



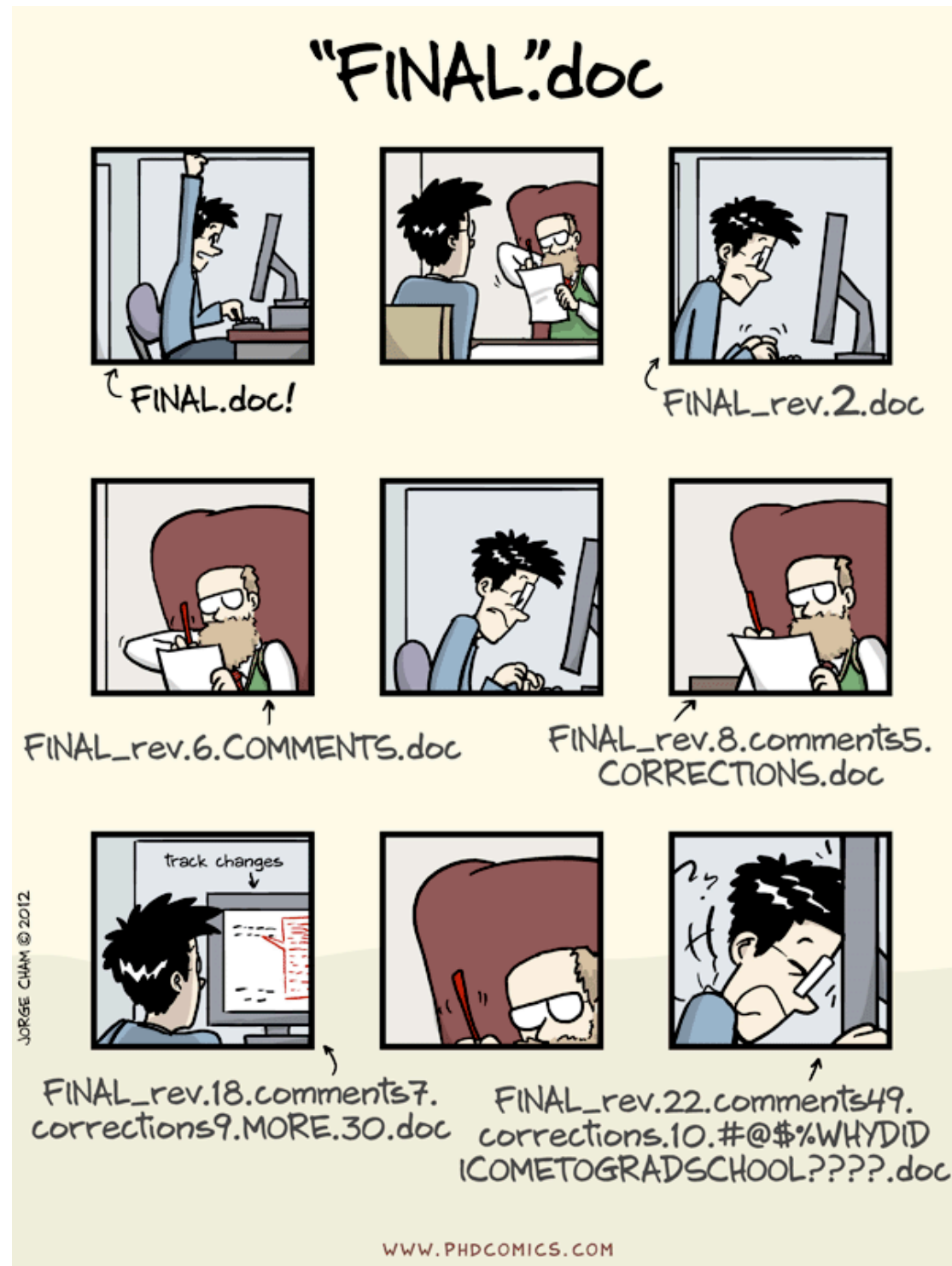
# Intro to version control, Git, and GitHub

Hermína Ghenu  
15 Oct 2024



# What is version control?

- Keeping track of changes to documents



# What is automated version control?

- Program to keep track of changes to documents  
e.g. Git, Mercurial, CVS, Subversion

# What is automated version control?

- Program to keep track of changes to documents  
e.g. Git, Mercurial, CVS, Subversion

# What is automated version control?

- Program to keep track of changes to documents  
e.g. Git, Mercurial, CVS, Subversion
- For text files, lets you see line-by-line changes

334	335	
335		- maleinitialfreq=Thread[males->maleratios/Total[maleratios]]
	336	+ maleinitialfreq=Thread[males->maleratios/Total[maleratios]/N]
336	337	

# What is automated version control?

- Program to keep track of changes to documents  
e.g. Git, Mercurial, CVS, Subversion

- For text files, lets you see line-by-line changes

334	335	
335		- maleinitialfreq=Thread[males->maleratios/Total[maleratios]]
	336	+ maleinitialfreq=Thread[males->maleratios/Total[maleratios]//N]
336	337	

- For non-text files, indicates that files were changed

# What is automated version control?

- Program to keep track of changes to documents  
e.g. Git, Mercurial, CVS, Subversion

- For text files, lets you see line-by-line changes

334	335	
335		- maleinitialfreq=Thread[males->maleratios/Total[maleratios]]
	336	+ maleinitialfreq=Thread[males->maleratios/Total[maleratios]/N]
336	337	

- For non-text files, indicates that files were changed
- Can be used locally on your computer

# What is automated version control?

- Program to keep track of changes to documents  
e.g. Git, Mercurial, CVS, Subversion

- For text files, lets you see line-by-line changes

334	335	
335		- maleinitialfreq=Thread[males->maleratios/Total[maleratios]]
	336	+ maleinitialfreq=Thread[males->maleratios/Total[maleratios]//N]
336	337	

- For non-text files, indicates that files were changed
- Can be used locally on your computer
- Can be synchronised with a server so that data and changes are backed up to the cloud



# How does Git work?

- MS Word track changes, Google Docs version history:

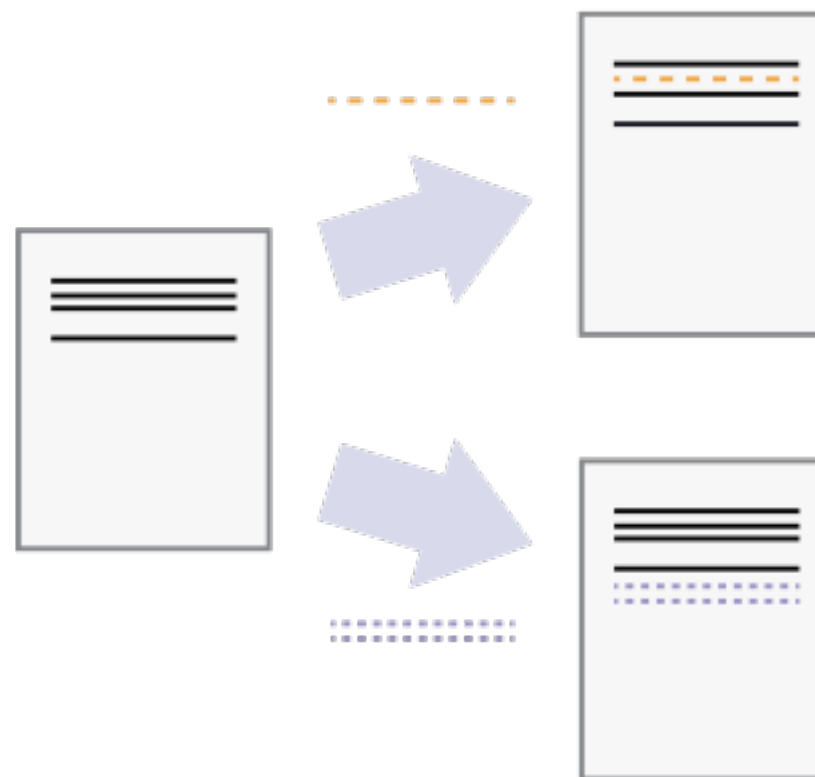


# How does Git work?

- MS Word track changes, Google Docs version history:

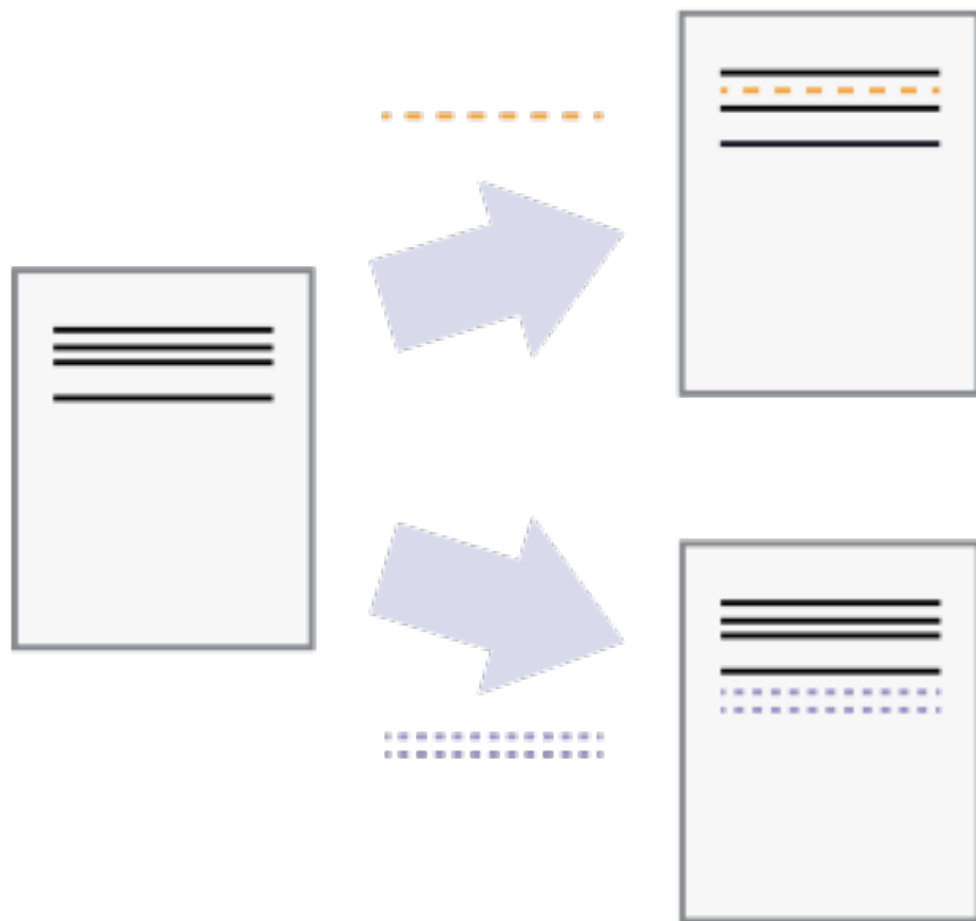


- can also think of the changes as separate from the document itself, resulting in different **versions** of the document



# How does Git work?

- Different versions of the document can be **merged**:



# How does Git work?

- Different versions of the document can be **merged**:



# How does Git work in practice?



# How does Git work in practice?



1. Do your thing as usual.

# How does Git work in practice?



1. Do your thing as usual.
2. Add changes to the **staging area**.  
This can be done as many times as you want per “stash” of work.

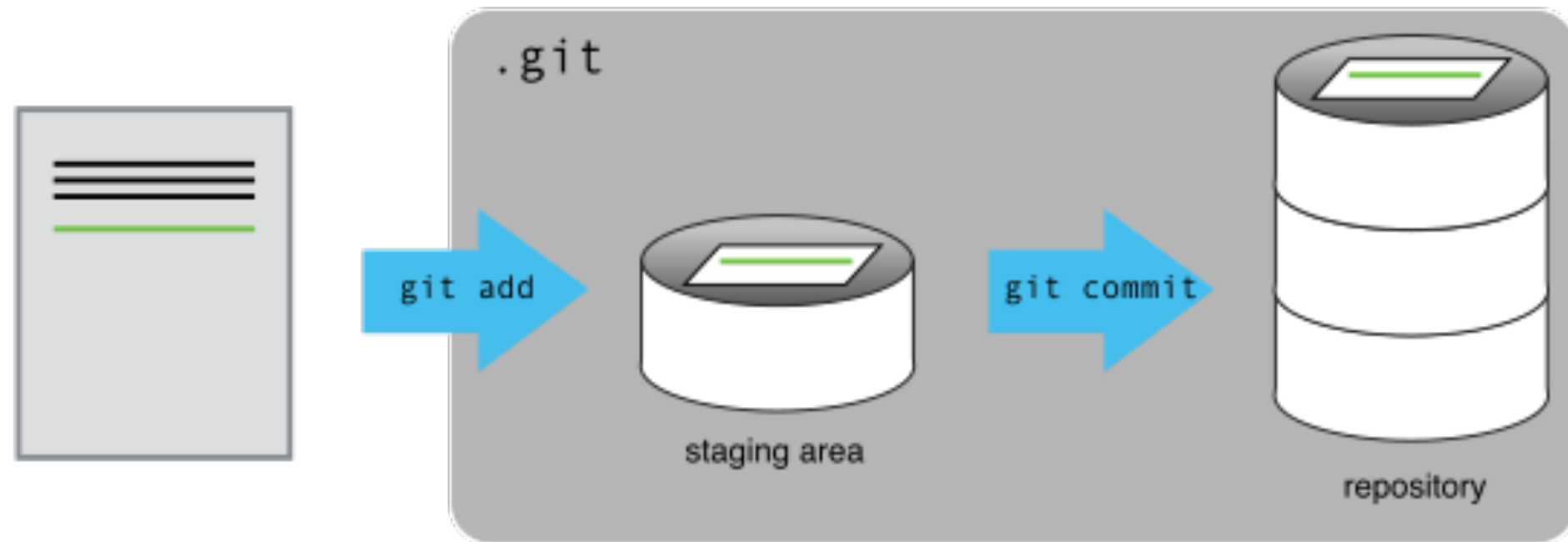
# How does Git work in practice?



1. Do your thing as usual.
2. Add changes to the **staging area**.  
This can be done as many times as you want per “stash” of work.
3. Decide you have completed a **stash** of work, then **commit** the stash to the Git **repository** with a message.



# How does Git work in practice?



1. Do your thing as usual.
2. Add changes to the **staging area**.  
This can be done as many times as you want per “stash” of work.
3. Decide you have completed a **stash** of work, then **commit** the stash to the Git **repository** with a message.

# Using Git on your computer

- Using Git yields a time-series of changes



# Using Git on your computer

- Using Git yields a time-series of changes
- **Stashes** of changes are grouped in a **commit**, the time-series as a whole is a **branch**



# Using Git on your computer

- Using Git yields a time-series of changes
- **Stashes** of changes are grouped in a **commit**, the time-series as a whole is a **branch**
- You can easily see changes over time and even go back in time to older versions



# Using Git on your computer

- Using Git yields a time-series of changes
- **Stashes** of changes are grouped in a **commit**, the time-series as a whole is a **branch**
- You can easily see changes over time and even go back in time to older versions
- Can be done on command line, using a GUI (e.g., GitHub Desktop, [and others](#)), or directly from the editor (e.g., RStudio, Atom)



# Using Git on your computer

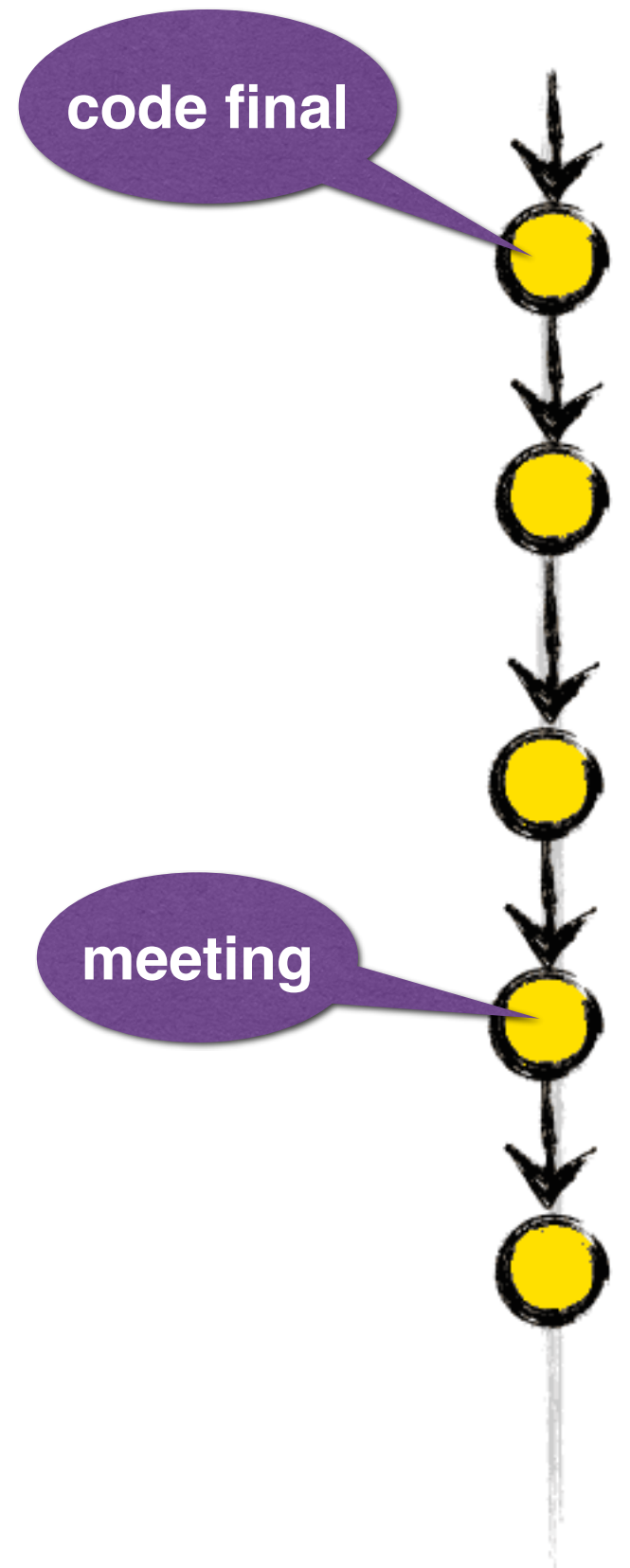
- Using Git yields a time-series of changes
- **Stashes** of changes are grouped in a **commit**, the time-series as a whole is a **branch**
- You can easily see changes over time and even go back in time to older versions
- Can be done on command line, using a GUI (e.g., GitHub Desktop [and others](#)), or directly from the editor (e.g., RStudio, Atom)



# Challenges!:

1. Open GitHub Desktop
2. Create a new repository.  
*Do NOT publish it to GitHub yet!*
3. Copy files you created before the break to this repository.  
Add and commit this change.  
*Remember to use a meaningful commit message!*
4. Identify **three** other things GitHub Desktop lets you do locally.

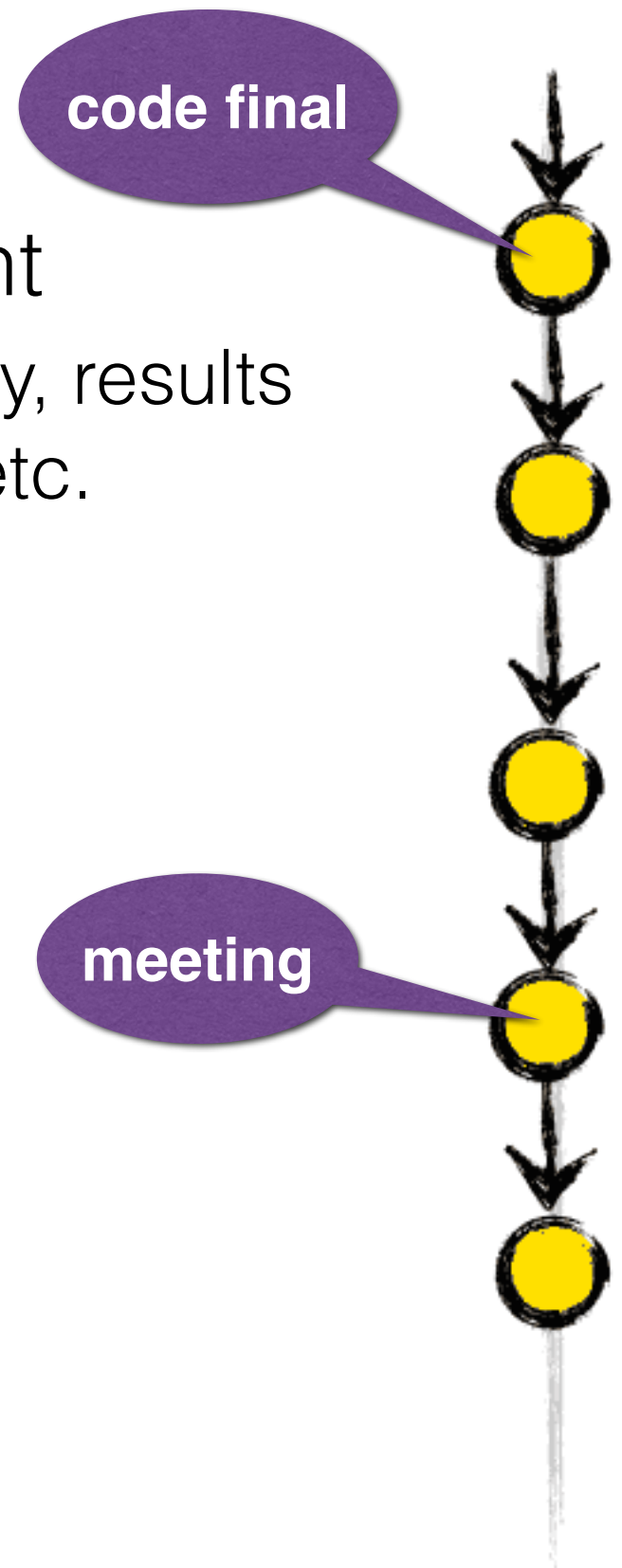
# Tagging important commits





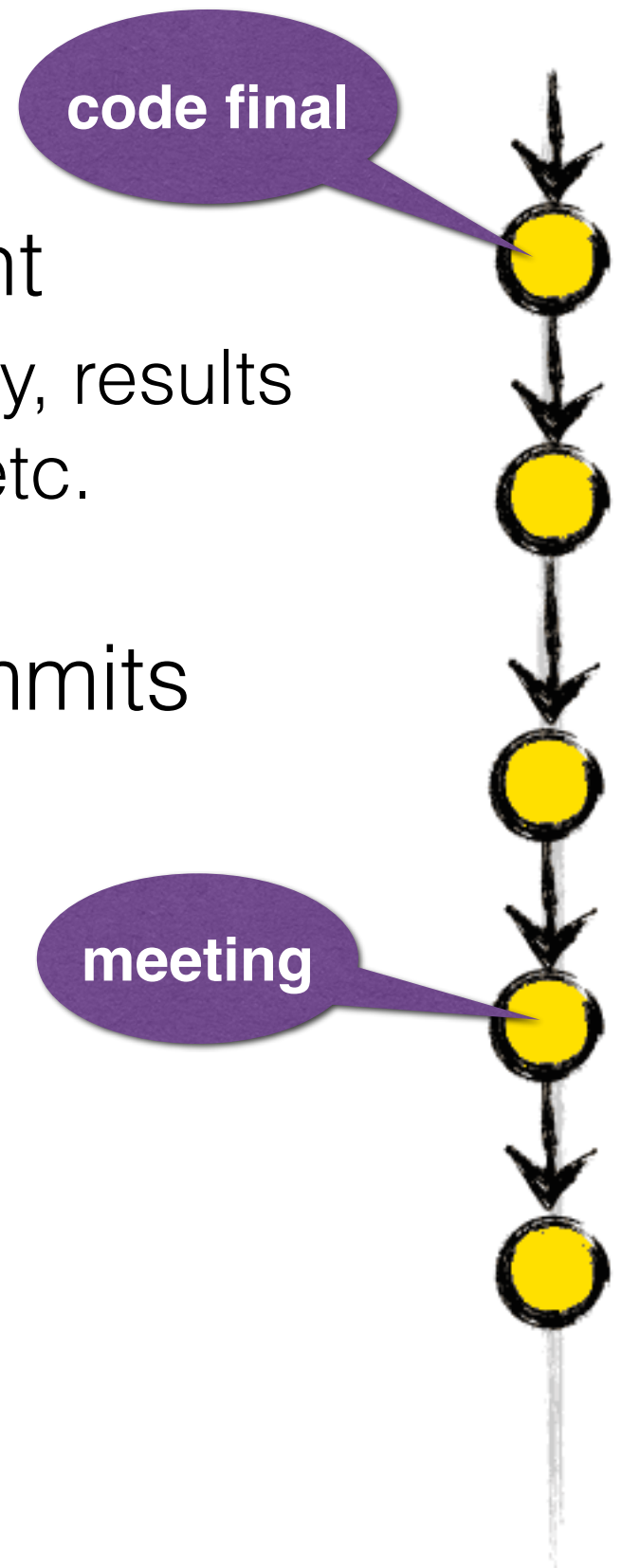
# Tagging important commits

- Some commits are especially important  
e.g. the last time your code worked properly, results  
you presented at your important meeting, etc.



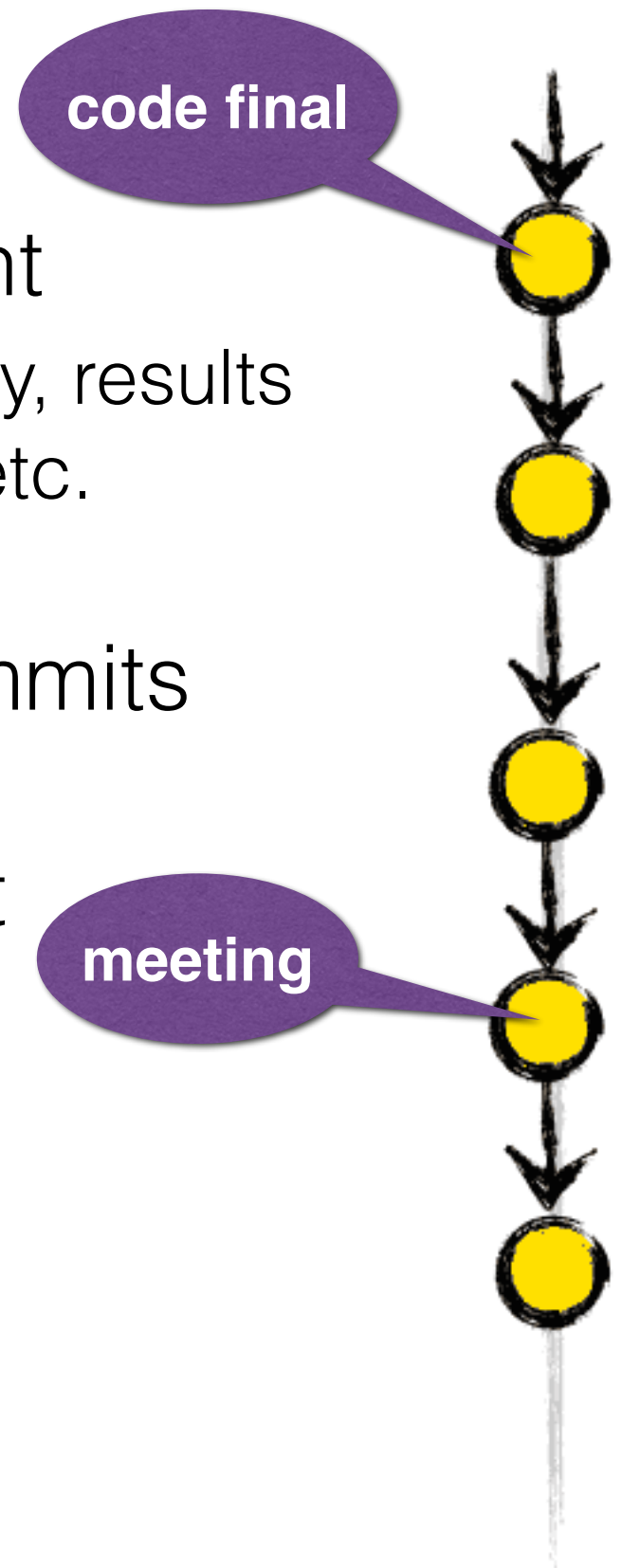
# Tagging important commits

- Some commits are especially important  
e.g. the last time your code worked properly, results you presented at your important meeting, etc.
- **Tags** let you label these important commits



# Tagging important commits

- Some commits are especially important  
e.g. the last time your code worked properly, results you presented at your important meeting, etc.
- **Tags** let you label these important commits
- It's easier to return to a tagged commit



# Tagging important commits

- Some commits are especially important  
e.g. the last time your code worked properly, results you presented at your important meeting, etc.
- **Tags** let you label these important commits
- It's easier to return to a tagged commit

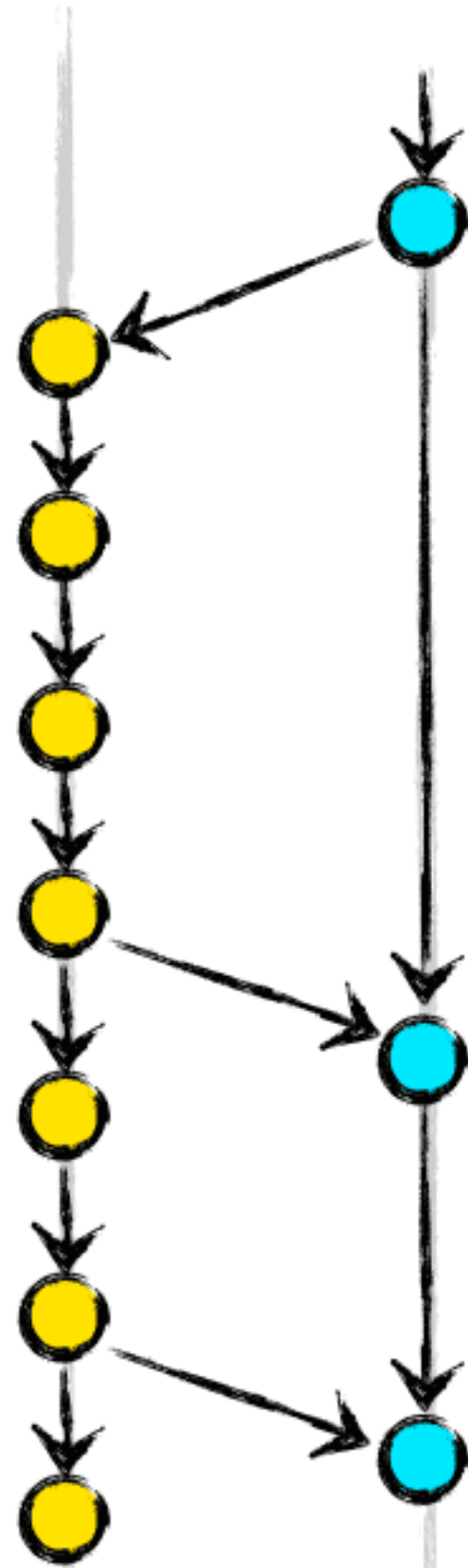
**Challenge:** Tag your commit as “presentation”.  
*You may have to google “github desktop tag”...*

code final

meeting

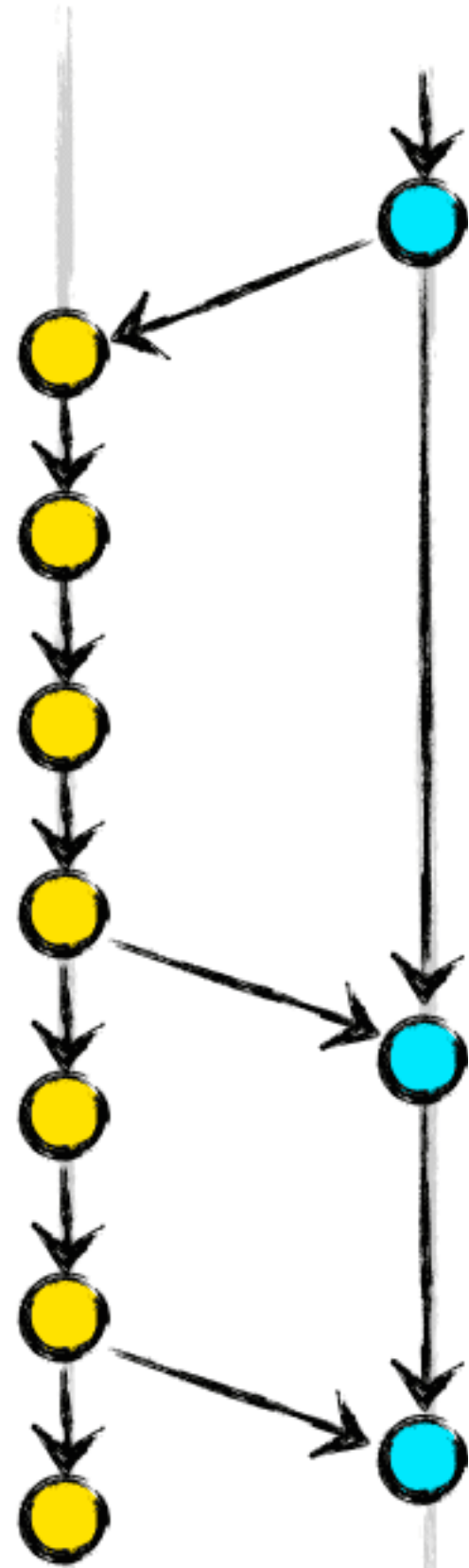


# Using Git with a server



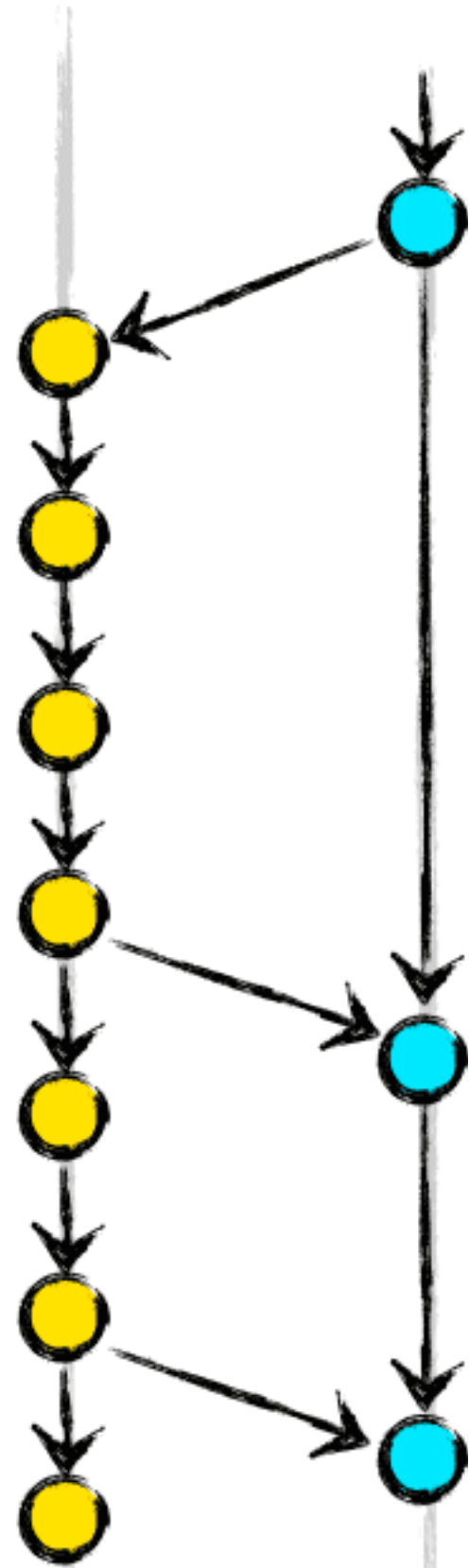
# Using Git with a server

- Git is good on its own, but pairing it with an online server (“cloud”) is especially powerful



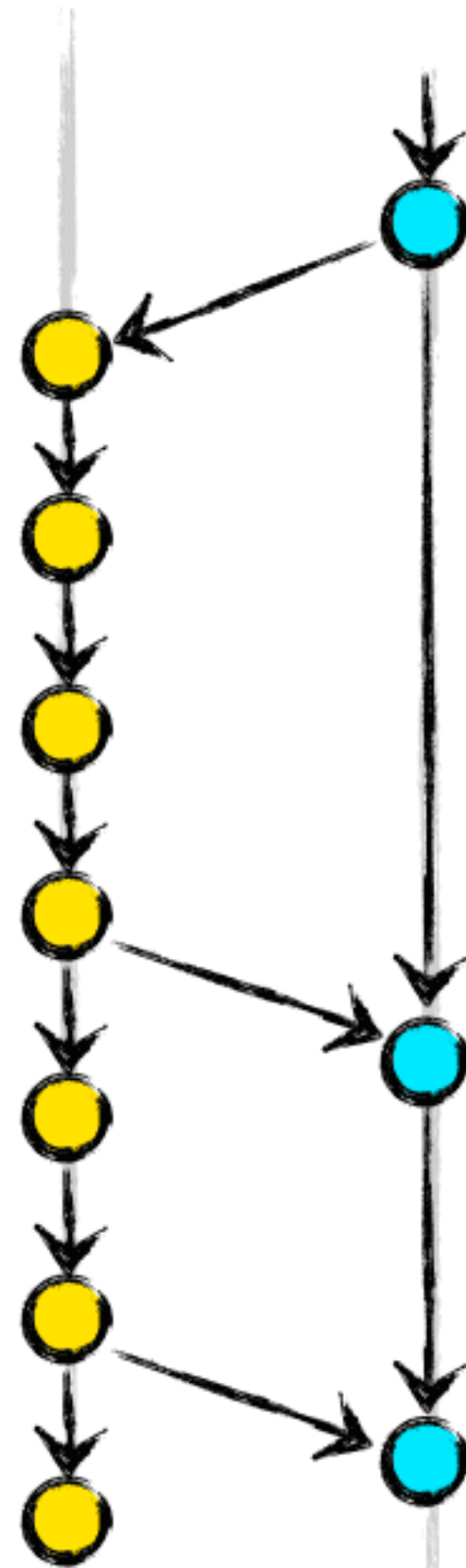
# Using Git with a server

- Git is good on its own, but pairing it with an online server (“cloud”) is especially powerful
- Facilitates cloud backups, collaboration, and distribution all in one!



# Using Git with a server

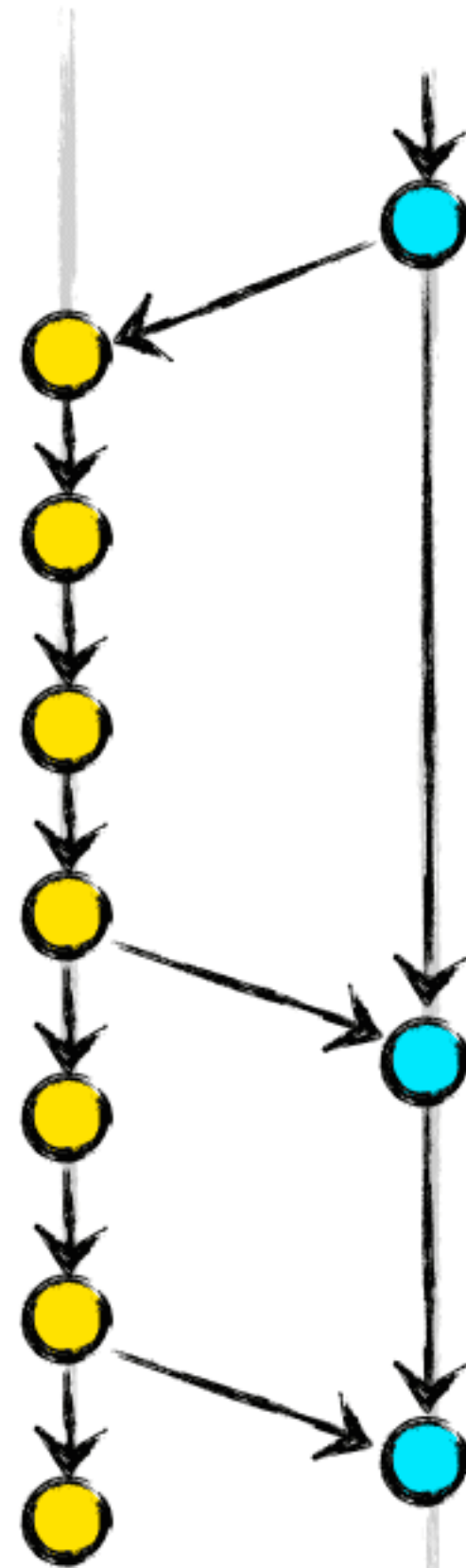
- Git is good on its own, but pairing it with an online server (“cloud”) is especially powerful
- Facilitates cloud backups, collaboration, and distribution all in one!
- **Yellow:** local **branch**, **cyan:** server **branch**





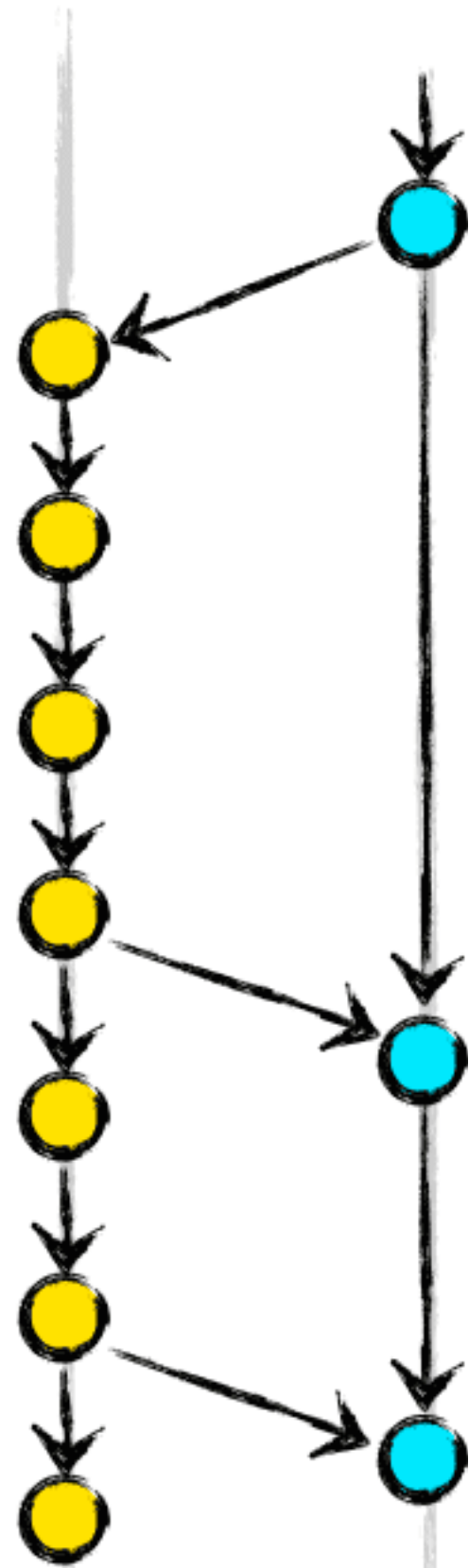
# Using Git with a server

- Git is good on its own, but pairing it with an online server (“cloud”) is especially powerful
- Facilitates cloud backups, collaboration, and distribution all in one!
- **Yellow:** local **branch**, **cyan:** server **branch**
- GitHub, GitLab, BitBucket, etc.



# Using Git with a server


- Git is good on its own, but pairing it with an online server (“cloud”) is especially powerful
- Facilitates cloud backups, collaboration, and distribution all in one!
- **Yellow:** local **branch**, **cyan:** server **branch**
- GitHub, GitLab, BitBucket, etc.



# Challenges!:


1. From GitHub Desktop, publish your repository to GitHub.  
*Decide if you want it to be public or private.*
2. Use GitHub Desktop to make a change.  
Add and commit the change.  
Why is this change not visible on GitHub?
3. Push the commit to GitHub.  
Is this change visible on GitHub version now?
4. Identify **three** other things GitHub lets you do on the cloud.

# Working collaboratively is the best part about Git


 master ▾

Commits on Nov 2, 2022

Update publications.md ...

 banklab committed 3 minutes ago ✓

Verified




9462242


<>

Commits on Oct 30, 2022

Merge pull request #49 from banklab/news ...

 AdamandiaMado committed 3 days ago ✓


Verified




b9d888d

<>

Update index.md

 AdamandiaMado committed 3 days ago

Verified




62e3948


<>

Commits on Oct 12, 2022

Merge pull request #48 from banklab/photo ...

 AdamandiaMado committed 21 days ago ✓


Verified




710433f

<>

Update index.md

 AdamandiaMado committed 21 days ago


Verified




e3a7798

<>

Update people.md

 AdamandiaMado committed 21 days ago


Verified




6475001

<>

Add files via upload

 AdamandiaMado committed 21 days ago

Verified




9fb9b18


<>

Commits on Sep 23, 2022

Update index.md

 AdamandiaMado committed on 23 Sep ✓

Verified



b6b6bdb

<>

# Working collaboratively is the best part about Git

- To add a collaborator, go to the repository on GitHub Settings -> Collaborators -> type in their GitHub username

The screenshot shows the GitHub repository settings page for 'EvoNerd / very\_cool\_project'. The repository is marked as 'Private'. At the top, there are buttons for 'Unwatch' (1), 'Fork' (0), and 'Star' (0). Below this is a navigation bar with links for 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', 'Insights', and 'Settings' (which is highlighted). On the left sidebar, under the 'Access' section, 'Collaborators' is selected. The main content area is titled 'Who has access' and contains two panels: 'PRIVATE REPOSITORY' (with a lock icon) stating 'Only those with access to this repository can view it.' and a 'Manage' link; and 'DIRECT ACCESS' (with a person icon) stating '0 collaborators have access to this repository. Only you can contribute to this repository.' Below this is a section titled 'Manage access' with a dropdown arrow. At the bottom, there is a large box with a lock icon and the text 'You haven't invited any collaborators yet'.

EvoNerd / **very\_cool\_project** Private

Unwatch 1 Fork 0 Star 0

<> Code Issues Pull requests Actions Projects Security Insights **Settings**

General

Access

**Collaborators**

Code and automation

Branches

Tags

Actions

Webhooks

Pages

Security

Code security and analysis

Deploy keys

## Who has access

PRIVATE REPOSITORY

Only those with access to this repository can view it.

[Manage](#)

DIRECT ACCESS

0 collaborators have access to this repository. Only you can contribute to this repository.

## Manage access

You haven't invited any collaborators yet

# Working collaboratively is the best part about Git

- To add a collaborator, go to the repository on GitHub Settings -> Collaborators -> type in their GitHub username

The screenshot shows the GitHub repository settings page for 'EvoNerd / very\_cool\_project'. The repository is marked as 'Private'. At the top, there are buttons for 'Unwatch' (1), 'Fork' (0), and 'Star' (0). Below these are navigation tabs: 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', 'Insights', and 'Settings'. The 'Settings' tab is highlighted with a red box. On the left sidebar, under the 'Access' section, the 'Collaborators' option is selected. The main content area is titled 'Who has access' and contains two panels: 'PRIVATE REPOSITORY' (with a lock icon) stating 'Only those with access to this repository can view it.' and a 'Manage' link; and 'DIRECT ACCESS' (with a person icon) stating '0 collaborators have access to this repository. Only you can contribute to this repository.' Below this is a 'Manage access' section with a dropdown arrow. At the bottom, there is a large box with a lock icon and the text 'You haven't invited any collaborators yet'.

EvoNerd / **very\_cool\_project** Private

Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Security Insights **Settings**

General

Access

**Collaborators**

Code and automation

Branches

Tags

Actions

Webhooks

Pages

Security

Code security and analysis

Deploy keys

## Who has access

**PRIVATE REPOSITORY**

Only those with access to this repository can view it.

[Manage](#)

**DIRECT ACCESS**

0 collaborators have access to this repository. Only you can contribute to this repository.

## Manage access

You haven't invited any collaborators yet

# Working collaboratively is the best part about Git

- To add a collaborator, go to the repository on GitHub Settings -> Collaborators -> type in their GitHub username

The screenshot shows the GitHub repository settings page for 'EvoNerd / very\_cool\_project'. The repository is marked as 'Private'. At the top, there are buttons for 'Unwatch' (1), 'Fork' (0), and 'Star' (0). Below these are navigation tabs: 'Code', 'Issues', 'Pull requests', 'Actions', 'Projects', 'Security', 'Insights', and 'Settings'. The 'Settings' tab is highlighted with a red box. On the left sidebar, under the 'Access' section, the 'Collaborators' option is highlighted with a red box. The main content area is titled 'Who has access' and contains two panels: 'PRIVATE REPOSITORY' and 'DIRECT ACCESS'. The 'PRIVATE REPOSITORY' panel states 'Only those with access to this repository can view it.' and has a 'Manage' link. The 'DIRECT ACCESS' panel states '0 collaborators have access to this repository. Only you can contribute to this repository.' Below these panels is a section titled 'Manage access' which is currently empty, showing a message 'You haven't invited any collaborators yet' with a lock icon.

EvoNerd / **very\_cool\_project** Private

Unwatch 1 Fork 0 Star 0

Code Issues Pull requests Actions Projects Security Insights **Settings**

General

**Access**

**Collaborators**

Code and automation

Branches

Tags

Actions

Webhooks

Pages

Security

Code security and analysis

Deploy keys

## Who has access

PRIVATE REPOSITORY

Only those with access to this repository can view it.

[Manage](#)

DIRECT ACCESS

0 collaborators have access to this repository. Only you can contribute to this repository.

## Manage access

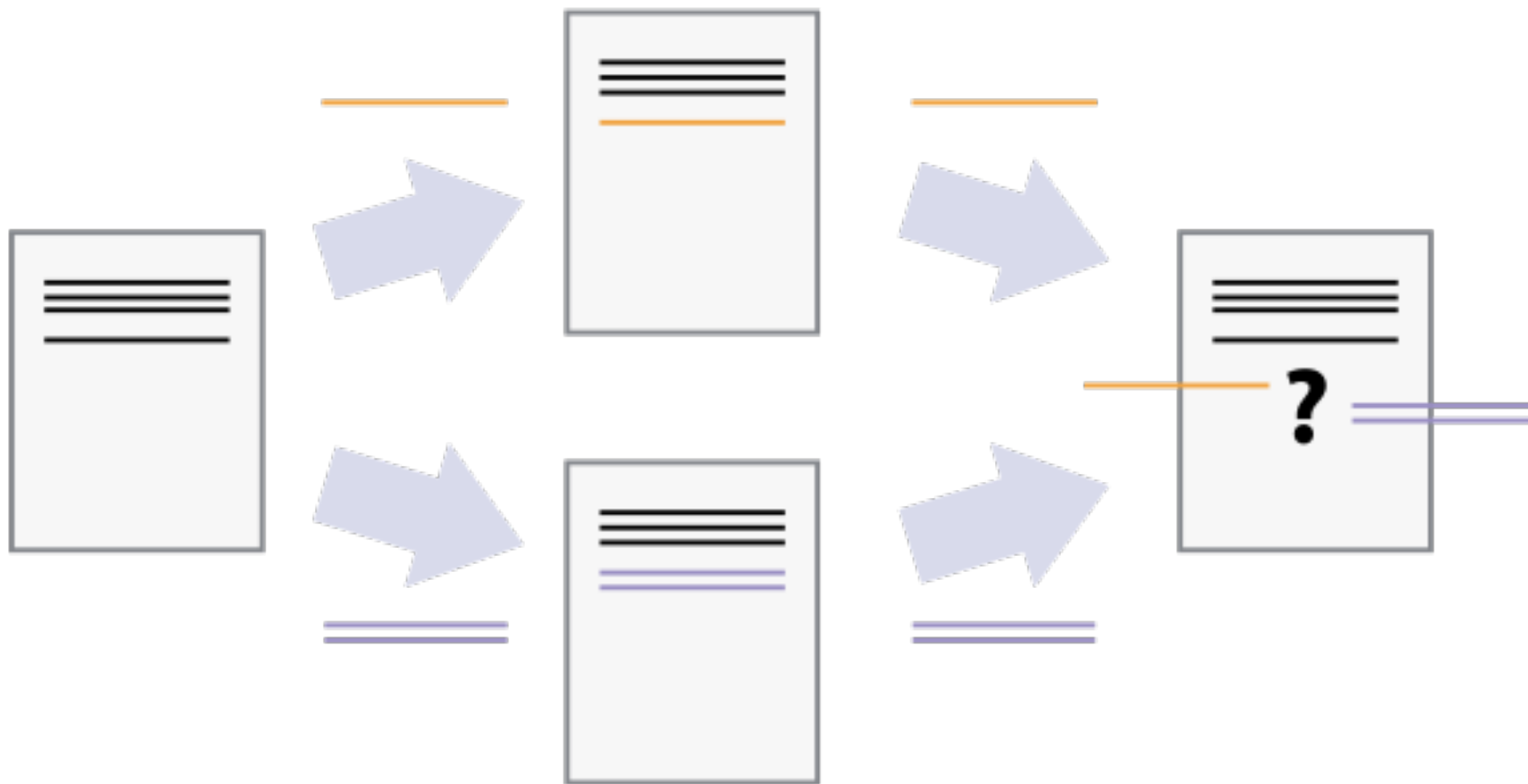
You haven't invited any collaborators yet

Working collaboratively is the  
**worst** part about Git



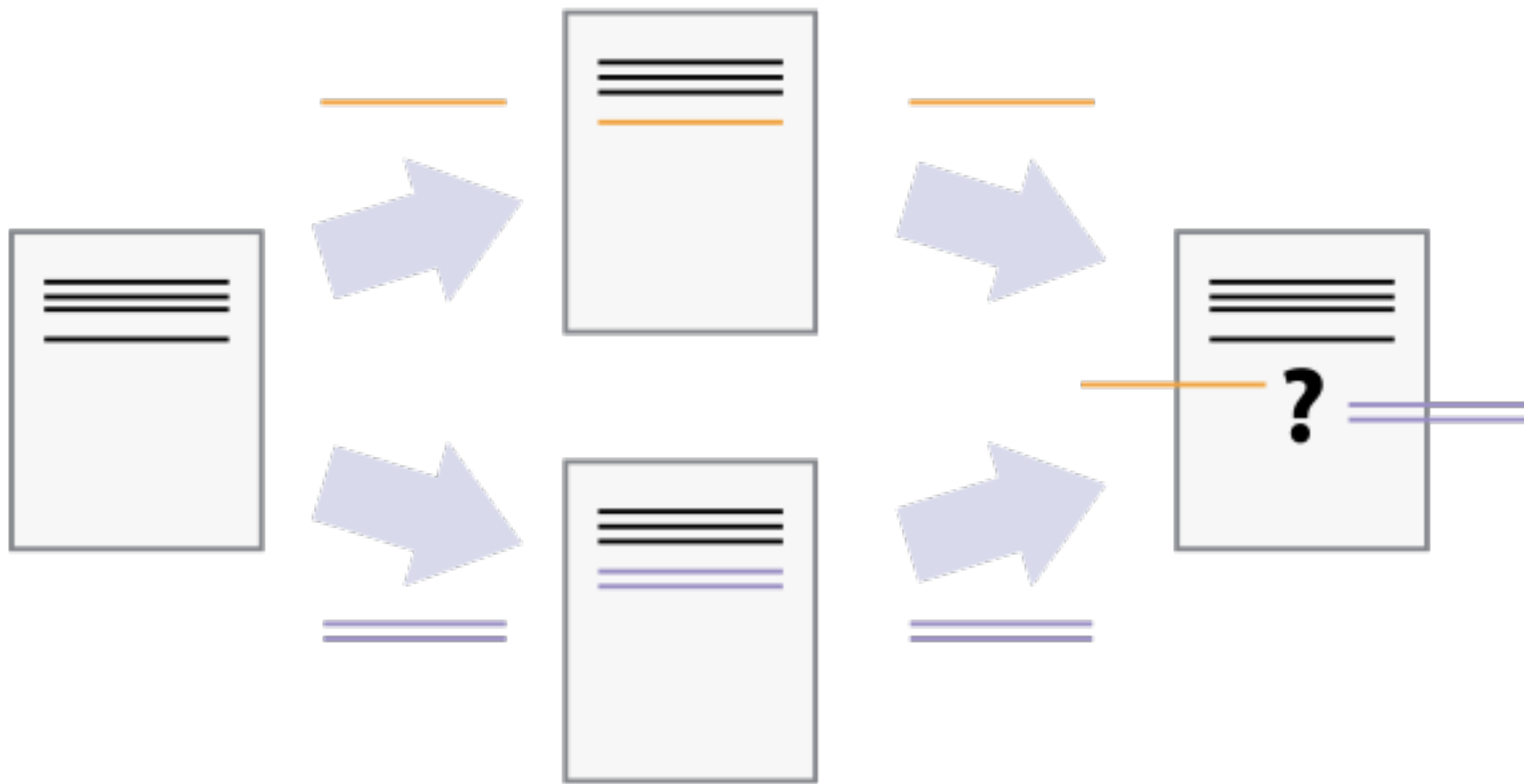
# Working collaboratively is the **worst** part about Git

- If different versions modify the same line, Git does not know how to merge: this creates a **conflict**,



# Working collaboratively is the **worst** part about Git

- If different versions modify the same line, Git does not know how to merge: this creates a **conflict**,



- Learn more about Git from [Software Carpentry](#)

# Challenges!:

1. From GitHub Desktop, clone a repository from this URL  
*[https://github.com/EvoNerd/THEE\\_2024reproducibility.git](https://github.com/EvoNerd/THEE_2024reproducibility.git)*

# Challenges!:

1. From GitHub Desktop, clone a repository from this URL  
*[https://github.com/EvoNerd/THEE\\_2024reproducibility.git](https://github.com/EvoNerd/THEE_2024reproducibility.git)*
2. Make a change to the file called *reproducibility.Rmd*

# Challenges!:

1. From GitHub Desktop, clone a repository from this URL  
*[https://github.com/EvoNerd/THEE\\_2024reproducibility.git](https://github.com/EvoNerd/THEE_2024reproducibility.git)*
2. Make a change to the file called *reproducibility.Rmd*
3. Push the commit to GitHub.  
Did this create a conflict or was git able to merge it successfully?

# Challenges!:

1. From GitHub Desktop, clone a repository from this URL  
*[https://github.com/EvoNerd/THEE\\_2024reproducibility.git](https://github.com/EvoNerd/THEE_2024reproducibility.git)*
2. Make a change to the file called *reproducibility.Rmd*
3. Push the commit to GitHub.  
Did this create a conflict or was git able to merge it successfully?
4. Keep making changes & committing them until we create a conflict and everyone has had a chance to practice merging conflicts.

# Conclusions

# Conclusions

- Automated version control is useful  
*You don't have to use Git but do consider using something.*



# Conclusions

- Automated version control is useful  
*You don't have to use Git but do consider using something.*
- Git can be used locally for version control

# Conclusions

- Automated version control is useful  
*You don't have to use Git but do consider using something.*
- Git can be used locally for version control
- Git can also be used with an online server  
*Facilitates cloud backups, collaboration, and distribution all in one!*

# Conclusions

- Automated version control is useful  
*You don't have to use Git but do consider using something.*
- Git can be used locally for version control
- Git can also be used with an online server  
*Facilitates cloud backups, collaboration, and distribution all in one!*
- GitHub is an internet hosting service for Git

# Conclusions

- Automated version control is useful  
*You don't have to use Git but do consider using something.*
- Git can be used locally for version control
- Git can also be used with an online server  
*Facilitates cloud backups, collaboration, and distribution all in one!*
- GitHub is an internet hosting service for Git
- Coupling Git with an online server makes reproducibility easy:

**Hoard**



**Share**

