Git Gud

Neil Ashford

UQ Computing Society

March 23, 2018

Outline

Who am I

- Neil Ashford

Who am I

- Neil Ashford
- 4th year Software[1] Engineering

[1] ish

Who am I

- Neil Ashford
- 4th year Software[1] Engineering
- Committee member

[1] ish

Who am I

- Neil Ashford
- 4th year Software[1] Engineering
- Committee member
- **@artemis** on slack, `ashfordneil0@gmail.com`

[1] ish

Who am I

- Neil Ashford
- 4th year Software[1] Engineering
- Committee member
- **@artemis** on slack, `ashfordneil0@gmail.com`
- Tutor for CSSE2002, CSSE2010 and CSSE2310

[1] ish

What is this talk

Target Audience

What is this talk

Target Audience

- People who have never used version control.

What is this talk

Target Audience

- People who have never used version control.
- People who have "memorised 5 shell commands and trust them implicitly without understanding"

What is this talk

Target Audience

- People who have never used version control.
- People who have "memorised 5 shell commands and trust them implicitly without understanding"

What is this talk

Target Audience

- People who have never used version control.
- People who have "memorised 5 shell commands and trust them implicitly without understanding"

What I'll Cover

- Some really basic version control you're familiar with, and its flaws.

What is this talk

Target Audience

- People who have never used version control.
- People who have "memorised 5 shell commands and trust them implicitly without understanding"

What I'll Cover

- Some really basic version control you're familiar with, and its flaws.
- The pieces of a mental model of proper version control.

What is this talk

Target Audience

- People who have never used version control.
- People who have "memorised 5 shell commands and trust them implicitly without understanding"

What I'll Cover

- Some really basic version control you're familiar with, and its flaws.
- The pieces of a mental model of proper version control.
- How git represents each of those pieces to you.

What is this talk

Target Audience

- People who have never used version control.
- People who have "memorised 5 shell commands and trust them implicitly without understanding"

What I'll Cover

- Some really basic version control you're familiar with, and its flaws.
- The pieces of a mental model of proper version control.
- How git represents each of those pieces to you.
- Flaws with git, and alternatives.

What is this talk

Target Audience

- People who have never used version control.
- People who have "memorised 5 shell commands and trust them implicitly without understanding"

What I'll Cover

- Some really basic version control you're familiar with, and its flaws.
- The pieces of a mental model of proper version control.
- How git represents each of those pieces to you.
- Flaws with git, and alternatives.

What is this talk

Target Audience

- People who have never used version control.
- People who have "memorised 5 shell commands and trust them implicitly without understanding"

What I'll Cover

- Some really basic version control you're familiar with, and its flaws.
- The pieces of a mental model of proper version control.
- How git represents each of those pieces to you.
- Flaws with git, and alternatives.

What I Won't Cover

- How to **cheat** when using git

Outline

Meet the Culprit

- Live demos suck, so we're trying to stick with something familiar.

Meet the Culprit

- Live demos suck, so we're trying to stick with something familiar.
- Please welcome our old friend...

Meet the Culprit

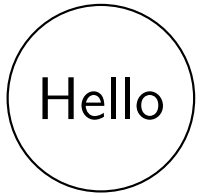
- Live demos suck, so we're trying to stick with something familiar.
- Please welcome our old friend...
- Microsoft Word[2]

[2] Okay I'm on a mac so it's pages but still

Undo / Reverts

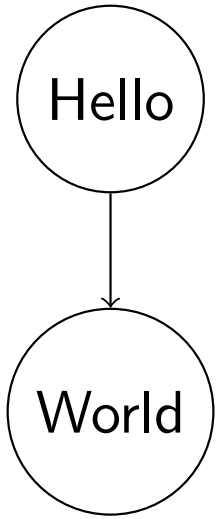
Undo / Reverts

- Type “Hello”



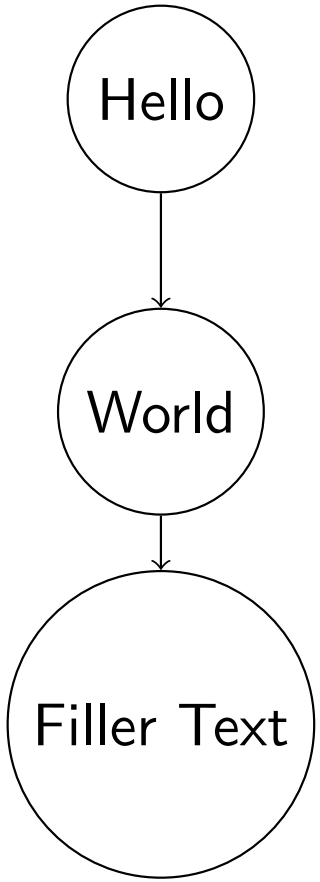
Undo / Reverts

- Type “Hello”
- Type “World”



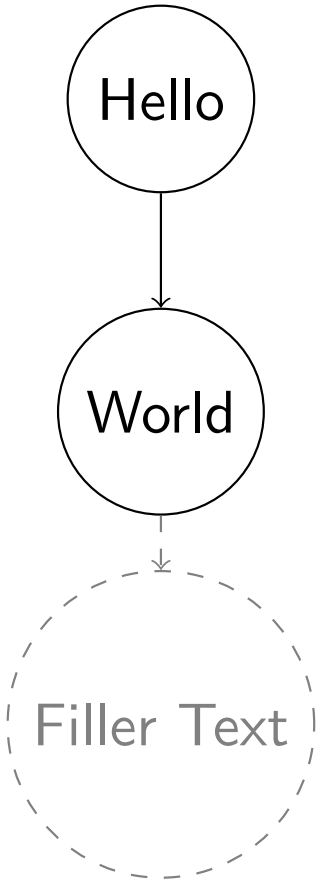
Undo / Reverts

- Type “Hello”
- Type “World”
- Type “Filler Text”



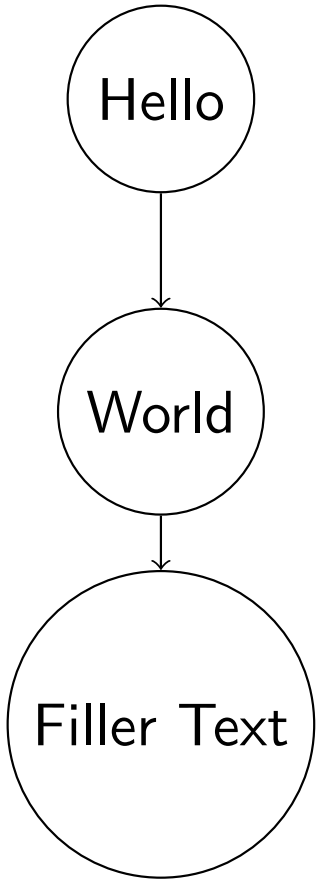
Undo / Reverts

- Type “Hello”
- Type “World”
- Type “Filler Text”
- Press Undo



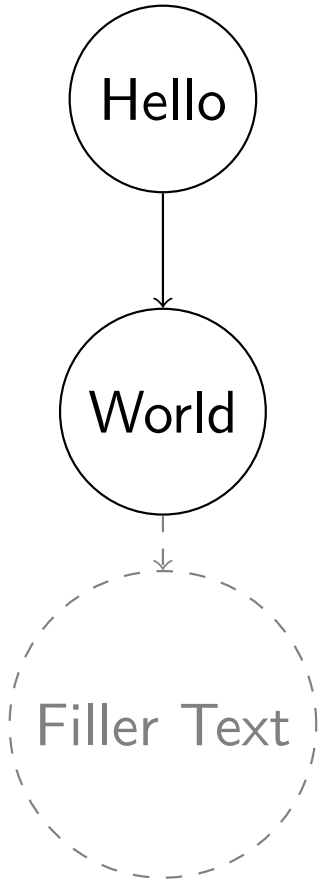
Undo / Reverts

- Type “Hello”
- Type “World”
- Type “Filler Text”
- Press Undo
- Press Redo



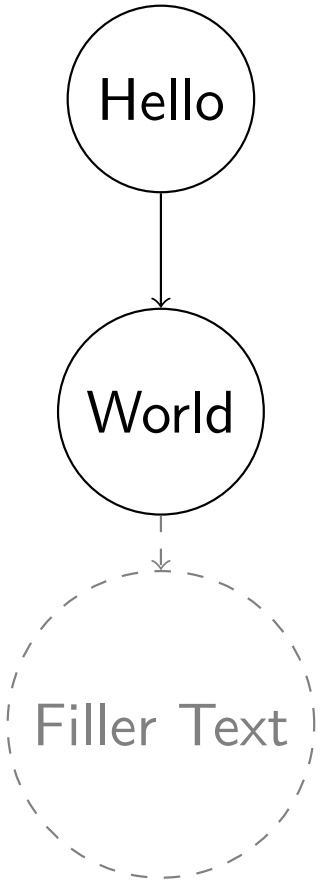
Undo / Reverts

- Type “Hello”
- Type “World”
- Type “Filler Text”
- Press Undo
- Press Redo
- Press Undo again



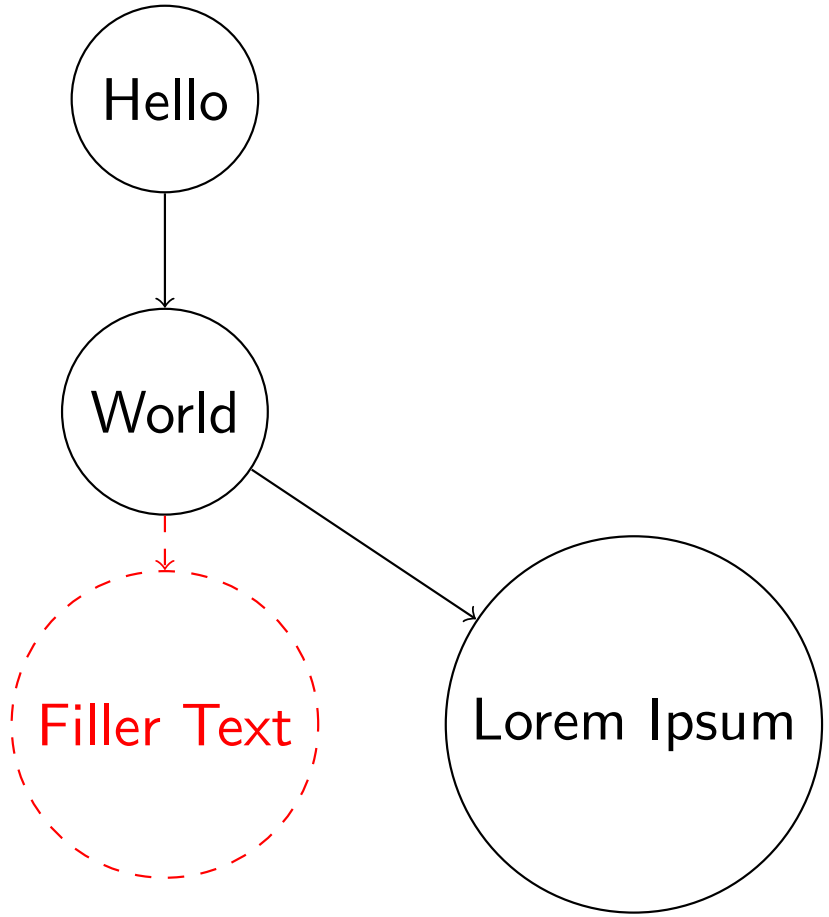
Branches

- From where we were before...



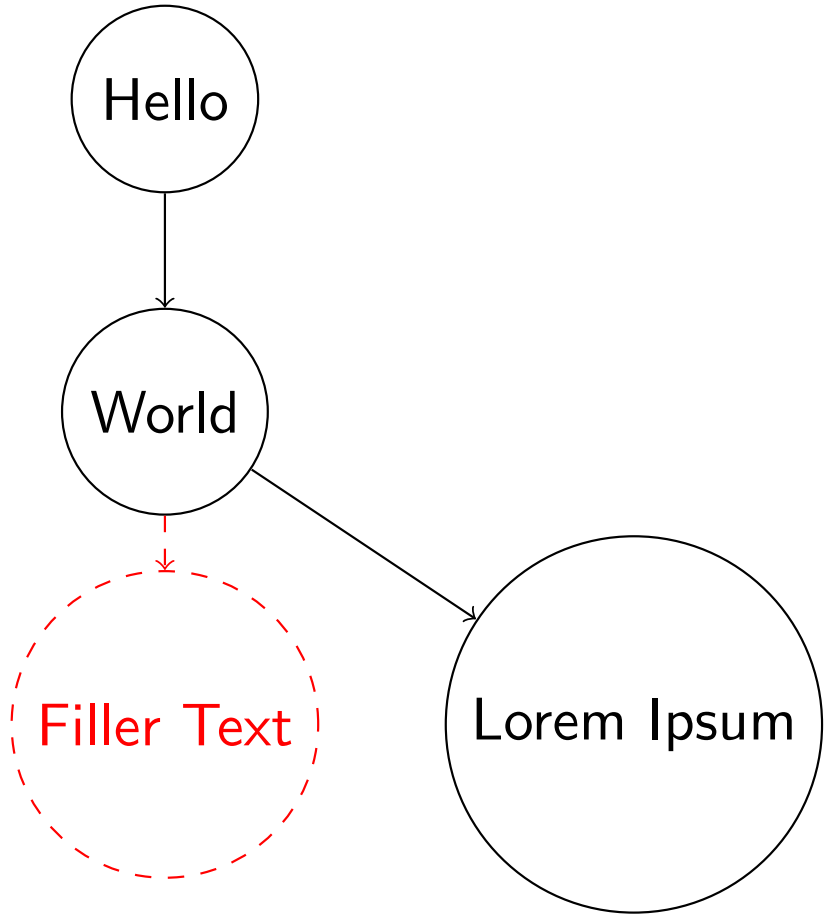
Branches

- From where we were before...
- Type “Lorem Ipsum”



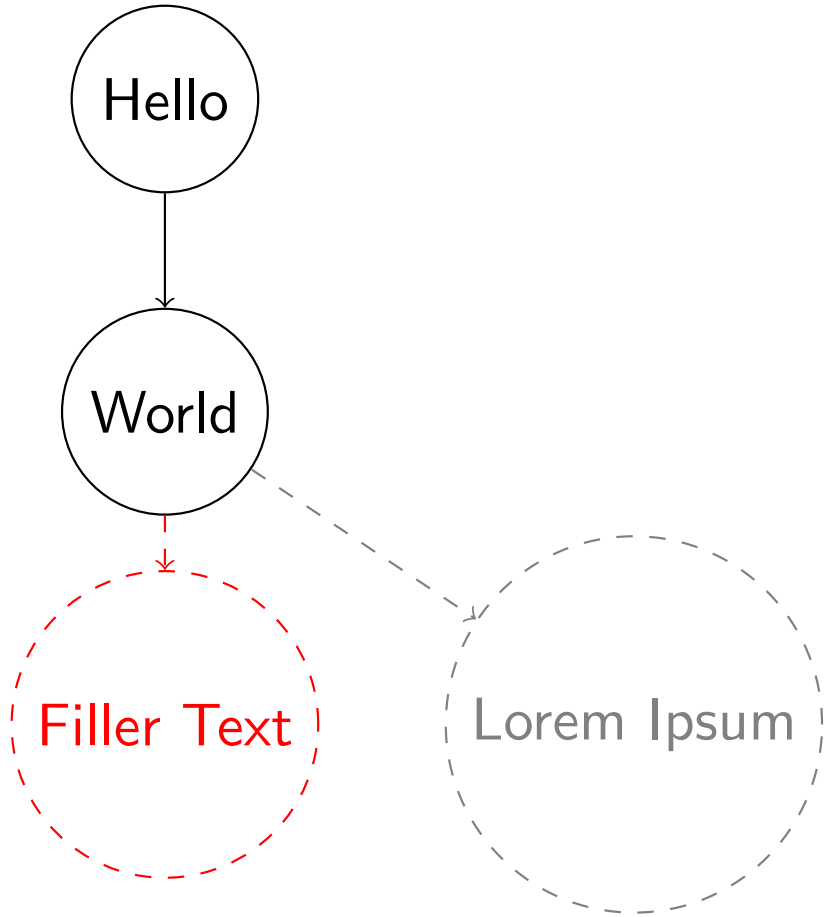
Branches

- From where we were before...
- Type "Lorem Ipsum"
- Realise we want "Filler Text" back



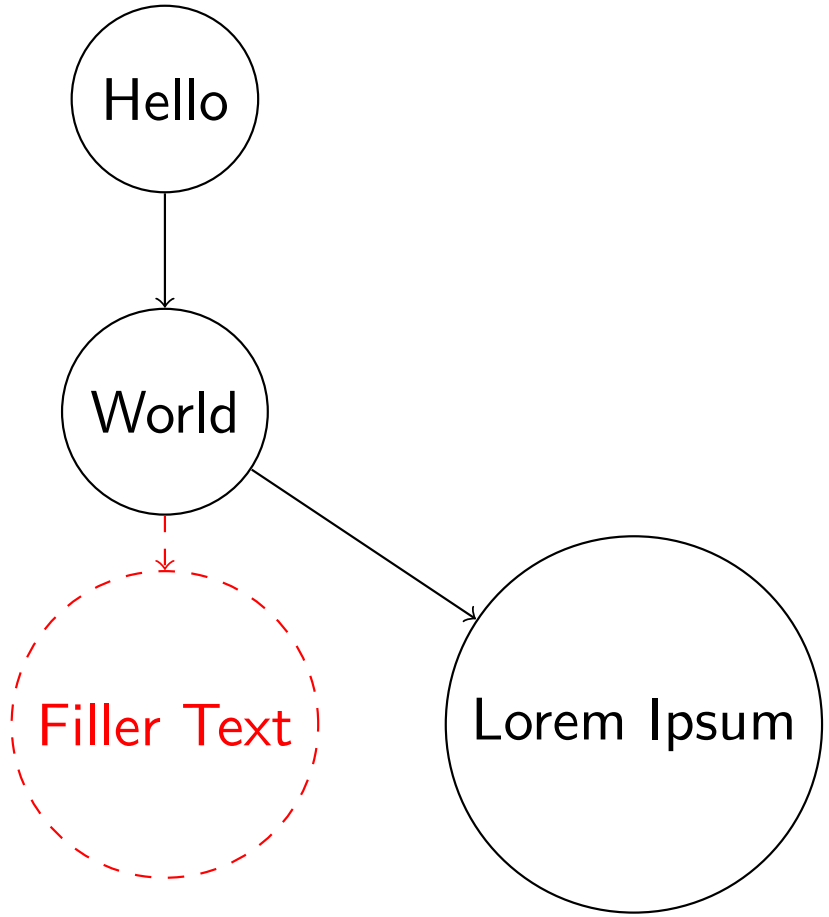
Branches

- From where we were before...
- Type “Lorem Ipsum”
- Realise we want “Filler Text” back
- Press Undo



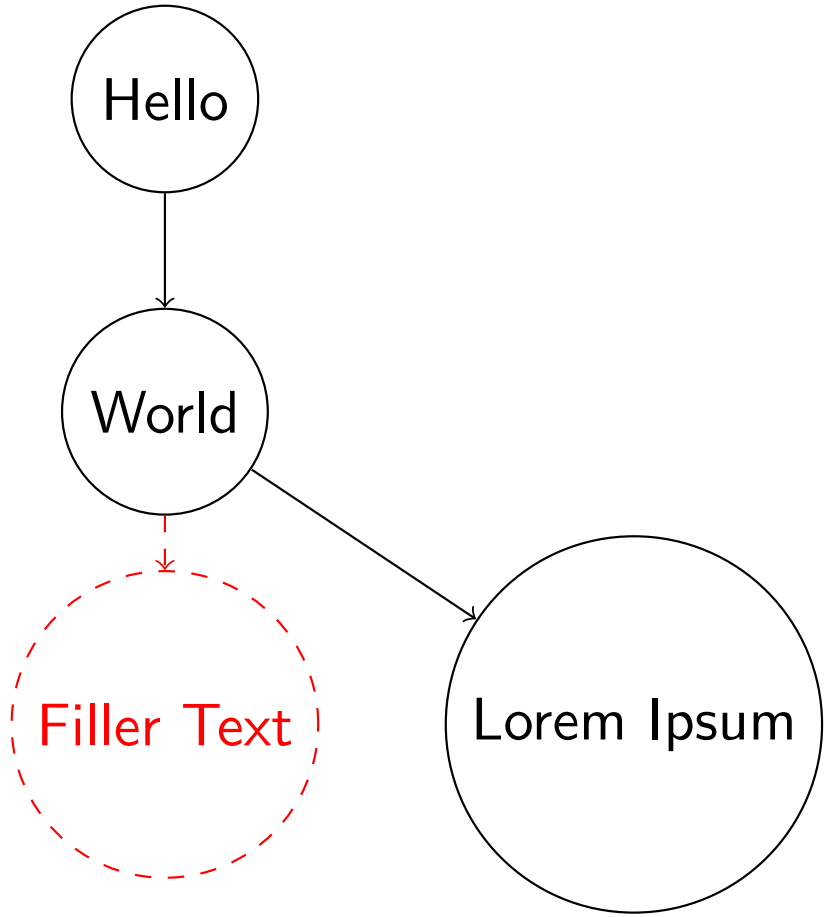
Branches

- From where we were before...
- Type “Lorem Ipsum”
- Realise we want “Filler Text” back
- Press Undo
- Press Redo



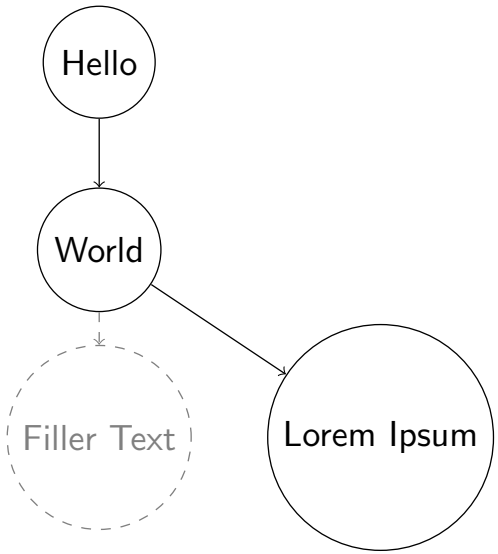
Branches

- From where we were before...
- Type “Lorem Ipsum”
- Realise we want “Filler Text” back
- Press Undo
- Press Redo
- Weep



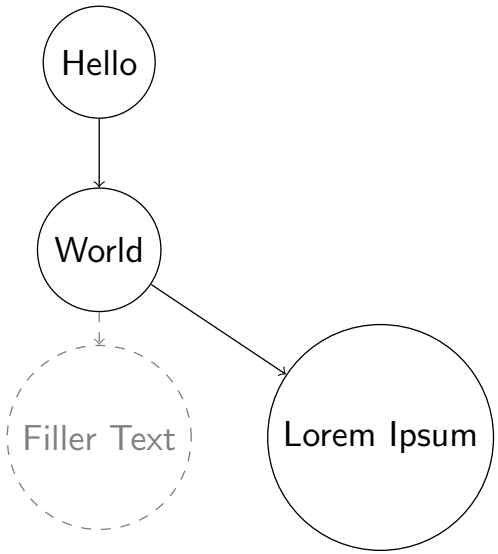
Merges and Collaborative Work

- From where we were before...



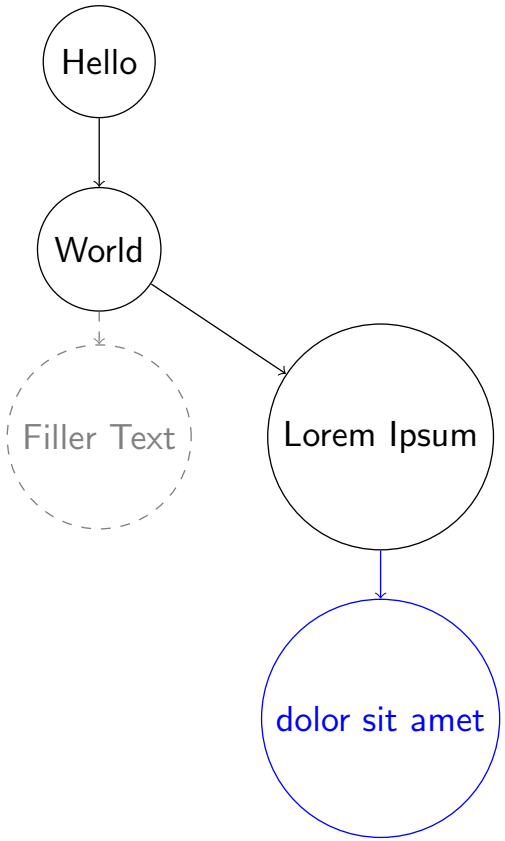
Merges and Collaborative Work

- From where we were before...
- Save it, throw it on a USB, give it to a friend



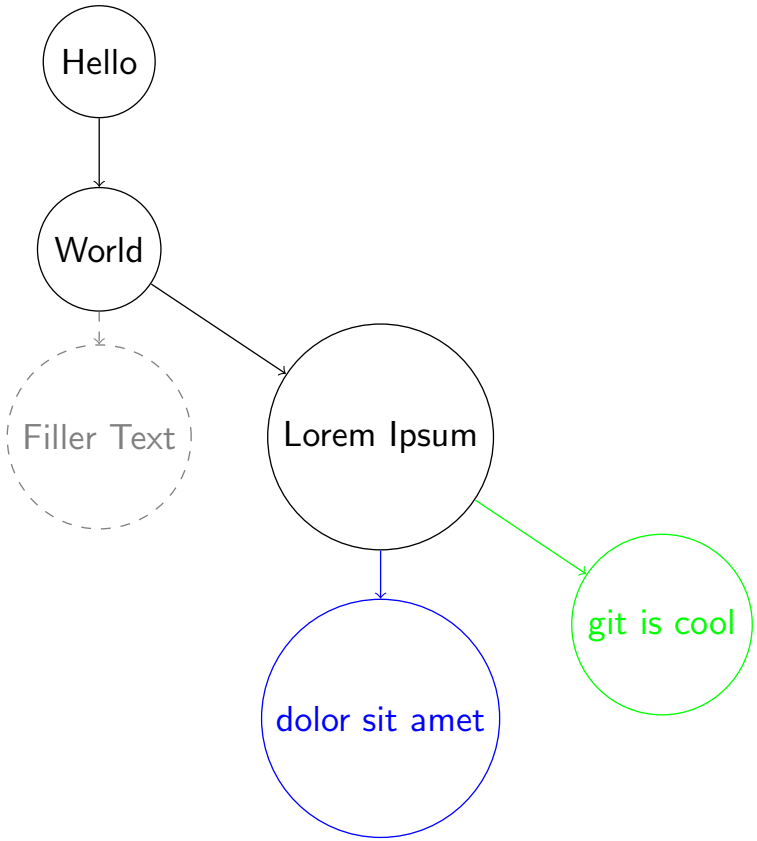
Merges and Collaborative Work

- From where we were before...
- Save it, throw it on a USB, give it to a friend
- **We** write "dolor sit amet" at the **end** of the document



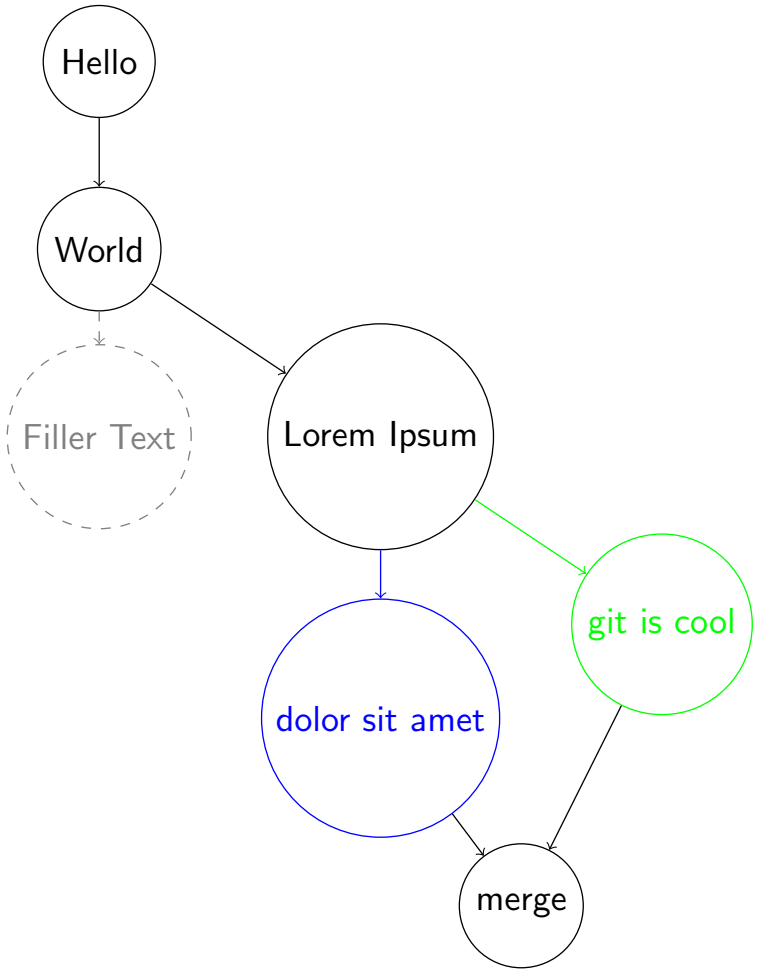
Merges and Collaborative Work

- From where we were before...
- Save it, throw it on a USB, give it to a friend
- **We** write "dolor sit amet" at the **end** of the document
- **They** write "git is cool" at the **top** of the document



Merges and Collaborative Work

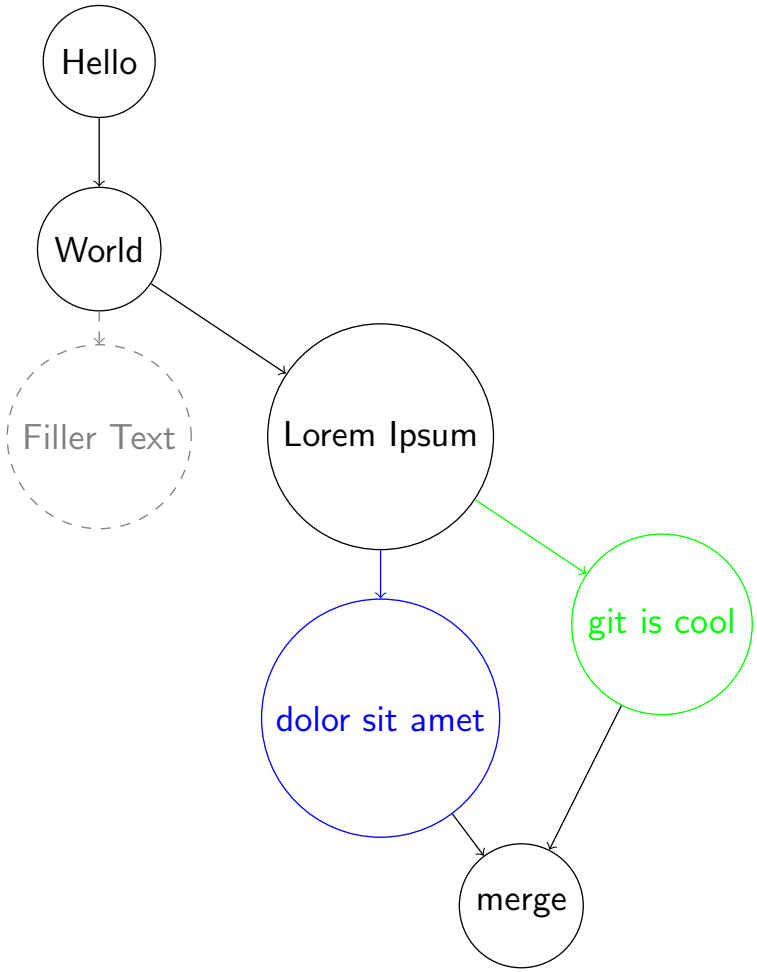
- From where we were before...
- Save it, throw it on a USB, give it to a friend
- **We** write "dolor sit amet" at the **end** of the document
- **They** write "git is cool" at the **top** of the document
- How to combine these?



Outline

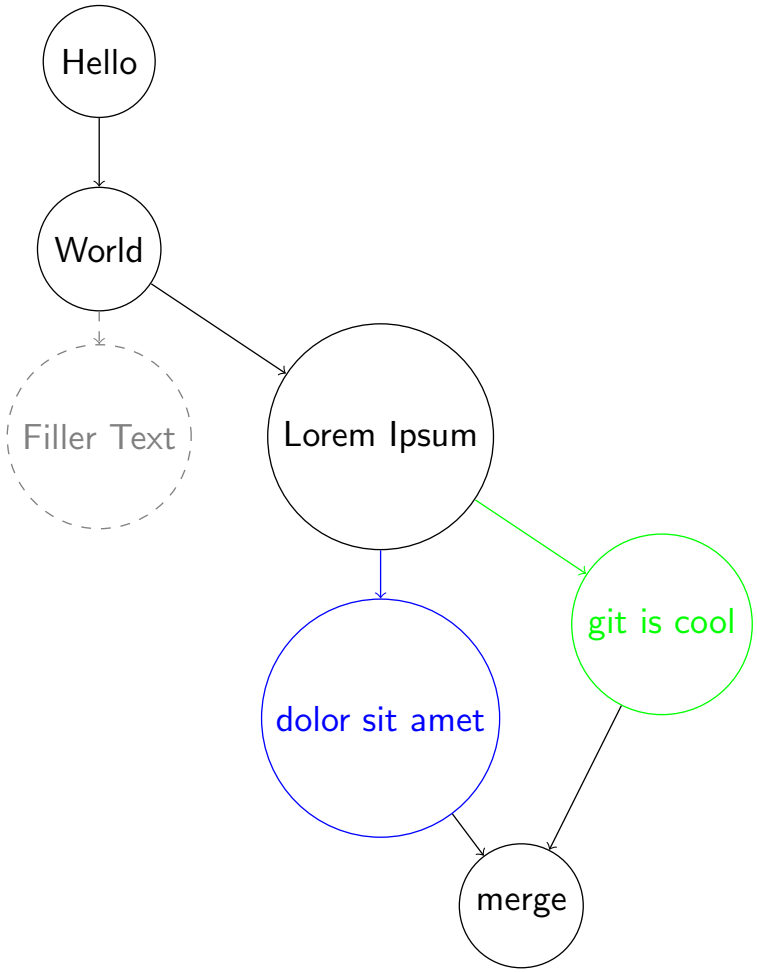
Differences Between Our Live Demo and Git

- Microsoft Word can't do all of these things



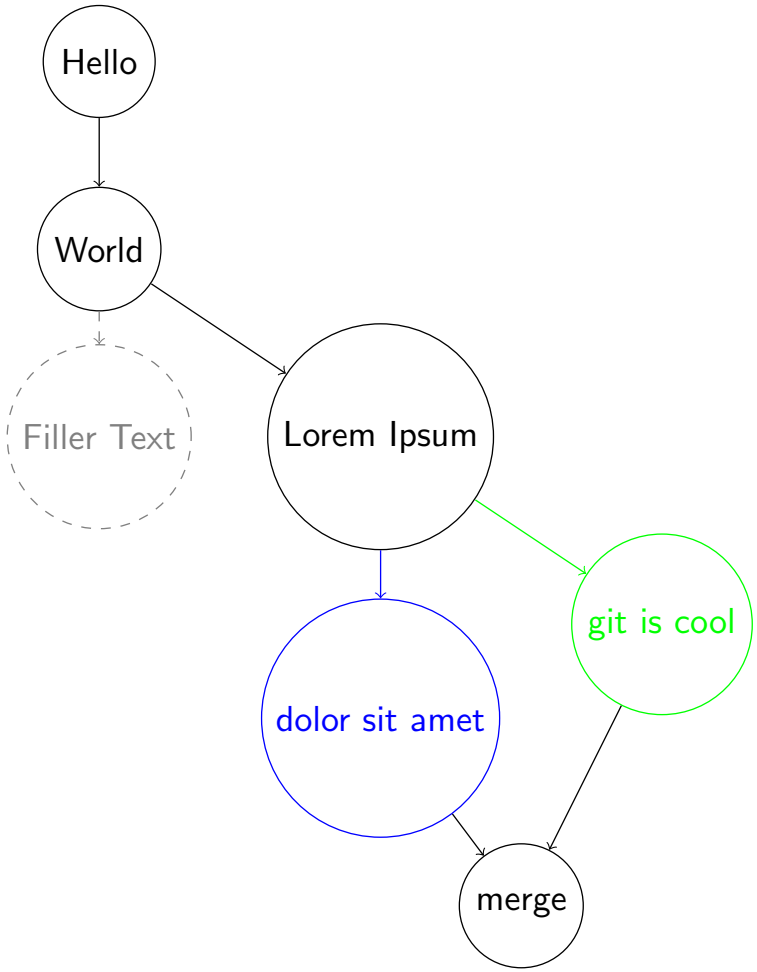
Differences Between Our Live Demo and Git

- Microsoft Word can't do all of these things
- (Git can)



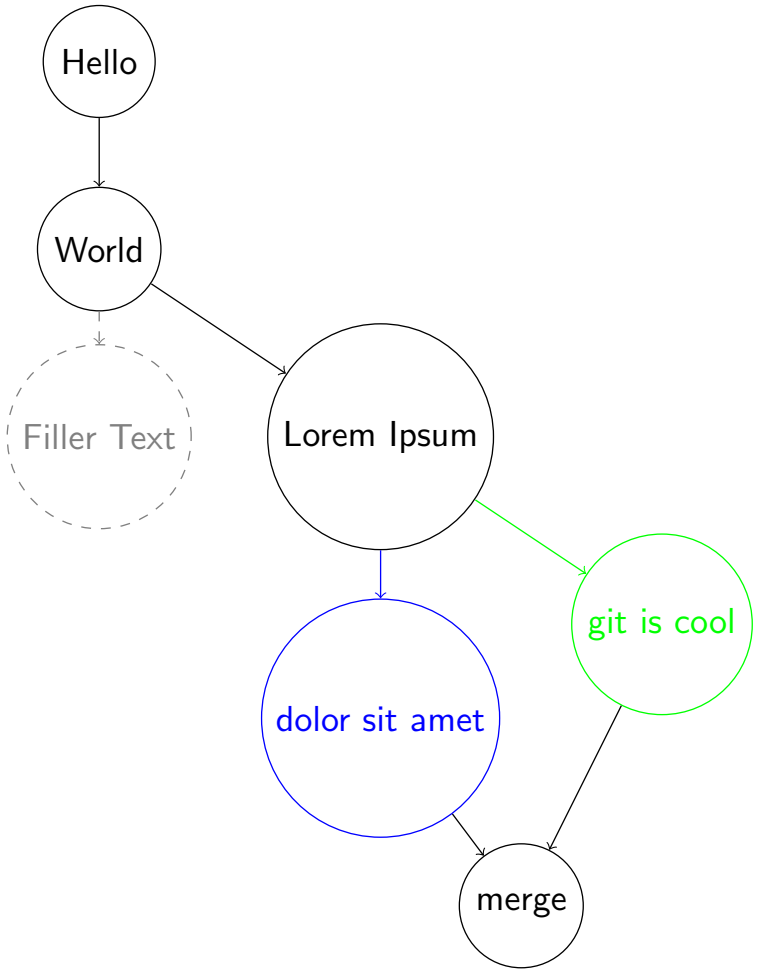
Differences Between Our Live Demo and Git

- Microsoft Word can't do all of these things
- (Git can)
- Undo and Redo track really small changes in a single file



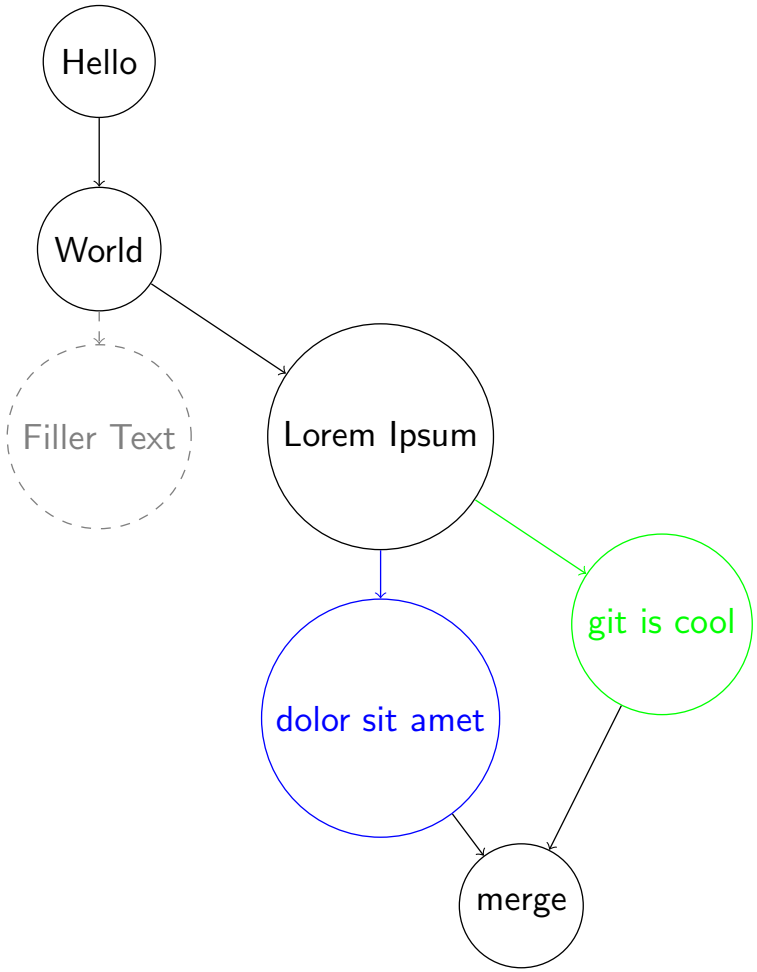
Differences Between Our Live Demo and Git

- Microsoft Word can't do all of these things
- (Git can)
- Undo and Redo track really small changes in a single file
- Git tracks larger changes, in a directory (hereafter, repository)



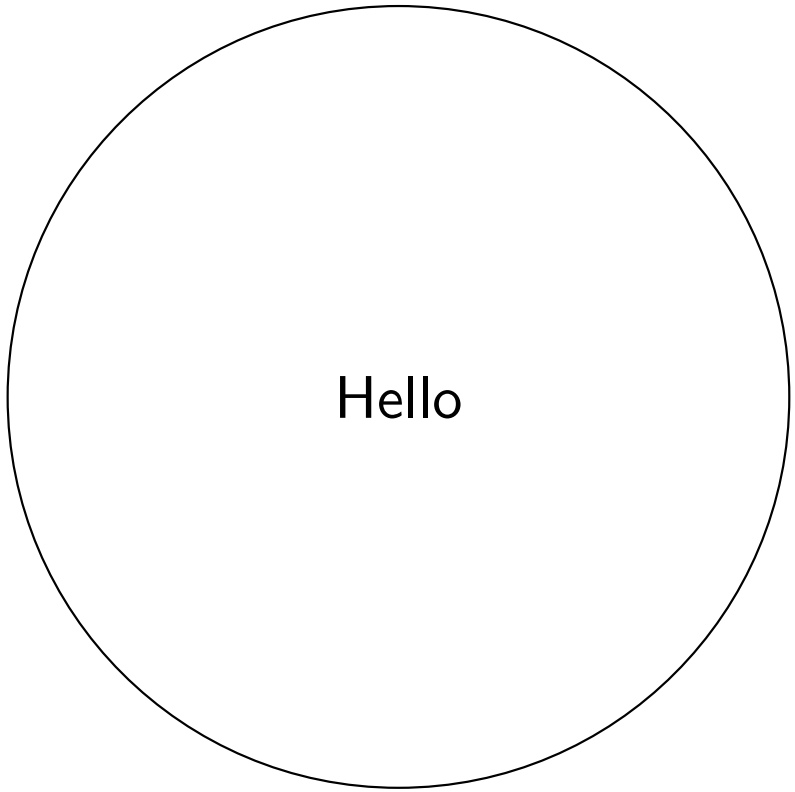
Differences Between Our Live Demo and Git

- Microsoft Word can't do all of these things
- (Git can)
- Undo and Redo track really small changes in a single file
- Git tracks larger changes, in a directory (hereafter, repository)
- ...and more



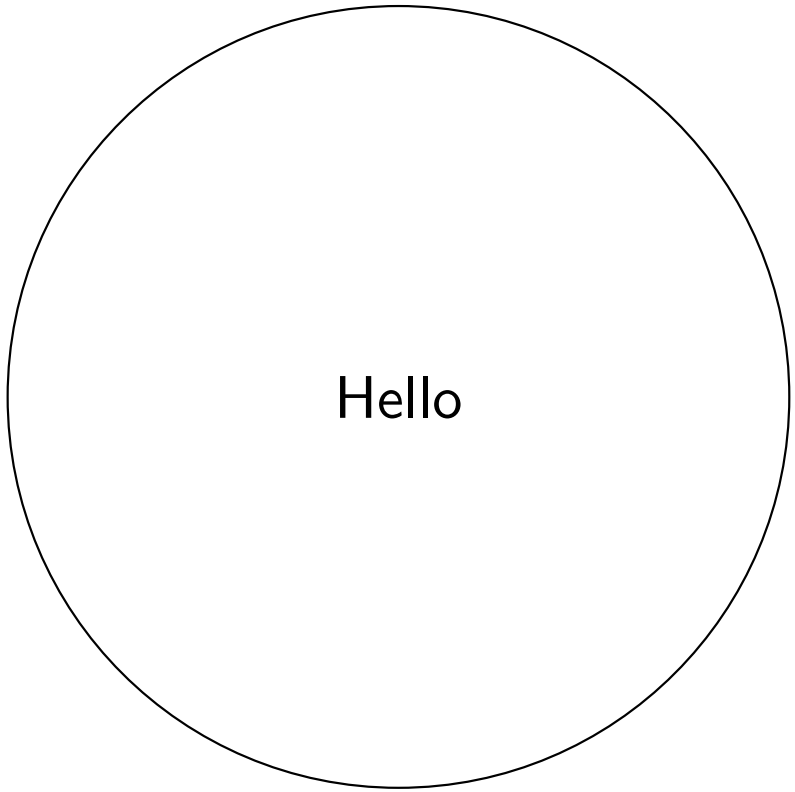
Commits

- The **atomic unit** of git



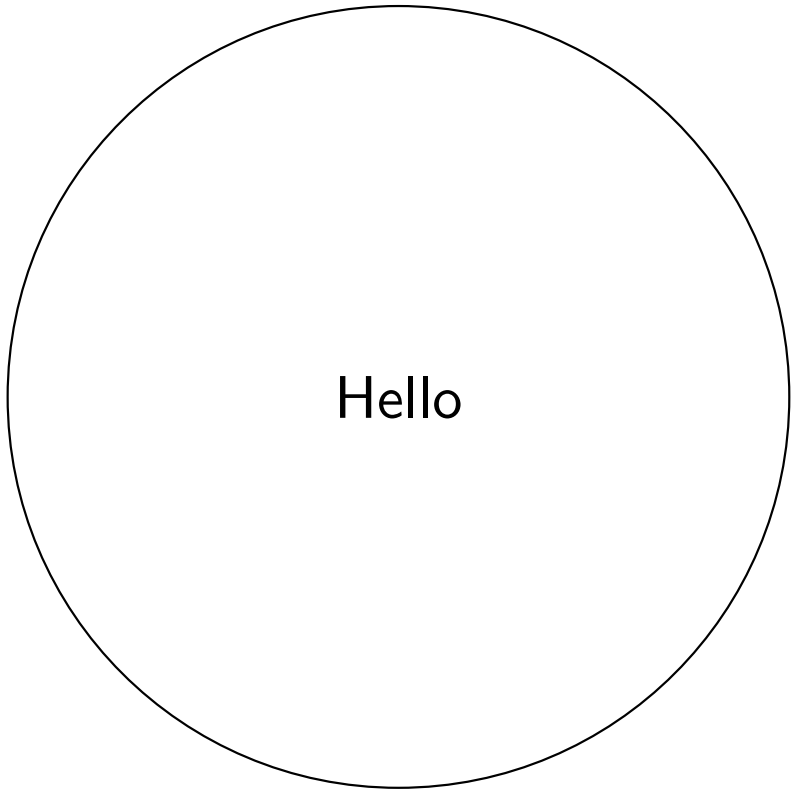
Commits

- The **atomic unit** of git
- You make the commits



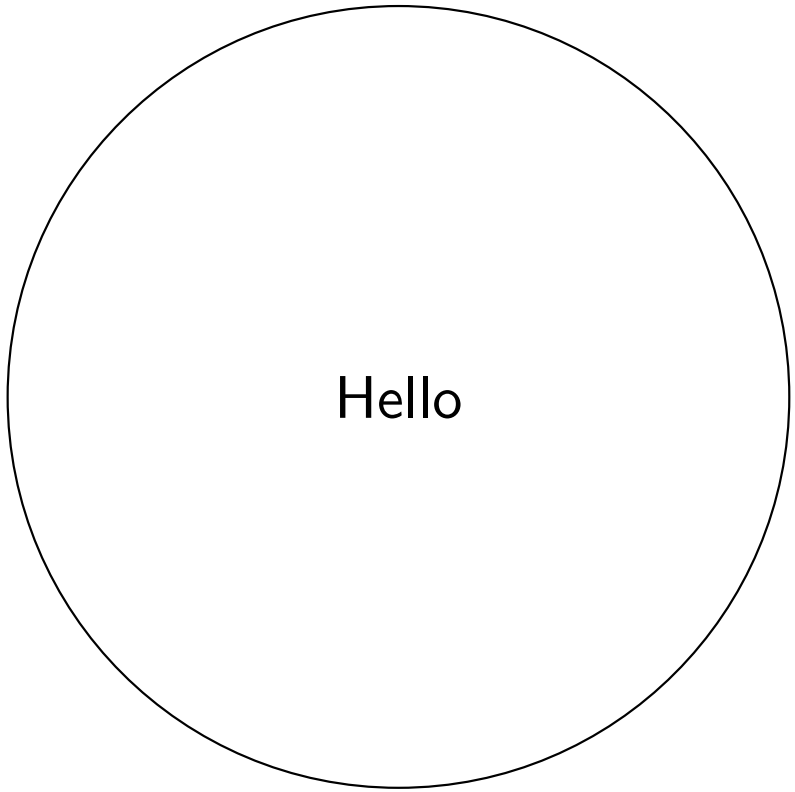
Commits

- The **atomic unit** of git
- You make the commits
- A commit consists of 3 things



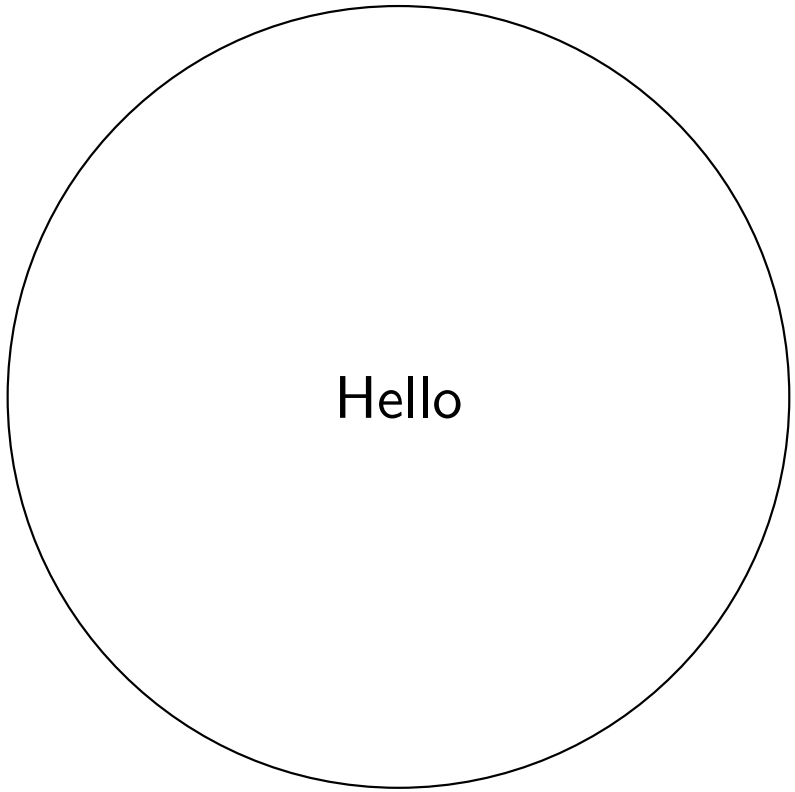
Commits

- The **atomic unit** of git
- You make the commits
- A commit consists of 3 things
- The state of the repo when that commit was made



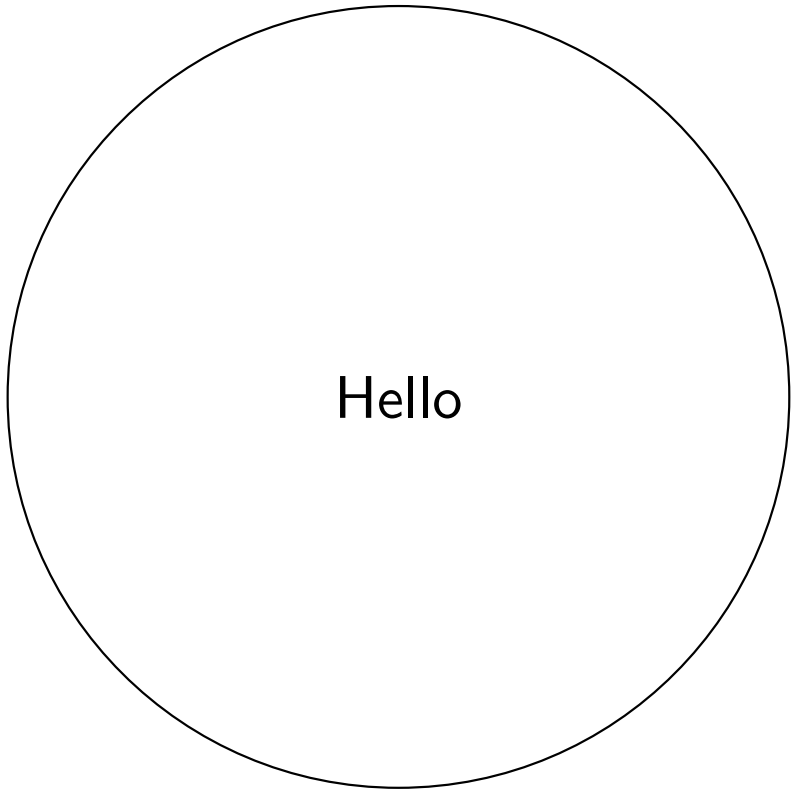
Commits

- The **atomic unit** of git
- You make the commits
- A commit consists of 3 things
- The state of the repo when that commit was made
- What commit(s) came immediately before it



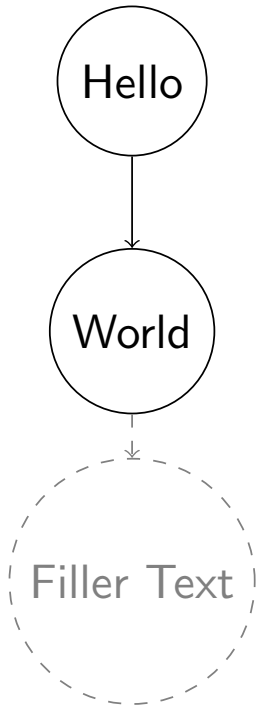
Commits

- The **atomic unit** of git
- You make the commits
- A commit consists of 3 things
- The state of the repo when that commit was made
- What commit(s) came immediately before it
- Metadata (timestamp, author, cryptographic hash)



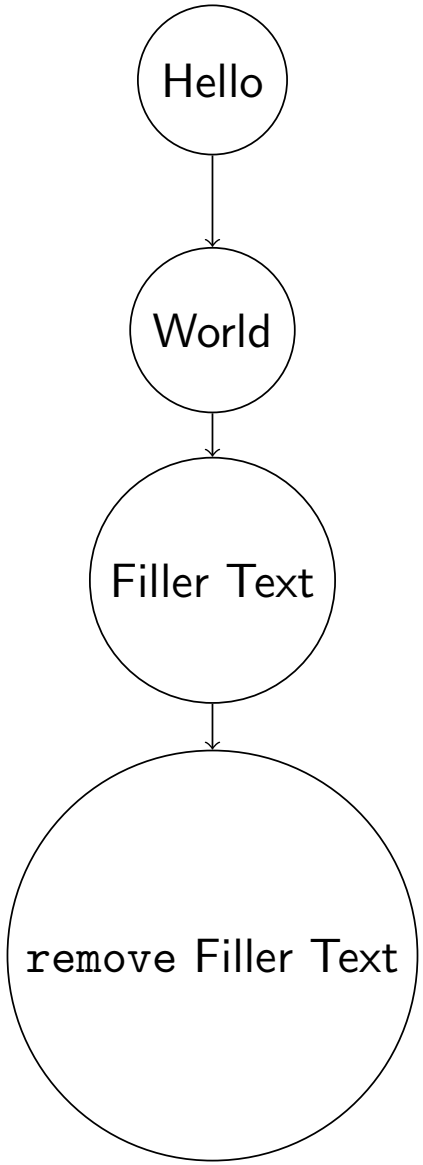
Reverts

- Expressed in terms of commits



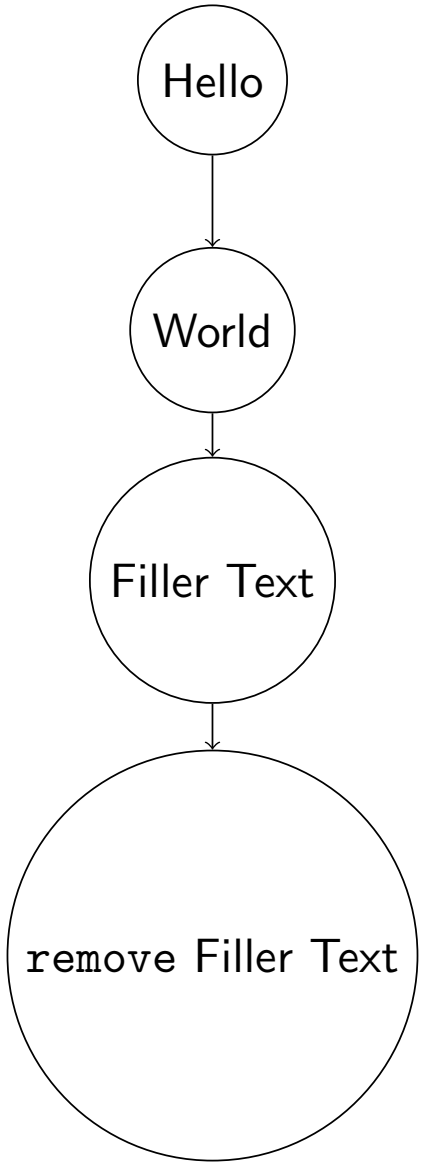
Reverts

- Expressed in terms of commits
- Only appends to the history



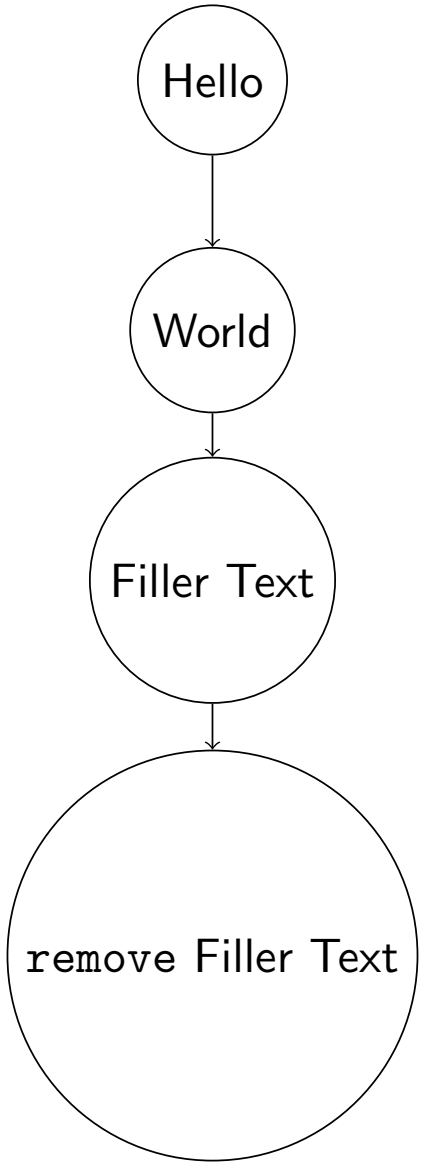
Reverts

- Expressed in terms of commits
- Only appends to the history
- Can revert multiple commits at once



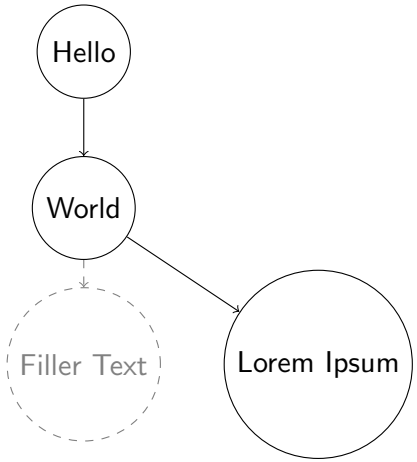
Reverts

- Expressed in terms of commits
- Only appends to the history
- Can revert multiple commits at once
- Don't have to revert the most recent commit



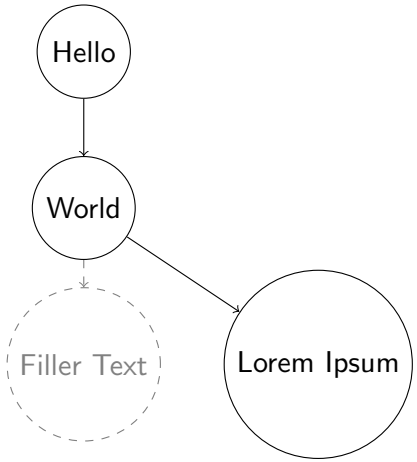
Branches and Merges

- From our previous branches slide



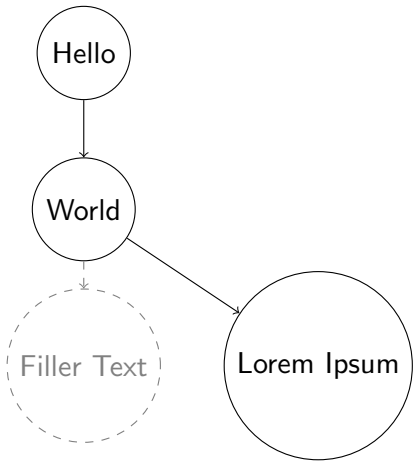
Branches and Merges

- From our previous branches slide
- Git is able to store different versions of your history simultaneously



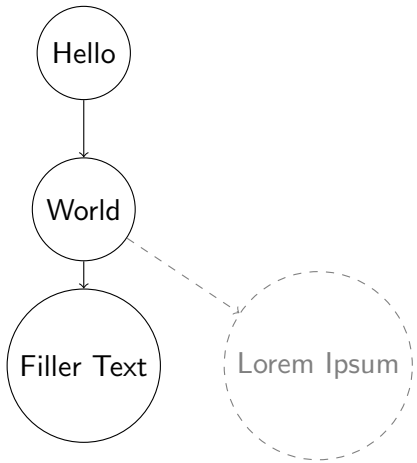
Branches and Merges

- From our previous branches slide
- Git is able to store different versions of your history simultaneously
- You choose which one you want to **check out** and work on



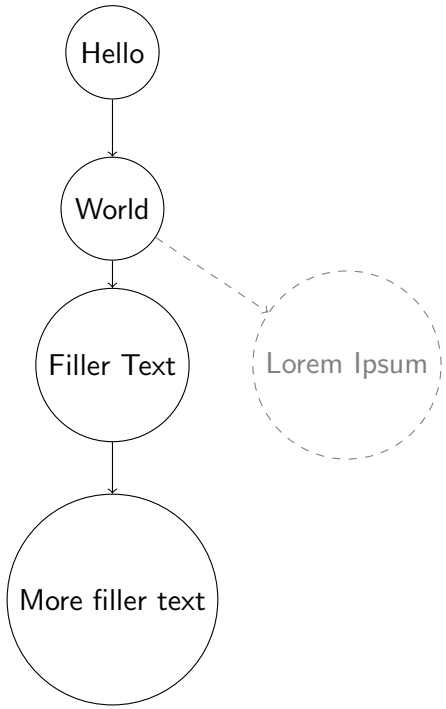
Branches and Merges

- From our previous branches slide
- Git is able to store different versions of your history simultaneously
- You choose which one you want to **check out** and work on



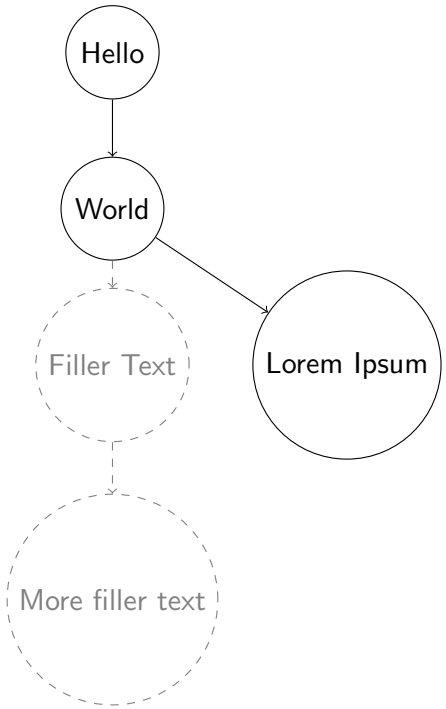
Branches and Merges

- From our previous branches slide
- Git is able to store different versions of your history simultaneously
- You choose which one you want to **check out** and work on



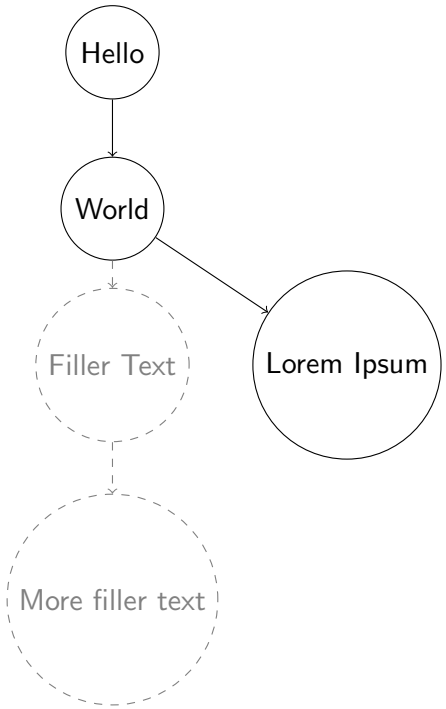
Branches and Merges

- From our previous branches slide
- Git is able to store different versions of your history simultaneously
- You choose which one you want to **check out** and work on



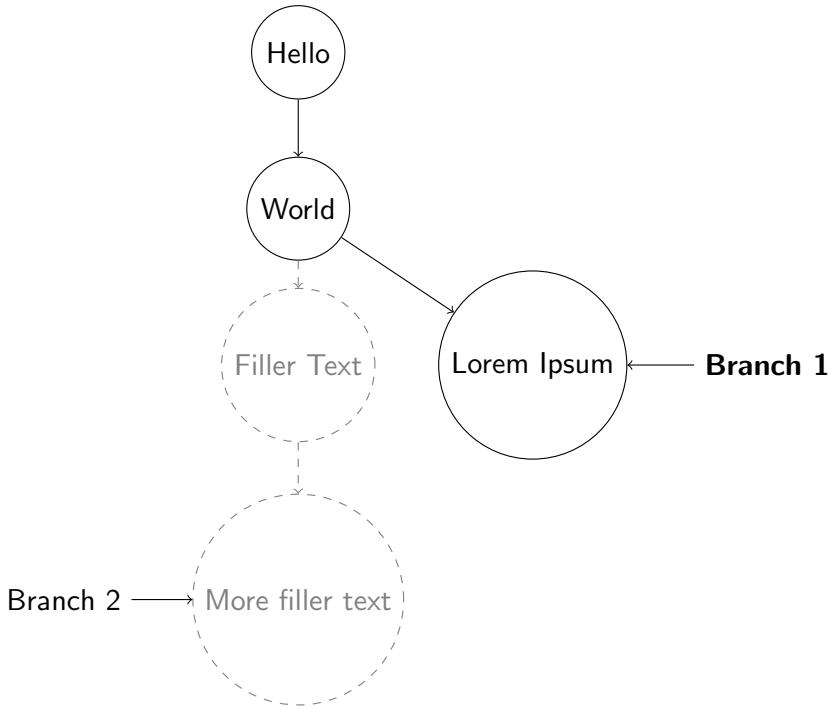
Branches and Merges

- From our previous branches slide
- Git is able to store different versions of your history simultaneously
- You choose which one you want to **check out** and work on
- A branch in git is just a pointer to somewhere on this graph



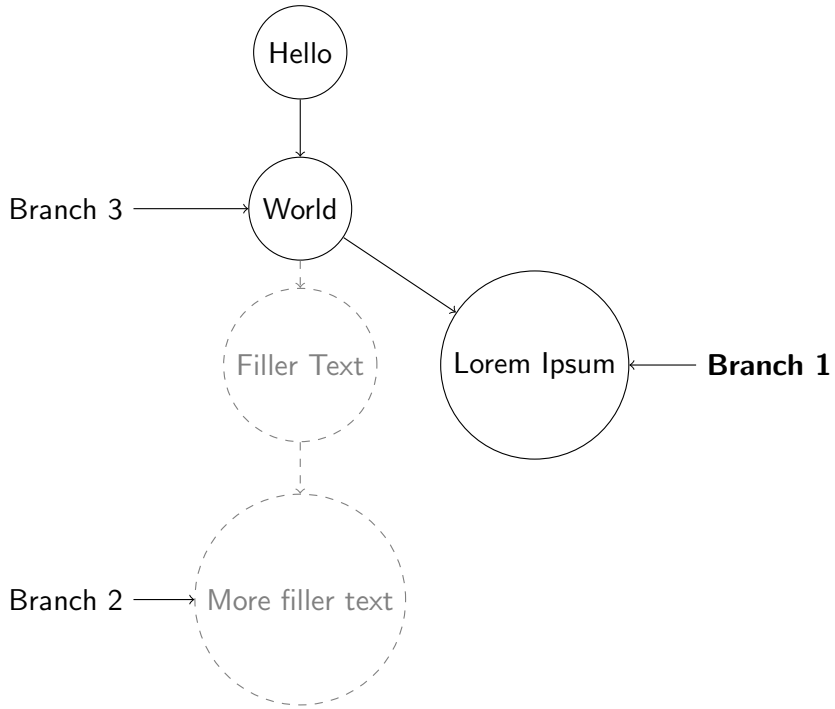
Branches and Merges

- From our previous branches slide
- Git is able to store different versions of your history simultaneously
- You choose which one you want to **check out** and work on
- A branch in git is just a pointer to somewhere on this graph



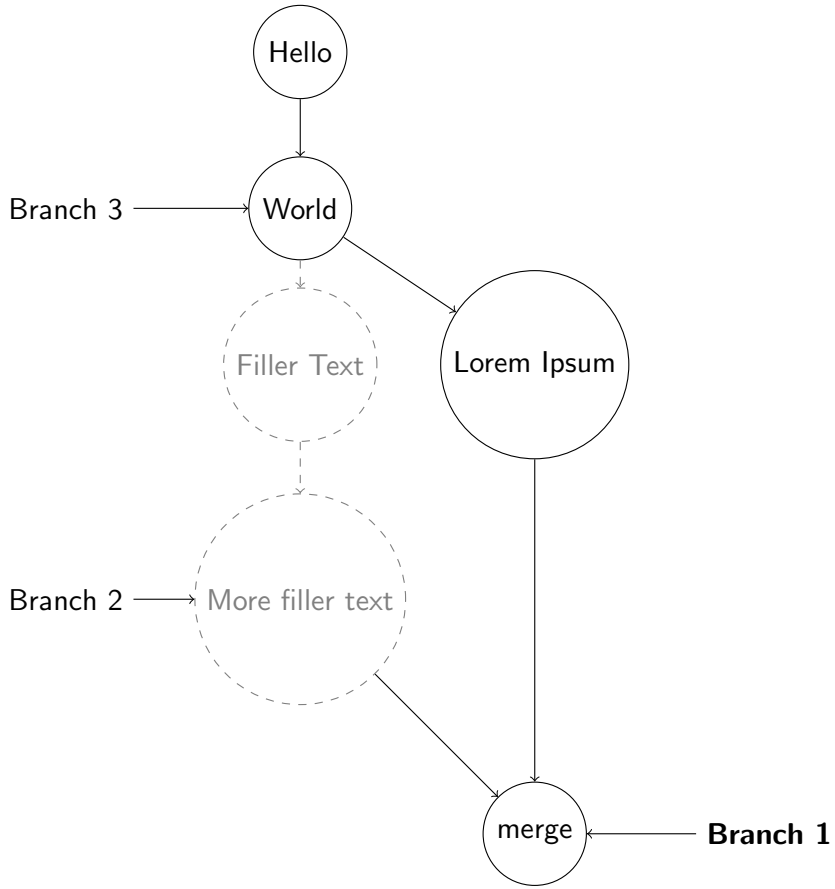
Branches and Merges

- From our previous branches slide
- Git is able to store different versions of your history simultaneously
- You choose which one you want to **check out** and work on
- A branch in git is just a pointer to somewhere on this graph



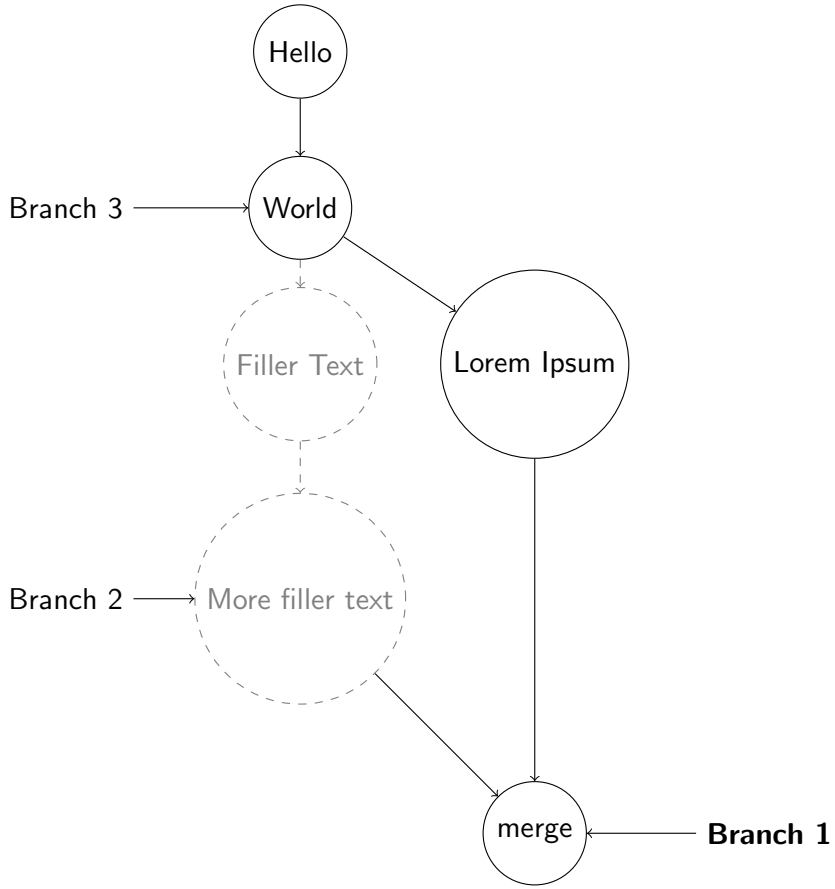
Branches and Merges

- From our previous branches slide
- Git is able to store different versions of your history simultaneously
- You choose which one you want to **check out** and work on
- A branch in git is just a pointer to somewhere on this graph
- As promised, git can take two branches and merge them



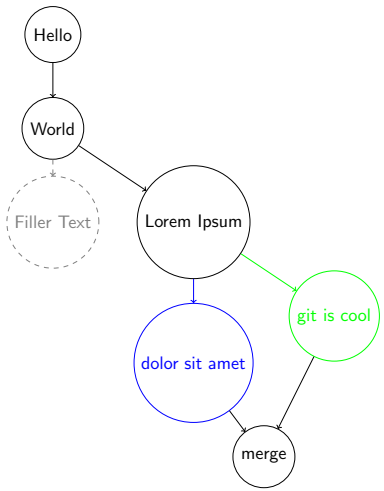
Branches and Merges

- From our previous branches slide
- Git is able to store different versions of your history simultaneously
- You choose which one you want to **check out** and work on
- A branch in git is just a pointer to somewhere on this graph
- As promised, git can take two branches and merge them
- But how? (explanation)



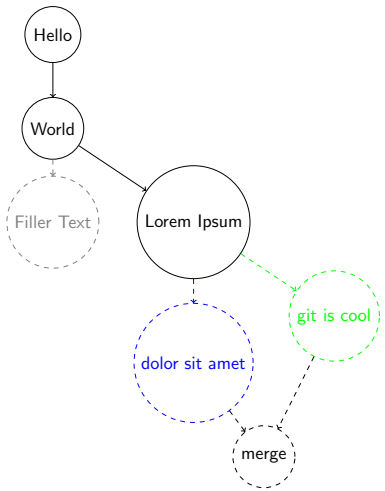
Collaborative Work

- From our previous collaboration slide...



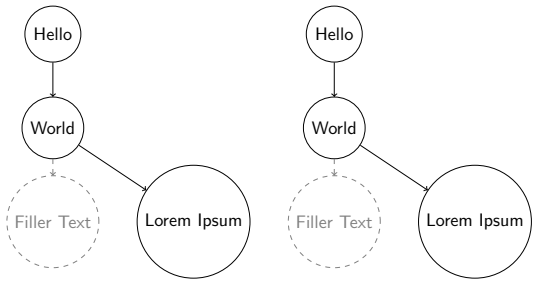
Collaborative Work

- From our previous collaboration slide...
- Going back in time...



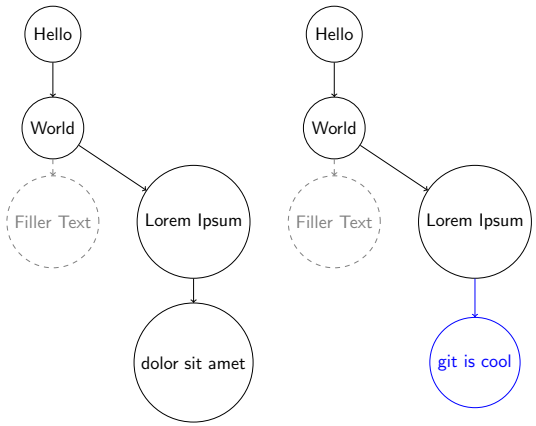
Collaborative Work

- From our previous collaboration slide...
- Going back in time...
- Git is a **distributed** version control system



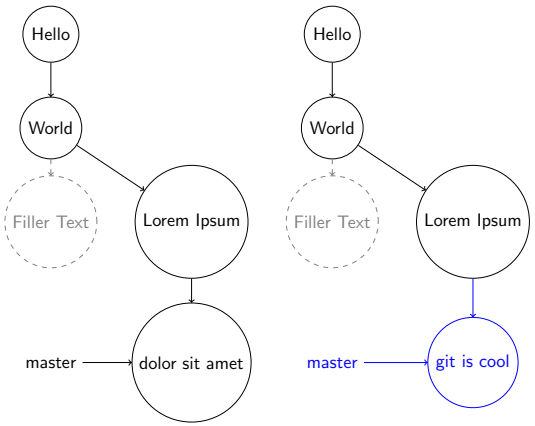
Collaborative Work

- From our previous collaboration slide...
- Going back in time...
- Git is a **distributed** version control system
- Then you can each do your own work



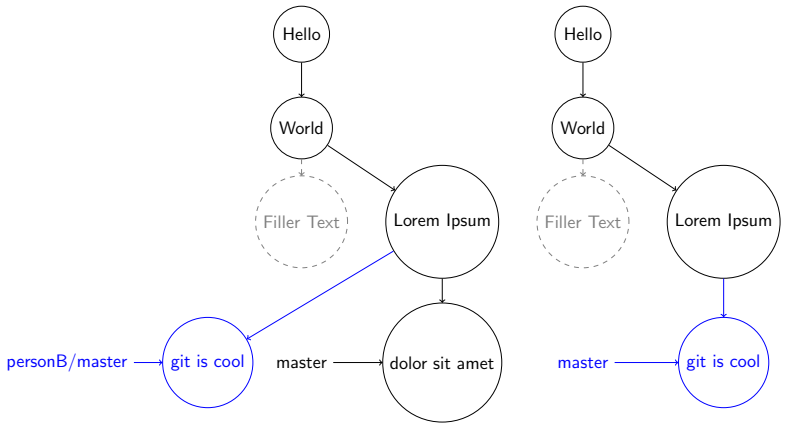
Collaborative Work

- From our previous collaboration slide...
- Going back in time...
- Git is a **distributed** version control system
- Then you can each do your own work
- How to merge your work with someone else's



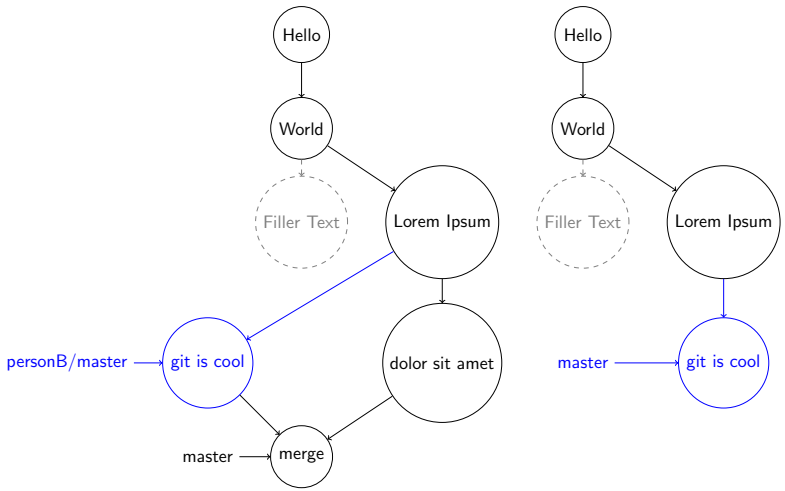
Collaborative Work

- From our previous collaboration slide...
- Going back in time...
- Git is a **distributed** version control system
- Then you can each do your own work
- How to merge your work with someone else's
- Git allows you to **fetch** someone else's history into your history



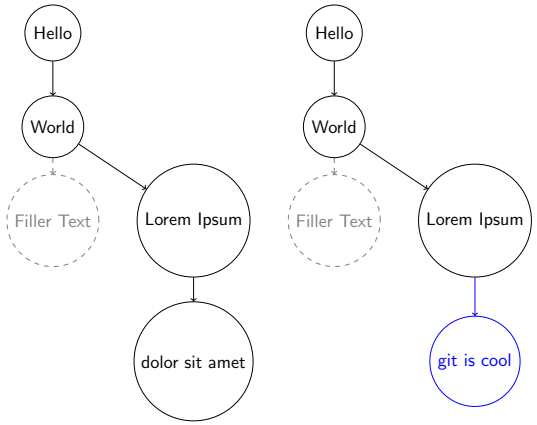
Collaborative Work

- From our previous collaboration slide...
- Going back in time...
- Git is a **distributed** version control system
- Then you can each do your own work
- How to merge your work with someone else's
- Git allows you to **fetch** someone else's history into your history
- From there, it's just another branch that you can merge



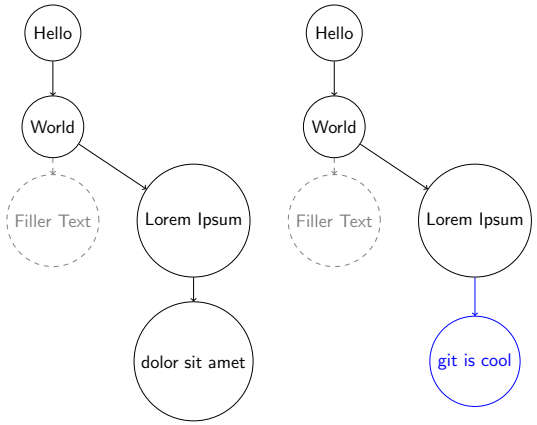
Collaborative Work II

- From our previous collaborative work slide



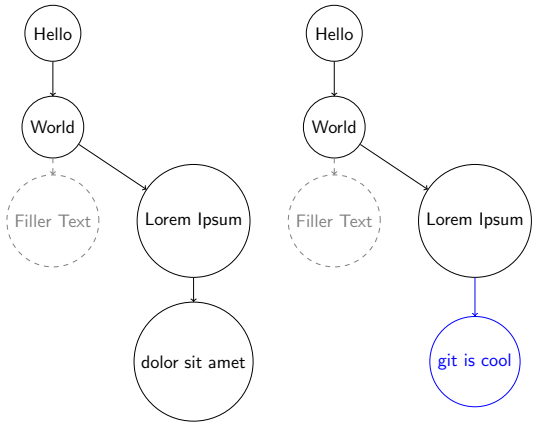
Collaborative Work II

- From our previous collaborative work slide
- Fetch-only collaboration is painful



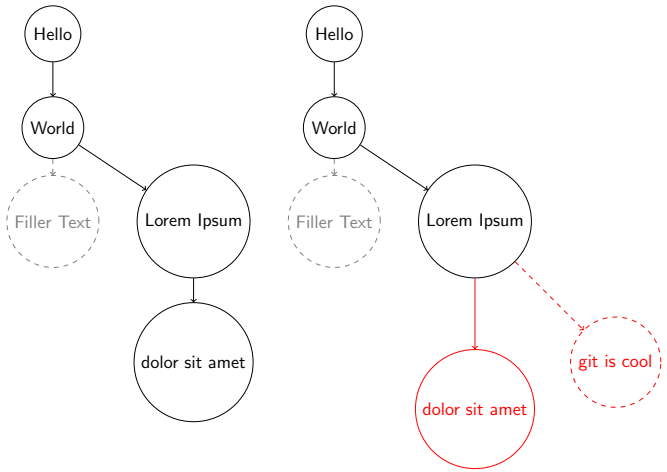
Collaborative Work II

- From our previous collaborative work slide
- Fetch-only collaboration is painful
- Need a way to **push** your changes to the other computer



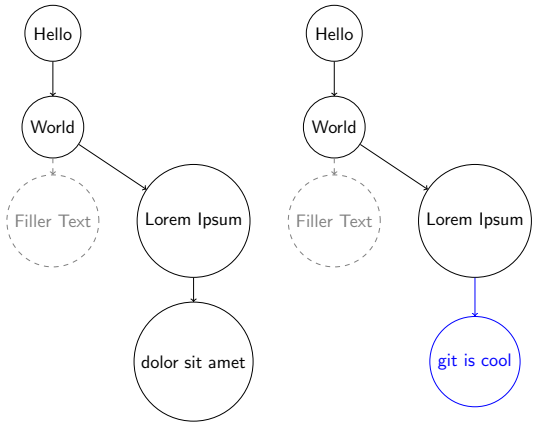
Collaborative Work II

- From our previous collaborative work slide
- Fetch-only collaboration is painful
- Need a way to **push** your changes to the other computer



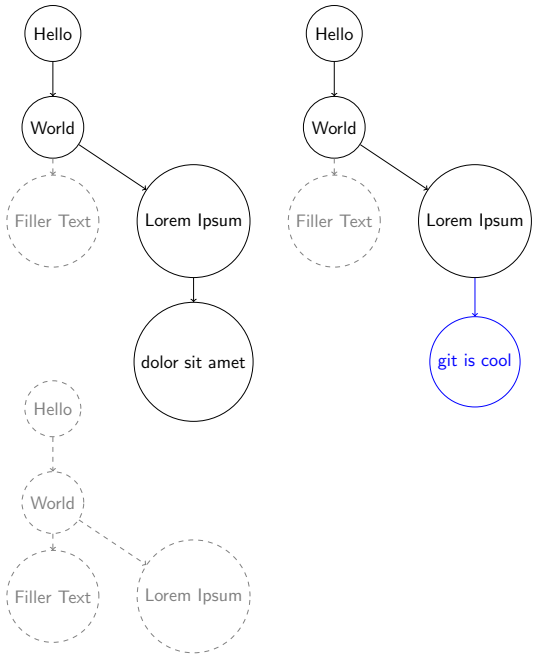
Collaborative Work II

- From our previous collaborative work slide
- Fetch-only collaboration is painful
- Need a way to **push** your changes to the other computer



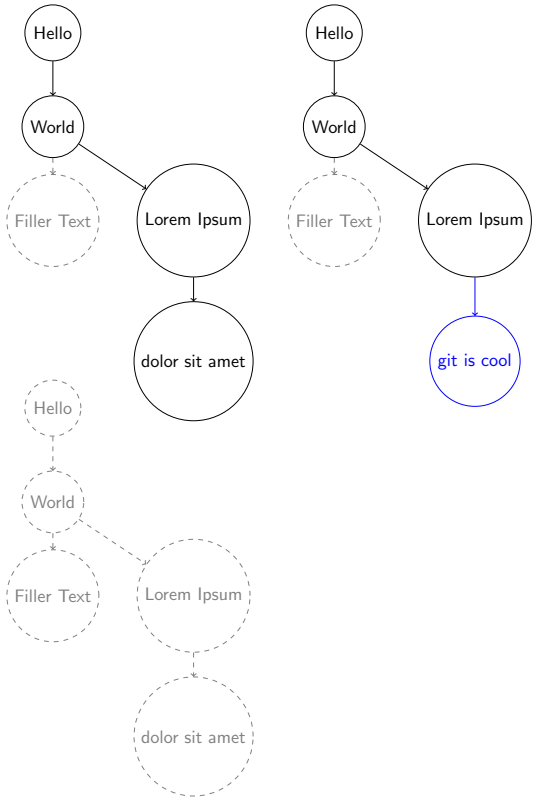
Collaborative Work II

- From our previous collaborative work slide
- Fetch-only collaboration is painful
- Need a way to **push** your changes to the other computer
- Enter git bare mode (for servers)



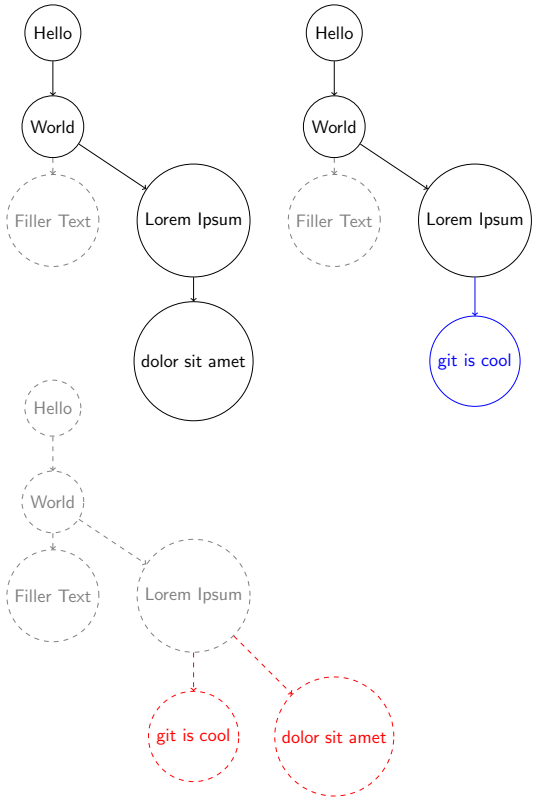
Collaborative Work II

- From our previous collaborative work slide
- Fetch-only collaboration is painful
- Need a way to **push** your changes to the other computer
- Enter git bare mode (for servers)
- You push to the server



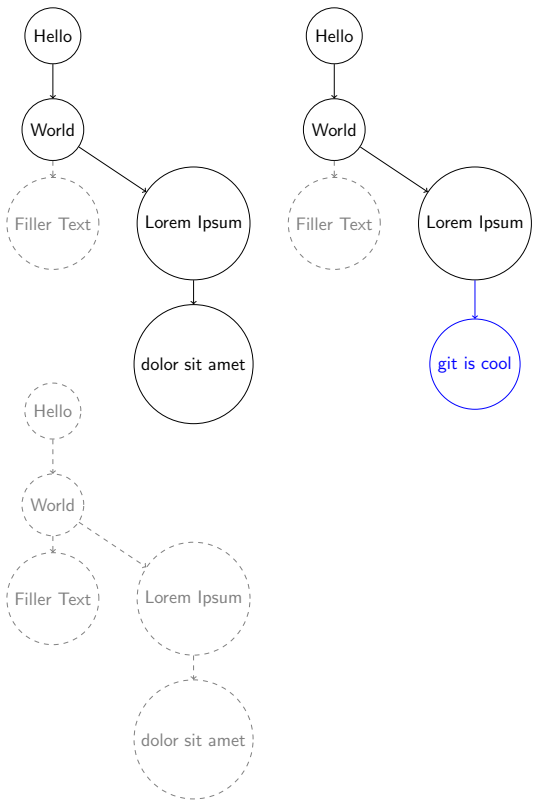
Collaborative Work II

- From our previous collaborative work slide
- Fetch-only collaboration is painful
- Need a way to **push** your changes to the other computer
- Enter git bare mode (for servers)
- You push to the server



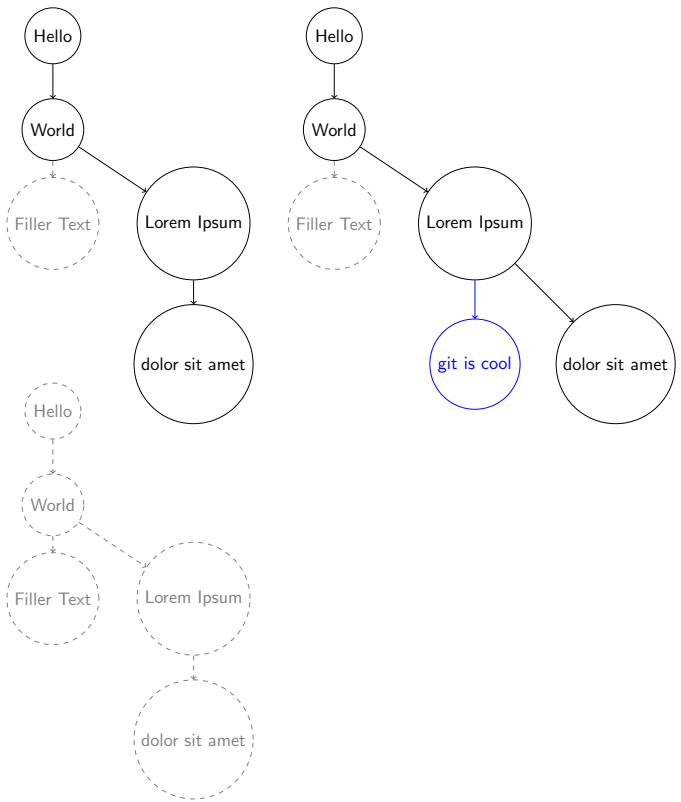
Collaborative Work II

- From our previous collaborative work slide
- Fetch-only collaboration is painful
- Need a way to **push** your changes to the other computer
- Enter git bare mode (for servers)
- You push to the server
- The server isn't smart enough to handle merges, so can only push fast-forwards



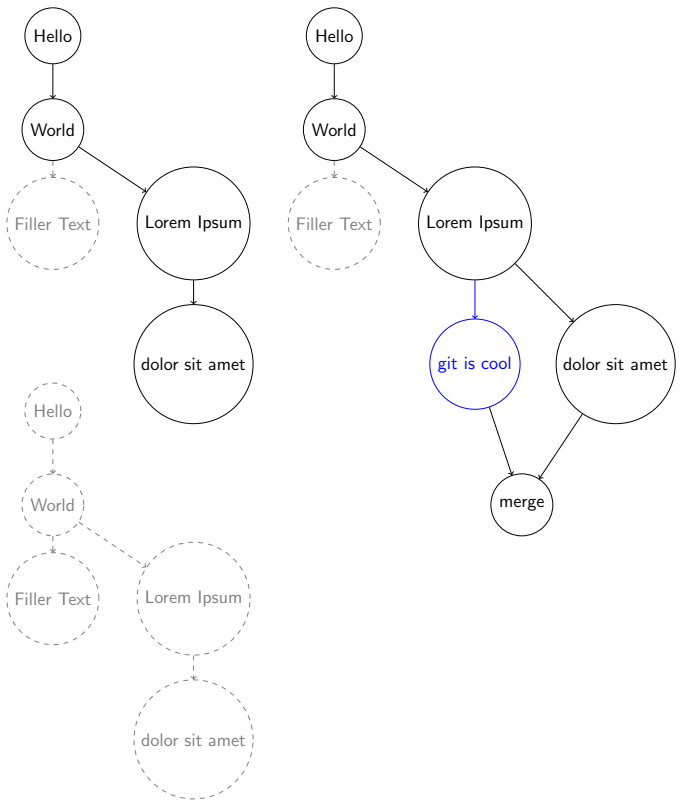
Collaborative Work II

- From our previous collaborative work slide
- Fetch-only collaboration is painful
- Need a way to **push** your changes to the other computer
- Enter git bare mode (for servers)
- You push to the server
- The server isn't smart enough to handle merges, so can only push fast-forwards



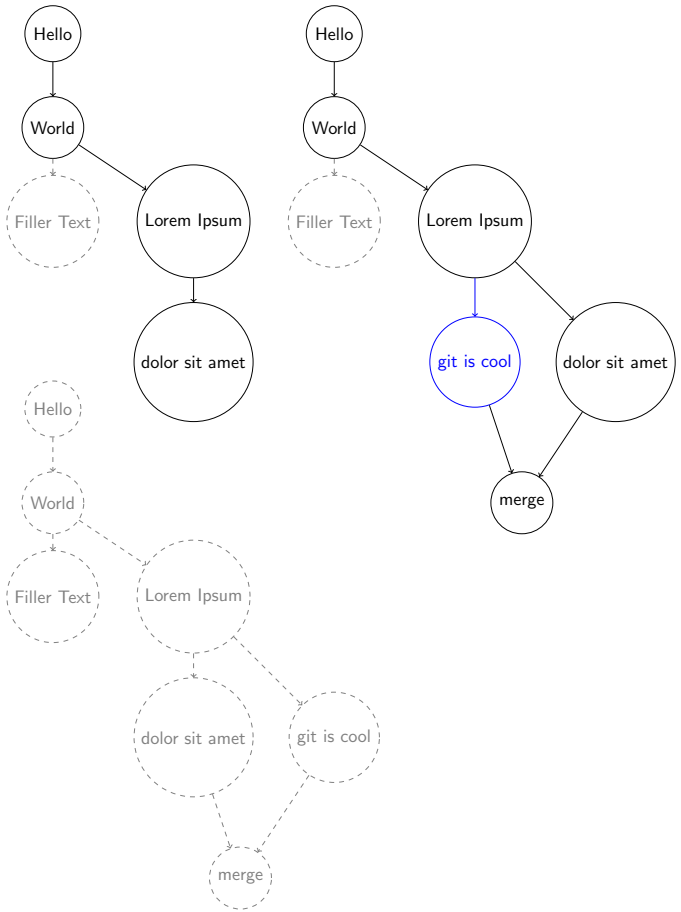
Collaborative Work II

- From our previous collaborative work slide
- Fetch-only collaboration is painful
- Need a way to **push** your changes to the other computer
- Enter git bare mode (for servers)
- You push to the server
- The server isn't smart enough to handle merges, so can only push fast-forwards



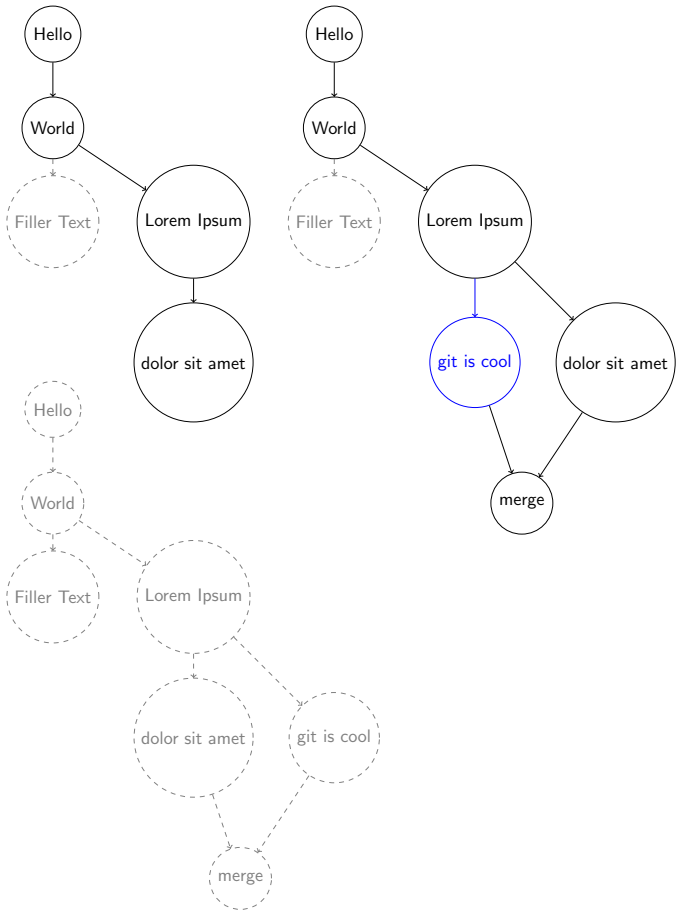
Collaborative Work II

- From our previous collaborative work slide
- Fetch-only collaboration is painful
- Need a way to **push** your changes to the other computer
- Enter git bare mode (for servers)
- You push to the server
- The server isn't smart enough to handle merges, so can only push fast-forwards



Collaborative Work II

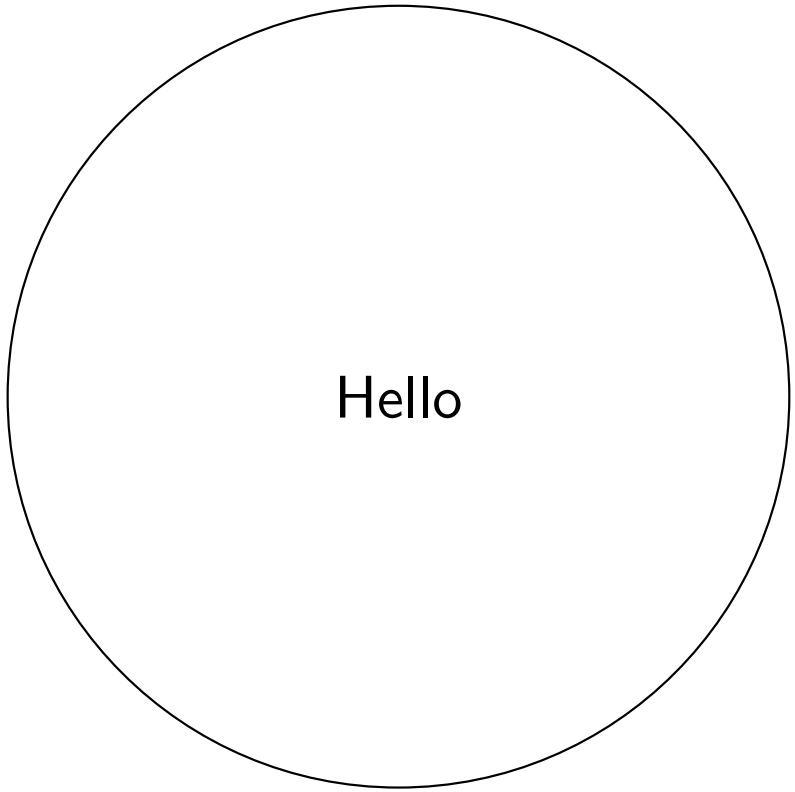
- From our previous collaborative work slide
- Fetch-only collaboration is painful
- Need a way to **push** your changes to the other computer
- Enter git bare mode (for servers)
- You push to the server
- The server isn't smart enough to handle merges, so can only push fast-forwards
- Common servers Github, Bitbucket, Gitlab, Self-hosted



Outline

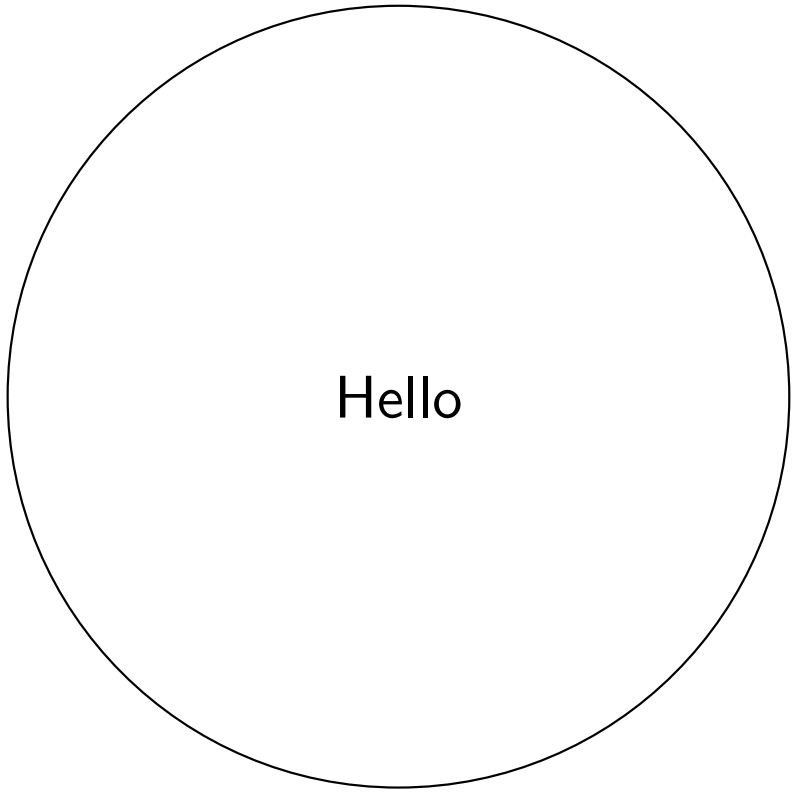
Commits

- From before, the **atomic unit** of git



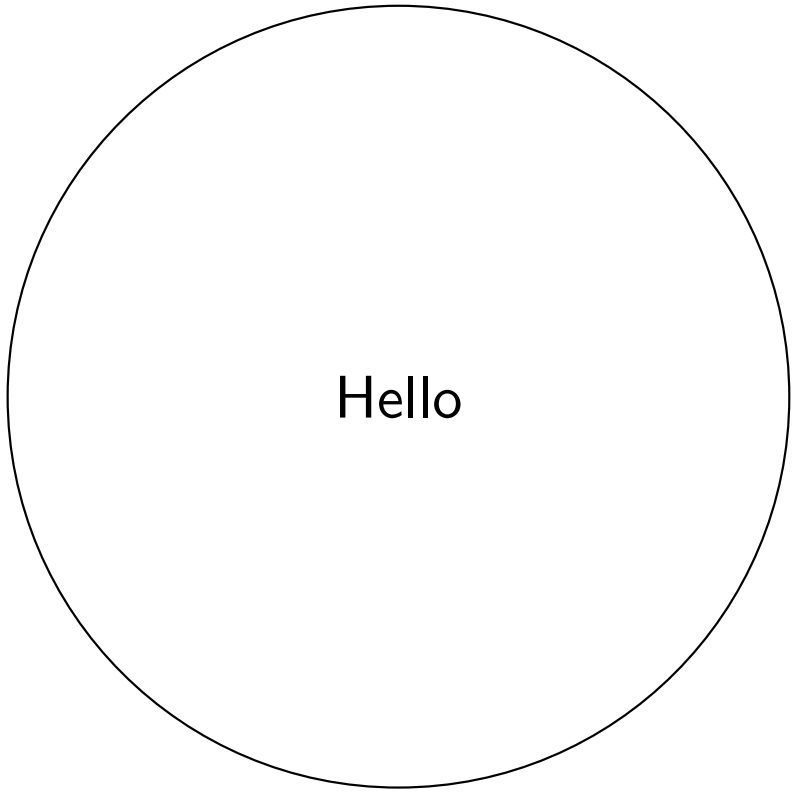
Commits

- From before, the **atomic unit** of git
- First need to put things into a staging area



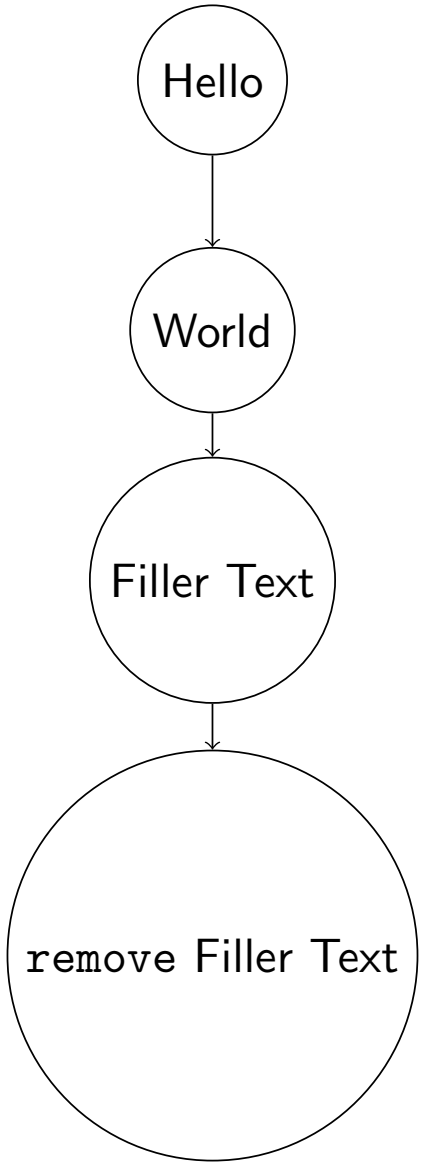
Commits

- From before, the **atomic unit** of git
- First need to put things into a staging area
- Git can show you all your commits in a log



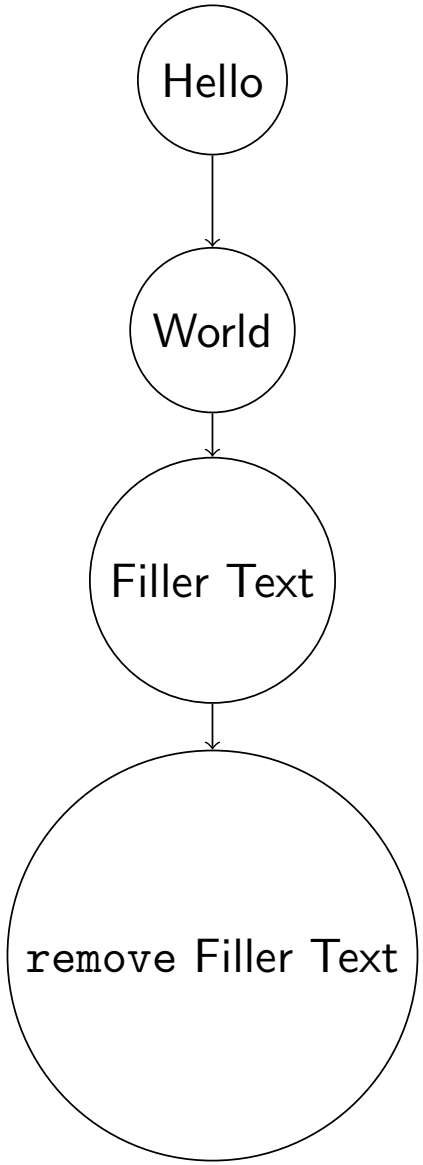
Reverts

- From before, append to history the invert of a previous commit



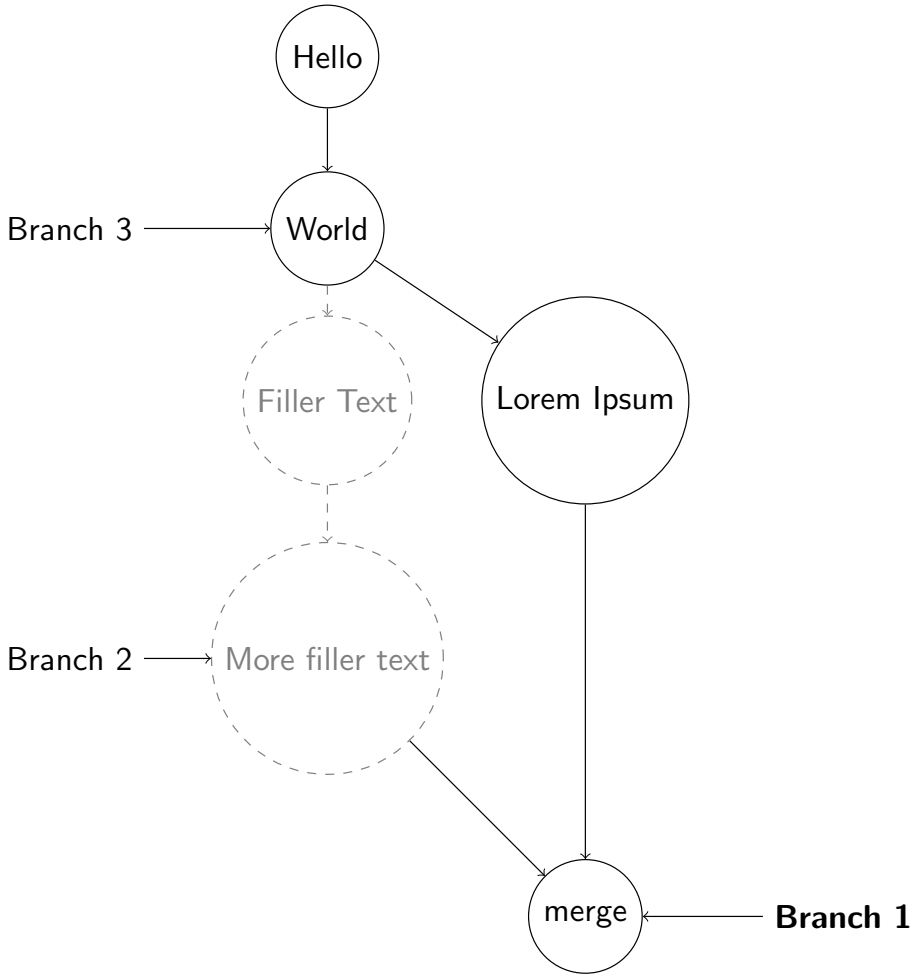
Reverts

- From before, append to history the invert of a previous commit
- Can revert any commit in your history



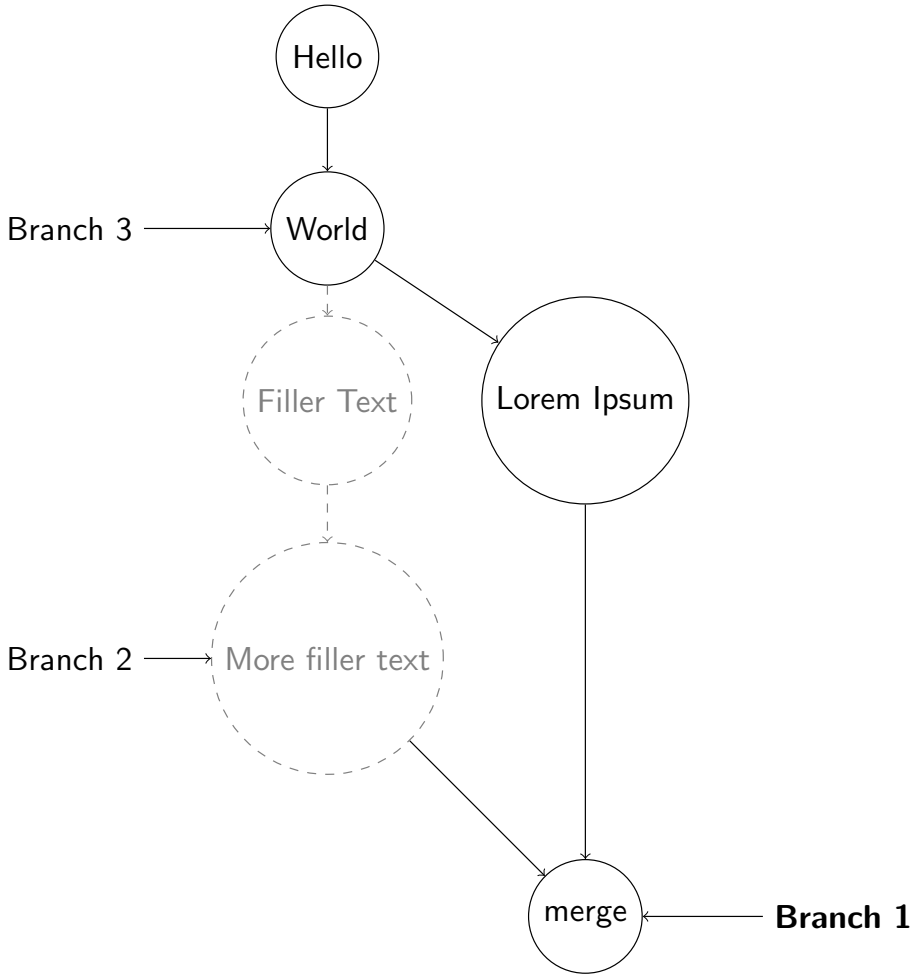
Branches and Merges

- From our previous branching slide...



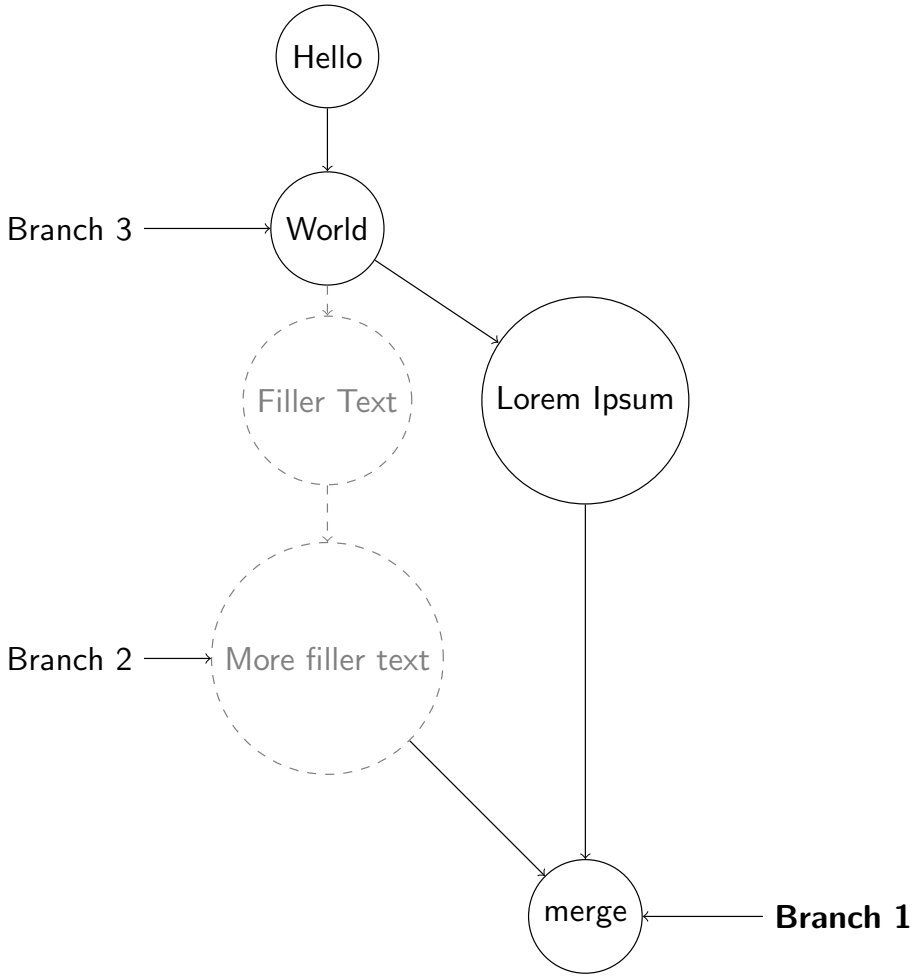
Branches and Merges

- From our previous branching slide...
- We can view all the available branches



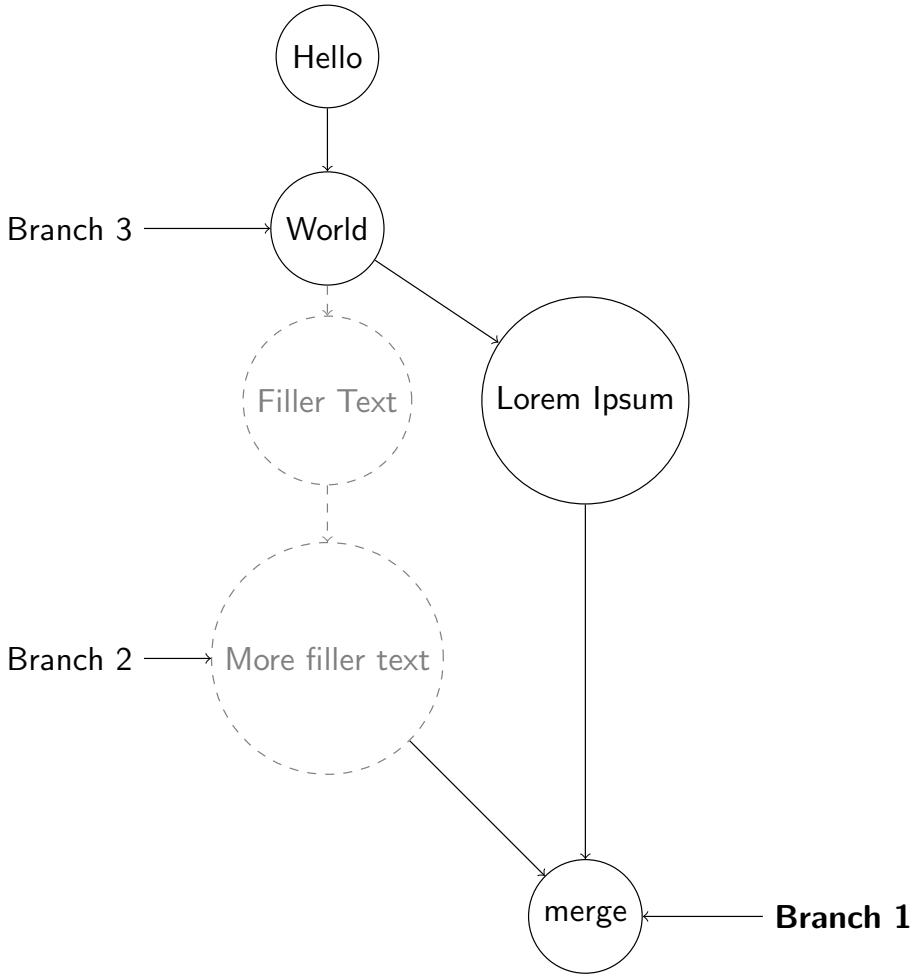
Branches and Merges

- From our previous branching slide...
- We can view all the available branches
- We can swap between branches



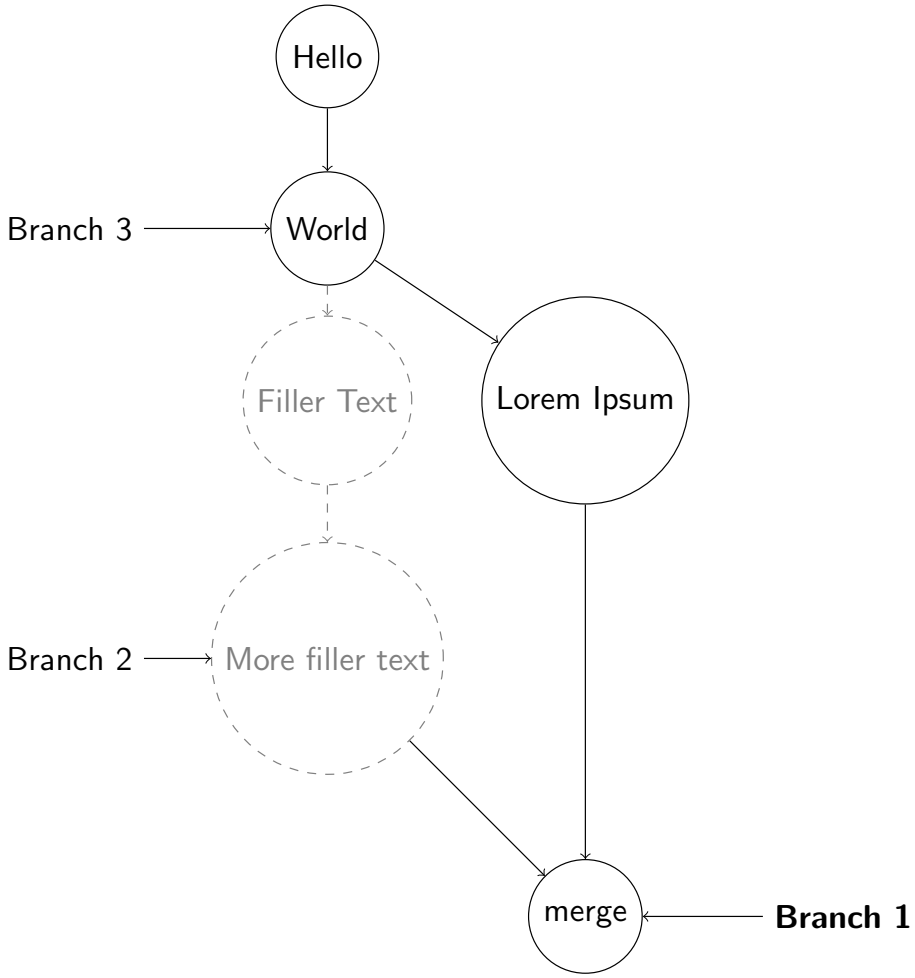
Branches and Merges

- From our previous branching slide...
- We can view all the available branches
- We can swap between branches
- We can do work on branches



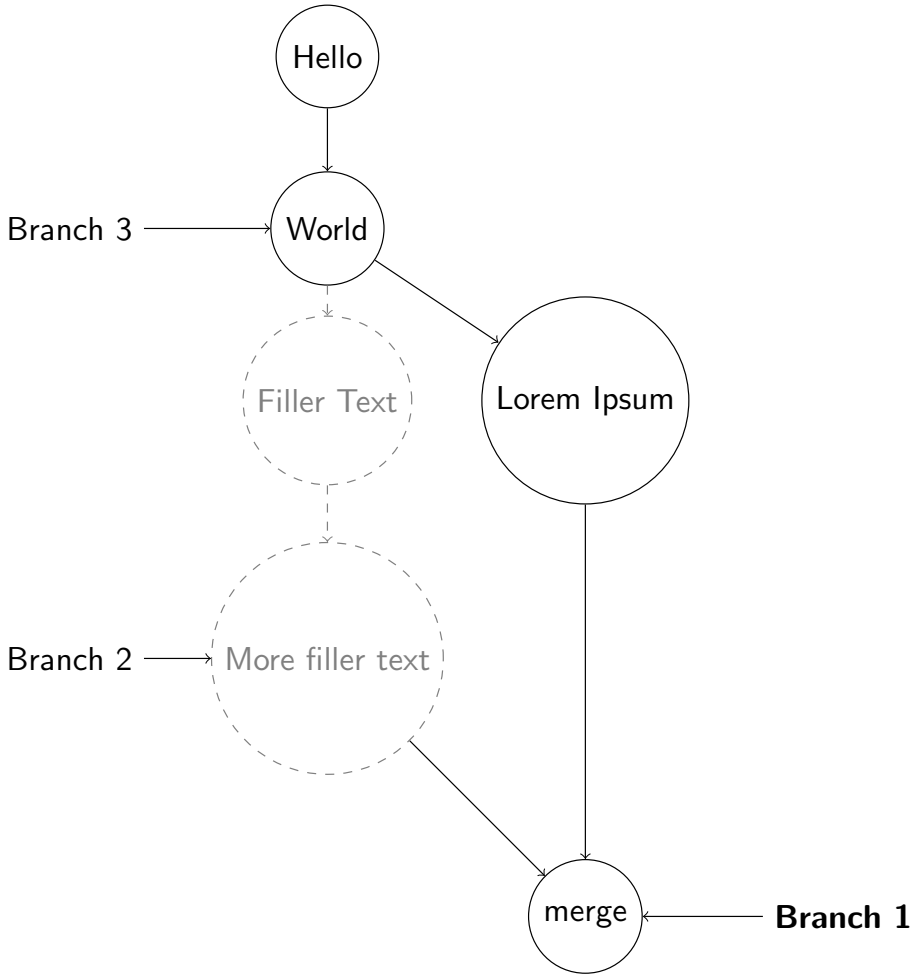
Branches and Merges

- From our previous branching slide...
- We can view all the available branches
- We can swap between branches
- We can do work on branches
- We can merge branches



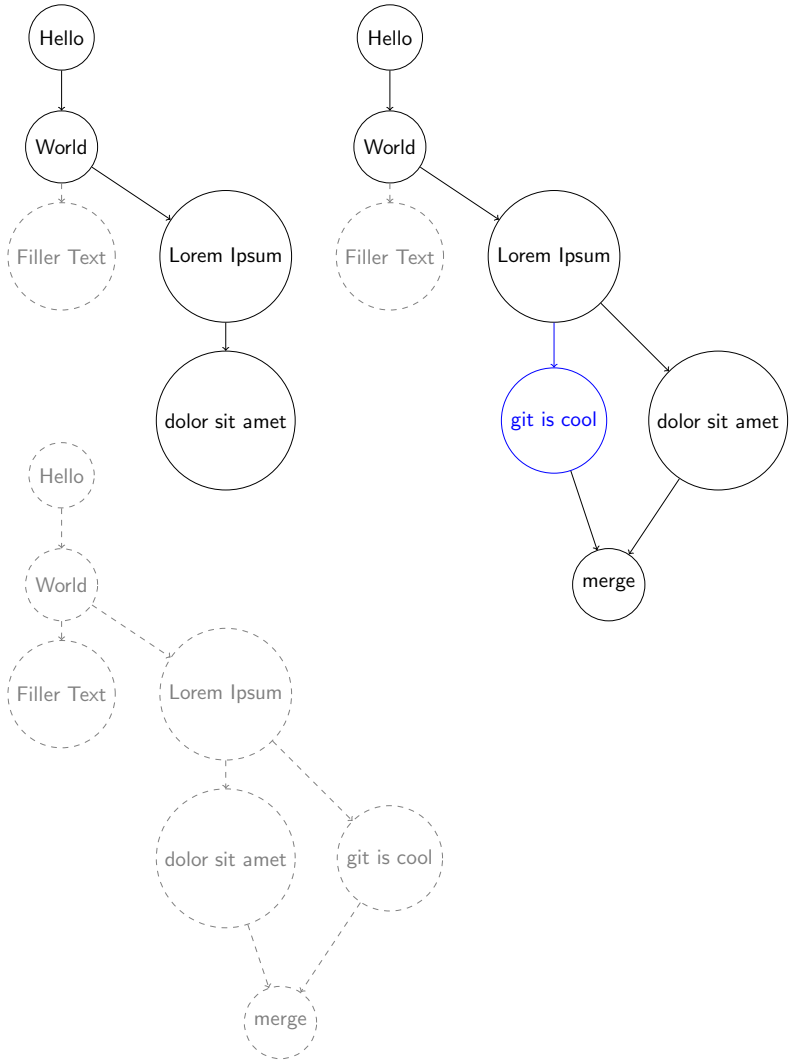
Branches and Merges

- From our previous branching slide...
- We can view all the available branches
- We can swap between branches
- We can do work on branches
- We can merge branches
- Merging doesn't always work



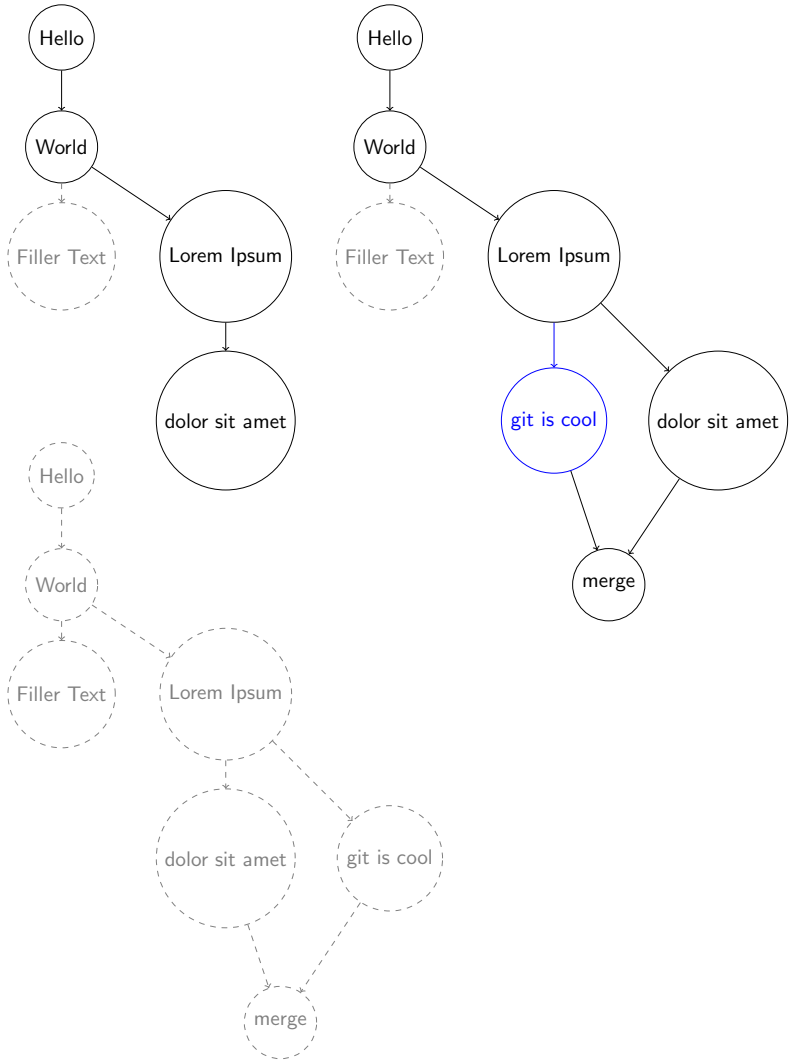
Collaborative Work

- From our previous collaborative work slide...



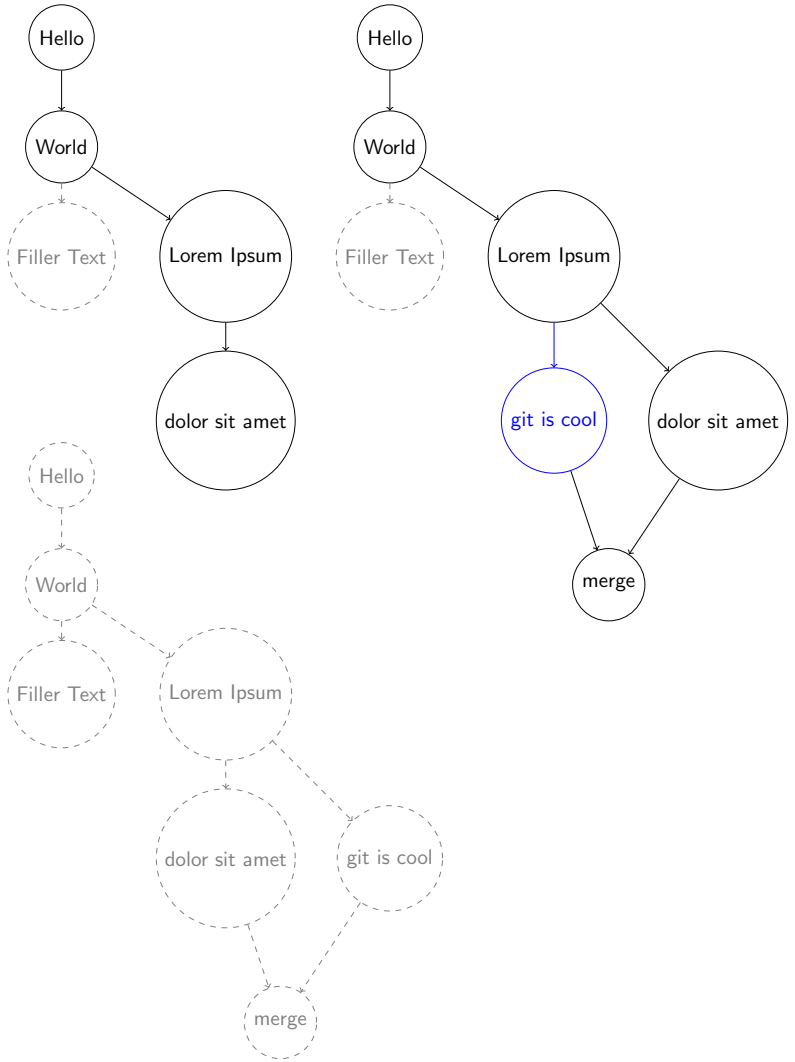
Collaborative Work

- From our previous collaborative work slide...
- We can clone a repository from somewhere else



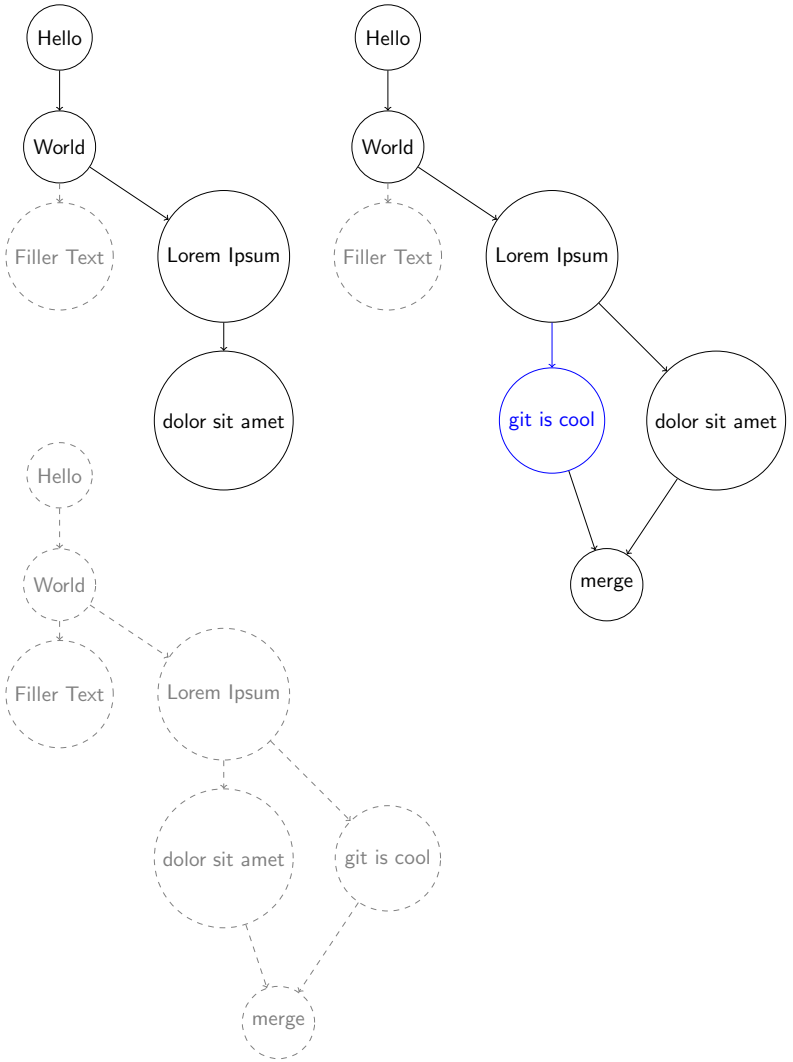
Collaborative Work

- From our previous collaborative work slide...
- We can clone a repository from somewhere else
- We can push to a repository



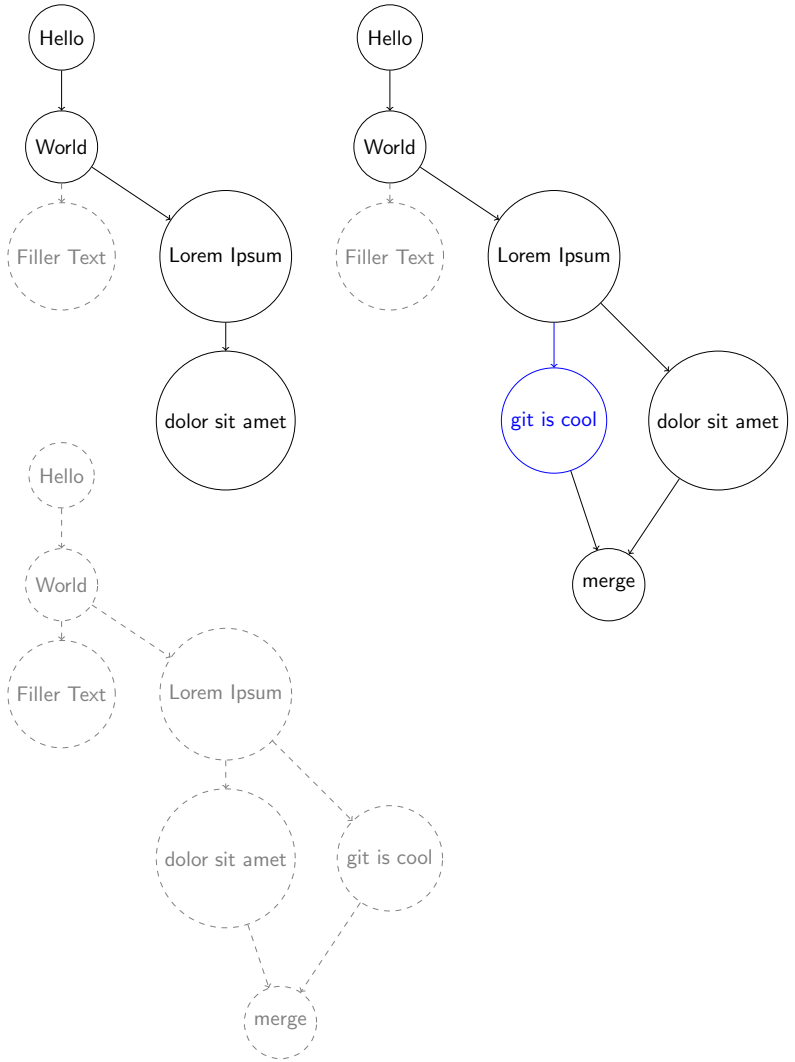
Collaborative Work

- From our previous collaborative work slide...
- We can clone a repository from somewhere else
- We can push to a repository
- We can fetch from a repository



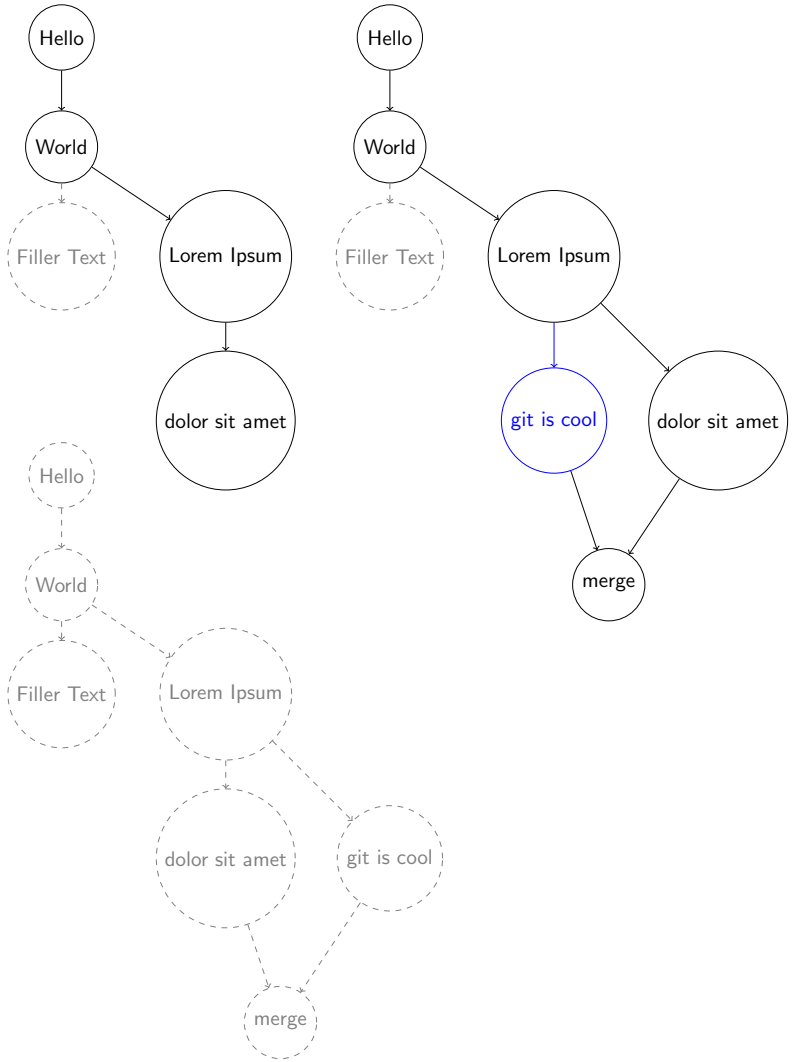
Collaborative Work

- From our previous collaborative work slide...
- We can clone a repository from somewhere else
- We can push to a repository
- We can fetch from a repository
- We can merge the remote work in with our own



Collaborative Work

- From our previous collaborative work slide...
- We can clone a repository from somewhere else
- We can push to a repository
- We can fetch from a repository
- We can merge the remote work in with our own
- We can use **pull** to combine these two processes



Outline

Flaws with Git

There are none

Flaws with Git

- Has issues with files that aren't plain text

Flaws with Git

- Has issues with files that aren't plain text
- Can get slower as you start dealing with huge files

Things That Aren't Git

- Other version control tools are either **distributed** or **centralised**

Things That Aren't Git

- Other version control tools are either **distributed** or **centralised**
- Common ones you may hear about are subversion (svn), mercurial (hg) or bazaar (bzr)

Things That Aren't Git

- Other version control tools are either **distributed** or **centralised**
- Common ones you may hear about are subversion (svn), mercurial (hg) or bazaar (bzr)
- Why centralised?

Things That Aren't Git

- Other version control tools are either **distributed** or **centralised**
- Common ones you may hear about are subversion (svn), mercurial (hg) or bazaar (bzr)
- Why centralised?
- In house version control

For Help

- Git-scm website

For Help

- Git-scm website
- A Successful Git Branching Model

For Help

- Git-scm website
- A Successful Git Branching Model
- `#projects` on slack

If You're Keen

- How does git do diffs? Search Levenshtein distance, or longest common subsequence problem

If You're Keen

- How does git do diffs? Search Levenshtein distance, or longest common subsequence problem
- How does git stop you changing history? Search SHA1 hashes

If You're Keen

- How does git do diffs? Search Levenshtein distance, or longest common subsequence problem
- How does git stop you changing history? Search SHA1 hashes
- Can you set up a git server?

If You're Keen

- How does git do diffs? Search Levenshtein distance, or longest common subsequence problem
- How does git stop you changing history? Search SHA1 hashes
- Can you set up a git server?
- What goes on inside git? Read the original source for git 1.0

Grand Finale

Grand Finale

Does anyone have any questions?

Grand Finale

Does anyone have any questions?
Thanks for coming :realheart: