Working with Git Branches

Understanding Git Branch Basics

Craig Golightly
Senior Software Consultant

@seethatgo www.seethatgo.com

GitHub Default Branch Naming

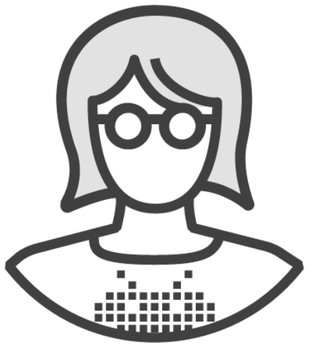
Master (adjective) main; principal

Oct 2020 changed default from `master` to `main`

Organizations can set a default branch name

This course will use `main`

Existing repositories will likely use `master`



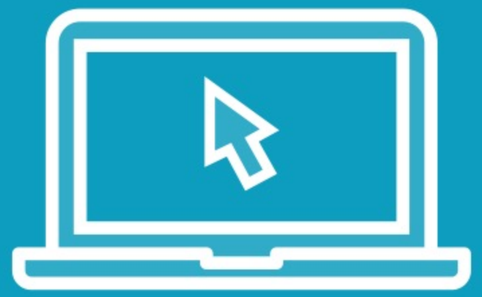
Oops Path

“Quick” Fix

How to save work?

Commit on a branch

Demo



- Command line
  - Available in any environment
  - Understand what is happening
- Initialize directory
- Add files to main for current state
- Make changes to files
- Move those changes into a branch

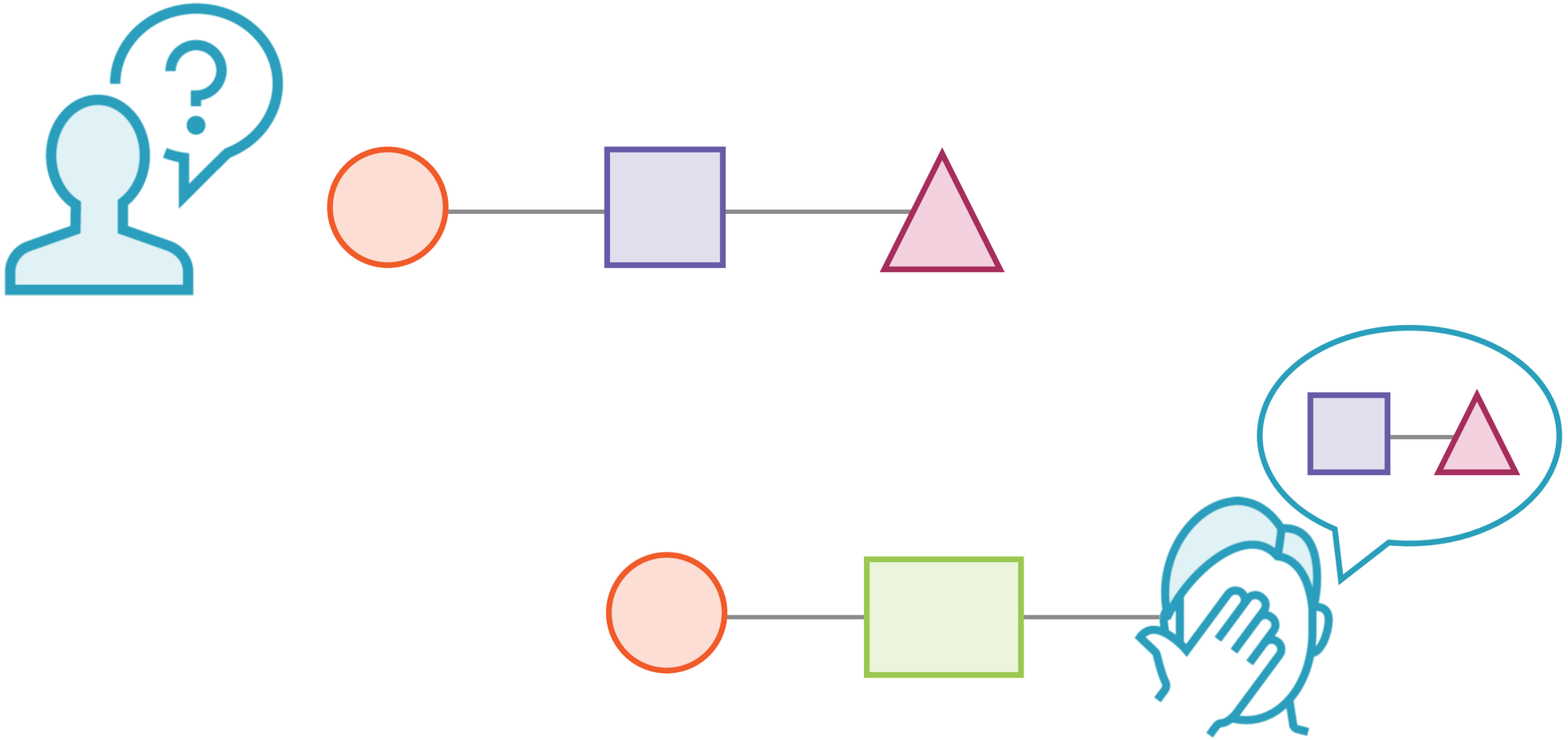
```
git checkout -b quickfix
```

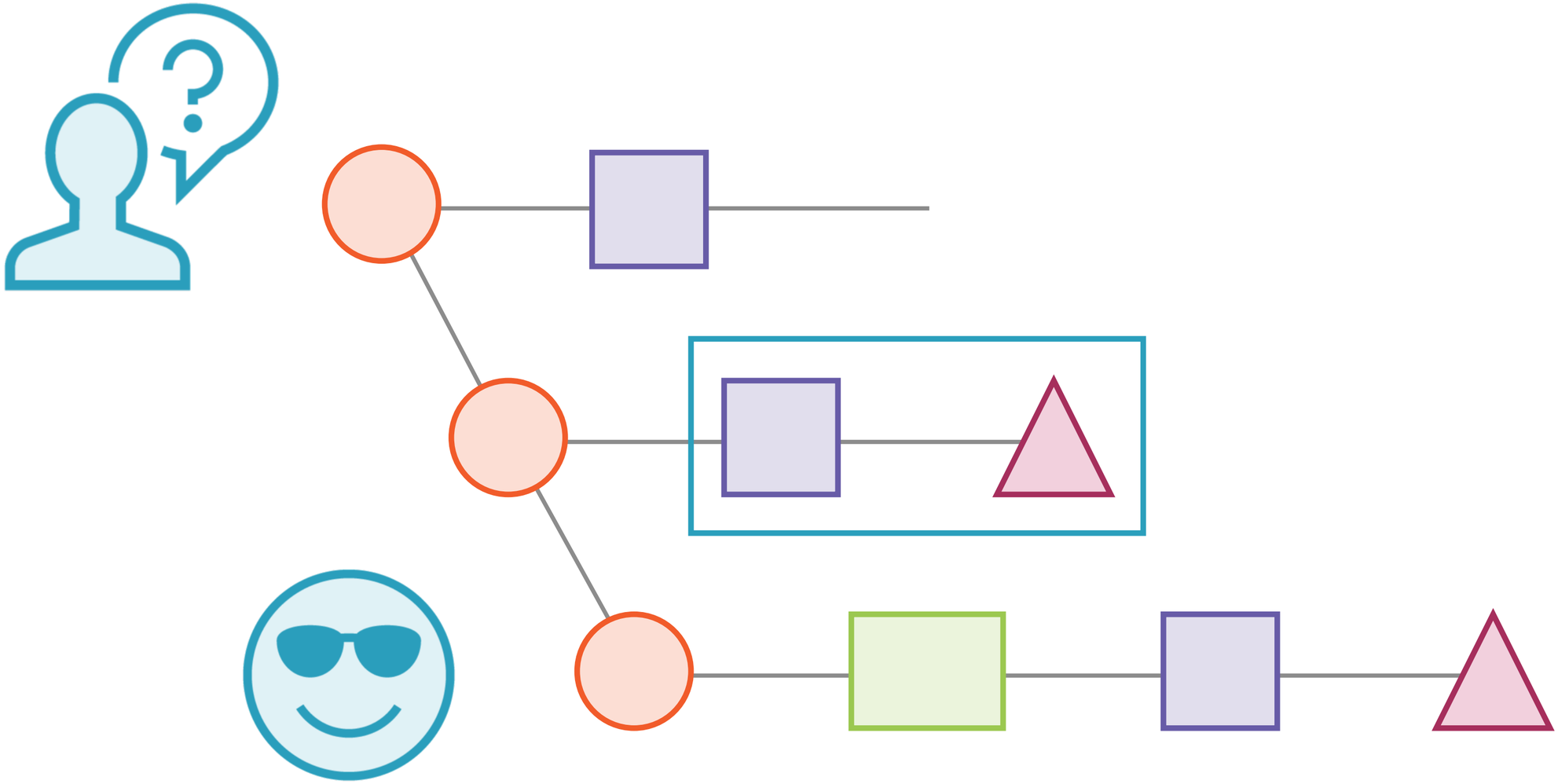
```
git add *
```

```
git commit -m "start of changes
for quickfix"
```

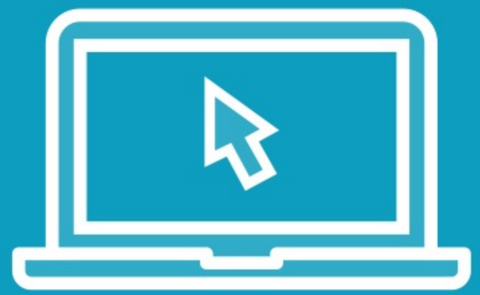
```
git checkout main
```

- **Create a branch called `quickfix` and move to that branch**
- **Stage the changes**
- **Commit the changes in the `quickfix` branch**
- **`main` is now clean and changes are saved in `quickfix` branch**





Demo



- Create a branch to solve a problem
  - Commit the first attempt
- Create a second branch from main
  - Commit partial solution
- Go back to first branch to copy step
- Complete solution in second branch

```
git branch
```

- List your branches

```
git checkout <branch-name>
```

- Switch to work in a different branch

```
git switch <branch-name>
```

```
git checkout -b <branch-name>
git switch -c <branch-name>
```

- Create a new branch and switch to that new branch

Dirty Branch

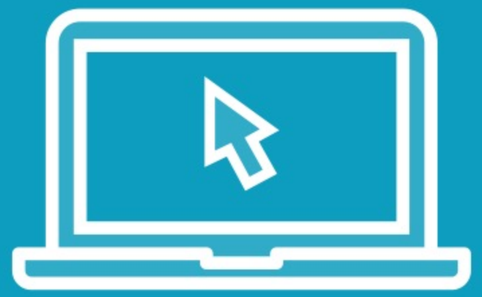
Dirty

(uncommitted changes)

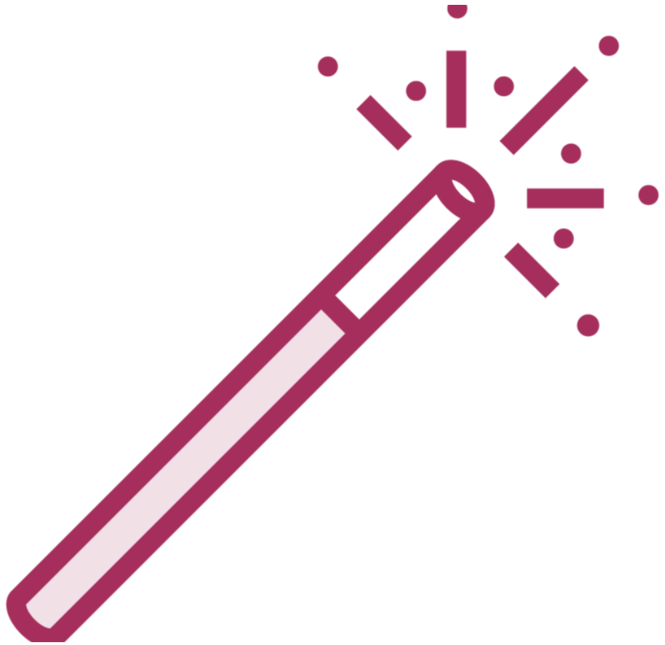
Clean

(no uncommitted changes)

Demo



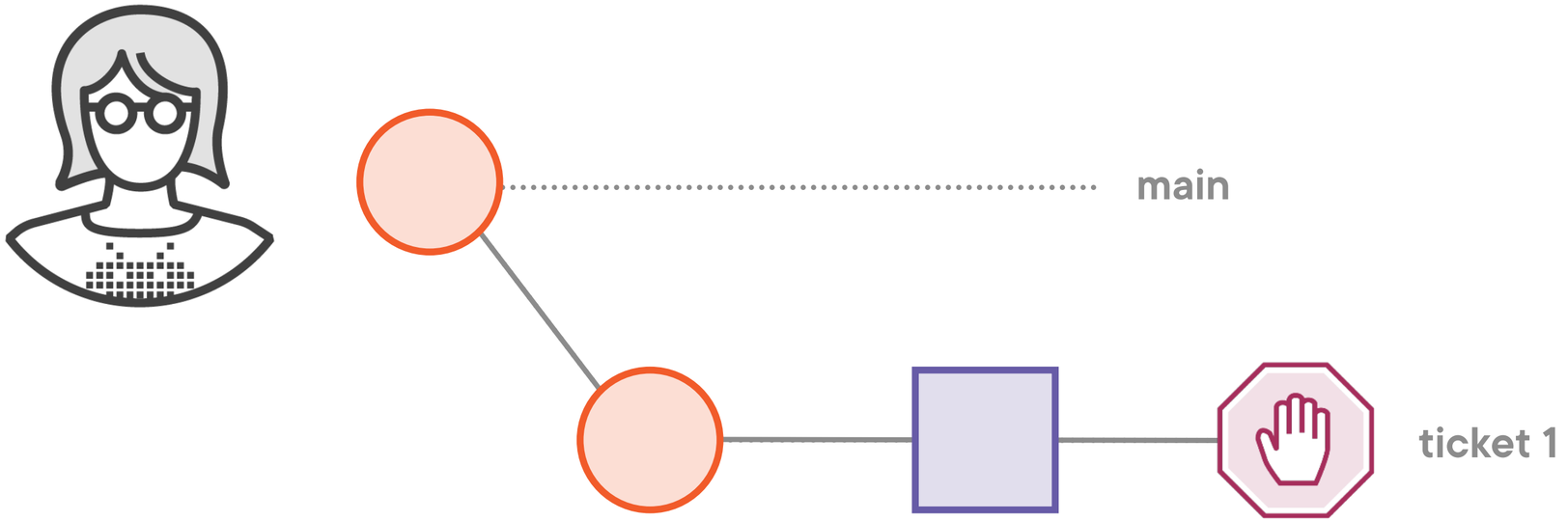
- Make a change on a branch
- Attempt to move to another branch
- Resolve error message
  - Switch branches

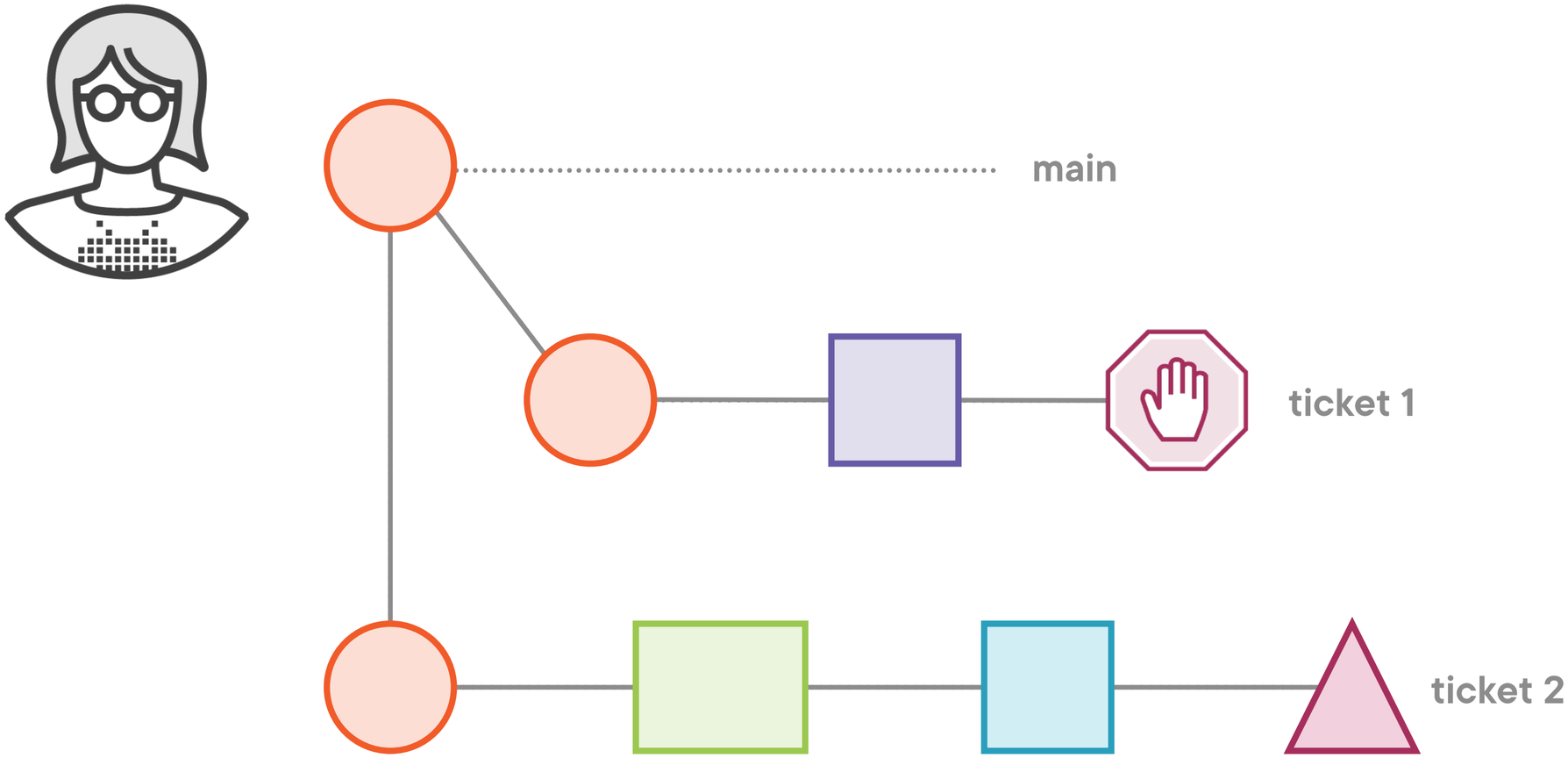


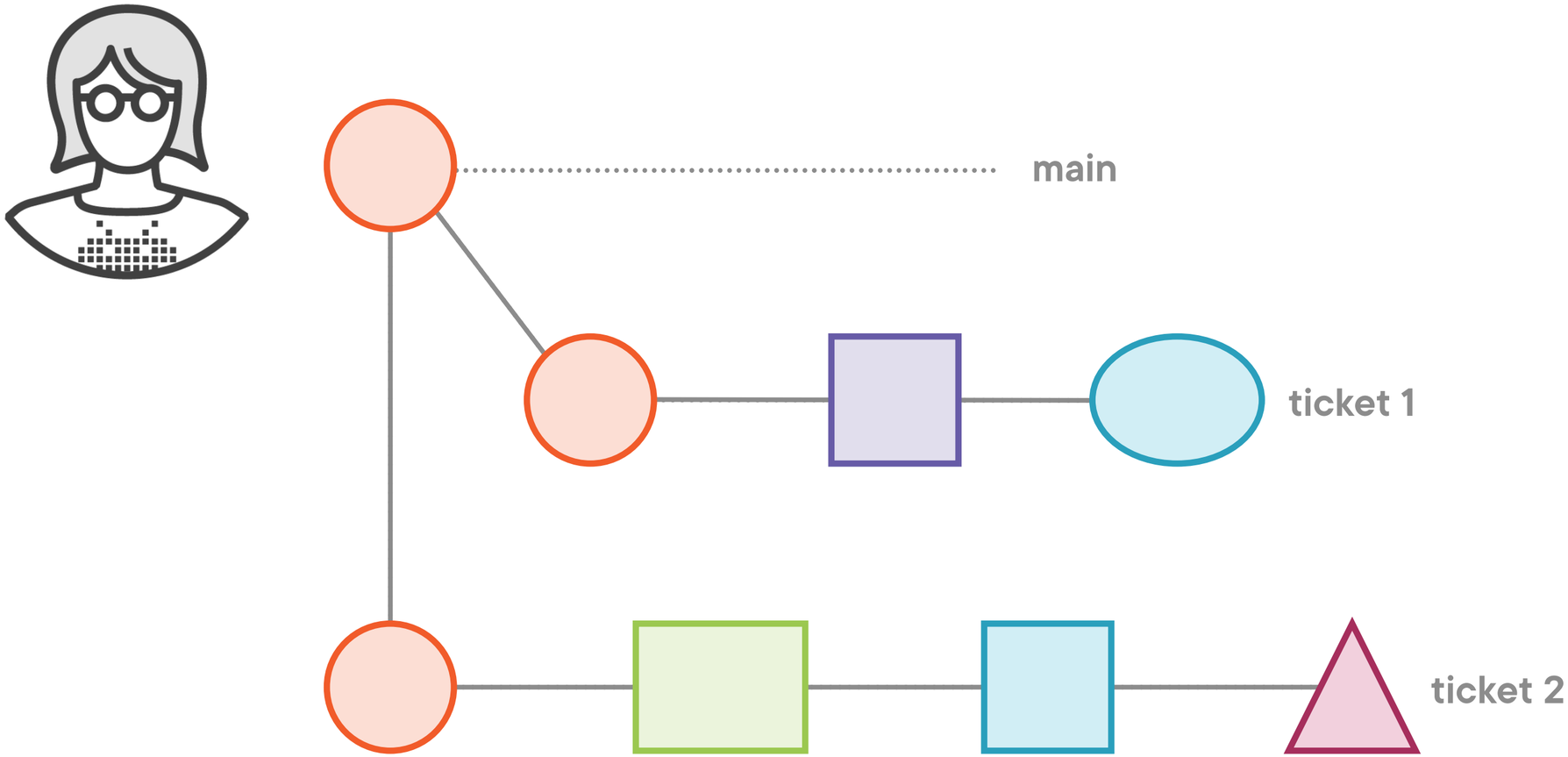
Git updates filesystem when change branches

Open editors reflect changes

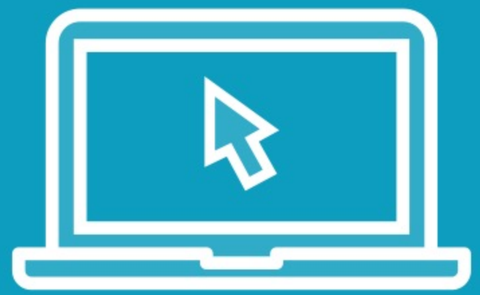
Most IDEs indicate what branch you are working on


main
ticket 1


main
ticket 1
ticket 2


main
ticket 1
ticket 2

Demo



- Work on a ticket in a branch
  - Blocked
- New branch for different ticket
- Continue first ticket when receive info

Branches help you organize your work

Changes are isolated in each branch

File system automatically updates when switch branches

Cleaner and less error-prone than trying to do it yourself without branches

“There are 2 hard problems in computer science: cache invalidation, naming things, and off-by-1 errors.”

Phil Karlton and Leon Bambrick

```
git branch -m <current name> <new name>
```

```
git branch -m quickfix longfix
```

```
git branch -m <new name for current branch>
```

```
git branch -m hotfix1
```

Rename a Branch

Shortcut to rename current working branch

```
git branch -d <branchname>
```

```
git branch -d longfix
```

```
git branch -D longfix
```

Delete a Branch

Error message if unmerged commits (will lose that work)

Can force delete if sure

Cannot delete current working branch

- Switch to a different branch first

Git File Management

Untracked

Not added or committed

Stay in directory

Staged

Added and ready for commit

Stay in staged state

Committed

Belong to branch

Changes not visible when switch branch

Summary

Create, rename, delete

Move from branch to branch

How branching can help you

- Oops case
- Iterating to solve a problem
- When you are blocked

Git filesystem management

- Untracked, staged, committed

Up Next:
Merging Made Easy