VERSION CONTROL

Orhun Ulusahin

```
1 <h3>Version control</h3>
2 <p>Orhun Ulusahin</p>
```

git_kpls_2022.html

Key Practices for the Language Scientist 2022

```
1 <h3>Version control</h3>
2 <p style="opacity: 0.2; margin-top: 20px;">Orhun Ulusahin</p
3 <p>Key Practices for the Language Scientist 2022</p>
```

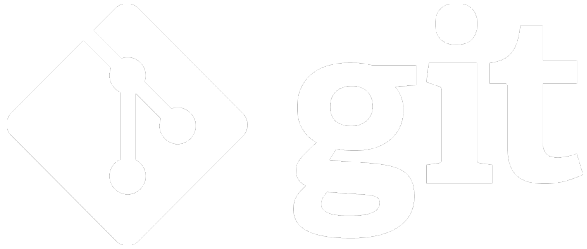
git_kpls_2022 (2).html

🤔"Wish I had a convenient way to track all these changes..."

```
1 <h3>Version control</h3>
2 <p>🤔"Wish I had a convenient way to track all these changes
```

git_kpls_2022_wish.html

🤔"Wish I had a convenient way to track all these changes!"



```
1 <h3>Version control Using Git</h3>
2 <p style="opacity: 0.2; margin-top: 20px;">
3   🤔"Wish I had a convenient way to track all these changes
4 </p>
5 <img src="media/git_logo_light.png" style="height: 120px;">
```

git_kpls_2022_wish (2).html

By the end of this lecture you will...

By the end of this lecture you will...

- Understand Git dataflow

By the end of this lecture you will...

- Understand Git dataflow
- Have a working local Git repository

By the end of this lecture you will...

- Understand Git dataflow
- Have a working local Git repository
- Have a working remote Git repository

By the end of this lecture you will...

- Understand Git dataflow
- Have a working local Git repository
- Have a working remote Git repository
- Not be afraid of command line interfaces
  (if you were, to begin with)

Because on a typical day...

Because on a typical day...

- You write things on a computer

Because on a typical day...

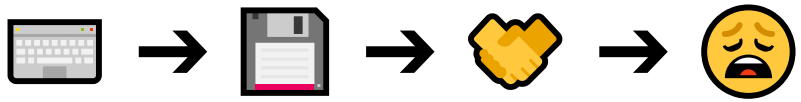
- You write things on a computer
- You save what you wrote

Because on a typical day...

- You write things on a computer
- You save what you wrote
- You meet your supervisor

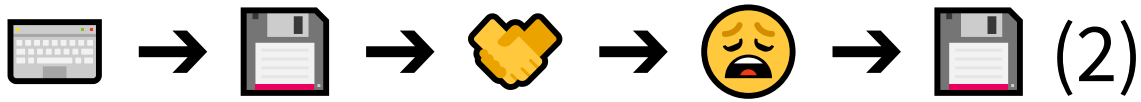
Because on a typical day...

- You write things on a computer
- You save what you wrote
- You meet your supervisor
- You edit what you wrote earlier



Because on a typical day...

- You write things on a computer
- You save what you wrote
- You meet your supervisor
- You edit what you wrote earlier
- You save the new thing you wrote

 (2)

And while you do all these things...

And while you do all these things...

- You want to track your progress

And while you do all these things...

- You want to track your progress
- You want to have redundancy/history

And while you do all these things...

- You want to track your progress
- You want to have redundancy/history
- You want to revert to older versions if something goes wrong

And while you do all these things...

- You want to track your progress
- You want to have redundancy/history
- You want to revert to older versions if something goes wrong
- You want to track which changes introduced what parts

```
<h4>Why version control?</h4>
<p>And while you do all these things...</p>
<ul>
  <li class="fragment fade-up">
    You want to track your progress
  </li>
  <li class="fragment fade-up">You want to have redundancy
  </li>
  <li class="fragment fade-up">
    You want to revert to older versions if something goes
  </li>
  <li class="fragment fade-up">
    You want to track which changes introduced what
  </li>
</ul>
```

git_kpls_2022 (3).html

Because within 10 minutes, I created...

Because within 10 minutes, I created...

- git_kpls_2022.html
- git_kpls_2022 (2).html
- git_kpls_2022_wish.html
- git_kpls_2022_wish (2).html
- git_kpls_2022 (3).html

Because within 10 minutes, I created...

- git_kpls_2022.html
- git_kpls_2022 (2).html
- git_kpls_2022_wish.html
- git_kpls_2022_wish (2).html
- git_kpls_2022 (3).html

And now...

```
<h4>Why version control?</h4>
<p>Because within 10 minutes, I created...</p>
<ul class="fragment fade-up">
  <li>git_kpls_2022.html</li>
  <li>git_kpls_2022 (2).html</li>
  <li>git_kpls_2022_wish.html</li>
  <li>git_kpls_2022_wish (2).html</li>
  <li>git_kpls_2022 (3).html</li>
</ul>
<p class="fragment fade-up">And now...</p>
```

oh_no.html

- Institutional version control tools might be inconveniently large

- Institutional version control tools might be inconveniently large
- They might also require centralized servers

- Institutional version control tools might be inconveniently large
- They might also require centralized servers
- Private services might charge money for version control

- Institutional version control tools might be inconveniently large
- They might also require centralized servers
- Private services might charge money for version control
- Collaboration with version control might be difficult or impossible on some platforms

TG backs up everything on network drives daily. But daily backups...

TG backs up everything on network drives daily. But daily backups...

- Are usually too infrequent to be useful

TG backs up everything on network drives daily. But daily backups...

- Are usually too infrequent to be useful
- Might be a hassle to revert to (both for TG and you)

TG backs up everything on network drives daily. But daily backups...

- Are usually too infrequent to be useful
- Might be a hassle to revert to (both for TG and you)
- Don't provide multiple versions

TG backs up everything on network drives daily. But daily backups...

- Are usually too infrequent to be useful
- Might be a hassle to revert to (both for TG and you)
- Don't provide multiple versions
- Don't provide clear file histories

Dropbox will let you keep a file history...

Dropbox will let you keep a file history...

- But only if you pay them

Dropbox will let you keep a file history...

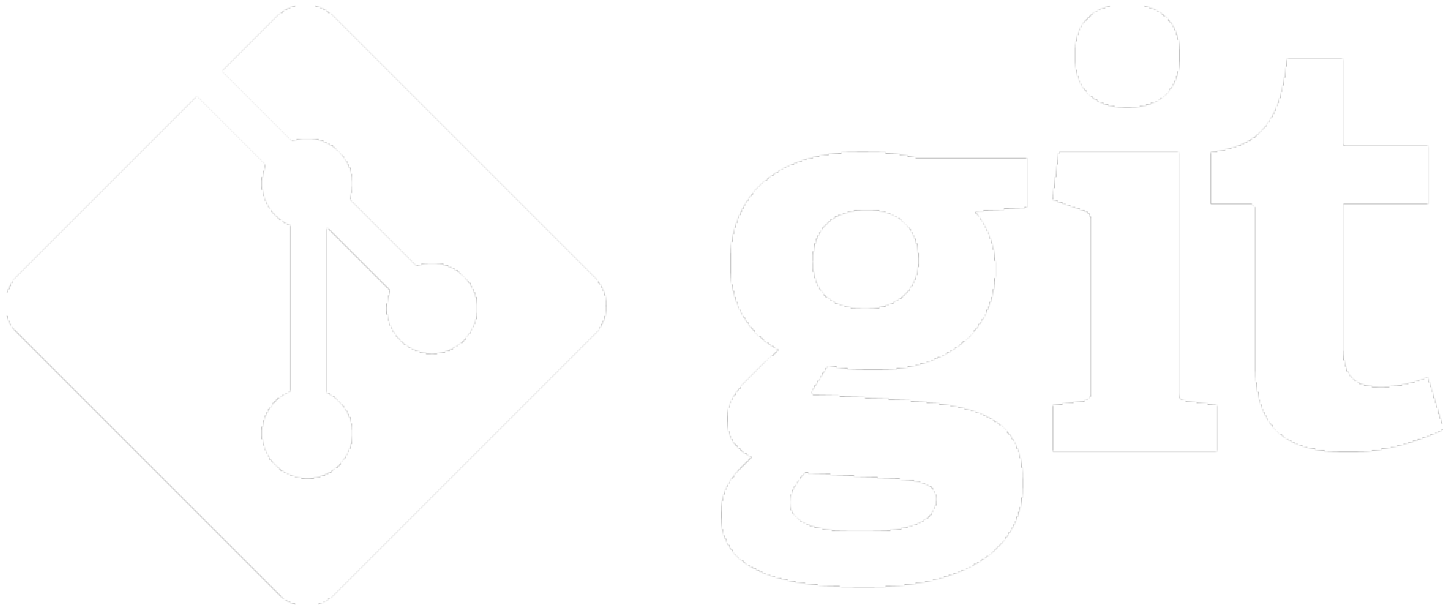
- But only if you pay them
- With poor tracking of individual versions

Dropbox will let you keep a file history...

- But only if you pay them
- With poor tracking of individual versions
- With extremely limited collaboration options

Dropbox will let you keep a file history...

- But only if you pay them
- With poor tracking of individual versions
- With extremely limited collaboration options
- While keeping your data in a remote server in god knows where


git

- Committed to data integrity

- Committed to data integrity
- Uncompromisingly fast

- Committed to data integrity
- Uncompromisingly fast
- Built by and for extensive parallel branching and collaboration

- Committed to data integrity
- Uncompromisingly fast
- Built by and for extensive parallel branching and collaboration
- Free and open source

- For any kind of data

- For any kind of data
- For any kind of "knowledge worker"

- For any kind of data
- For any kind of "knowledge worker"
- For any degree of collaboration

- For any kind of data
- For any kind of "knowledge worker"
- For any degree of collaboration
- For any operating system

- You choose what files to ***add***

- You choose what files to ***add***

- You choose what files to ***add***
- You choose when to ***commit*** which version

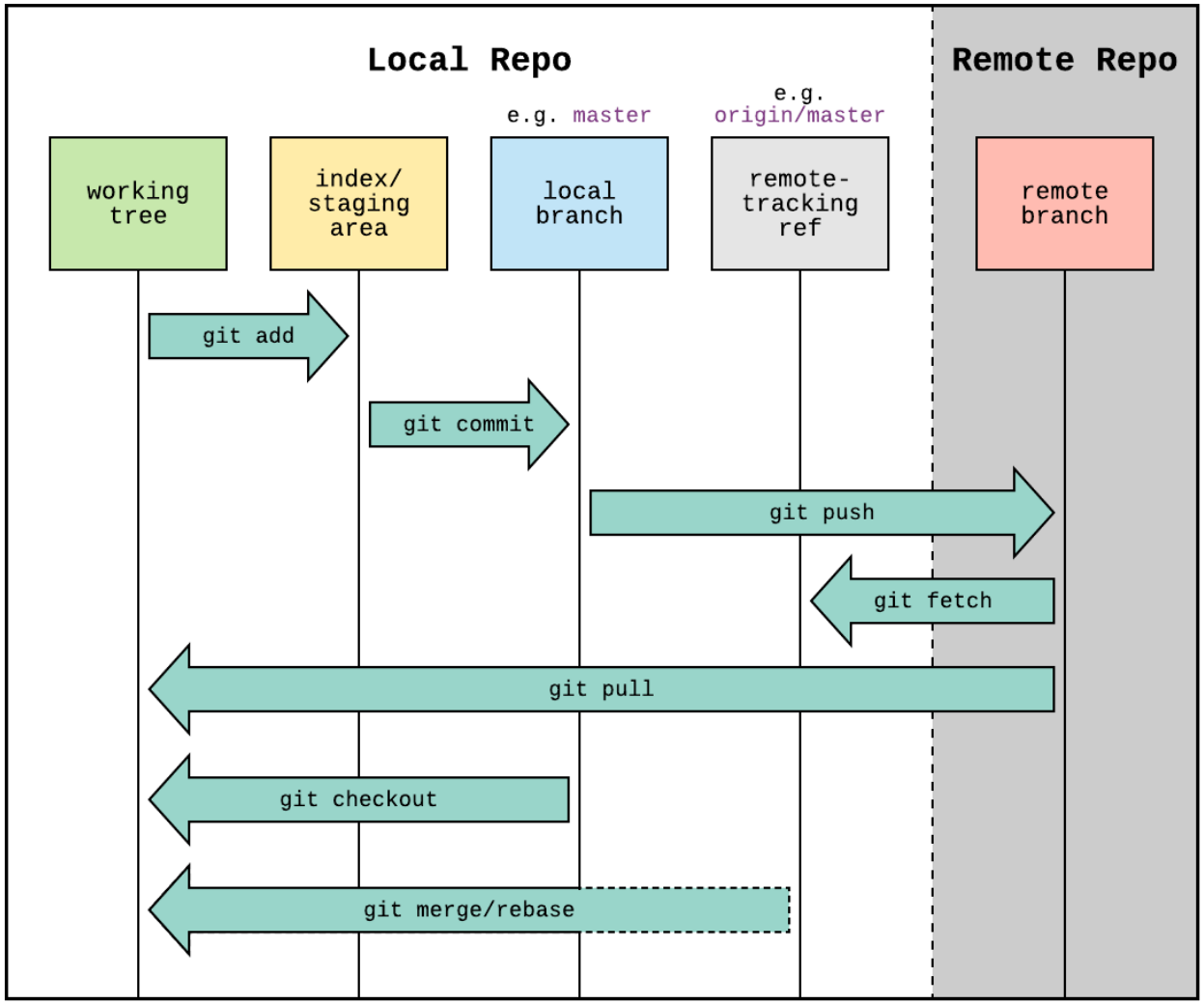
- You choose what files to ***add***
- You choose when to ***commit*** which version

- You choose what files to ***add***
- You choose when to ***commit*** which version
- You choose which previous versions to ***revert/restore*** to

- You choose what files to ***add***
- You choose when to ***commit*** which version
- You choose which previous versions to ***revert/restore*** to

- You choose what files to ***add***
- You choose when to ***commit*** which version
- You choose which previous versions to ***revert/restore*** to
- You choose when to synchronize with a ***push/pull***

- You choose what files to ***add***
- You choose when to ***commit*** which version
- You choose which previous versions to ***revert/restore*** to
- You choose when to synchronize with a ***push/pull***


Local Repo
Remote Repo
e.g. master
e.g. origin/master
working tree
index/ staging area
local branch
remote-tracking ref
remote branch
git add
git commit
git push
git fetch
git pull
git checkout
git merge/rebase

- You can ***fork*** an existing project

- You can ***fork*** an existing project

- You can ***fork*** an existing project
- You can ***branch*** and isolate parts of a project

- You can ***fork*** an existing project
- You can ***branch*** and isolate parts of a project

- You can ***fork*** an existing project
- You can ***branch*** and isolate parts of a project
- You can ***merge*** with other branches

- You can ***fork*** an existing project
- You can ***branch*** and isolate parts of a project
- You can ***merge*** with other branches

- You can ***fork*** an existing project
- You can ***branch*** and isolate parts of a project
- You can ***merge*** with other branches
- You can store your repository on a ***remote*** server

- You can ***fork*** an existing project
- You can ***branch*** and isolate parts of a project
- You can ***merge*** with other branches
- You can store your repository on a ***remote*** server

GIT IN ACTION: BRANCHING!

Let's say you're writing up your discussion...

Let's say you're writing up your discussion...

```
Our results prove that people use their brains while listening
```

Let's say you're writing up your discussion...

```
Our results prove that people use their brains while listening
```

Unsure about this radical statement? Branch!

Let's say you're writing up your discussion...

```
Our results prove that people use their brains while listening
```

Unsure about this radical statement? Branch!

```
1 # Let's see the name of the current branch
2 git branch --show-current
```

Let's say you're writing up your discussion...

```
Our results prove that people use their brains while listening
```

Unsure about this radical statement? Branch!

```
1 # Let's see the name of the current branch
2 git branch --show-current
```

```
3 * main
```

Let's say you're writing up your discussion...

```
Our results prove that people use their brains while listening
```

Unsure about this radical statement? Branch!

```
1 # Let's see the name of the current branch
2 git branch --show-current
```

```
3 * main
```

```
6 # Let's branch to mark this version and switch to the new br
7 git branch discussion-revision
```

Let's see what branch we're on now...

Let's see what branch we're on now...

```
1 git branch --show-current
```

Let's see what branch we're on now...

```
1 git branch --show-current
```

```
2   main
3 * discussion-revision
```

Let's see what branch we're on now...

```
1 git branch --show-current
```

```
2   main
3 * discussion-revision
```

You've successfully branched to discussion-revision

Let's see what branch we're on now...

```
1 git branch --show-current
```

```
2   main
3 * discussion-revision
```

You've successfully branched to discussion-revision

You meet your supervisor.

You meet your supervisor.

They suggest you "soften" your wording.

You meet your supervisor.

They suggest you "soften" your wording.

```
Our results prove that people use their brains while listening
```

```
Our results suggest that people use their brains while listeni
```

You meet your supervisor.

They suggest you "soften" your wording.

```
Our results prove that people use their brains while listening
```

```
Our results suggest that people use their brains while listeni
```

You commit your changes...

```
1 git commit -m "words softened"
```

You meet your supervisor.

They suggest you "soften" your wording.

```
Our results prove that people use their brains while listening
```

```
Our results suggest that people use their brains while listeni
```

You commit your changes...

```
1 git commit -m "words softened"
```

These changes are only committed to the discussion-revision branch.

You meet your supervisor.

They suggest you "soften" your wording.

```
Our results prove that people use their brains while listening
```

```
Our results suggest that people use their brains while listeni
```

You commit your changes...

```
1 git commit -m "words softened"
```

These changes are only committed to the discussion-revision branch.

On branch main

Your discussion reads...

```
Our results prove that people use their brains while listening
```

On branch main

Your discussion reads...

```
Our results prove that people use their brains while listening
```

Another day, another meeting...

Another day, another meeting...

Your other supervisor suggests you avoid certainty.

Another day, another meeting...

Your other supervisor suggests you avoid certainty.

```
Our results suggest that people use their brains while listeni
```

```
Our results suggest that people might use their brains while l
```

Another day, another meeting...

Your other supervisor suggests you avoid certainty.

```
Our results suggest that people use their brains while listeni
```

```
Our results suggest that people might use their brains while l
```

You commit your changes...

```
1 git commit -m "certainty avoided"
```

On branch main

Your discussion still reads...

```
Our results prove that people use their brains while listening
```

On branch main

Your discussion still reads...

```
Our results prove that people use their brains while listening
```

A week passes... 

A week passes... 

The first supervisor now suggests that you use more assertive language to better sell your cool results!

A week passes... 

The first supervisor now suggests that you use more assertive language to better sell your cool results!

Suddenly, an idea...

A week passes... 📅

The first supervisor now suggests that you use more assertive language to better sell your cool results!

Suddenly, an idea...

```
1 git branch --show-current
```

A week passes... 🗓️

The first supervisor now suggests that you use more assertive language to better sell your cool results!

Suddenly, an idea...

```
1 git branch --show-current
```

```
2   main
3 * discussion-revision
```

A week passes... 🗓️

The first supervisor now suggests that you use more assertive language to better sell your cool results!

Suddenly, an idea...

```
1 git branch --show-current
```

```
2   main
3 * discussion-revision
```

```
6 git checkout main
```

Back to where you started! 

Back to where you started! 


```
1 git branch --show-current
```

Back to where you started! 


```
1 git branch --show-current
```

```
2 * main
3   discussion-revision
```

Back to where you started! 


```
1 git branch --show-current
```

```
2 * main
3   discussion-revision
```

Although changes are still present in the discussion-revision branch, your discussion reads...

Back to where you started! 


```
1 git branch --show-current
```

```
2 * main
3   discussion-revision
```

Although changes are still present in the discussion-revision branch, your discussion reads...

Back to where you started! 


```
git branch --show-current
```

```
* main
  discussion-revision
```

Although changes are still present in the discussion-revision branch, your discussion reads...

```
Our results prove that people use their brains while listening
```

Another week passes... 

Another week passes... 

You meet both your supervisors!

Another week passes... 

You meet both your supervisors!

They first look at you...

Another week passes... 

You meet both your supervisors!

They first look at you...

Then to the discussion...

Another week passes... 

You meet both your supervisors!

They first look at you...

Then to the discussion...

Then to each other...

Another week passes... 

You meet both your supervisors!

They first look at you...

Then to the discussion...

Then to each other...

The more senior one turns to you and says:

Another week passes... 🗓️

You meet both your supervisors!

They first look at you...

Then to the discussion...

Then to each other...

The more senior one turns to you and says:
"Looks good! 👍"

Another week passes... 

You meet both your supervisors!

They first look at you...

Then to the discussion...

Then to each other...

The more senior one turns to you and says:
"Looks good! 👍"

THANK YOU GIT!

Open your computer's command line interface

You should have Git installed!

Let's create a local Git repository

Let's create a local Git repository

```
# First tell Git who you are
git config --global user.name Orhun Ulusahin
git config --global user.email orhunulusahin@pm.me
```

Let's create a local Git repository

```
# First tell Git who you are
git config --global user.name Orhun Ulusahin
git config --global user.email orhunulusahin@pm.me
```

Note that these are not login credientials for any services. The username is for crediting commits and the e-mail is for allowing collaborators to reach you.

Let's create a local Git repository

Let's create a local Git repository

```
# Create a new git repository in a new folder
git init my_repo

# Navigate into the new folder
cd my_repo

# Tell Git to track all current files in the folder
git add .

# Commit your first version with a message!
git commit -m "First commit!"
```

Let's create a local Git repository

```
# Create a new git repository in a new folder
git init my_repo

# Navigate into the new folder
cd my_repo

# Tell Git to track all current files in the folder
git add .

# Commit your first version with a message!
git commit -m "First commit!"
```

Let's create a local Git repository

```
# Create a new git repository in a new folder
git init my_repo

# Navigate into the new folder
cd my_repo

# Tell Git to track all current files in the folder
git add .

# Commit your first version with a message!
git commit -m "First commit!"
```

Let's create a local Git repository

```
# Create a new git repository in a new folder
git init my_repo

# Navigate into the new folder
cd my_repo

# Tell Git to track all current files in the folder
git add .

# Commit your first version with a message!
git commit -m "First commit!"
```

Let's create a local Git repository

```
# Create a new git repository in a new folder
git init my_repo

# Navigate into the new folder
cd my_repo

# Tell Git to track all current files in the folder
git add .

# Commit your first version with a message!
git commit -m "First commit!"
```

Now create a new file in your repository.

Bonus points for doing it in the command line!

Now create a new file in your repository.

Bonus points for doing it in the command line!

```
# Create a markdown (.md) file with some text in it
echo Important information! > readme.md
```

Let's see what Git thinks about it!

```
1 git status
```

Let's see what Git thinks about it!

```
1 git status
```

Git detects that there are new untracked files!

```
 2 On branch master
 3
 4 No commits yet
 5
 6 Untracked files:
 7 (use "git add <file>..." to include in what will be committ
 8   readme.md
 9
10 nothing added to commit but untracked files present (use "g
```

Let's see what Git thinks about it!

```
1 git status
```

Git detects that there are new untracked files!

```
 2 On branch master
 3
 4 No commits yet
 5
 6 Untracked files:
 7 (use "git add <file>..." to include in what will be committ
 8   readme.md
 9
10 nothing added to commit but untracked files present (use "g
```

Let's see what Git thinks about it!

```
1 git status
```

Git detects that there are new untracked files!

```
 2 On branch master
 3
 4 No commits yet
 5
 6 Untracked files:
 7 (use "git add <file>..." to include in what will be committ
 8   readme.md
 9
10 nothing added to commit but untracked files present (use "g
```

Let's see what Git thinks about it!

```
1 git status
```

Git detects that there are new untracked files!

```
 2 On branch master
 3
 4 No commits yet
 5
 6 Untracked files:
 7 (use "git add <file>..." to include in what will be committ
 8   readme.md
 9
10 nothing added to commit but untracked files present (use "g
```

Git looks for a special file called ".gitignore" which instructs Git to ignore files!

Git looks for a special file called ".gitignore" which instructs Git to ignore files!

```
# Create a folder you don't want to track
mkdir secrets
# Create '.gitignore' and tell Git to ignore any 'secrets' f
echo secrets/ > .gitignore
```

Git looks for a special file called ".gitignore" which instructs Git to ignore files!

```
1 # Create a folder you don't want to track
2 mkdir secrets
3 # Create '.gitignore' and tell Git to ignore any 'secrets' f
4 echo secrets/ > .gitignore
```

Git looks for a special file called ".gitignore" which instructs Git to ignore files!

```
# Create a folder you don't want to track
mkdir secrets
# Create '.gitignore' and tell Git to ignore any 'secrets' f
echo secrets/ > .gitignore
```

Git looks for a special file called ".gitignore" which instructs Git to ignore files!

```
# Create a folder you don't want to track
mkdir secrets
# Create '.gitignore' and tell Git to ignore any 'secrets' f
echo secrets/ > .gitignore
```

If you run git status again, you'll see that Git won't report on the secrets folder.

Git looks for a special file called ".gitignore" which instructs Git to ignore files!

```
1 # Create a folder you don't want to track
2 mkdir secrets
3 # Create '.gitignore' and tell Git to ignore any 'secrets' f
4 echo secrets/ > .gitignore
```

If you run git status again, you'll see that Git won't report on the secrets folder.

- Personal data (e.g., addresses, passwords, voice recordings)

- Personal data (e.g., addresses, passwords, voice recordings)
- Any other "data that shouldn't leave work"

- Personal data (e.g., addresses, passwords, voice recordings)
- Any other "data that shouldn't leave work"
- Large files (e.g., hour-long video stimuli)

- Personal data (e.g., addresses, passwords, voice recordings)
- Any other "data that shouldn't leave work"
- Large files (e.g., hour-long video stimuli)
- Hidden files

Let's go online with GitHub now!

Let's go online with GitHub now!

But before we do, let's remember:
Git != GitHub

Let's go online with GitHub now!

But before we do, let's remember:
Git != GitHub

Git is an open-source version control system.
GitHub is a private hosting service specializing in hosting Git repositories.

Go to GitHub and create a new (empty) repository!

Go to GitHub and create a new (empty) repository!

Then, return to the command line.

Go to GitHub and create a new (empty) repository!

Then, return to the command line.

```
# Declare your local repository's remote location
git remote add origin https://github.com/username/reponame.g
# Verify the remote location
git remote -v
```

Go to GitHub and create a new (empty) repository!

Then, return to the command line.

```
# Declare your local repository's remote location
git remote add origin https://github.com/username/reponame.g
# Verify the remote location
git remote -v
```

Go to GitHub and create a new (empty) repository!

Then, return to the command line.

```
# Declare your local repository's remote location
git remote add origin https://github.com/username/reponame.g
# Verify the remote location
git remote -v
```

Now PUSH!

Now PUSH!

```
# Push your changes to the remote repository
git push -u origin main
```

Now PUSH!

```
1 # Push your changes to the remote repository
2 git push -u origin main
```

After the first push, you can use just git push by itself.

Now PUSH!

```
# Push your changes to the remote repository
git push -u origin main
```

After the first push, you can use just git push by itself.

Return to GitHub and go to your repository!

Return to GitHub and go to your repository!

CONGRATULATIONS! 

Return to GitHub and go to your repository!

CONGRATULATIONS! 🎉

Your repository is now online! 👏

Your repository is now online! 👏

Your repository is now online! 👏

To sync on another machine or to catch up with collaborators...

Your repository is now online! 👏

To sync on another machine or to catch up with collaborators...

```
1 git fetch # Will download changes from the remote
```

Your repository is now online! 👏

To sync on another machine or to catch up with collaborators...

```
1 git fetch # Will download changes from the remote
```

```
2 git merge # Will implement changes to the local repo
```

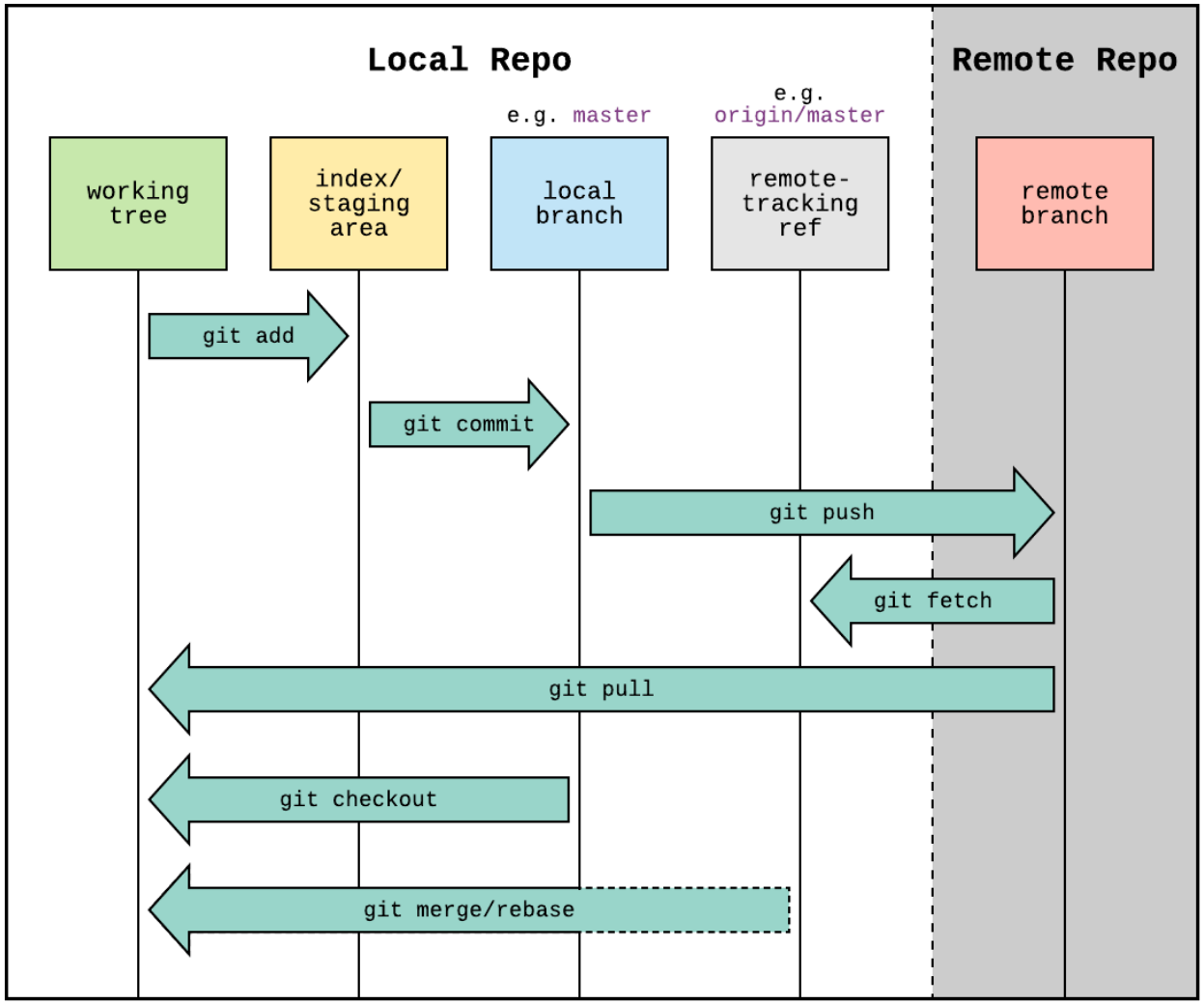
Your repository is now online! 👏

To sync on another machine or to catch up with collaborators...

```
1 git fetch # Will download changes from the remote
```

```
2 git merge # Will implement changes to the local repo
```

```
3 git pull # Will run git fetch and git merge together
```


Local Repo
Remote Repo
e.g. master
e.g. origin/master
working tree
index/ staging area
local branch
remote-tracking ref
remote branch
git add
git commit
git push
git fetch
git pull
git checkout
git merge/rebase

TUTORIAL PART OVER!

TUTORIAL PART OVER!

A few small things before we finish...

- Commit early, commit often

- Commit early, commit often
- Test first, commit later

- Commit early, commit often
- Test first, commit later
- Write good commit messages

- Commit early, commit often
- Test first, commit later
- Write good commit messages
- Branch before you try something crazy

- Remember that private repositories exist (and use them appropriately)

- Remember that private repositories exist (and use them appropriately)
- Pay attention to licenses when you fork a repository

- Remember that private repositories exist (and use them appropriately)
- Pay attention to licenses when you fork a repository
- Enable two-factor authentication

Git is not a substitute for backups, although it can help. You should always try to use the 3-2-1 backup method:

Git is not a substitute for backups, although it can help. You should always try to use the 3-2-1 backup method:

- 3 copies of data

Git is not a substitute for backups, although it can help. You should always try to use the 3-2-1 backup method:

- 3 copies of data
- On 2 different media

Git is not a substitute for backups, although it can help. You should always try to use the 3-2-1 backup method:

- 3 copies of data
- On 2 different media
- With 1 copy being offsite

Git is not a substitute for backups, although it can help. You should always try to use the 3-2-1 backup method:

- 3 copies of data
- On 2 different media
- With 1 copy being offsite

(A remote repository helps with all of these)

- The Git wiki

- The Git wiki
- The Git visualizer

- The Git wiki
- The Git visualizer
- The Git manual

- The Git wiki
- The Git visualizer
- The Git manual
- GitHub docs

- The Git wiki
- The Git visualizer
- The Git manual
- GitHub docs
- Git GUIs

Feel free to ask questions: Here and now, or later on Slack!

Feel free to ask questions: Here and now, or later on Slack!

The assignment for this lecture will be shared later today.