Git –
Git Remote - GitHub

Persistent University

Key Learning Points

- Setup GitHub Account
- Adding and Cloning remote repository
- Push, Fetch, Merge remote repository
- Creating, Tracking and Deleting remote branches

Git Remote

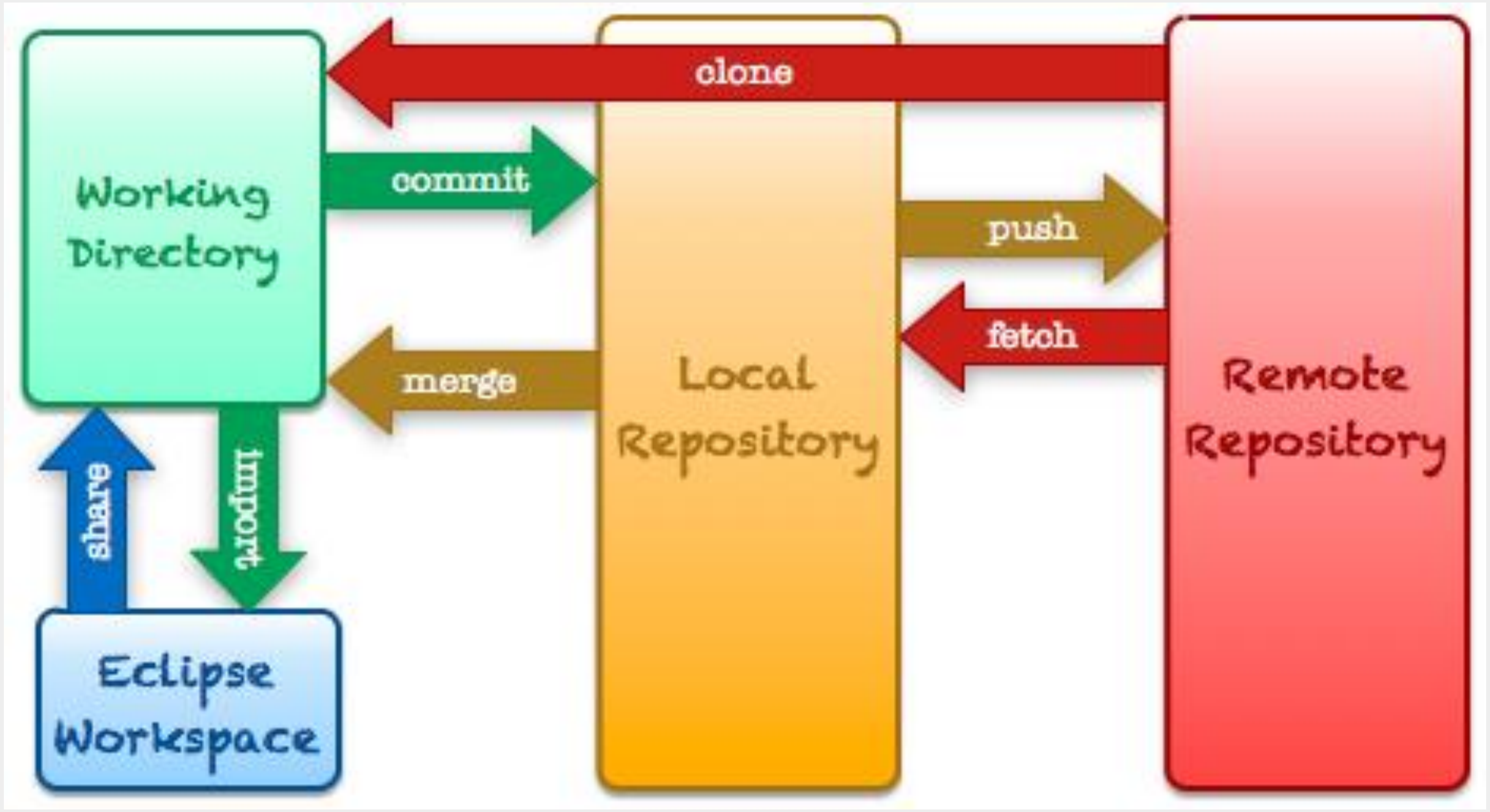
Adding Remote…!!!

- This remote repository sort of a **central clearing house** for all of these different changes that are going on.
- That remote server is just simply a Git repository.
- Remember that Git is distributed version control, there's no real difference between the different repositories,
- So there's not a big difference between the server and our computer or the client.
- This one repository(**Remote**) as the place where we all **sync up** our changes.
- We are going to use a **GitHub** as a Remote Repository.

What is GitHub?

- GitHub is how people build software
- With a community of million people, **developers** can **discover, use, and contribute** to projects using a powerful **collaborative development** workflow.
- GitHub is a **code hosting platform** for **version control** and collaboration.
- It lets you and others **work together** on projects from **anywhere**.
- GitHub is now the largest **online storage space of collaborative works** that exists in the world.

This is What We're Gonna Do…!!!



Persistent

Steps for Remote…!!!

1. Setup a GitHub Account
2. Create Repository
3. Add Remote or
4. Copy Remote(Clone)
5. Push Changes to Remote(Push)
6. Check Changes in Remote(Fetch)
7. Merge in Fetched Changes(Merge)

OR

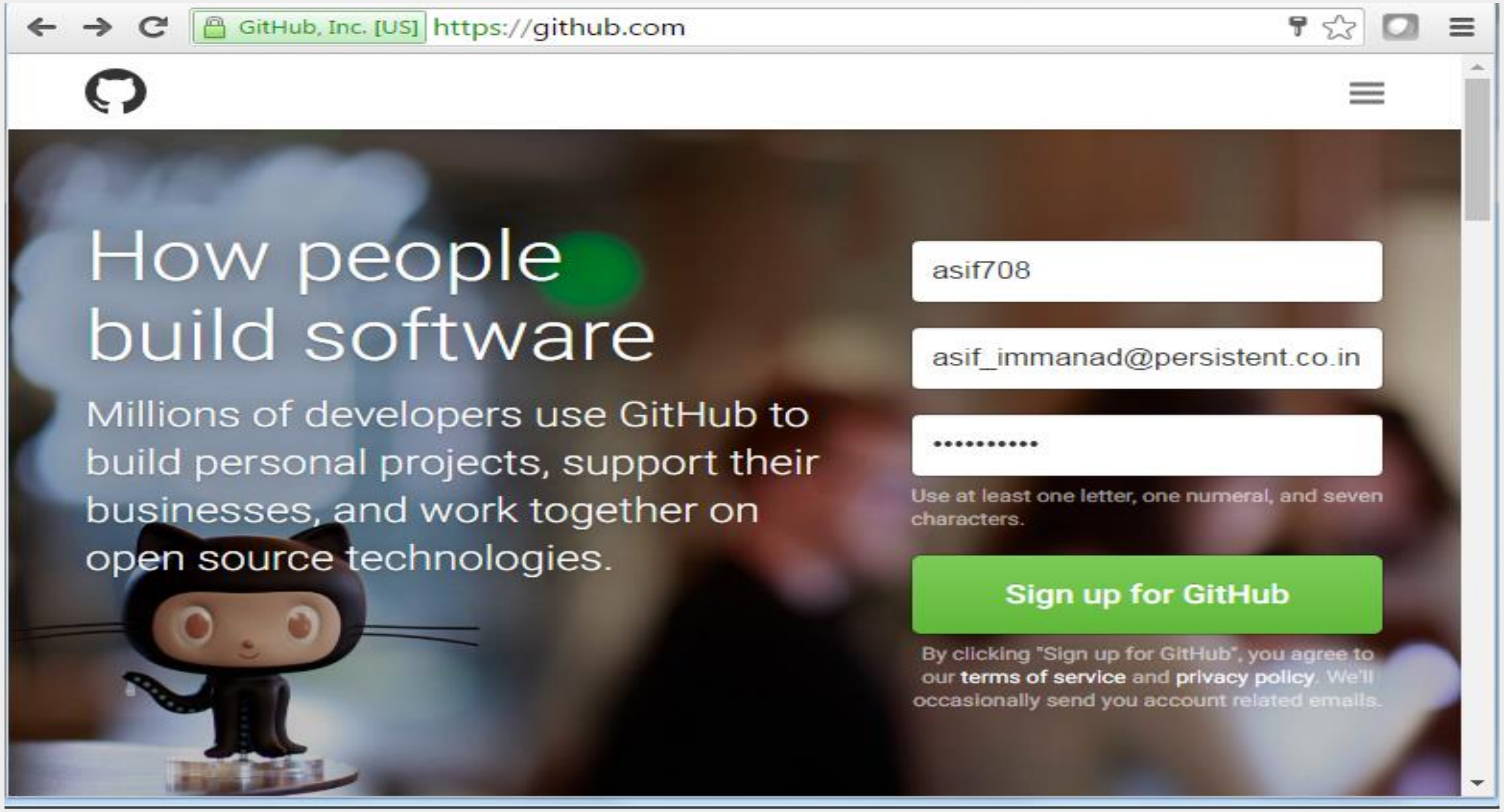
Pull to Local
(Pull = Fetch + Merge)

We may also perform :-

- Creating Remote Branch
- Checking Out Remote Branch
- Deleting Remote Branch

Setup GitHub Account

Visit https://github.com/ and Signup



How people build software

Millions of developers use GitHub to build personal projects, support their businesses, and work together on open source technologies.

asif708

asif_immanad@persistent.co.in

•••••••••••

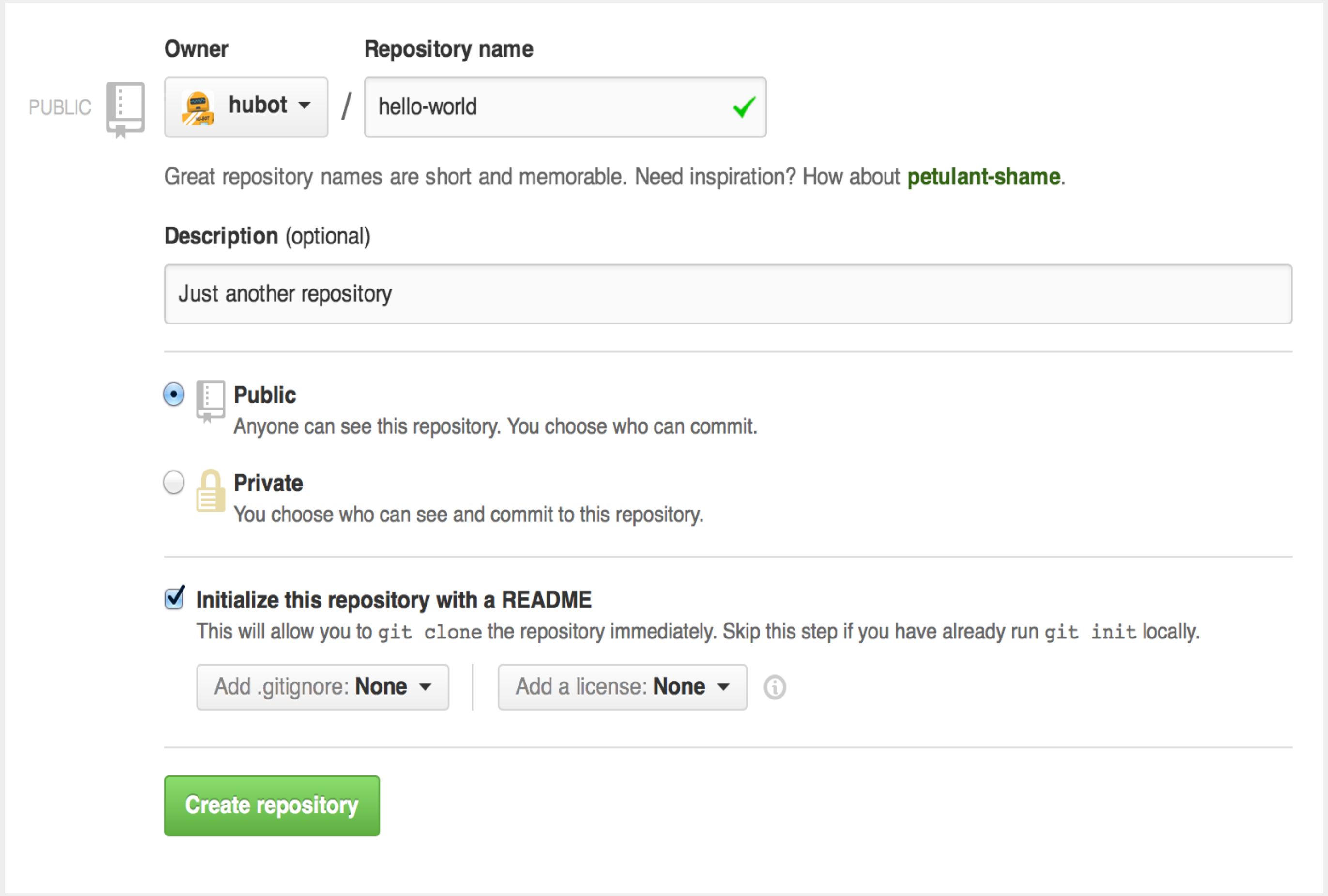
Use at least one letter, one numeral, and seven characters.

By clicking "Sign up for GitHub", you agree to our **terms of service** and **privacy policy**. We'll occasionally send you account related emails.

Create Repository

1. In the upper right corner, click + and then select New repository.
2. Name your repository hello-world.
3. Write a short description.
4. Select Initialize this

repository with a README


PUBLIC

Owner hubot / **Repository name** hello-world ✓

Great repository names are short and memorable. Need inspiration? How about **petulant-shame**.

Description (optional)

Just another repository

[✓ **Public**] Anyone can see this repository. You choose who can commit.

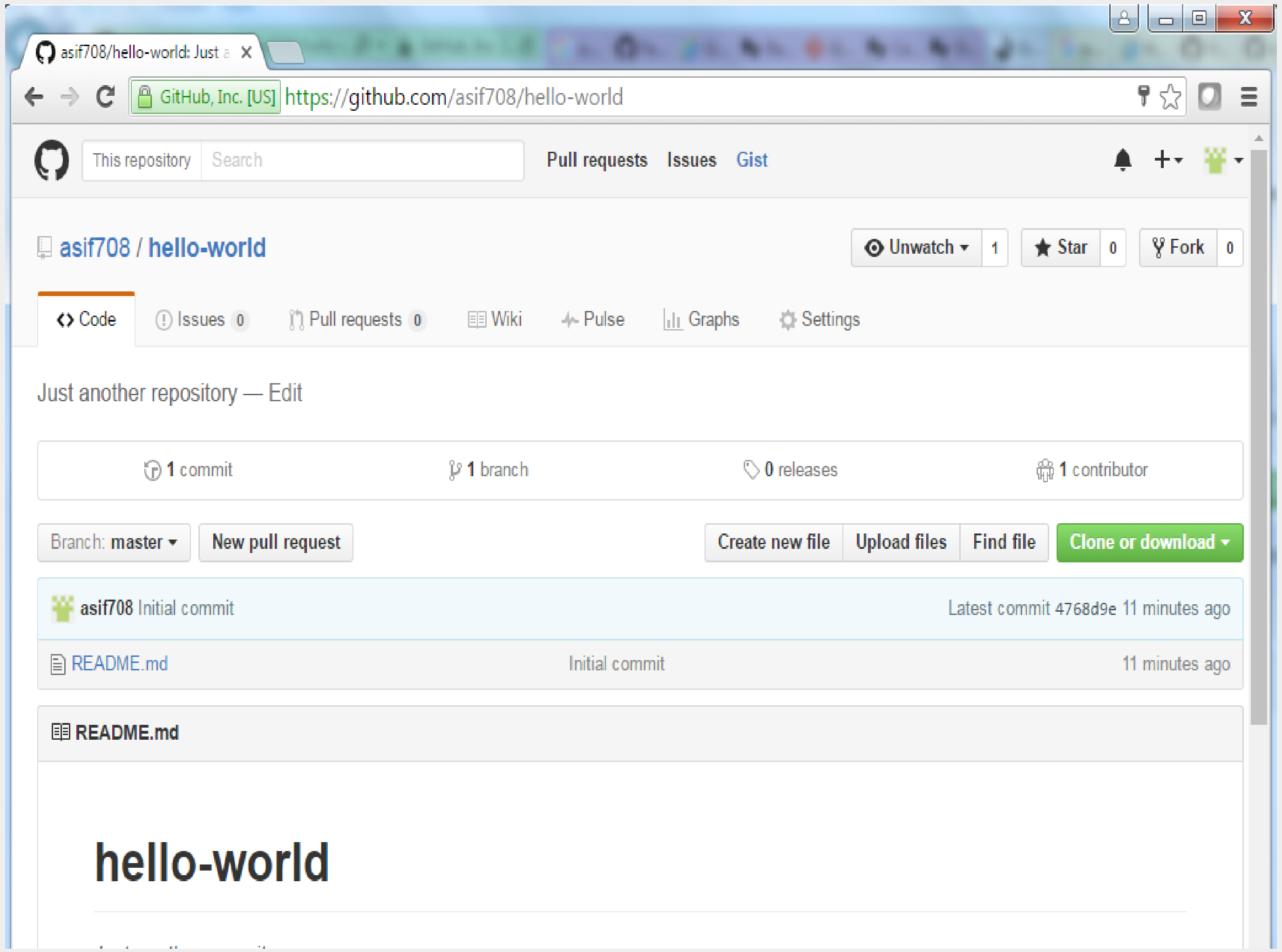
Private You choose who can see and commit to this repository.

[✓ **Initialize this repository with a README**]
This will allow you to `git clone` the repository immediately. Skip this step if you have already run `git init` locally.

Add .gitignore: **None** | Add a license: **None**

Hello-World Repository

- This window helps you to perform operations on remote repository directly.
- Here you can:-
  - Create a file
  - Edit a file
  - Upload a file
  - Create a Branch and even ***Clone or Download*** Repository



Just another repository — Edit

asif708 Initial commit Latest commit 4768d9e 11 minutes ago

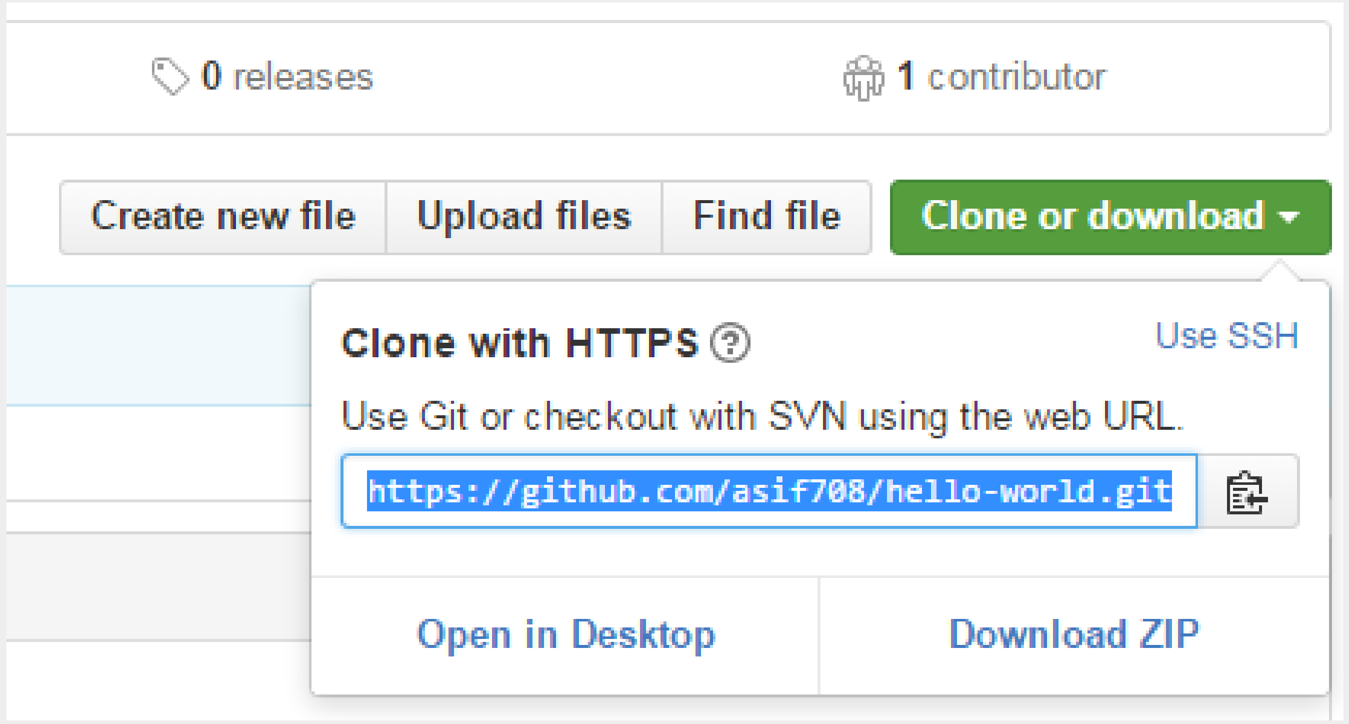
README.md Initial commit 11 minutes ago

README.md

hello-world

Let’s Add Remote

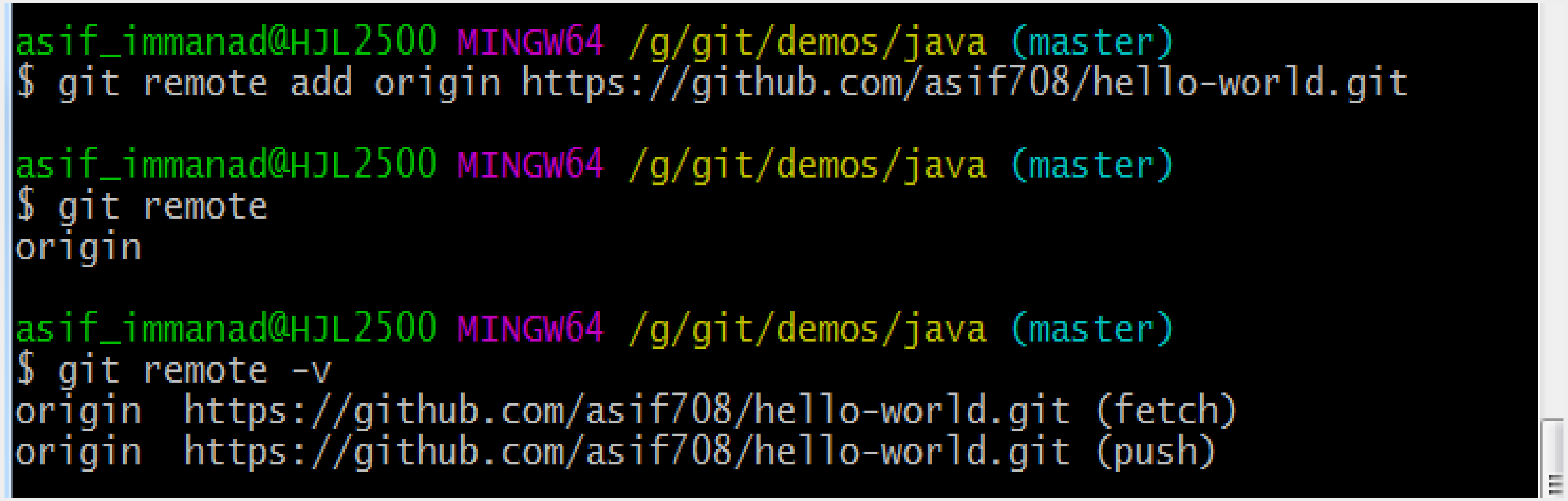
- Once we are done with setting up remote, we are ready to add remote to our local repository.
- Click on Clone to get the URL of your remote Repository.
- Copy the URL and use :-
  - **git remote add <convention> <URL>**



Use Git or checkout with SVN using the web URL.

```
https://github.com/asif708/hello-world.git
```

Adding Remote



```
asif_immanad@HJL2500 MINGW64 /g/git/demos/java (master)
$ git remote add origin https://github.com/asif708/hello-world.git

asif_immanad@HJL2500 MINGW64 /g/git/demos/java (master)
$ git remote
origin

asif_immanad@HJL2500 MINGW64 /g/git/demos/java (master)
$ git remote -v
origin  https://github.com/asif708/hello-world.git (fetch)
origin  https://github.com/asif708/hello-world.git (push)
```

- Name a remote whatever you want
- ‘origin’ here is just a convention

```
>git remote add origin https://github.com/asif708/hello-world.git
```

`>git remote` → Get names of all the remotes

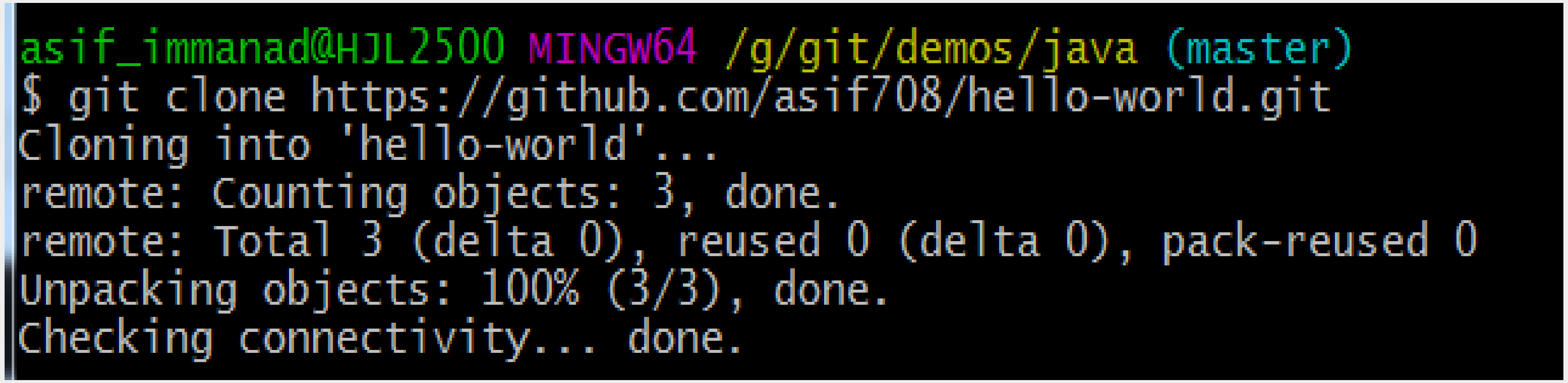
`>git remote -v` → Get names and details of all the remotes

Remote Added….Now Lets Copy

- Once remote is added we can take a complete copy of repository to local by using ***Clone*** command.

Copy Remote → Clone

- If we don’t have a local copy of the repository we can clone the repository from the remote.
- Doing this pulls the full repository locally, and sets up the remote connection information.



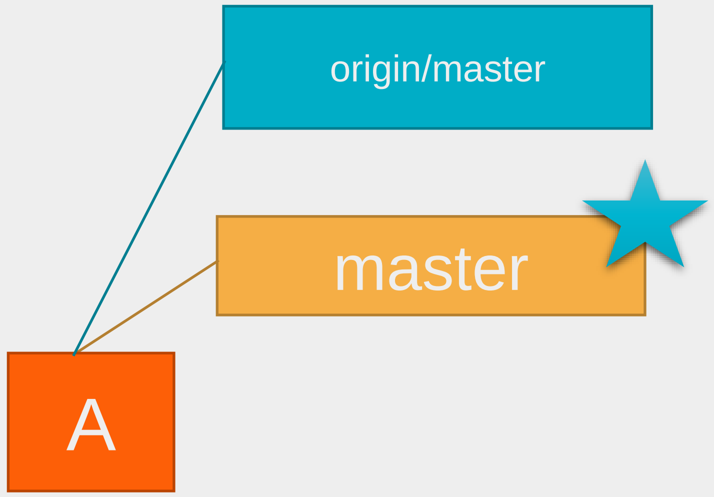
```
asif_immanad@HJL2500 MINGW64 /g/git/demos/java (master)
$ git clone https://github.com/asif708/hello-world.git
Cloning into 'hello-world'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
Checking connectivity... done.
```

Clone will auto setup the remote

```
>git clone https://github.com/asif708/hello-world.git
```

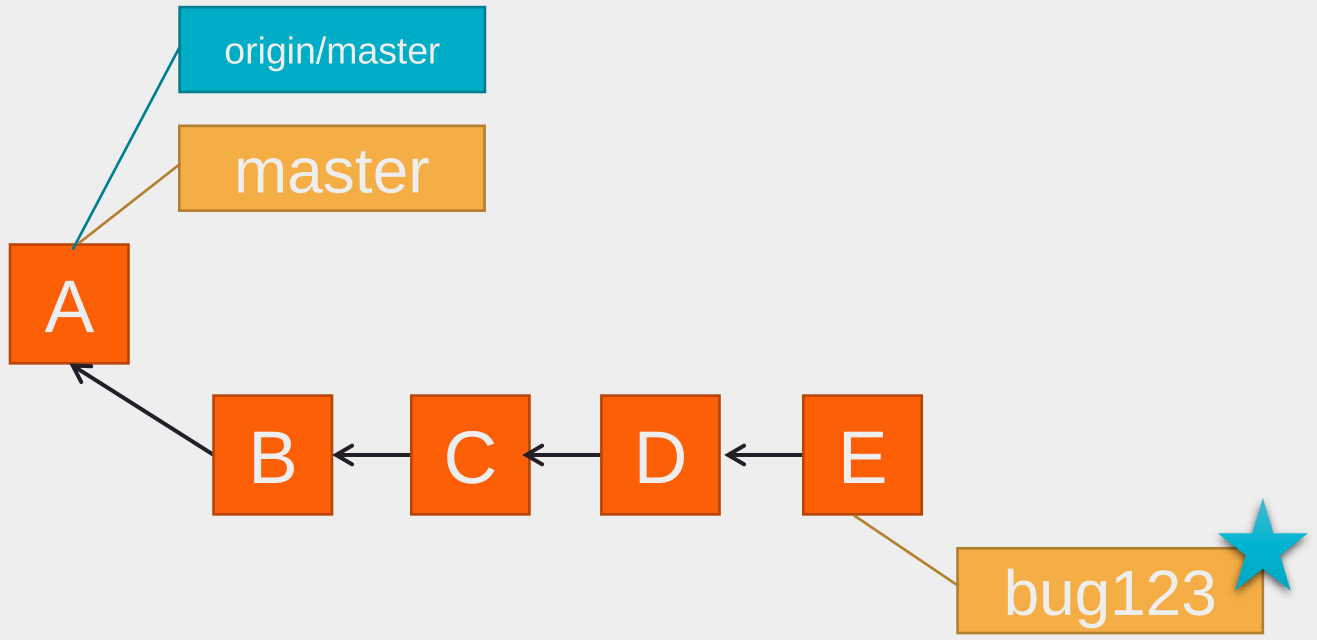
What Locally Happens After Clone?

- Here at our local repository master is the local pointer pointing to A
- Once we clone or pull remote master branch is also pointing to A.
- Origin/master is the just the new branch created locally to keep a track.



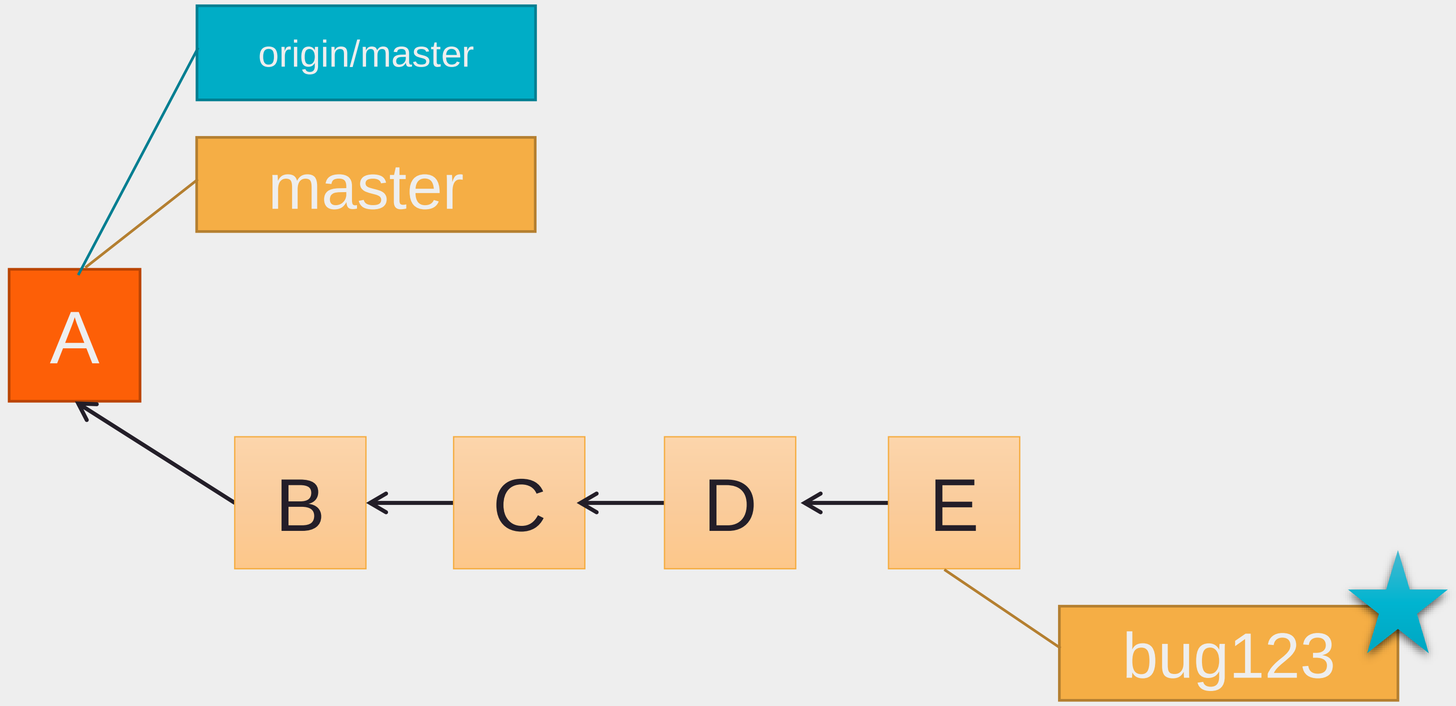
Branches Illustrated

- Now suppose we are here. We have cloned master on (A) and have been fixing bug 123 in our story branch.
- Here we are as before with our local master branch and the remote master branch both pointing at (A)



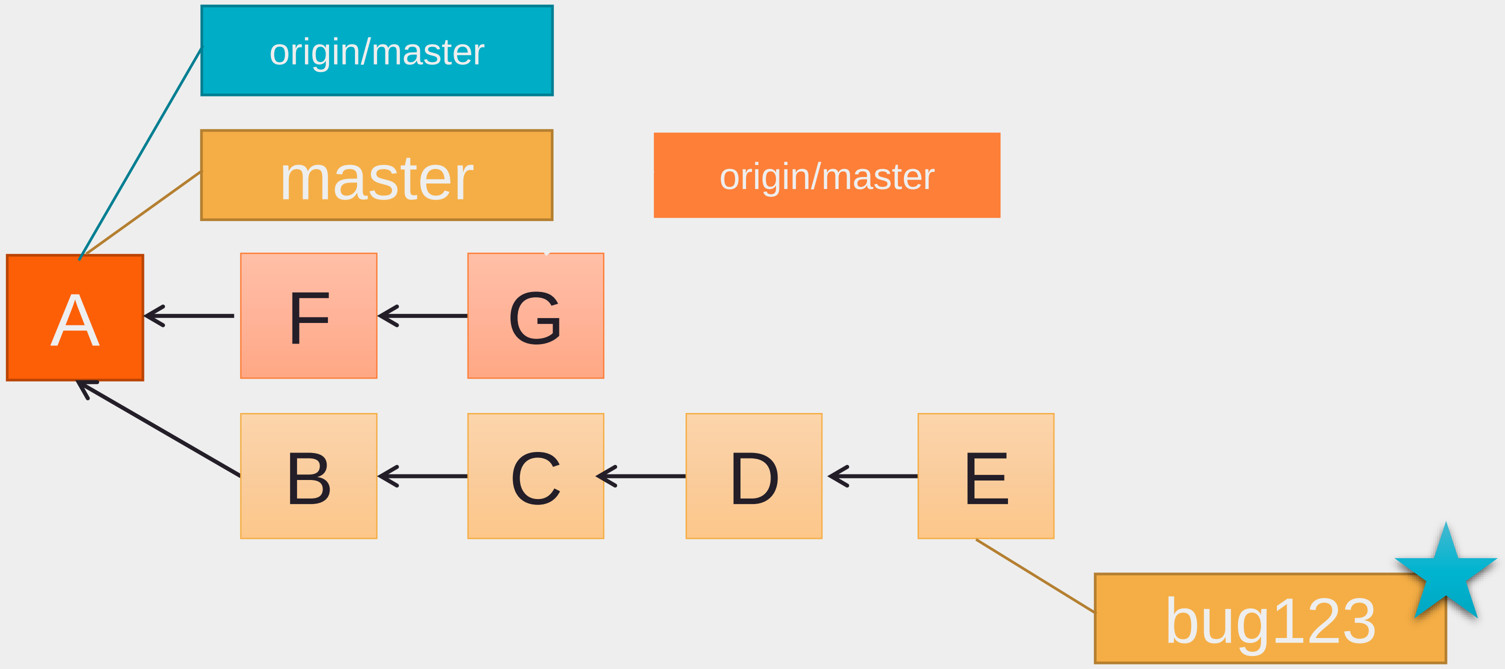
Branches Illustrated

- The highlighted changes(green) on the Bug123 branch are only known to my local machine.
- The remote server does not have these changes or the bug123 branch for that matter.



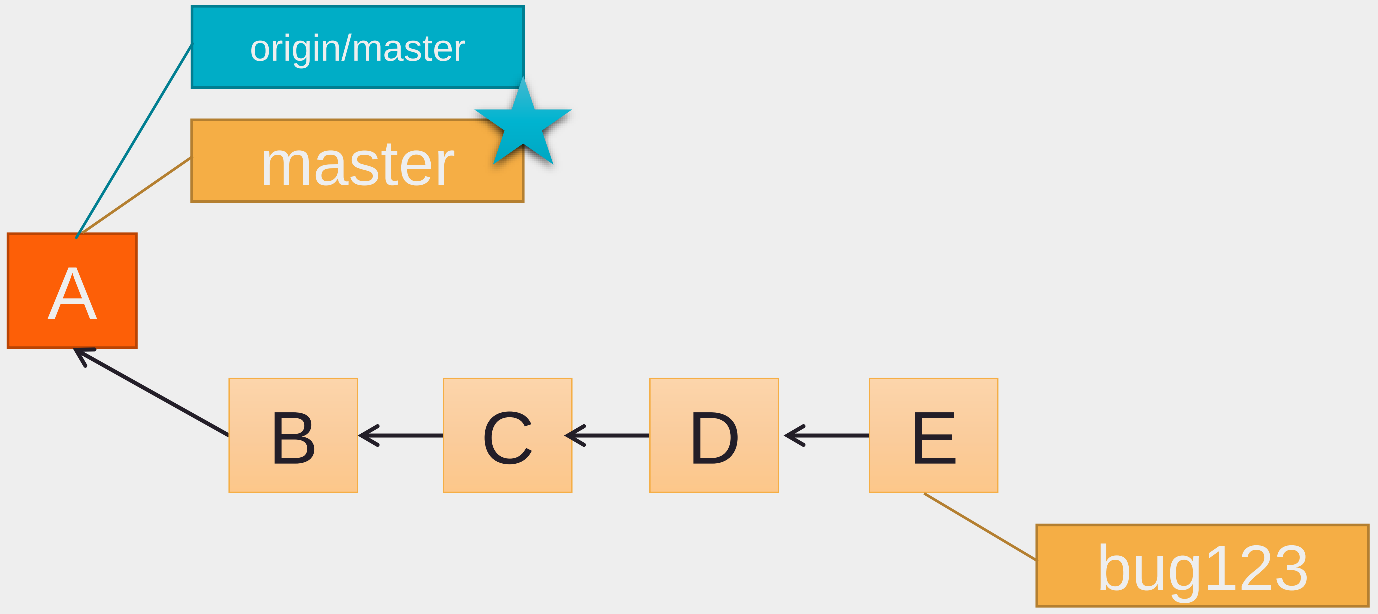
Branches Illustrated

- While Working on bug123 some more commits are made on remote which we don’t know about(Orange box)
- Orange box here is to indicate where the master pointer is on the remote server.



Branches Illustrated

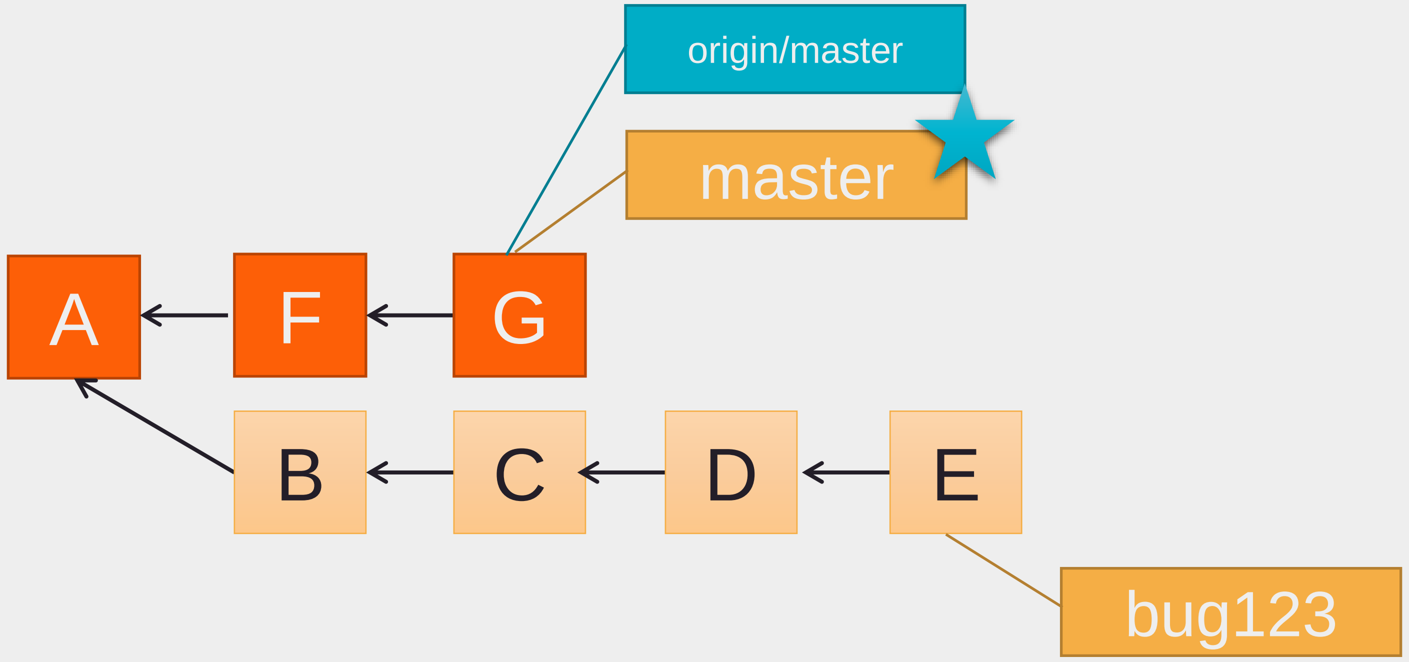
- So if this is what we know, we can update our master to catch up.
- First we checkout master which moves our current (*) to there.
- Note that we are actually on our master, not the upstream one. That is always true. But the tracking branch is also pointing to (A) at this point.



```
> git checkout master
```

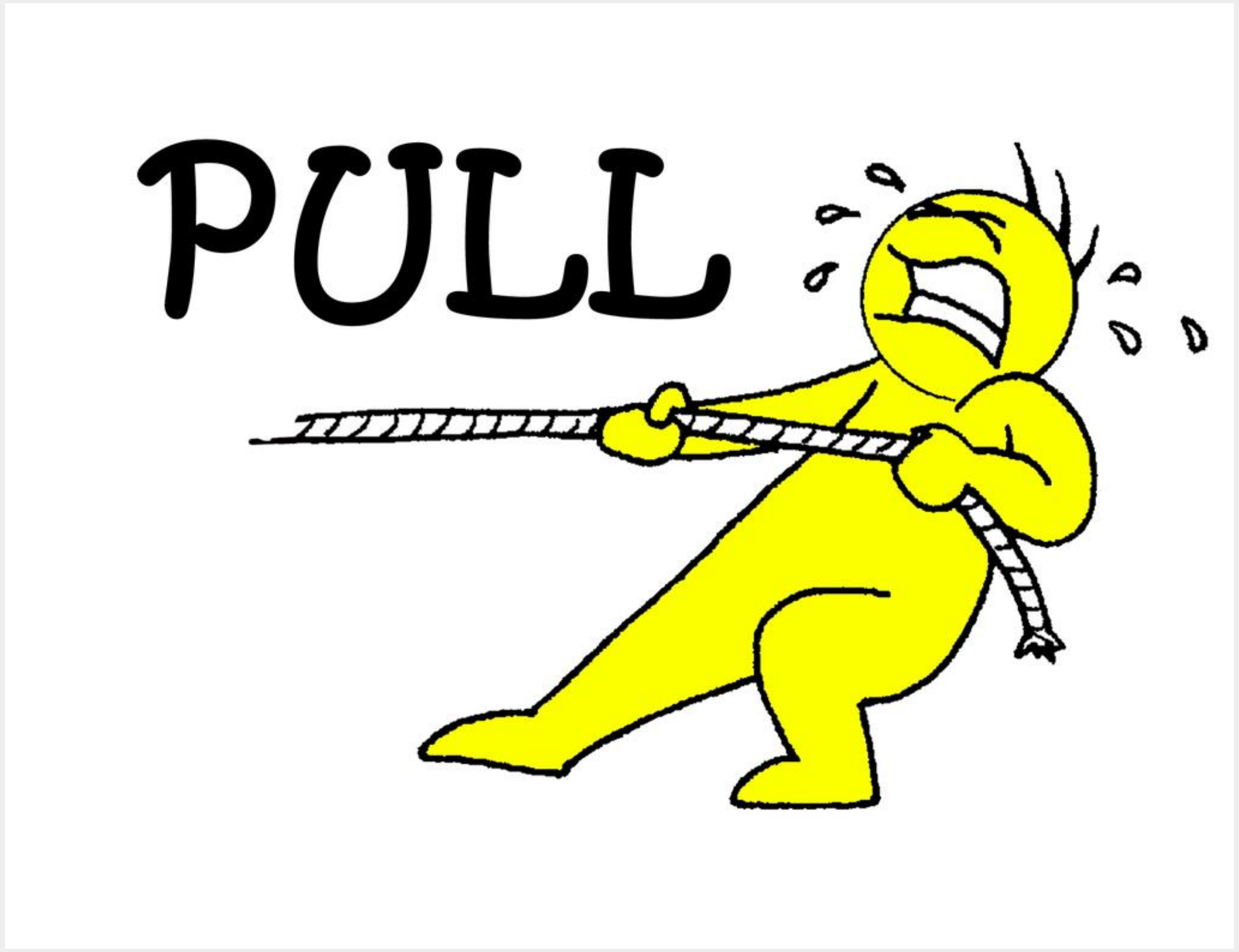
Lets Catch Up With Remote

- Now we can do a pull on the origin (our source remote) and move both along to their new place.



```
> git pull origin
```

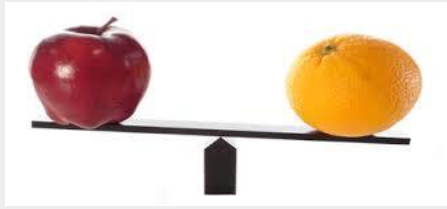
Hahn…What’s this Pull??



Pull = Fetch + Merge

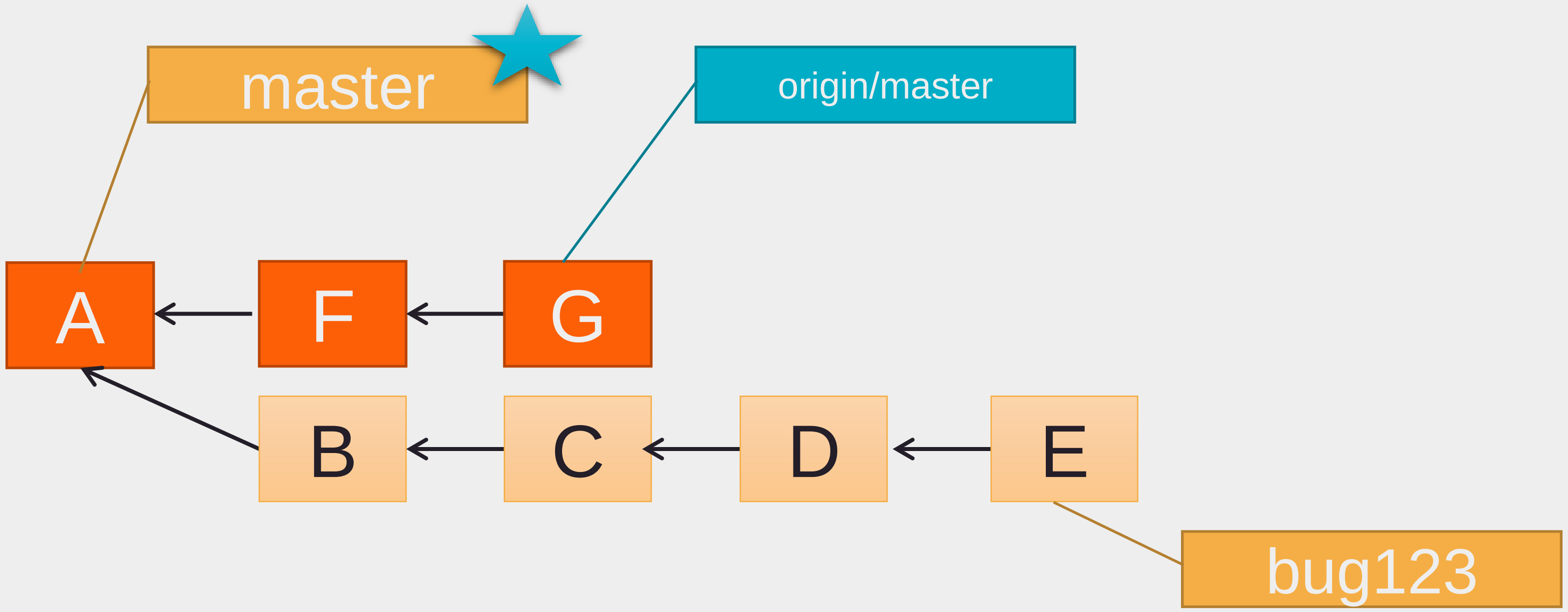
- Fetch - updates your local copy of the remote branch
- Pull essentially does a fetch and then runs the merge in one step.
- The pull command is combination of a fetch from the remote server and a merge of the changes.
- We can do these steps separately, but if we are not working on the branch we are pulling down, pull is just a nice way to get up to date.

Fetch Or Pull ?



- If we consider fetch, fetch just updates the local copy and move the remote pointer to latest.
- But note that our local master pointer is till pointing to old version .

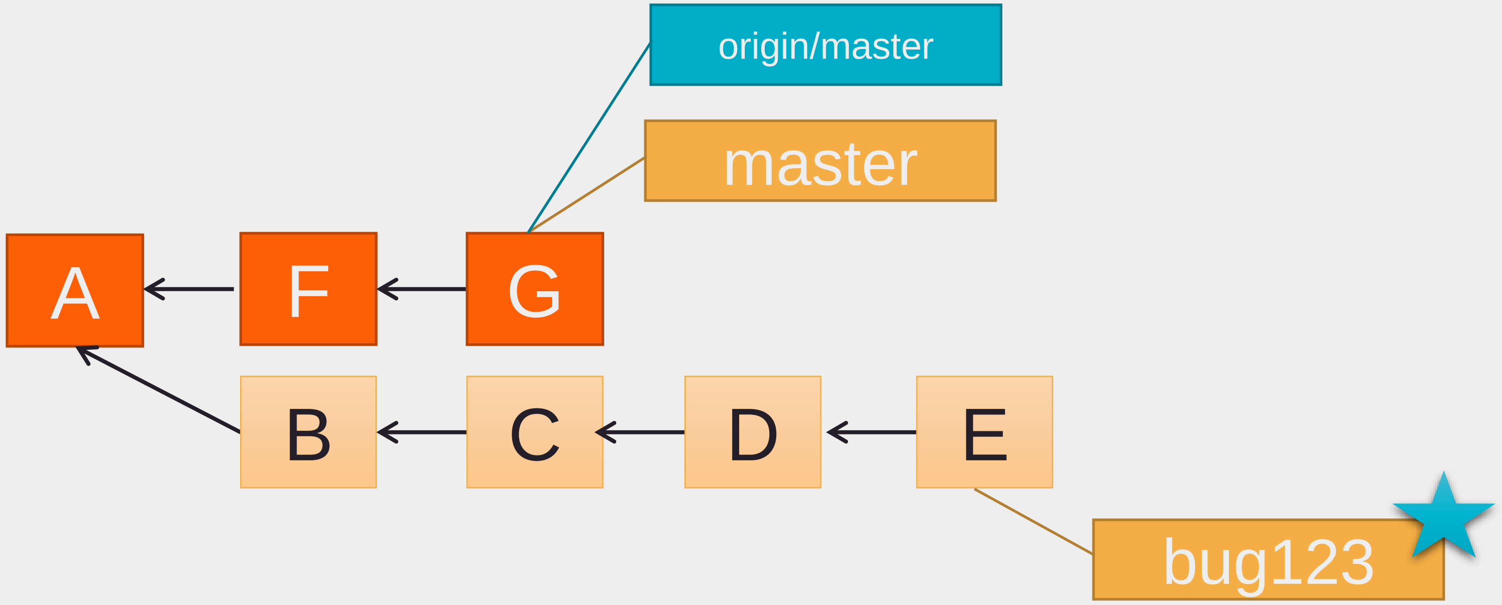
Fetch does not merge the changes



```
> git fetch origin
```

Branches Illustrated

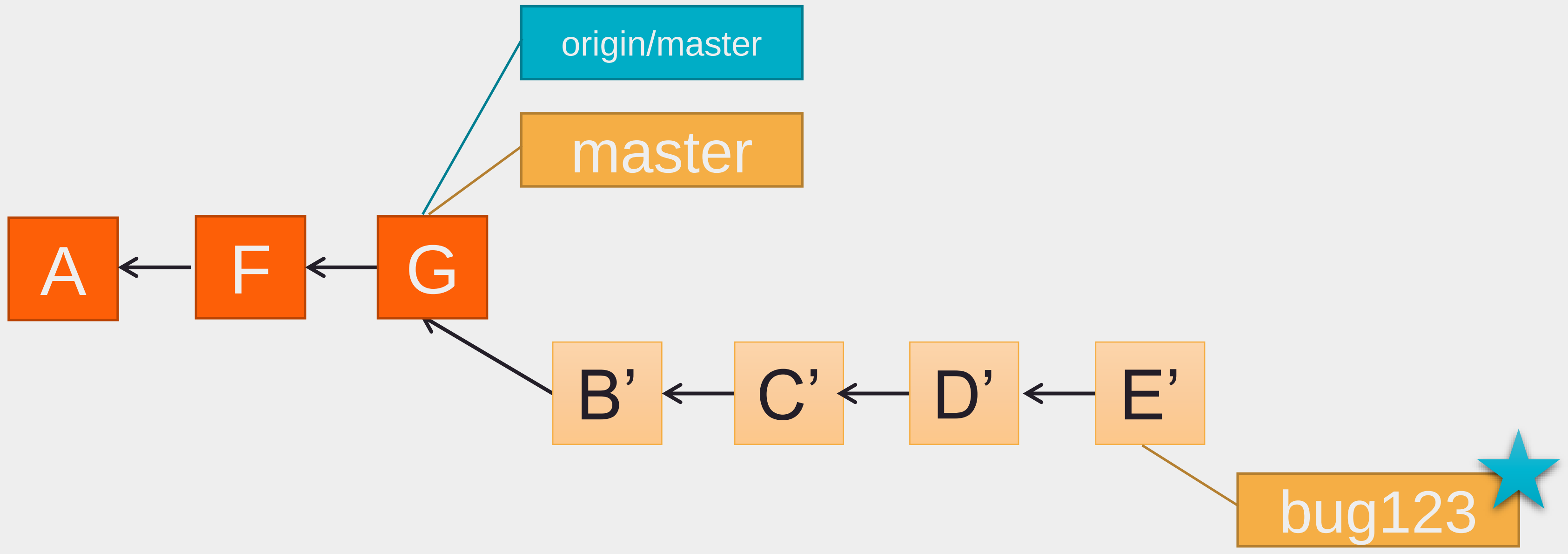
- Once we **update** our local branch, we can say that we are in **synch** with the **remote**.
- Remote branch(origin/master) and local branch both are at same place.
- So now lets go back to bug fix branch(bug123) move the pointer by checkout.



```
> git checkout bug123
```

Branches Illustrated

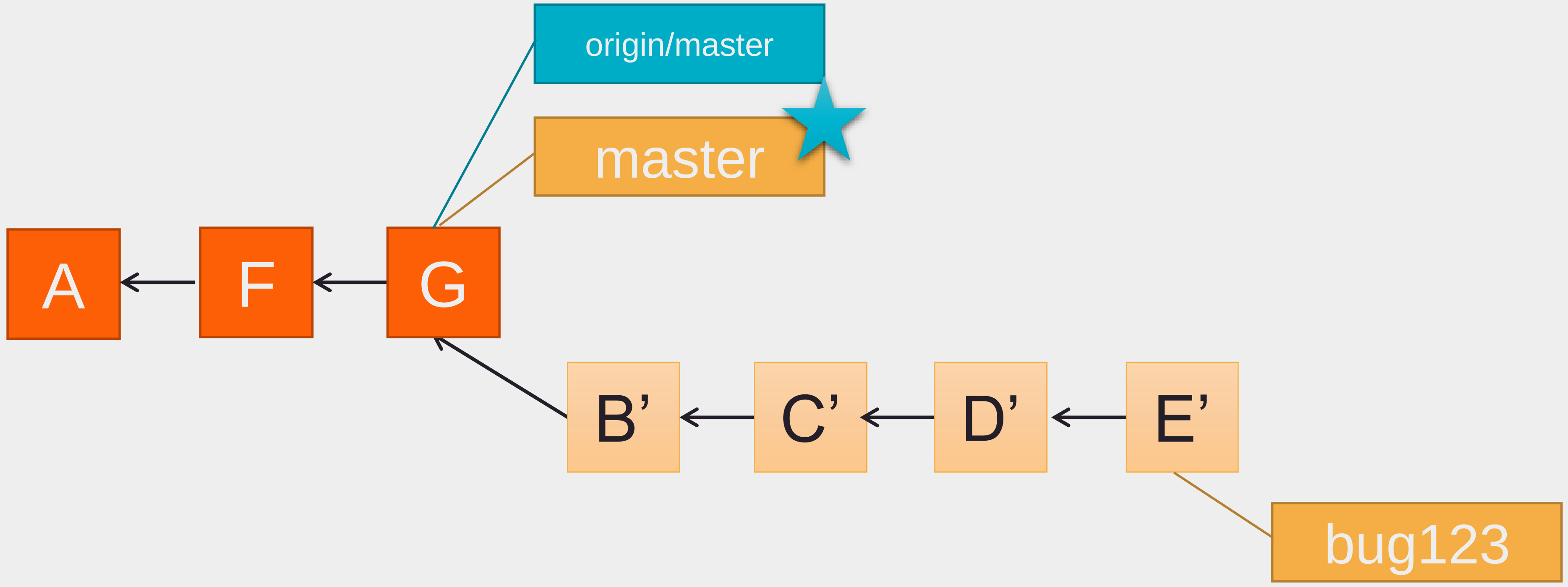
- But now we have problem to merge bug123 to master, Isn't it?
- B-C-D-E all come before F and G. Merging would create issues, right?
- ***So we use* rebase *to rewind and replay B-C-D-E after G.***



```
> git rebase master
```

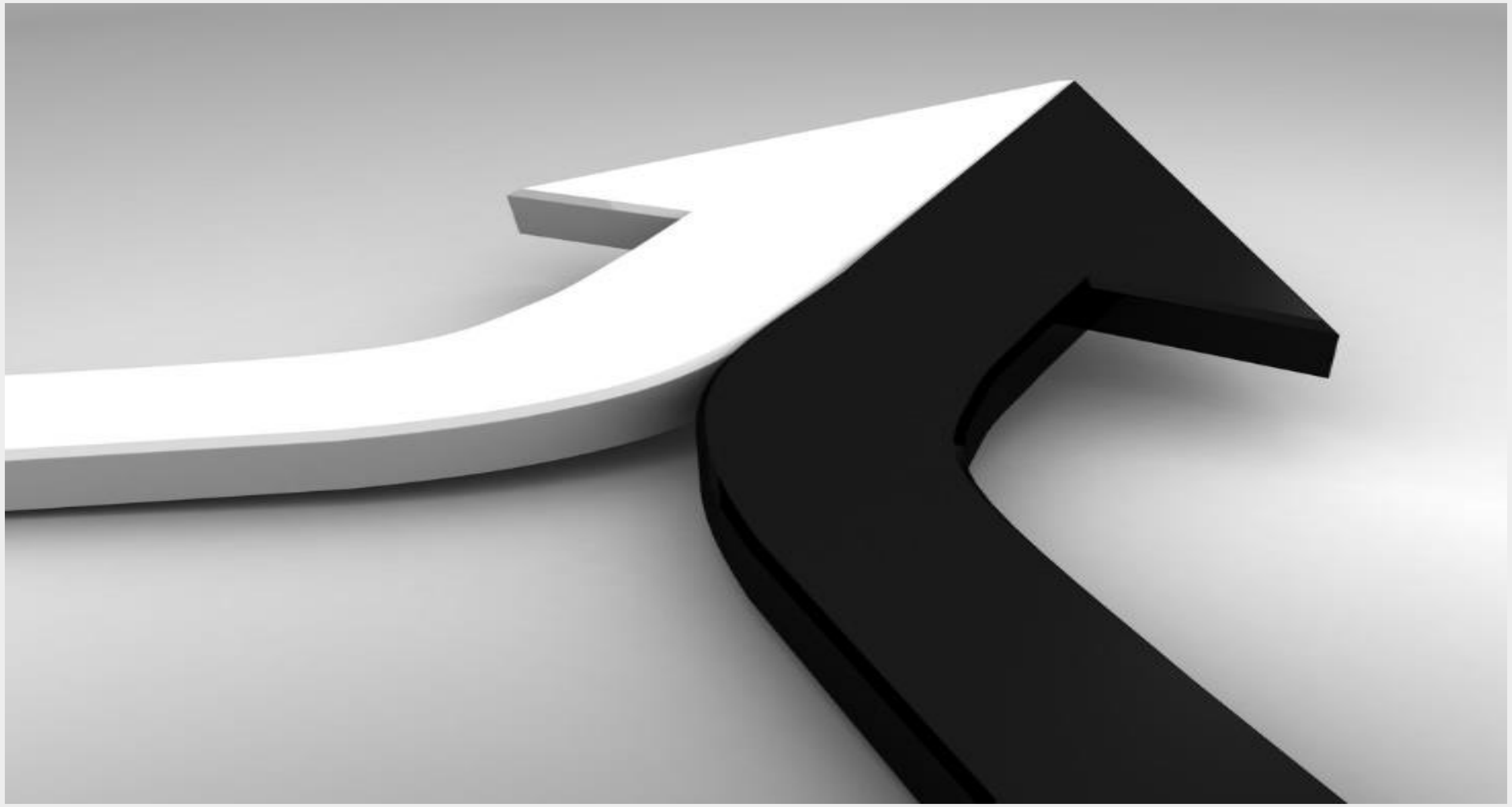
Branches Illustrated

- To merge with master we need to go back to master.
- So let’s checkout to master.



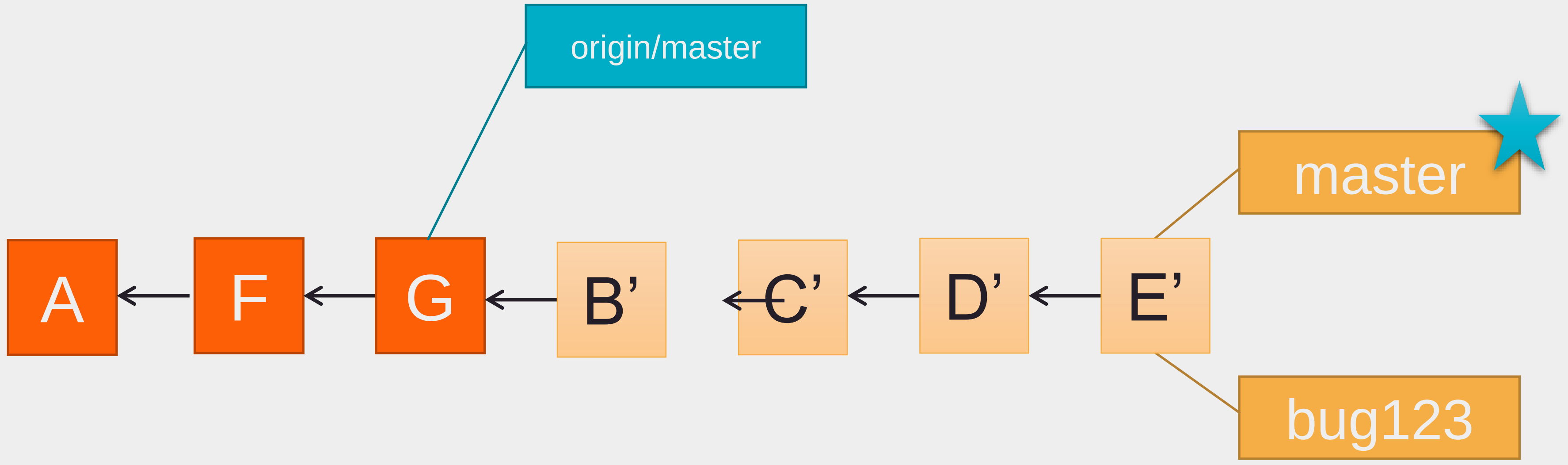
```
> git checkout master
```

And Now we MERGE…!!!



And Now we MERGE…!!!

- We already know to merge any branch with master we should checkout to master first.
- Once pointer is on master ***We Can Merge*.**
- ***Note that the remote origin/master (upstream) pointer is still "back there".***



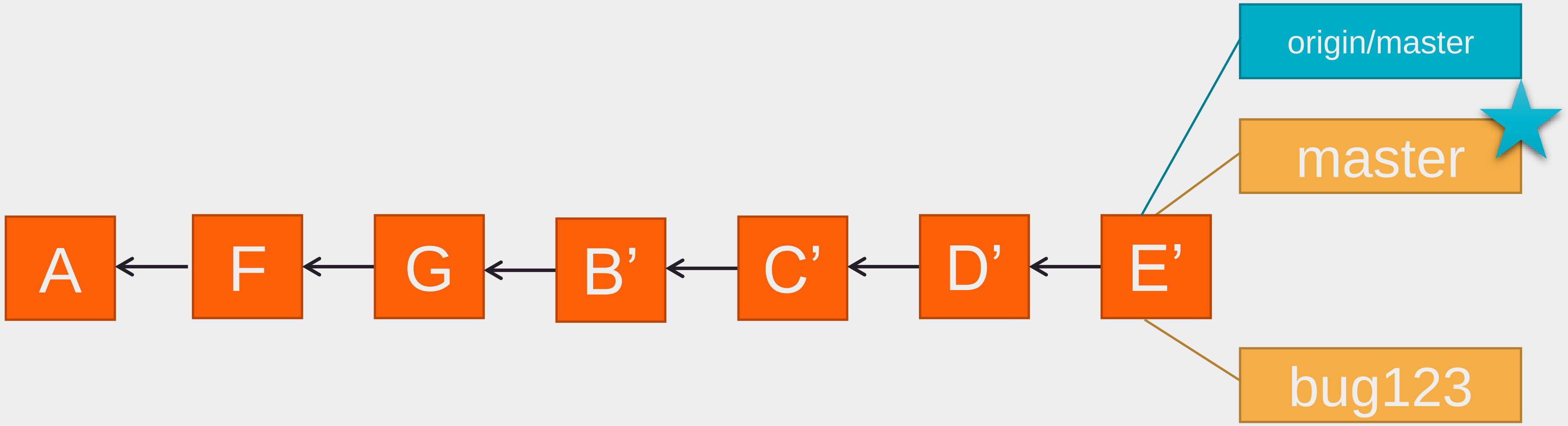
```
> git merge bug123
```

Now time to show your WORK…!!!

PUSH it on Remote…!!!

PUSH it on Remote…!!!

- And finally, because we want to publish these changes on remote repository, we ***push*** it to the remote origin branch.
- This moves the origin branch along
- Now from this we can say that we are in ***complete synch with remote*** as all our changes are pushed to remote.



```
> git push origin
```

Beware while Pushing !!!

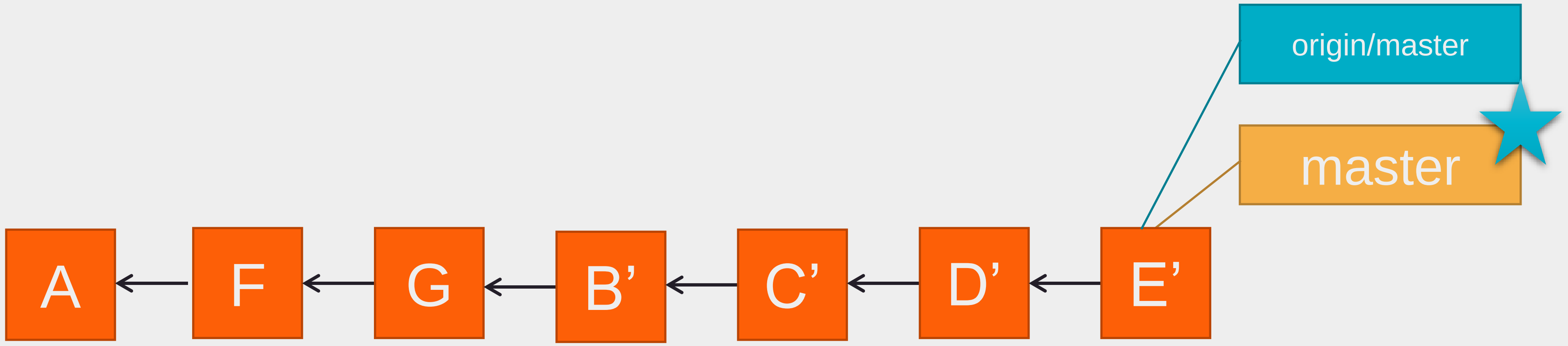
- Push will update the remote server.
- If you are **out of date**, Git will **reject** that **push**.
- Git will require you to ***merge locally, then push*** the results.
- Git will reject pushes if newer changes exist on remote.
- ***Good practice:***

Pull then Push

Persistent

Branches Illustrated

- **Delete the story branch and we're good to go**

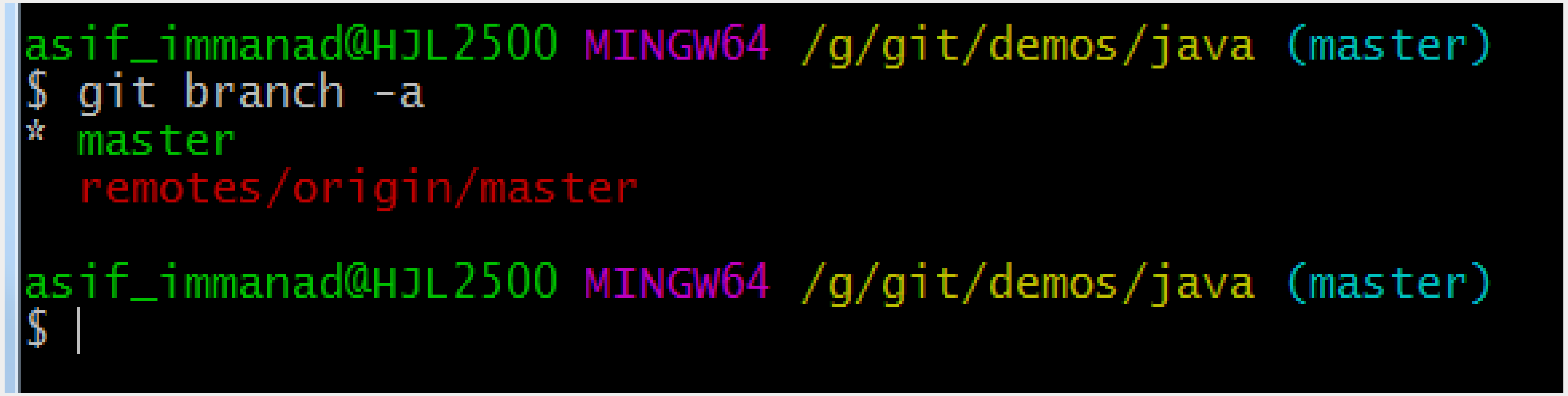


```
> git branch -d bug123
```

Working with Remote Branches…

View All Branches…!!!

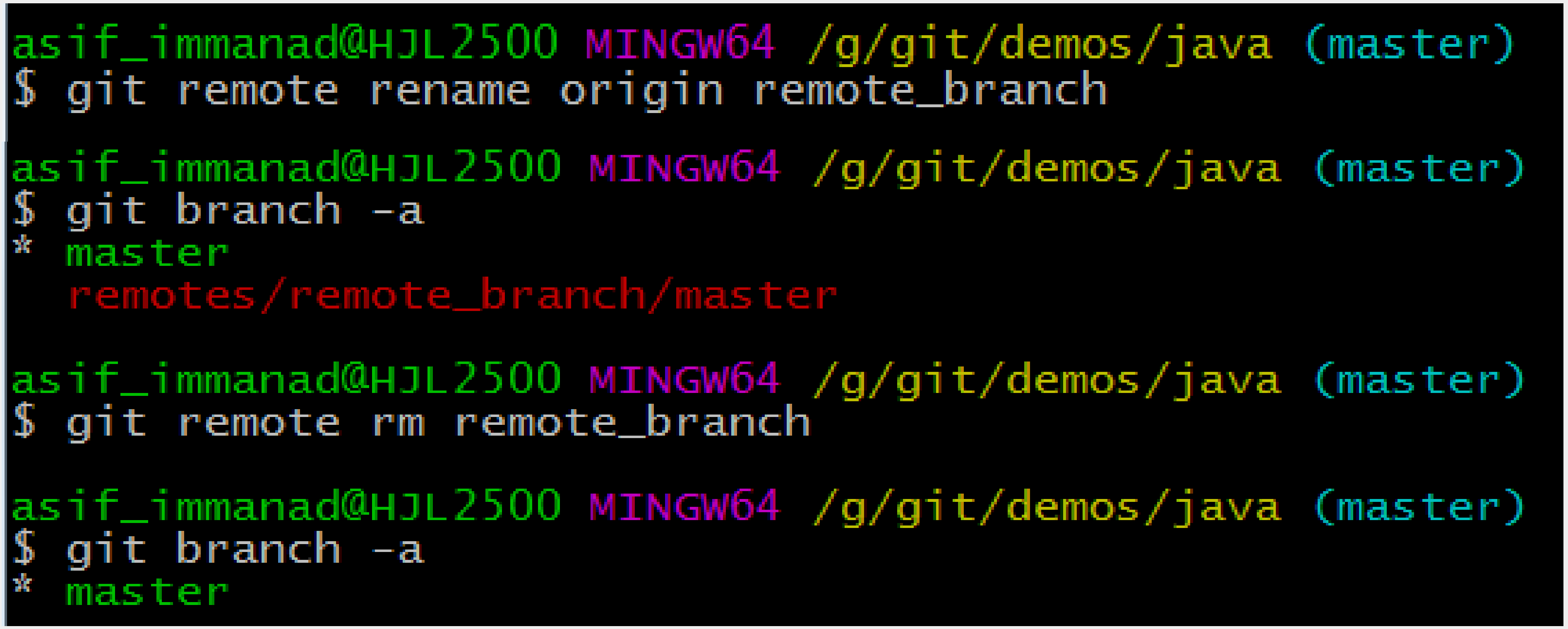
- We may check the branches before performing some operation.
- We know how to view local branches.
- While working with remote we may need to check remote branches as well.



```
asif_immanad@HJL2500 MINGW64 /g/git/demos/java (master)
$ git branch -a
* master
  remotes/origin/master

asif_immanad@HJL2500 MINGW64 /g/git/demos/java (master)
$ |
```

```
> git branch -a
```

Rename and Remove Remote Branch…!!!

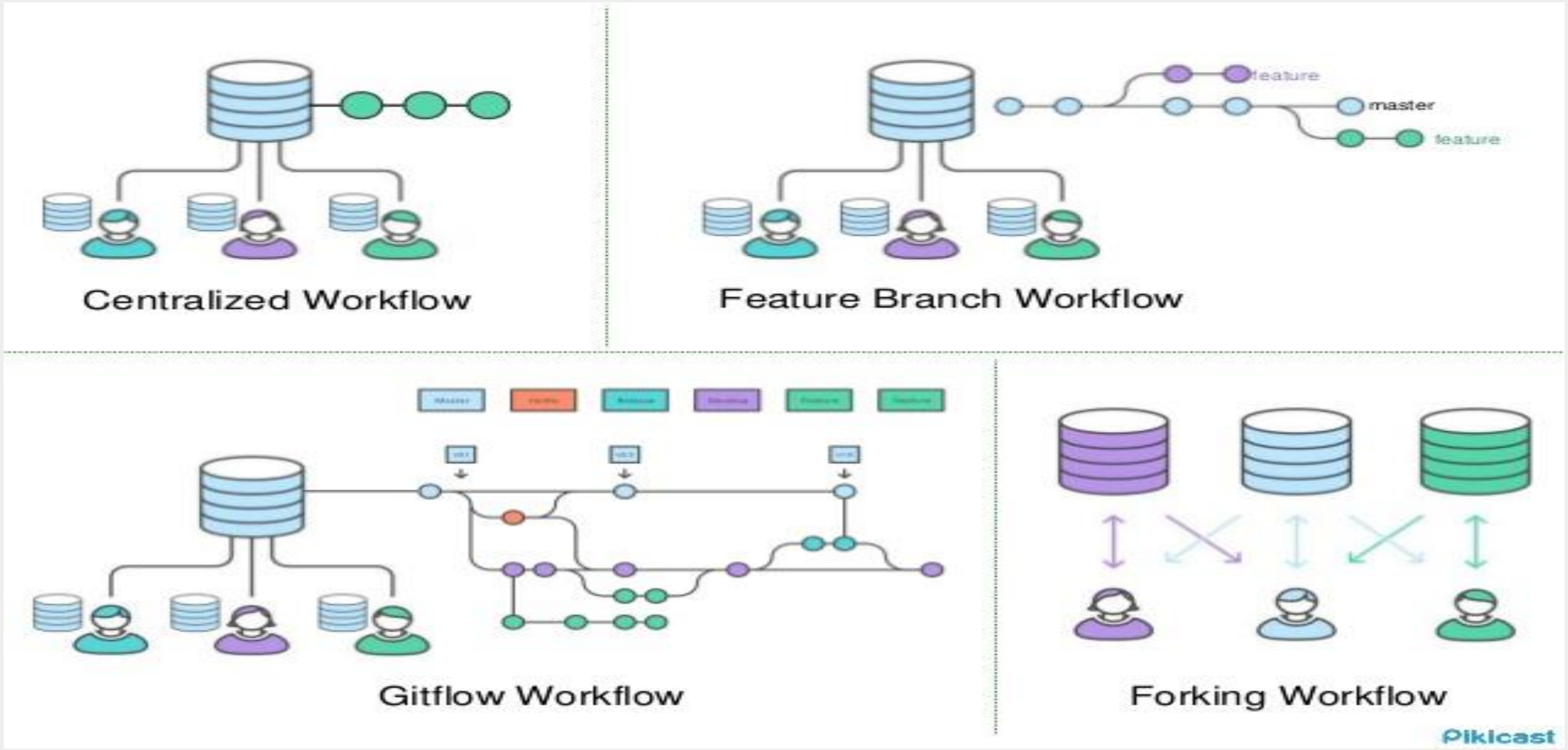


```
asif_immanad@HJL2500 MINGW64 /g/git/demos/java (master)
$ git remote rename origin remote_branch

asif_immanad@HJL2500 MINGW64 /g/git/demos/java (master)
$ git branch -a
* master
  remotes/remote_branch/master

asif_immanad@HJL2500 MINGW64 /g/git/demos/java (master)
$ git remote rm remote_branch

asif_immanad@HJL2500 MINGW64 /g/git/demos/java (master)
$ git branch -a
* master
```

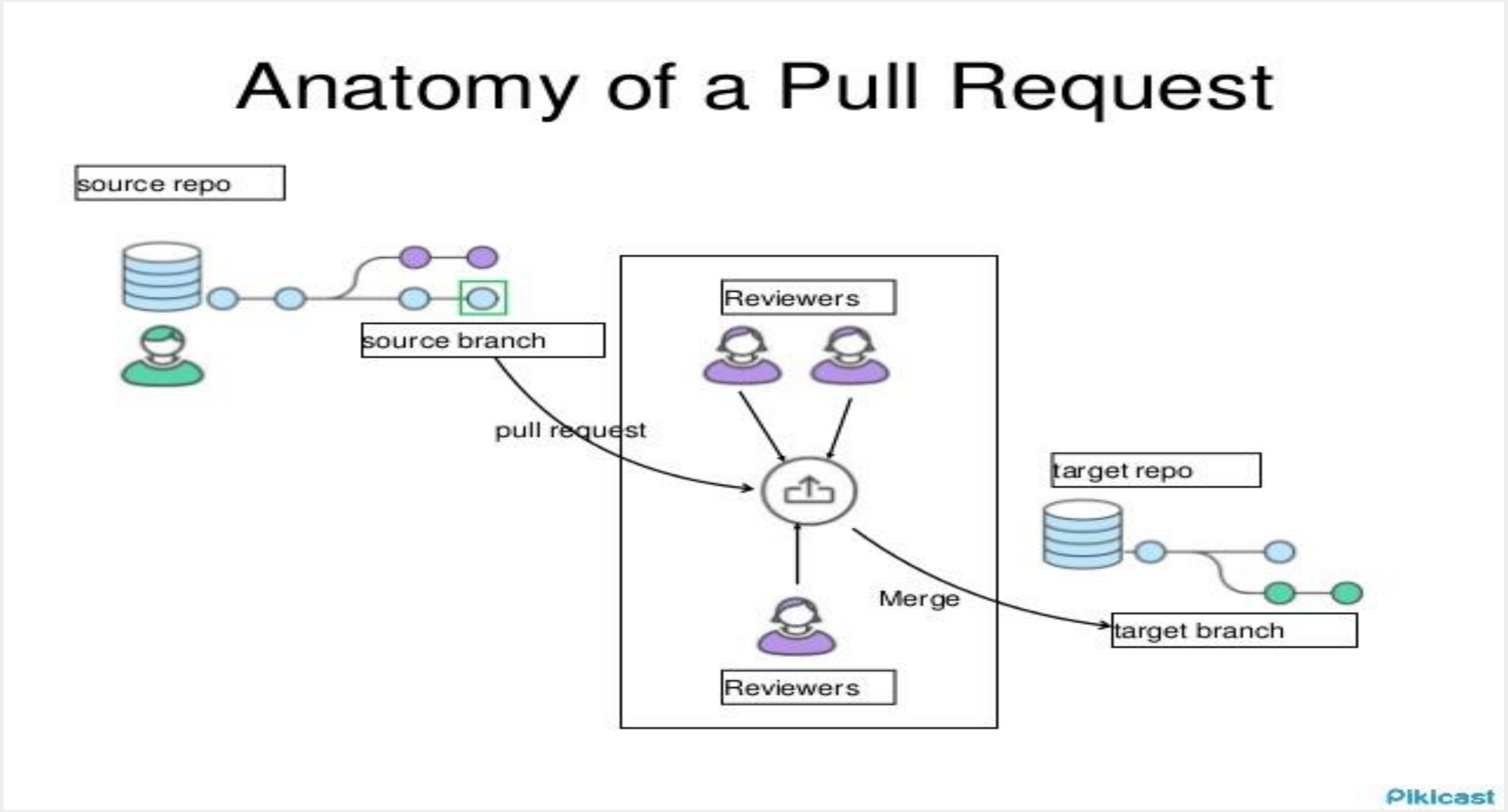
> git remote rename <old_name> <new_name>

> git remote rm <remote_branch_name>

Git Branching Workflows



Pull Request in GitHub



Things to Remember…!!!

Adding a remote makes it easy to share

Pulling from the remote often helps keep you up to date

Update your local repository regularly, It makes easier to see what is going on upstream(Remote).

FAQ

- How to setup git remote?
- Adding git remote
- Push to remote
- Pull from remote
- Workflows with remote
- Pull Request

Summary

With this we have come to an end of our the session, where we discussed about

- Working with remote repositories(GitHub)

Reference Material : Websites & Blogs

- https://git-scm.com/book/en/v1/Git-Branching
- https://git-scm.com/book/en/v2/Git-Branching-Basic-Branching-and-Merging
- http://learngitbranching.js.org/
- https://www.digitalocean.com/community/tutorials/how-to-use-git-branches
- https://github.com/Kunena/Kunena-Forum/wiki/Create-a-new-branch-with-git-and-manage-branches

Reference Material : Books

- ***Pro Git***
  - By Scott Chacon and Ben Straub
  - *Publisher:* Apress

- ***Version Control with Git***
  - By Jon Loeliger, Matthew McCullough
  - Publisher: O'Reilly Media

Key contacts

Git Interactive :-

Vaishali Khatal
vaishali_khatal@persistent.co.in

Asif Immanad
asif_immanad@persistent.co.in

Vice President

Shubhangi Kelkar
shubhangi_kelkar@persistent.co.in

Persistent

Thank you!

Persistent University