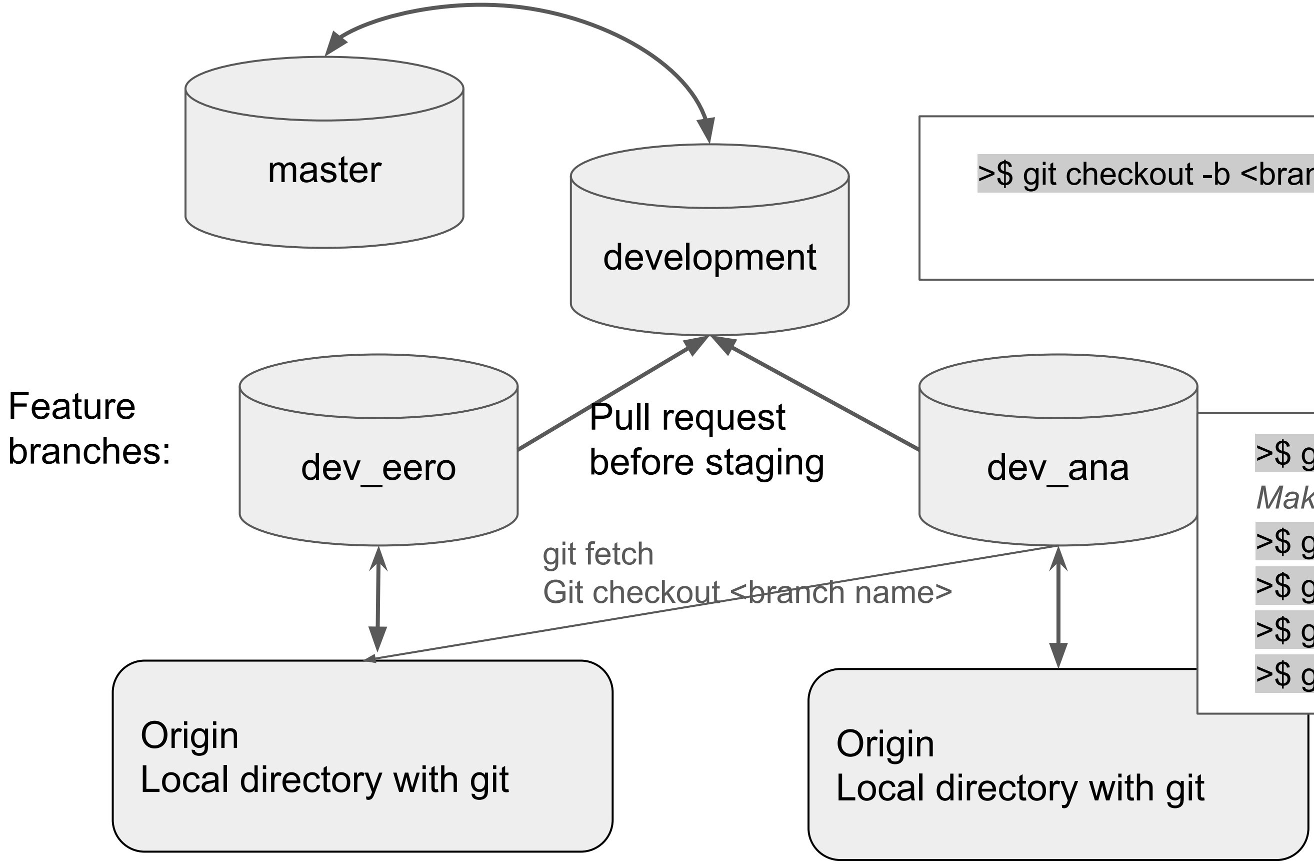
Git Structure in the Repo

Good source:
https://www.robinwieruch.de/git-team-workflow



```
>$ git checkout -b <branch_name>  // Create a branch
```

```
>$ git checkout <branch_name>
```
Make changes in the files
```
>$ git add .
>$ git commit -m "<message>"
>$ git push origin <branch_name>
>$ git pull --rebase origin <branch_name>
```

keep your feature branch up to date with the development branch:

```
>$ git checkout <branch_name>            // checks out your feature branch
```

follow: How to keep a branch up-to-date?

```
>$ git checkout development                        // checks out development branch
```

follow: How to keep the development branch up-to-date?

```
>$ git checkout      <branch_name>// checks out your feature branch
```

```
>$ git rebase development           // merges all your changes from your feature branch on top of the development branch
```

```
>$ git push origin <branch_name>   // pushes the updated feature branch to your remote repository
```

If the last command fails, it can be because some of your branch has been pushed already, and now after the last changed the history is different, which causes a conflict. Try then this, which will overwrite all changes in the repo (not smart if someone else has made changes in the same branch):

```
>$ git push -f origin <branch_name>   // pushes the updated feature branch to your remote repository by force
```

Merge from feature branch to development to master

Open a pull request for my feature branch

Wait for discussion and review

Possibly add some changes with commits

```
>$ git checkout development
>$ git merge --no-ff <my feature branch>  //Merges my branch with dev.branch
>$ git push origin development
```