Introducing Git version control into your team

Mark Groves
mgroves@microsoft.com
@mgroves84

patterns & practices Symposium 2013

History

Created by Linus Torvalds for work on the Linux kernel ~2005

Some of the companies that use git:

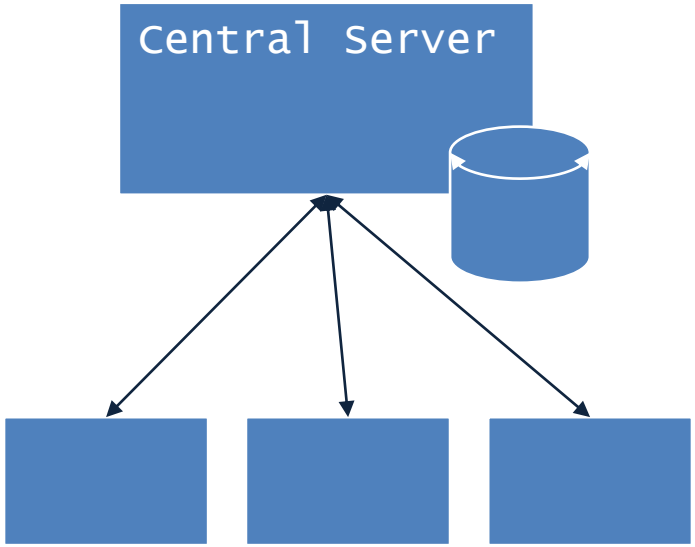
Strength of Git

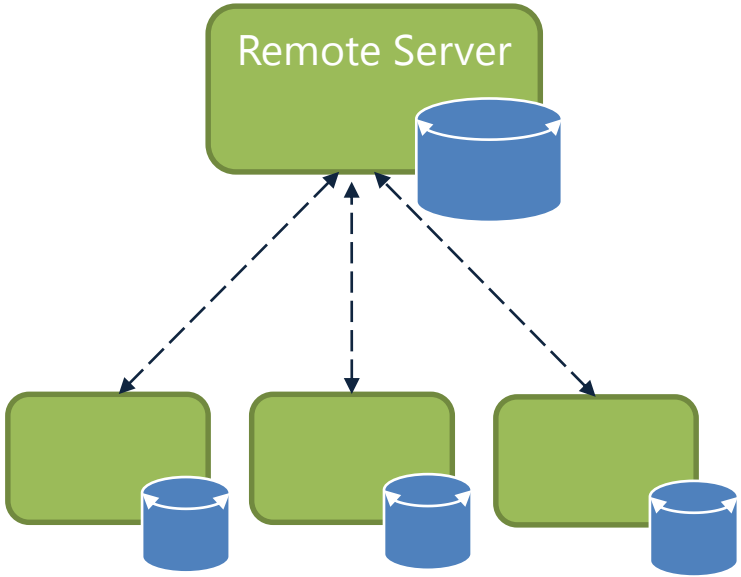
Everyone has the complete history

Everything is done offline

…except `push/pull`

Centralized VC vs. Distributed VC

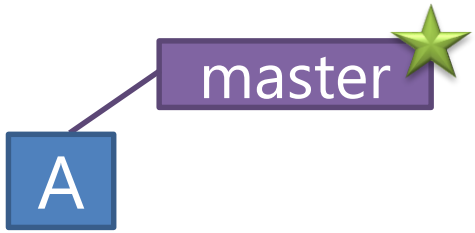




Initialization

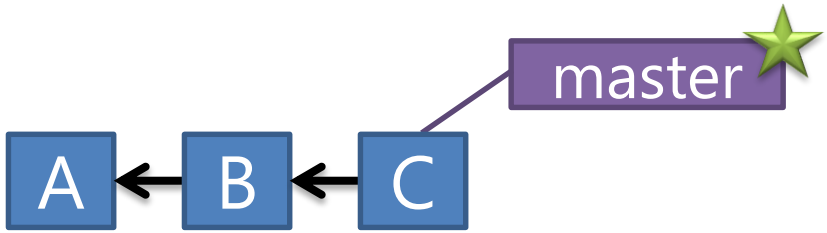
```
C:\> mkdir CoolProject
C:\> cd CoolProject
C:\CoolProject > git init
Initialized empty Git repository in
C:/CoolProject/.git
C:\CoolProject > notepad README.txt
C:\CoolProject > git add .
C:\CoolProject > git commit -m 'my first
commit'
[master (root-commit) 7106a52] my first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.txt
```

Branches Illustrated



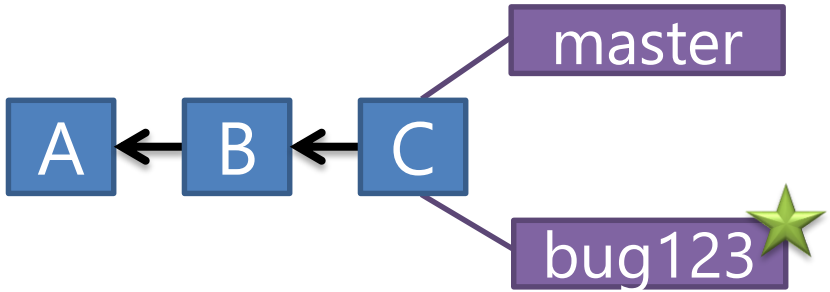

```
> git commit –m ‘my first commit’
```

Branches Illustrated



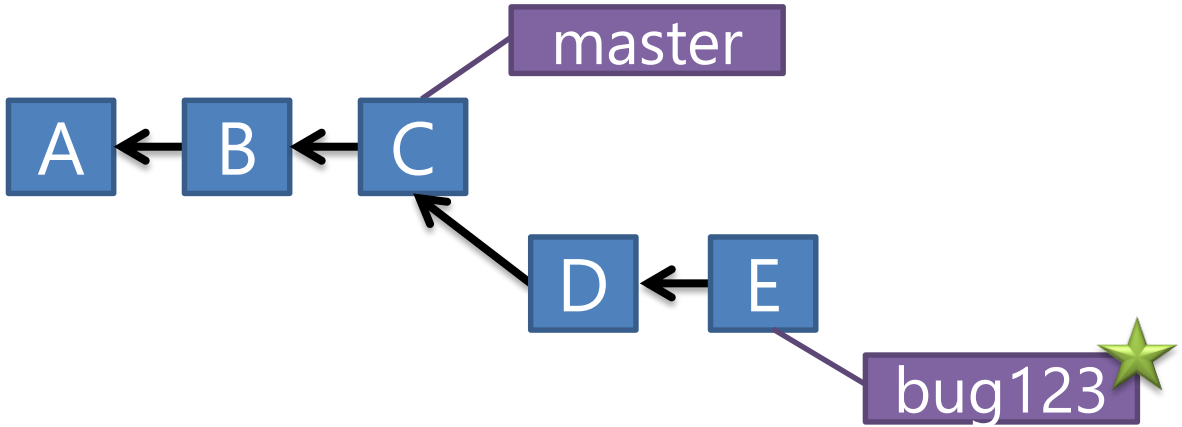

```
> git commit (x2)
```

Branches Illustrated



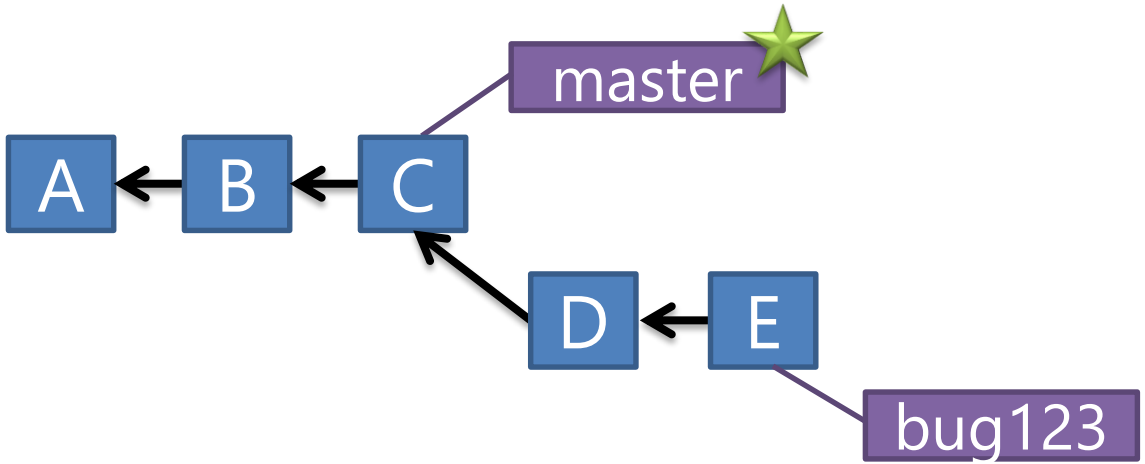

```
> git checkout -b bug123
```

Branches Illustrated



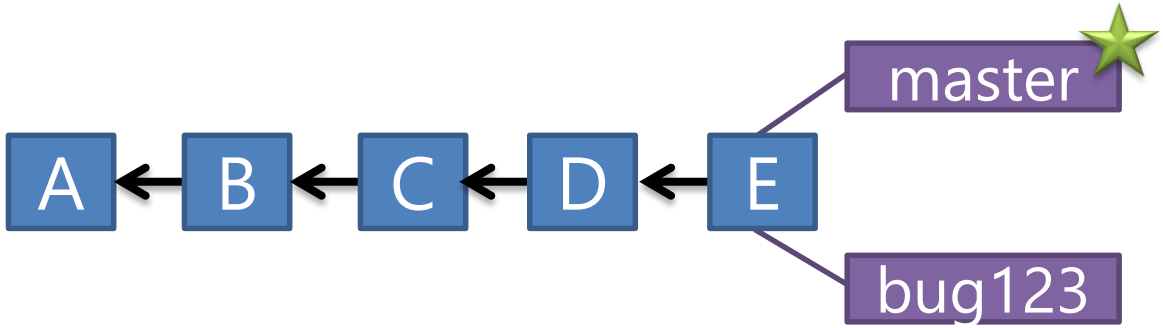

```
> git commit (x2)
```

Branches Illustrated



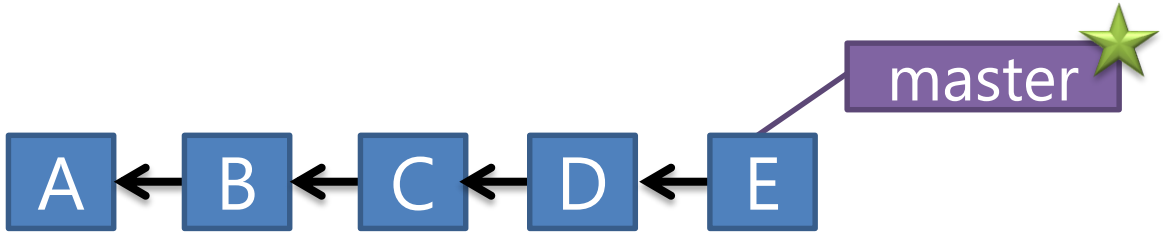

```
> git checkout master
```

Branches Illustrated



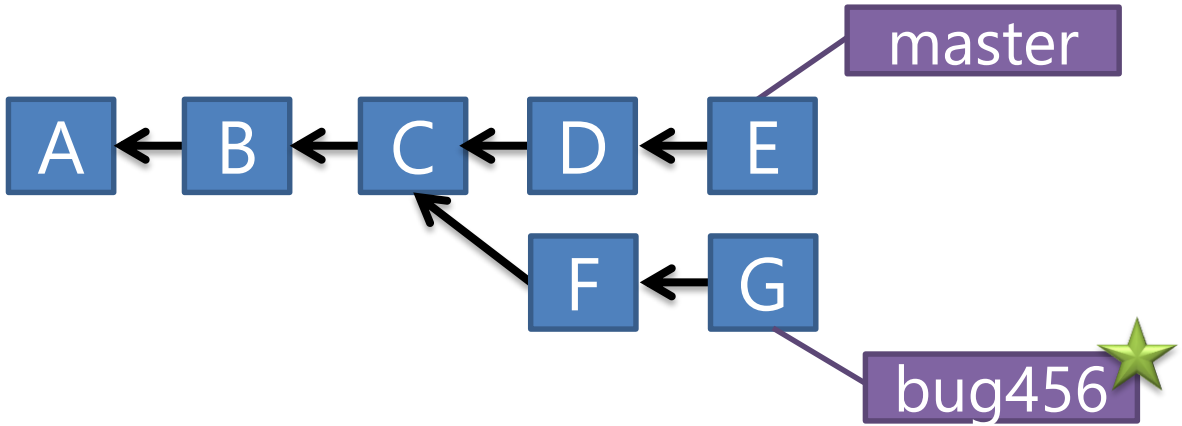

```
> git merge bug123
```

Branches Illustrated

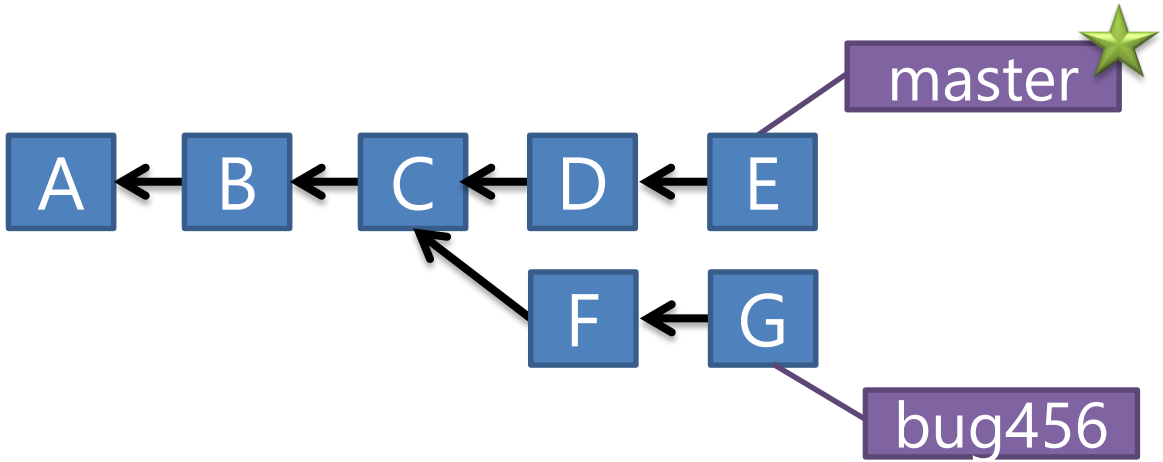



```
> git branch -d bug123
```

Branches Illustrated

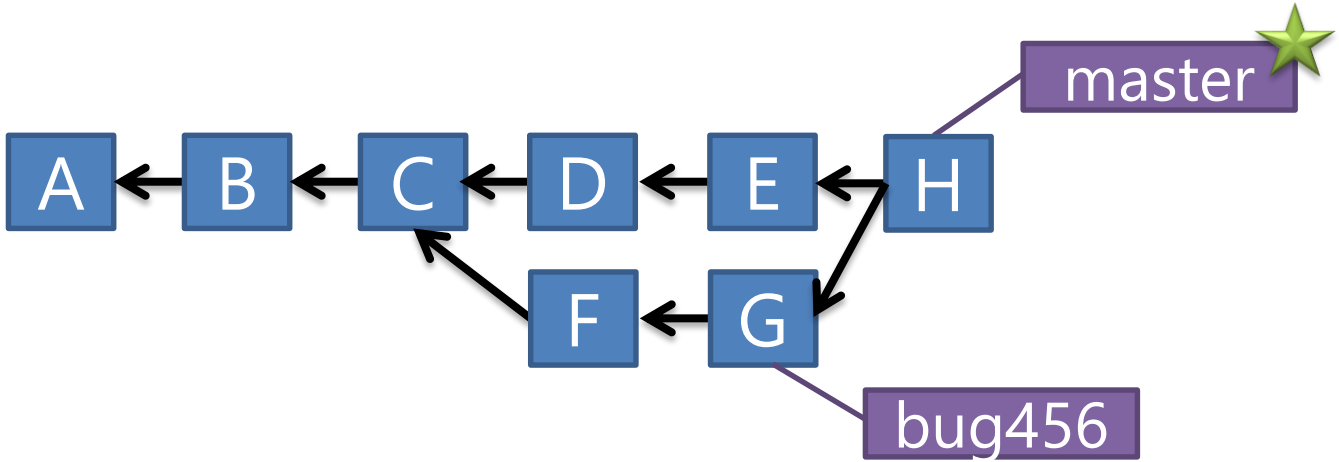


Branches Illustrated



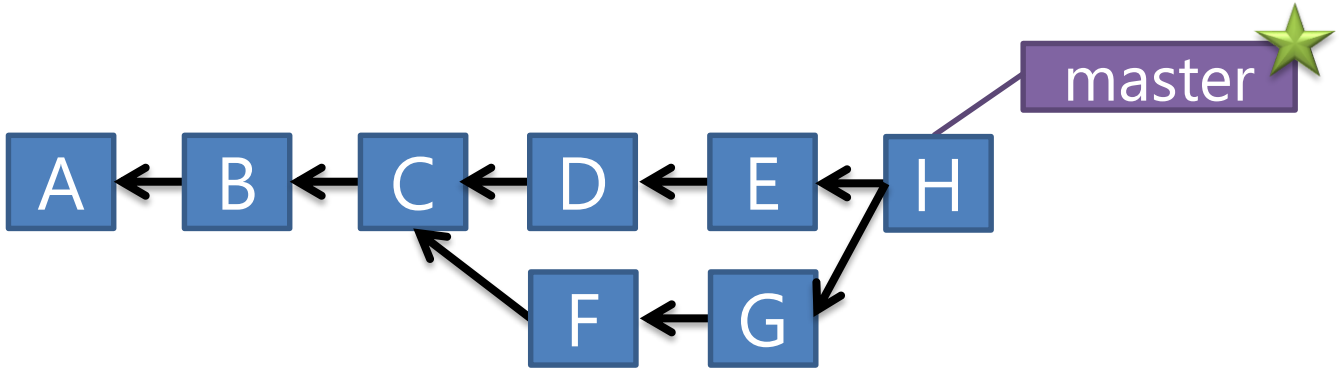

```
> git checkout master
```

Branches Illustrated



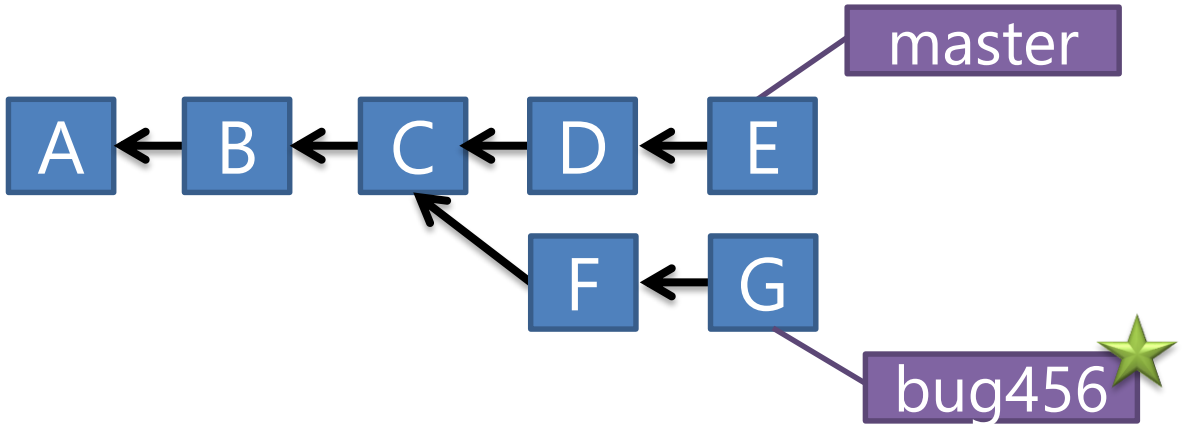

```
> git merge bug456
```

Branches Illustrated

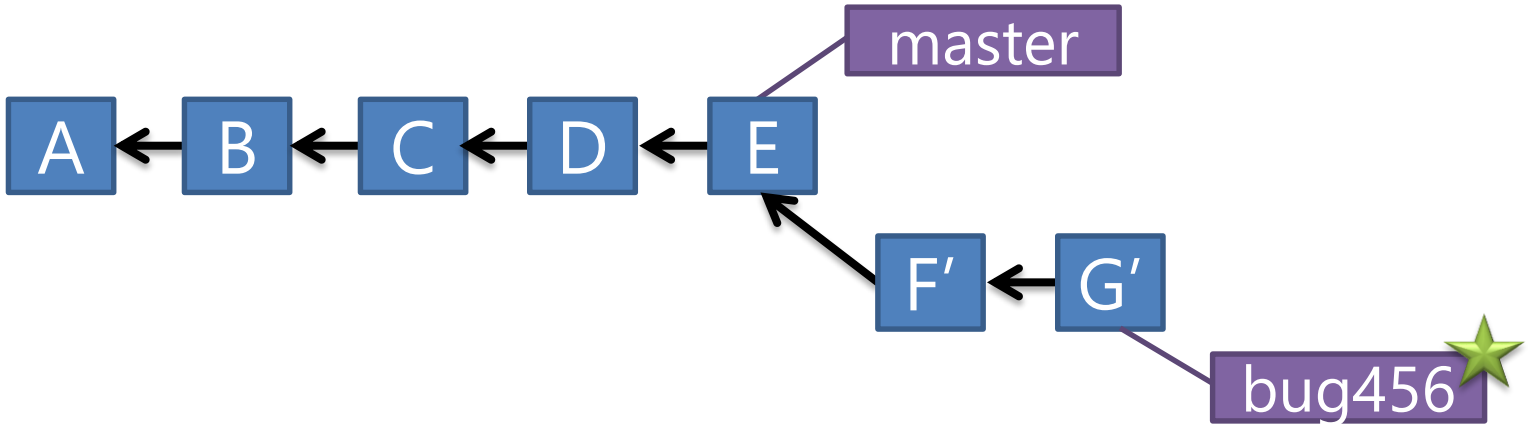



```
> git branch -d bug456
```

Branches Illustrated

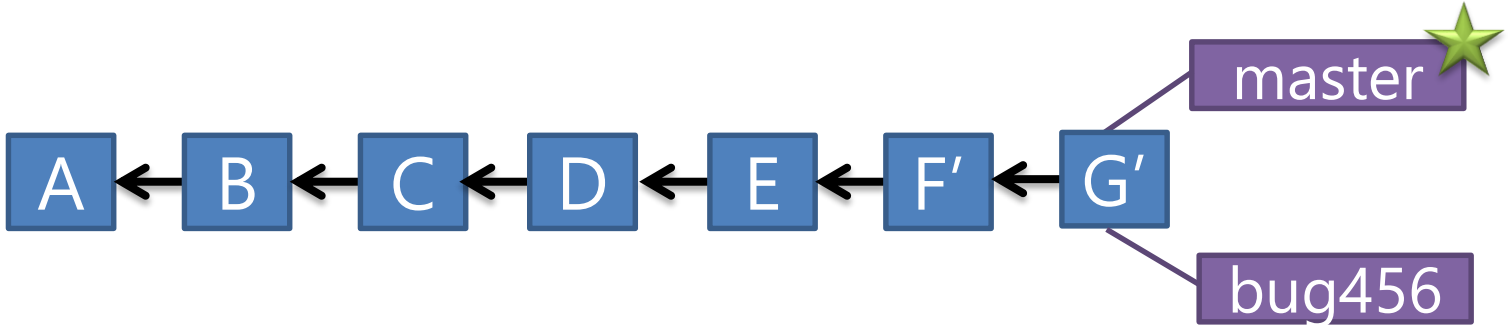


Branches Illustrated




```
> git rebase master
```

Branches Illustrated




```
> git checkout master
> git merge bug456
```

Branching Review

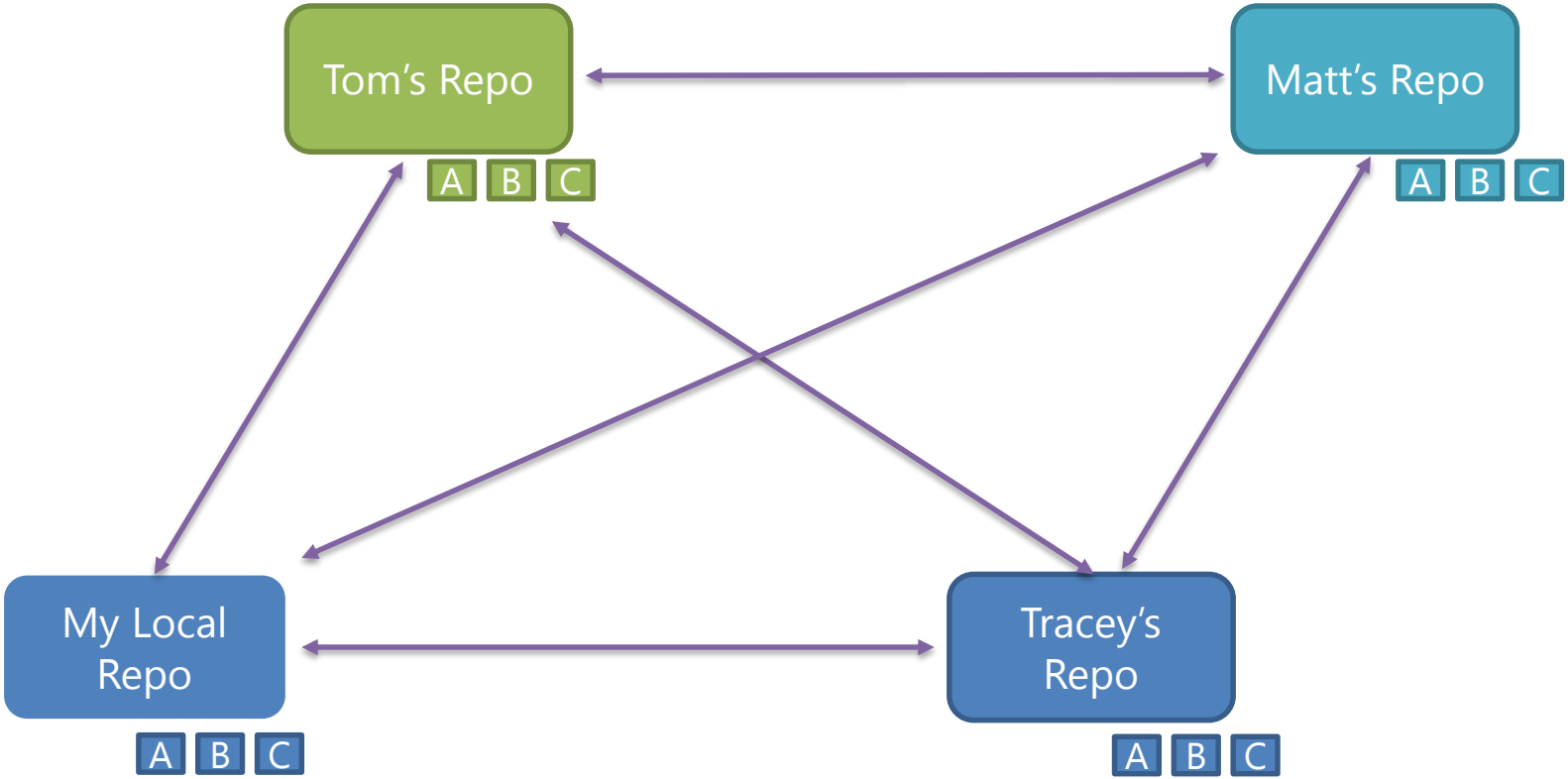
Quick and Easy to create ‘Feature’ Branches

Local branches are very powerful

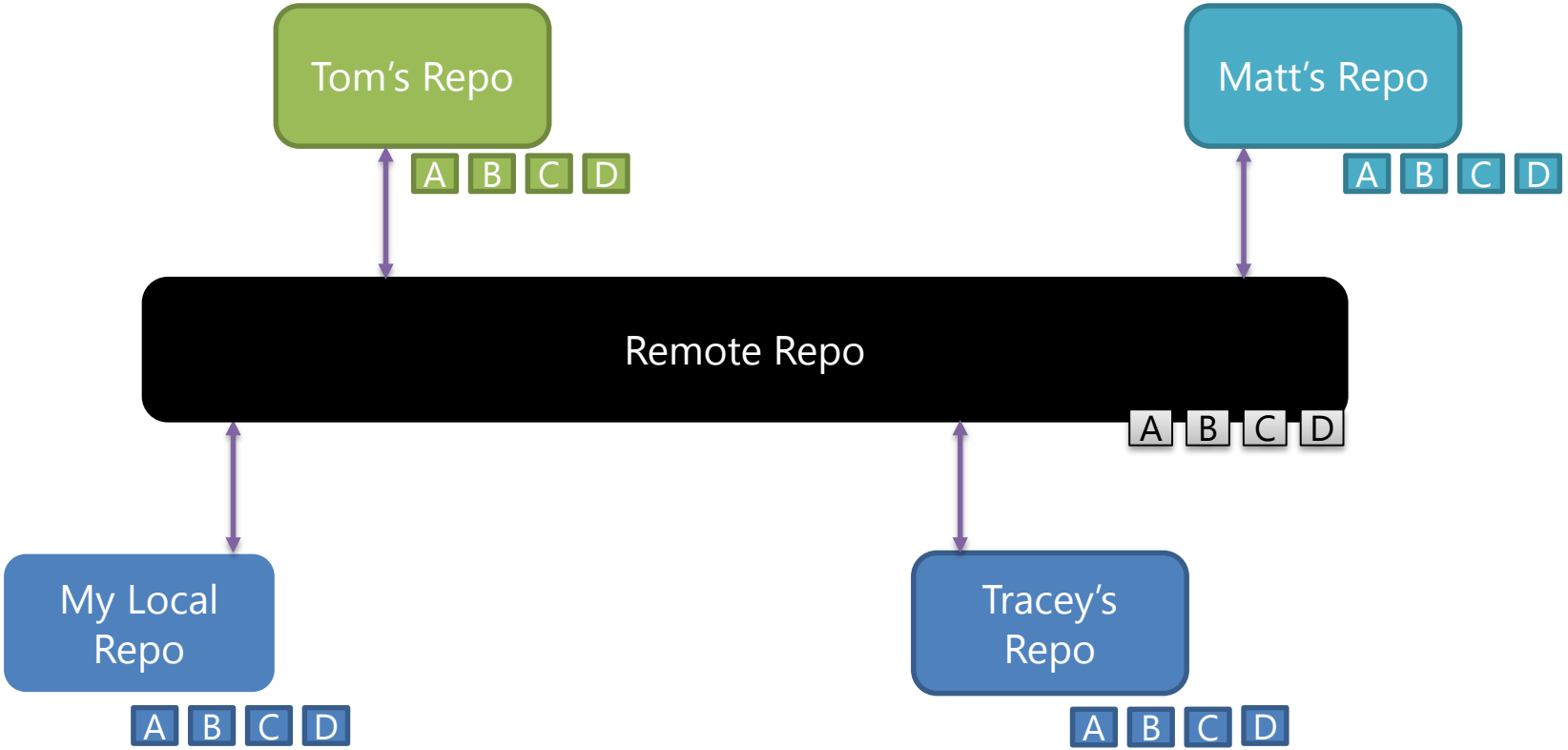
Rebase is not scary

Software is a Team Sport

Sharing commits



Sharing commits



Setting up a Remote

Adding a remote to an existing local repo

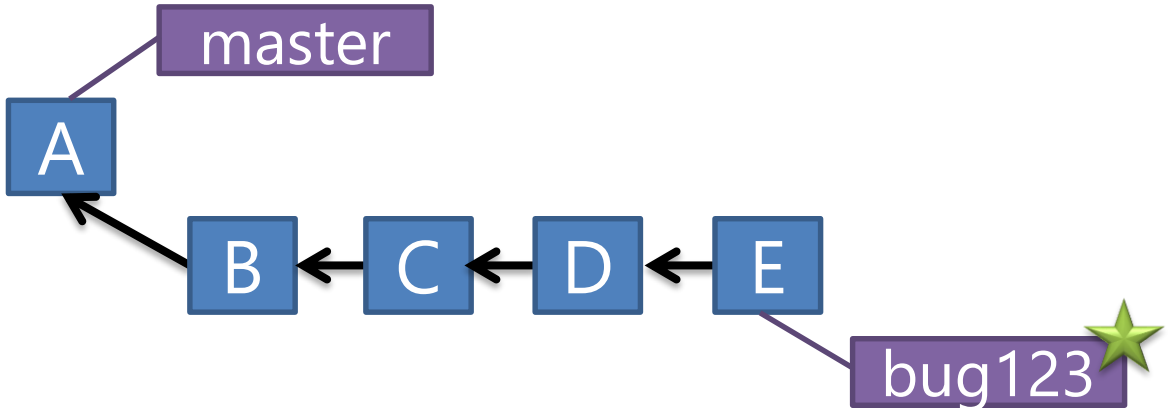
```
C:\CoolProject > git remote add origin https://git01.codeplex.com/coolproject
C:\CoolProject > git remote -v
origin  https://git01.codeplex.com/coolproject (fetch)
origin  https://git01.codeplex.com/coolproject (push)
```

Setting up a Remote

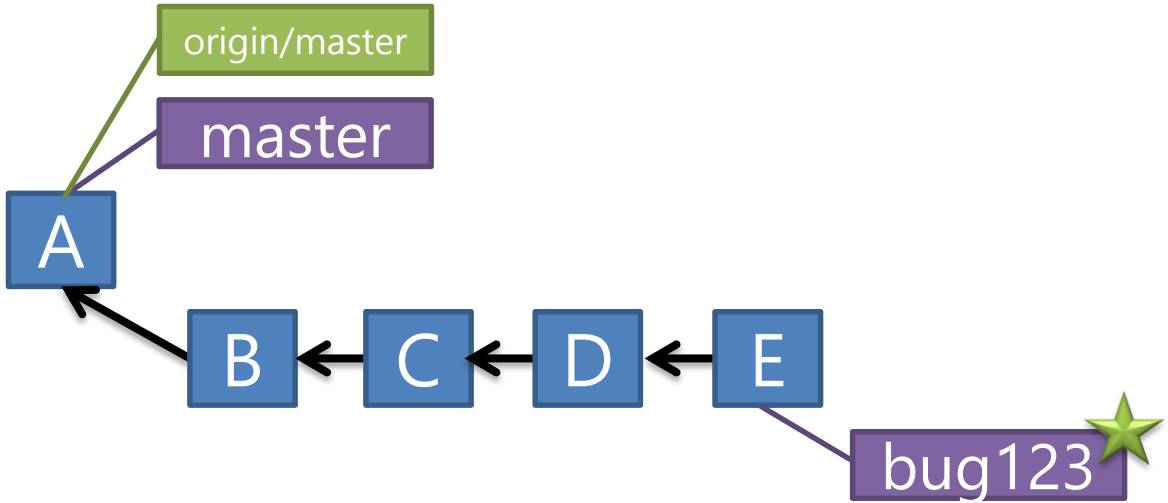
Clone will auto setup the remote

```
C:\> git clone https://git01.codeplex.com/coolproject
Cloning into 'coolproject'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
C:\> cd .\coolproject
C:\CoolProject> git remote -v
origin  https://git01.codeplex.com/coolproject (fetch)
origin  https://git01.codeplex.com/coolproject (push)
```

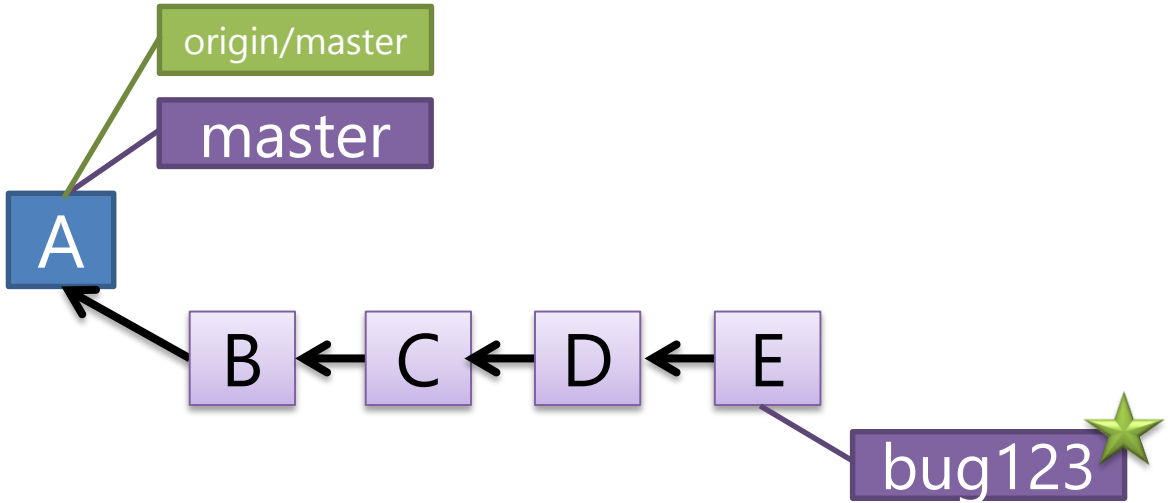
Branches Illustrated



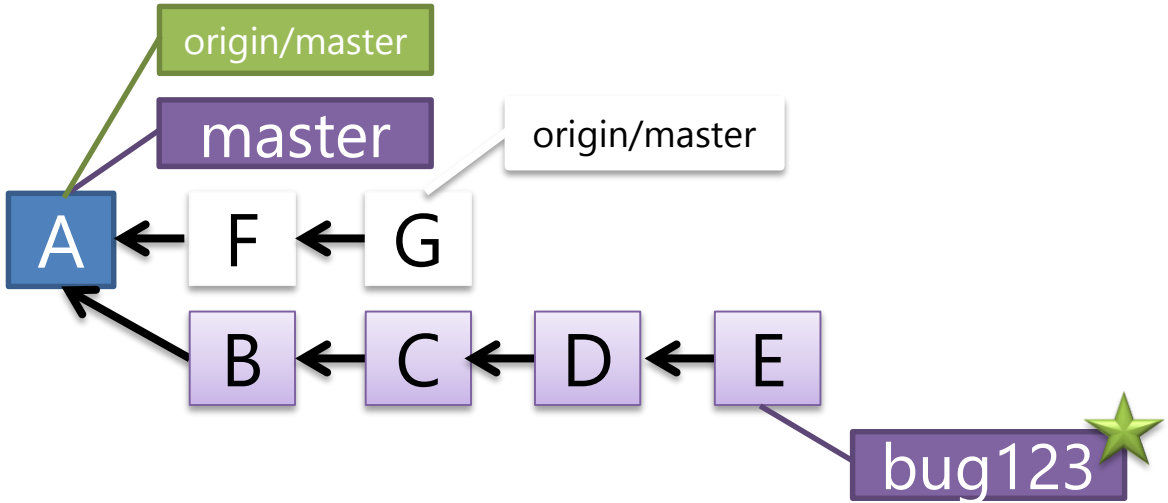
Branches Illustrated



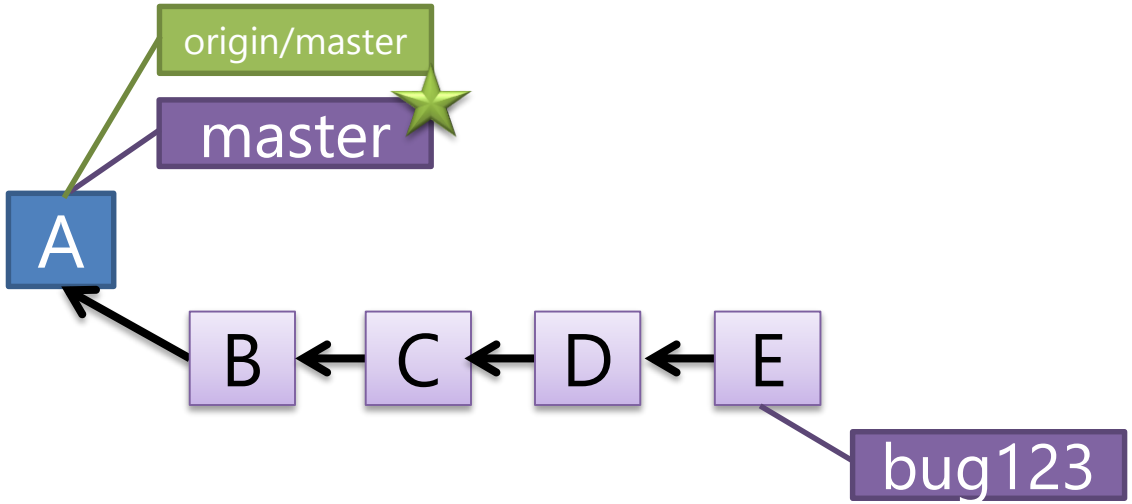
Branches Illustrated



Branches Illustrated

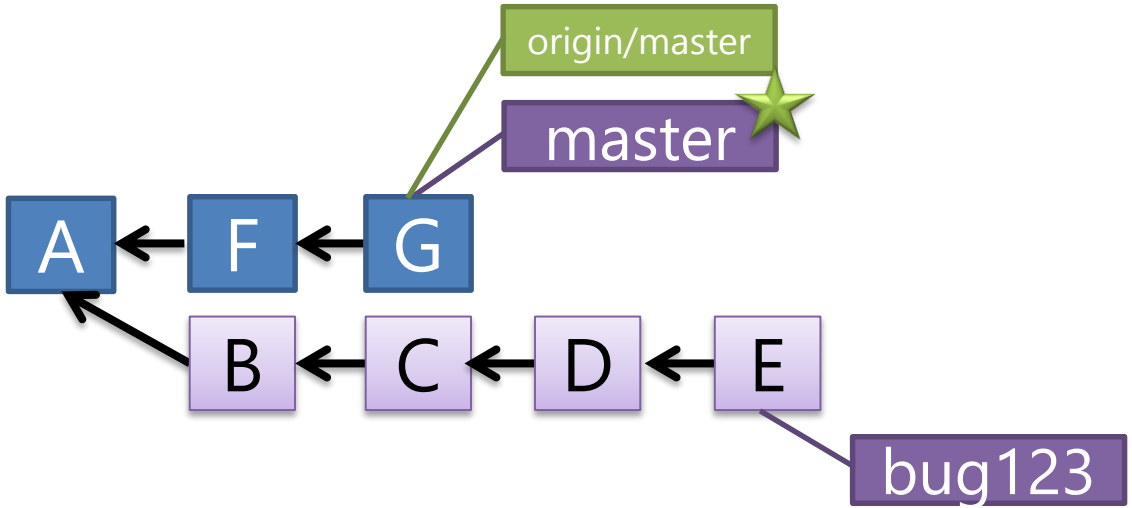


Branches Illustrated




```
> git checkout master
```

Branches Illustrated



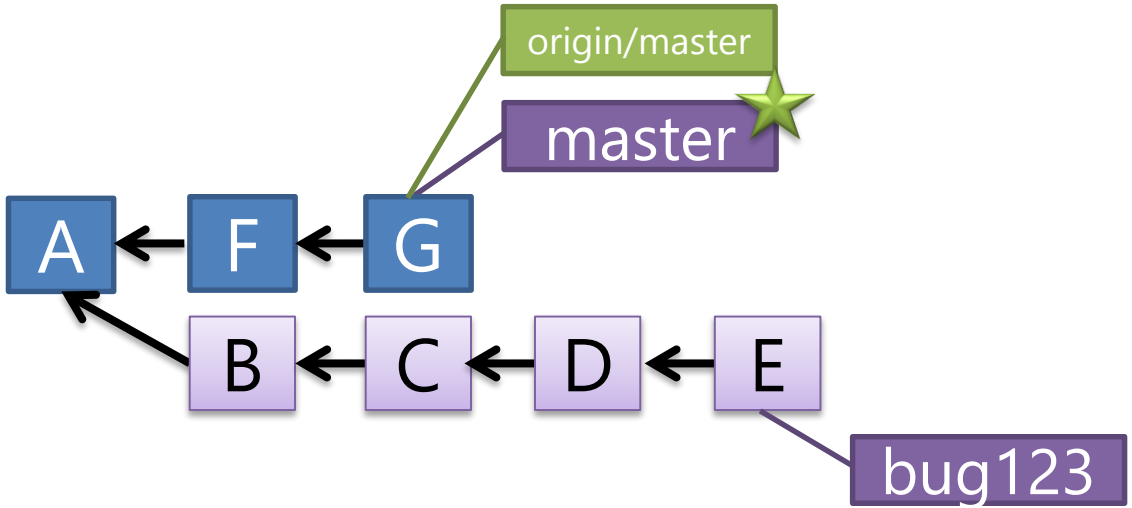

```
> git pull origin
```

Pull = Fetch + Merge

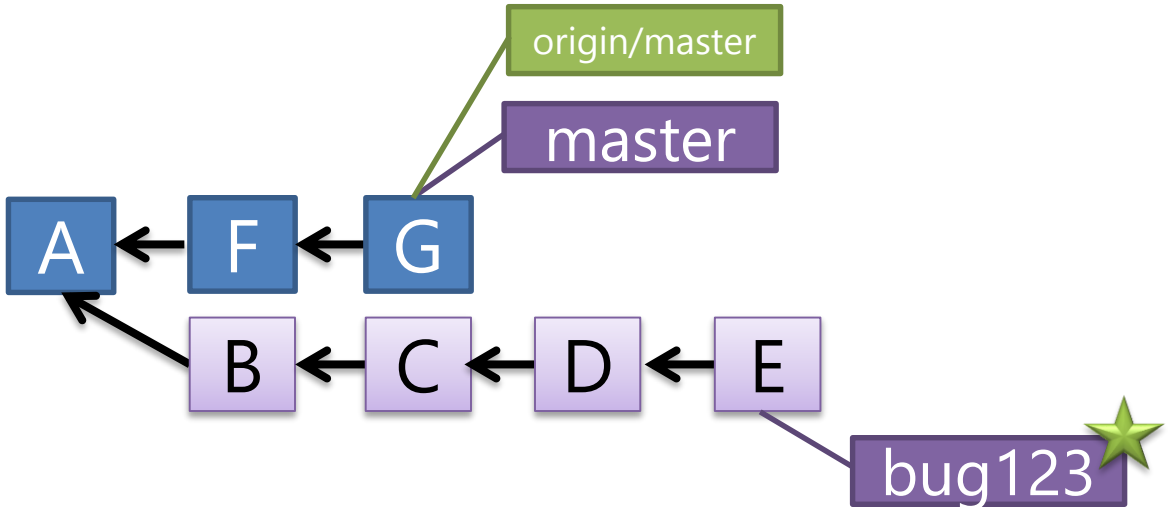
Fetch - updates your local copy of the remote branch

Pull essentially does a fetch and then runs the merge in one step.

Branches Illustrated

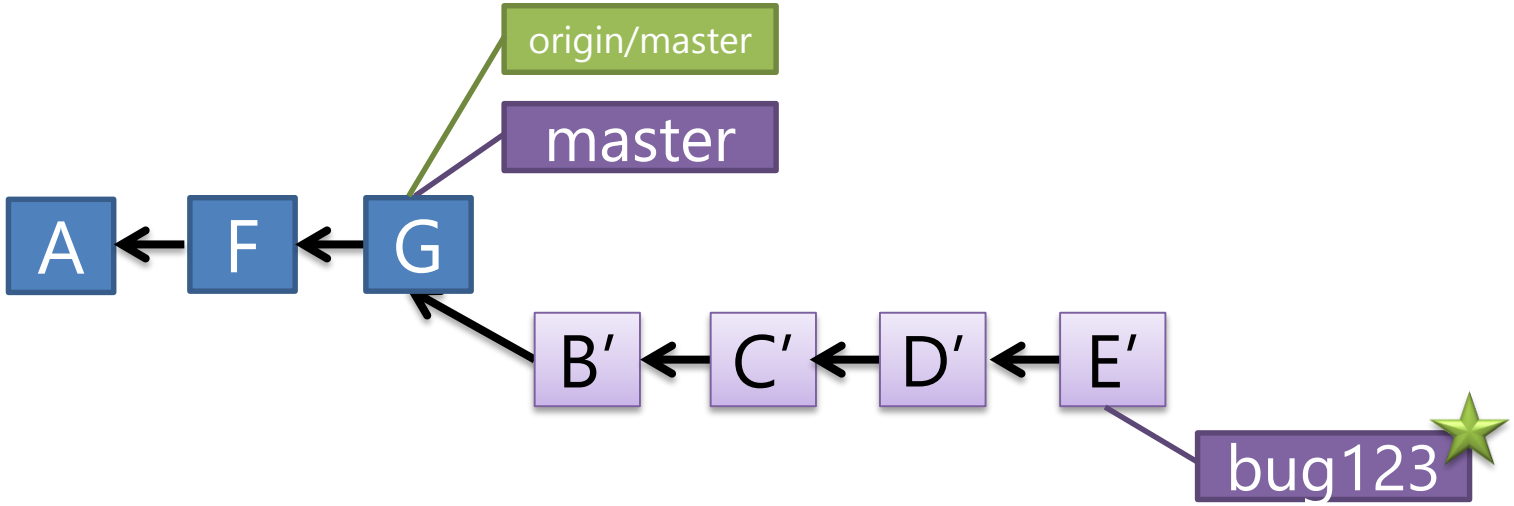


Branches Illustrated



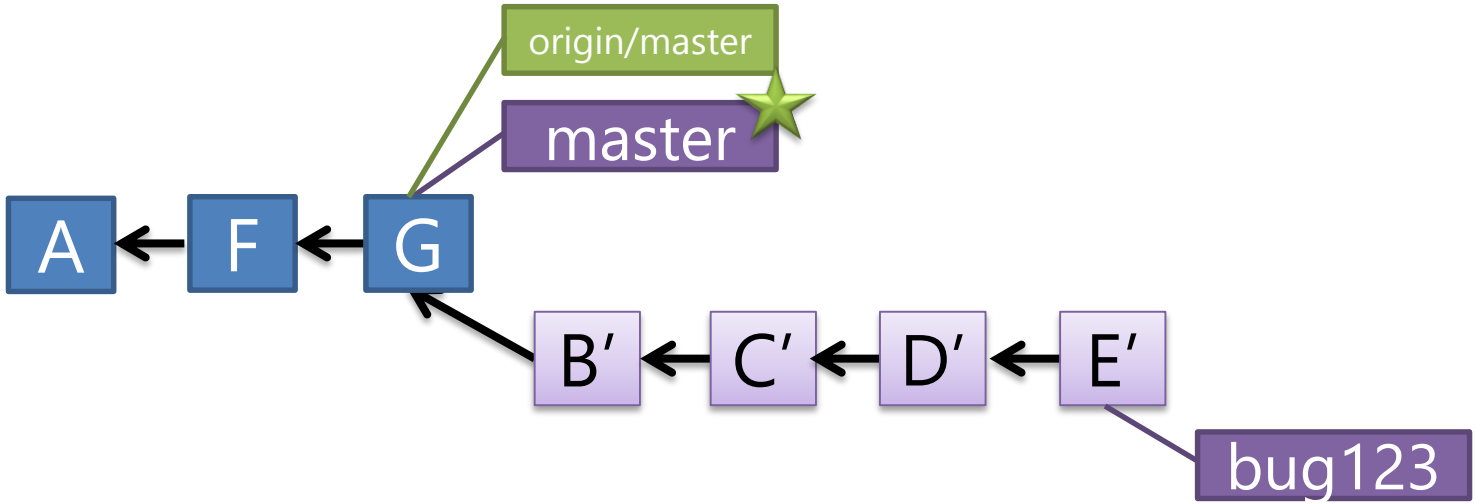

```
> git checkout bug123
```

Branches Illustrated



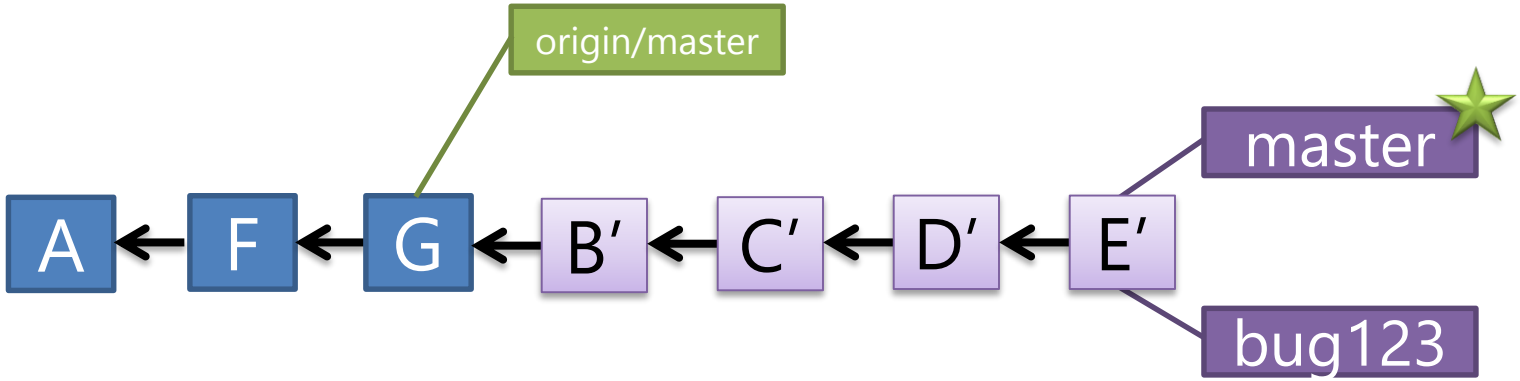

```
> git rebase master
```

Branches Illustrated



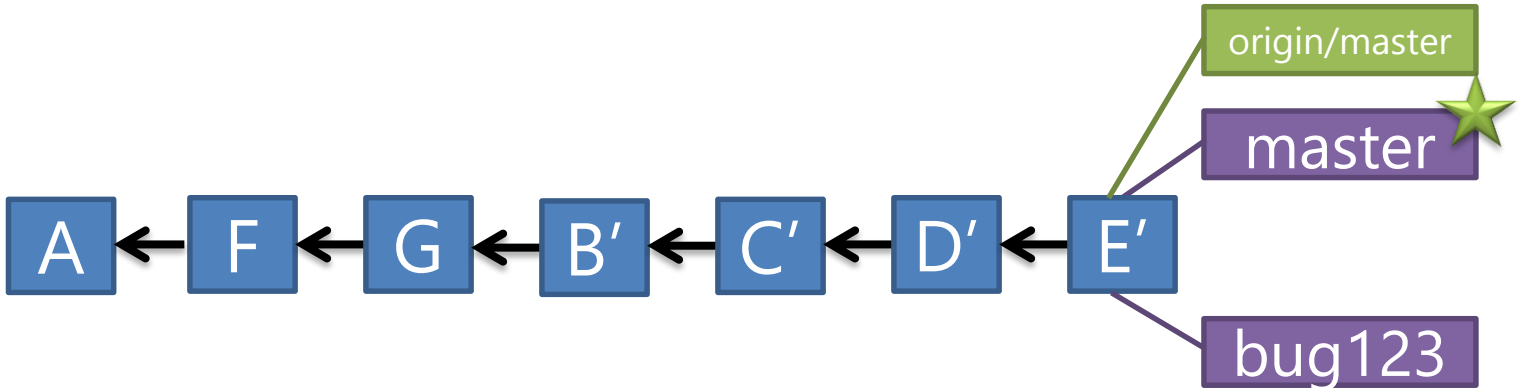

```
> git checkout master
```

Branches Illustrated




```
> git merge bug123
```

Branches Illustrated




```
> git push origin
```

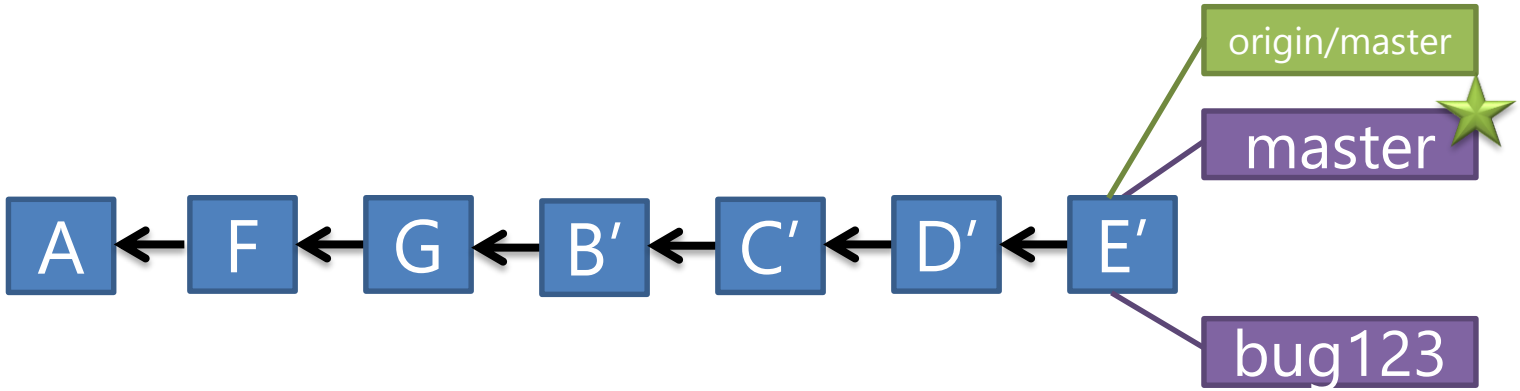
Push

Pushes your changes upstream

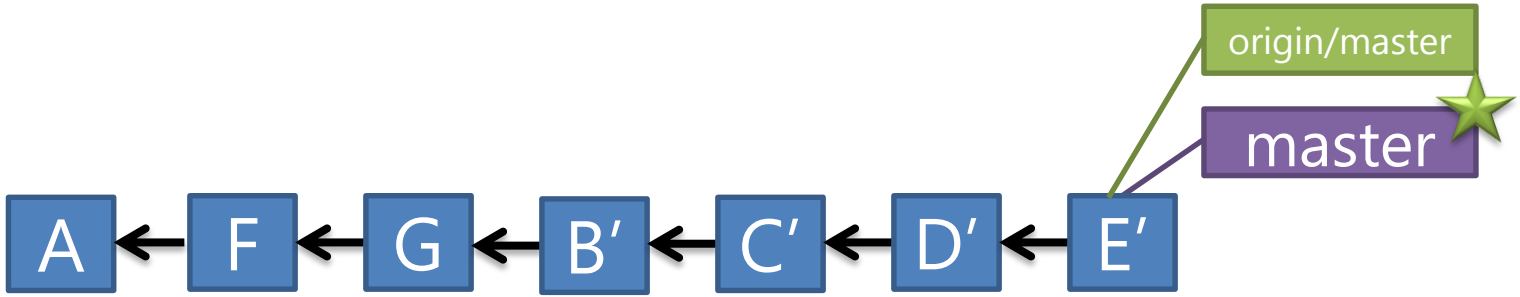
Git will reject pushes if newer changes exist on remote.

Good practice: Pull then Push

Branches Illustrated



Branches Illustrated

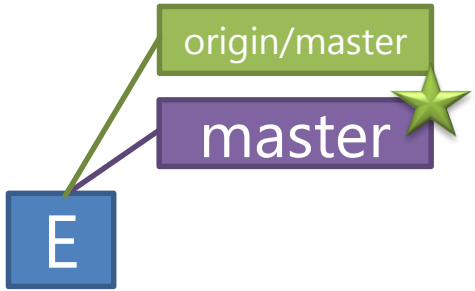



```
> git branch -d bug123
```

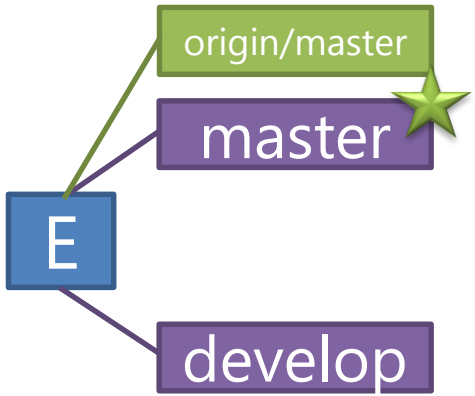
Short vs. Long-Lived Branches

Great for multi-version work
Follow same rules as Master…Story branches
Integrate frequently
Pushed to Remotes

Branches Illustrated

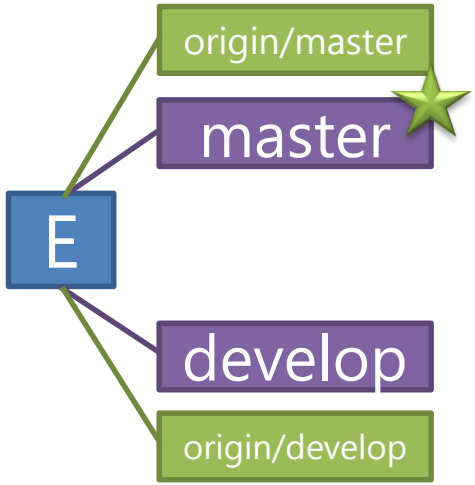


Branches Illustrated



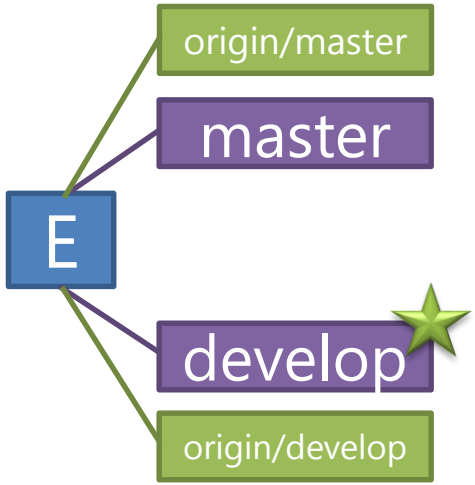

```
> git branch develop
```

Branches Illustrated



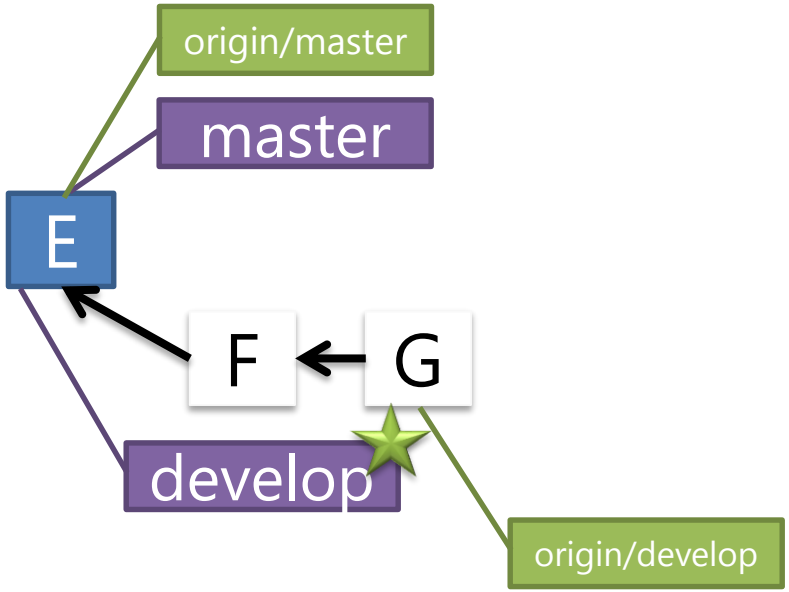

```
> git push origin develop
```

Branches Illustrated

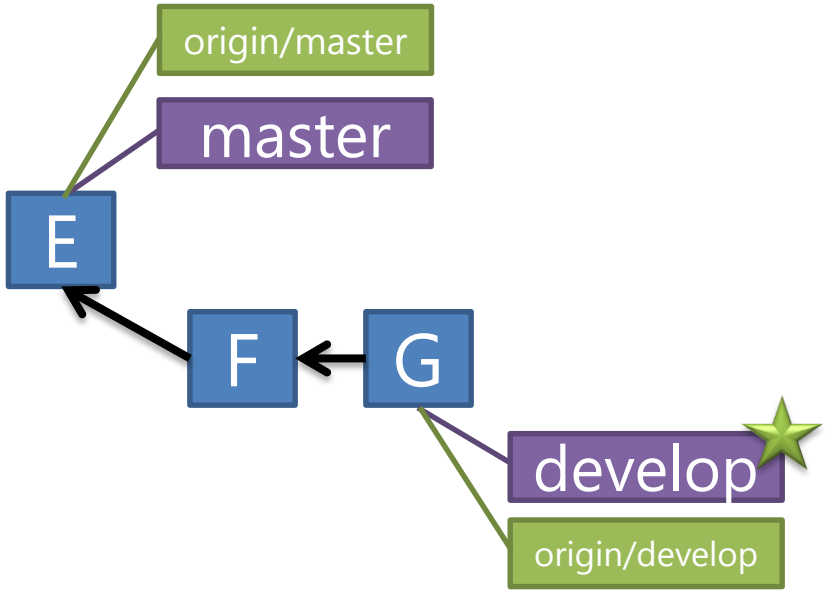



```
> git checkout develop
```

Branches Illustrated

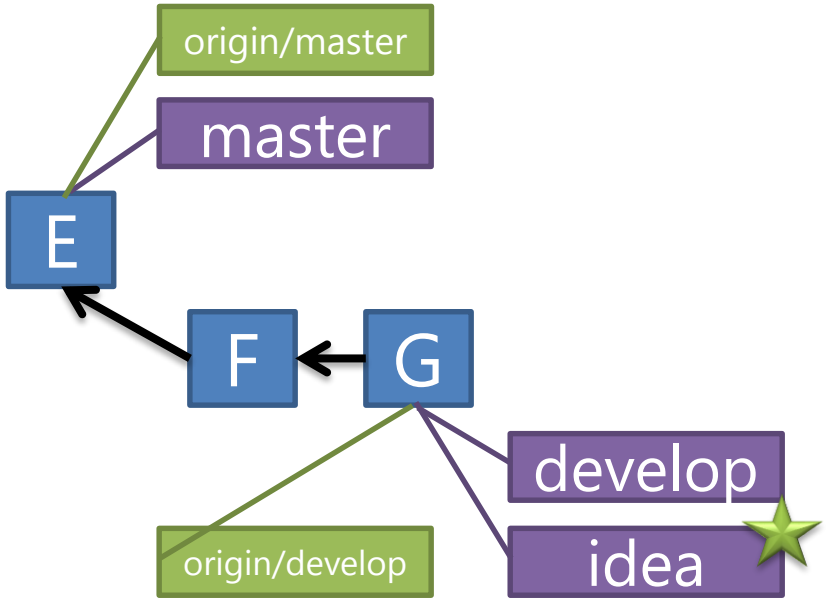


Branches Illustrated



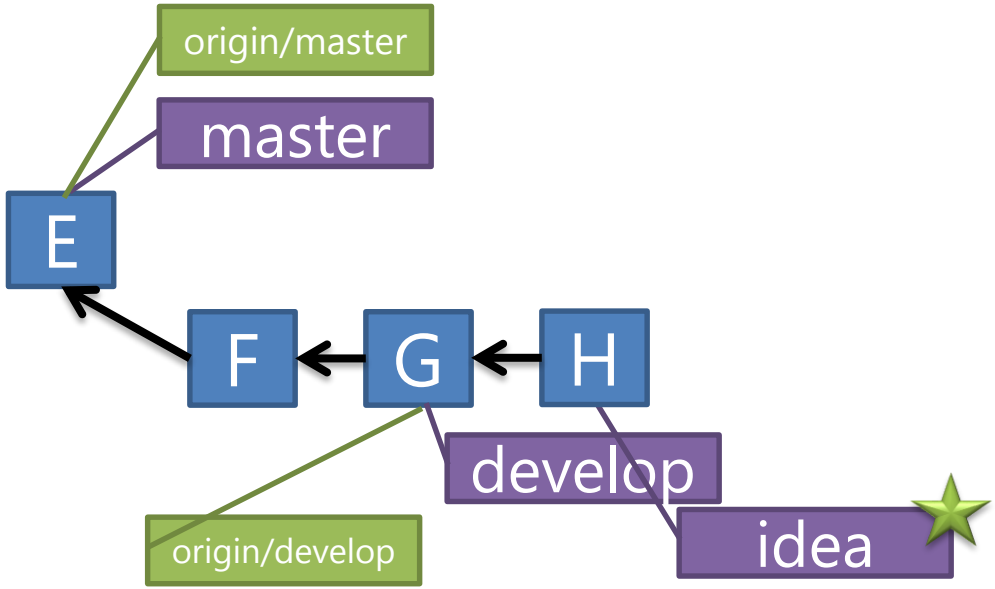

```
> git pull origin develop
```

Branches Illustrated



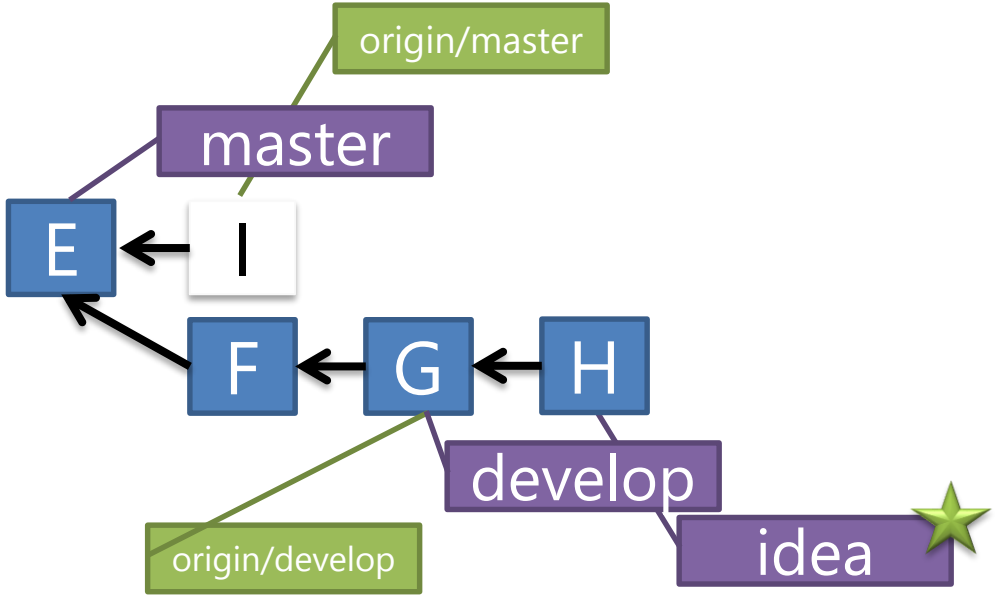

```
> git checkout -b idea
```

Branches Illustrated

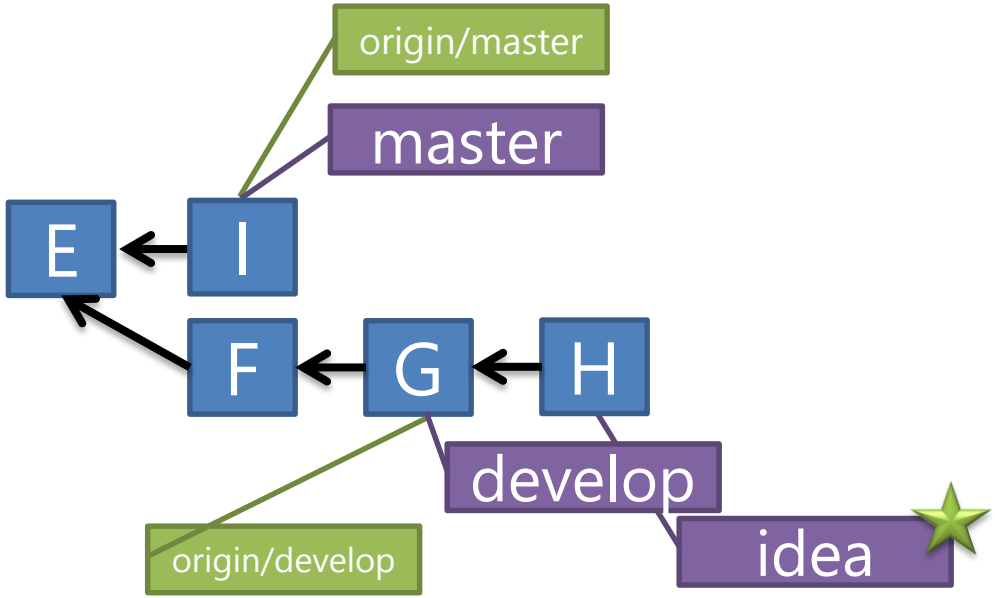



```
> git commit
```

Branches Illustrated

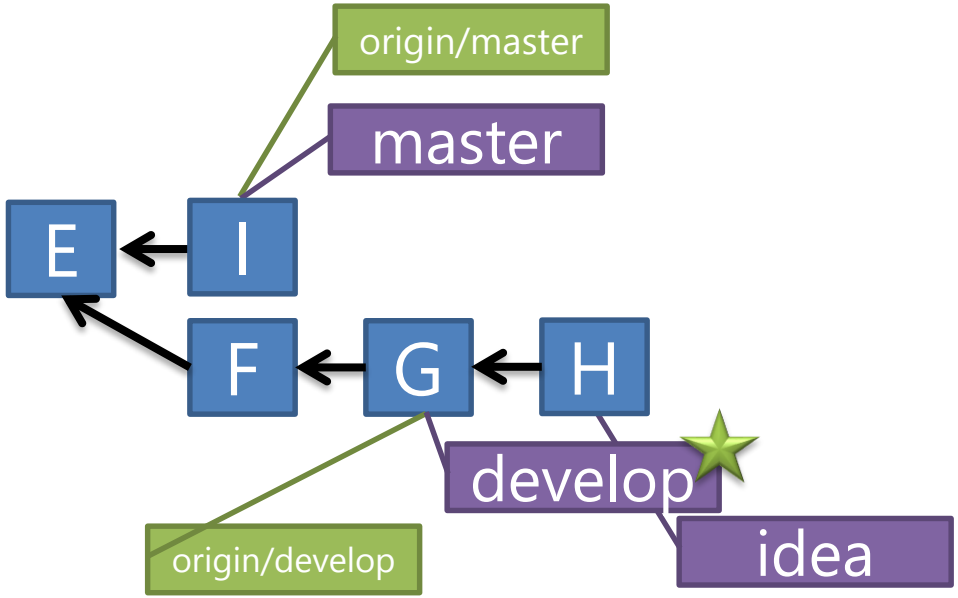


Branches Illustrated



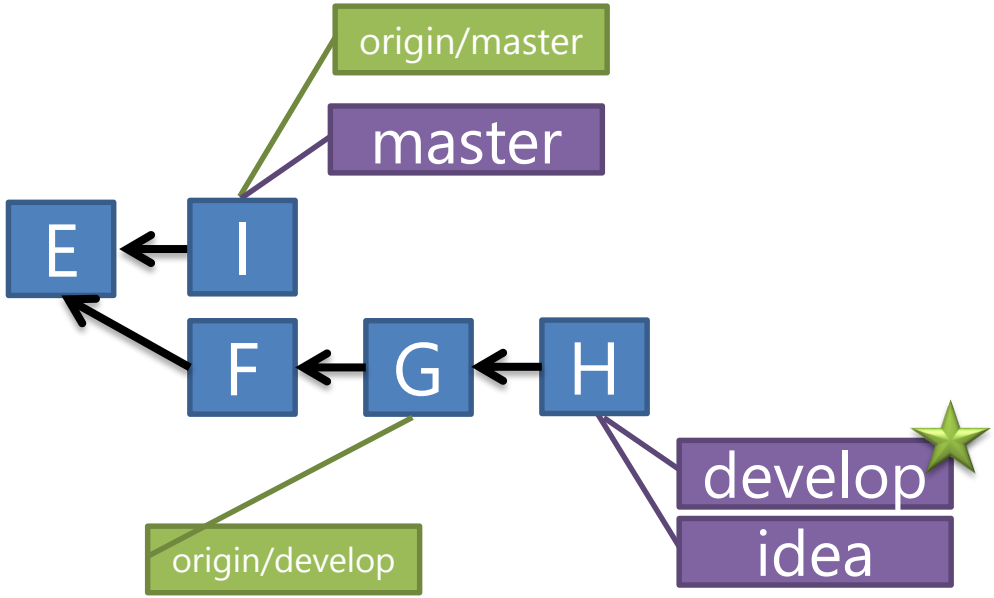

```
> git pull (at least daily)
```

Branches Illustrated



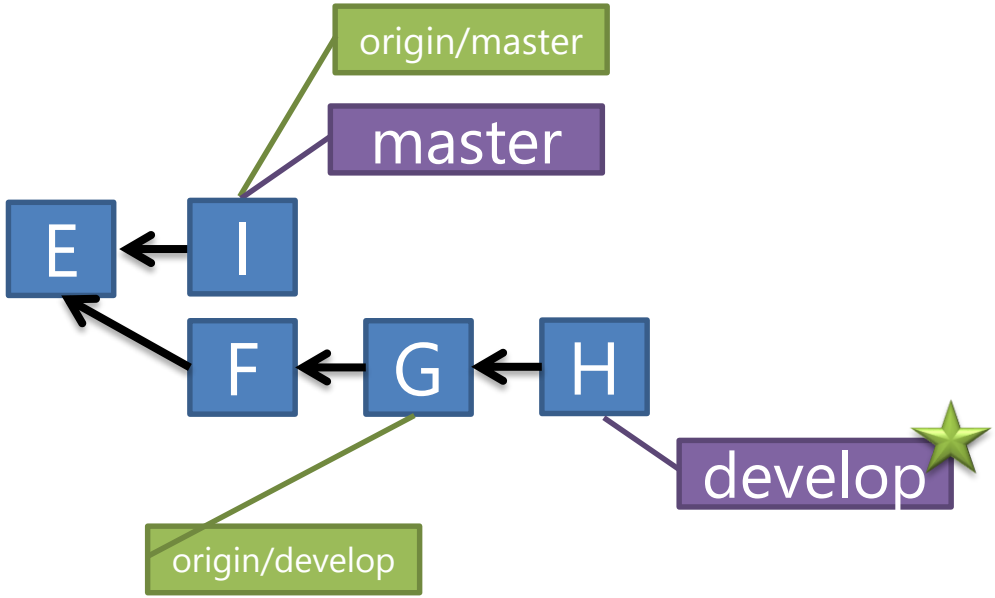

```
> git checkout develop
```

Branches Illustrated



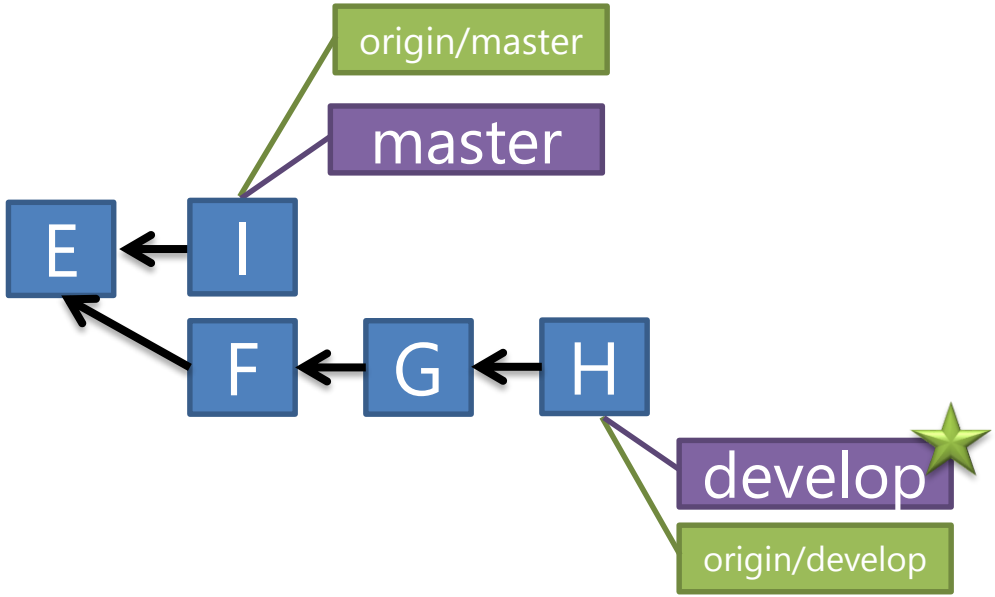

```
> git merge idea (fast forward merge)
```

Branches Illustrated



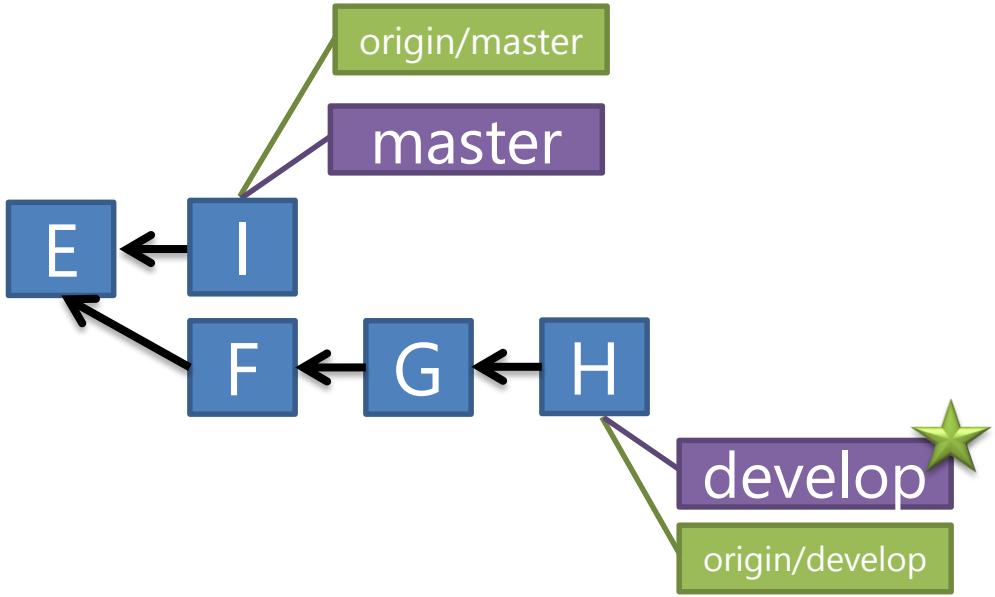

```
> git branch -d idea
```

Branches Illustrated



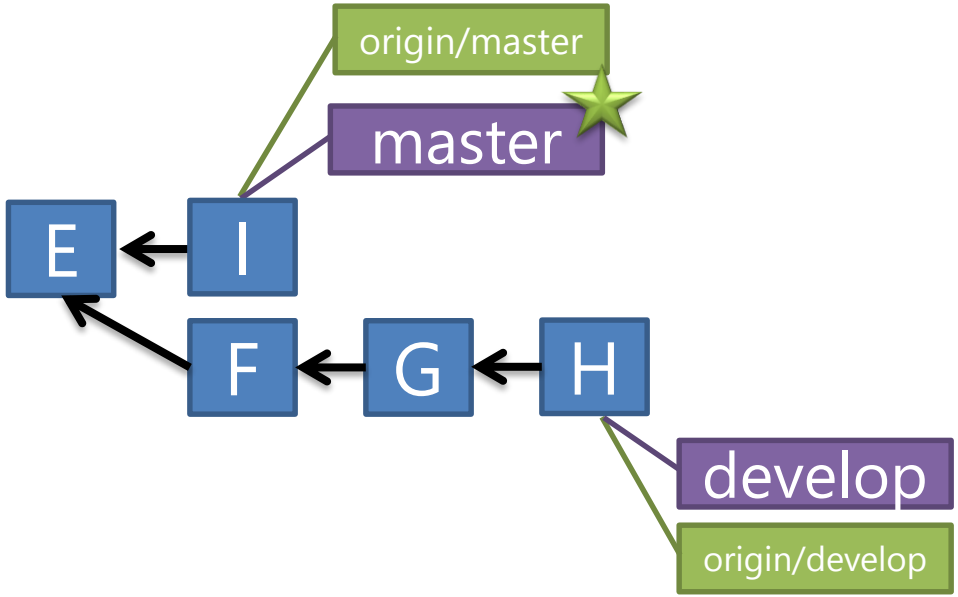

```
> git push origin develop
```

Merge Flow vs. Rebase Flow



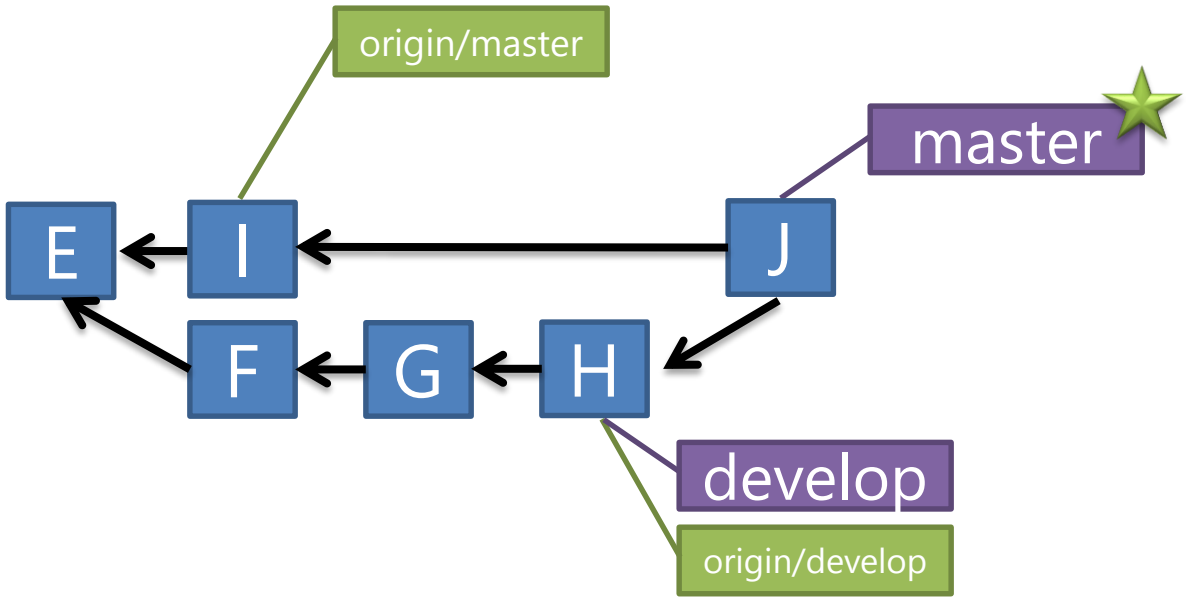

```
> git push origin develop
```

Branches Illustrated – Merge Flow



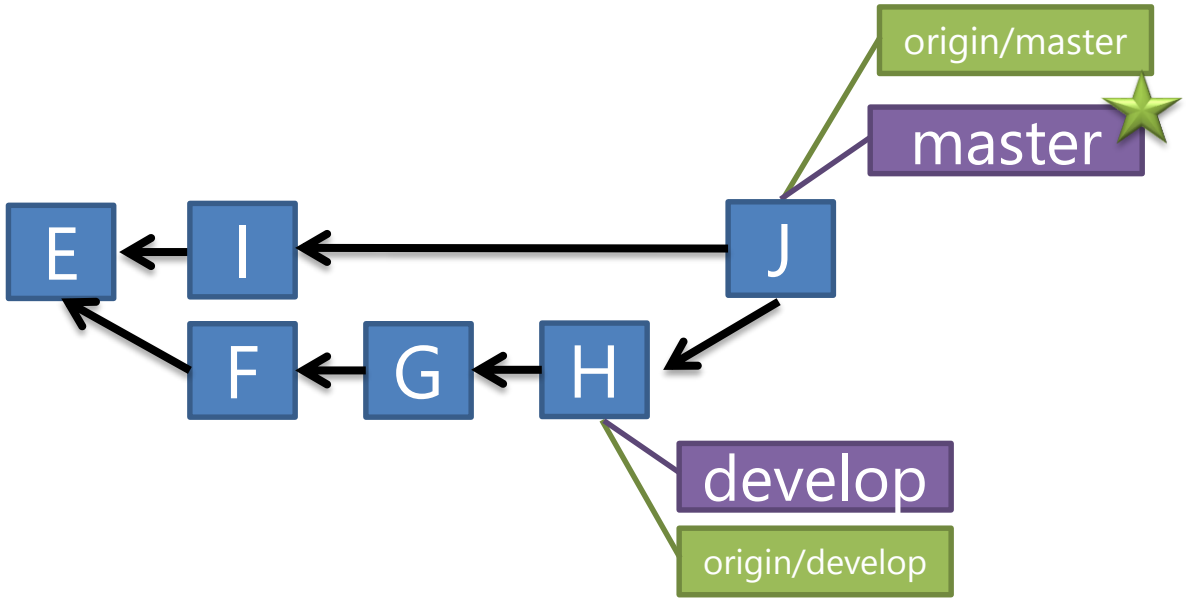

```
> git checkout master
```

Branches Illustrated – Merge Flow



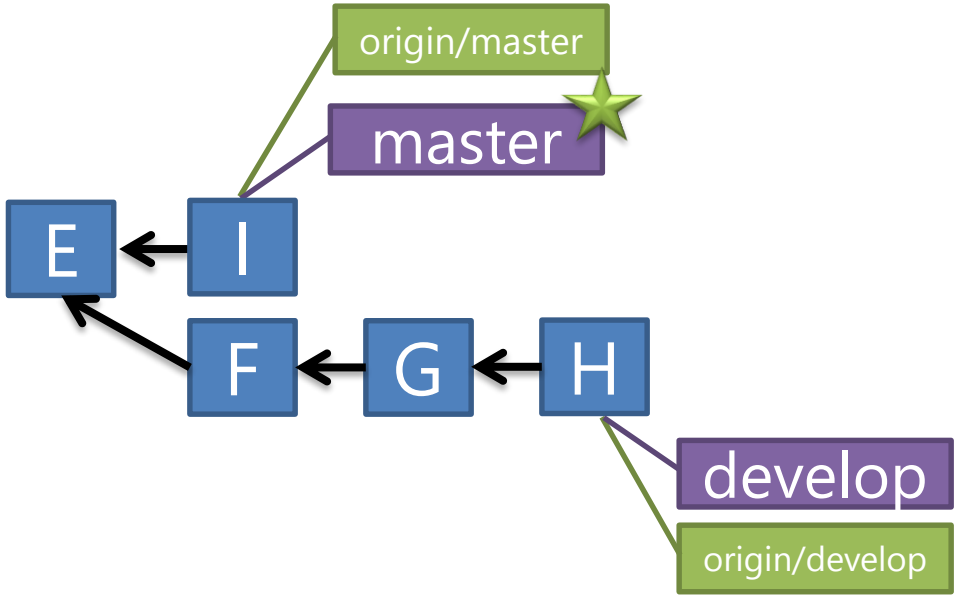

```
> git merge develop
```

Branches Illustrated – Merge Flow



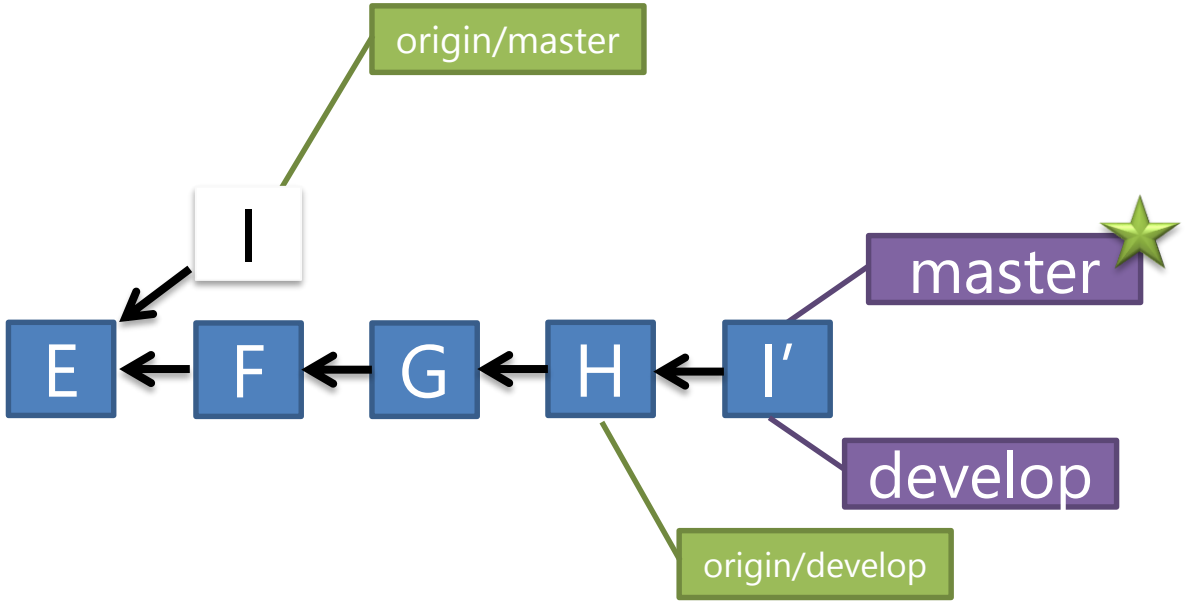

```
> git push origin
```

Branches Illustrated – Rebase Flow



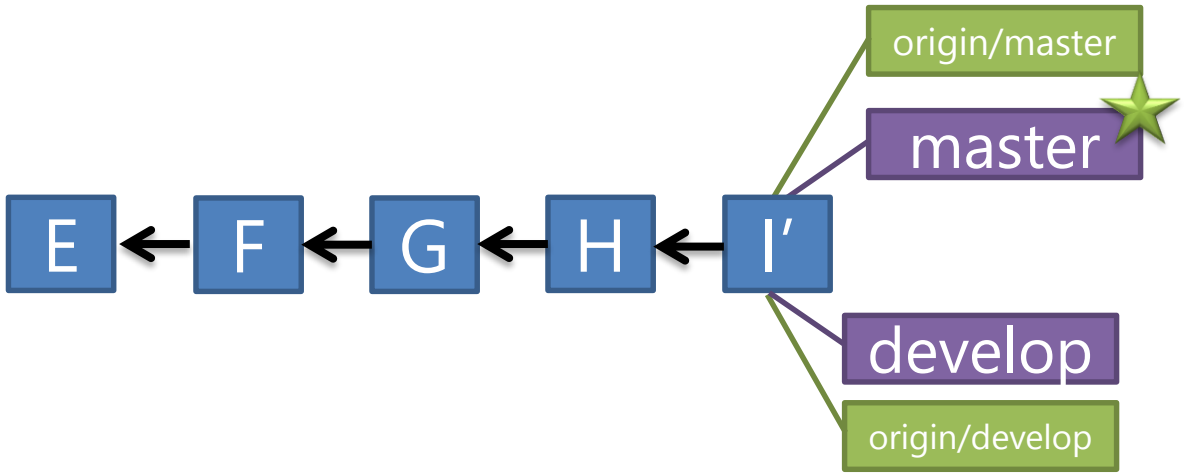

```
> git checkout master
```

Branches Illustrated – Rebase Flow



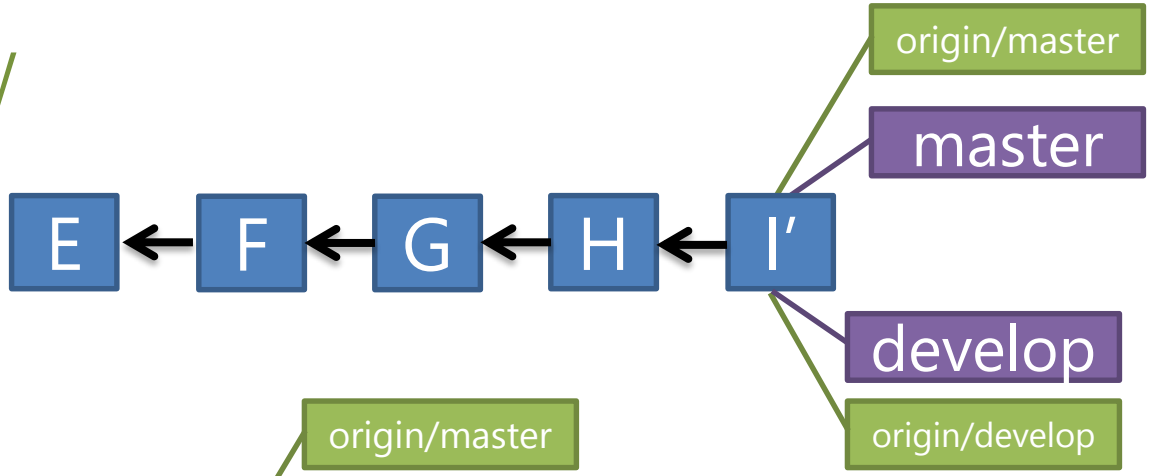

```
> git rebase develop
```

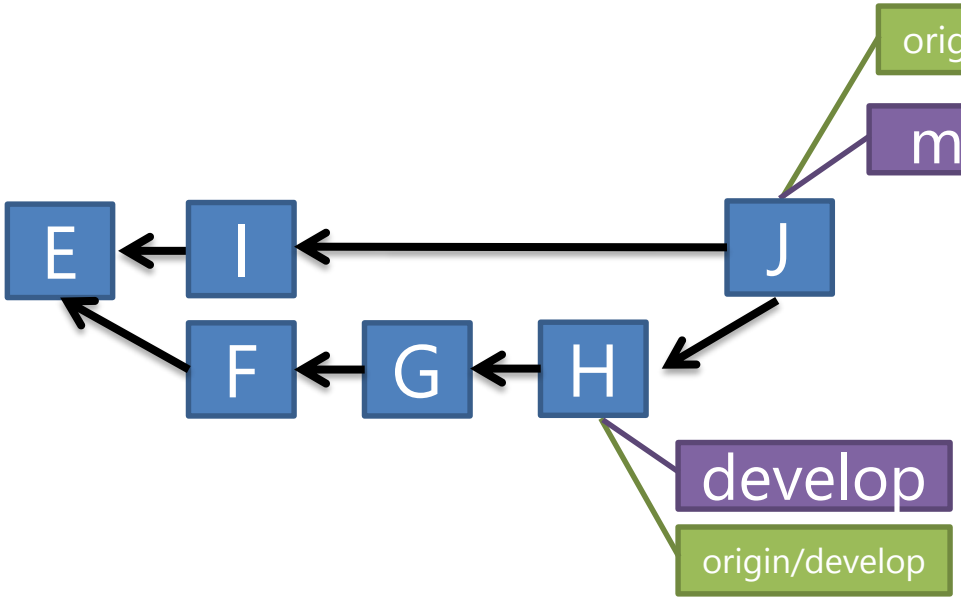
Branches Illustrated – Rebase Flow




```
> git push origin
```

Rebase Flow





Merge Flow

Short vs. Long-Lived Branches

Great for multi-version work

Follow same rules as Master

- ...use Story branches

Define your conventions

- What branches do you want to share?
- Branch per environment?

Other very powerful tutorial

https://www.atlassian.com/git/tutorials/comparing-workflows/centralized-workflow