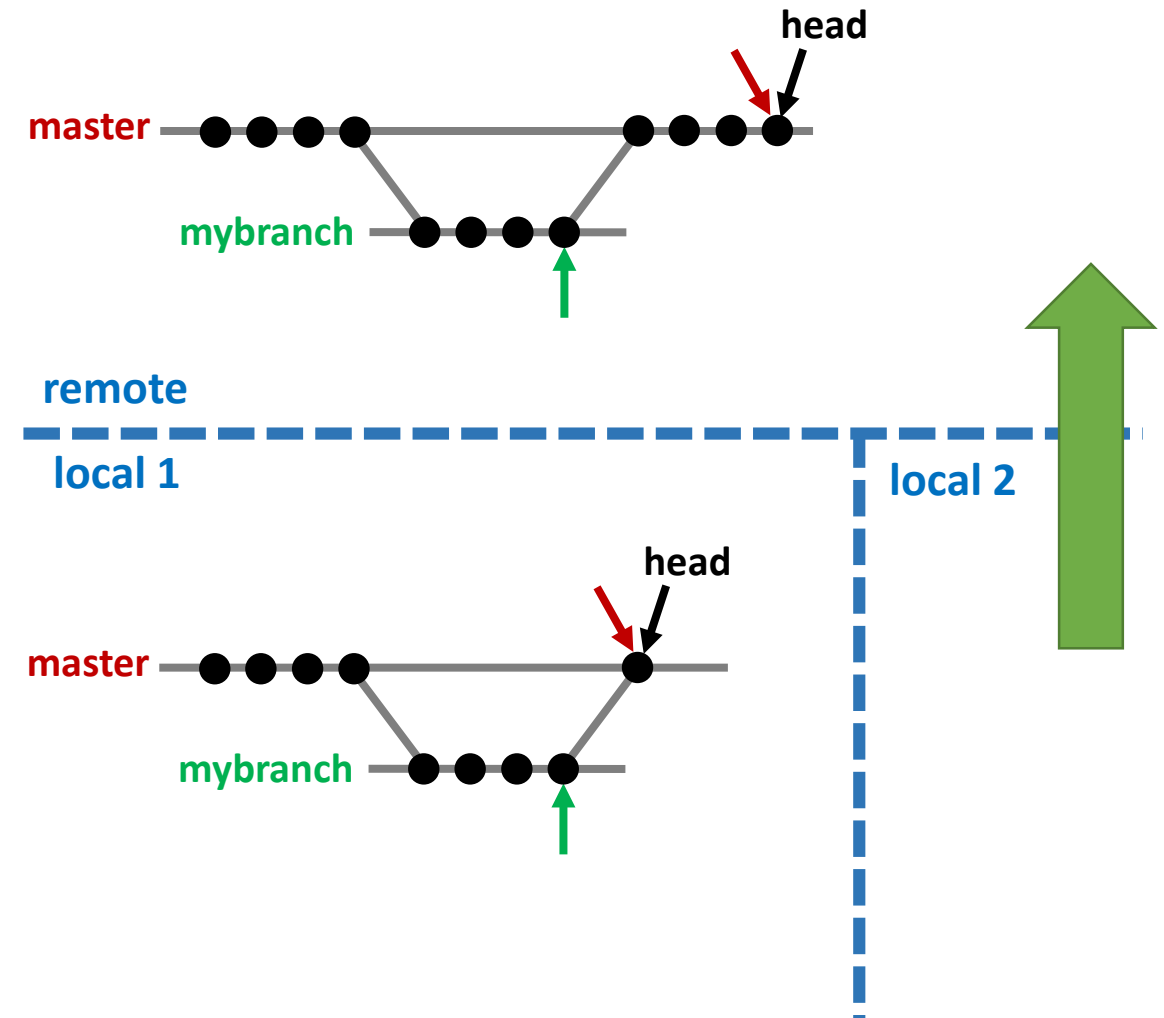
### 3. Common and Useful Commands (Visually)

- **Push** all changes to the remote repository

> git push

> git push origin mybranch

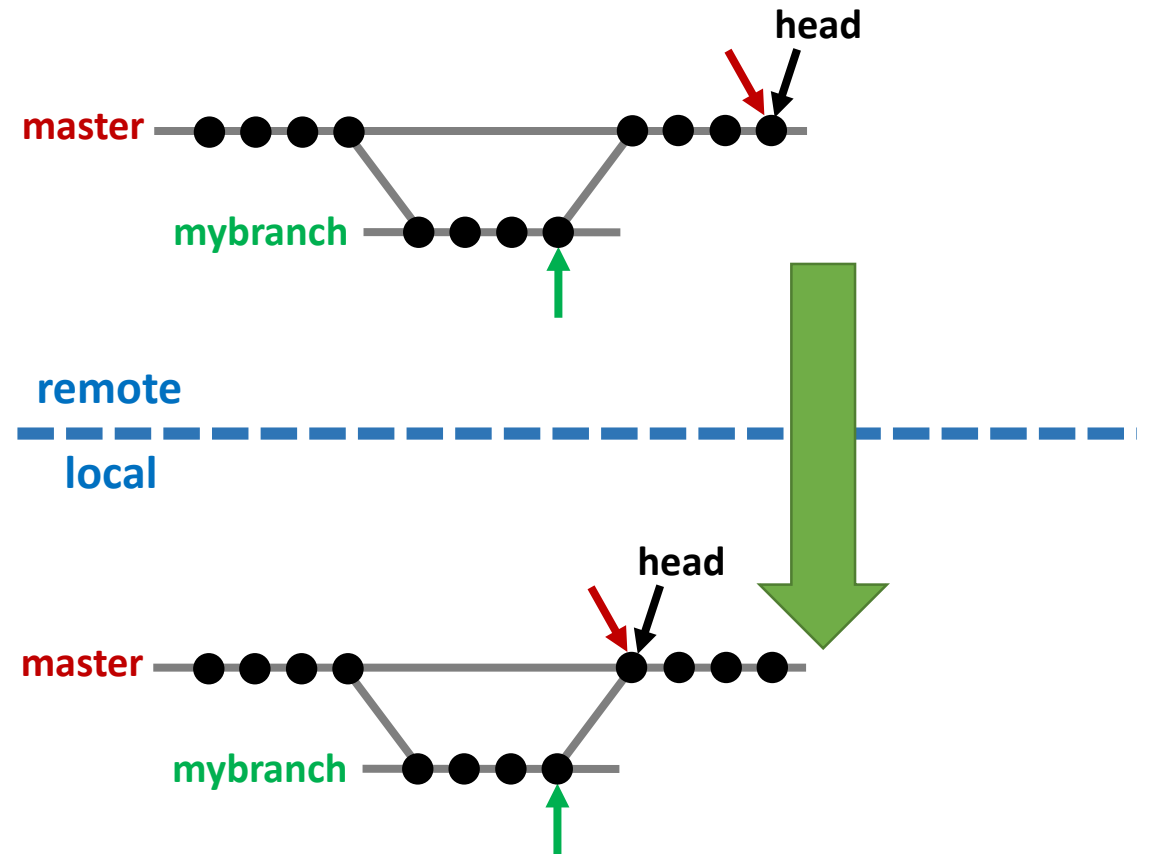
- Suppose another developer **pushes** changes to the **remote repository**





### 3. Common and Useful Commands (Visually)

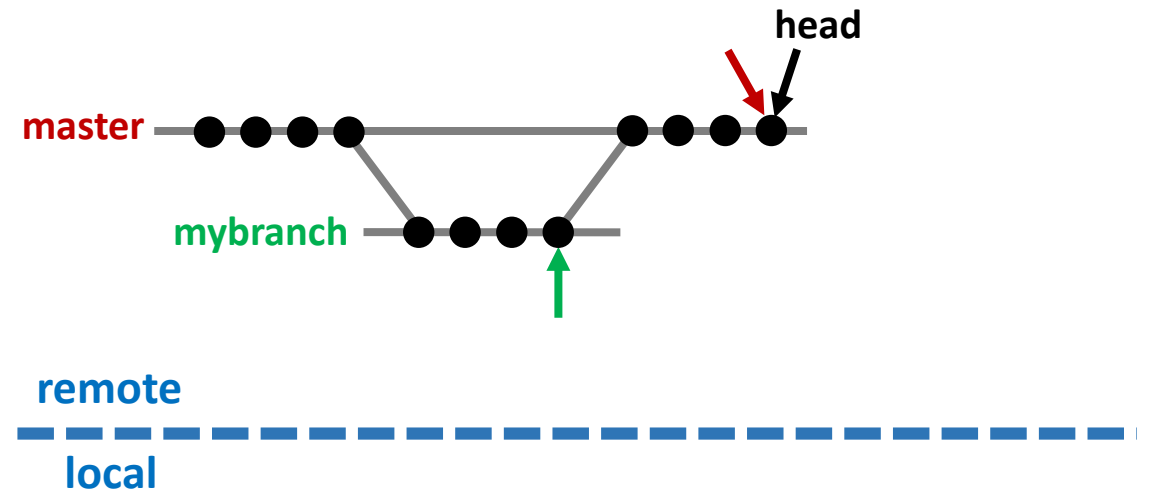
- Suppose another developer **pushes** changes to the **remote repository**
- **Fetch** the new commits from the **remote repository**
  - > `git checkout mybranch`
  - > `git fetch`
  - > `git fetch origin mybranch`



# 3. Common and Useful Commands (Visually)

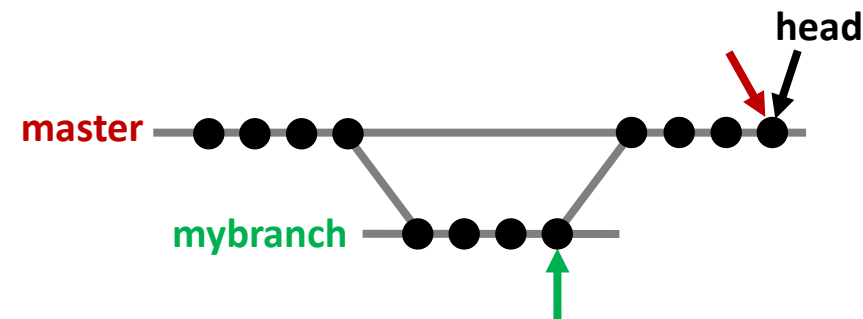
- **Fetch** the new commits from the remote repository

- > `git checkout mybranch`
- > `git fetch`
- > `git fetch origin mybranch`



- **Merge** the differences in your local repository

- > `git merge origin mybranch`



# 3. Typical Workflow with Common Commands

# Clone the repository

```
> git clone git@github.com:<user>/<repo_name>.git
```

# Create a new 'feature' branch

```
> git checkout -b my_feature_branch
```

# Do work locally on your files...

# Get ready to commit your code changes

```
> git status
```

```
> git add <file1> <file2> ... <fileN>
```

```
> git commit -m "A meaningful commit message..."
```

# Make sure you branch is sync'd with the remote repository before pushing commits

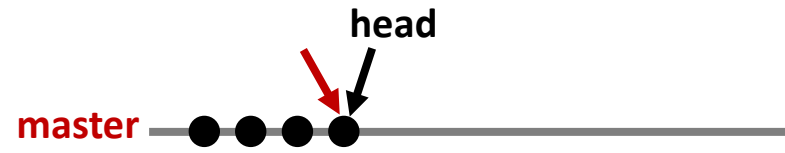
```
> git pull <remote-repo> <branch>
```

```
> git push <remote-repo> <branch>
```

## 4. Step-by-Step Walkthrough with Git

- Below is the state information for the example to be presented:
  - There exists a **remote repository** with the **master branch** and one file
  - Only one individual is actively making changes to the repository at a time
- The following will be demonstrated:
  - **Branch** creation
  - Changes local to the **workspace**
  - **Staging** and **committing** files
  - Updating the **local** and **remote repositories**

No files in local repository



remote

local

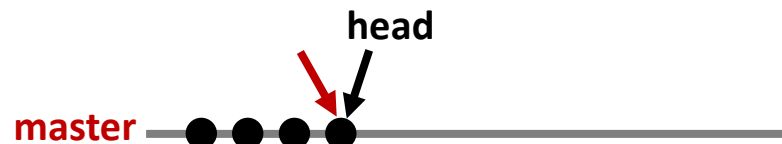
No local repository

No files in staging area

todo.txt

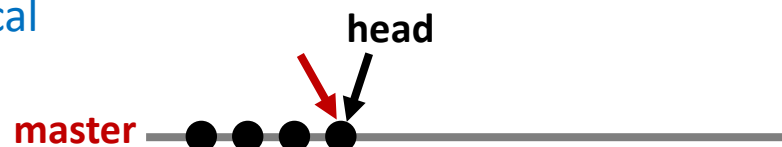
Things to do:

1. Clone repo
2. Make changes
3. Commit changes
4. Update remote



remote

local



Changes for Commit:

Changed Files:

Untracked Files:

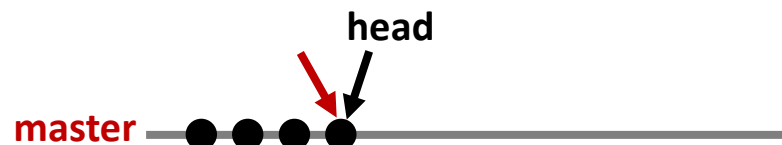
```
> git clone git@github.com:user/myrepo.git
>
```

todo.txt

test.txt

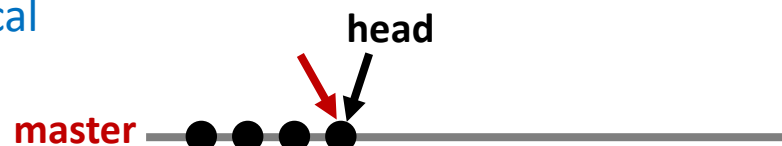
Things to do:

1. Clone repo
2. Make changes
3. Commit changes
4. Update remote



remote

local



```
> git clone git@github.com:user/myrepo.git
>
```

Changes for Commit:

Changed Files:

Untracked Files:

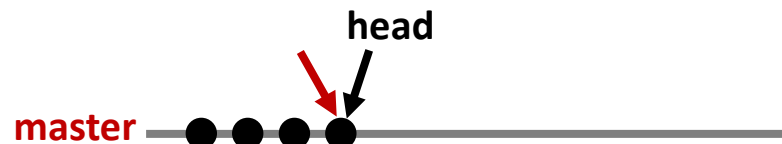
test.txt

todo.txt

test.txt

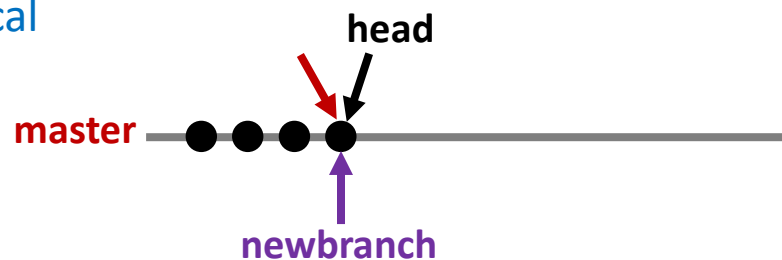
Things to do:

1. Clone repo
2. Make changes
3. Commit changes
4. Update remote



remote

local



```
> git clone git@github.com:user/myrepo.git  
> git checkout -b newbranch
```

Changes for Commit:

Changed Files:

Untracked Files:

test.txt



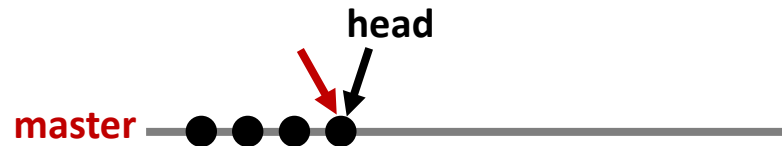
todo.txt

Things to do:

1. Clone repo
2. Make changes
3. Commit changes
4. Update remote

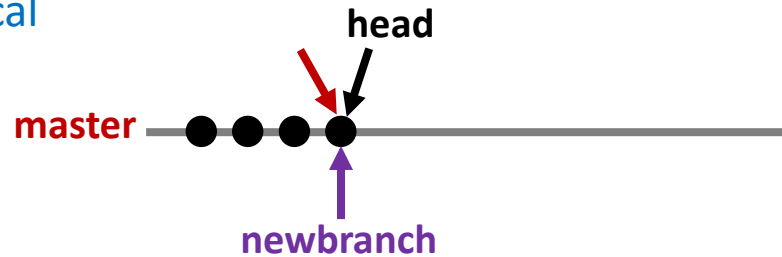
test.txt

Testing...testing...1...2...3



remote

local



Changes for Commit:

Changed Files:

Untracked Files:

test.txt

```
> git clone git@github.com:user/myrepo.git
> git checkout -b newbranch
```

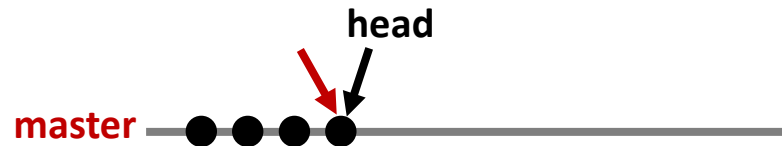
todo.txt

Things to do:

1. Clone repo
2. Make changes
3. Commit changes
4. Update remote

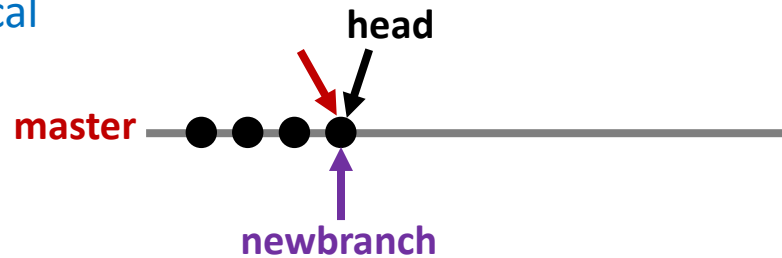
test.txt

Testing...testing...1...2...3



remote

local



Changes for Commit:

test.txt

Changed Files:

Untracked Files:

todo.txt

Things to do:

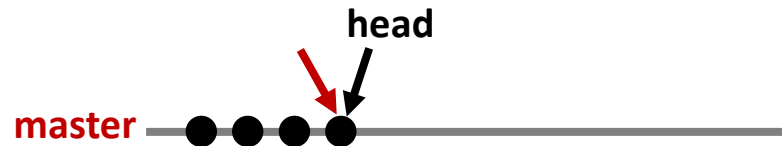
1. Clone repo
2. Make changes
3. Commit changes
4. Update remote
5. Make more changes

test.txt

Testing...testing...1...2...3

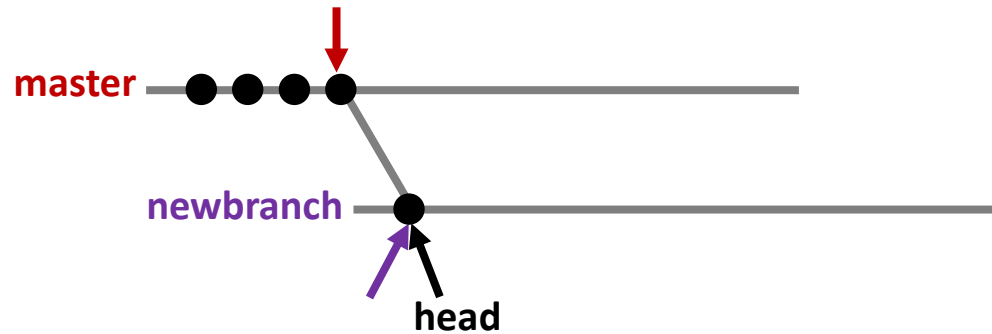
Testing...1

```
> git clone git@github.com:user/myrepo.git
> git checkout -b newbranch
> git add test.txt
> git commit -m "Updated test.txt"
>
```



remote

local



Changes for Commit:

Changed Files:

Untracked Files:

test.txt

todo.txt

todo.txt

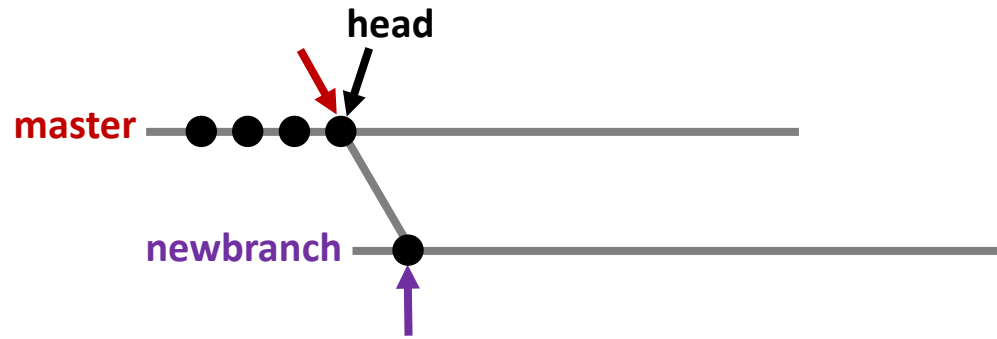
Things to do:

1. Clone repo
2. Make changes
3. Commit changes
4. Update remote
5. Make more changes

test.txt

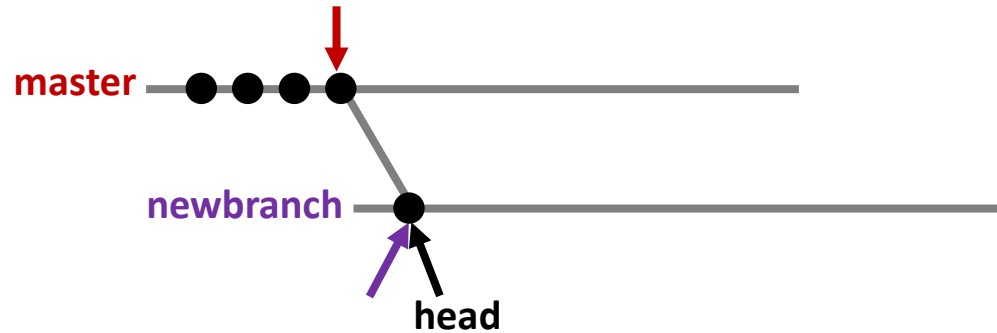
Testing...testing...1...2...3  
Testing...1

```
> git clone git@github.com:user/myrepo.git
> git checkout -b newbranch
> git add test.txt
> git commit -m "Updated test.txt"
> git push -u origin newbranch
>
```



remote

local



Changes for Commit:

Changed Files:

Untracked Files:

test.txt

todo.txt

todo.txt

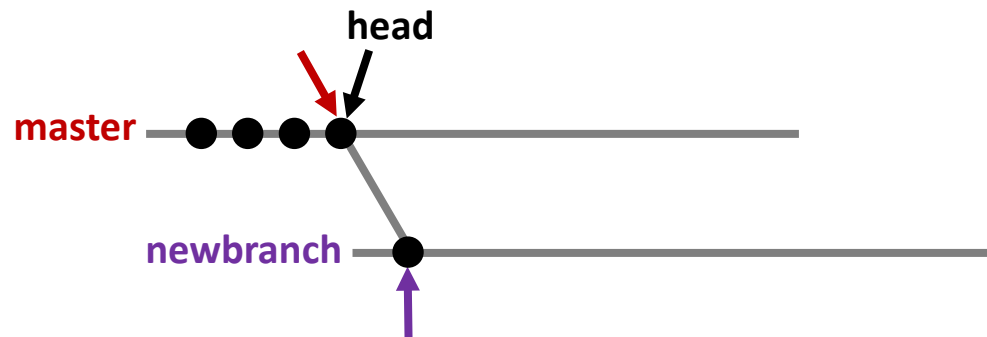
Things to do:

1. Clone repo
2. Make changes
3. Commit changes
4. Update remote
5. Make more changes

test.txt

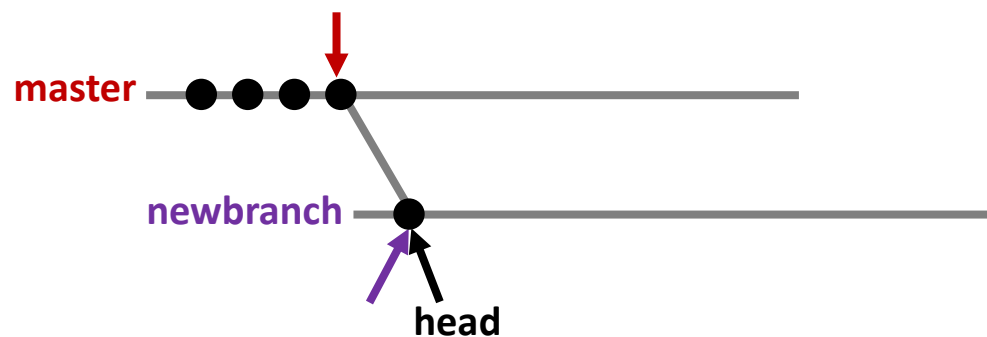
Testing...testing...1...2...3

Testing...1



remote

local



Changes for Commit:

Changed Files:

Untracked Files:

test.txt

todo.txt

```
> git clone git@github.com:user/myrepo.git
> git checkout -b newbranch
> git add test.txt
> git commit -m "Updated test.txt"
> git push -u origin newbranch
> git add text.txt todo.txt
>
```

todo.txt

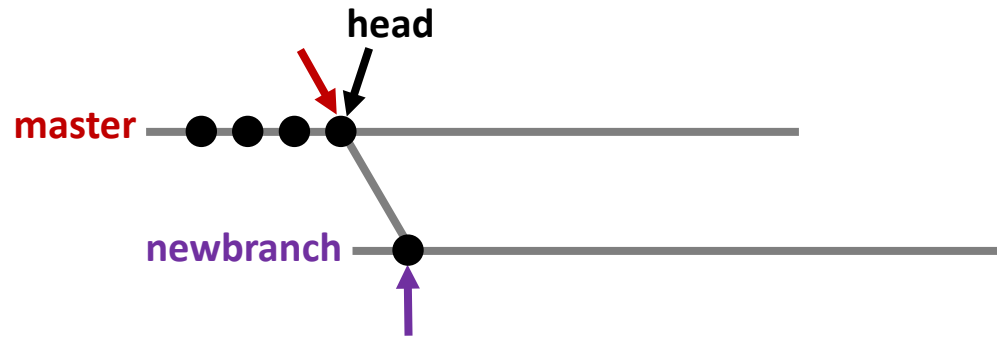
Things to do:

1. Clone repo
2. Make changes
3. Commit changes
4. Update remote
5. Make more changes

test.txt

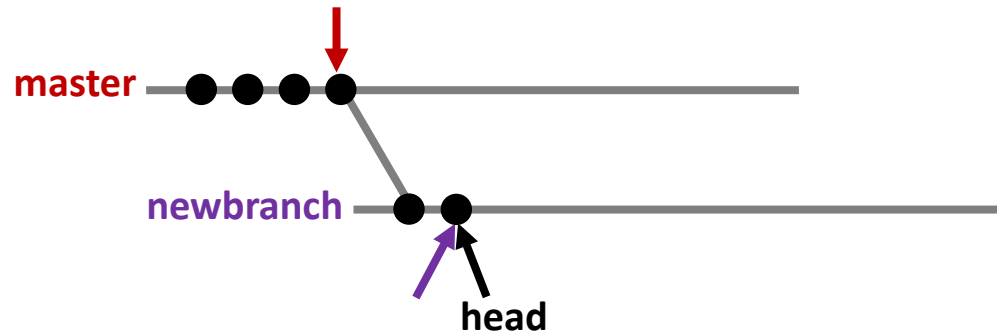
Testing...testing...1...2...3  
Testing...1

```
> git clone git@github.com:user/myrepo.git
> git checkout -b newbranch
> git add test.txt
> git commit -m "Updated test.txt"
> git push -u origin newbranch
> git add text.txt todo.txt
> git commit -m "Updated test.txt + todo.txt"
>
```



remote

local



Changes for Commit:

Changed Files:

Untracked Files:

todo.txt

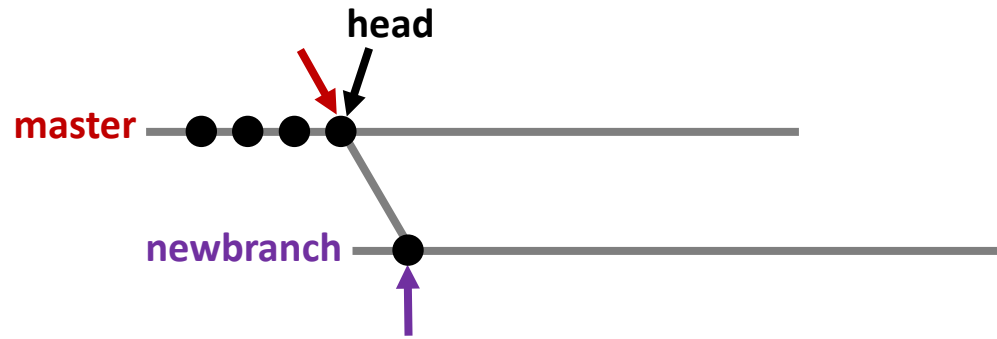
Things to do:

1. Clone repo
2. Make changes
3. Commit changes
4. Update remote
5. Make more changes

test.txt

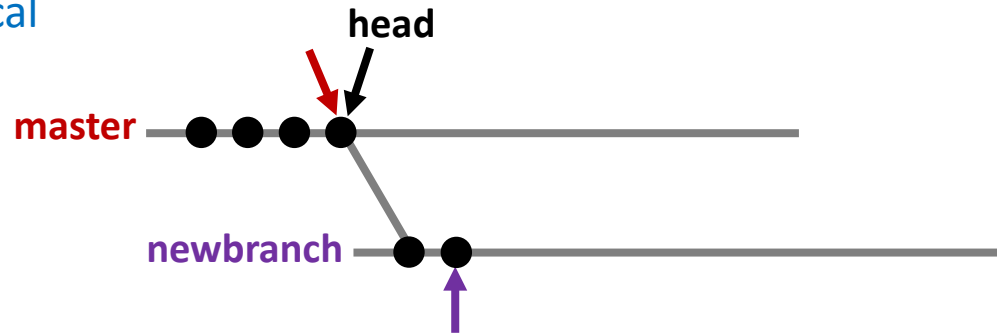
Testing...testing...1...2...3  
Testing...1

```
> git clone git@github.com:user/myrepo.git
> git checkout -b newbranch
> git add test.txt
> git commit -m "Updated test.txt"
> git push -u origin newbranch
> git add text.txt todo.txt
> git commit -m "Updated test.txt + todo.txt"
> git checkout master
>
```



remote

local



Changes for Commit:

Changed Files:

Untracked Files:

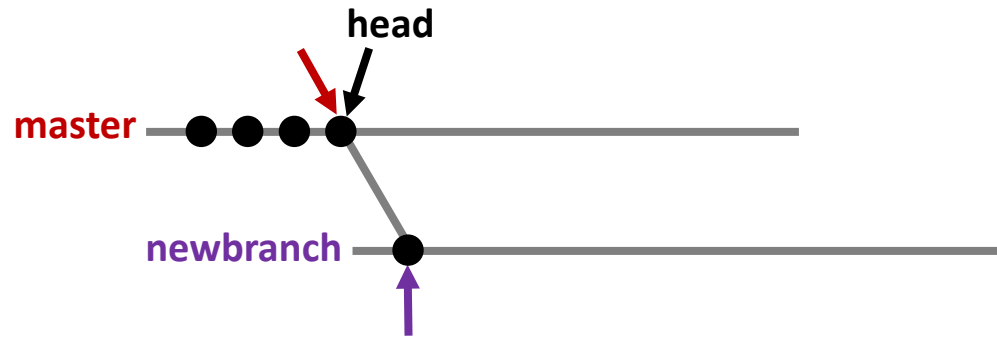
todo.txt

Things to do:

1. Clone repo
2. Make changes
3. Commit changes
4. Update remote
5. Make more changes

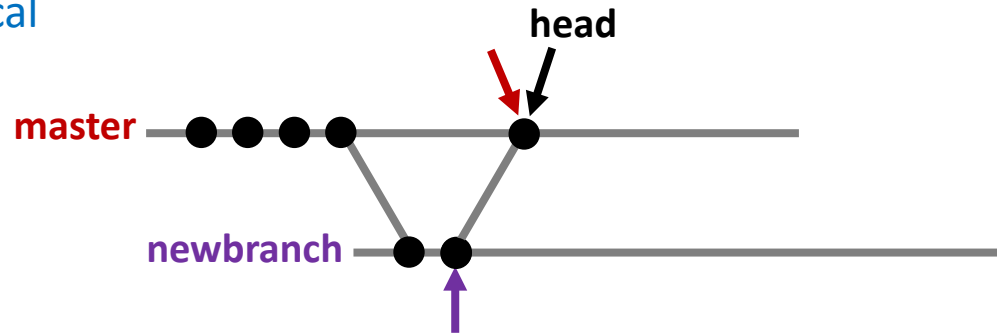
test.txt

Testing...testing...1...2...3  
Testing...1



remote

local



Changes for Commit:

Changed Files:

Untracked Files:

```
> git clone git@github.com:user/myrepo.git
> git checkout -b newbranch
> git add test.txt
> git commit -m "Updated test.txt"
> git push -u origin newbranch
> git add text.txt todo.txt
> git commit -m "Updated test.txt + todo.txt"
> git checkout master
> git merge newbranch
>
```



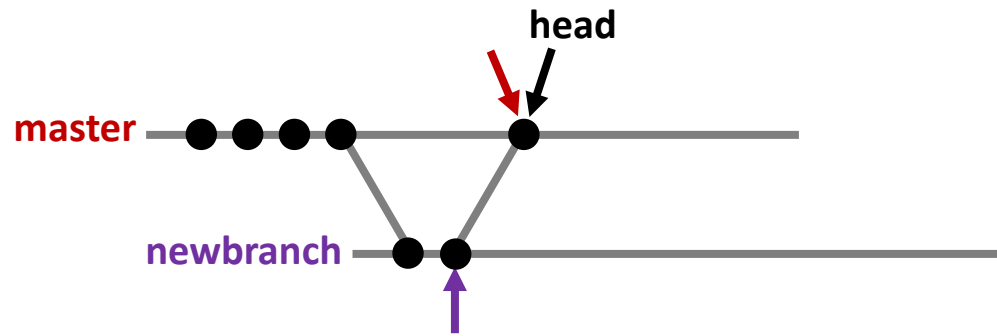
todo.txt

Things to do:

1. Clone repo
2. Make changes
3. Commit changes
4. Update remote
5. Make more changes

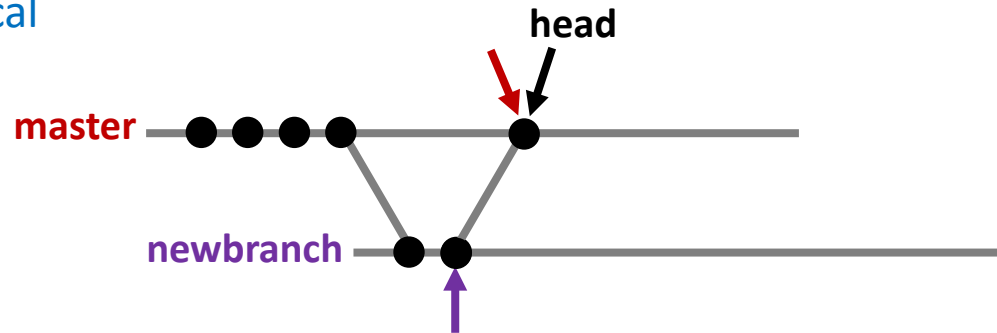
test.txt

Testing...testing...1...2...3  
Testing...1



remote

local



Changes for Commit:

Changed Files:

Untracked Files:

```
> git clone git@github.com:user/myrepo.git
> git checkout -b newbranch
> git add test.txt
> git commit -m "Updated test.txt"
> git push -u origin newbranch
> git add text.txt todo.txt
> git commit -m "Updated test.txt + todo.txt"
> git checkout master
> git merge newbranch
> git push
>
```

todo.txt

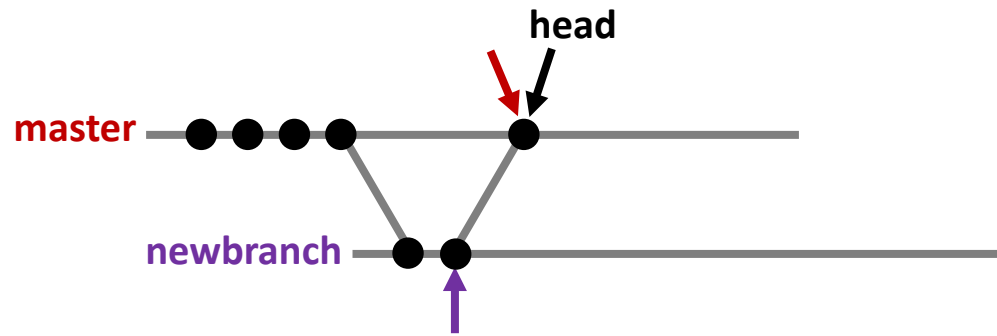
Things to do:

1. Clone repo
2. Make changes
3. Commit changes
4. Update remote
5. Make more changes

test.txt

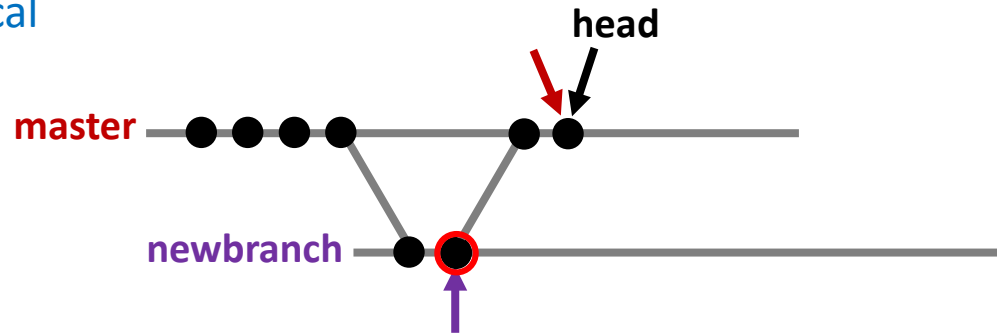
Testing...testing...1...2...3

Testing...1



remote

local



Changes for Commit:

Changed Files:

Untracked Files:

```
> git clone git@github.com:user/myrepo.git
> git checkout -b newbranch
> git add test.txt
> git commit -m "Updated test.txt"
> git push -u origin newbranch
> git add text.txt todo.txt
> git commit -m "Updated test.txt + todo.txt"
> git checkout master
> git merge newbranch
> git push
> git revert <commit-id>
>
```

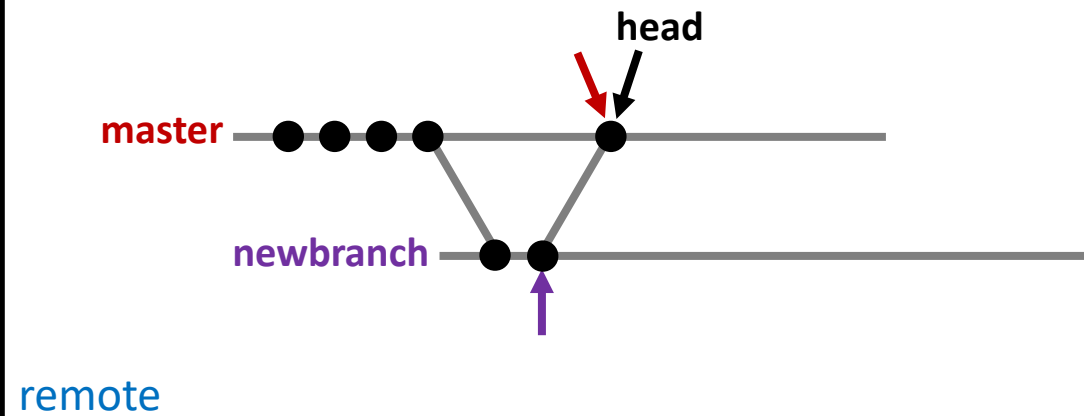
todo.txt

Things to do:

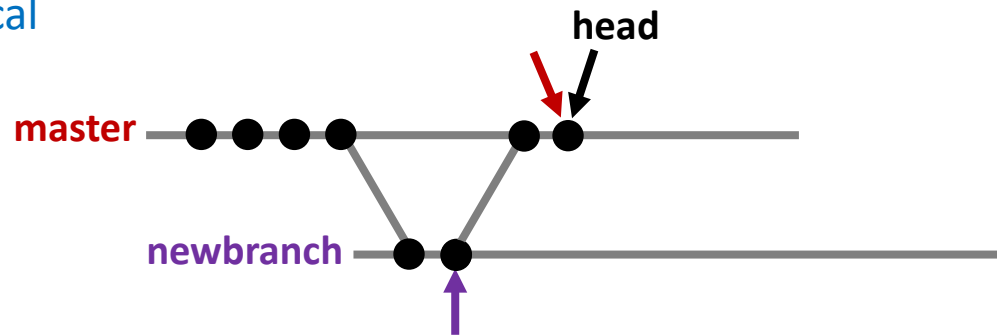
1. Clone repo
2. Make changes
3. Commit changes
4. Update remote

test.txt

Testing...testing...1...2...3



local



```
> git clone git@github.com:user/myrepo.git
> git checkout -b newbranch
> git add test.txt
> git commit -m "Updated test.txt"
> git push -u origin newbranch
> git add text.txt todo.txt
> git commit -m "Updated test.txt + todo.txt"
> git checkout master
> git merge newbranch
> git push
> git revert <commit-id>
>
```

Changes for Commit:

Changed Files:

Untracked Files:

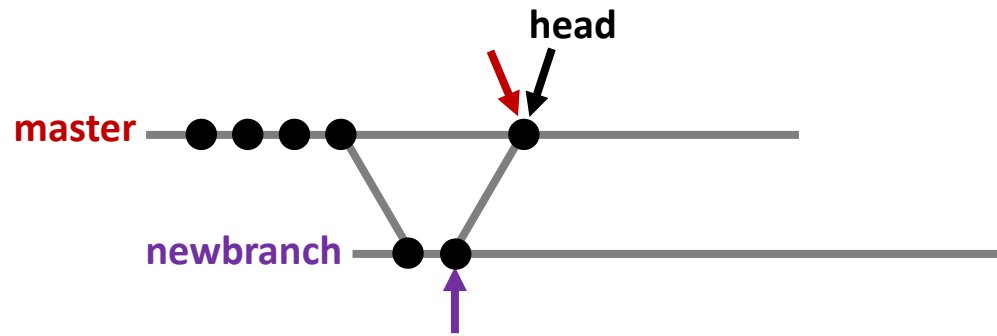
todo.txt

Things to do:

1. Clone repo
2. Make changes
3. Commit changes
4. Update remote

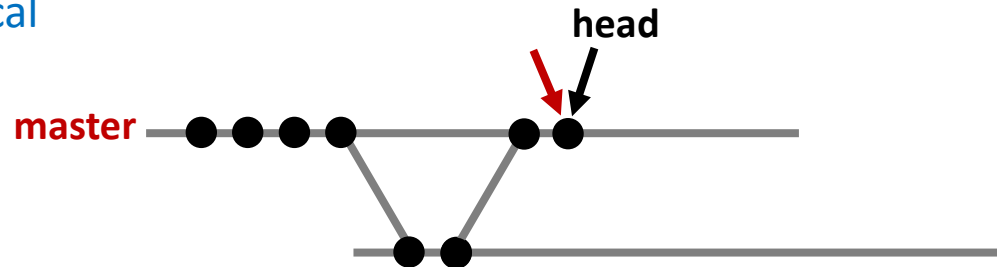
test.txt

Testing...testing...1...2...3



remote

local



Changes for Commit:

Changed Files:

Untracked Files:

```
> git clone git@github.com:user/myrepo.git
> git checkout -b newbranch
> git add test.txt
> git commit -m "Updated test.txt"
> git push -u origin newbranch
> git add text.txt todo.txt
> git commit -m "Updated test.txt + todo.txt"
> git checkout master
> git merge newbranch
> git push
> git revert <commit-id>
> git branch -d newbranch
>
```

# \*\*\* Forking vs Cloning \*\*\*



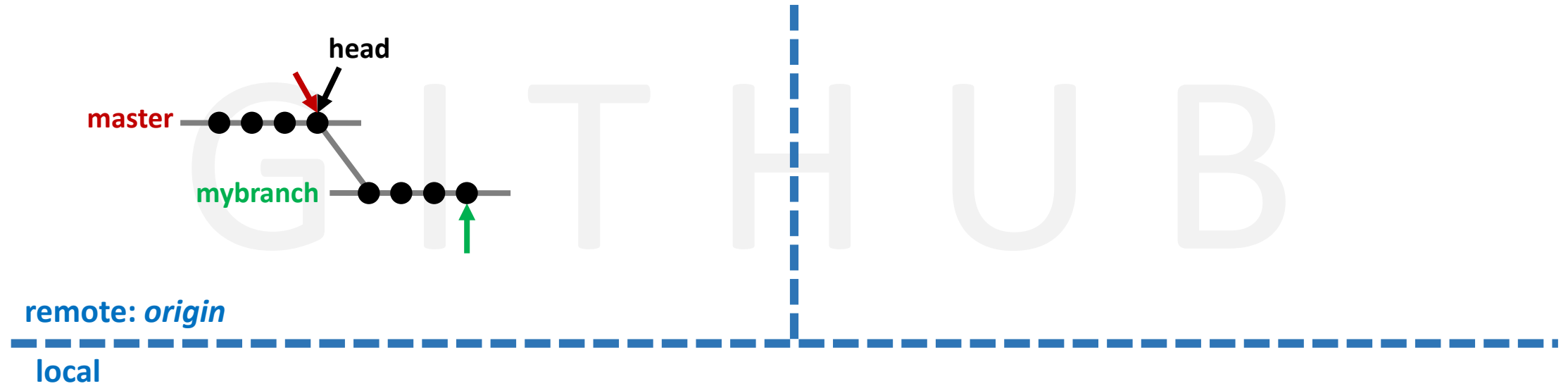
**VS**



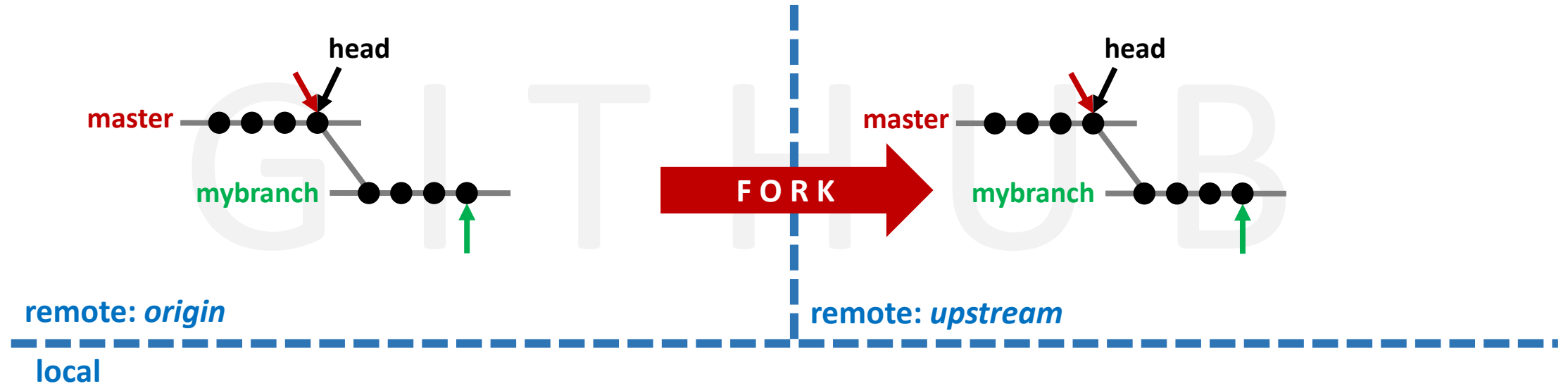
# \*\*\* Forking vs Cloning \*\*\*

- **Fork** – a **remote** copy of a **remote repository** at a certain point in time
  - A **GitHub** construct (e.g. applies to all public **remote repositories**)
  - **Ex.** Making a copy of someone else's **GitHub** [remote] repository in your **GitHub** account
  - All public repositories on **GitHub** can be **forked**
  
- **Clone** – a **local** copy of a **remote repository** at a certain point in time
  - A **Git** construct (independent of **GitHub**)
  - All public repositories on **GitHub** (including **forks**) can be **cloned** to a **local repository**
  - You can always **push** to your **fork**, but you might not have permission to **push** to an arbitrary **repo**
  
- **Collaborative Workflow:**
  1. Create a **fork** of a **project or repository** that you want to contribute to in **GitHub**
  2. **Clone** your **fork** to your **local repository** (refer to your **fork** as **origin**)
  3. Add the **original remote repository** to your **local repository** (refer to this as **upstream**)
  4. **Pull** changes from **upstream** and **push** changes to **origin**
  5. When you are ready to make your changes available to the **original remote repository**, create a **pull-request** from your **fork**
  6. Repeat steps (4) – (5)

# \*\*\* Forking vs Cloning (Visualized) \*\*\*

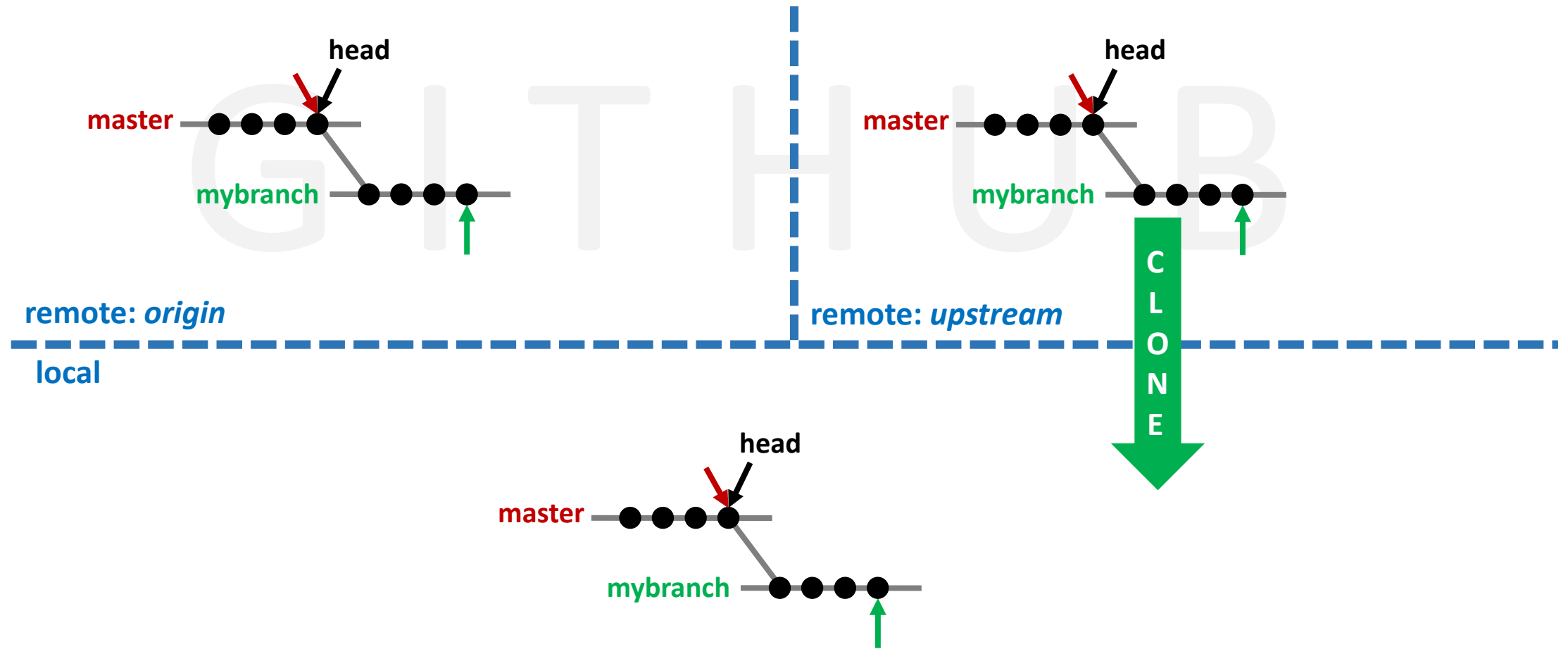


# \*\*\* Forking vs Cloning (Visualized) \*\*\*

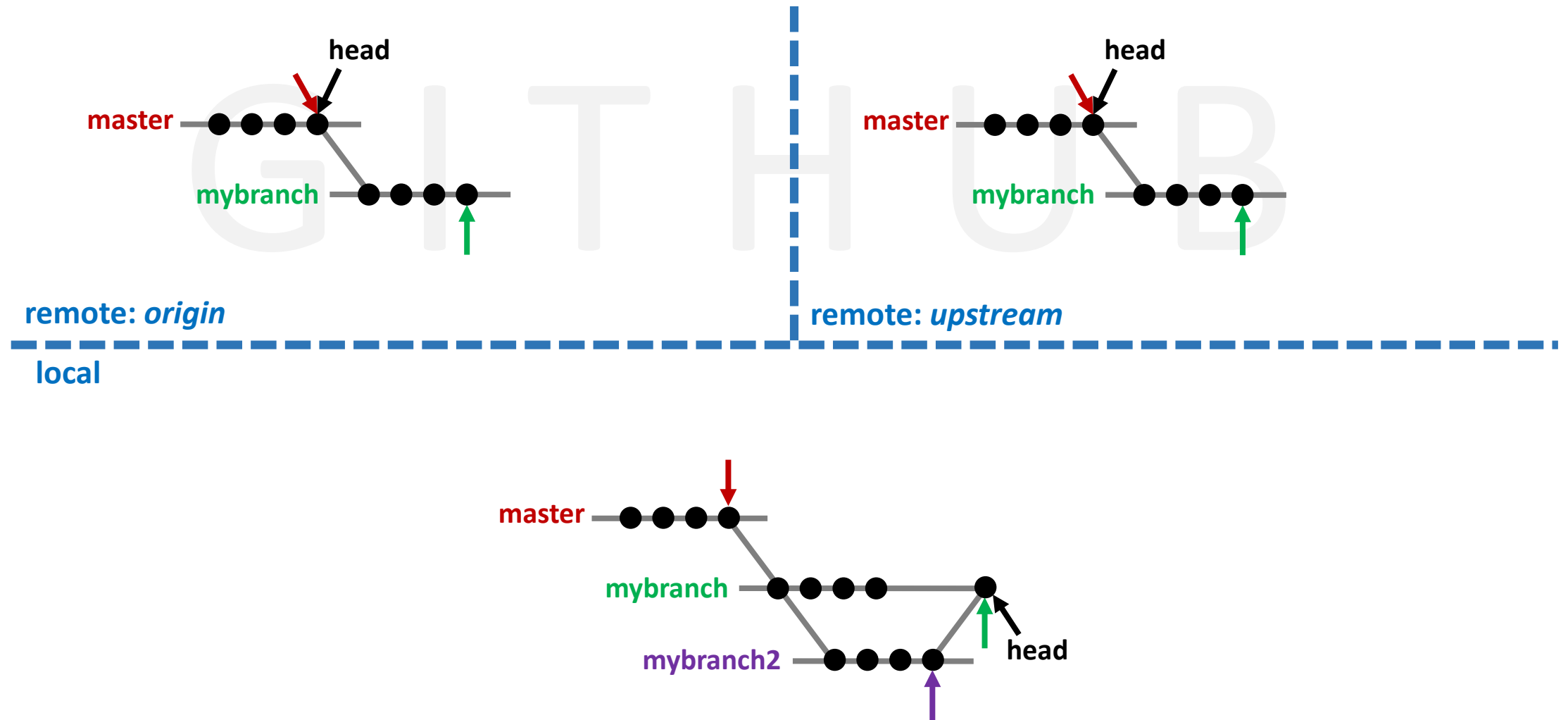




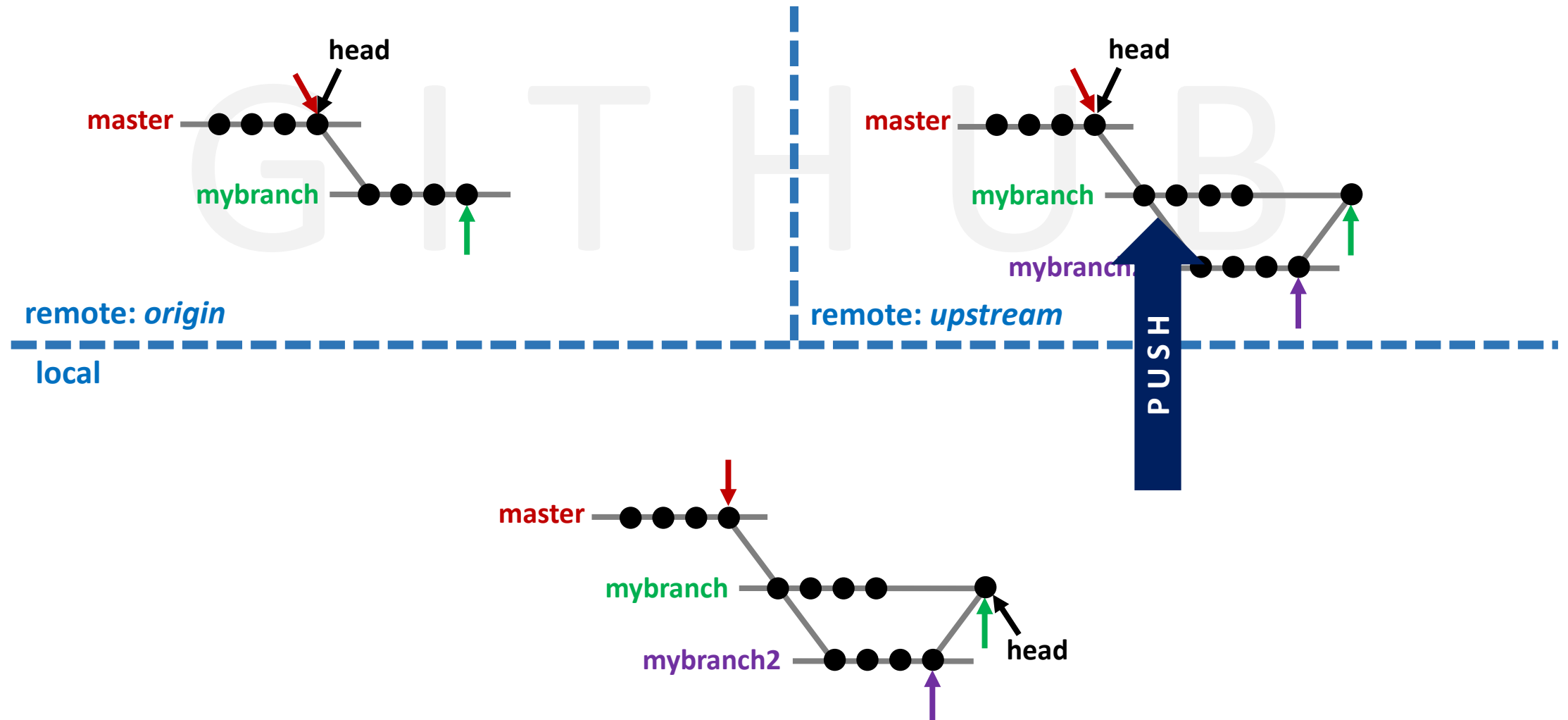
# \*\*\* Forking vs Cloning (Visualized) \*\*\*



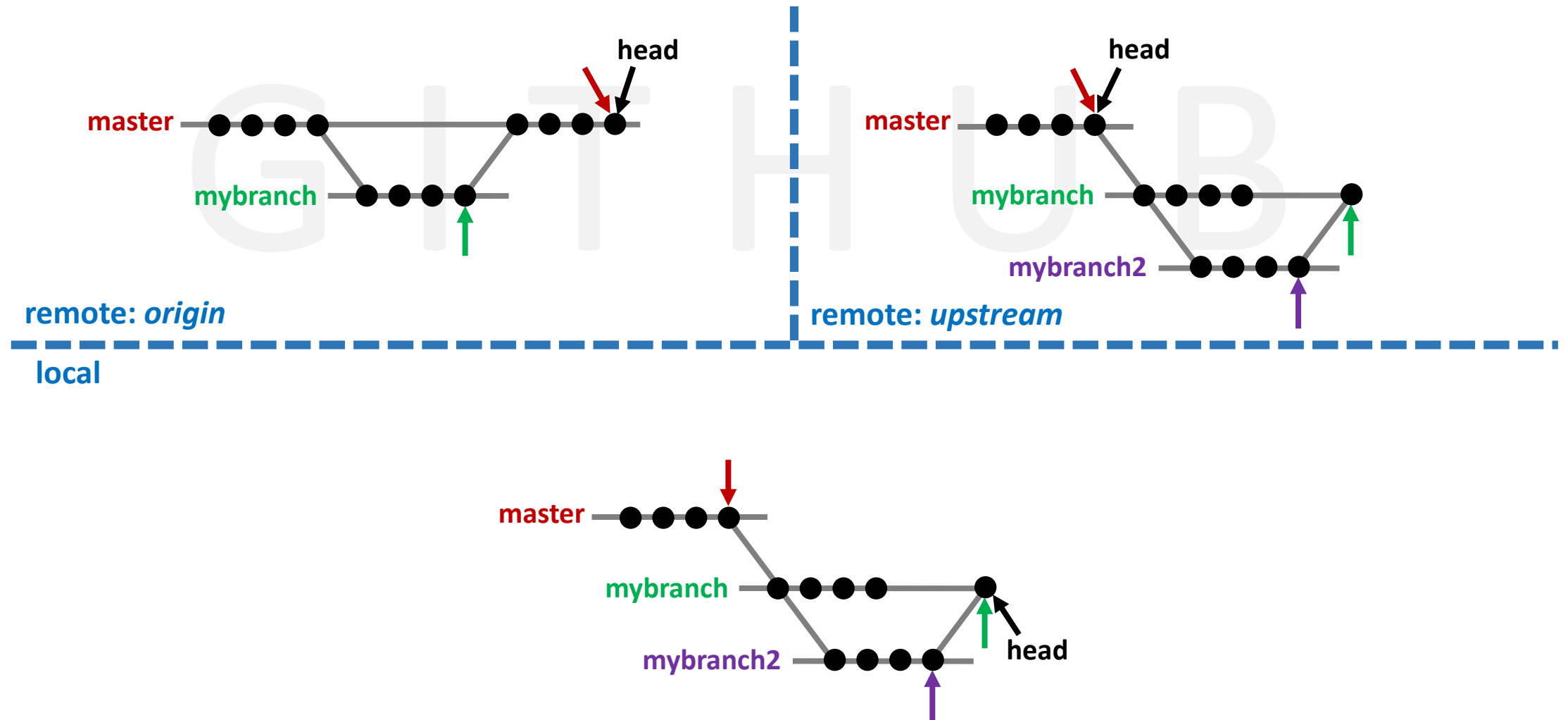
# \*\*\* Forking vs Cloning (Visualized) \*\*\*



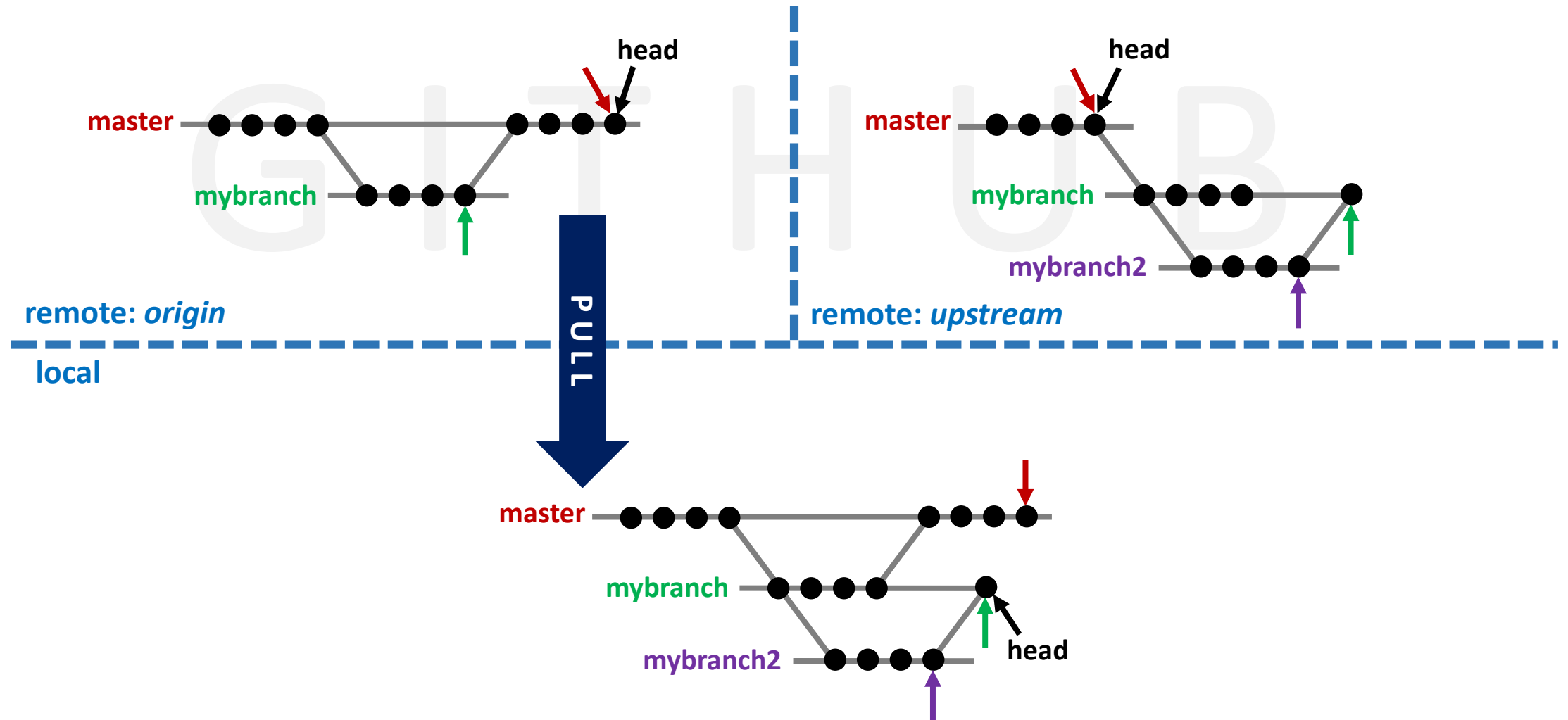
# \*\*\* Forking vs Cloning (Visualized) \*\*\*



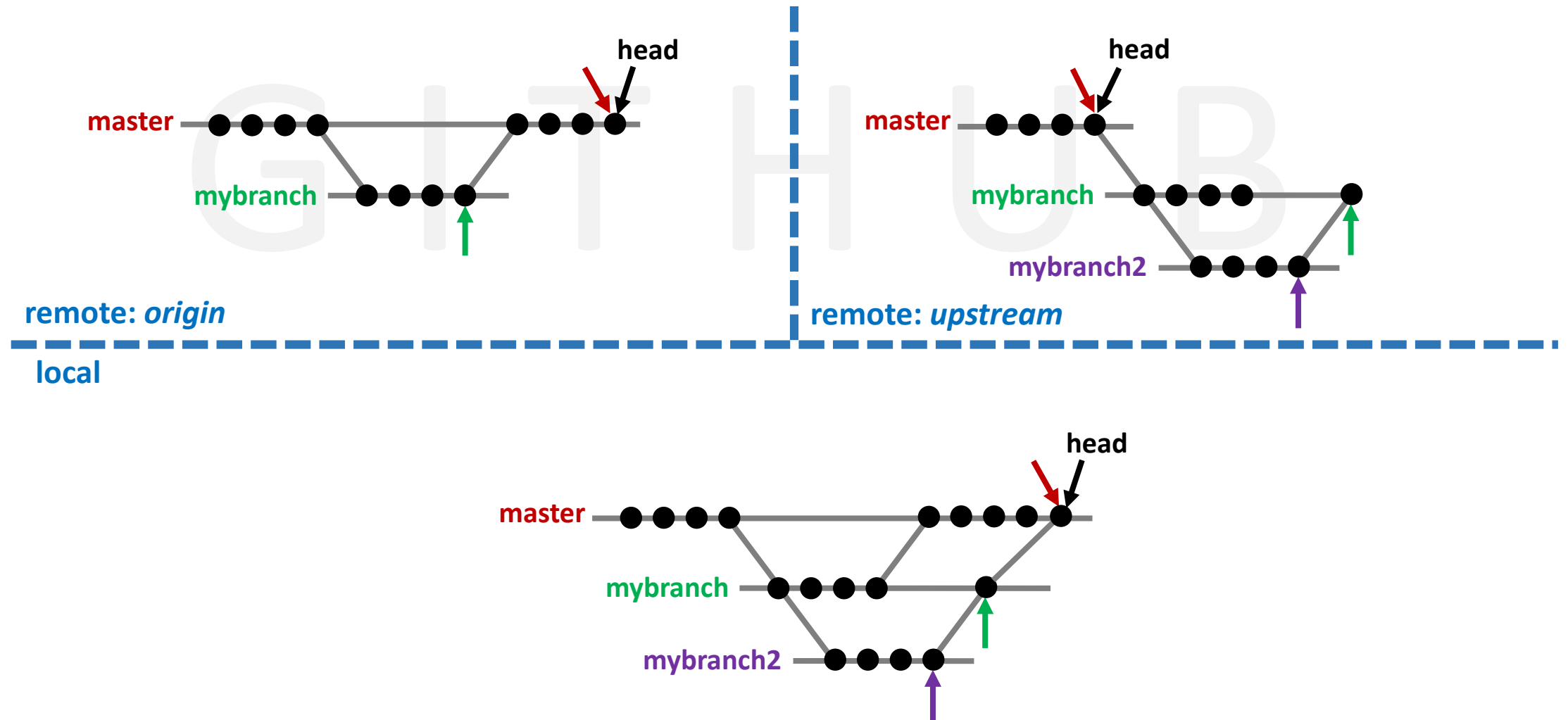
# \*\*\* Forking vs Cloning (Visualized) \*\*\*



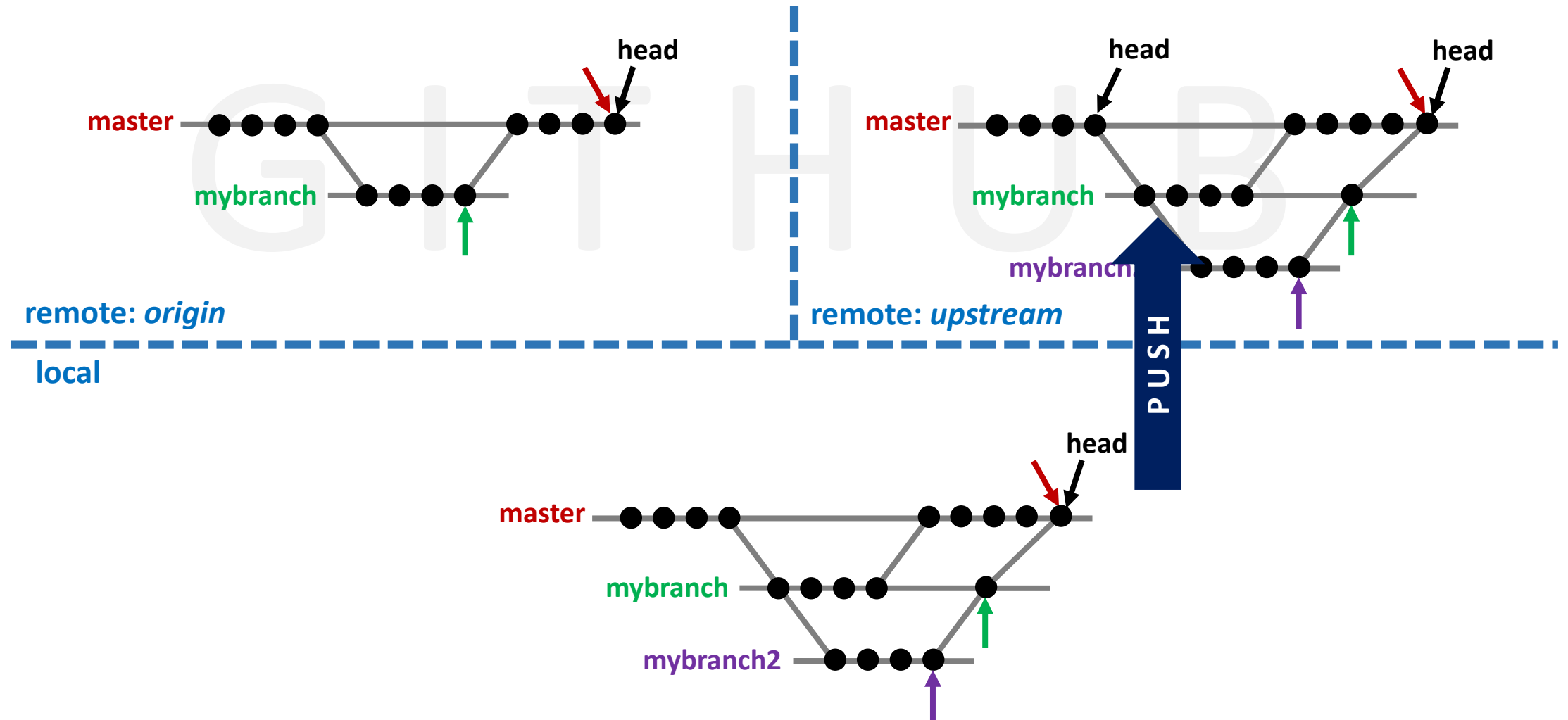
# \*\*\* Forking vs Cloning (Visualized) \*\*\*



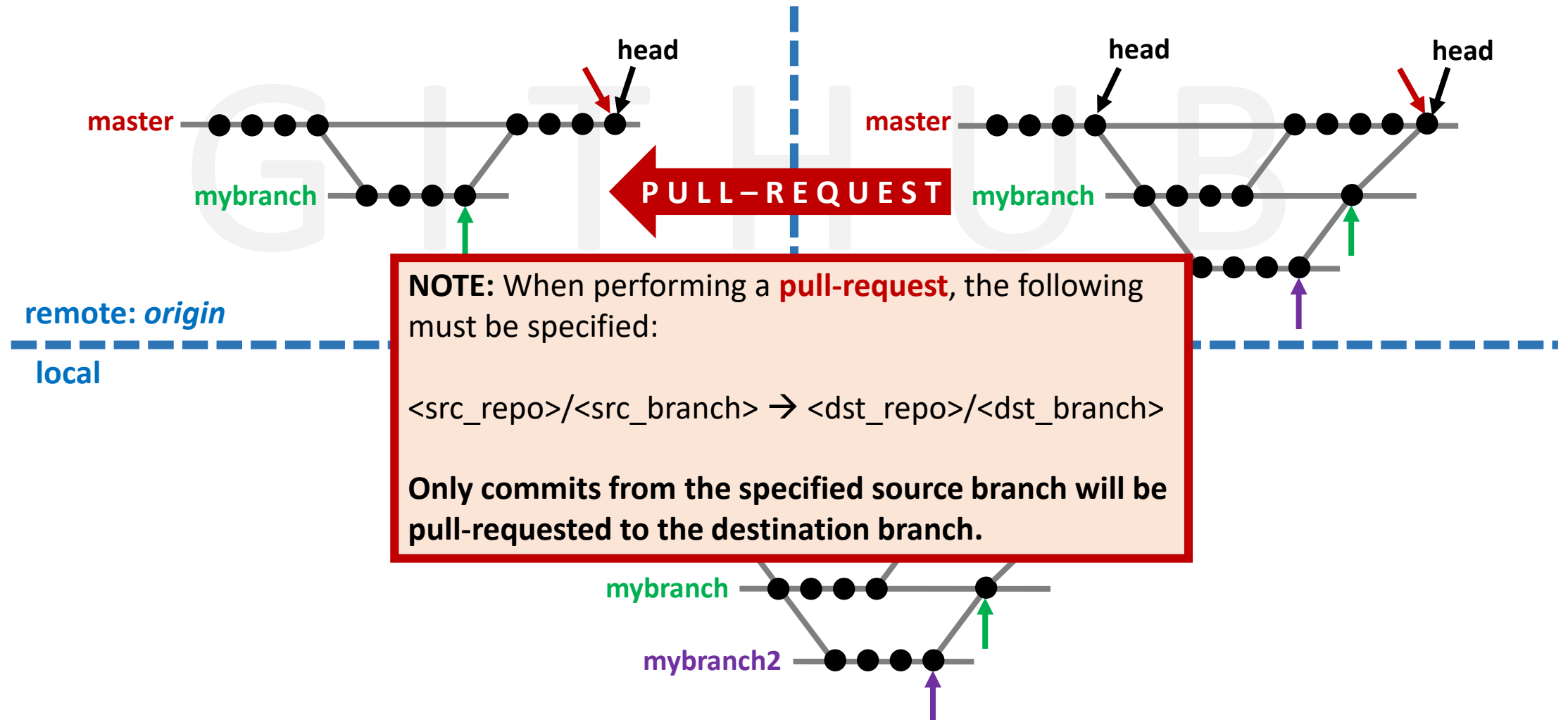
# \*\*\* Forking vs Cloning (Visualized) \*\*\*



# \*\*\* Forking vs Cloning (Visualized) \*\*\*

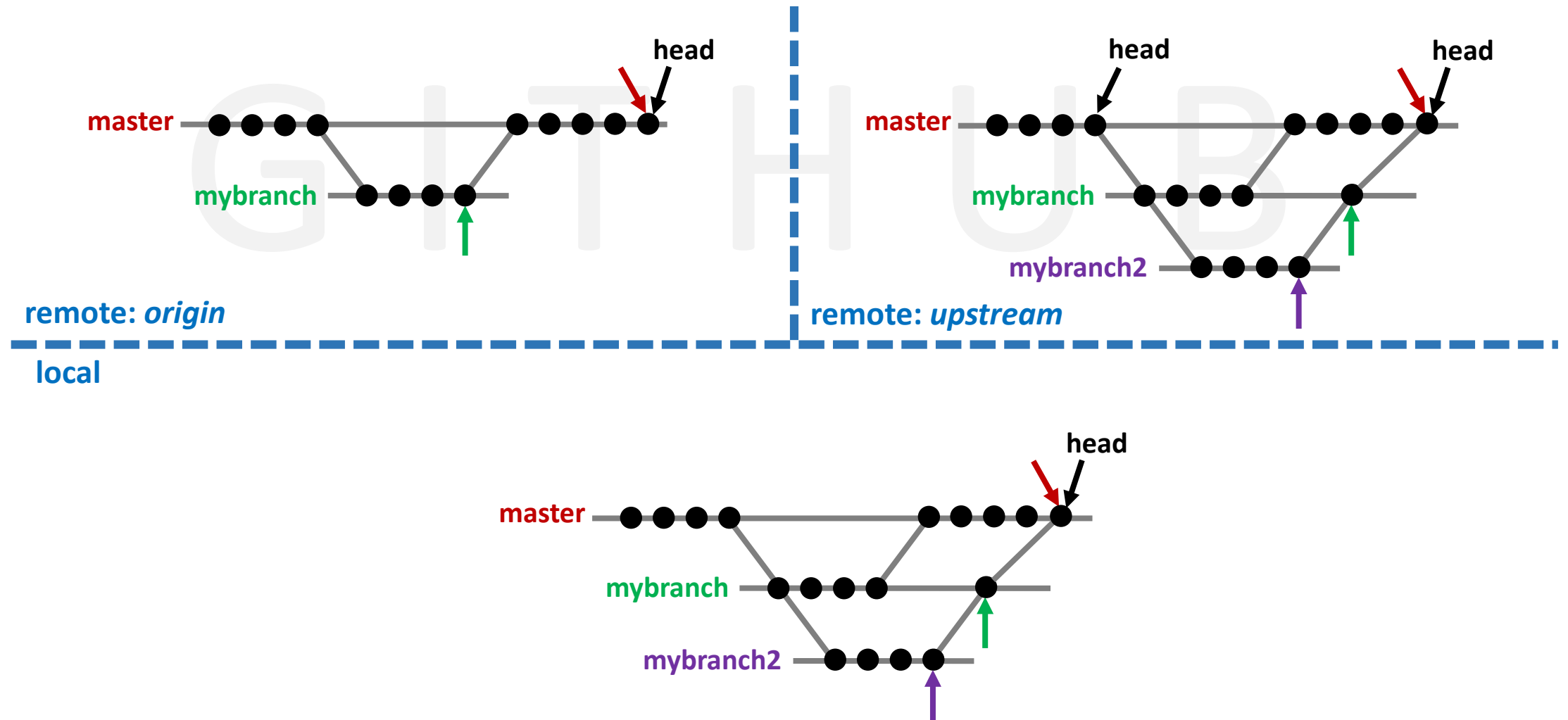


# \*\*\* Forking vs Cloning (Visualized) \*\*\*





# \*\*\* Forking vs Cloning (Visualized) \*\*\*



# 5. Getting Set-Up with Git

1. If you haven't already, you will need to do the following:
  - Install Git on your computer
  - Create an account on GitHub
2. Fork a repository on GitHub, make changes, and generate a pull-request
3. Create a public/private repository on GitHub and practice the Git workflow

## 6. Topics Not Covered...

The topics listed below slightly **advanced** and have not been covered. They considered **advanced** because they involve **changing history**. They can have **dangerous** consequences if applied improperly.

- **Rebasing** or **squashing** commits
- **Resetting** commits (at various levels)
- **Force** pushing commits to branches in remote repositories

**NOTE:** The above topics are not necessary to use Git, although they may provide some convenience in certain situations... All required topics have already been addressed in this presentation.

# Useful References

- **Official Git documentation**

<https://git-scm.com/doc>

- **Git downloads (Windows / Mac)**

<https://git-scm.com/downloads>

- **Git/GitHub Guide: A Minimal Tutorial**

[https://kbroman.org/github\\_tutorial](https://kbroman.org/github_tutorial)

- **A statistician's initial experiences of Git/GitHub**

<https://thestatsgeek.com/2015/05/16/a-statisticians-initial-experiences-of-gitgithub>



Questions?

