Sofware Engineering

Session 2

Today: Building your development team

Lab

- Recap: technology stack, HTML, Git
- Docker
- JavaScript intro

Seminar

- Group forming!!!
- Group discussion
  - Code of conduct
  - Task board
  - Group name, group project

What happens in this module?

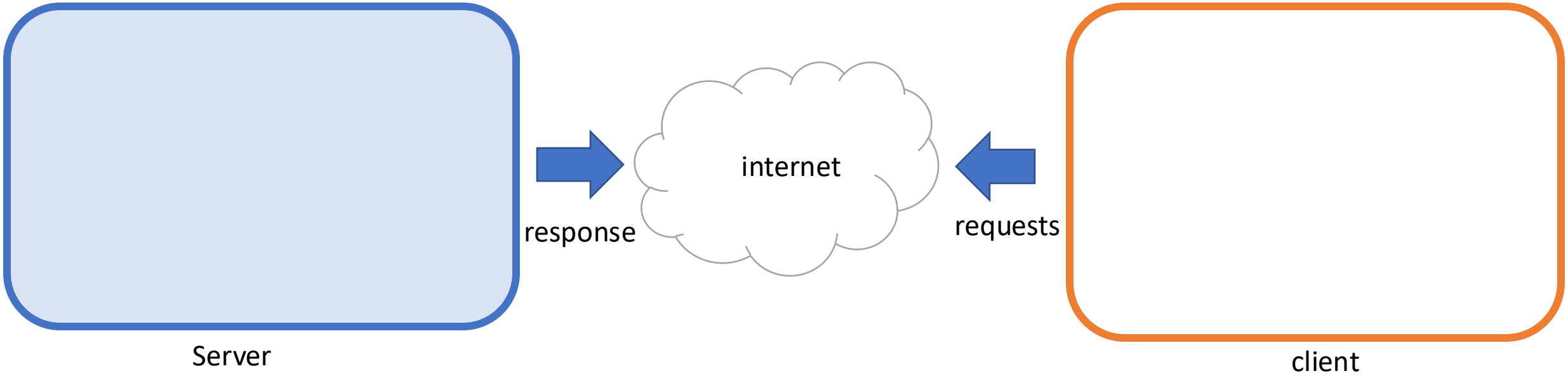
- You build a 'CRUD' web application, completely from scratch as a group project
- In the lab sessions, you are taught skills in:
  - **HTML**
  - **Front end Javascript**
  - Back end Javascript node.js / express
  - PUG templating engine
  - Connecting your code to a relational database
  - **GIT version control**
  - **Docker environment building**/management/deployment
  - Specification/ user research /**project management tools**

You will use these practical skills to build your project.

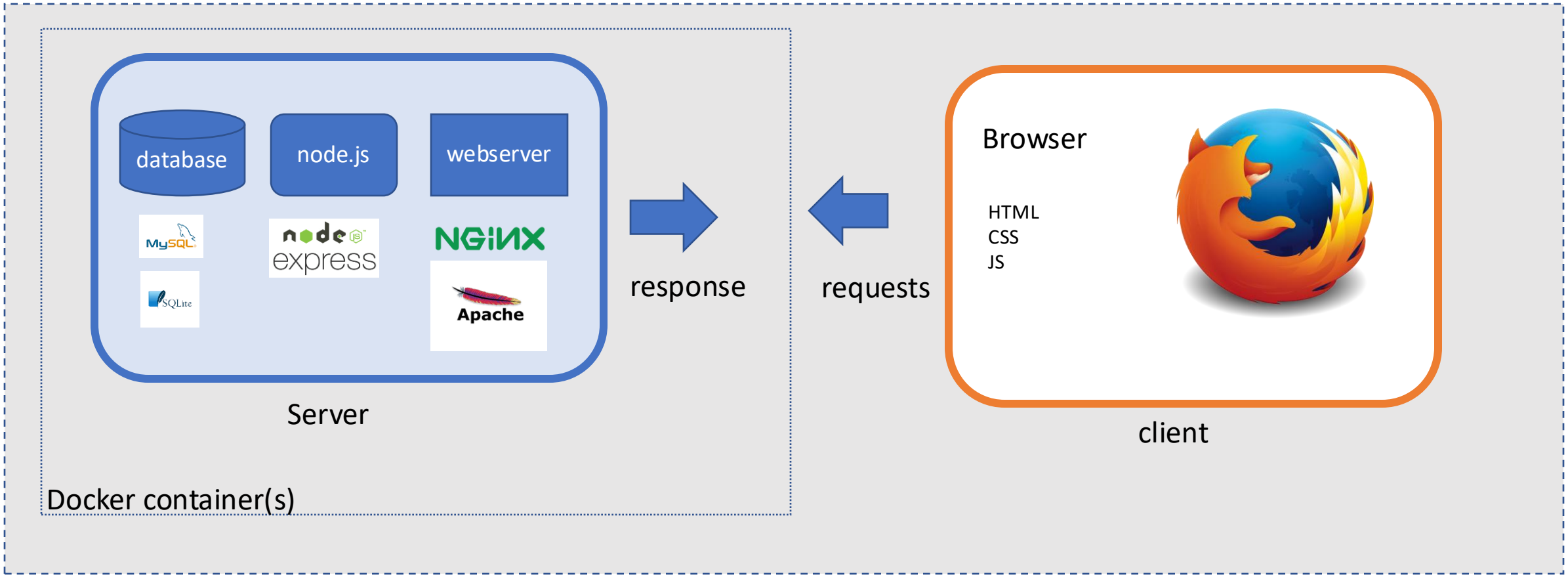
Recap

Web application technologies consist of software on the client (users) side, communicating over the internet with software on the server side.

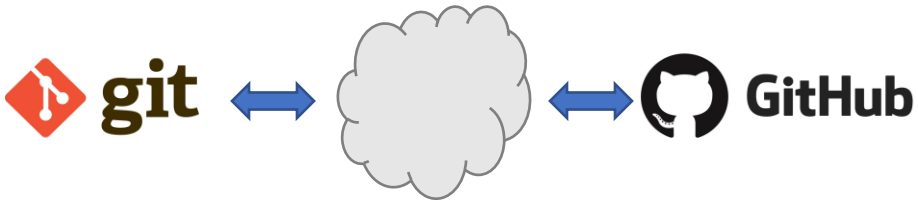
What software applications do you already know that are on the server or client side?



Development environments: When projects are in development, developers will have all the client and server software running on their own computer. Additionally they will use tools such as VS Code and Git to write and manage source code.



Visual studio code (IDE)



Recap

Last weeks lab:

HTML: https://codepen.io/lihashas/pen/gOEvWPr

GIT:

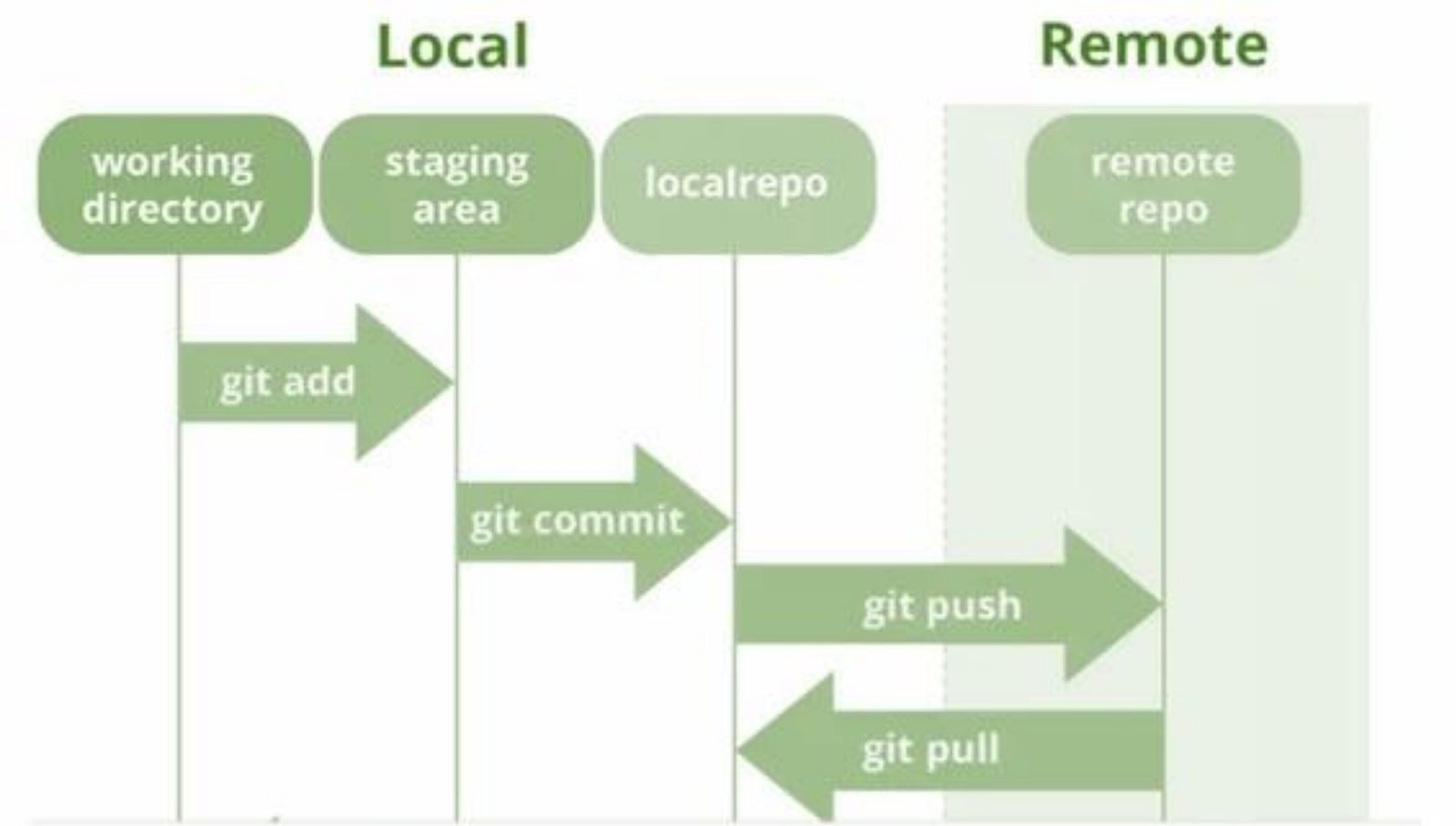
Clone a (remote) repository eg. github

Visual studio code (IDE)

git

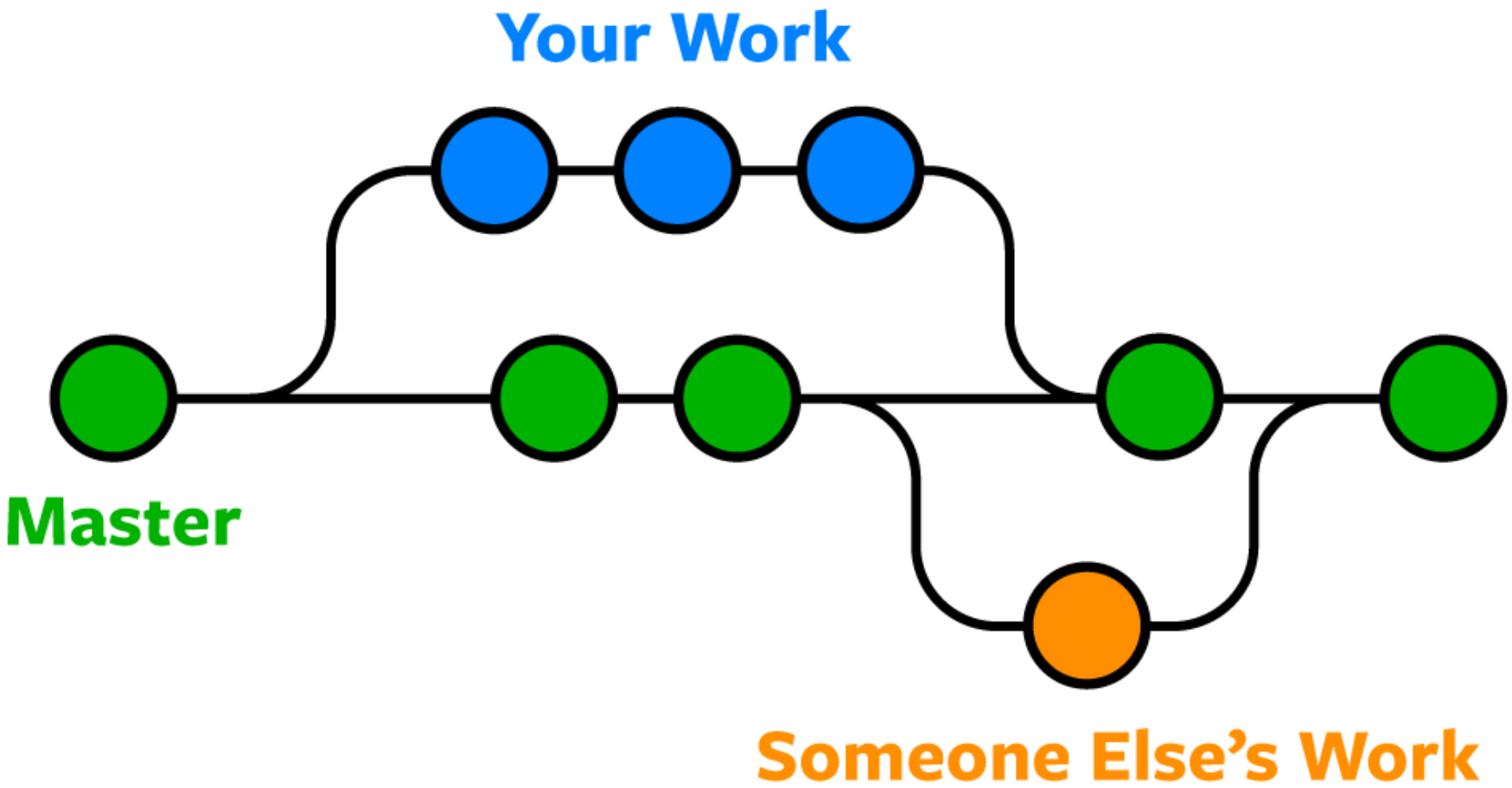
GitHub

Push and pull to the remote repo to share work with other developers



https://www.janbasktraining.com/blog/git-push-vs-git-push-origin/

Branching and merging

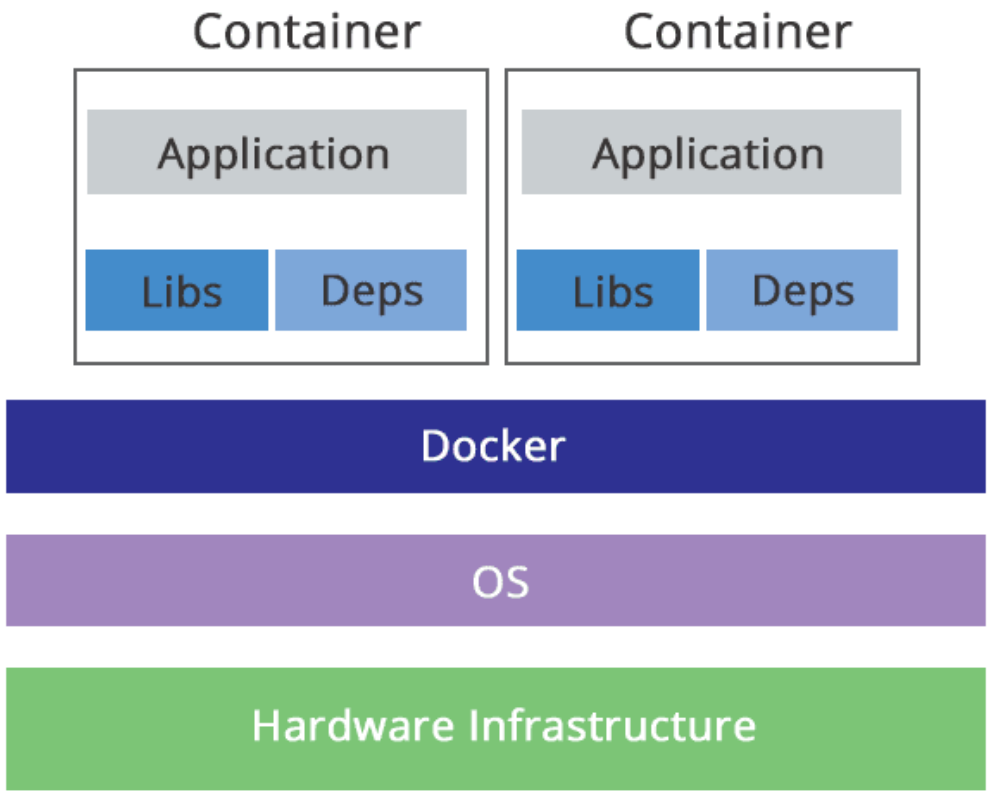


TODAY

- Building a development environment with Docker

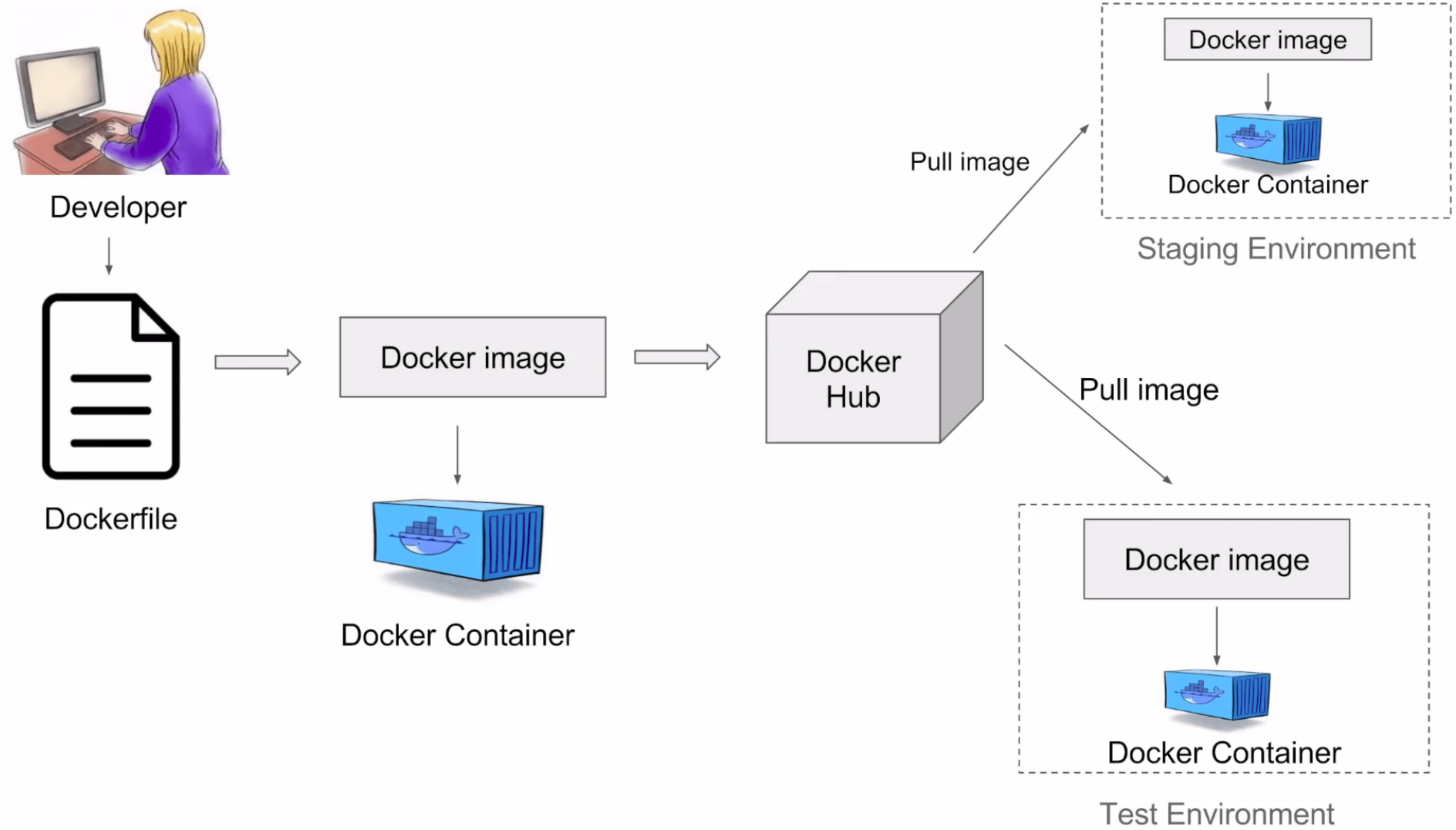
What is docker?

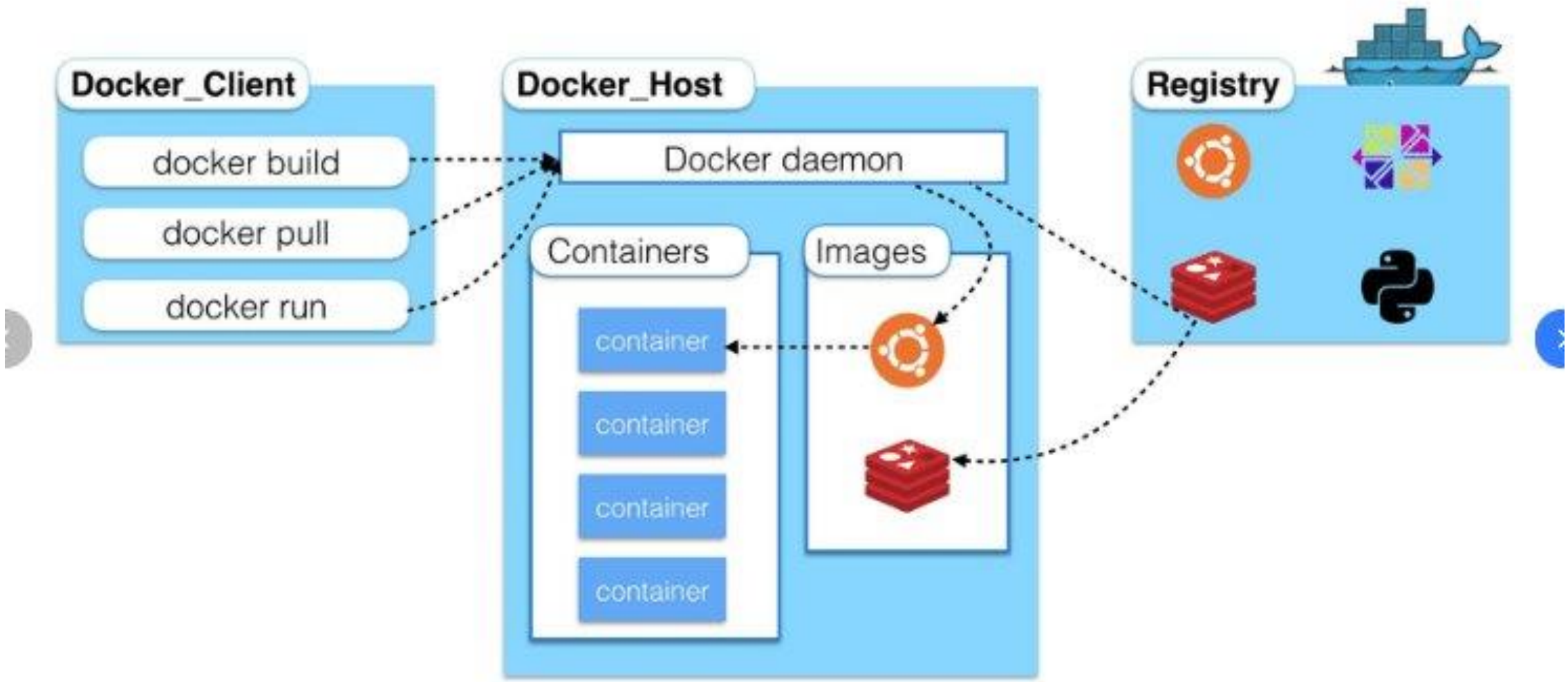
- An industry standard tool for ensuring that development, test and production environments are consistent
- A 'devOps' tool ie. Defines `infrastructure in code` / text based configuration files
- This can then be versioned, exchanged etc to create consistency
- Instead of installing software applications (such as MySQL) on your computer you 'pull' a container with the application installed, configured and ready to run and connect to it.



CONTAINERS

Containers are an abstraction at the app layer that packages code and dependencies together. Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space. Containers take up less space than VMs (container images are typically tens of MBs in size), can handle more applications and require fewer VMs and Operating systems.


Developer
Dockerfile
Docker image
Docker Container
Docker Hub
Pull image
Docker image
Docker Container
Staging Environment
Pull image
Docker image
Docker Container
Test Environment



High-level overview of Docker architecture.

Key components

- **Dockerfile**

"*A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.* "

- **docker-compose.yml**

"*Docker Compose is a tool for defining and running multi-container applications. It is the key to unlocking a streamlined and efficient development and deployment experience.*"

Typically – developer teams use git to store, share, track and version these files.

Ref: https://docs.docker.com/

Our 'scaffolding' files

A bespoke set of starter files for our full stack development. Contains:

- docker-compose.yml - to 'pull' and configure all the containers needed ie:
  - Base for node.js app
  - MySql
  - PhpMyAdmin
- dockerfile – to create an image for our custom node.js app

See: https://moodle.roehampton.ac.uk/mod/resource/view.php?id=2082832

Lab today

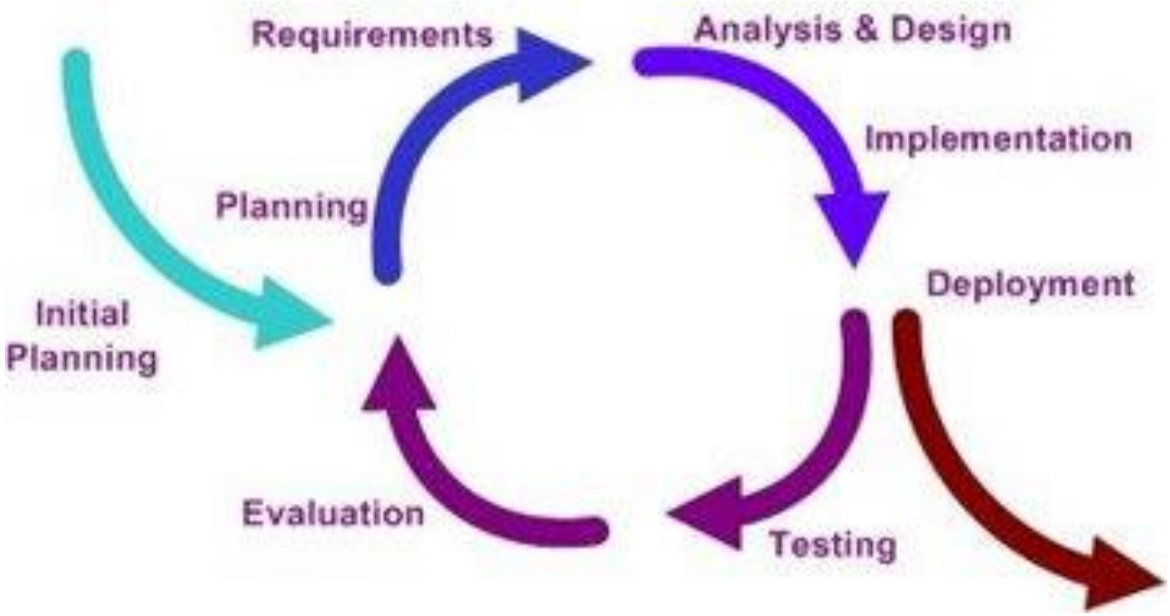
- Making sure you can run Docker (lab or your own laptop)
- Download and run the scaffolding files which will get you started (do not extract it on a onedrive location!)
- Check your connection to the database
- JavaScript intro and language fundamentals (frontend)

Break!!

Group project (Seminar sessions)

- The group project runs throughout the module and is your assessed work
- Work is delivered in 4 'sprints'. These are reviewed the week after the deadline and the next sprint is planned.
- You manage your work using industry standard tools
- Everyone should do a bit of everything, but group members will take specific responsibilities for certain roles.

Delivery of your group project will be divided into 4 sprints

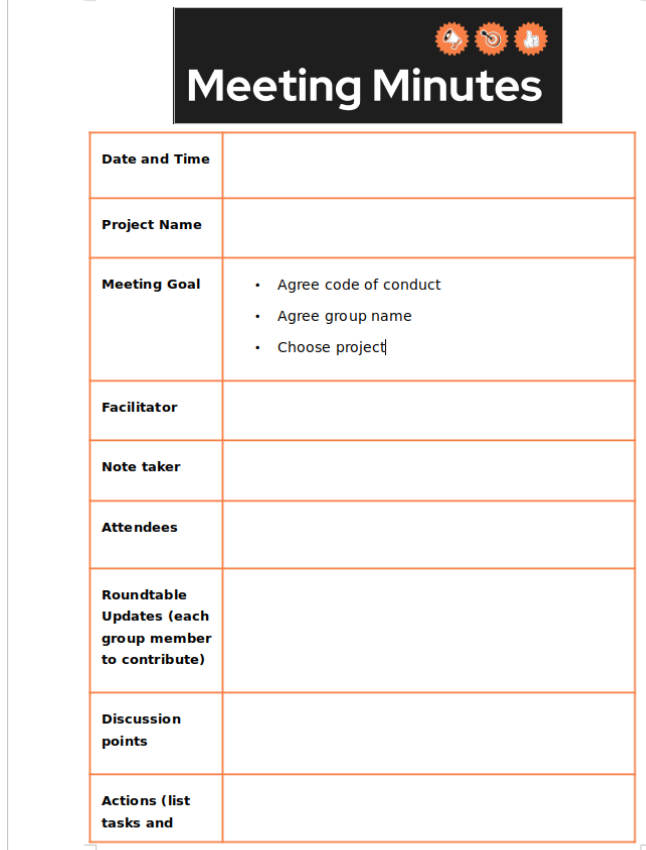


Sprint 1: proposal
Sprint 2: Designs
Sprint 3: First prototype / proof of concept
Sprint 4: MVP

Group Tasks Today

- Gather your group together Decide who will be your Scrum master today (ie. faciltates meeting)
- Decide who will be taking notes today
- Draft your code of conduct. Consider what is important to you and what will keep your project on track. Use the Mozilla code of conduct as a starting point: https://www.mozilla.org/en-US/about/governance/policies/participation/.
- Decide what additional points you should add that are important in our specific context. Eg.
  - Frequency of meetings
  - Gathering work in time for deadlines
  - Participation expectations: what do do if a team member does not particpate
  - Participation expectations: Attending lab sessions
- Talk about your team name and the topic you want to work on Review the sprint 1 tasks and decide on the steps you need to take to fulfill them.

At the end of each meeting the note taker should summarise the main points discussed, the action points going forward and confirm the date, time and place of the next meeting



Meeting Minutes

Date and Time	
Project Name	
Meeting Goal	• Agree code of conduct • Agree group name • Choose project
Facilitator	
Note taker	
Attendees	
Roundtable Updates (each group member to contribute)	
Discussion points	
Actions (list tasks and	