Learn Git Branching

$ help
$ levels

⚙ Select a level

Main Remote

Introduction Sequence

A nicely paced introduction to the majority of git commands

4: Rebase Introduction

Ramping Up

The next serving of 100% git awesomes-ness. Hope you're hungry

1: Detach yo' HEAD

Moving Work Around

"Git" comfortable with modifying the source tree :P

1: Cherry-pick Intro

A Mixed Bag

A mixed bag of Git techniques, tricks, and tips

1: Grabbing Just 1 Commit

Advanced Topics

For the truly brave!

1: Rebasing over 9000 times

Learn Git Branching

Show Goal   ⚡ Level *Pull arguments*   Instructions

$ clear
$ reset
$ git pull origin C3:foo
$ git pull origin C2:side

Great Job!!!

You solved the level in *2* command(s); our solution uses 2.

Awesome! You matched or exceeded our solution.

Wow! You finished the last level, great! (/^_^)/ (/^_^)/ (/^_^)/

Cancel   Confirm

Screenshot 1 (top):

Learn Git Branching

Show Goal | ↑ Level Source of nothing | Instructions

```
$ level remoteAdvanced7
$ hint
The branch command is disabled for this
level so you'll have to use fetch!
$ delay 2000
$ show goal
$ git push origin :foo
$ git fetch origin :bar
```

Great Job!!!

You solved the level in *2* command(s); our solution uses 2.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Pull arguments"*, the next level?

Cancel | Confirm

Screenshot 2 (bottom):

Learn Git Branching

Show Goal | ↑ Level Fetch arguments | Instructions

```
$ level remoteAdvanced6
$ hint
Pay attention how the commit ids may have
swapped! You can read slides again with
"help level"
$ delay 2000
$ show goal
$ git fetch origin c3:foo
$ git fetch origin c6main
The ref c6main does not exist or is
unknown
$ git fetch origin c6:main
$ git checkout foo
$ git merge main
```

Great Job!!!

You solved the level in *4* command(s); our solution uses 4.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Source of nothing"*, the next level?

Cancel | Confirm

Screenshot 1

Learn Git Branching

Show Goal | Level: Git push arguments -- Expanded! | Instructions

```
$ level remoteAdvanced5
$ hint
Remember you can admit defeat and type in
"show solution" :P
$ delay 2000
$ show goal
$ git push origin foo
$ git push origin main^:foo^2
That branch name "foo^2" is not allowed!
$ reset
$ git push origin foo^
That branch name "C2" is not allowed!
$ git push origin foo^:foo
$ git push origin foo^2:foo
The remote repository has diverged from
your local repository, so uploading your
changes is not a simple fast forward (and
thus your push was rejected). Please pull
down the new changes in the remote
repository, incorporate them into this
branch, and try again. You can do so with
git pull or git pull --rebase
$ reset
$ git push origin main^:foo
$ reset
$ git push origin foo:main
$ git push origin main^:foo
$
```

Great Job!!!

You solved the level in *2* command(s); our solution uses 2.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Fetch arguments"*, the next level?

Cancel | Confirm

Screenshot 2

Learn Git Branching

Show Goal | Level: Git push arguments | Instructions

```
$ level remoteAdvanced4
$ hint
You can always look at the last slide of
the dialog with "objective"
$ delay 2000
$ show goal
$ git push origion foo
origion is not a remote in your
repository! try adding origin to that
argument
$ git push origin foo
$ git push origin main
$
```

Great Job!!!

You solved the level in *2* command(s); our solution uses 2.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Git push arguments -- Expanded!"*, the next level?

Cancel | Confirm

Learn Git Branching

Show Goal | ↕ Level Remote Tracking | Instructions

```
$ level remoteAdvanced3
$ hint
Remember there are two ways to set remote
tracking!
$ delay 2000
$ show goal
$ git checkout -b side o/main
❶ local branch "side" set to track remote
branch "o/main"
$ git commit
$ git pull --rebase
$ git push
```

Great Job!!!

You solved the level in 4 command(s); our solution uses 4.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Git push arguments"*, the next level?

Cancel | Confirm

Learn Git Branching

Show Goal | ↕ Level Merging with remotes | Instructions

```
$ level remoteAdvanced2
$ hint
Pay attention to the goal tree!
$ delay 2000
$ show goal
$ git fetch
$ git merge
I expect at least 1 argument(s) for with
git merge
$ git merge o/main
$ undo
$ git merge side1 o/main
I expect at most 1 argument(s) for with
git merge
$ git checkout o/main
$ git merge side1
$ reset
$ git checkout main
$ git pull
Fast forwarding...
$ git merge side1
$ git merge side2
$ git merge side3
$ git push
```

Great Job!!!

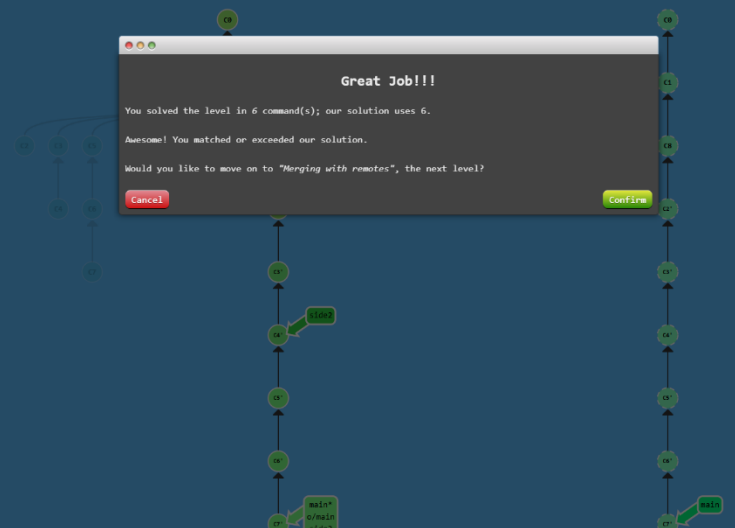You solved the level in 6 command(s); our solution uses 6.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Remote Tracking"*, the next level?

Cancel | Confirm

Screen 1 (top):

Learn Git Branching

Show Goal | ↕ Level *Diverged History* | Instructions

```
$ level remote7
$ hint
Check out the ordering from the goal
visualization
$ delay 2000
$ show goal
$ git clone
● local branch "main" set to track remote
branch "o/main"
$ git fakeTeamwork
$ git commit
$ git pull --rebase
$ git push
```

Great Job!!!

You solved the level in 5 command(s); our solution uses 5.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Locked Main"*, the next level?

Cancel | Confirm

Screen 2 (bottom):

Learn Git Branching

Show Goal | ↕ Level *Git Pushin'* | Instructions

```
$ level remote6
$ hint
Remember you have to clone before you can
push!
$ delay 2000
$ show goal
$ objective
$ git commit 2
That command accepts no general arguments
$ git commit
$ git commit
$ git push
```

Great Job!!!

You solved the level in 3 command(s); our solution uses 3.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Diverged History"*, the next level?

Cancel | Confirm

Screenshot 1 (top):

Learn Git Branching

Show Goal | ↕ Level Faking Teamwork | Instructions

```
$ level remote5
$ hint
Remember you can specify the number of
commits to fake
$ delay 2000
$ show goal
$ git clone
🛈 local branch "main" set to track remote
branch "o/main"
$ git fakeTeamwork 2
$ git commit
$ git pull
```
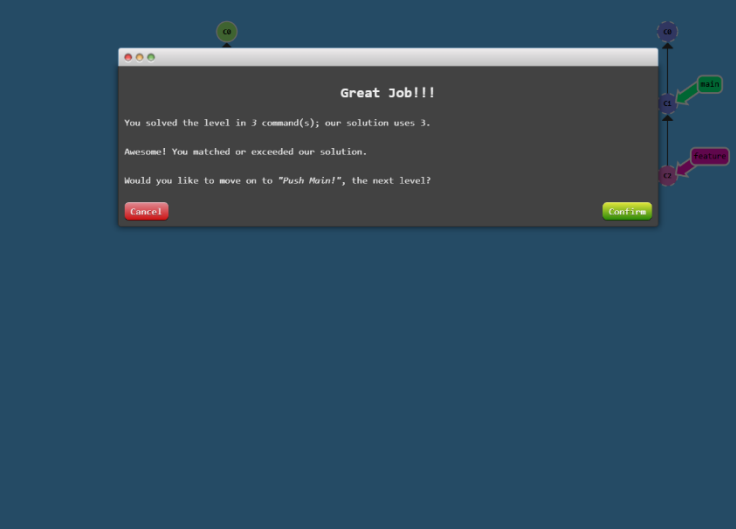
Great Job!!!

You solved the level in 4 command(s); our solution uses 4.

Awesome! You matched or exceeded our solution.

Would you like to move on to "Git Pushin'", the next level?

Cancel | Confirm

Screenshot 2 (bottom):

Learn Git Branching

Show Goal | ↕ Level Git Pushin' | Instructions

```
$ level remote4
$ hint
Just run git pull!
$ delay 2000
$ show goal
$ git pull
```

Great Job!!!

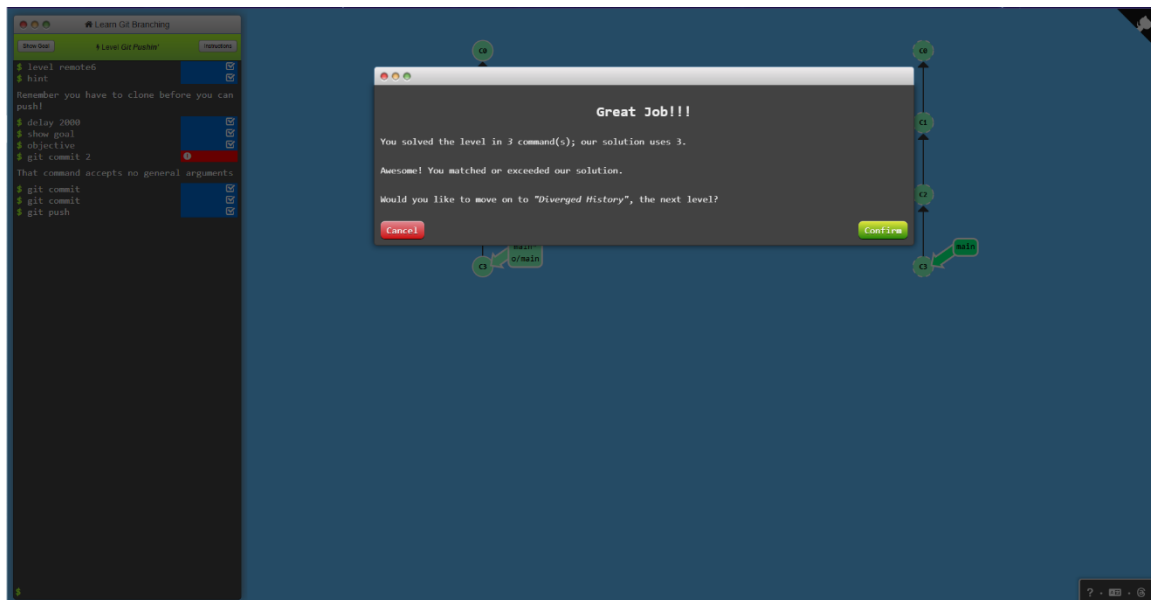You solved the level in 1 command(s); our solution uses 1.

Awesome! You matched or exceeded our solution.

Would you like to move on to "Faking Teamwork", the next level?

Cancel | Confirm

Learn Git Branching

Level Git Fetchin'

```
$ level remote3
$ hint
just run git fetch!
$ delay 2000
$ show goal
$ git fetch
```

Great Job!!!

You solved the level in *1* command(s); our solution uses 1.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Git Pullin'"*, the next level?

Cancel   Confirm

Learn Git Branching

Level Remote Branches

```
$ level remote2
$ hint
Pay attention to the ordering -- commit
on main first!
$ delay 2000
$ show goal
$ git commit
$ git checkout o/main
$ git commit
⚠ Warning!! Detached HEAD state
```

Great Job!!!

You solved the level in *3* command(s); our solution uses 3.
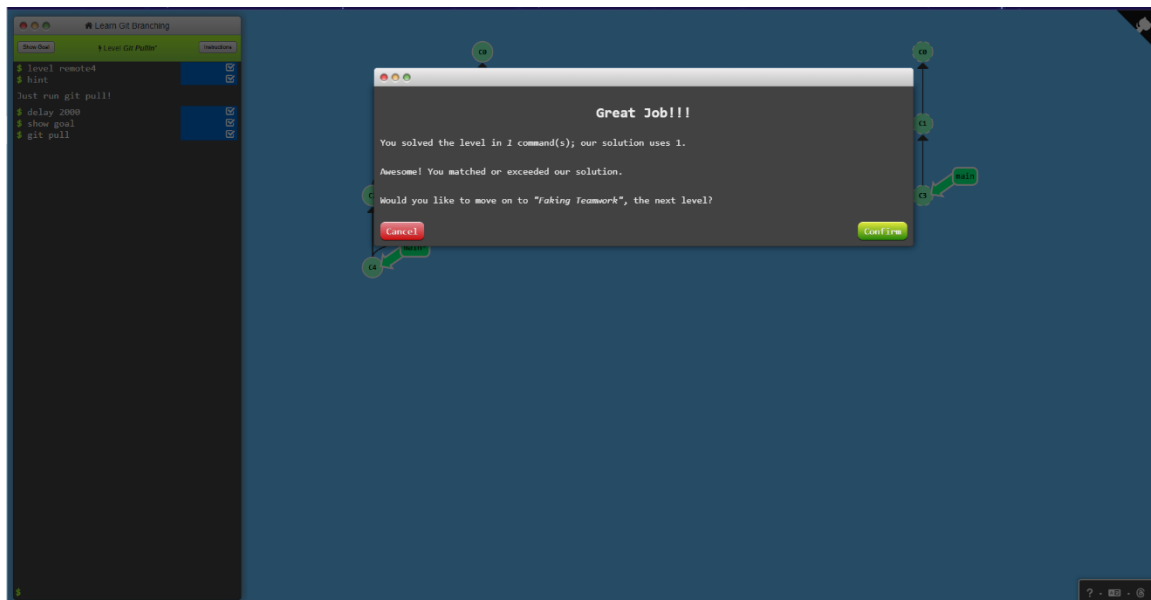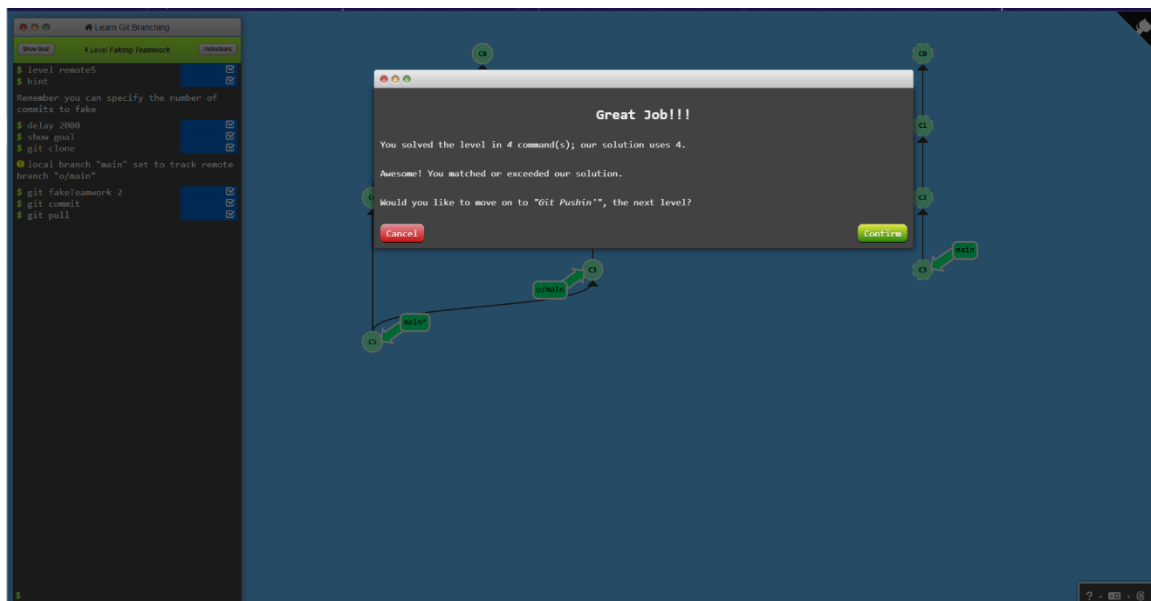
Awesome! You matched or exceeded our solution.

Would you like to move on to *"Git Fetchin'"*, the next level?

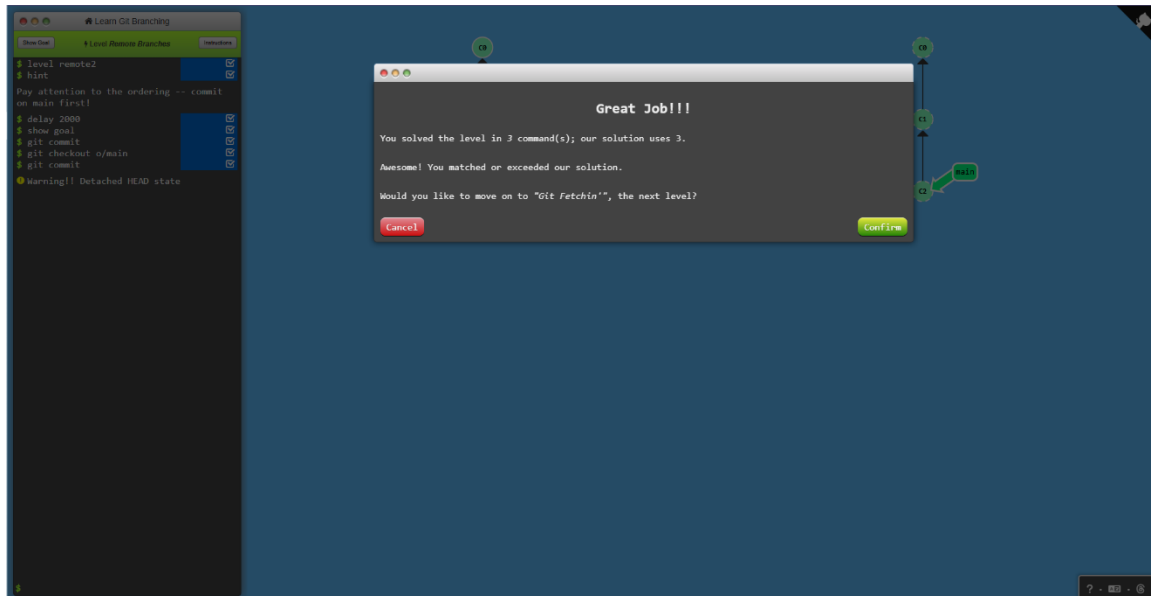Cancel   Confirm

Top screen — Learn Git Branching — Level: Multiple parents

```
$ level advanced2
$ hint
Use `git branch bugWork` with a target
commit to create the missing reference.
$ delay 2000
$ show goal
$ git checkout -b bugWork HEAD^2~2
$ undo
$ git checkout -b bugWork HEAD^2~1
$ git checkout main
```

Great Job!!!

You solved the level in *2* command(s); our solution uses 1.

See if you can whittle it down to 1 :D

Would you like to move on to *"Branch Spaghetti"*, the next level?

Cancel   Confirm

Bottom screen — Learn Git Branching — Level: Rebasing over 9000 times

```
$ level advanced1
$ hint
Remember, the most efficient way might be
to only update main at the end...
$ delay 2000
$ show goal
$ git rebase main bugFix
$ git rebase bugFix side
$ git rebase -i side another
$ git branch -f main
```

Great Job!!!

You solved the level in *4* command(s); our solution uses 4.

Awesome! You matched or exceeded our solution.

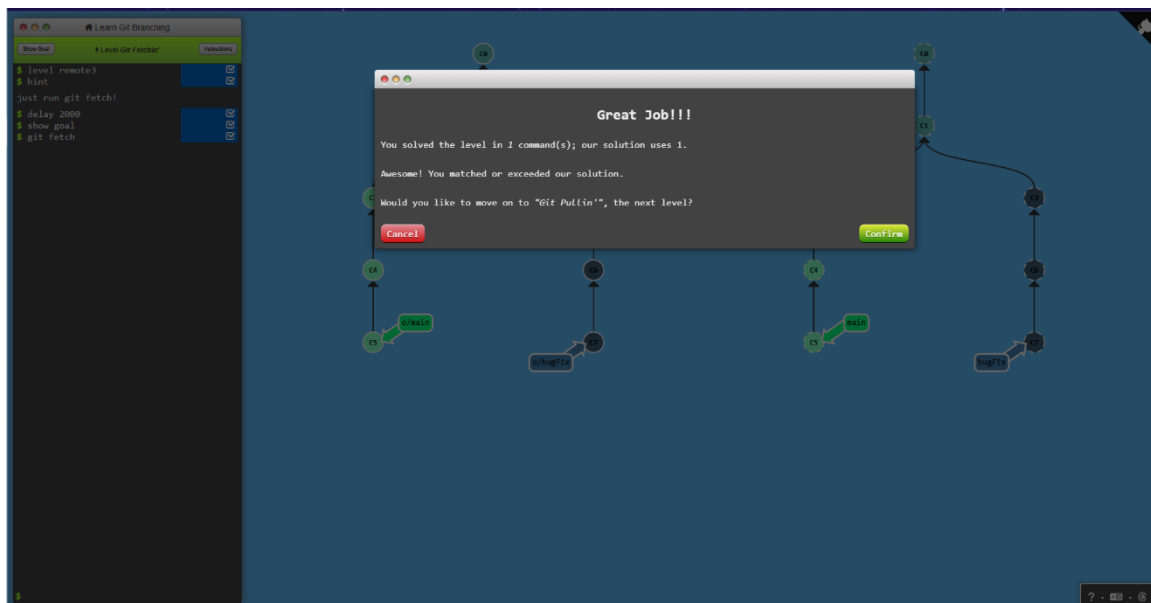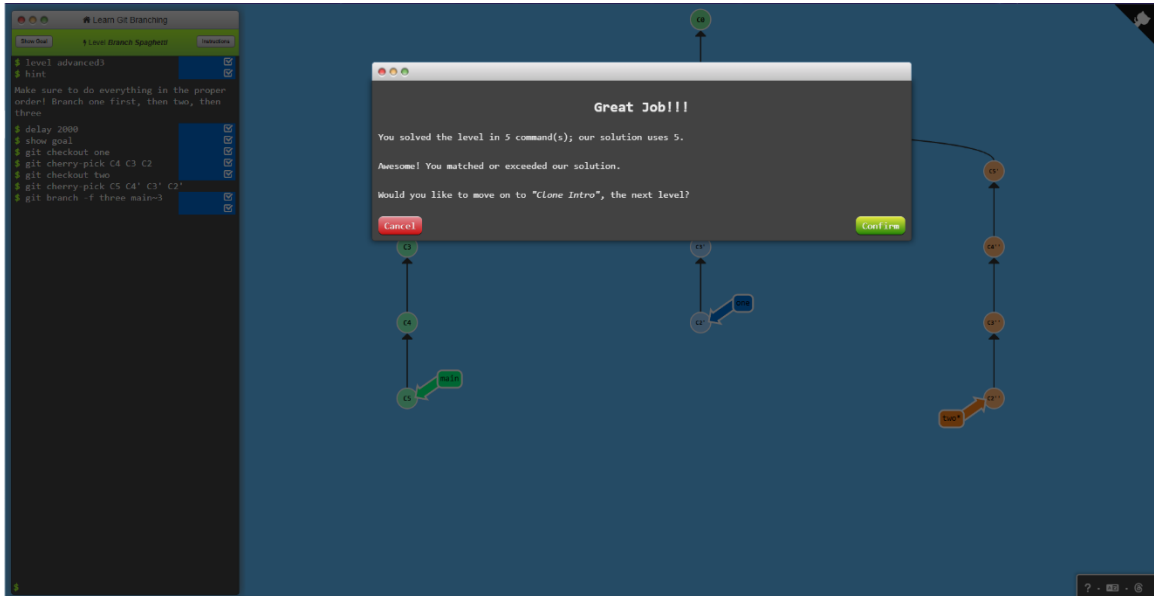Would you like to move on to *"Multiple parents"*, the next level?

Cancel   Confirm

Screen 1 (top):

Learn Git Branching

Show Goal | ▲ Level Git Describe | Instructions

```
$ level mixed5
$ hint
Just commit once on bugFix when you're
ready to move on
$ delay 2000
$ show goal
$ git describe
v1-2-gC6
$ git describe bugFix
v1-2-gC6
$ git describe C5
v1-1-gC5
$ git describe C3
v1
$ git describe C4
v1-1-gC4
$ git describe side
v1-1-gC4
$ git describe main
v0-2-gC2
$ git describe C1
v0-1-gC1
$ git commit
```

Great Job!!!

You solved the level in *1* command(s); our solution uses 1.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Rebasing over 9000 times"*, the next level?

Cancel | Confirm

Screen 2 (bottom):

Learn Git Branching

Show Goal | ▲ Level Git Tags | Instructions

```
$ level mixed4
$ hint
you can either check out the commit
directly or simply checkout the tag!
$ delay 2000
$ show goal
$ git tag C1 v0
The ref v0 does not exist or is unknown
$ git tag v0 C1
$ git tag v1 C2
$ git checkout v1
```

Great Job!!!

You solved the level in *3* command(s); our solution uses 3.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Git Describe"*, the next level?

Cancel | Confirm

Top panel — Learn Git Branching

Show Goal ↓ Level Juggling Commits #2 Instructions

```
$ level mixed3
$ hint
Don't forget to forward main to the
updated changes!
$ delay 2000
$ show goal
$ git checkout main
$ git cherry-pick c2
$ git branch -f HEAD^
That branch name "HEAD^" is not allowed!
$ git branch -f main HEAD^
$ git cherry-pick c2' c3
```

Great Job!!!

You solved the level in *4* command(s); our solution uses 4.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Git Tags"*, the next level?

Cancel   Confirm

Bottom panel — Learn Git Branching

Show Goal ↓ Level Juggling Commits Instructions

```
$ level mixed2
$ hint
The first command is git rebase -i HEAD~2
$ delay 2000
$ show goal
$ git rebase -i
I expect at least 1 argument(s) for with
rebase -i
$ git rebase -i main
$ git commit --amend
$ git rebase -i main
$ git branch -f main
```
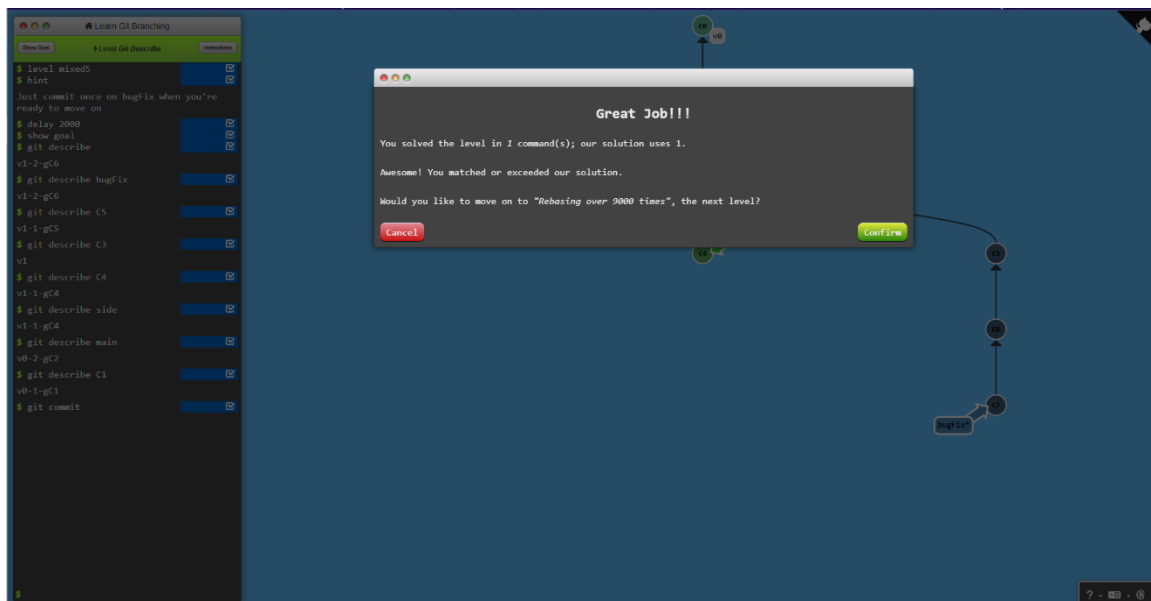
Great Job!!!

You solved the level in *4* command(s); our solution uses 4.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Juggling Commits #2"*, the next level?

Cancel   Confirm

Screenshot 1 (top):

Learn Git Branching

Show Goal | ↑ Level Grabbing Just 1 Commit | Instructions

```
$ level mixed1
$ hint
Remember, interactive rebase or cherry-pick is your friend here
$ delay 2000
$ show goal
$ git checkout main
$ git cherry-pick C4
```

Great Job!!!

You solved the level in *2* command(s); our solution uses 2.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Juggling Commits"*, the next level?

Cancel | Confirm

Screenshot 2 (bottom):

Learn Git Branching

Show Goal | ↑ Level Interactive Rebase Intro | Instructions

```
$ level move2
$ hint
you can use either branches or relative refs (HEAD~) to specify the rebase target
$ delay 2000
$ show goal
$ reset
$ git rebase -i HEAD~4
```

Great Job!!!

You solved the level in *1* command(s); our solution uses 1.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Grabbing Just 1 Commit"*, the next level?

Cancel | Confirm

Screenshot 1:

Learn Git Branching

Show Goal | Level: Cherry-pick Intro | Instructions

```
$ level move1
$ hint
git cherry-pick followed by commit names!
$ delay 2000
$ show goal
$ git cherry-pick main C3 C4 C7
The commit C1 already exists in your
changes set, aborting!
$ git cherry-pick C3 C4 C7
```

Great Job!!!

You solved the level in *1* command(s); our solution uses 1.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Interactive Rebase Intro"*, the next level?

Cancel | Confirm

C0

C3'

C4'

Main C7'

Screenshot 2:

Learn Git Branching

Show Goal | Level: Reversing Changes in Git | Instructions

```
$ level rampup4
$ hint
Notice that revert and reset take
different arguments.
$ delay 2000
$ show goal
$ git reset HEAD^
$ git checkout pushed
$ git revert HEAD
```

Great Job!!!

You solved the level in *3* command(s); our solution uses 3.

Awesome! You matched or exceeded our solution.

Would you like to move on to *"Cherry-pick Intro"*, the next level?

Cancel | Confirm

C0

Learn Git Branching

Show Goal   ↑ Level Introduction to Git   Instructions
Commits

```
$ level intro1
$ hint
Just type in 'git commit' twice to
finish!
$ delay 2000
$ show goal
$ git commit
$ git commit
```

Great Job!!!

You solved the level in 2 command(s); our solution uses 2.

Awesome! You matched or exceeded our solution.

Would you like to move on to "Branching in Git", the next level?

Cancel   Confirm

Learn Git Branching

```
$ help
$ levels
```

⚙ Select a level

Main   Remote

Push & Pull -- Git Remotes!

Time to share your 1's and 0's kids; coding just got social

1: Clone Intro

⭐ ⭐ ⭐ ⭐ ⭐ ⭐ ⭐ ⭐

To Origin And Beyond -- Advanced Git Remotes!

And you thought being a benevolent dictator would be fun...

1: Push Main!

⭐ ⭐ ⭐ ⭐ ⭐ ⭐ ⭐ ⭐