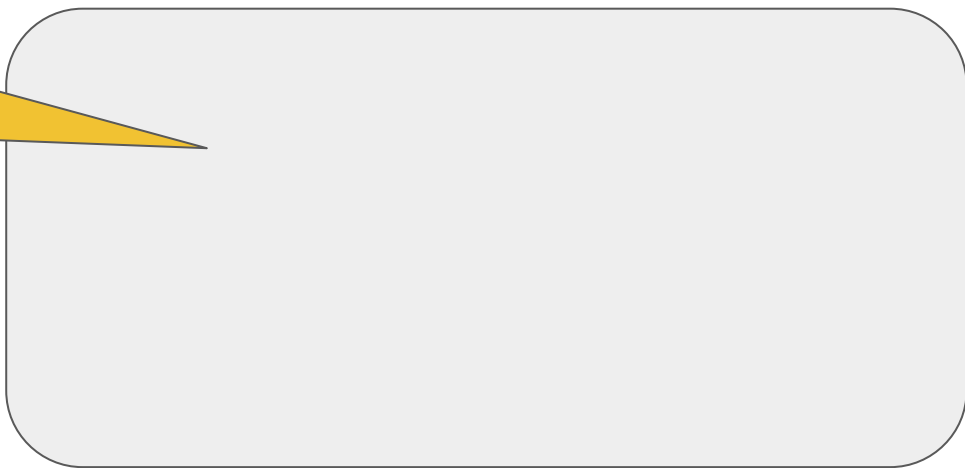
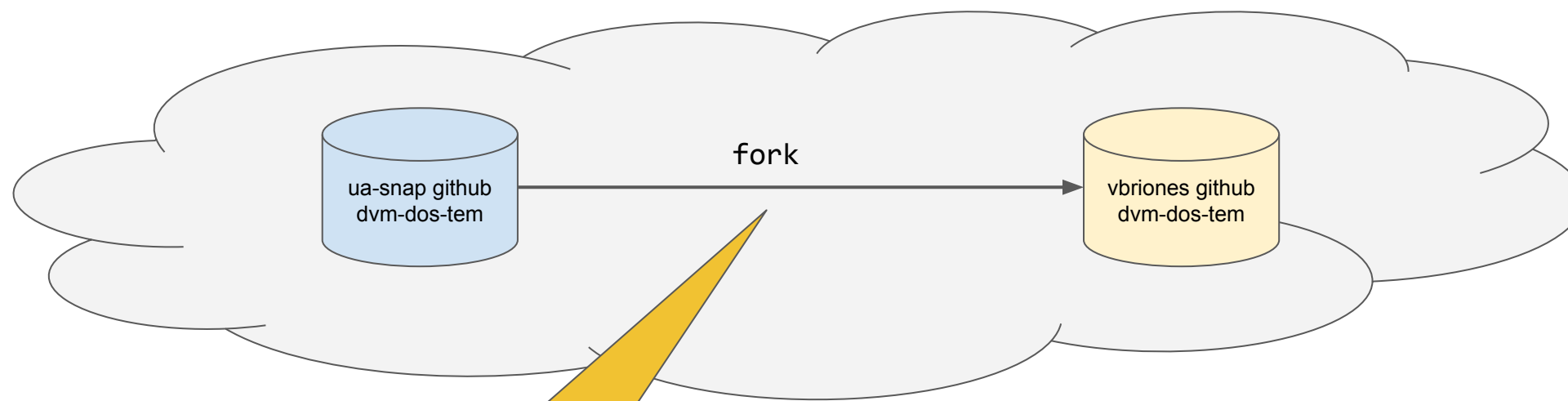


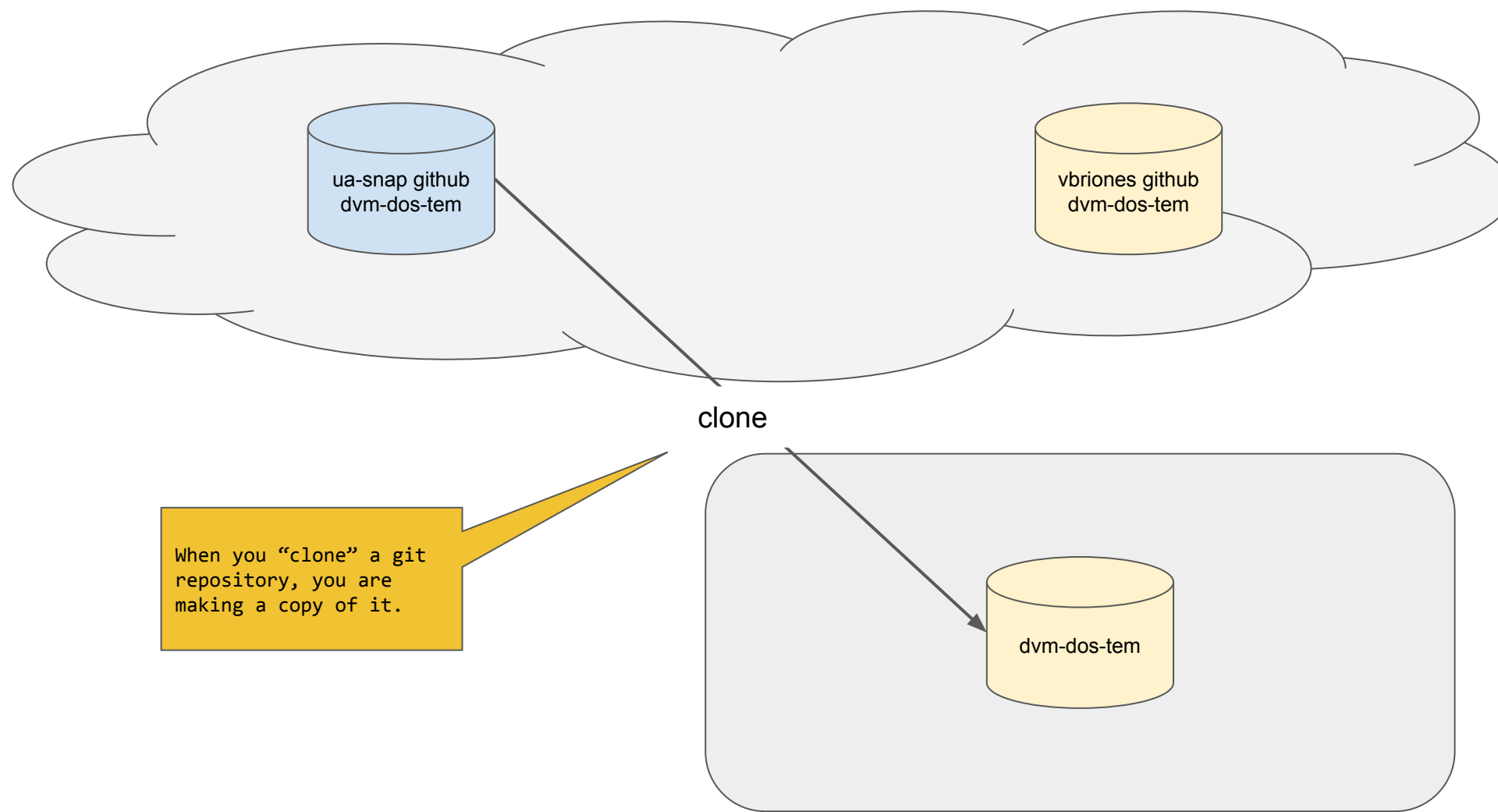
Your computer; could be:

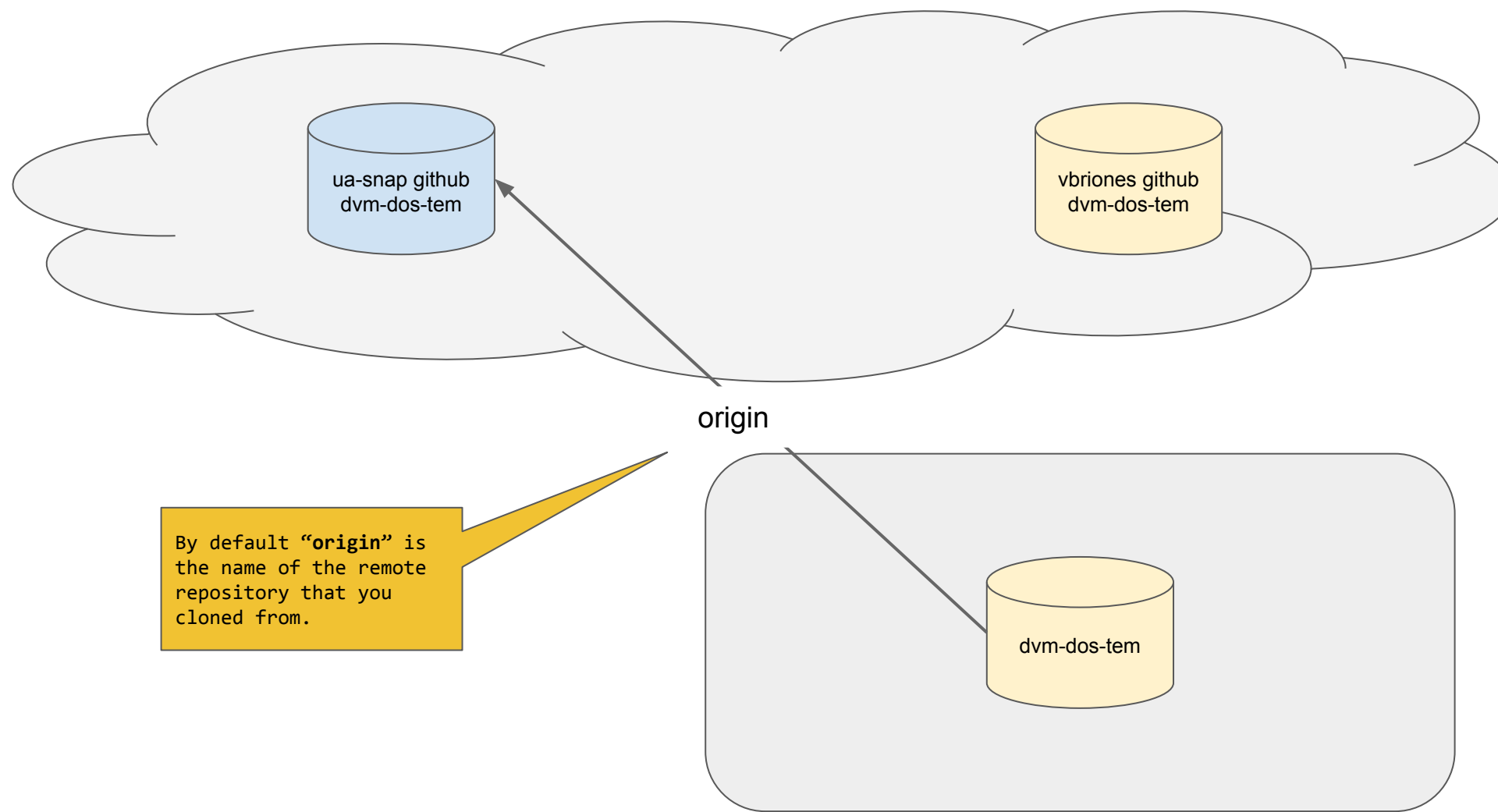
- A laptop.
- A server at your institution.
- A cloud instance (GCE, Amazon, Heroku, etc).
- An automated build server.
- etc.

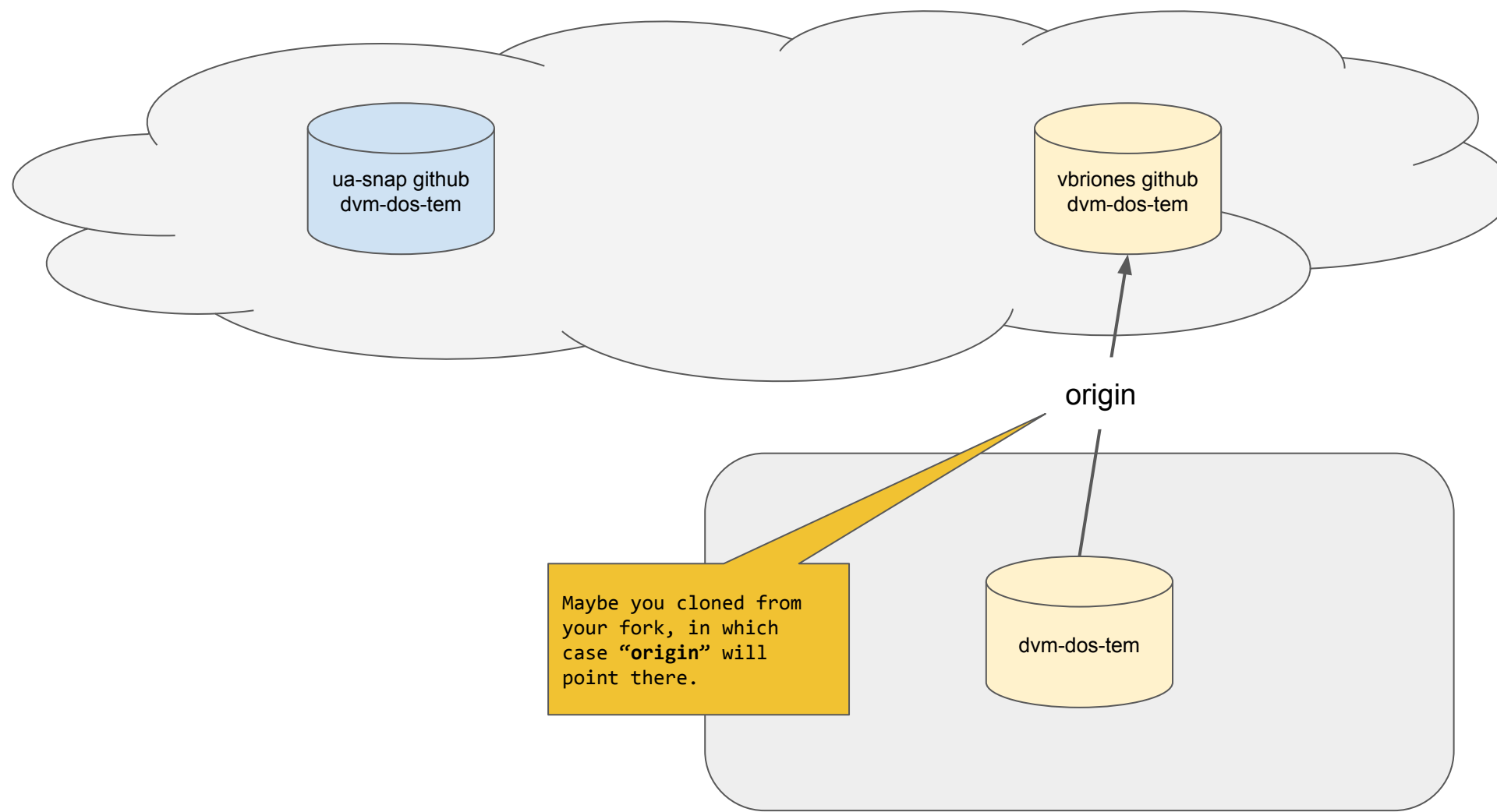


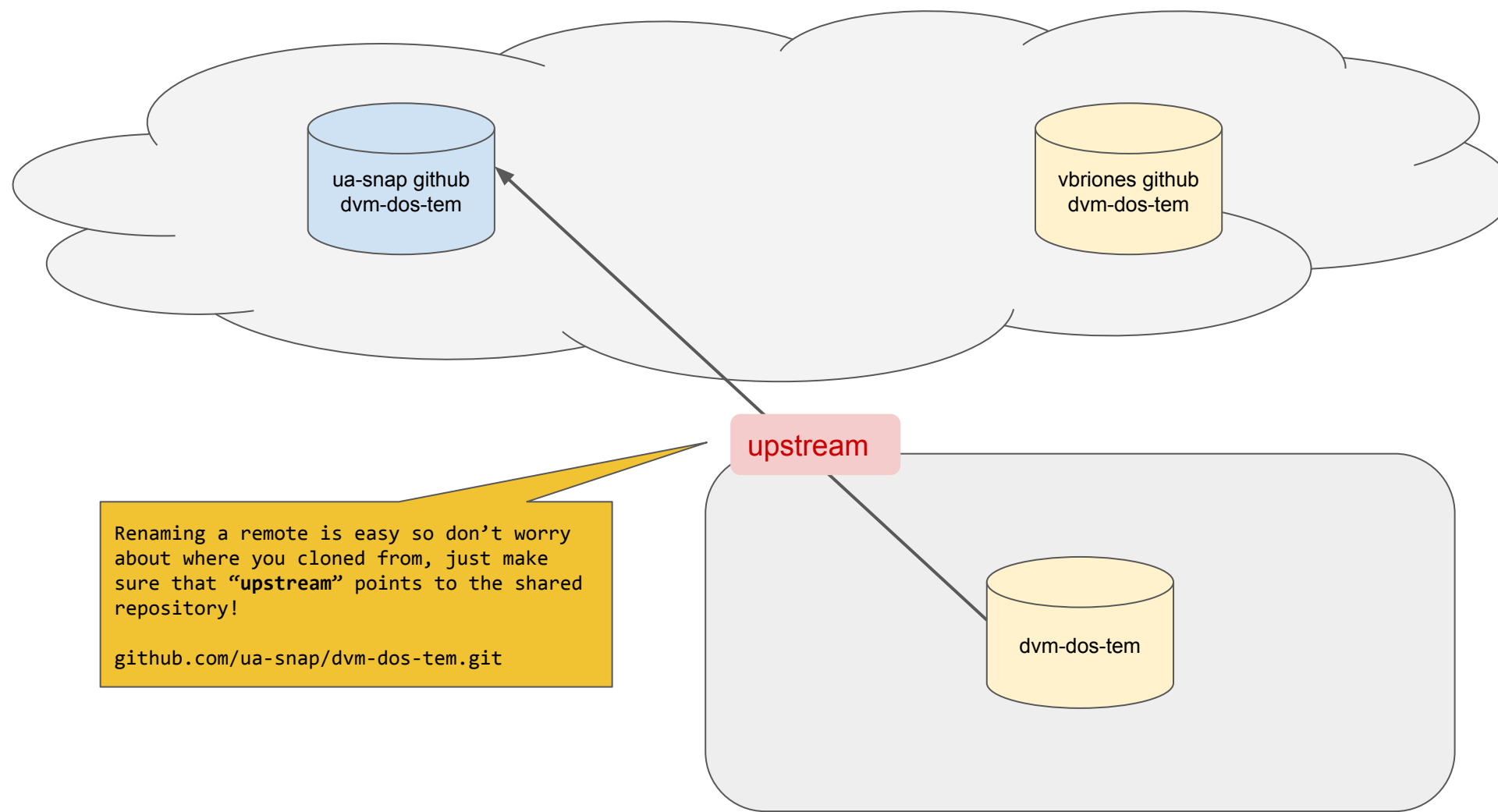


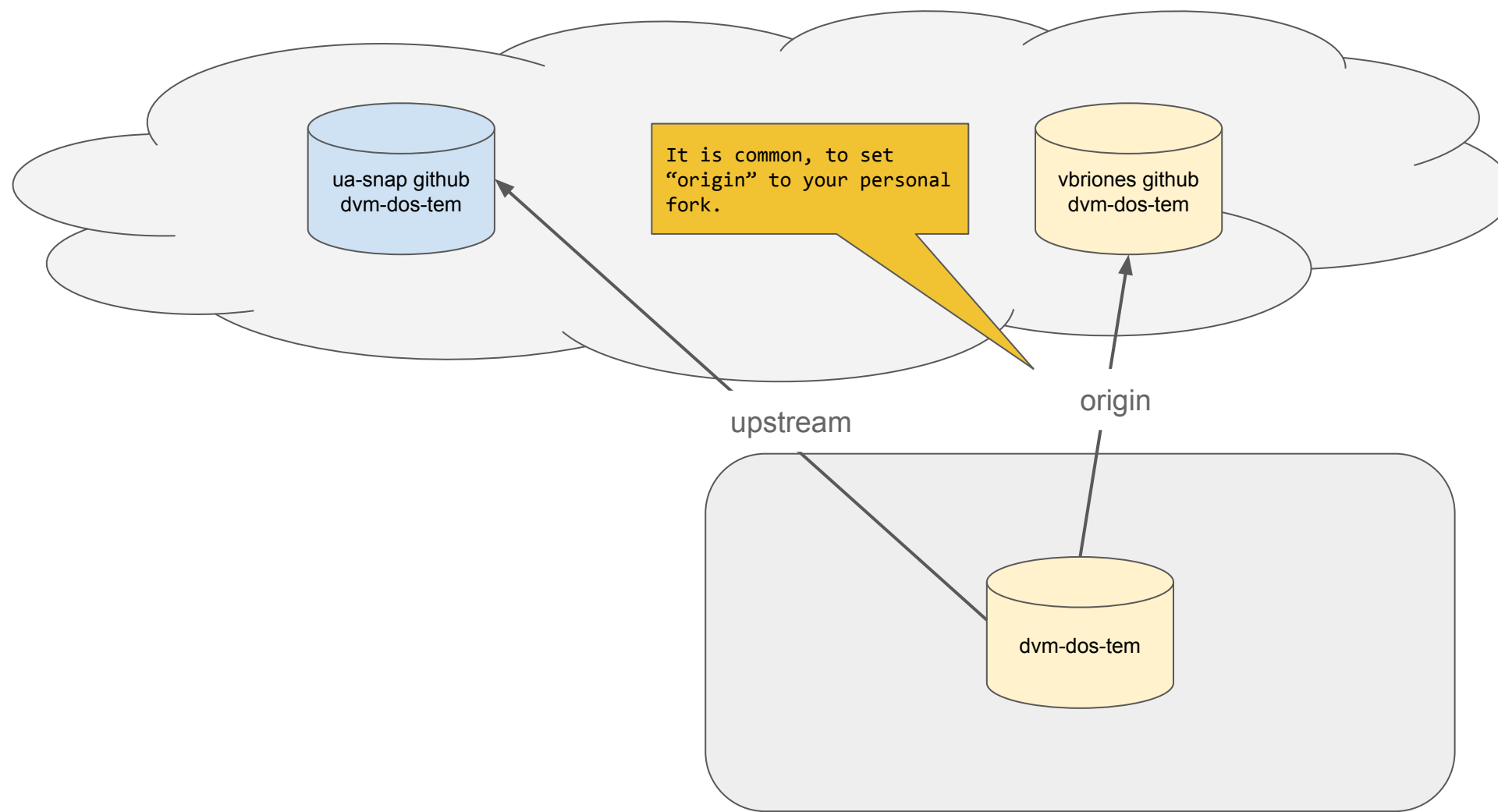
When you “fork” a project on Github,
you make a copy of the repository
under your Github account.

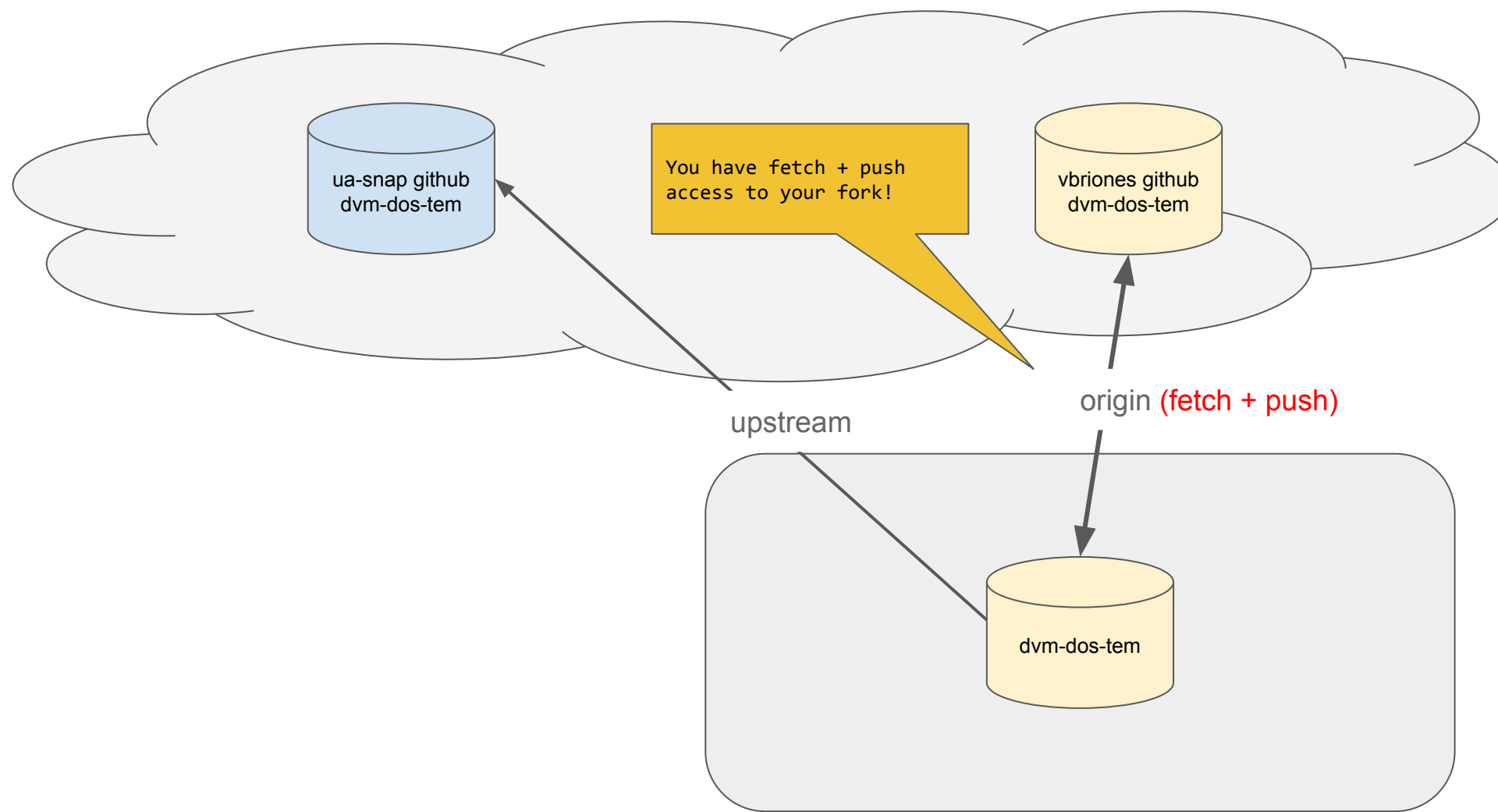


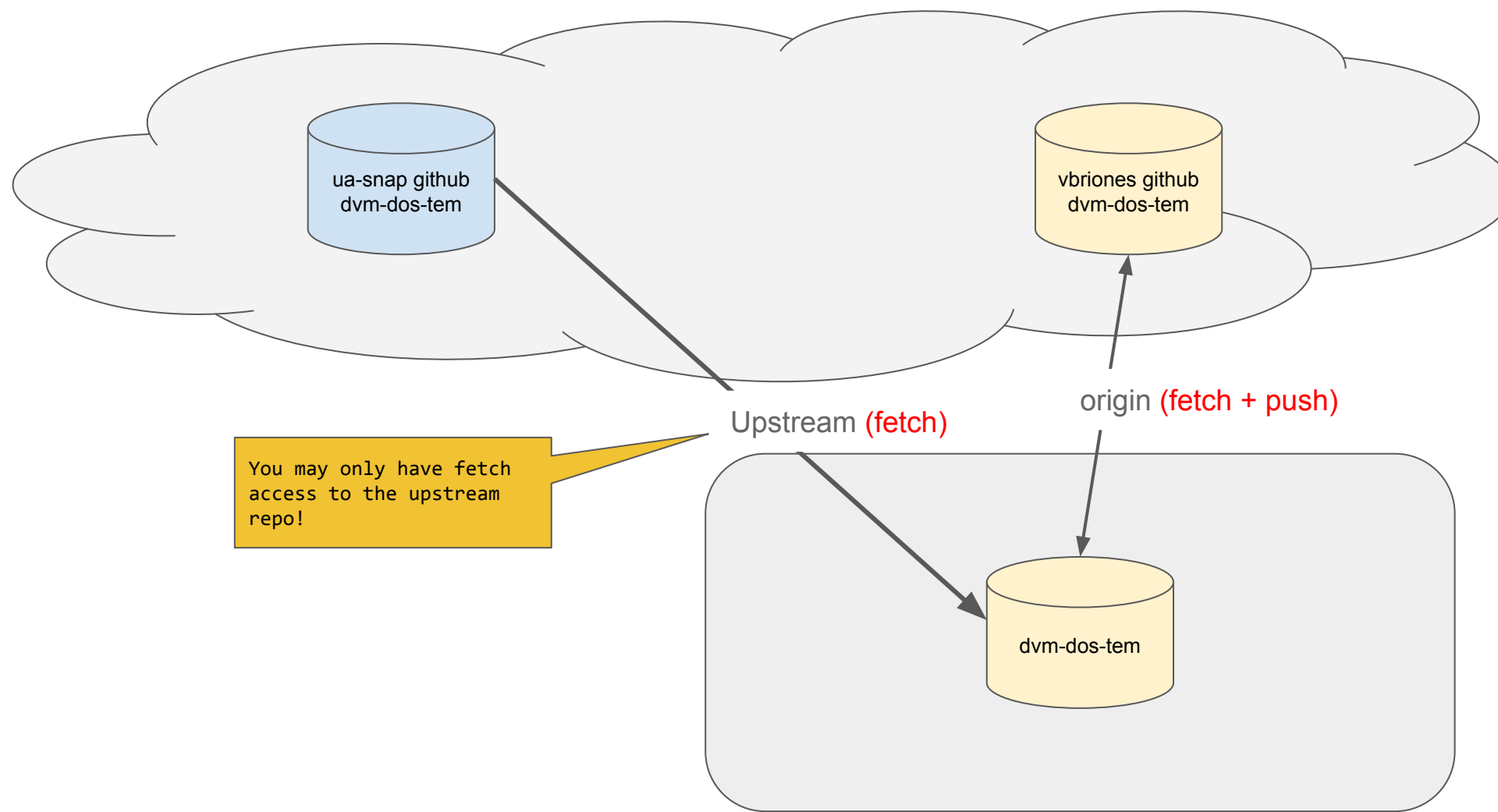


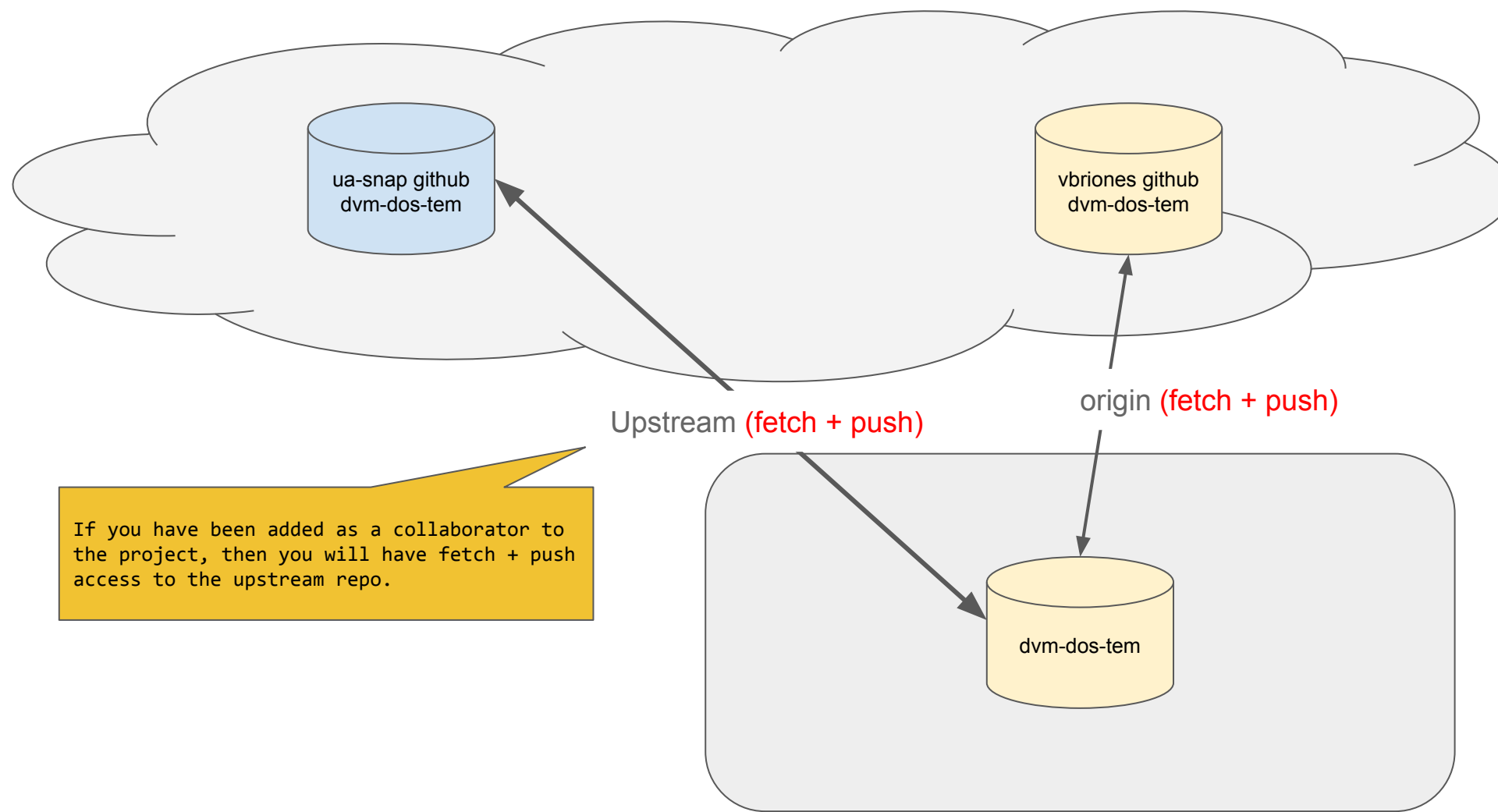












If you have push access to the upstream repo, you may not need to maintain your personal fork.

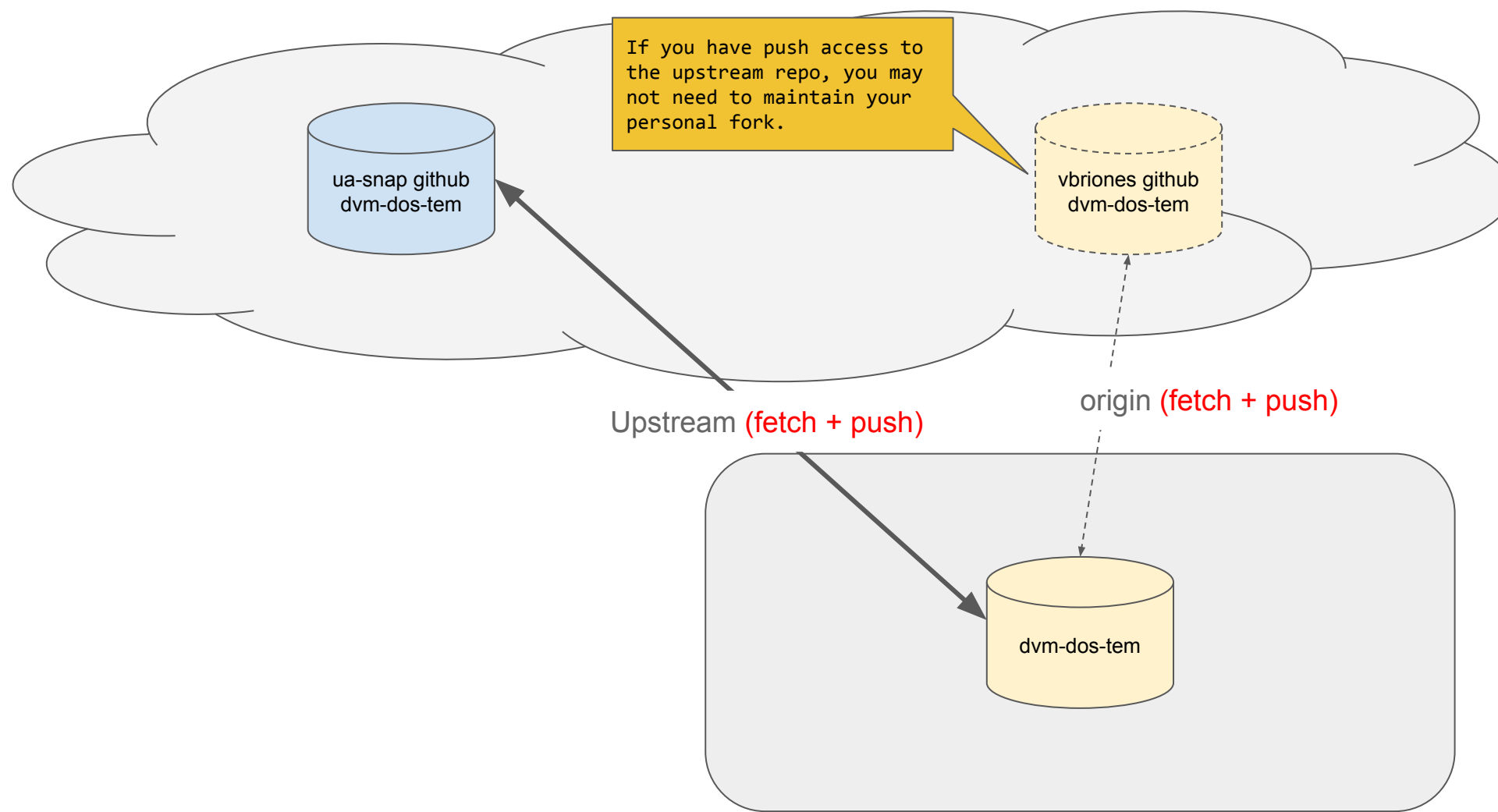
ua-snap github
dvm-dos-tem

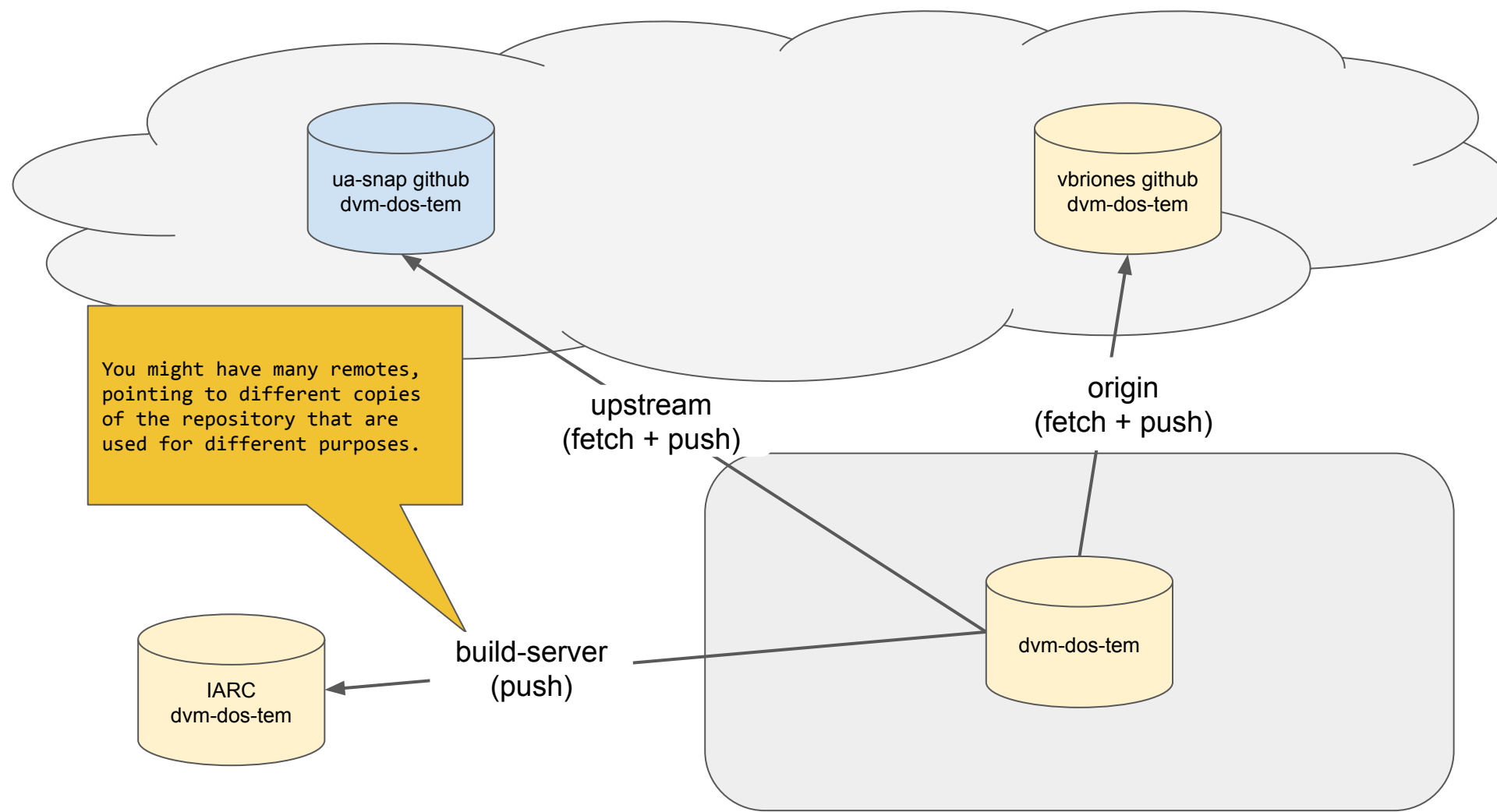
vbriones github
dvm-dos-tem

Upstream (fetch + push)

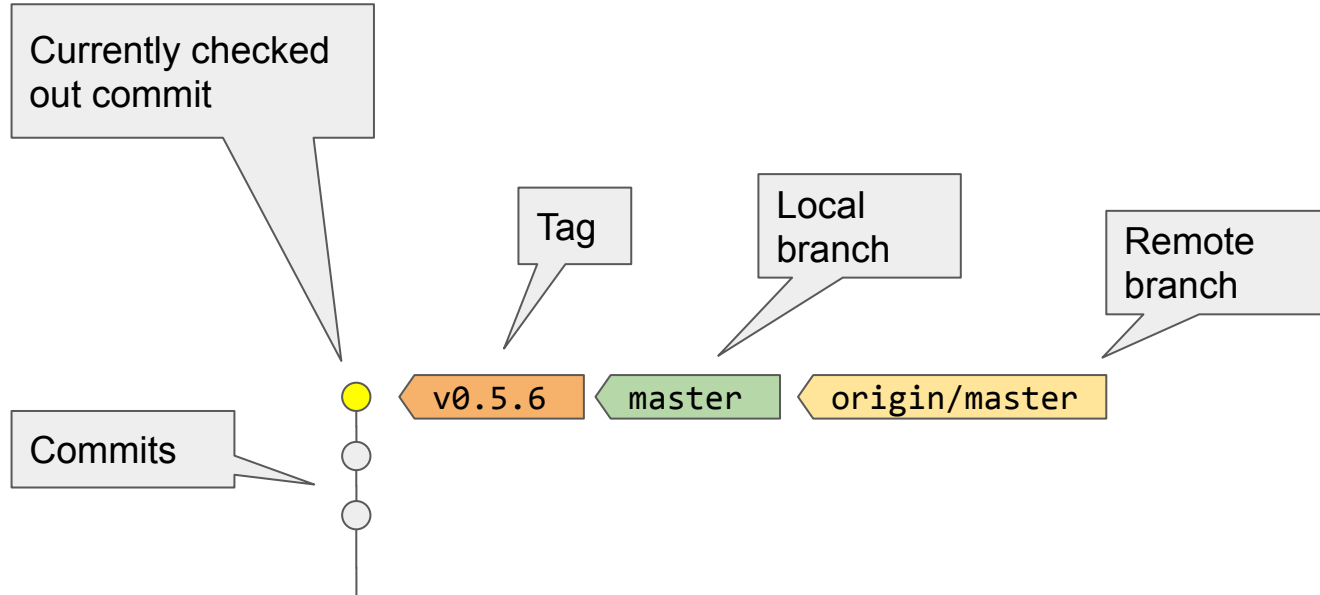
origin (fetch + push)

dvm-dos-tem

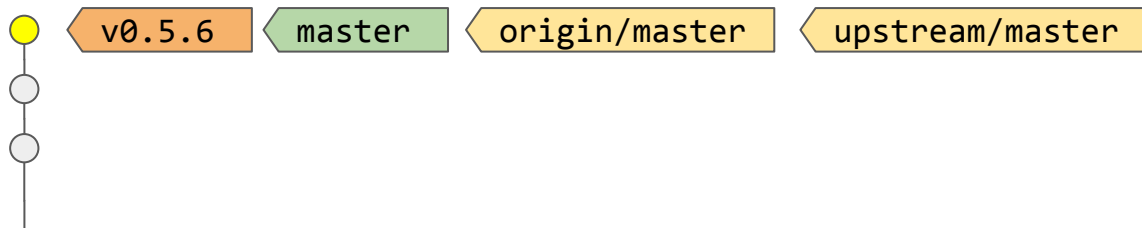




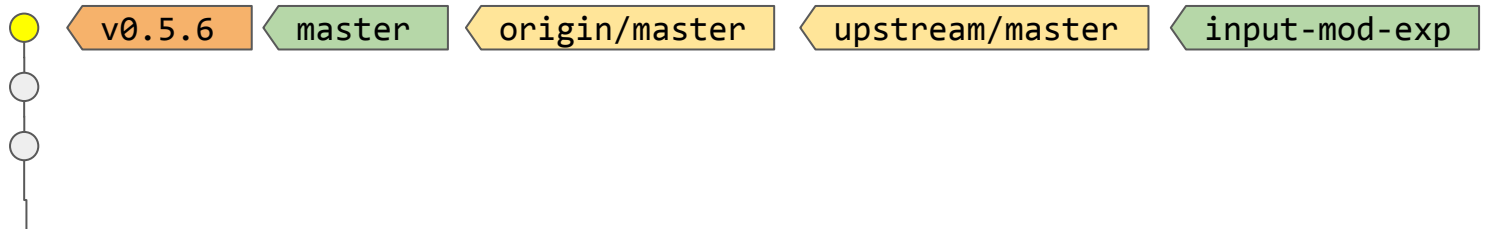
Single developer, using topic branch(es) and pull requests to contribute code to upstream repository.



```
$ git checkout master
```

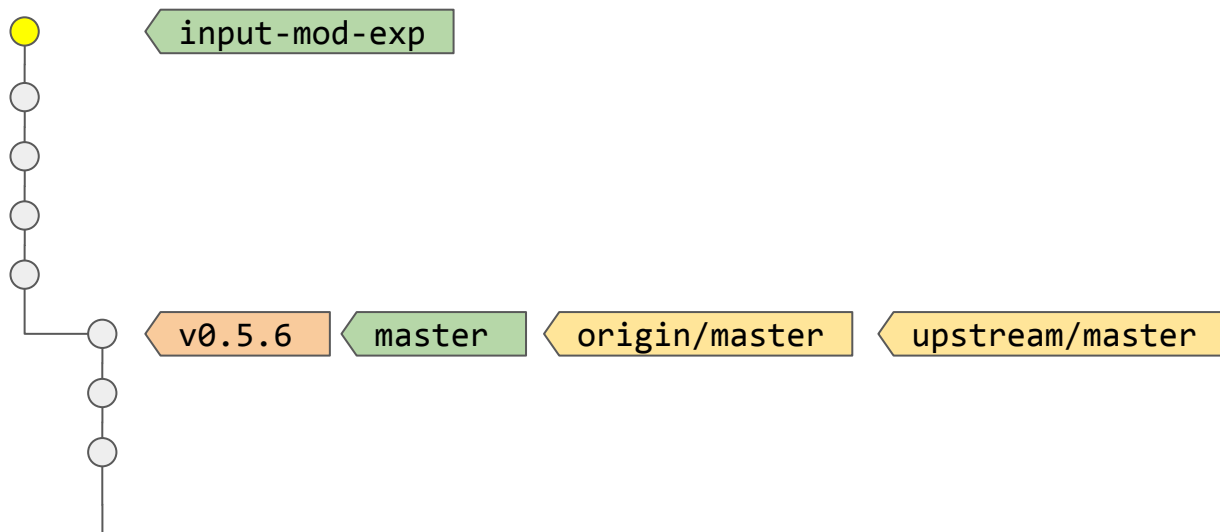
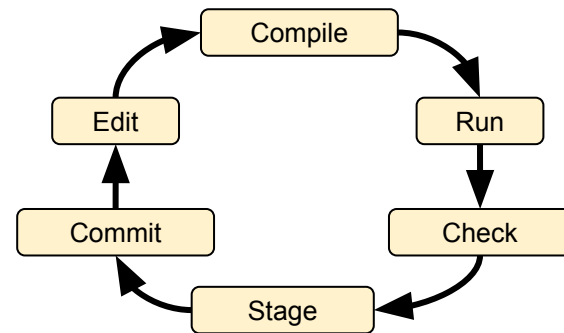


```
$ git checkout -b input-mod-exp
```

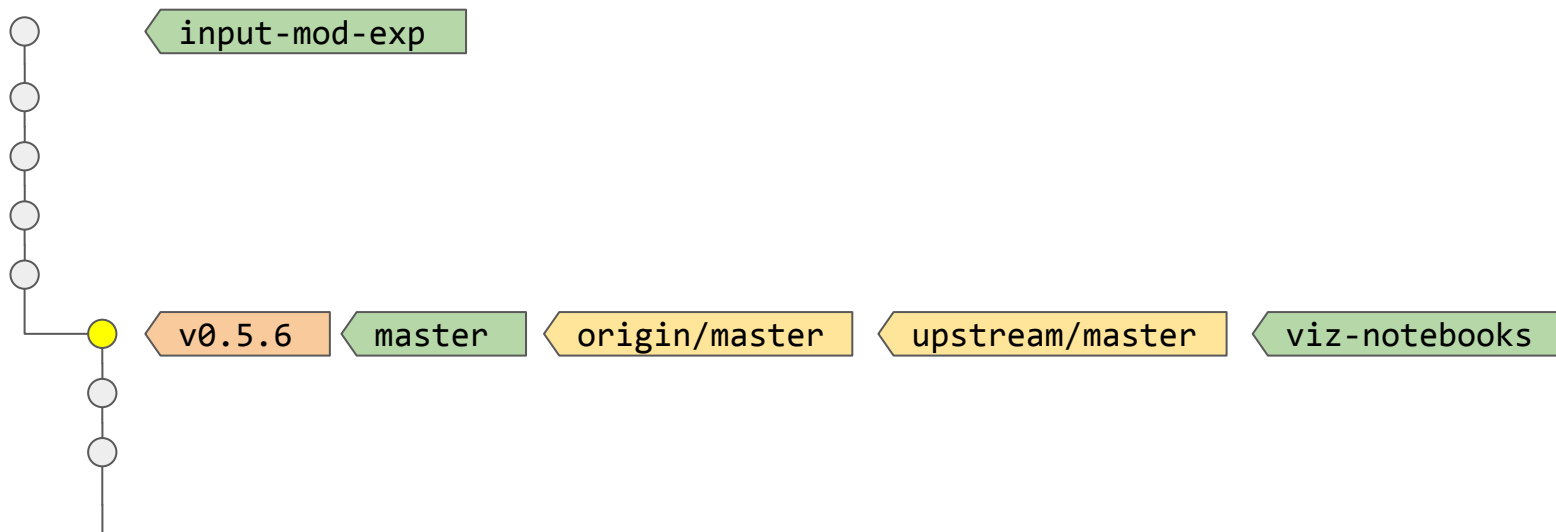


Core Developer Process Loop
with
version control

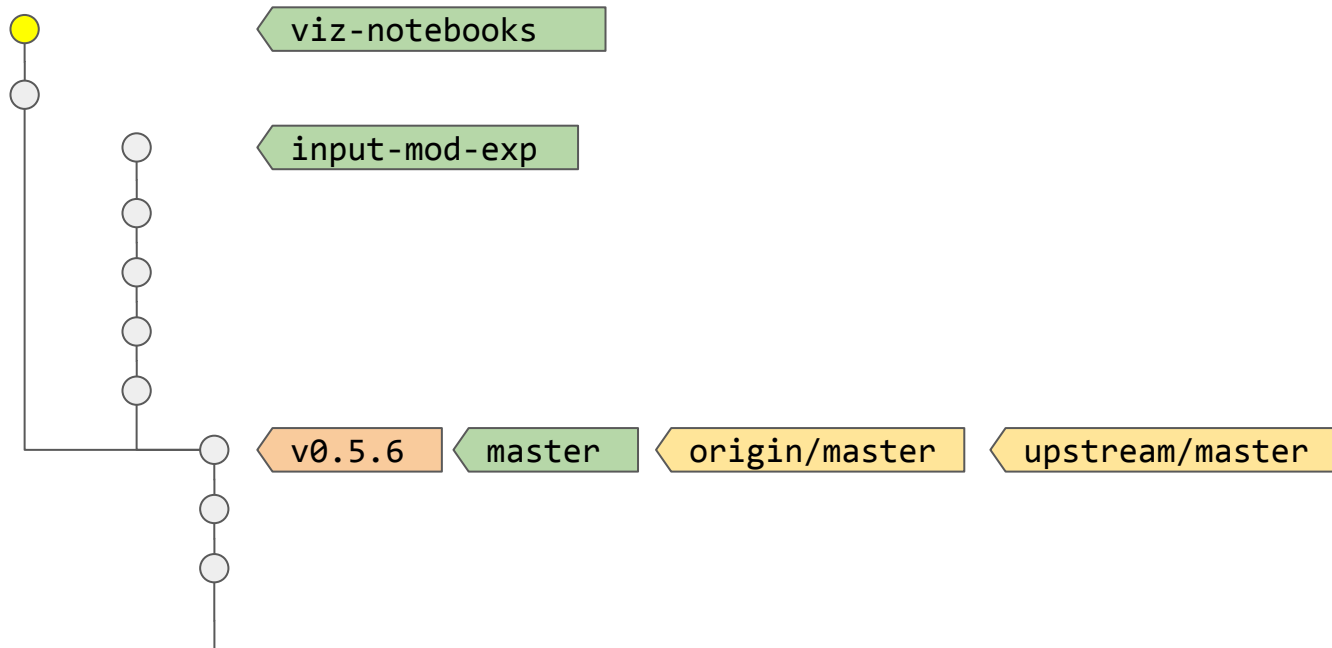
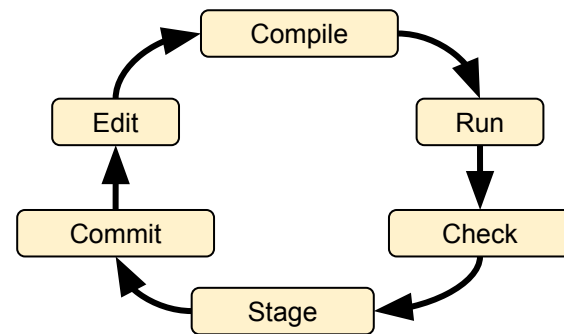
i.e. programming!!



```
$ git checkout master  
$ git checkout -b viz-notebooks
```

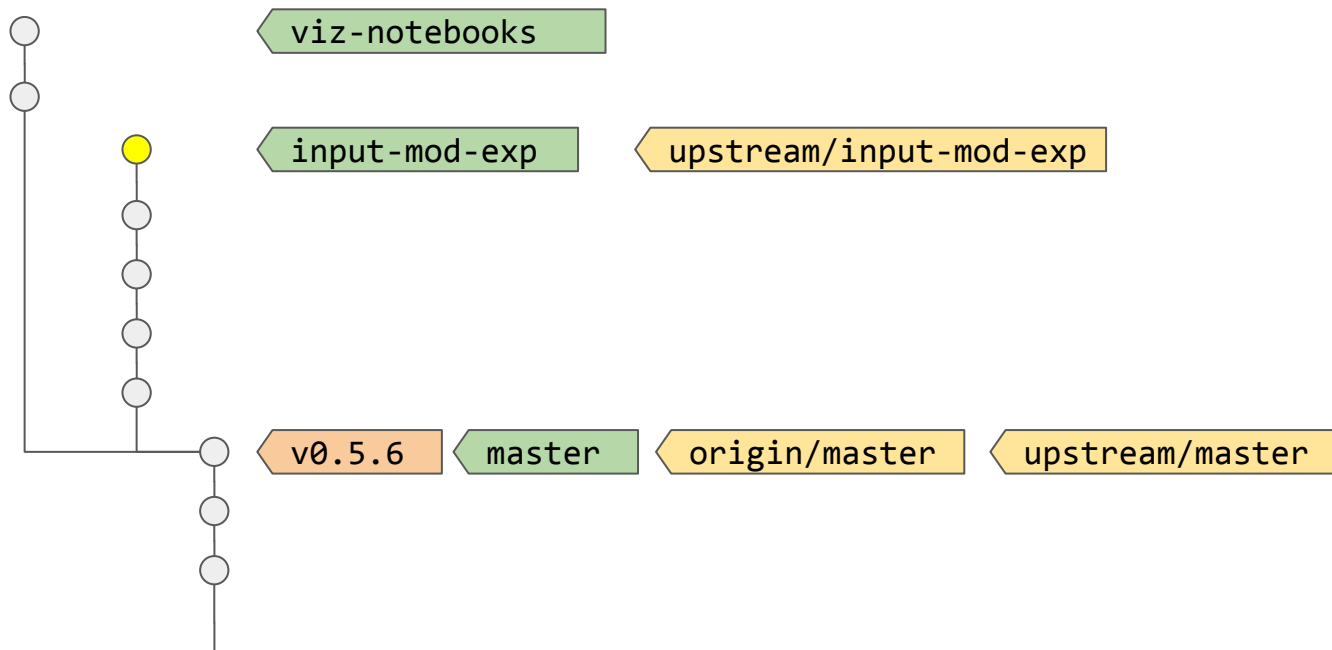


Core Developer Process Loop
with
version control!
i.e. programming!!



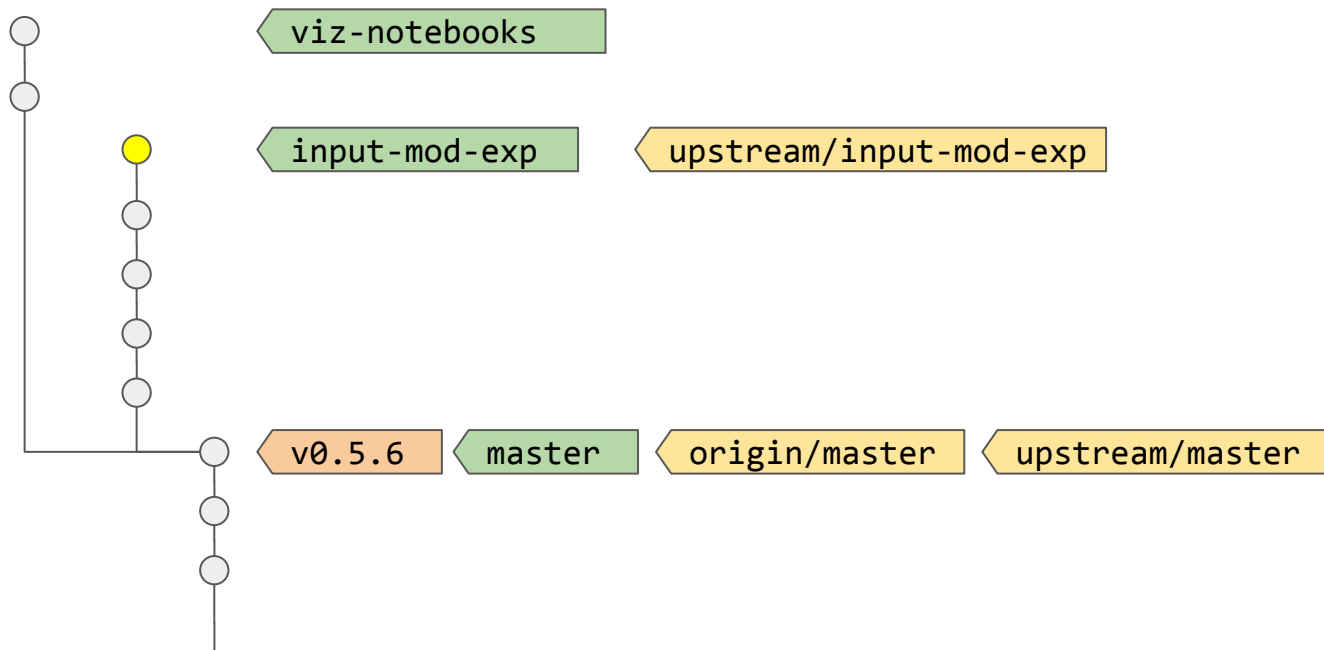
Share work with other people (and backup to cloud)

```
$ git checkout input-mod-exp  
$ git push upstream input-mod-exp
```



Github website:

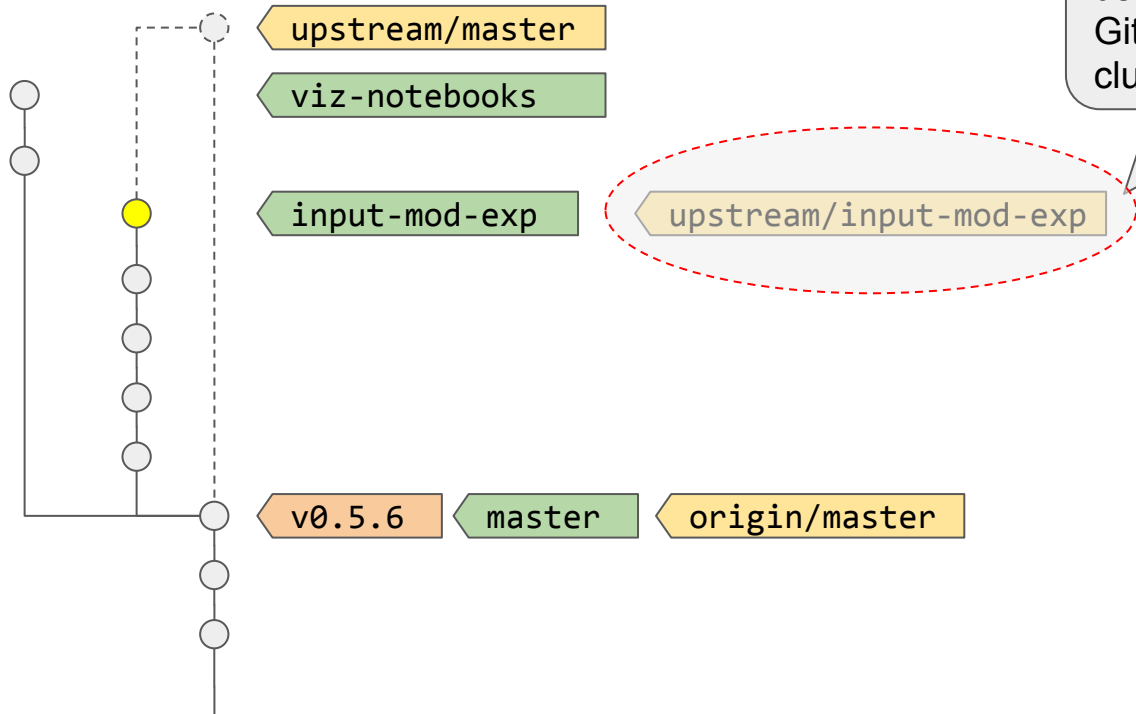
- Create Pull Request from input-mod-exp branch → master
- Group review, test, discussion, etc



Github website: tcarman2 or rarutter merges PR 🎉

See what happened upstream w/o changing anything locally:

```
$ git remote update
```

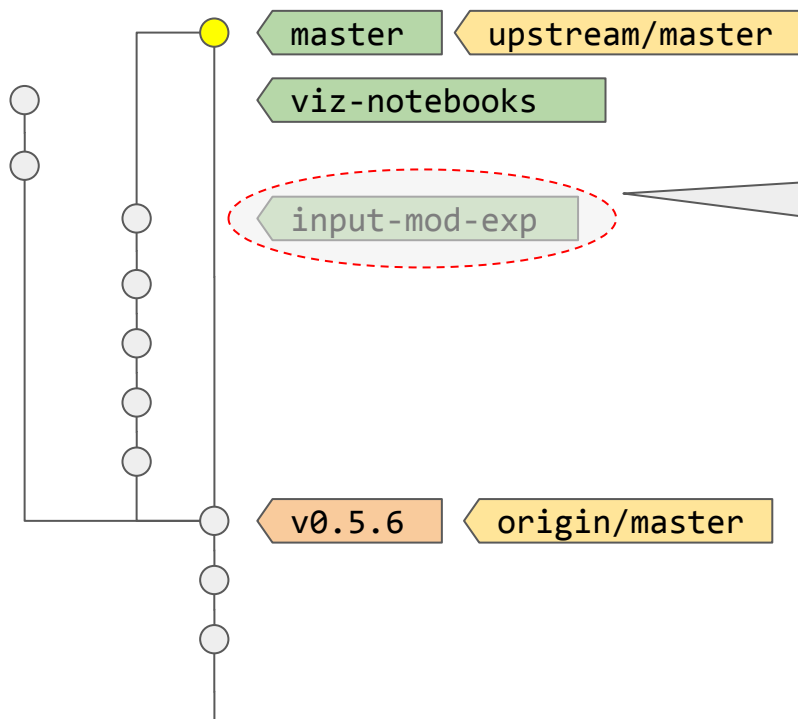


Merged branches are usually deleted on the Github repo to reduce clutter.

Update from upstream after new code merged to upstream/master

```
$ git checkout master
```

```
$ git pull upstream master
```



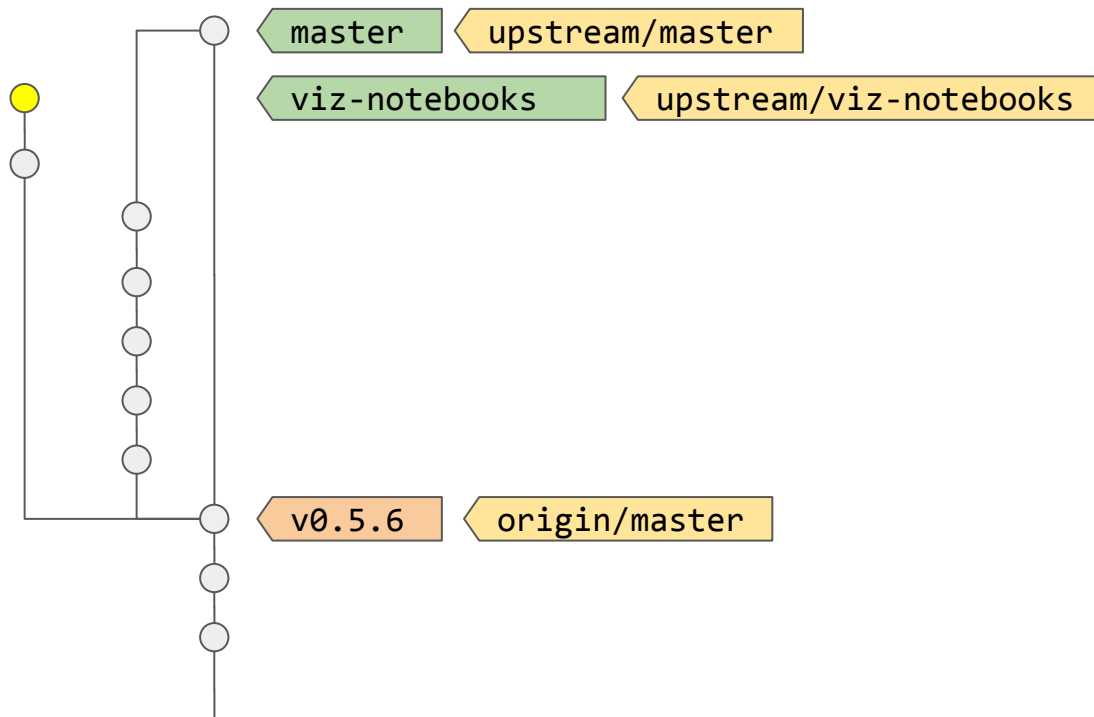
You may optionally delete this branch if you want to reduce clutter locally.

```
$ git branch -d input-mod-exp
```

Share work on viz-notebooks branch (and backup to cloud)

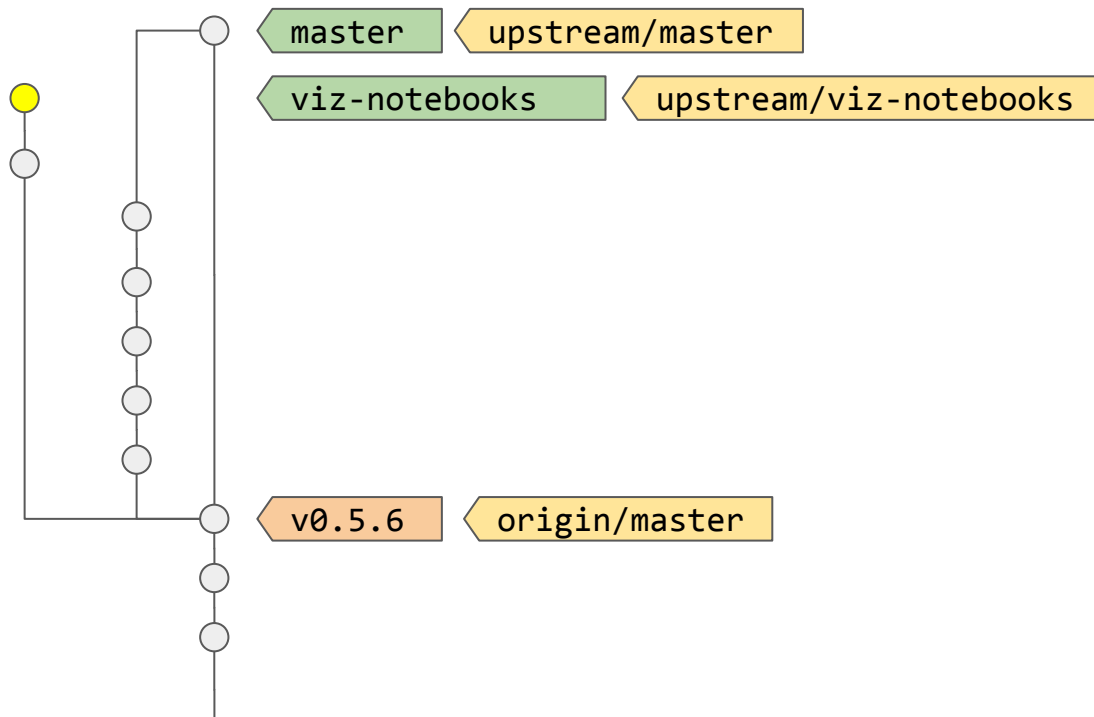
```
$ git checkout viz-notebooks
```

```
$ git push upstream viz-notebooks
```



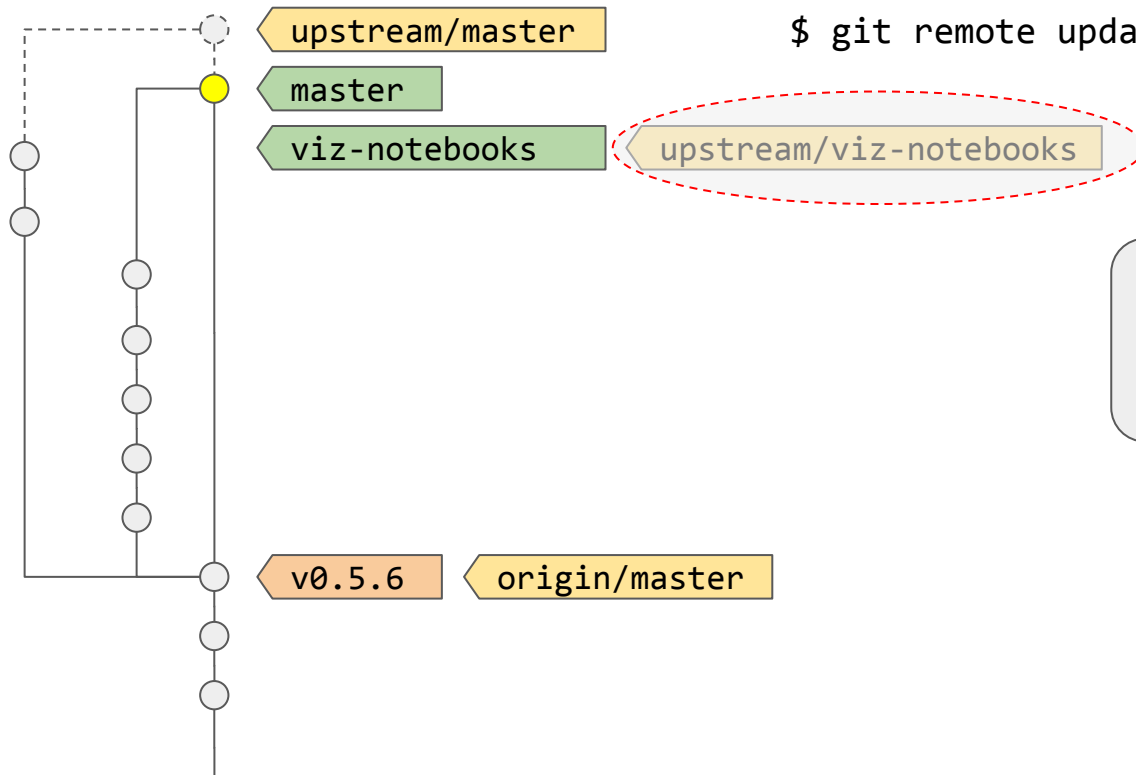
Github website:

- Create Pull Request from viz-notebooks → master
- Group review, test, discussion, etc



Github website: tcarman2
or rarutter merges PR 🎉

See what happened upstream w/o changing
anything locally:

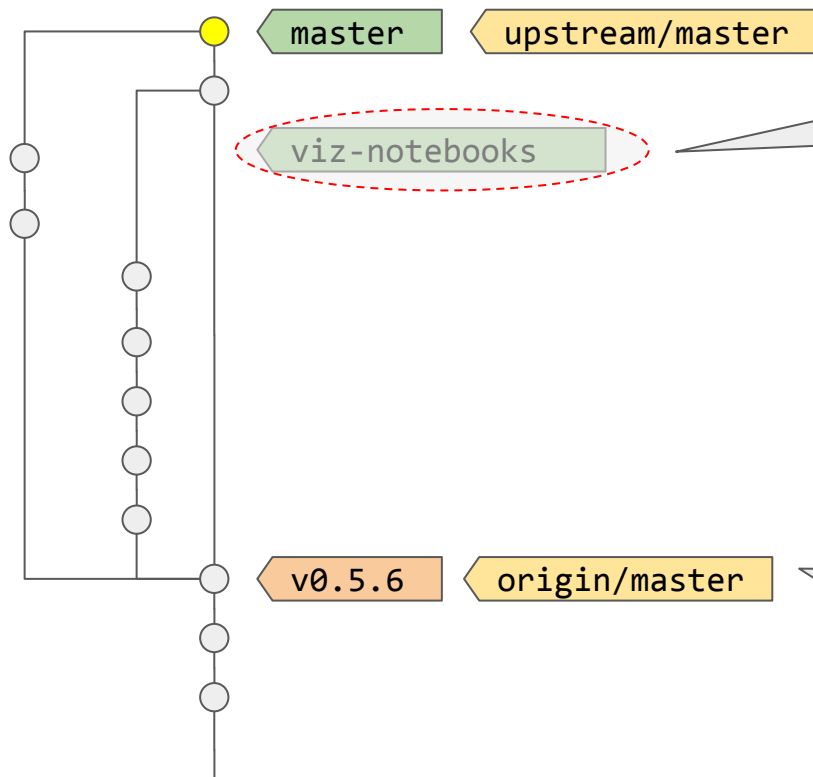


Merged branches are
usually deleted on the
Github repo to reduce
clutter.

Update from upstream after new code merged to upstream/master

```
$ git checkout master
```

```
$ git pull upstream master
```



You may optionally delete this branch if you want to reduce clutter locally.
`$ git branch -d viz-notebooks`

Note that we have left the `origin/master` branch behind. If you are maintaining your fork, then you would push from your master back to `origin/master`. In many cases, you don't need to maintain your own fork on Github.

Two developers working on a single topic branch and staying in sync using

```
$ git pull --rebase
```

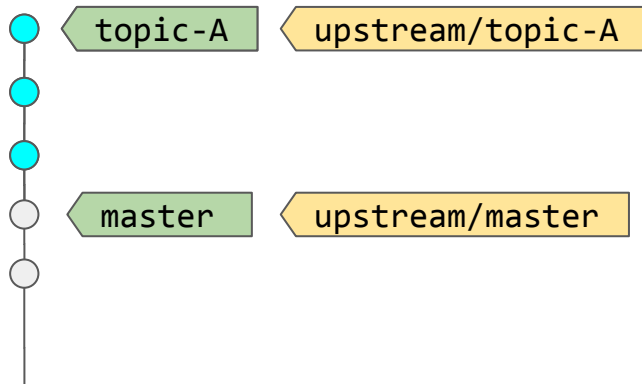
Person Y



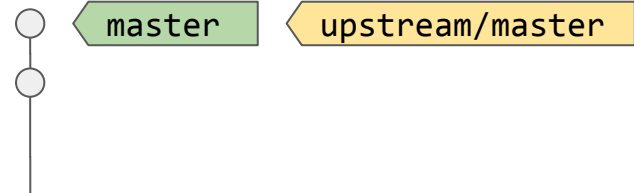
```
$ git checkout master  
$ git checkout -b topicA
```

```
# edit, compile, run, check commit, 3x
```

```
$ git push upstream topicA
```



Person Z



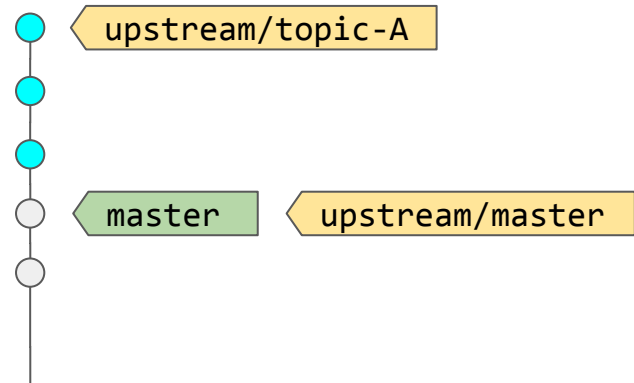
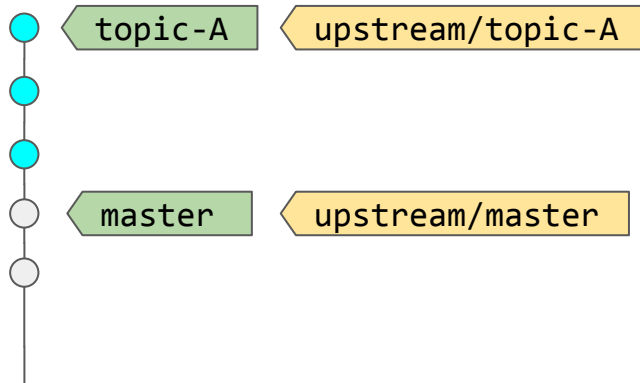
Person Y



Person Z



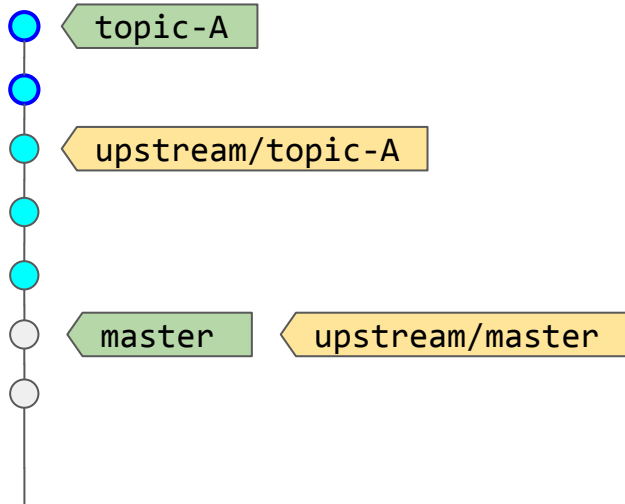
```
$ git remote update
```



Person Y



edit, compile, run, check commit, 2x

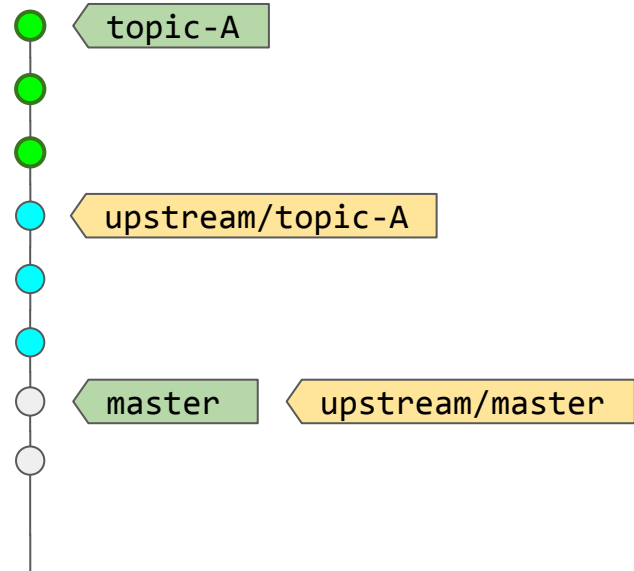


Person Z



\$ git checkout topicA

edit, compile, run, check commit, 3x

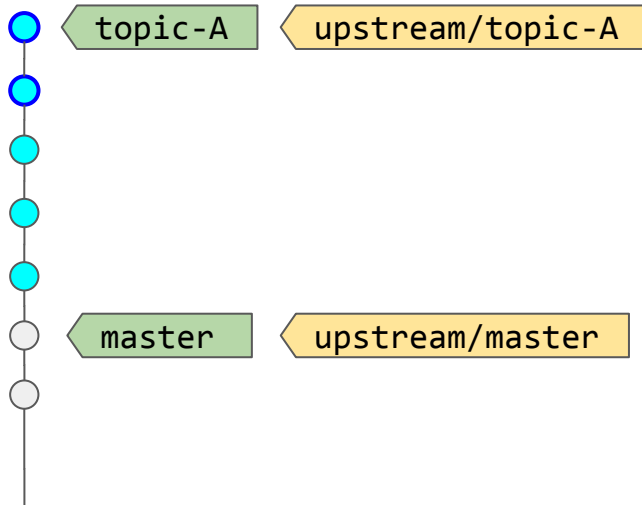


Person Y

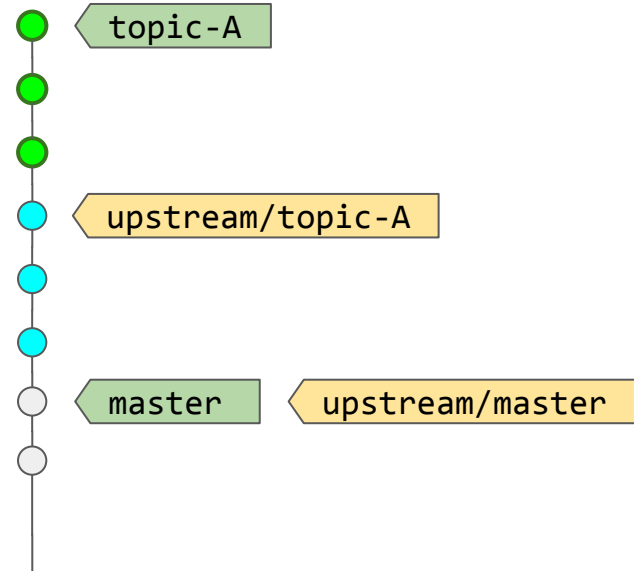


```
# Check that no one else pushed first!!  
$ git remote update
```

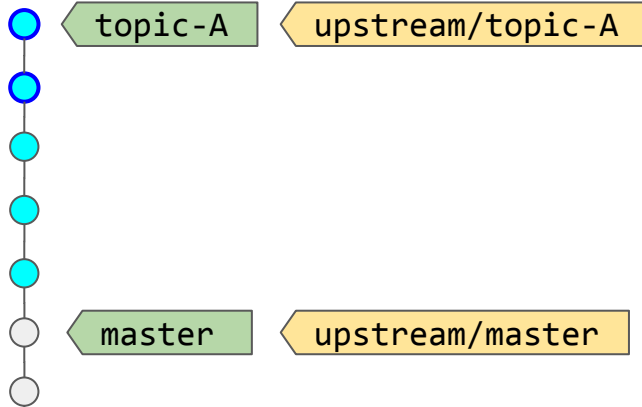
```
# Then push...  
$ git push upstream topicA
```



Person Z



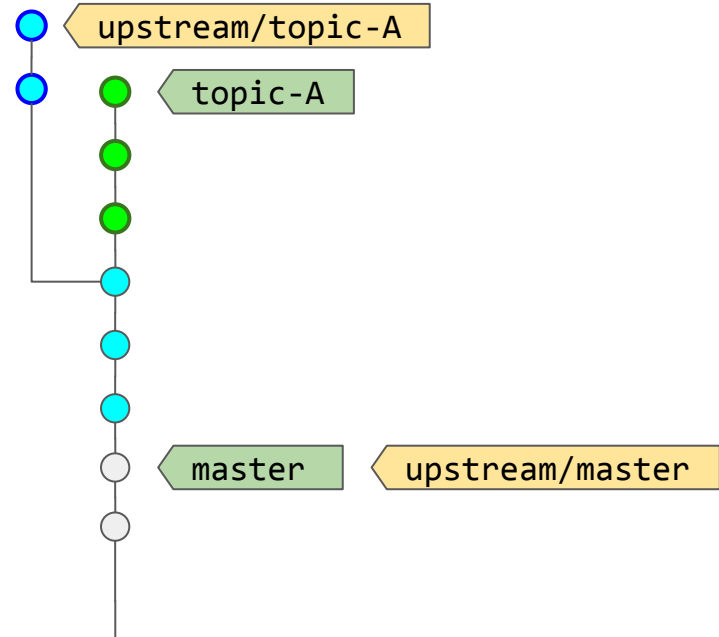
Person Y



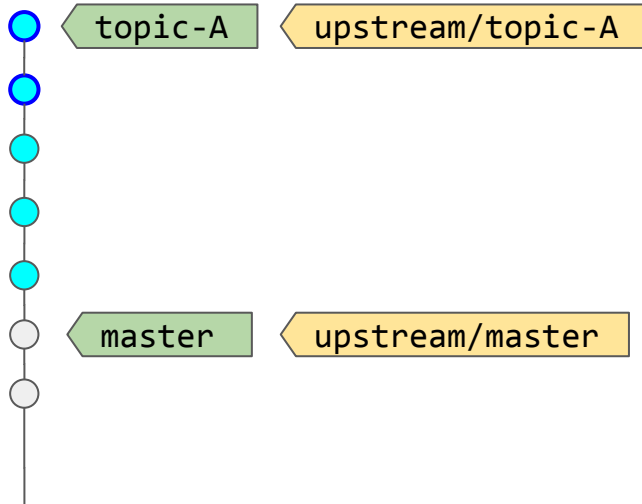
Person Z



\$ git remote update



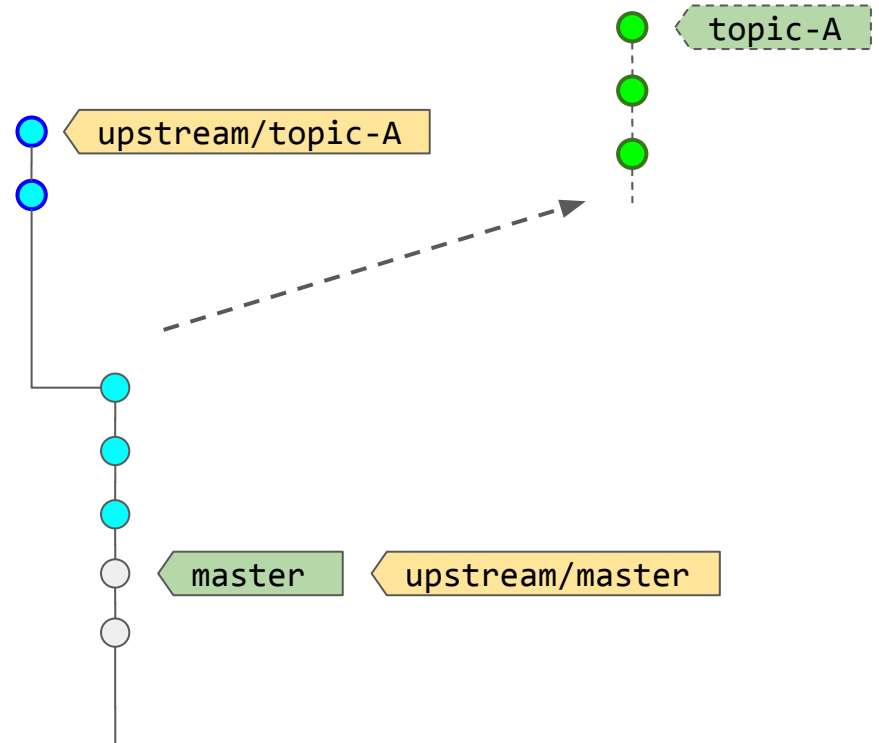
Person Y



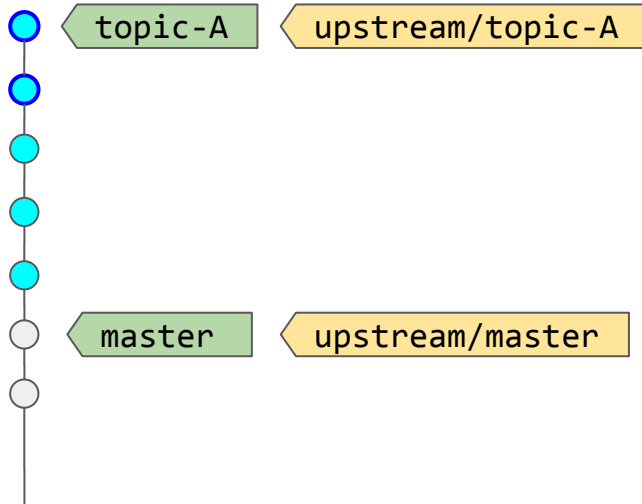
Person Z



```
$ git pull --rebase upstream topicA
```



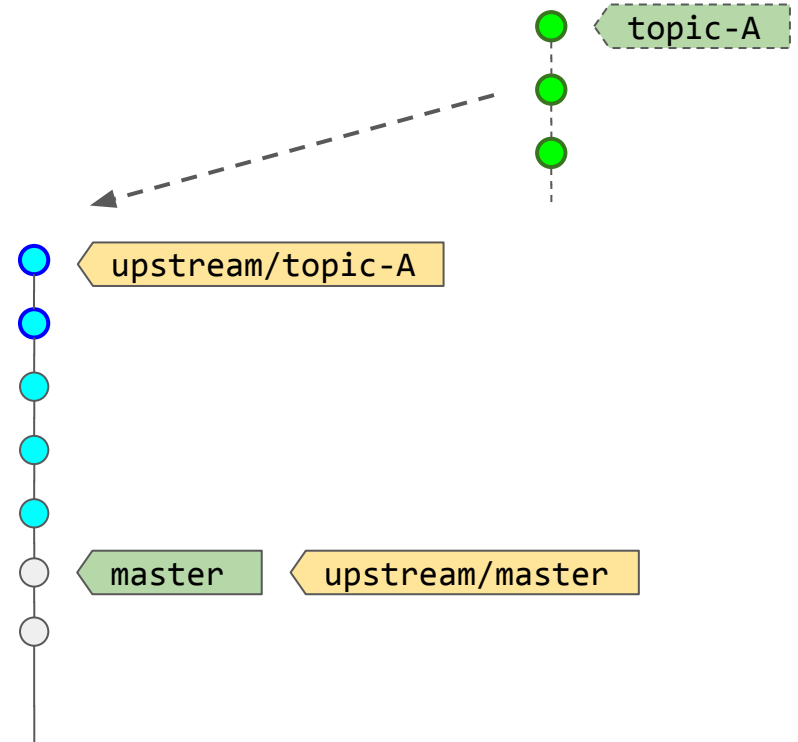
Person Y



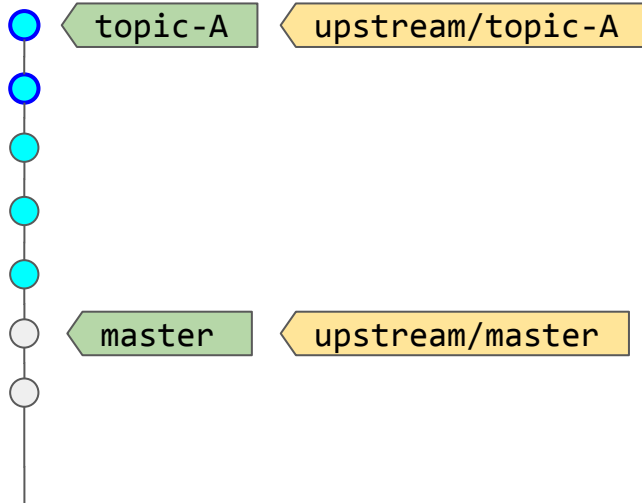
Person Z



```
$ git pull --rebase upstream topicA
```



Person Y

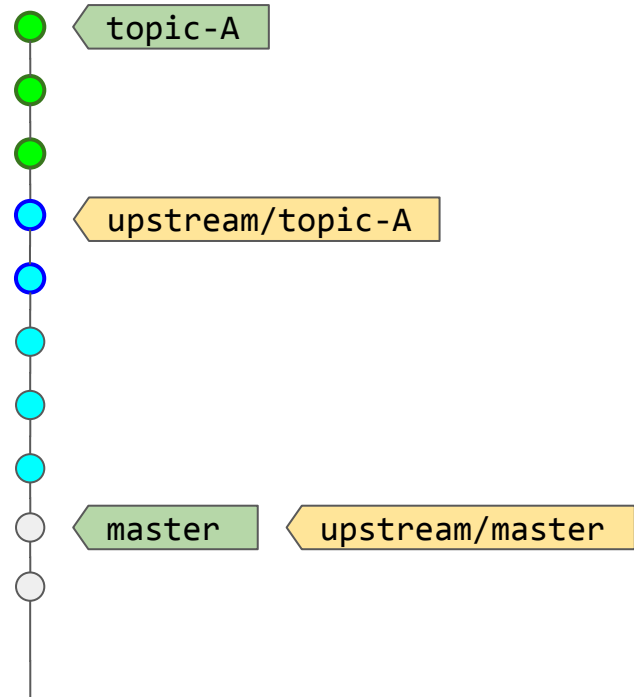


Person Z

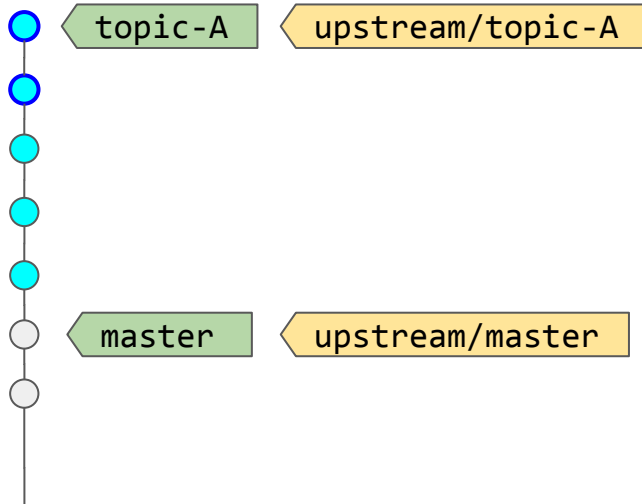


```
$ git pull --rebase upstream topicA
```

COMPLETE!



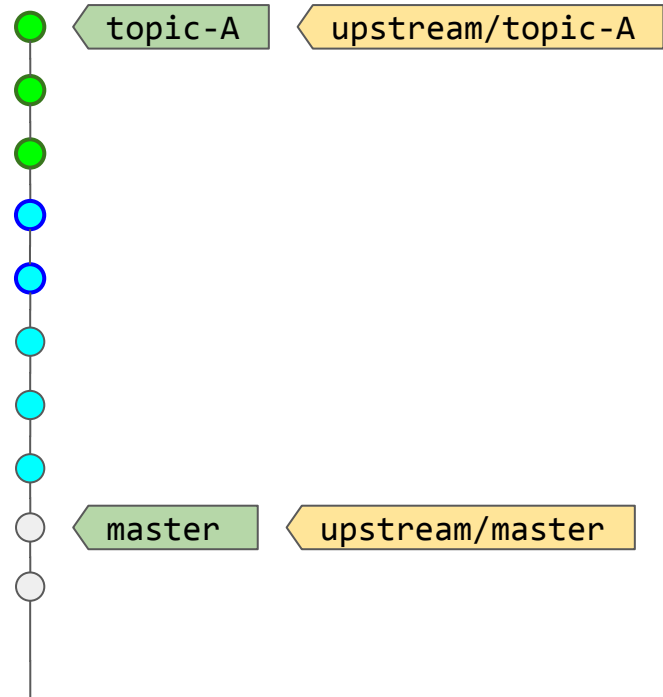
Person Y



Person Z



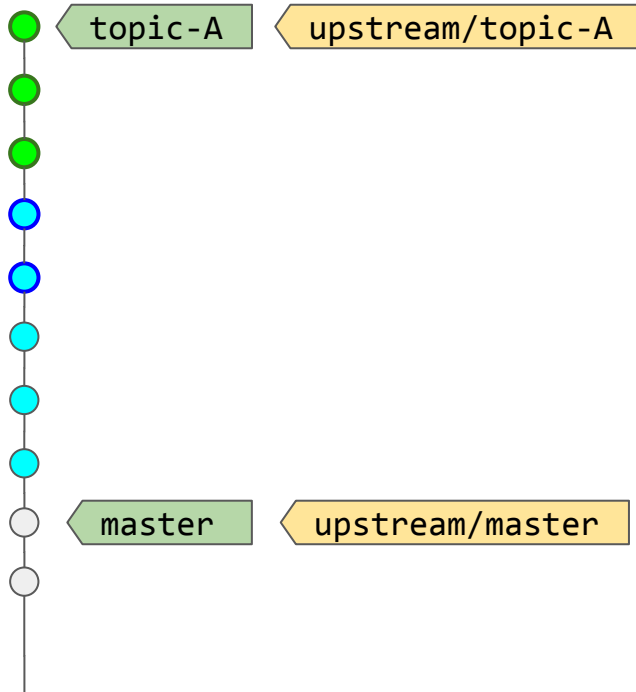
\$ git push upstream topicA



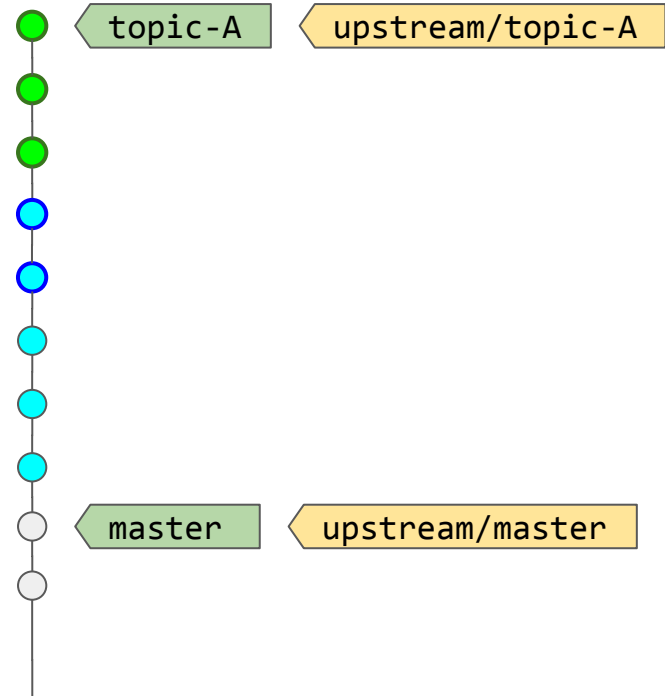
Person Y



```
$ git pull --rebase upstream topicA
```



Person Z



When using git, keep in mind what should and should not be tracked. In light of the previous workflows, what would happen if:

- You tracked model outputs?
- You tracked model configurations for runs?
- Every possible collaborator tracked model configurations for their runs?
- Person Y indents with 2 spaces and Person Z indents with 4 spaces?
- Person Y and Z track their personal setup files (.vscode, .vimrc, etc)

