# Advanced Git: Multiple Branches and Servers

Paul Grayson

2013-08-14

## Simple Git

- One server: “origin”
- One branch: “master”

## Demo: make a commit and push

## Goal

Use Git with multiple branches and servers, and know what is going on.

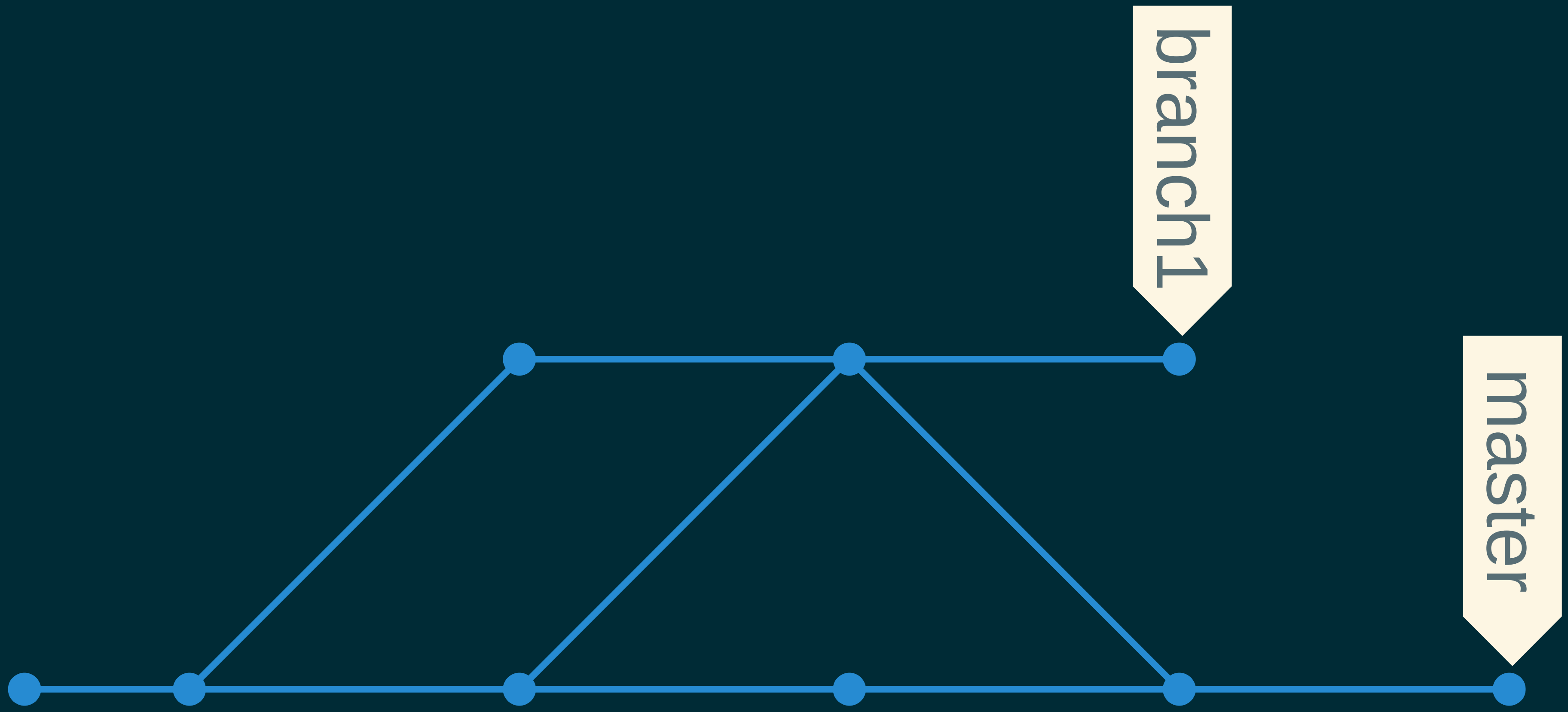
## What is a *branch*?

A label for a state of the repository -

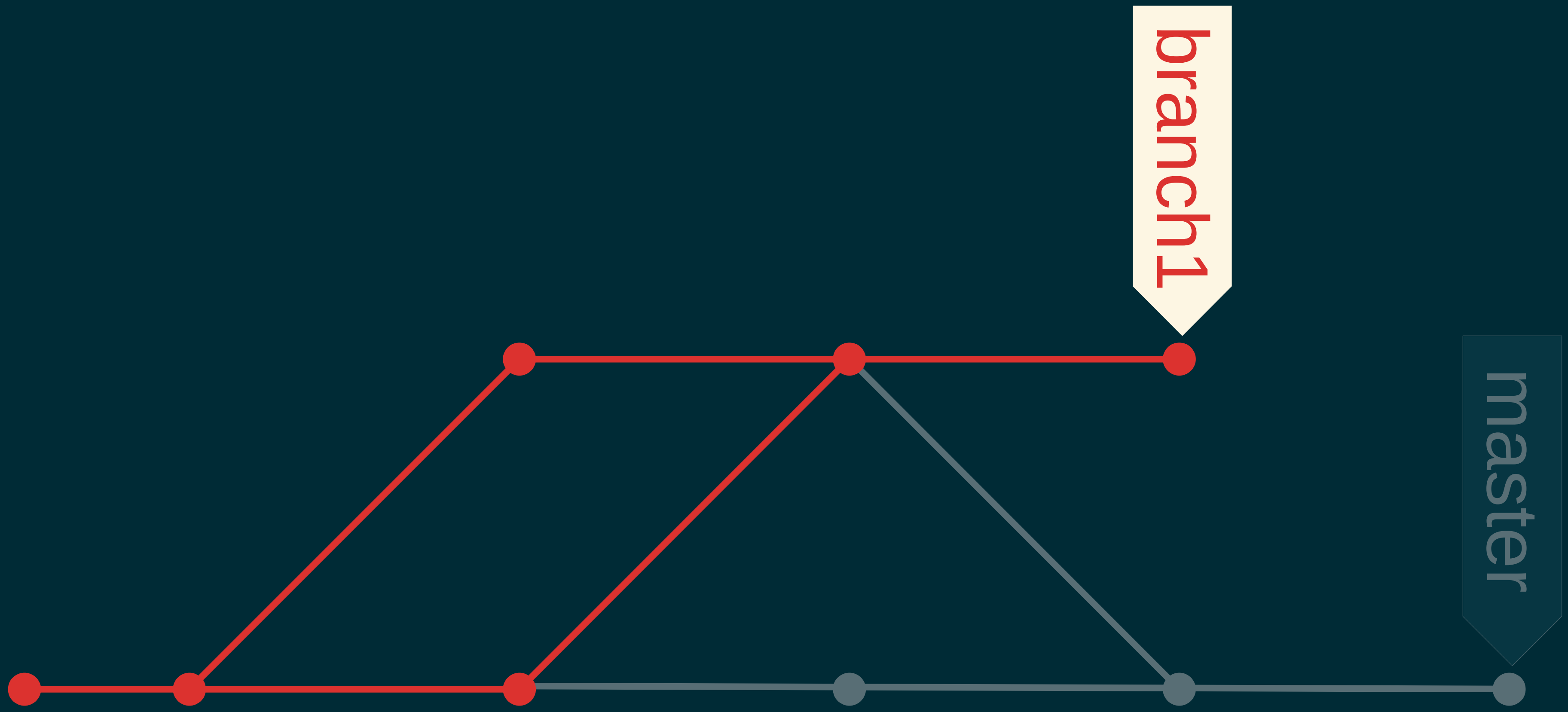
Current files (the commit) and all history.

- List: `git branch [-a]`
- Create: `git branch <name>`
- Read: `git show-branch, git log`
- Update: `git commit`
- Delete: `git branch -d <name>`

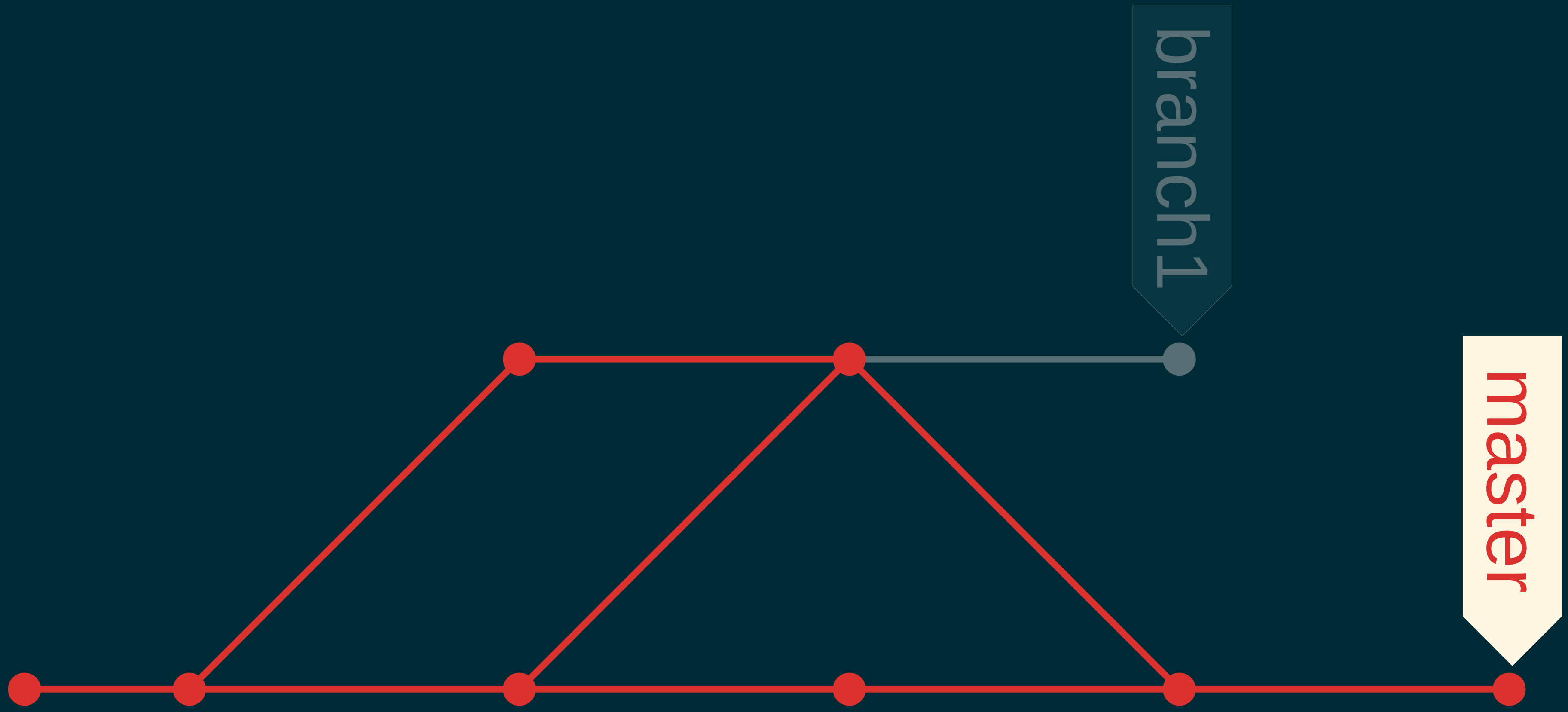
# Structure of Git history



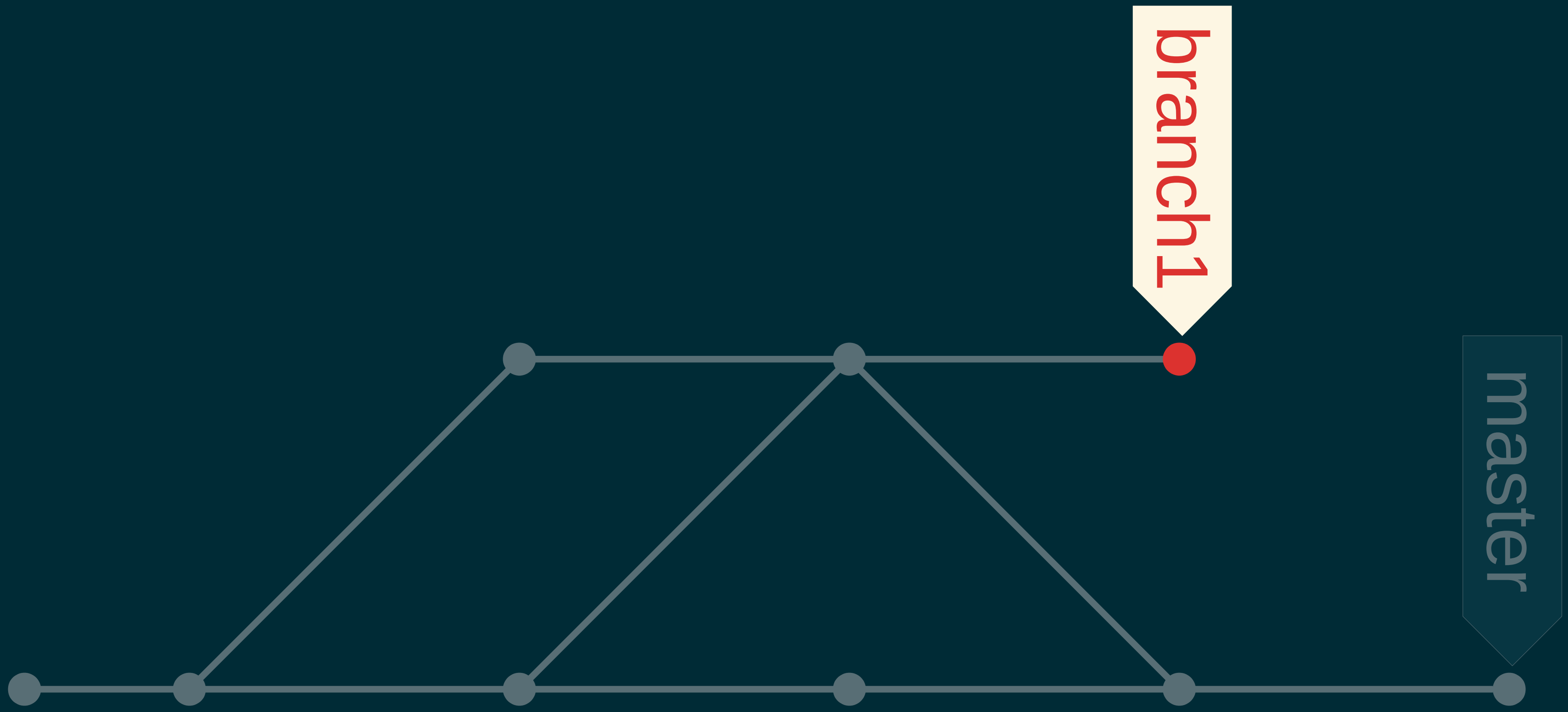
# History of branch1



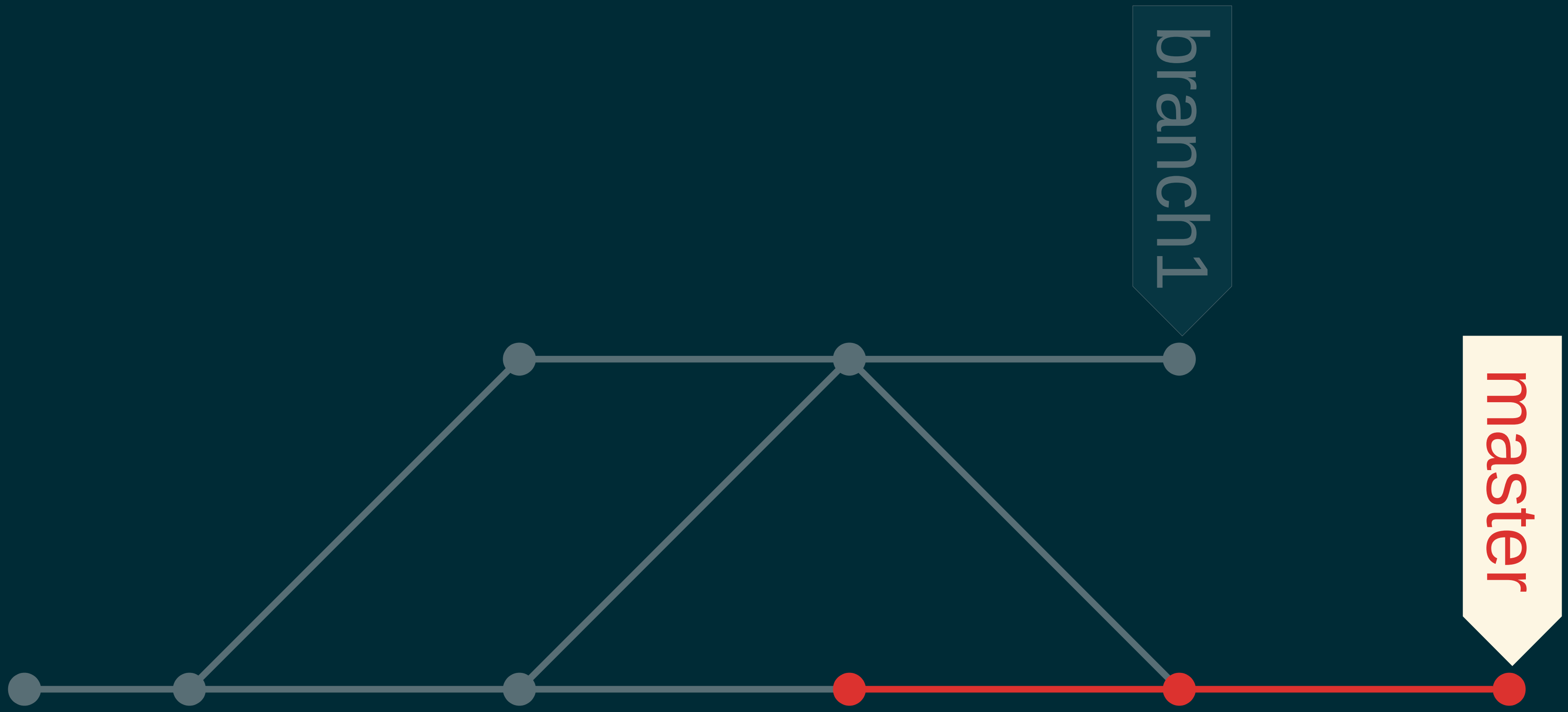
# History of master



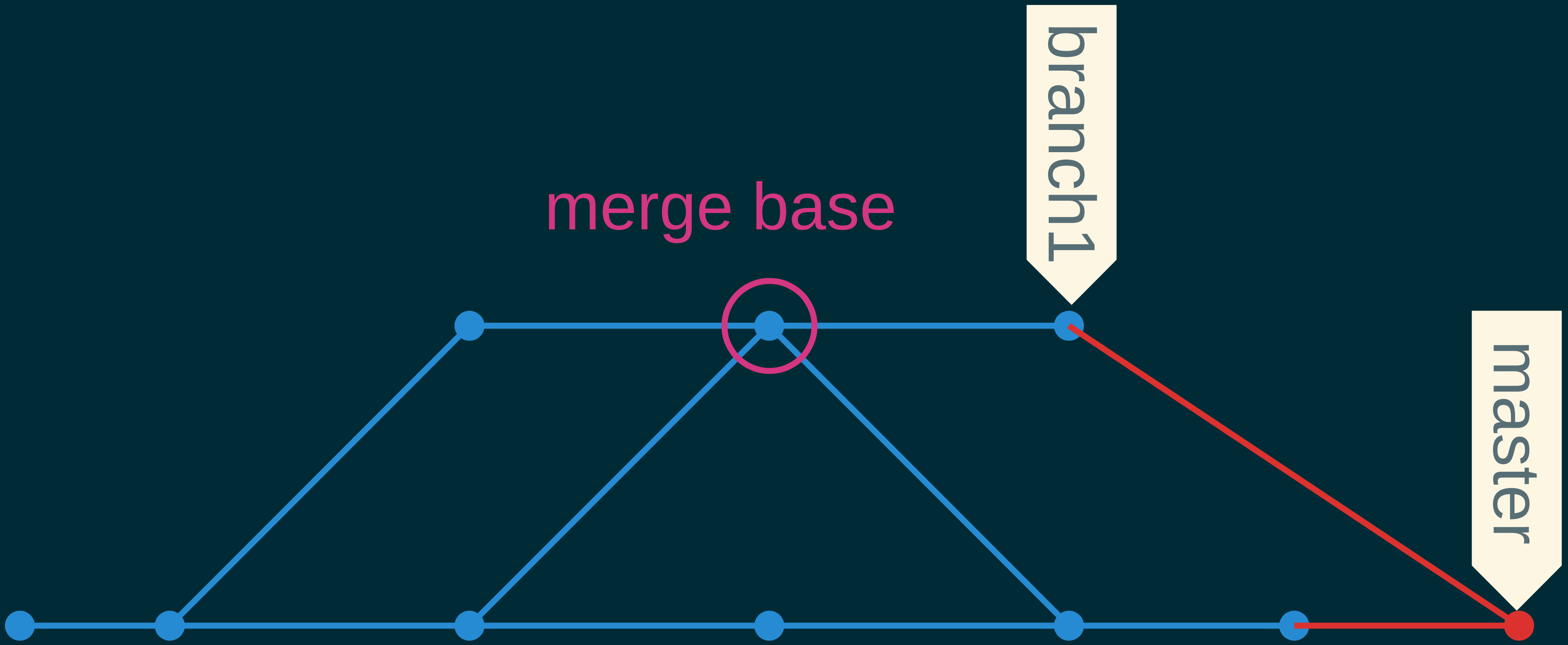
# branch1 --not master



## master --not branch1



# Merging: the main point of branches



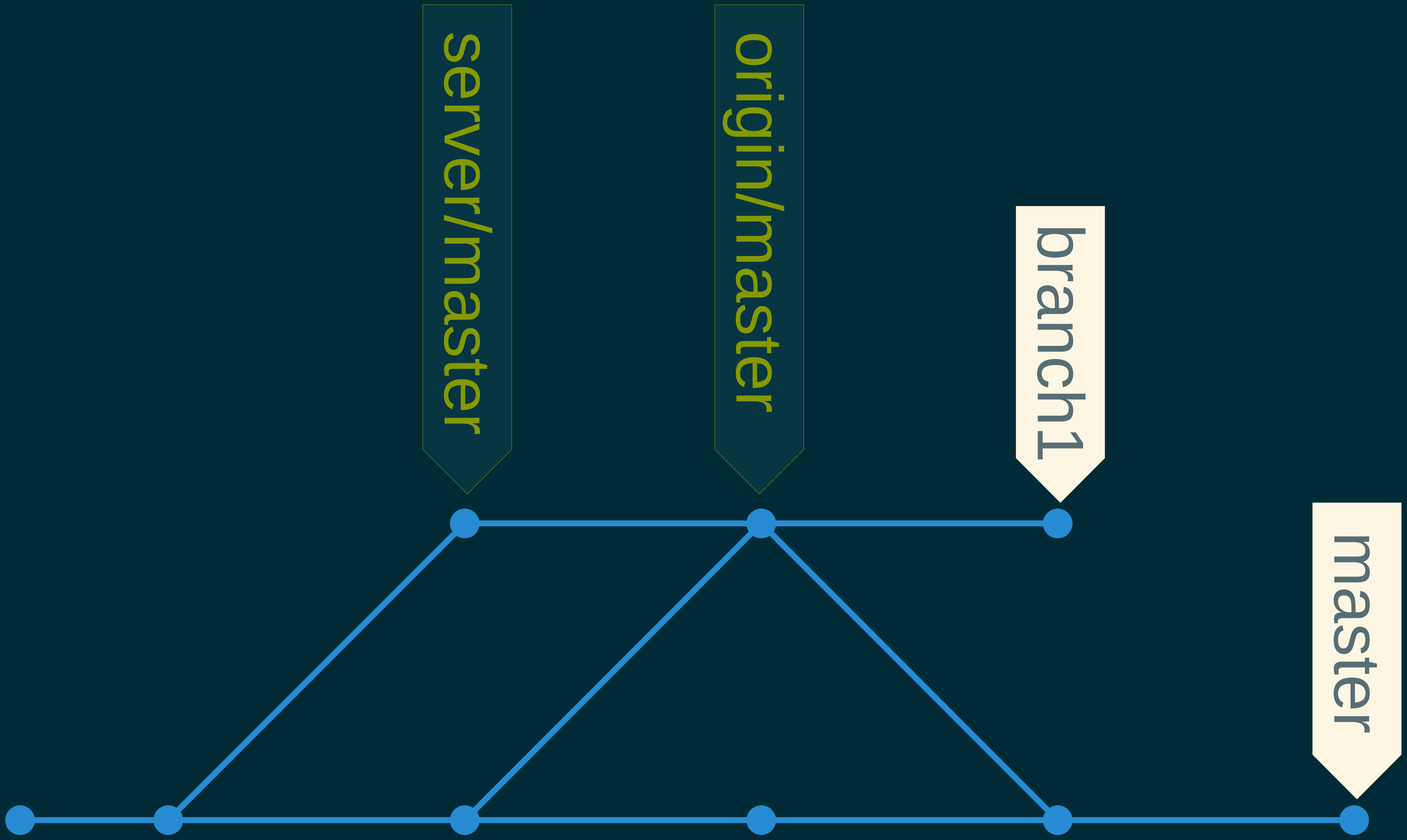
## Checking changes on a branch

- `git log A --not B`
- `gitk A --not B`
- `git merge-base A B`
- `git log -1` *`commit`*
- `git log --oneline --graph`
- `git diff A B`

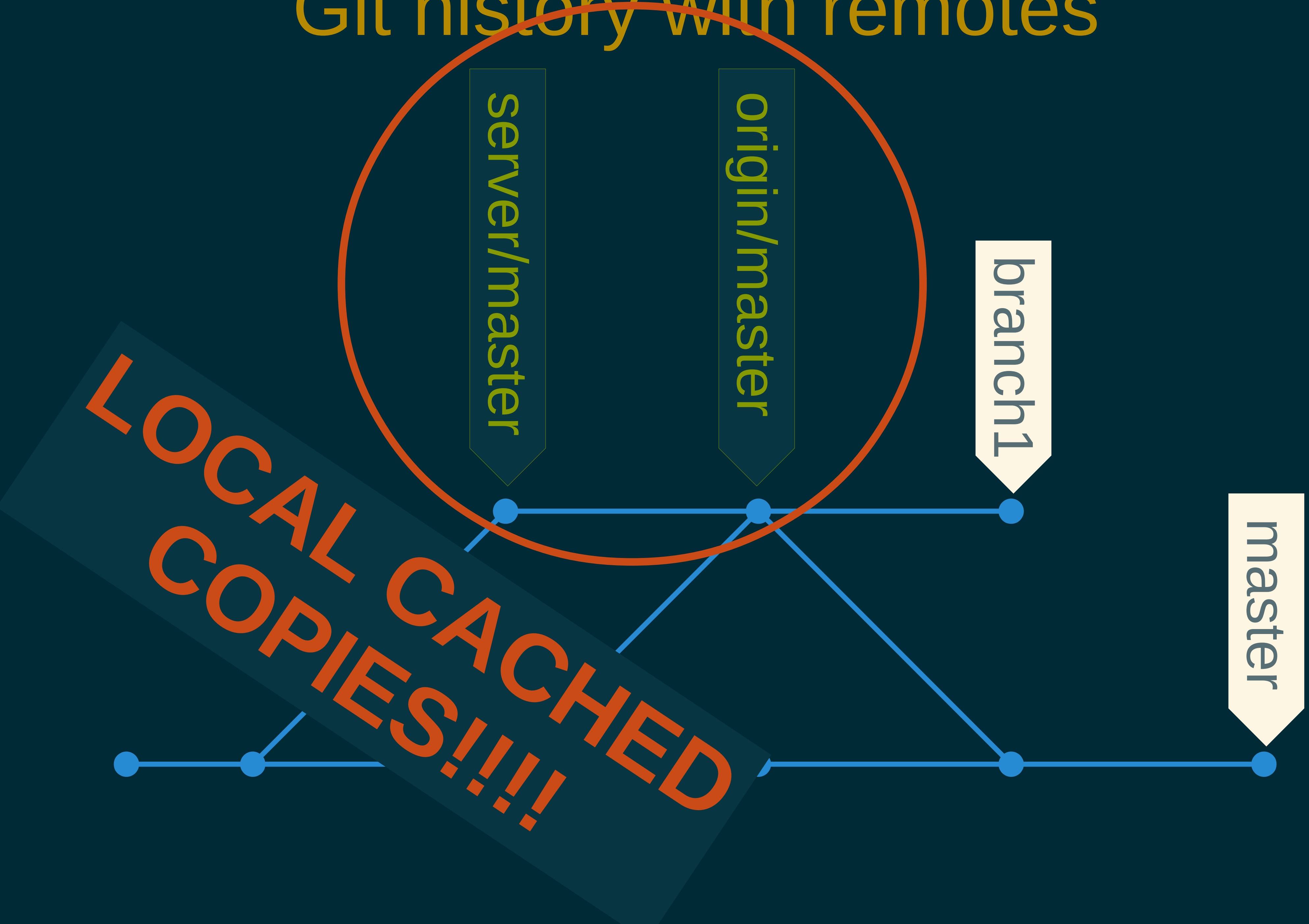
## Multiple servers

- Repositories on servers are called *remotes* - but so are some other things, watch out!
- Look at `.git/config`
- `git fetch` *`server`*
- `git push` *`server`* `A:B`

## Git history with remotes



# Git history with remotes



## Checking remote branches

`git fetch`

Then use same techniques as for branches

- `git log` *`remote`*`/A --not` *`remote`*`/B`
- `gitk` *`remote`*`/A --not` *`remote`*`/B`
- `git show-branch --merge-base ...`
- `git log -1` *`commit`*
- `git log --oneline --graph`
- `git diff` *`remote`*`/A` *`remote`*`/B`

## Putting it together: simple two-server workflow

origin - code archive
server - production

```
git showchanges branch1
git merge branch1
git push origin master
git push origin :branch1
git push server master
```

(on server) `git merge master`

## Final touch: make it like Heroku using post-receive

```
#!/bin/sh

unset GIT_DIR
cd ..
exec git merge master --ff
```