LAB SQUAD

# Git Workflows

# Agenda

1 Compare workflows

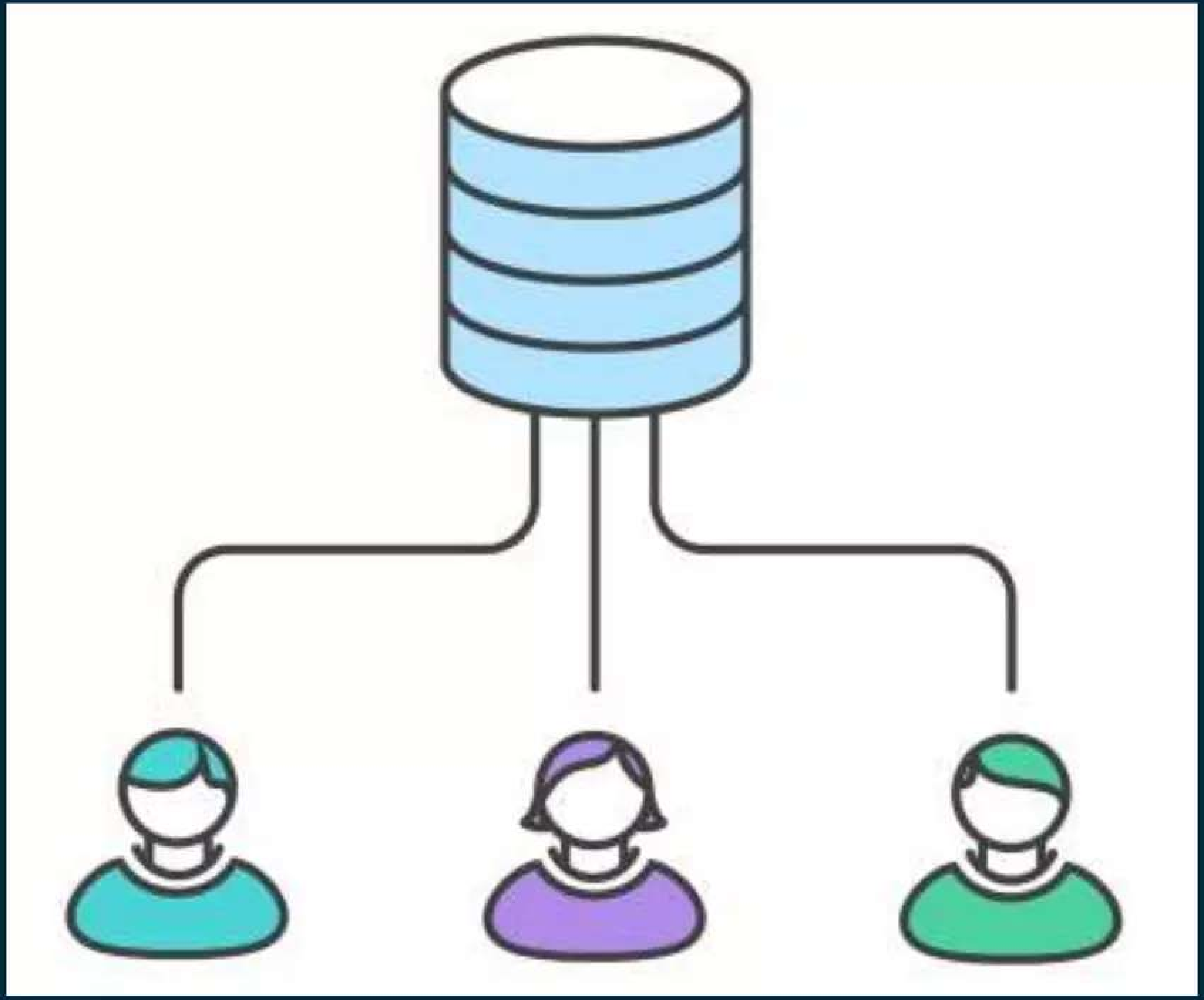
2 Git Workflows - branding model

3 Special cases - how to resolve

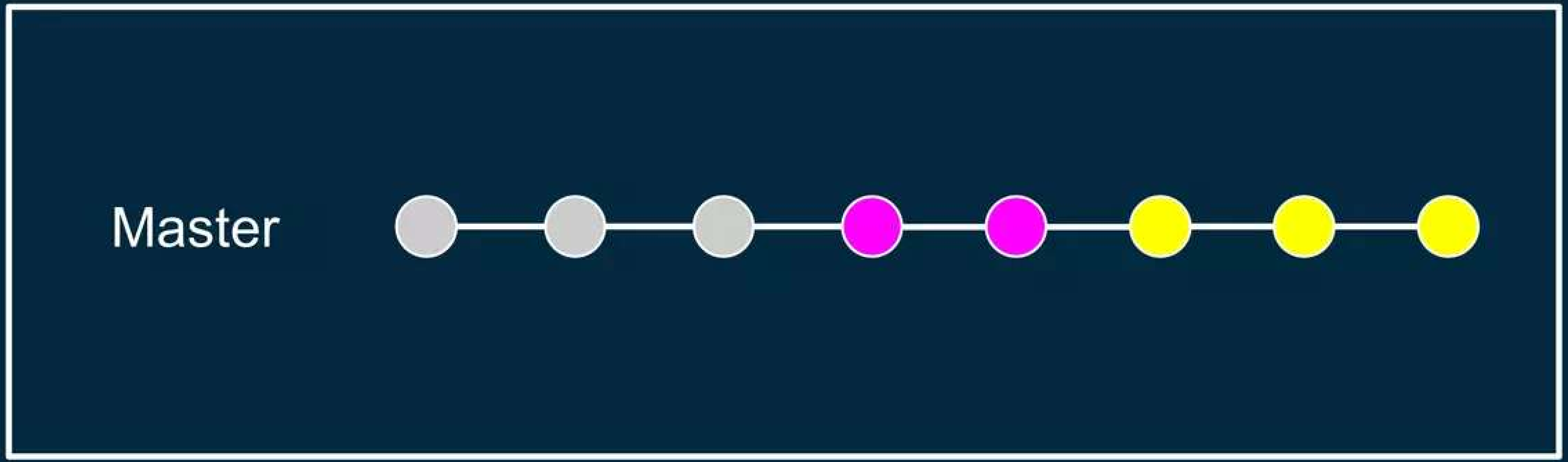
# Workflows

Compare and review

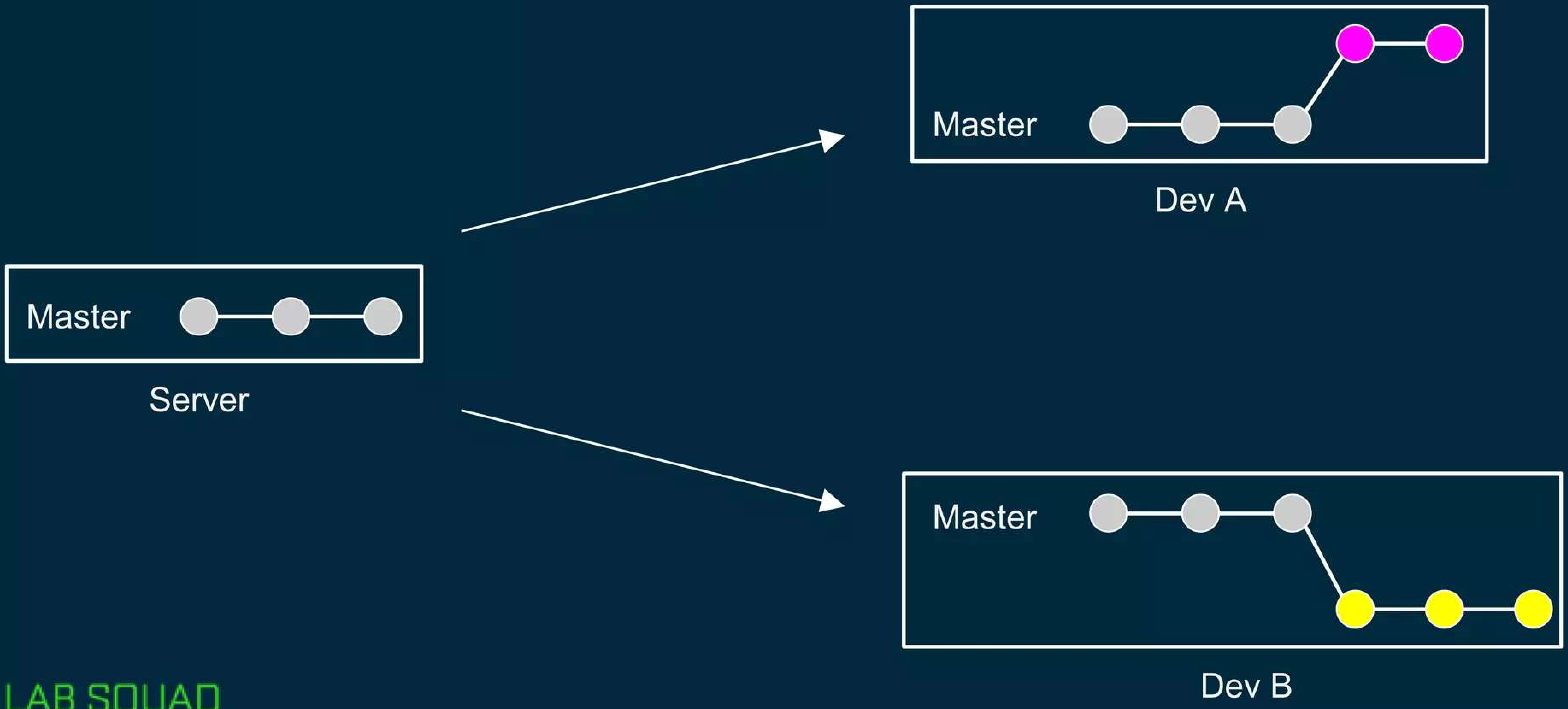
# Centralized Workflow (1)



# Centralized Workflow (1)

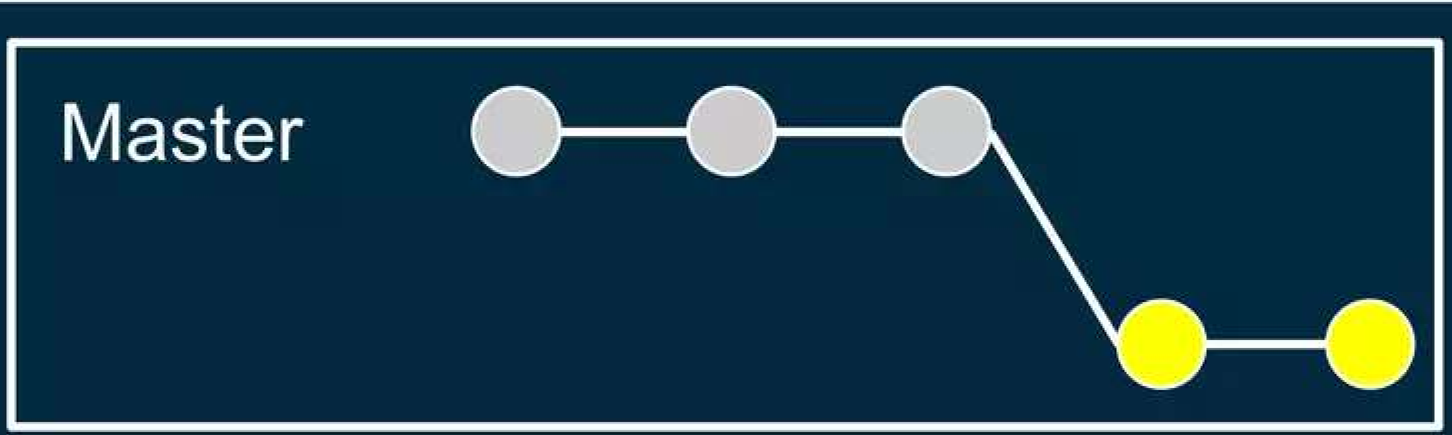


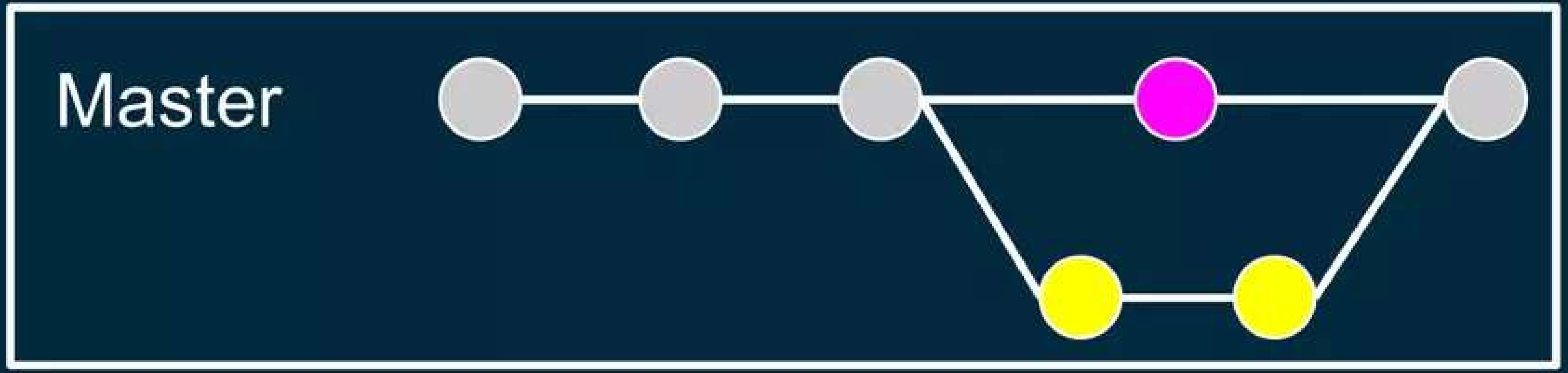
# Centralized Workflow (2)

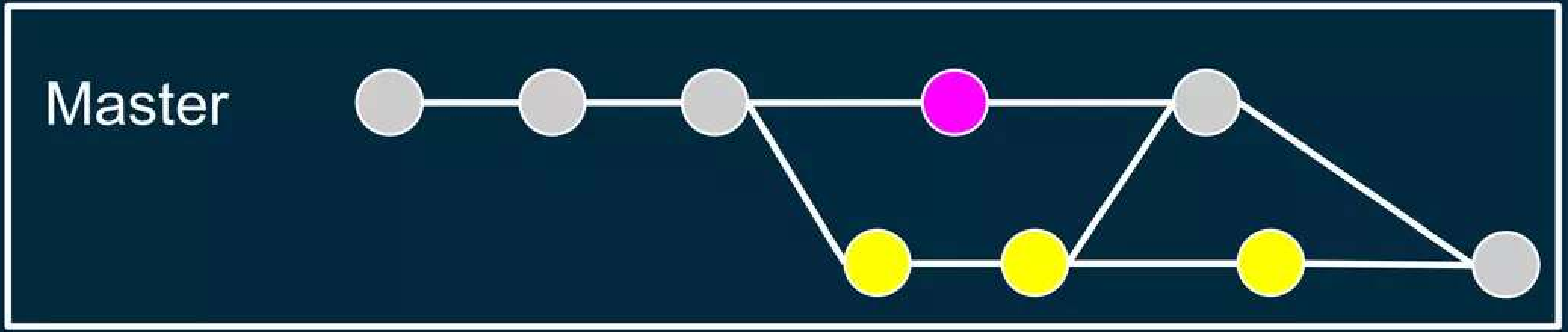


LAB SQUAD

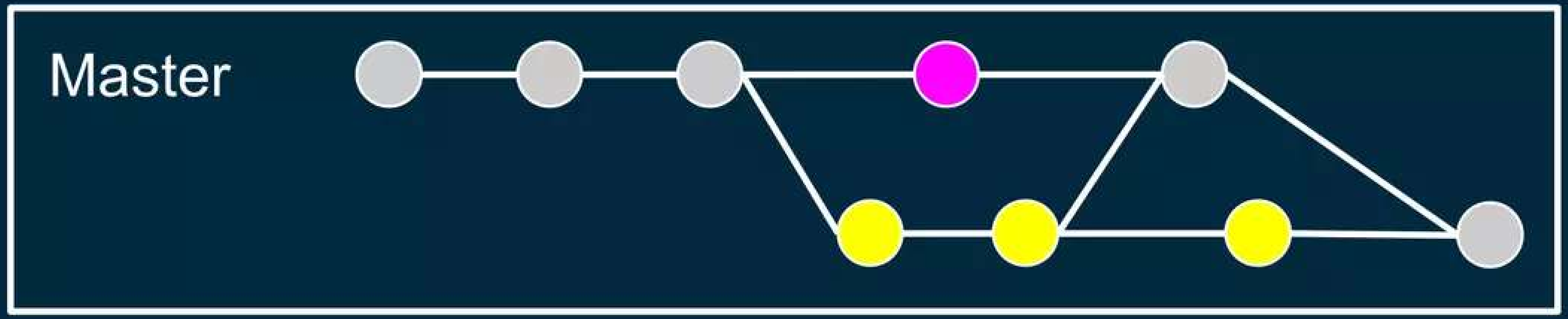
# Centralized Workflow (3)

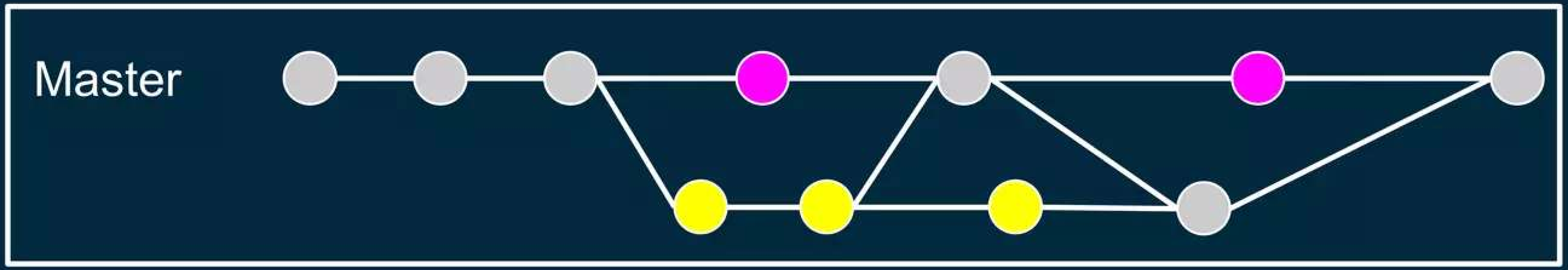






# Centralized Workflow (3)





LAB SQUAD

# Centralized Workflow (4)

## Advantages

. Simple with 1 dev

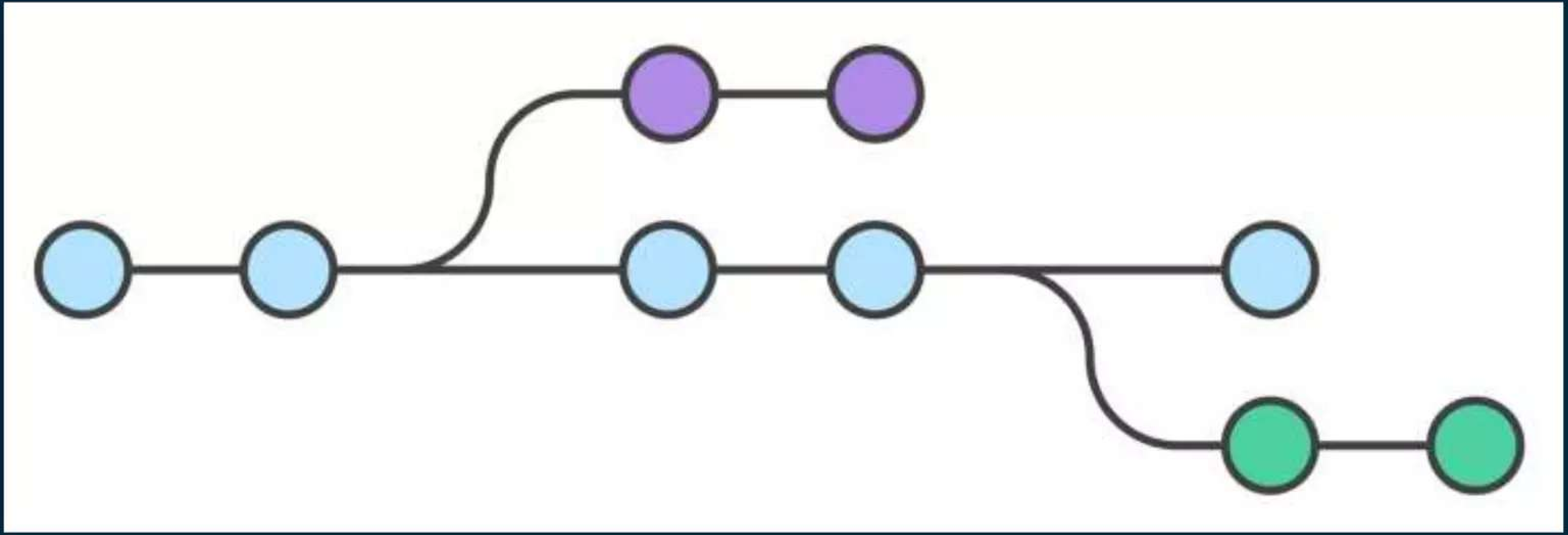
. Take less effort

## Disadvantages

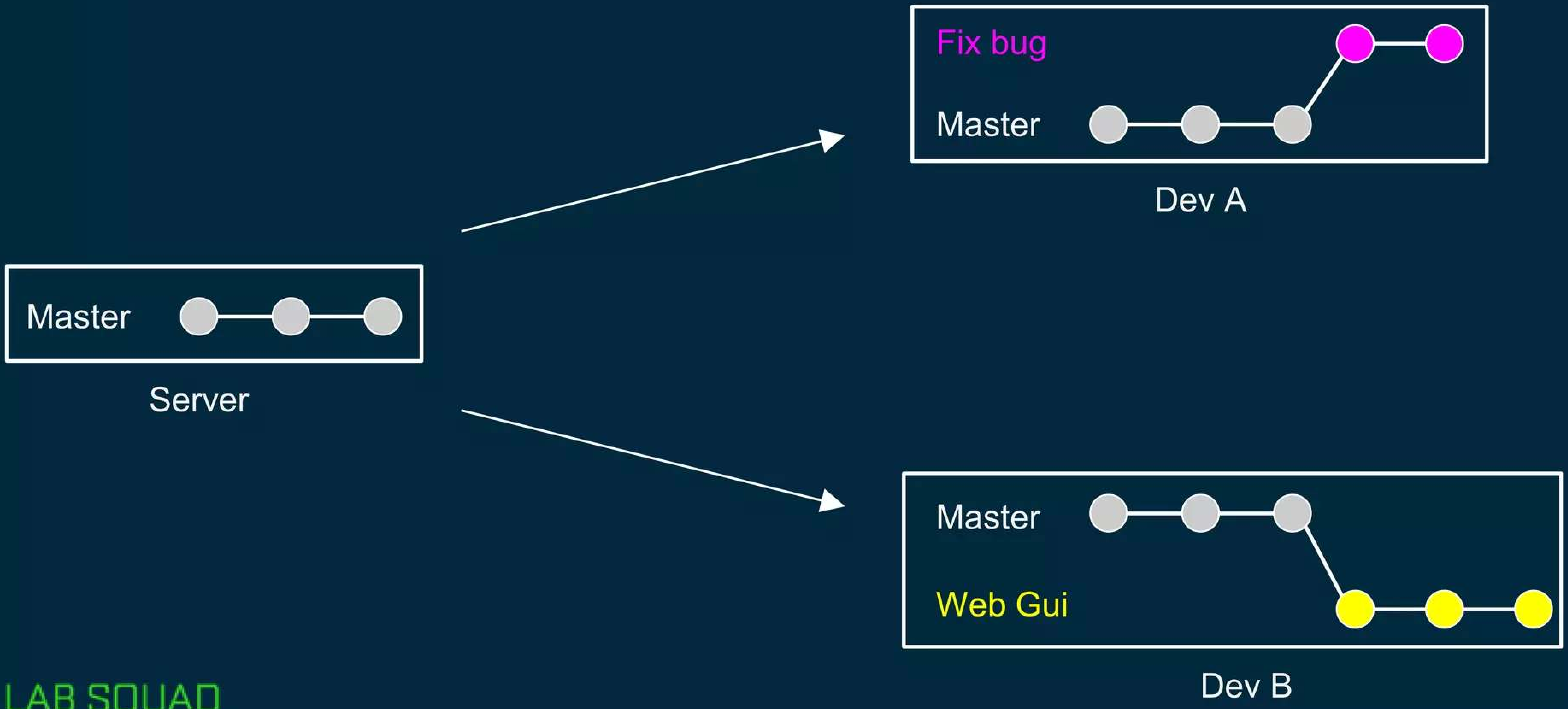
. Complex with > 2 devs

. A unstable commits will affect other devs

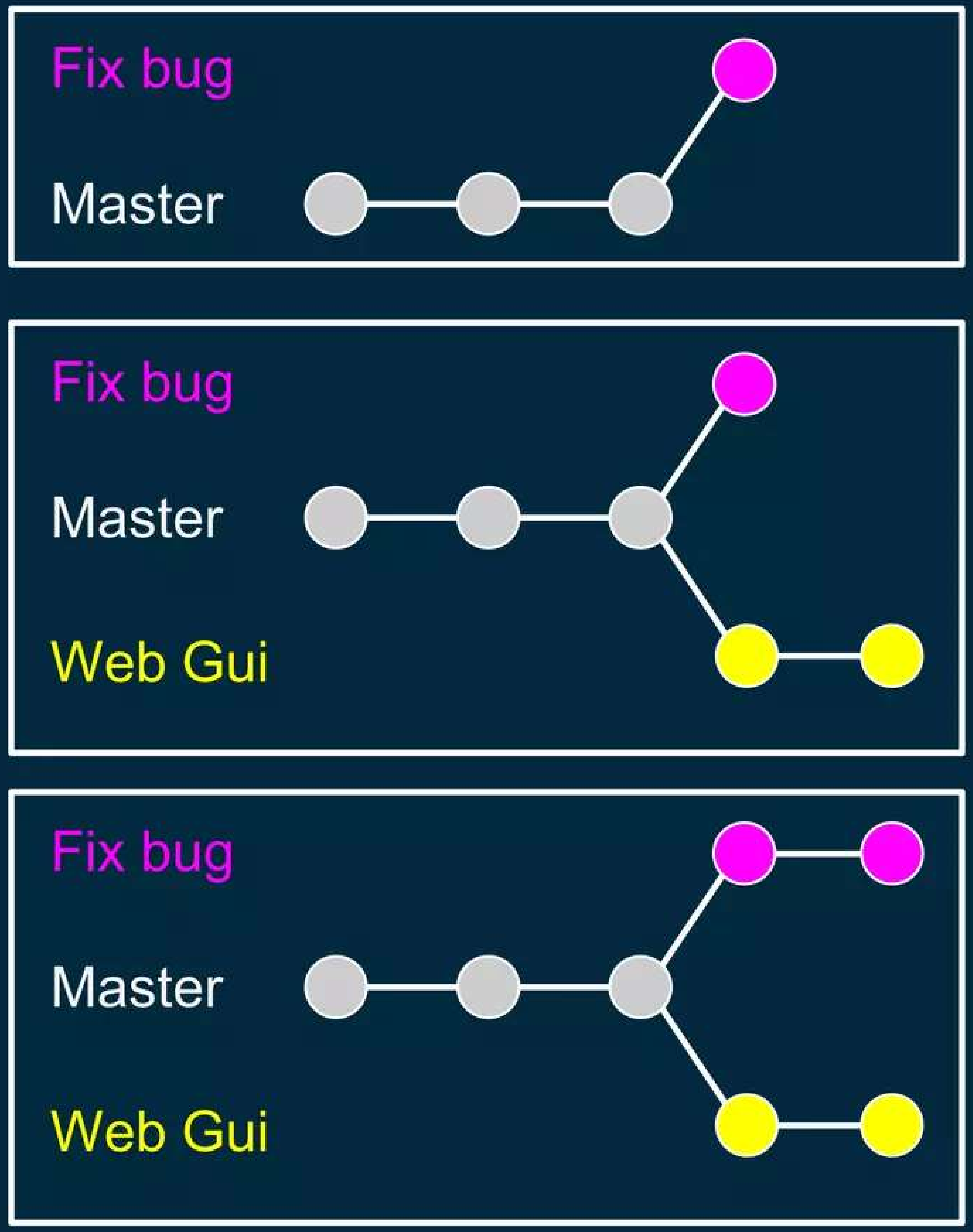
# Feature Branch Workflow (1)



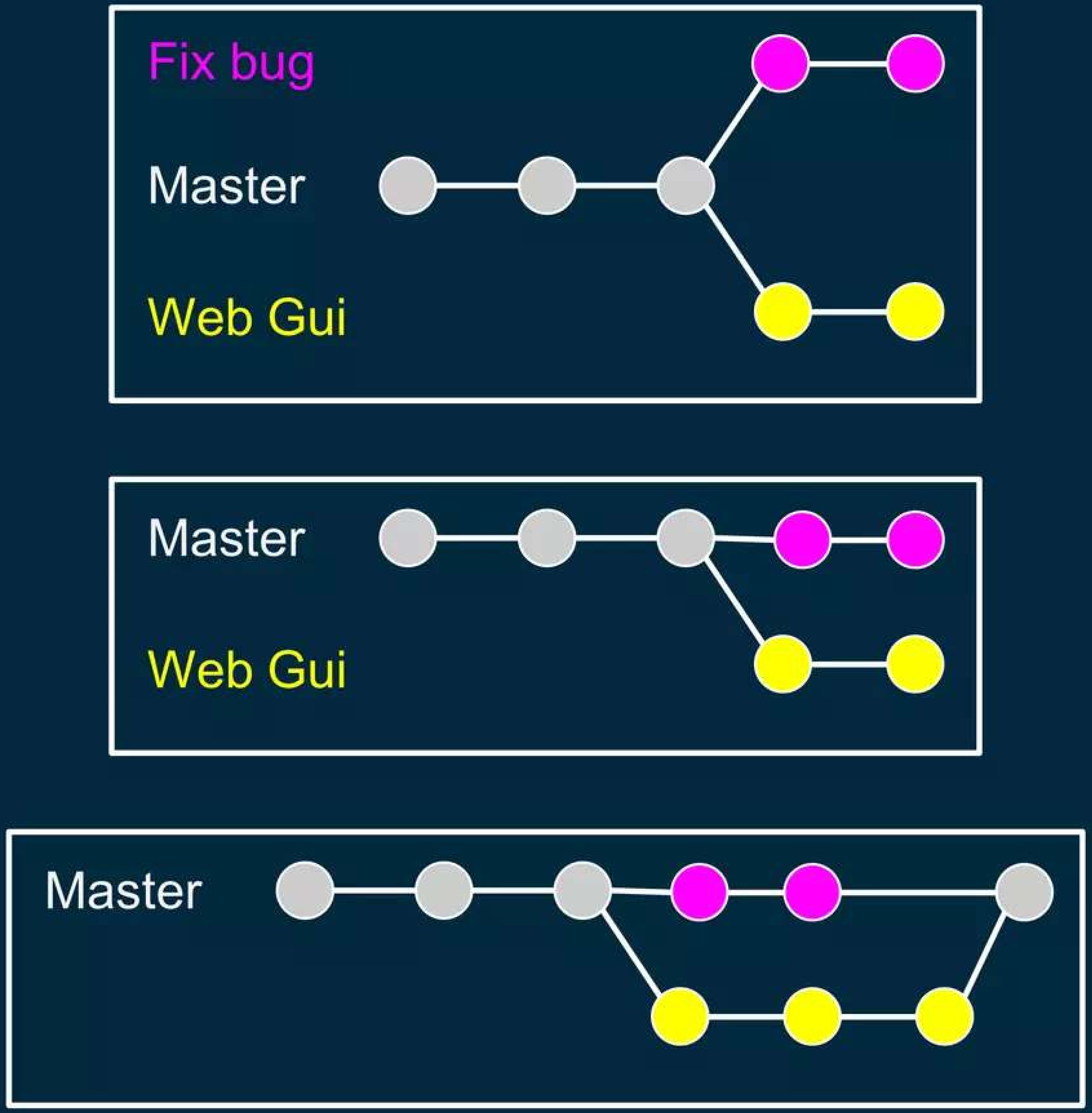
# Feature Branch Workflow (2)



# Feature Branch Workflow (3)



# Feature Branch Workflow (4)



# Feature Branch Workflow (4)

## Advantages

. Devs work together without merging frequently

. Features don’t affect together

## Disadvantages

. Not manage releasing flow yet

. Not support hotfix flow yet

. Master still unstable for production

# Git Workflows

Branding model

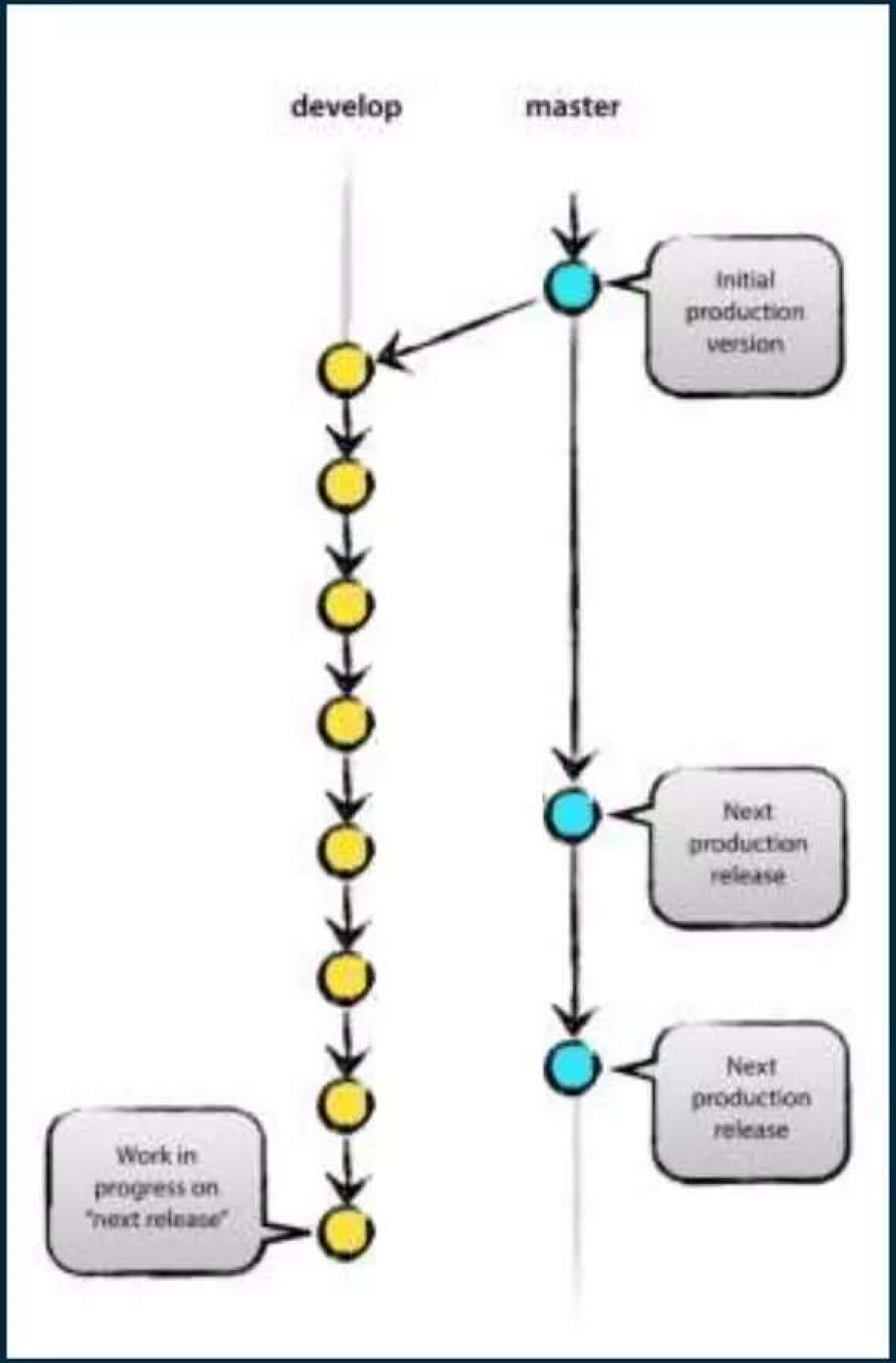
# Vincent Driessen

- January 05, 2010
- A post
- Standard for most git projects

# Two main branches

. master and develop

. infinite lifetime


develop
master
Initial production version
Next production release
Next production release
Work in progress on "next release"

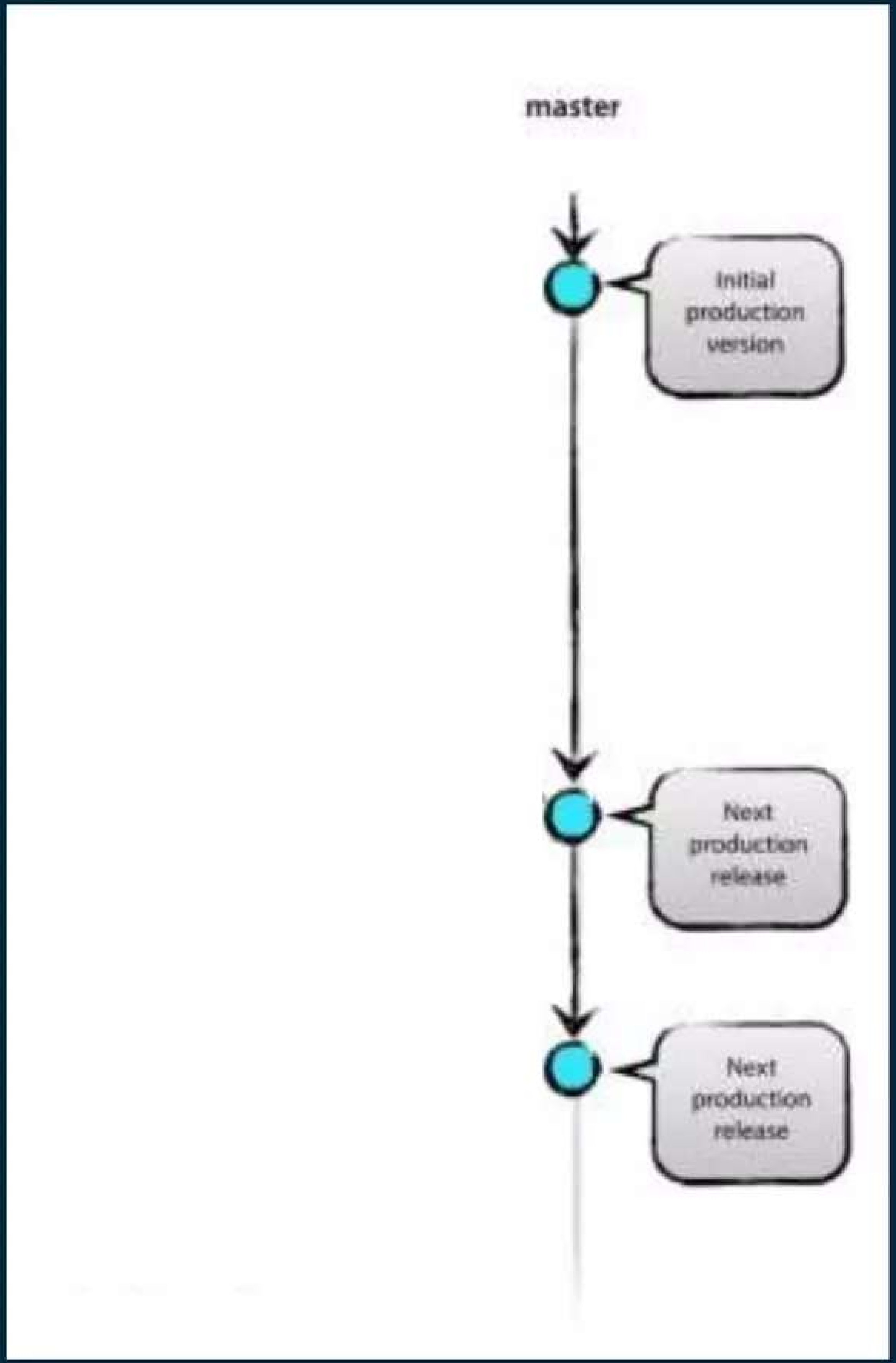
# Branch master

. Production release

. Stable

. Never implement new features in master branch

. Used for automation build production package



LAB SQUAD

# Branch develop

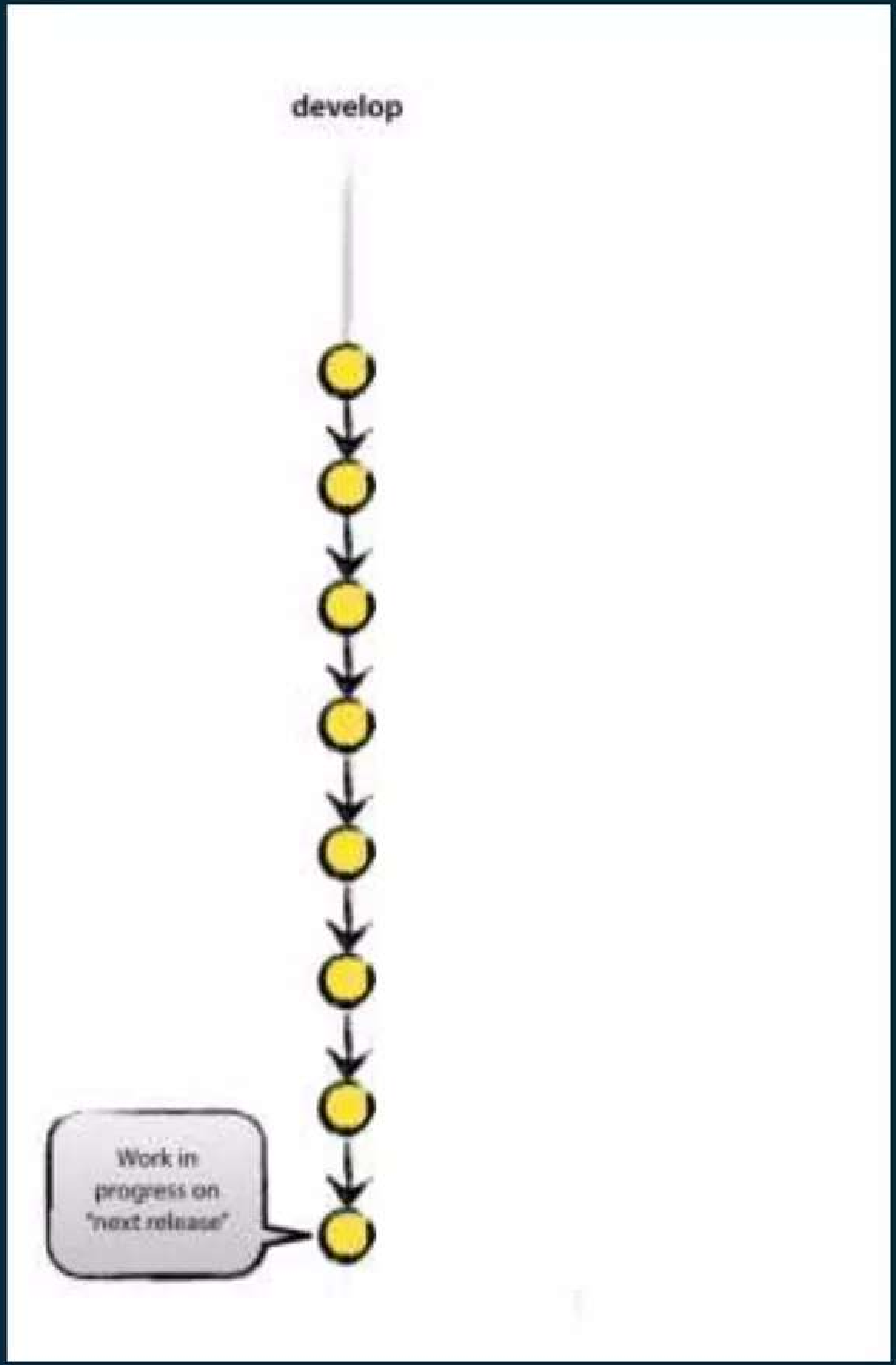
. Branch from master when init

. Features are ready

. Stable for integration test

. Used for build staging or nightly build

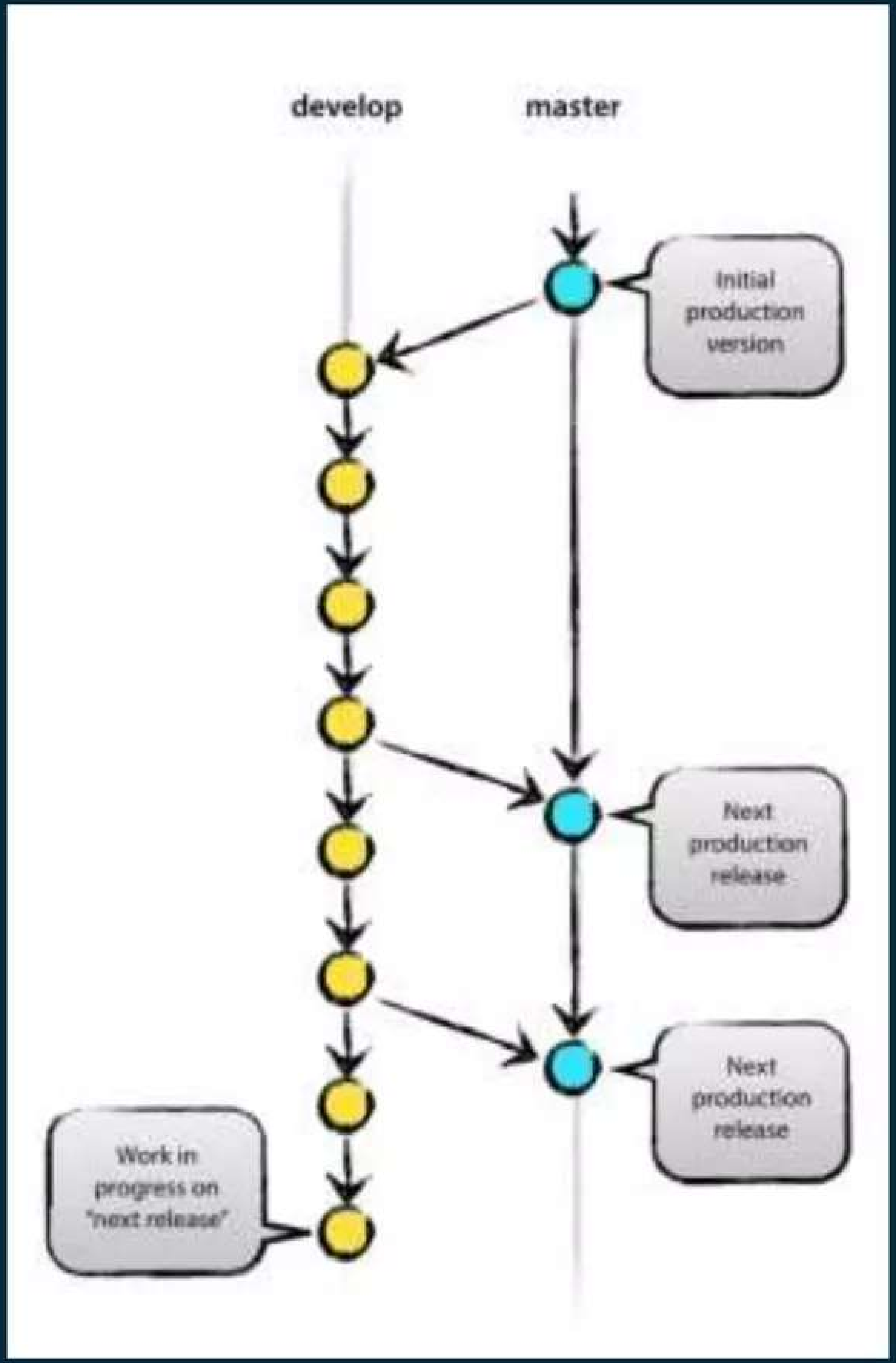
. Should not commit on develop directly



LAB SQUAD

# Master and develop

. When develop is enough stable for release:
-> Merge develop into master


develop
master
Initial production version
Next production release
Next production release
Work in progress on "next release"

# Supporting branches

. Feature branches

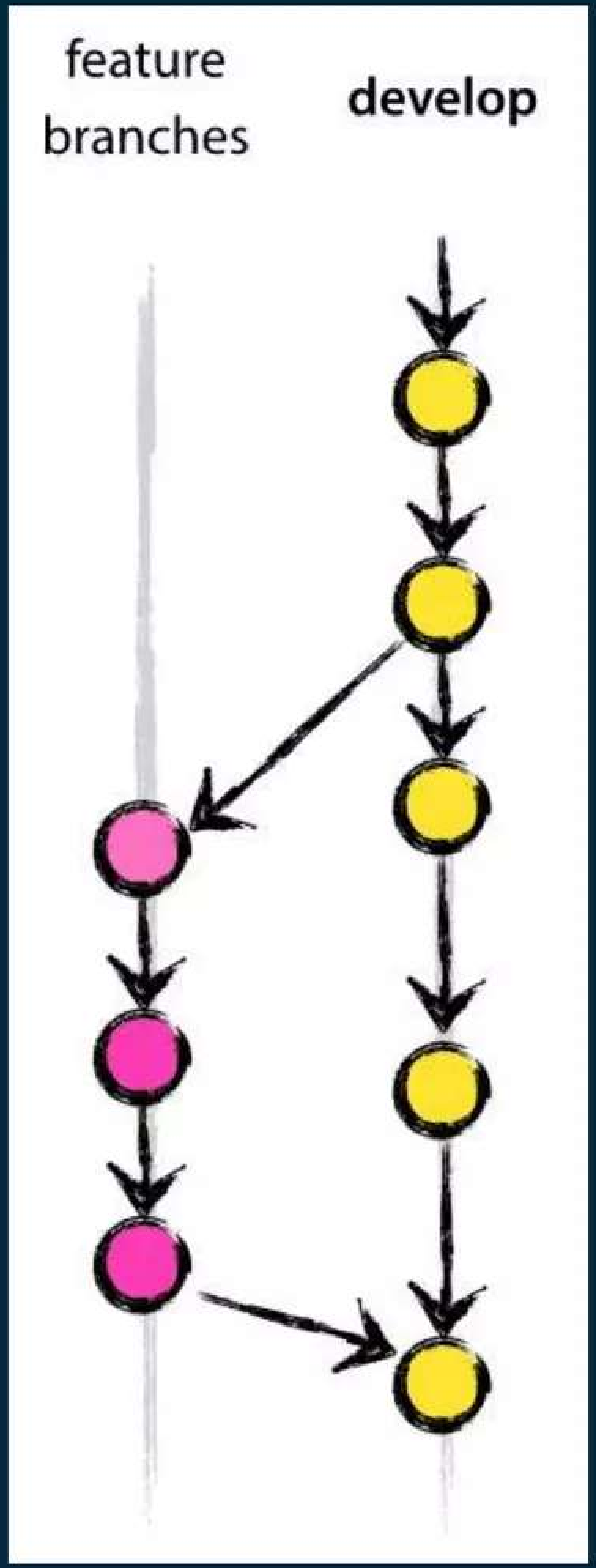
. Release branches

. Hotfix branches

. Have a limited life time

# Feature branches

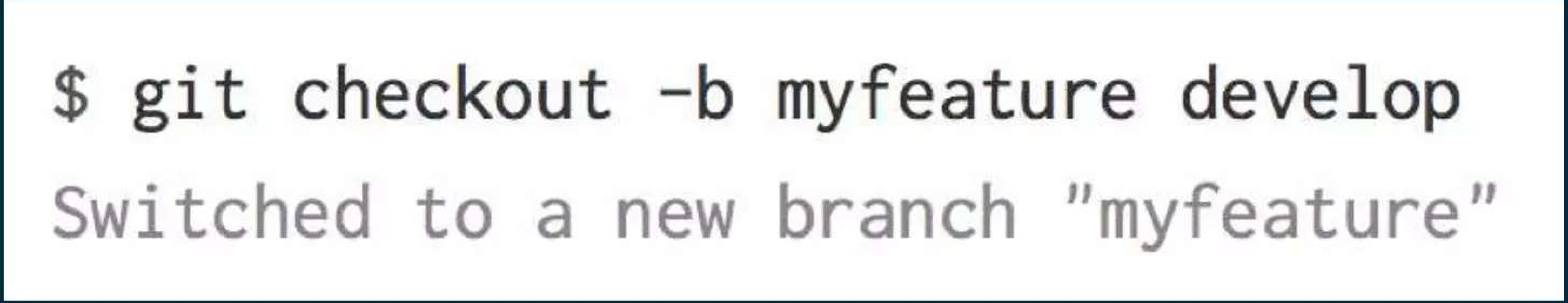
. 1 feature - 1 branch
. Branch from develop
. Merge into develop
. Only merge to develop when feature is done
. Delete when it was done



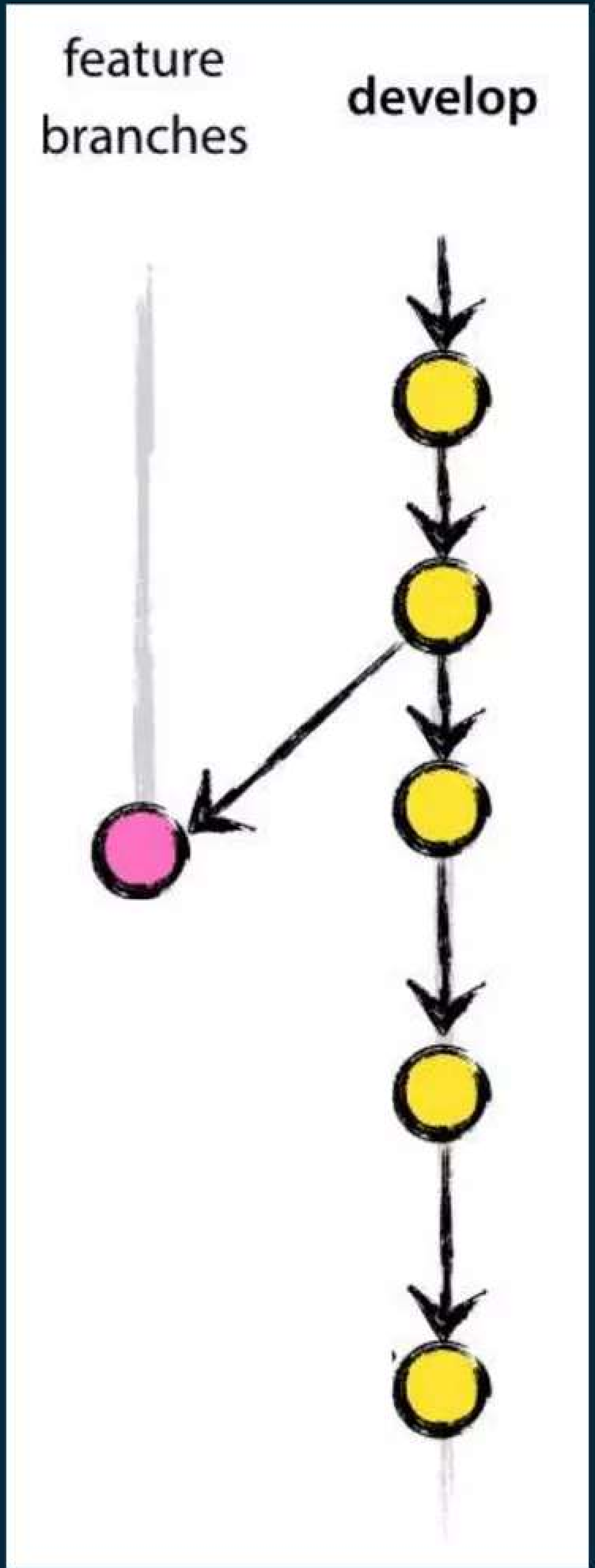
LAB SQUAD

# Feature branches

. Creating a feature branch

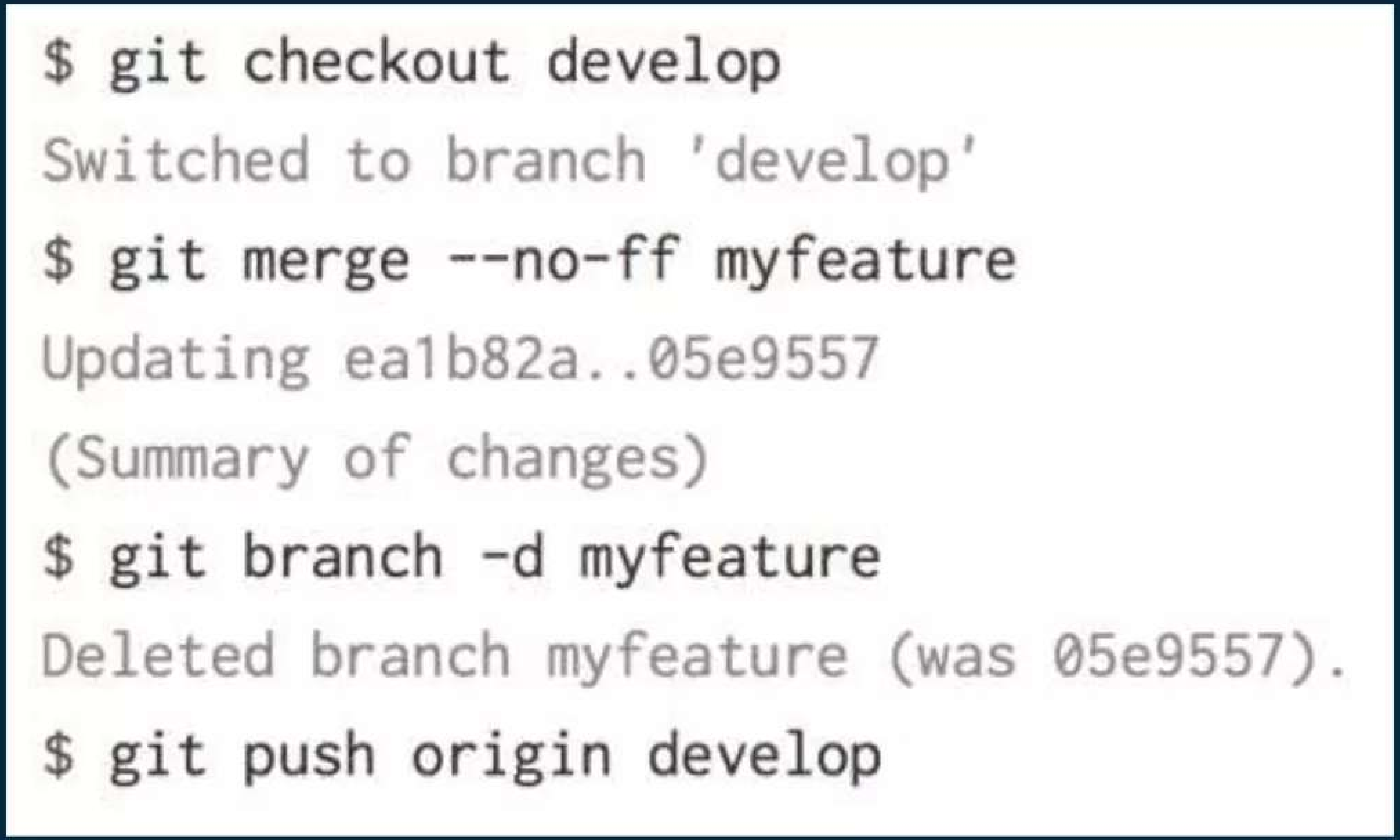


```
$ git checkout -b myfeature develop
Switched to a new branch "myfeature"
```

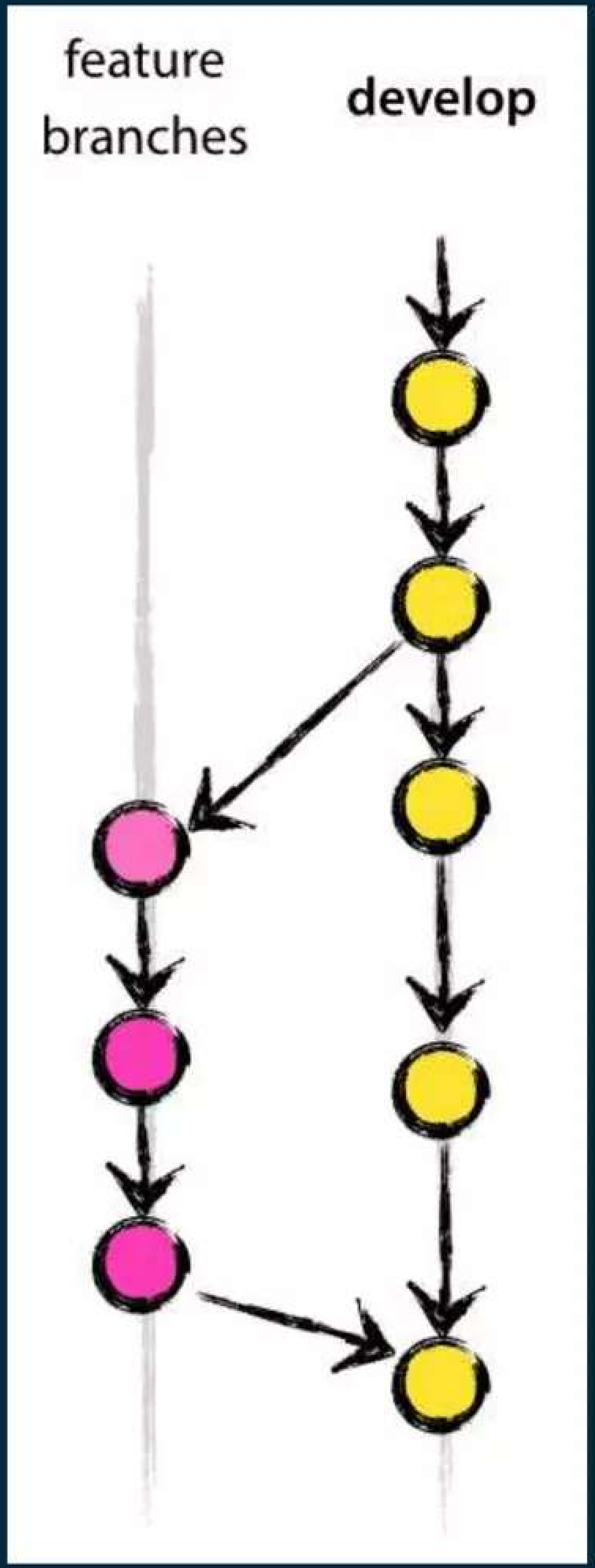


# Feature branches

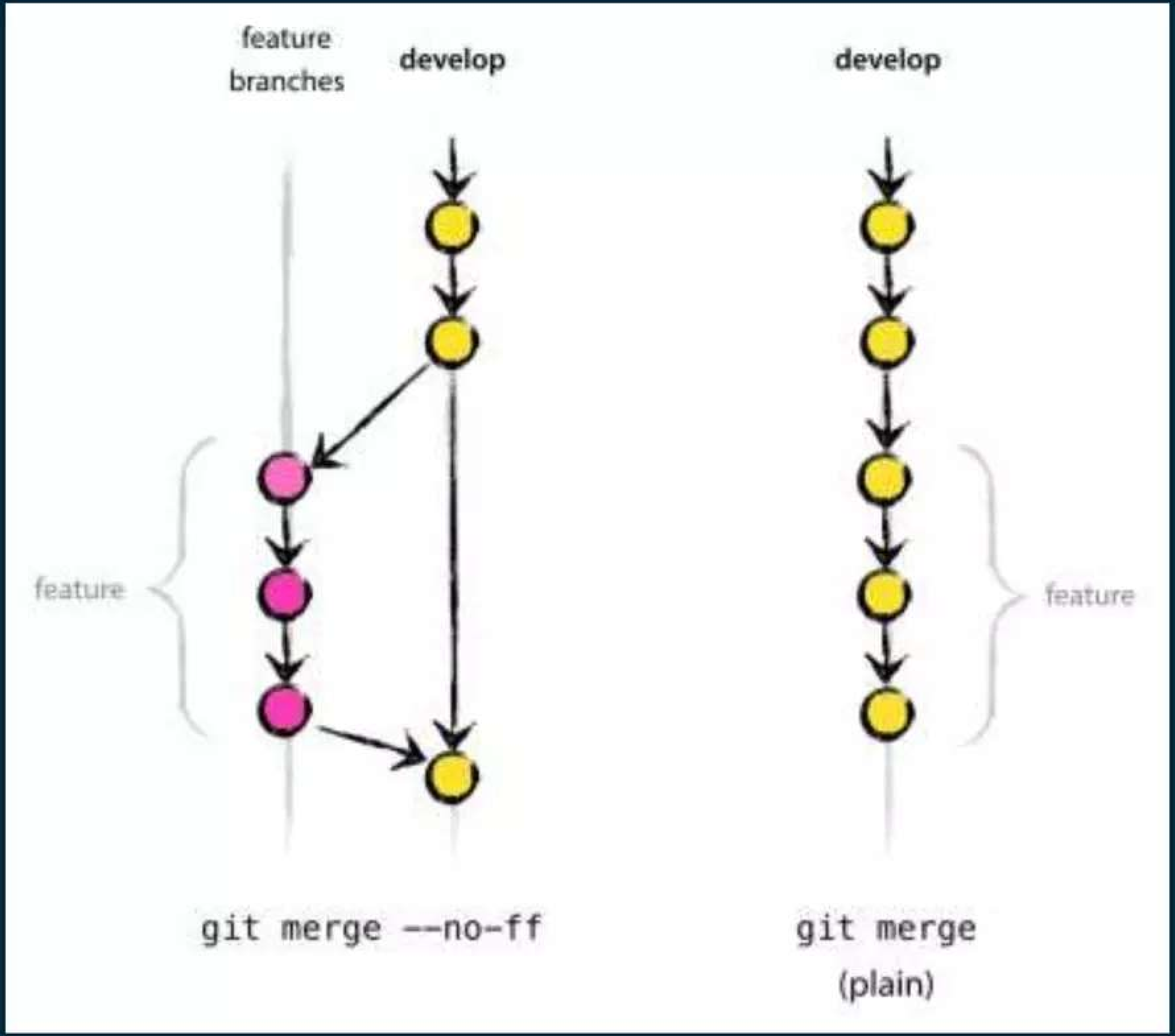
. Done feature and merge to develop



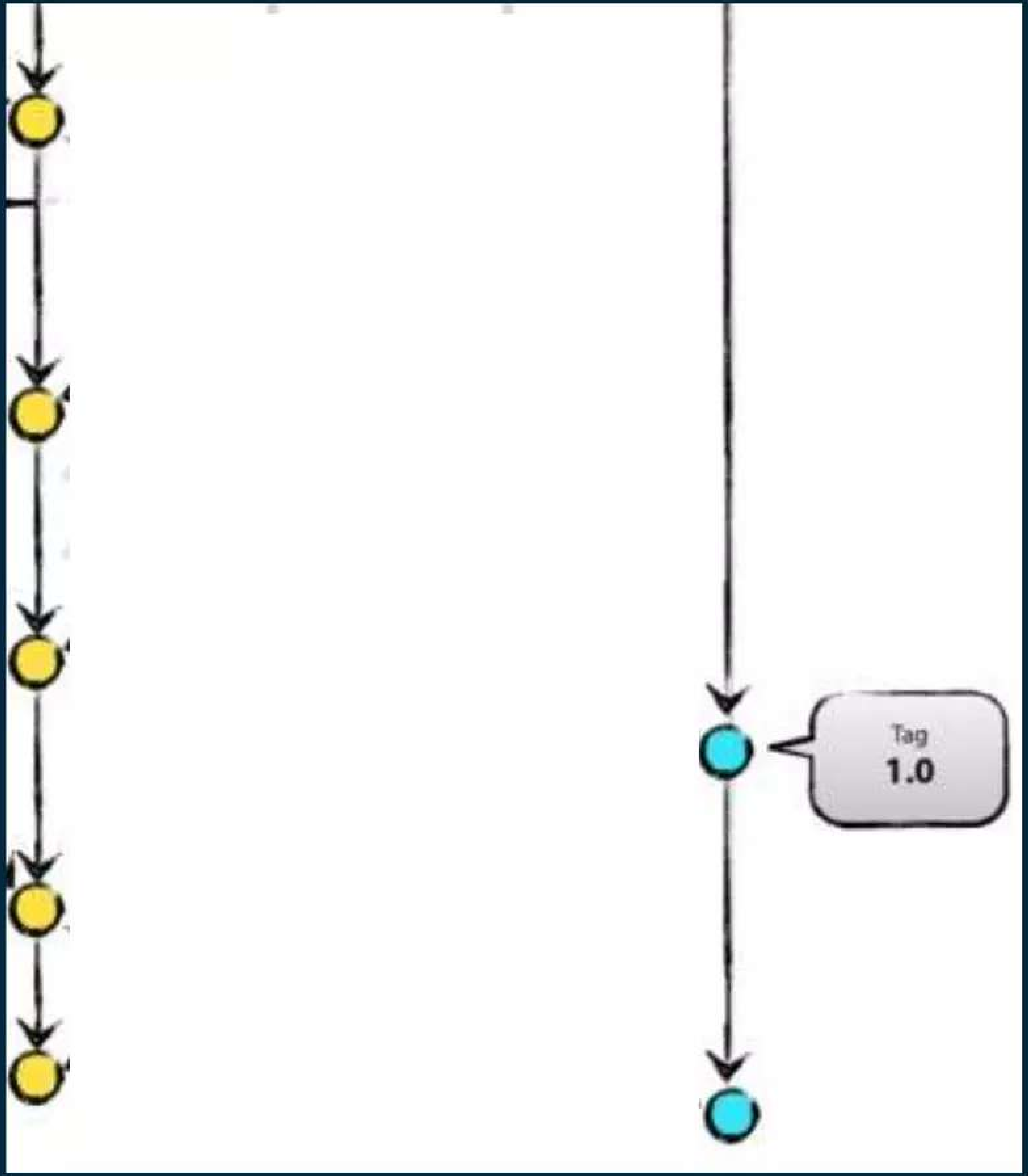
```
$ git checkout develop
Switched to branch 'develop'
$ git merge --no-ff myfeature
Updating ea1b82a..05e9557
(Summary of changes)
$ git branch -d myfeature
Deleted branch myfeature (was 05e9557).
$ git push origin develop
```



# Merge with --no-ff



# Release branches



LAB SQUAD

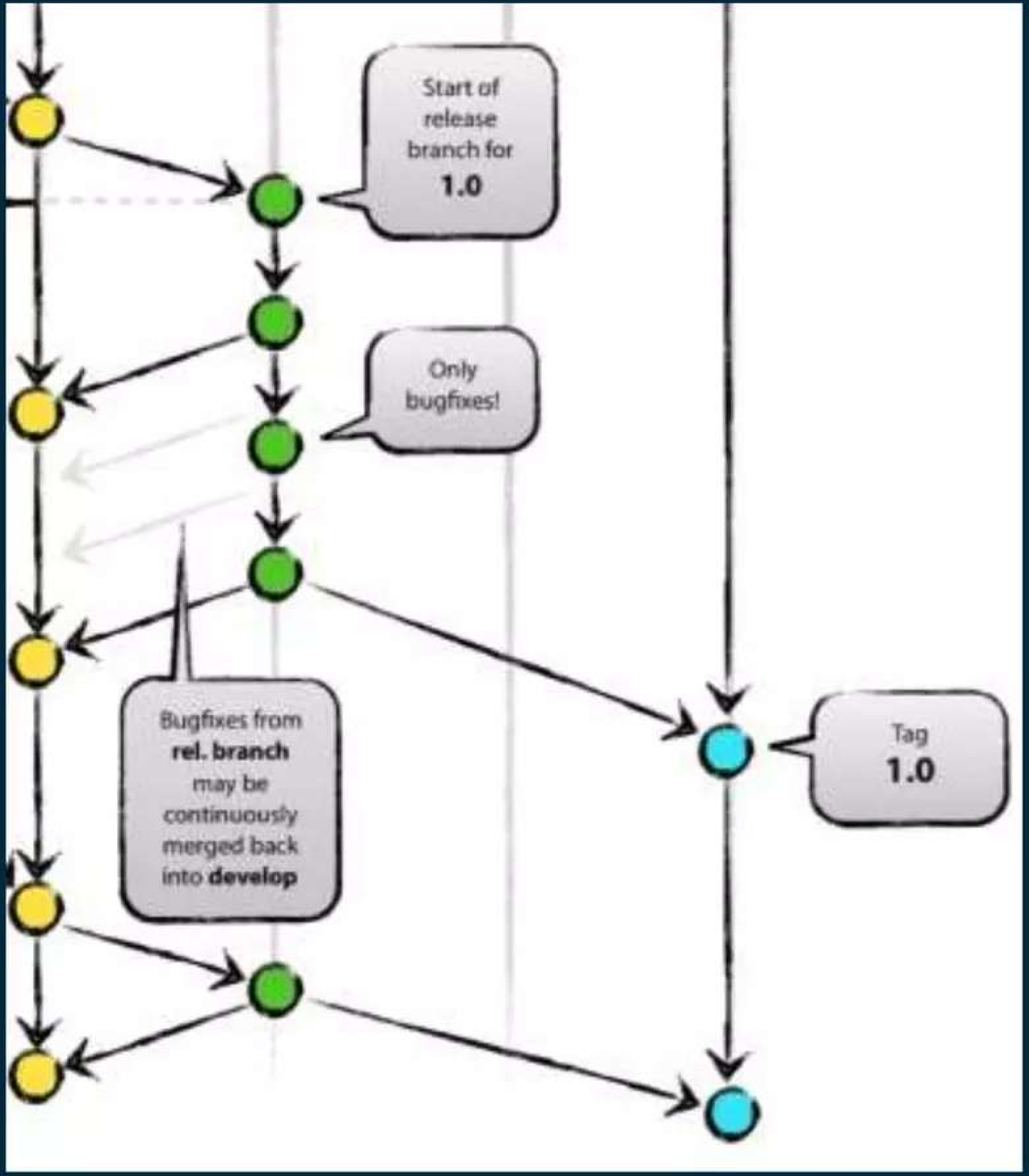
# Release branches

. A preparation for of a new production release.

. Branch from develop

. Merge into develop and master

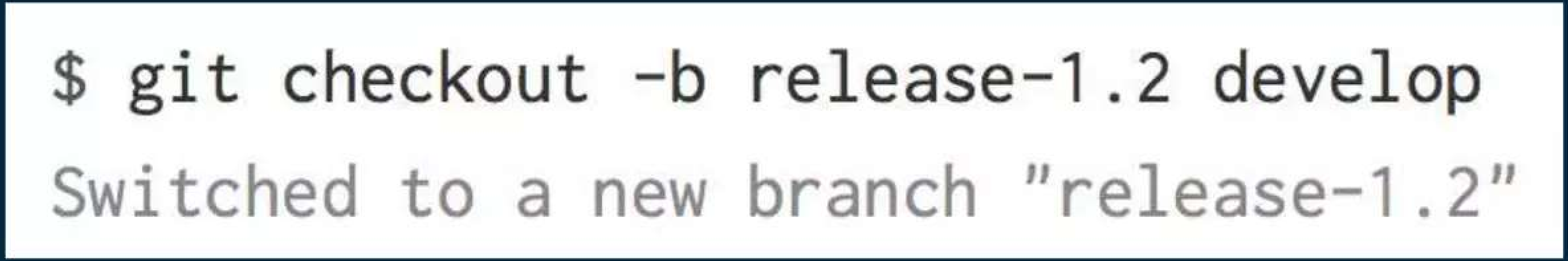
. Delete when it’s done



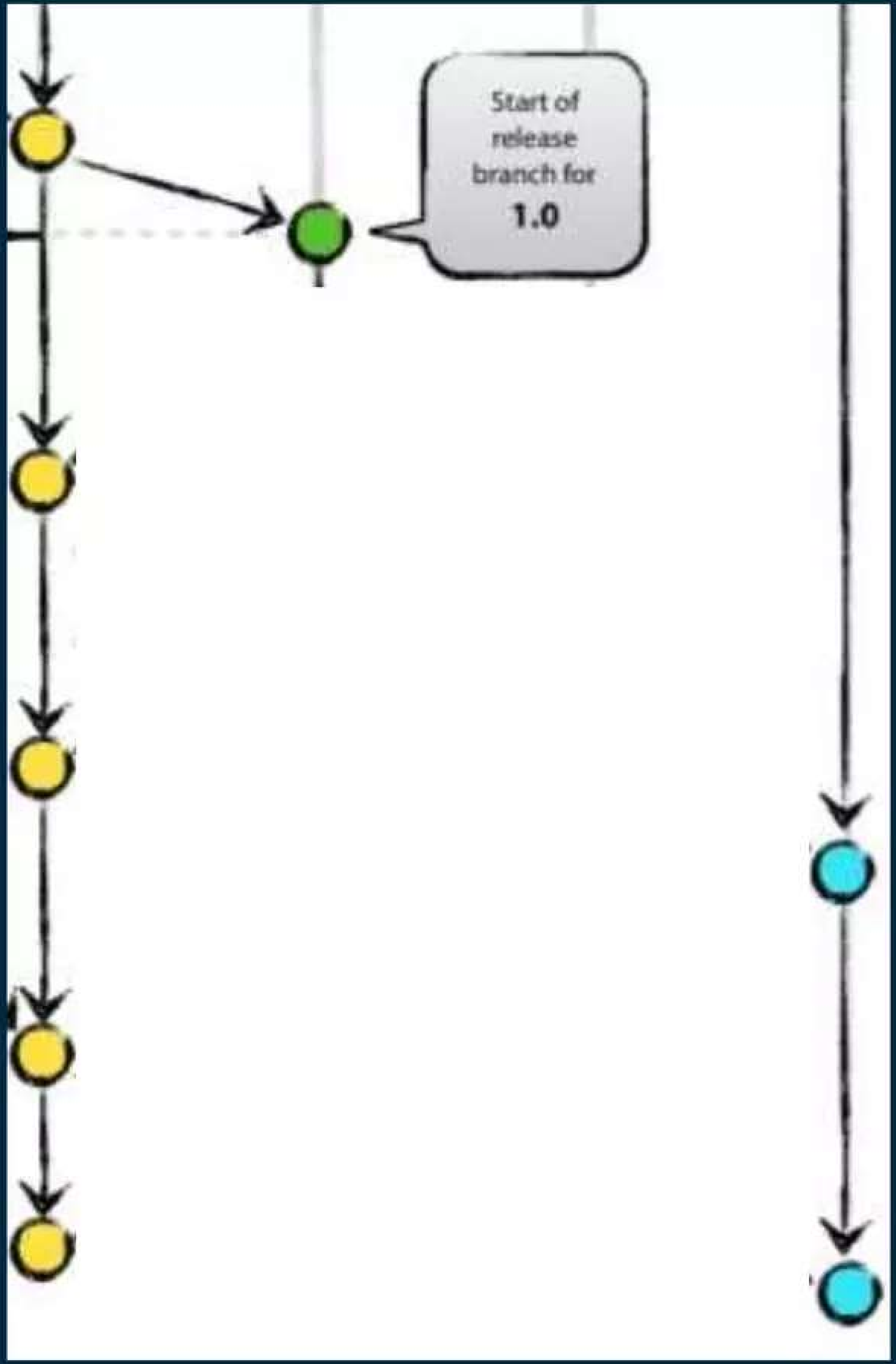
Start of release branch for 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

# Release branches

. Create release

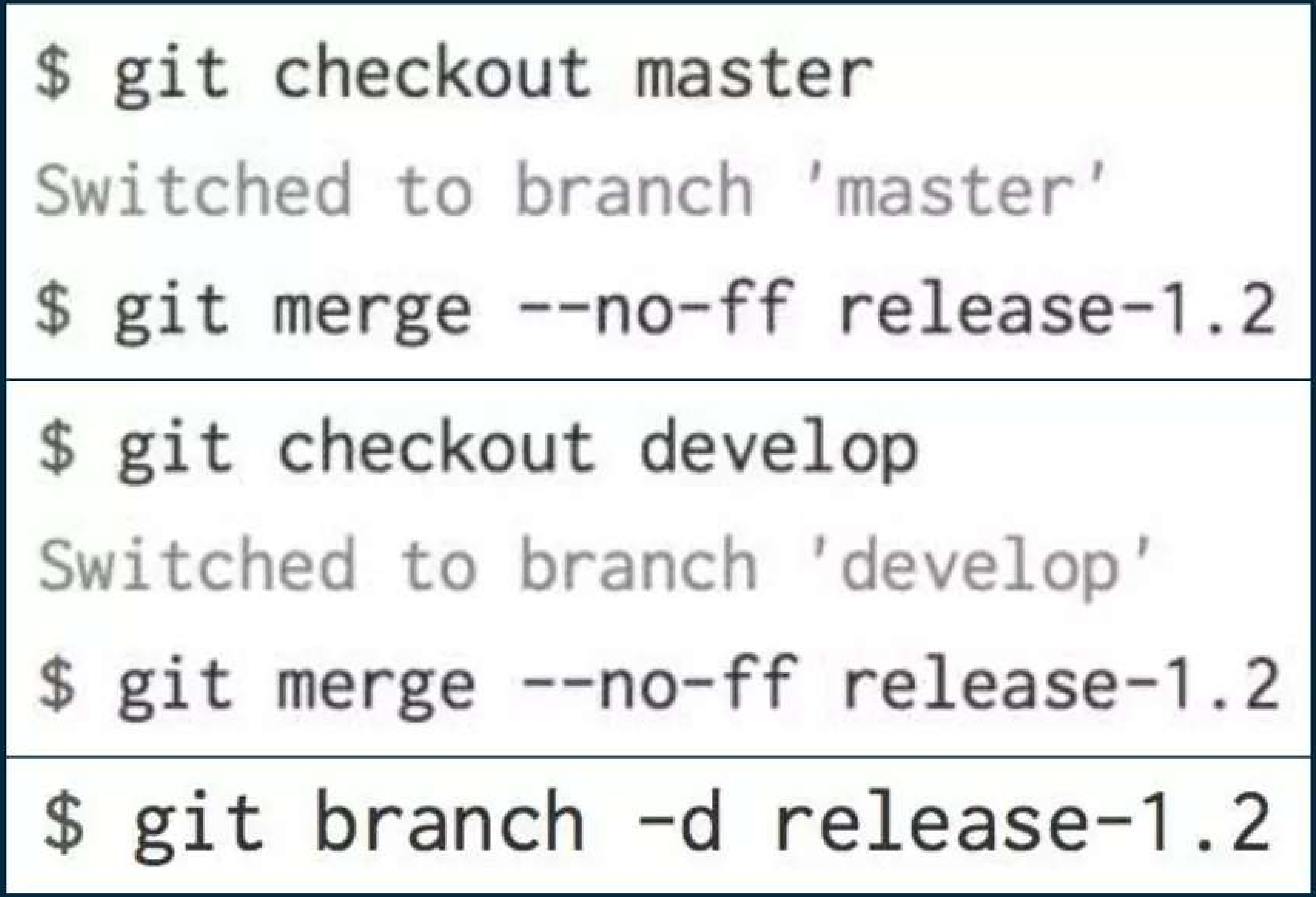


```
$ git checkout -b release-1.2 develop
Switched to a new branch "release-1.2"
```



# Release branches

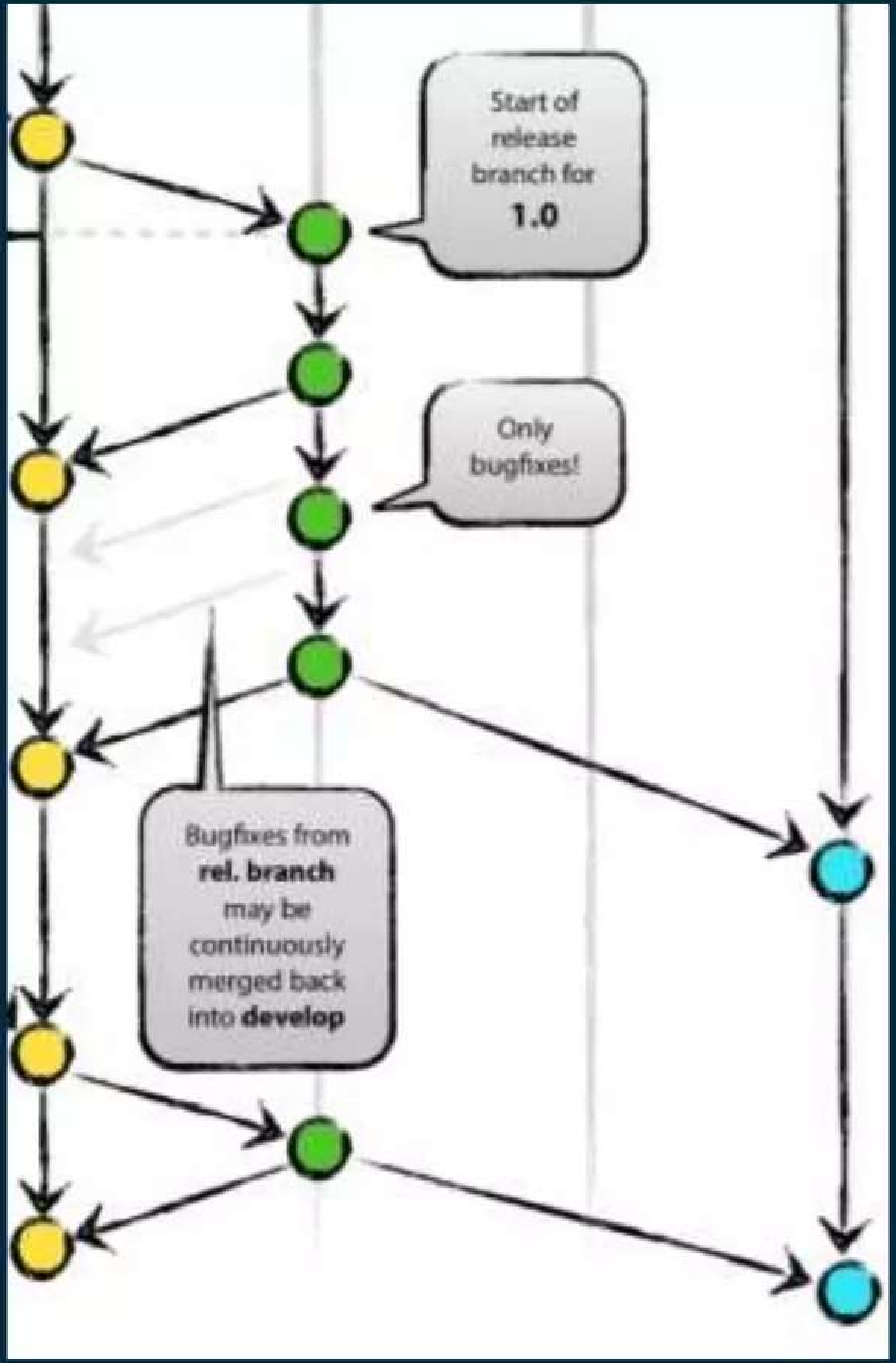
. Finish release branch



```
$ git checkout master
Switched to branch 'master'
$ git merge --no-ff release-1.2
```

```
$ git checkout develop
Switched to branch 'develop'
$ git merge --no-ff release-1.2
```

```
$ git branch -d release-1.2
```



Start of release branch for 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop

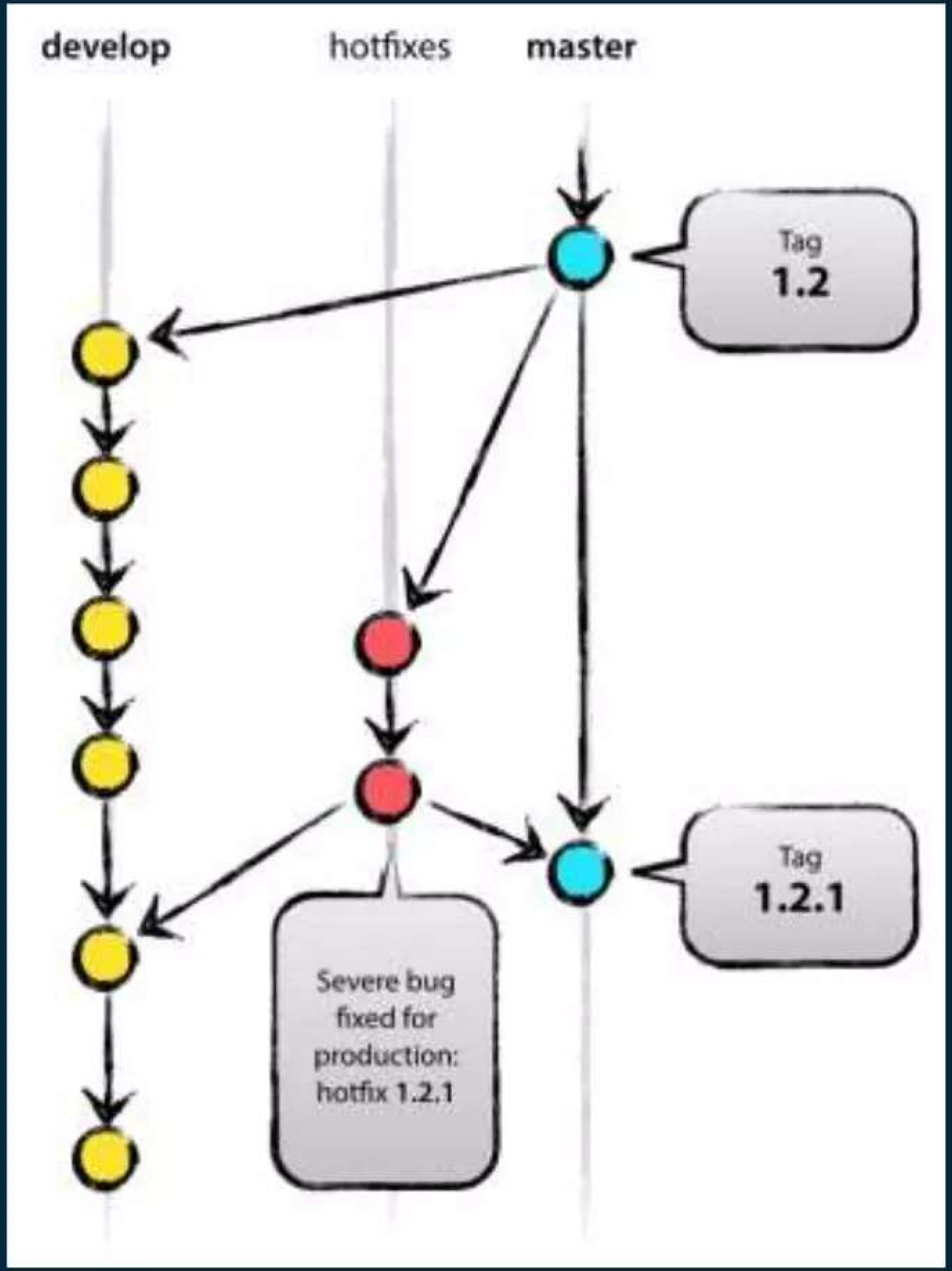
# Hotfix branches

. Use to fix issue in production

. Branch from master

. Merge into develop and master

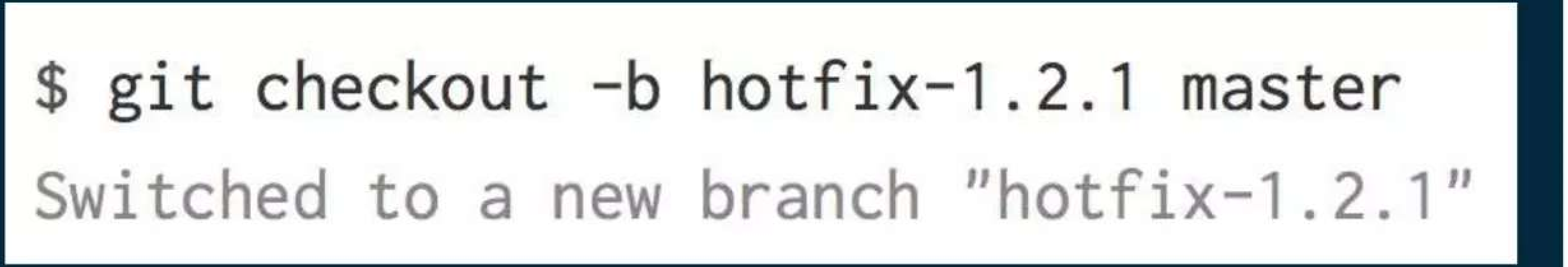
. Delete when it’s done



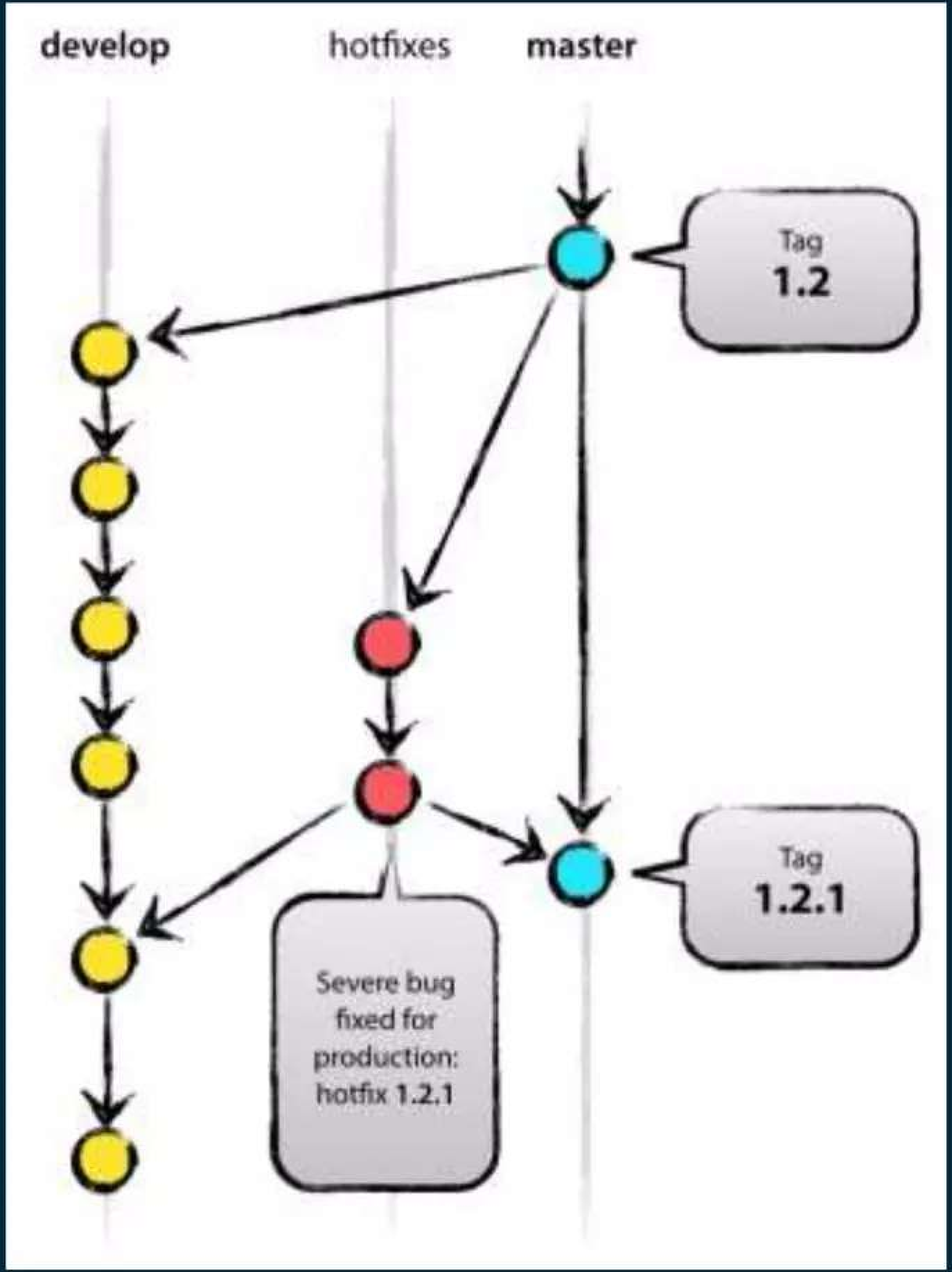
LAB SQUAD

# Hotfix branches

. Create hotfix branch

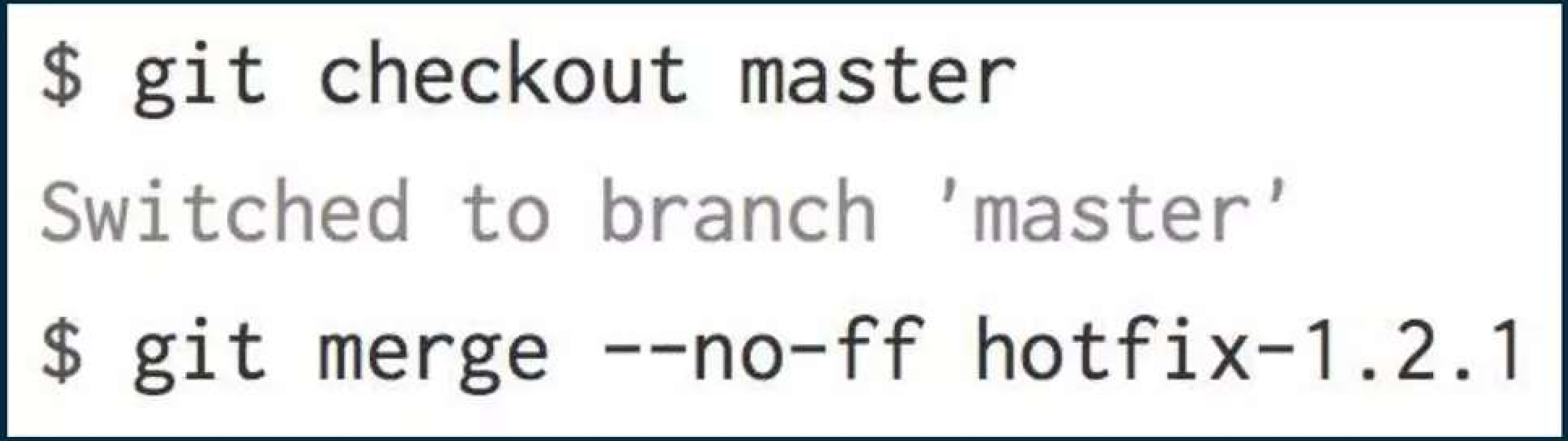


```
$ git checkout -b hotfix-1.2.1 master
Switched to a new branch "hotfix-1.2.1"
```

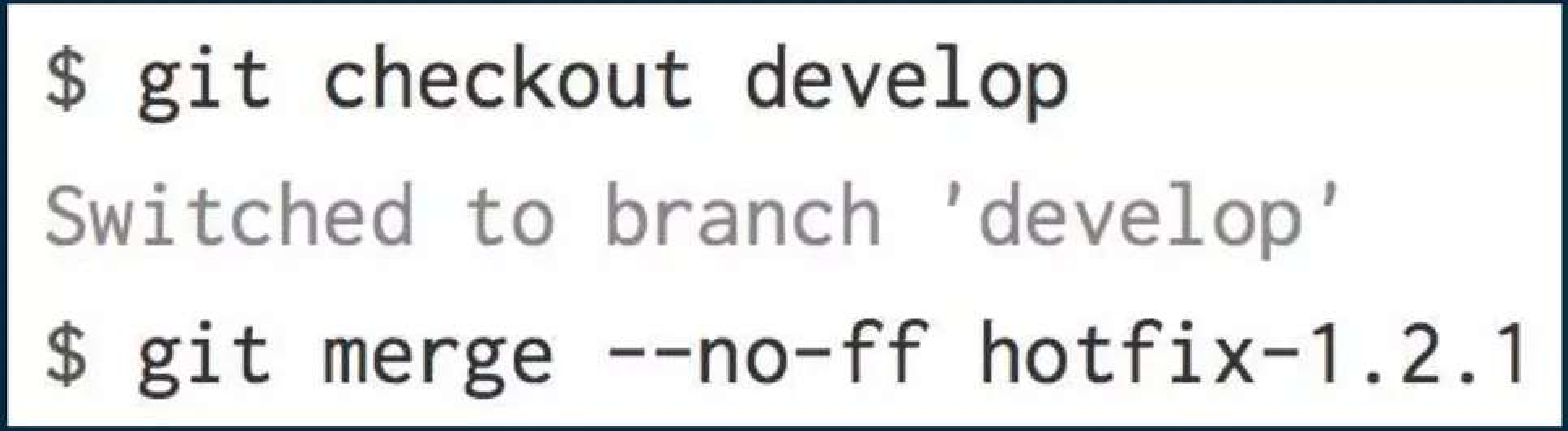


# Hotfix branches

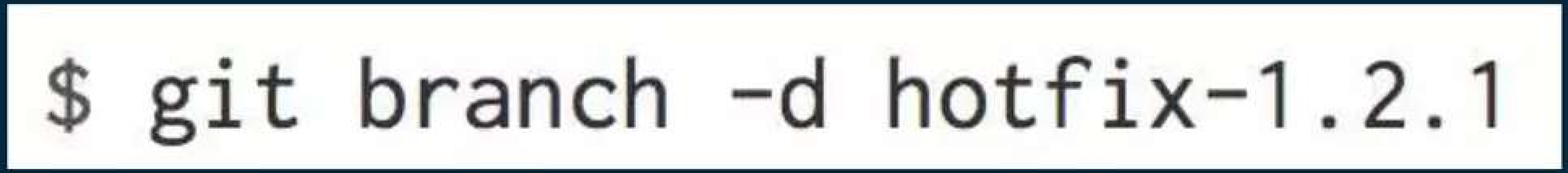
. Finish hotfix branch



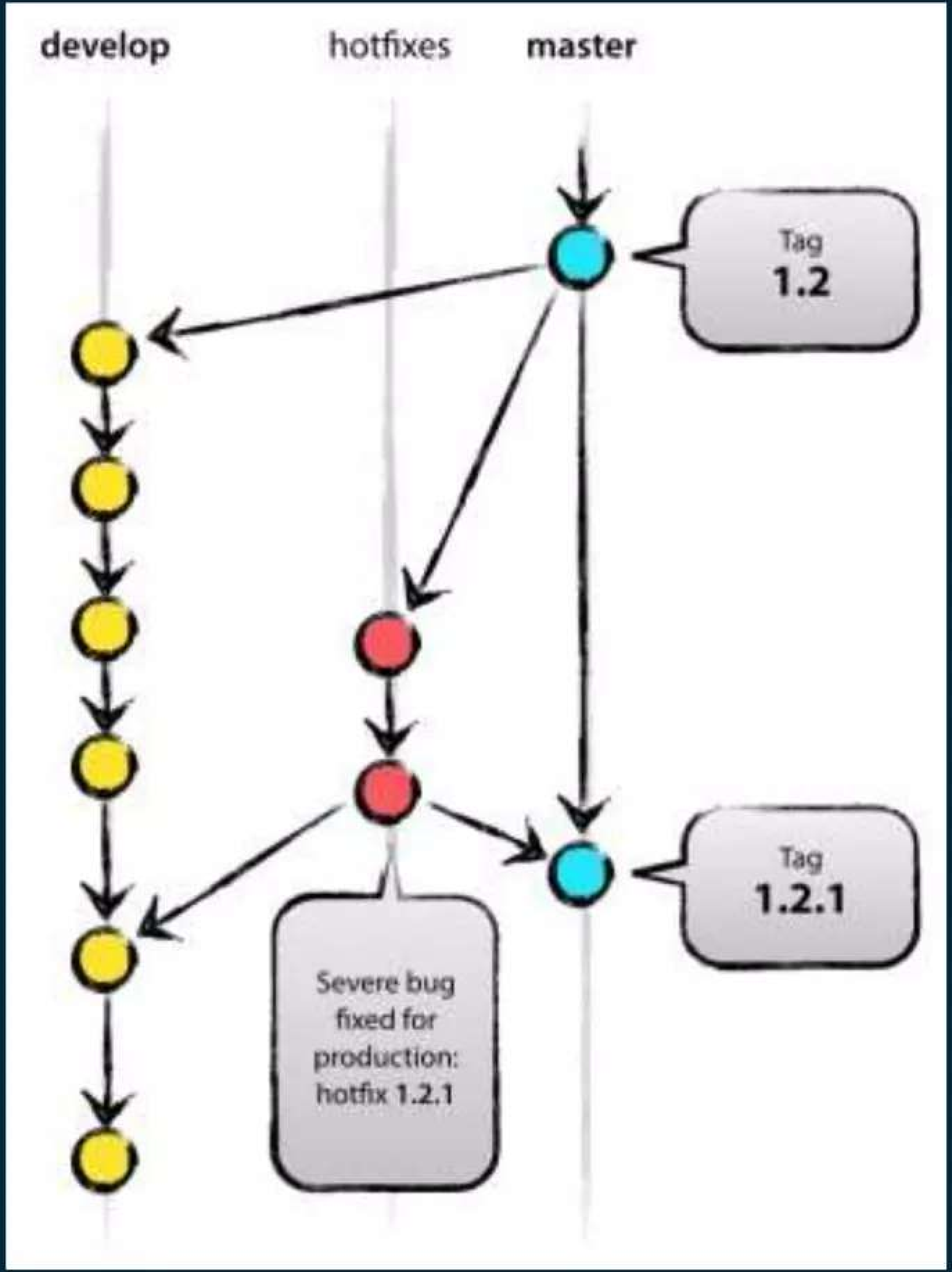
```
$ git checkout master
Switched to branch 'master'
$ git merge --no-ff hotfix-1.2.1
```



```
$ git checkout develop
Switched to branch 'develop'
$ git merge --no-ff hotfix-1.2.1
```



```
$ git branch -d hotfix-1.2.1
```



# Summary

. Two main branches: master, develop
. Three supporting branches: feature, release, hotfix

Fullflow:
http://nvie.com/files/Git-branching-model.pdf

# Special cases

Experience when applying

# Exception cases

1 Commit in master...
2 Feature branch A needs commits from develop
3 Feature branch A needs commits from feature branch B
4 Multiple environment: production, staging, testing, xxx