Git, HTML, CSS

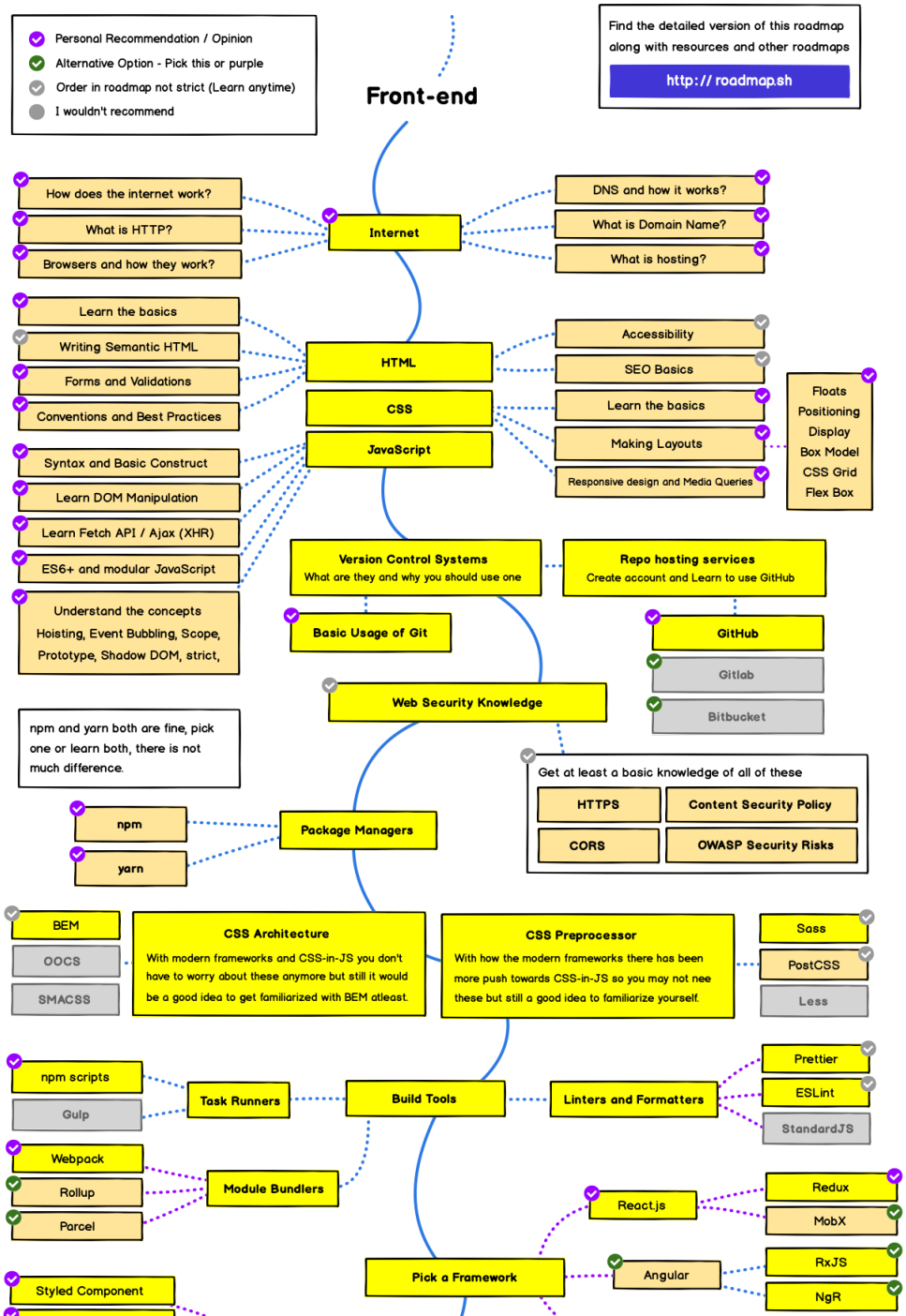
Workshops every Monday 4:10-5:30 CB 2400

	Date	Topics	Concepts
1	1/20	Overview Git HTML and CSS	Roadmap Basic concepts Building layouts Create basic project home page
2	1/27	Advanced HTML and CSS	Flexbox Media queries Bootstrap Build and style project home page
3	2/3	Introduction to React	Basic concepts JSX Components State Lifecycle methods Build assignment 1 with React in CodeSandbox
4	2/10	Project Setup TypeScript Build Tools	Create React App Module bundlers (Webpack) Linters (ESLint) Formatters (Prettier)

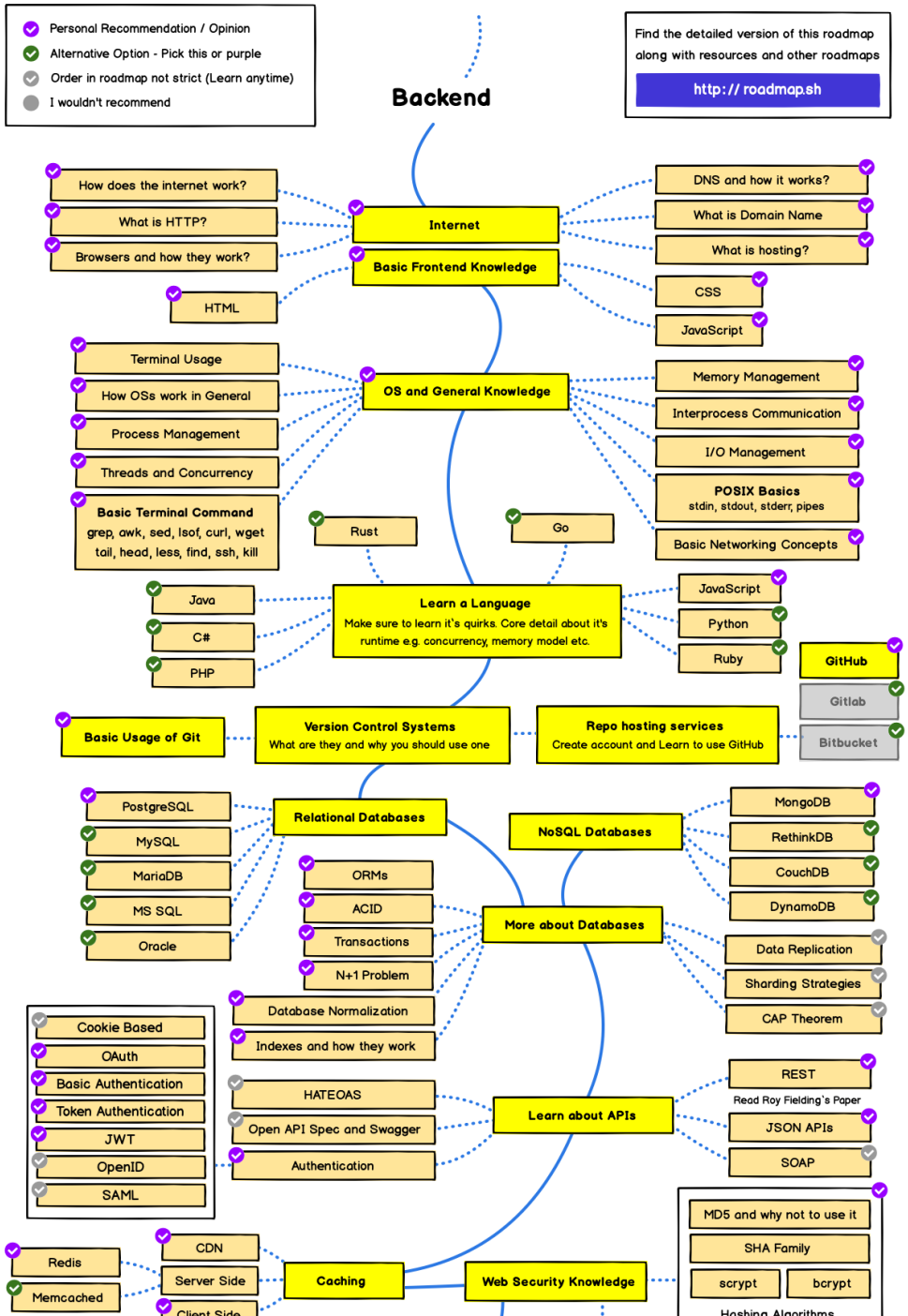
What's the point of these workshops?

- Learn modern web development
- Build a project
- Be good

Frontend roadmap



Backend roadmap



You should come if:

- You want to learn how to build a website
- You want a job
- You have nothing better to do for 1 hour

Frontend vs Backend

- Frontend is what users see and interact with
- HTML, CSS, Javascript
- Backend is everything else

HTML	Structure of web page (nouns)
CSS	Style of HTML (adjectives)
Javascript	Logic and interactivity of web page (verbs)

HTML

What is HTML?

- HyperText Markup Language
- Structure of web page

HTML Syntax

```
<tag attribute="value">content</tag>
```

HTML Structure

```
<!DOCTYPE html>
<html>
  <head>
    <title>Document</title>
  </head>
  <body>
    <h1>Hello</h1>
  </body>
</html>
```

HTML Structure

<!DOCTYPE html>	Indicates use of HTML5
<html>	Root element
<head>	Information about document
<title>	Title in browser tab
<body>	Content to be displayed

Block and Inline Elements

- **Block elements** start on a new line and take up full width of parent
- **Inline elements** don't start on a new line and only take up width of content

Heading

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

Paragraph and Link

```
<p>This is paragraph</p>
<a href="url">This is a link</a>
<p>Links are inline <a href="url">elements</a></p>
```

This is paragraph

This is a link

Links are inline elements

```
<h2>Headings are block elements</h2>
<p>So are paragraphs. <em>These</em> are <strong>inline</strong></p>
```

Headings are block elements

So are paragraphs. *These* are **inline**

Unordered List

```
<ul>
  <li>Item 1</li>
  <li>Item 2</li>
</ul>
```

- Item 1
- Item 2

Ordered List

```
<ol>
  <li>Item 1</li>
  <li>Item 2</li>
</ol>
```

1. Item 1
2. Item 2

Div and Span

- Containers to group elements together

```
<div>Divs are block elements</div>
<div>Spans are<span>inline</span></div>
```

Divs are block elements
Spans are inline

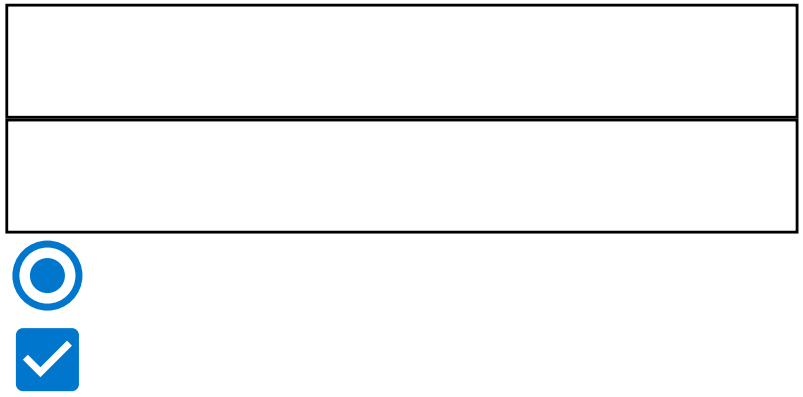
Attributes

- Add additional information to element

```
<div name="value">content</div>
```

Forms

```
<form>
  <input type="text" />
  <input type="password" />
  <input type="radio" />
  <input type="checkbox" />
</form>
```



Labels

```
<label for="example">Example</label>
<input type="text" id="example"/>
```

Example

Exercise

CSS

What is CSS?

- Cascading Stylesheets
- Style of the page

Syntax

```
selector {
  property: value;
  property: value;
}
```

Concepts

- Where do we write styles?
- How do we style elements?

Where do we write styles?

1. Inline
2. Style tags
3. Stylesheets

Inline

```
<p style="color: blue">Don't do this</p>
```

Don't do this

Style tags

```
<html>
  <head>
    <style>
      p: {
        color: blue
      }
    </style>
  </head>
```

Stylesheets

```
<html>
  <head>
    <link rel="stylesheet" href="styles.css">
  </head>
```

How do we style elements? Selectors

- **Type selector** elementname
- **Class selector** .classname
- **ID selector** #idname
- **Attribute selector** [attr=value]

Combining selectors

- div.classname
- div, span
- div span
- div > span
- div ~ span
- div + span

Specificity

- More specific selectors take precedence over less specific ones
- Selectors from least to most specific:
    1. **Type selector** elementname
    2. **Class selector** .classname
    3. **ID selector** #idname

Git

What is Git? Why do we use it?

- Distributed version control system
- Tracks changes in project files over time
- Coordinates work between multiple developers on project

Concepts

- Repositories
- Commits and Staging
- Branches

Repositories

- Project tracked by Git
- **Local:** Isolated repository stored on your own computer
- **Remote:** Stored on server (Github), share your project code with others

Staging and Commiting

- Commits are snapshots of your projects
- Staging adds files to be commited

Branches

- Individual timeline of project commits
- Master branch contains stable code
- Add new features in separate branches and merge

Download Git

https://git-scm.com/downloads

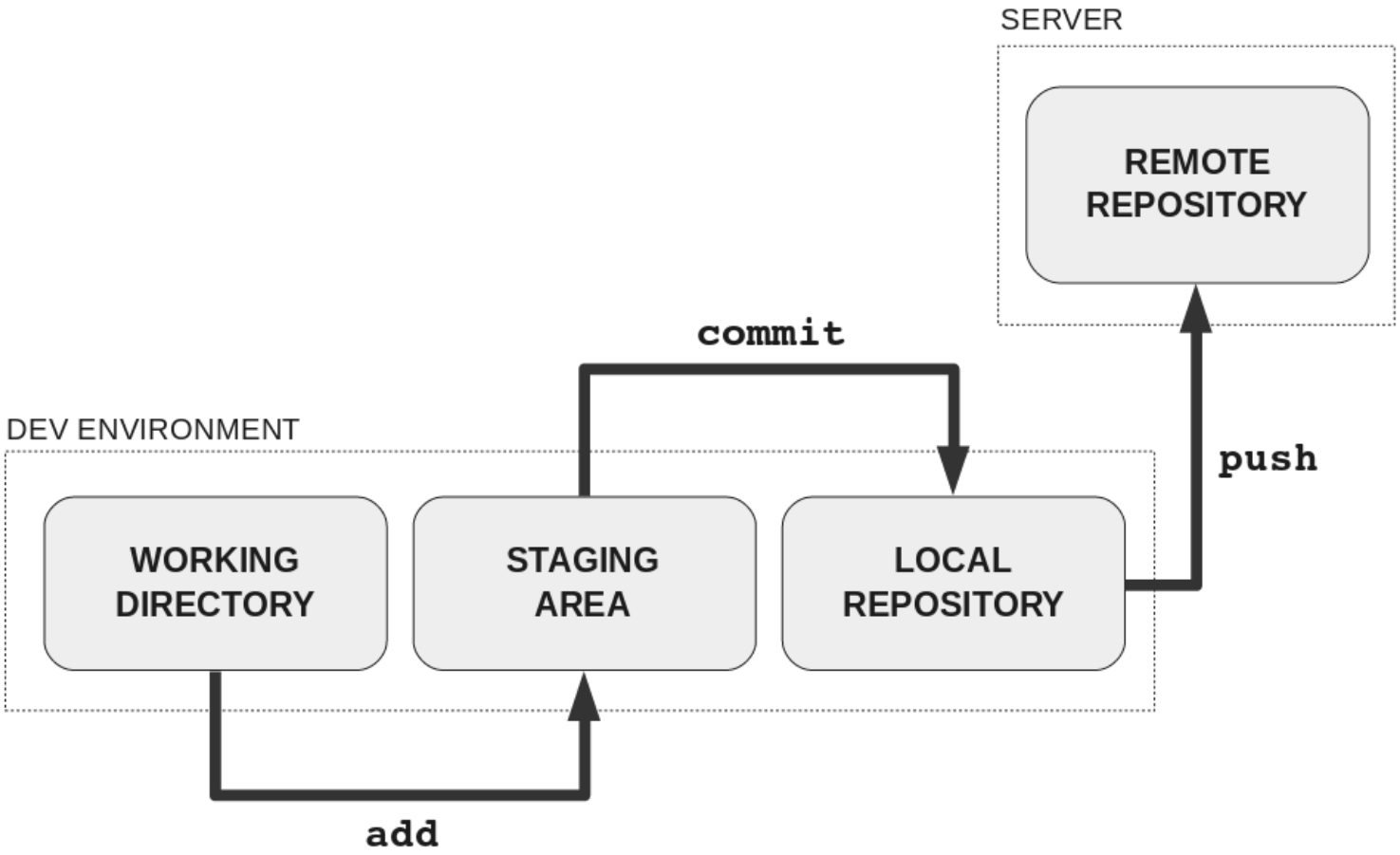
Setup

```
git config --global user.name "Your Name"
git config --global user.email “your@email.com”
```

Making a commit

```
git init
git add file.js
git commit -m "Commit message"
```

What's going on



Useful commands

```
git status
git log
```

Creating a branch

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

Merging changes

Merge changes from a different branch into current branch

```
git merge <name>
```

Exercise

1. Commit your HTML files
2. Create a new branch
3. Commit your CSS files
4. Merge your branch to master

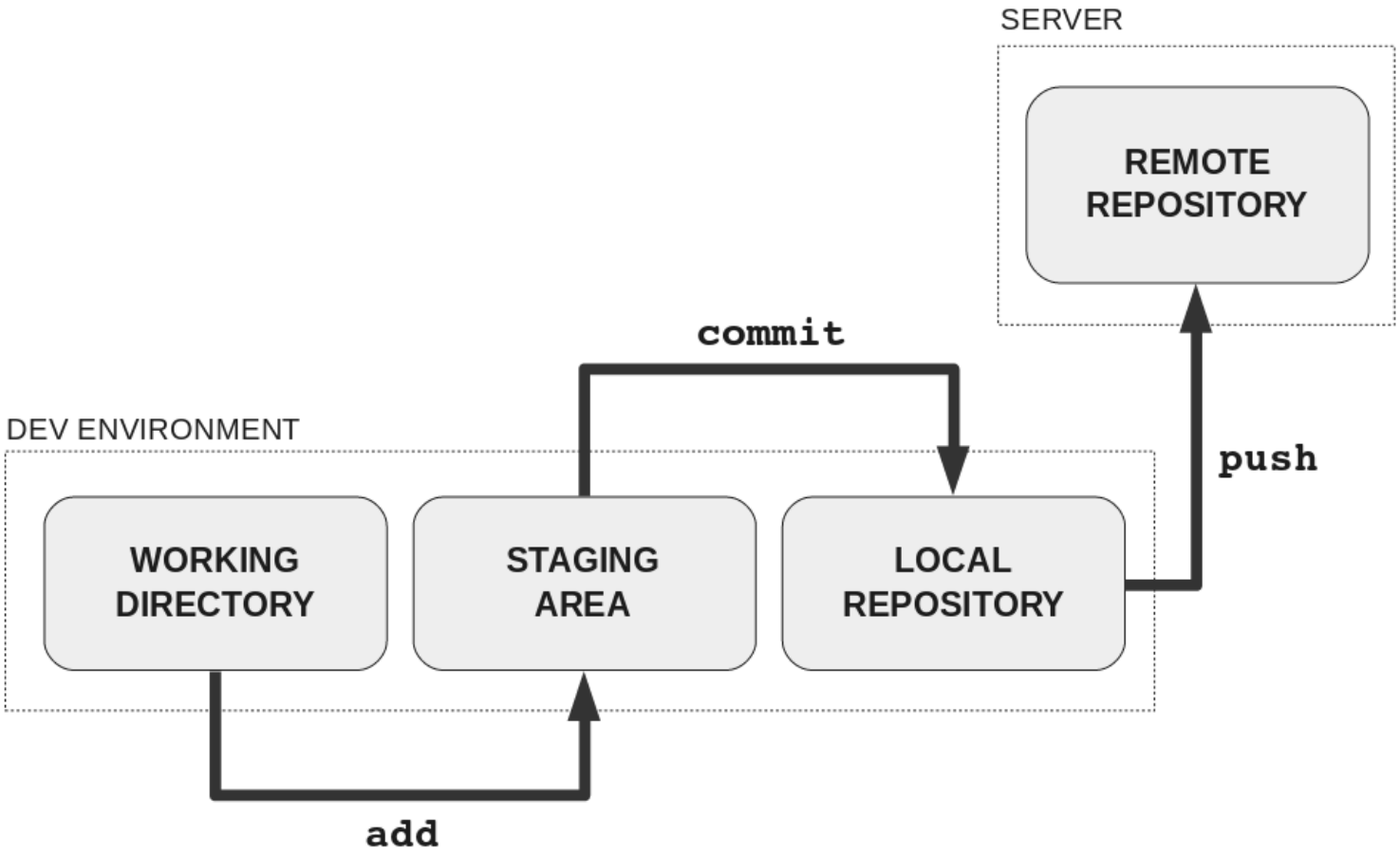
Github

- Hosting platform for Git repositories
- Makes it easy to collaborate and share code with others

Pushing our repo to Github

```
git remote add origin https://github.com/<username>/<repo>.git
git push -u origin master
```

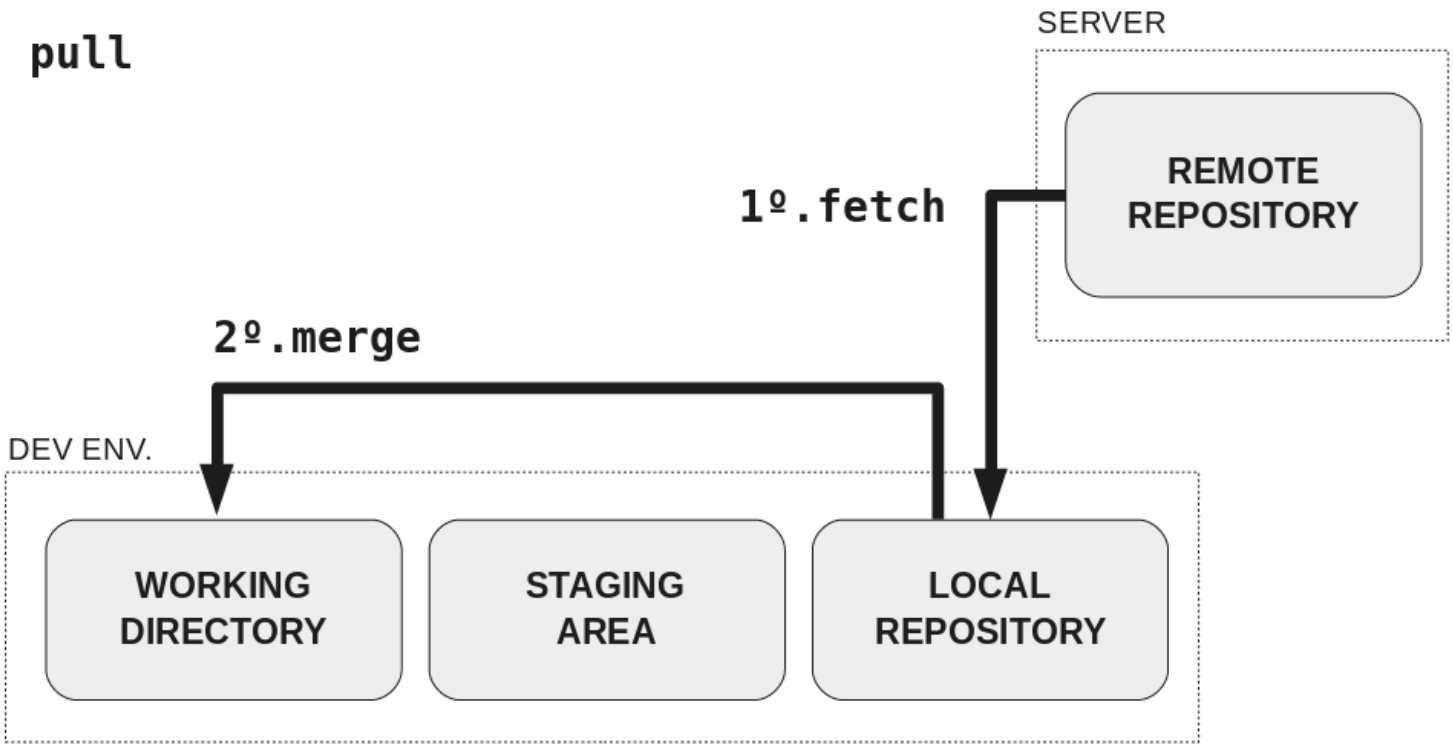
What's going on



Cloning our repo

```
git clone
git pull
```

What's going on



Branching and Pull requests

```
git branch <name>
git checkout <name>
git push -u origin <name>
```

Exercise

1. Push repository to Github
2. Create a new branch
3. Push branch to Github
4. Submit pull request to merge into master