Building Full-Stack Applications

Presented by JJ McCauley – SU Class of 2026

Nov. 2025

Table of Contents

- What is a full-stack application?
  - Basic example
- How to build a frontend
- How to build a backend
  - Popular API/networking frameworks
- How to build a database
  - Choosing the right database
- What is a stack?
- How does this all tie together?
  - Example revisited
- What is "the cloud"?
- Other cool concepts
- How to start building

01

What is a Full-Stack App?

Any type of software that includes three major components:

What makes a full-stack app?

01

Frontend

The part that users interact with directly “client-side operations”

02

Backend

Handles business logic, APIs, “server-side operations”

03

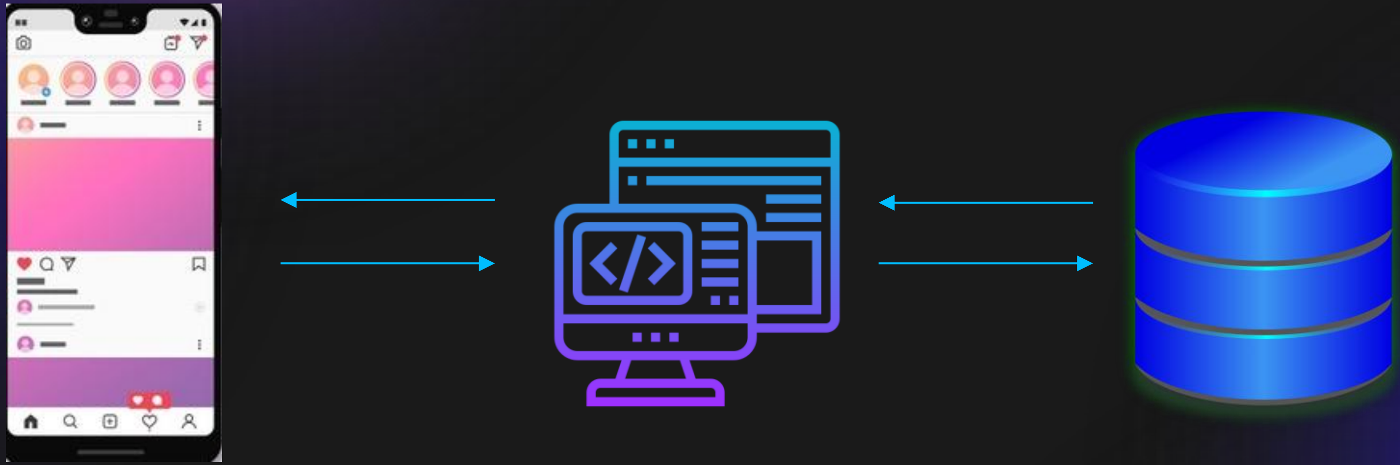
Database

Stores and retreives data

Example

Instagram – Social Media App

Instragram's full-stack components



Frontend

What the user sees – Profiles, Feed, Etc.

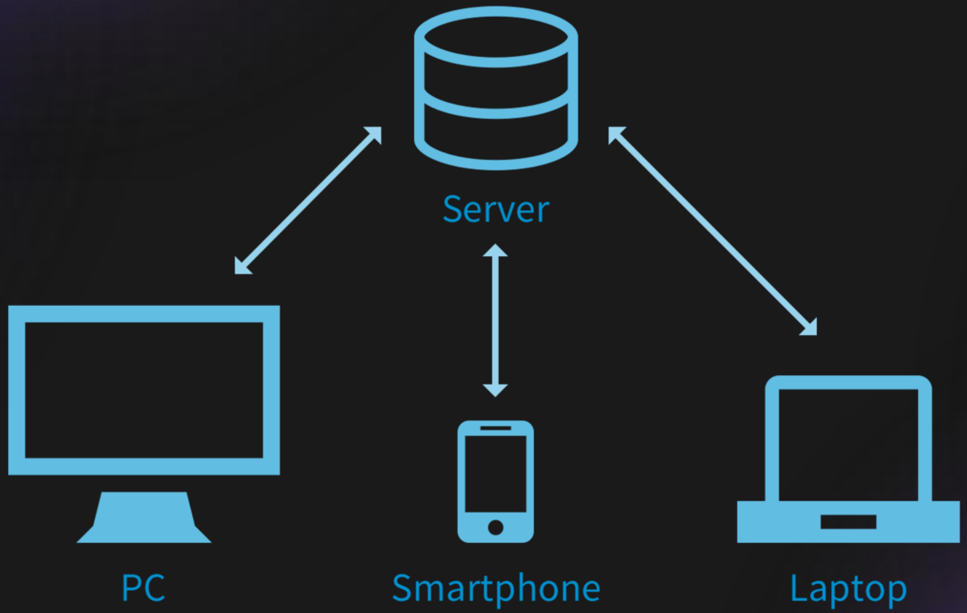
Backend

Reccomendation Engines, Network Trafficing, Etc.

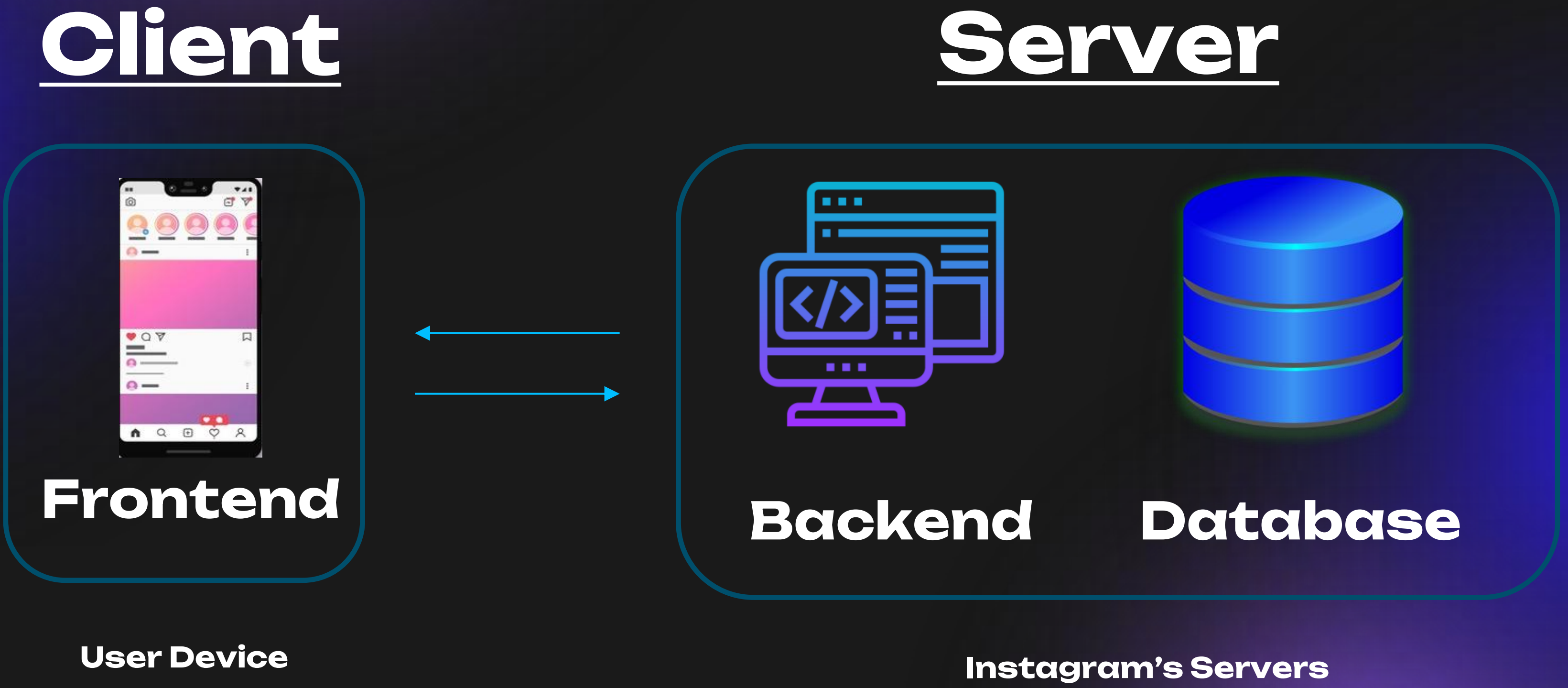
Database

User profiles, post, interactions, etc.

Client-Server Model



Instagram - Client-Server Model



02

Building a Frontend

Three types of frontends

Websites

- Web-based
- Built with HTML/CSS/JS

Desktop

- Built with GUI libraries

Mobile

- Built with mobile-specific GUI libraries

Web-based Frontends

HTML

Hyper-Text Markup Language

CSS

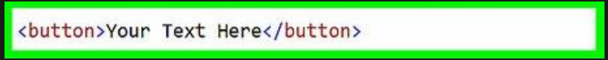
Cascade Style Sheets

JS

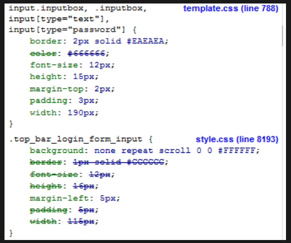
JavaScript

HTML/CSS/JS Example - Button

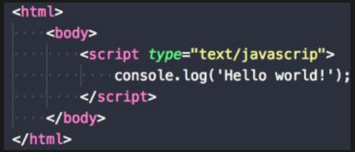
- **HTML**- Places the button on the screen
- CSS- Makes the button pretty
- JS- Makes the button do stuff



```
<button>Your Text Here</button>
```



```
input.inputbox, .inputbox,                template.css (line 788)
input[type="text"],
input[type="password"] {
    border: 2px solid #EAEAEA;
    ~~color~~: ~~#666666~~;
    font-size: 12px;
    height: 15px;
    margin-top: 2px;
    padding: 3px;
    width: 190px;
}
.top_bar_login_form_input {               style.css (line 8193)
    background: none repeat scroll 0 0 #FFFFFF;
    ~~border~~: ~~1px solid #CCCCCC~~;
    ~~font-size~~: ~~12px~~;
    ~~height~~: ~~16px~~;
    margin-left: 5px;
    ~~padding~~: ~~5px~~;
    ~~width~~: ~~115px~~;
}
```



```
<html>
    <body>
        <script type="text/javascrip">
            console.log('Hello world!');
        </script>
    </body>
</html>
```

HTML ‘Element’ Examples

- <div>
- <h1>
    - <strong>, <em>, <br>
- <img>
- <form>
- <textarea>
- <button>
- <select> / <option>

Popular Web-based Development Tools

React

JS/TS, helps build dynamic, component-based UIs

Tailwind CSS

CSS Framework with pre-defined classes

Figma

UI/UX design tool

Desktop & Mobile Frontends

Desktop

- Electron.js (HTML/CSS/JS)
- Qt (C++/Python/Rust/etc.)
- .NET (C#)

Mobile

- React Native (JS/TS)
- Flutter (Dart)
- SwitfUI (Swift – IOS)
- Jetpack Compose (Koitlin - Android)
- JavaFX (Java - Android)

03

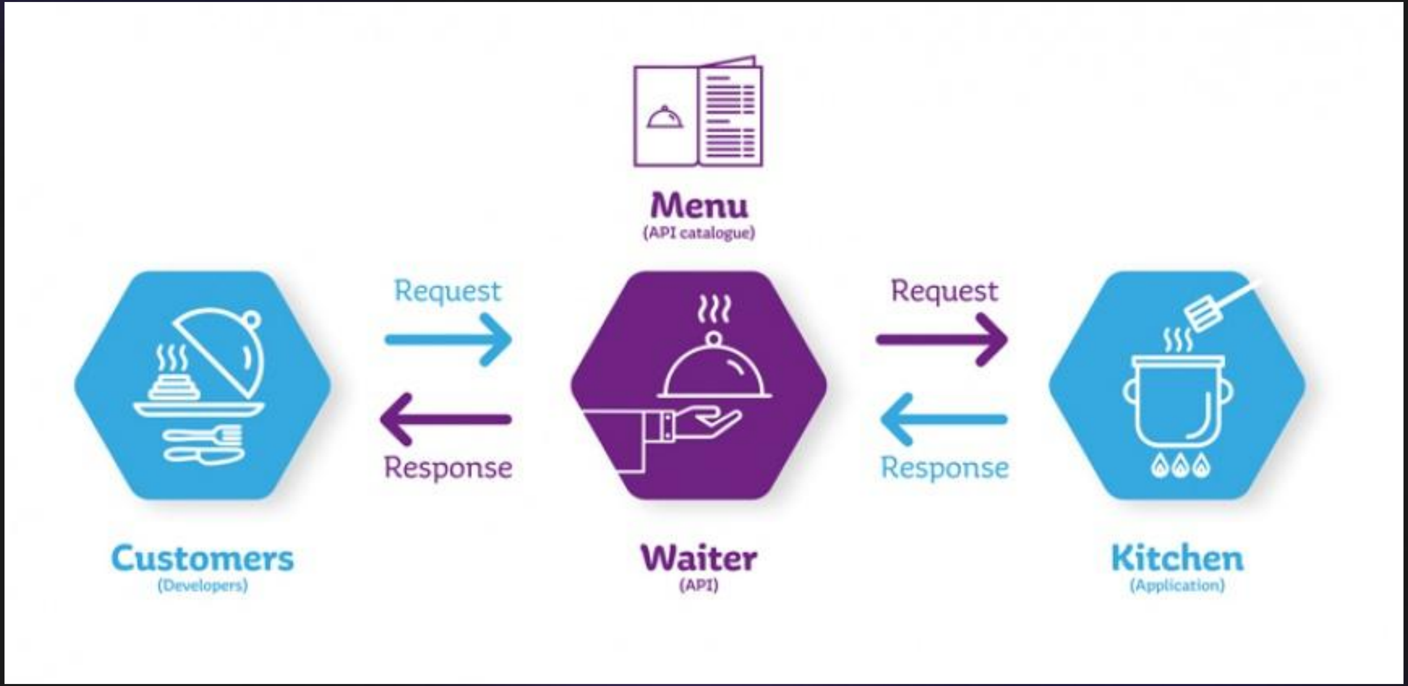
Building a Backend

What does a Backend Need to Do?

- Handle “business logic”
  - Calculating totals, managing user permissions, etc.
- Manage databases (will get into this later)
- Expose APIs
  - Communication Layer
- Handle any external services
  - Example) Stripe (Payment), Cloud-Services, Email/SMS Services, etc.

What is an API?

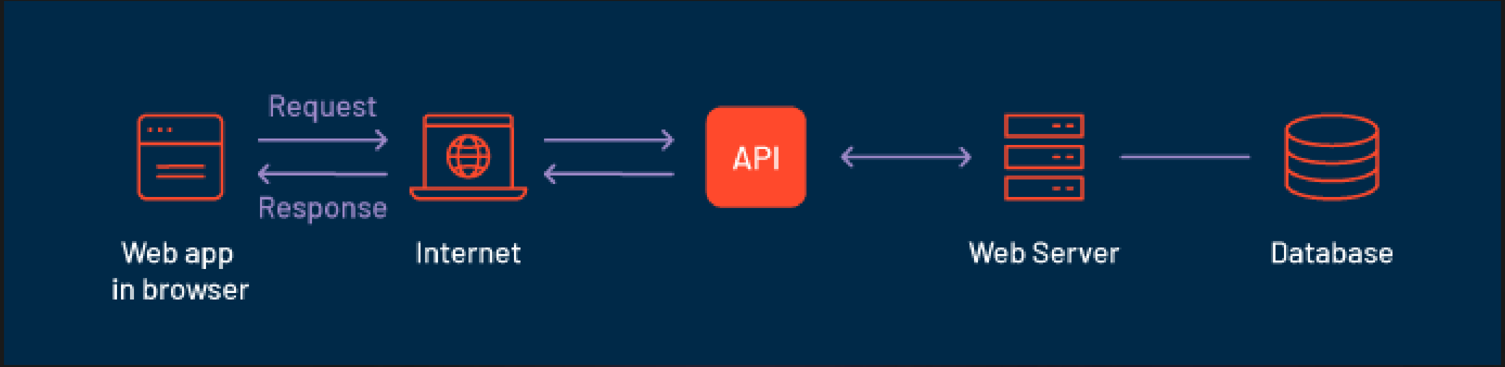
"Application Programming Interface"



What is an API? (In Practice)

<u>User-Defined Endpoints (REST APIs)</u>

- GET (/api/users) → Get a list of all users
- POST (/api/users) → Add a new user



Types of APIS

REST

HTTP Protocol, JSON Format

GraphQL

Single, flexible endpoint. Client requests exactly what they need

WebSockets

Real-time, two-way communication

Example external services

- Authentication
  - Autho, Firebase Authentication, AWS Cognito
- Payments/Subscriptions
  - Stripe, PayPal, Square
- Analytics
  - Google Analytics, ElasticSearch, Snowflake
- LLM/AI APIs
  - OpenAI (ChatGPT), Google Vision, Anthropic/Gemini/Grok/etc.
- Geolocation/Mapping
  - Google Maps, Mapbox, Geoapify
- Many, many others

Backend Languages

Pretty much anything:

- Python
- Java
- JavaScript/TypeScript (Node)
- Rust
- Go
- Ruby
- C#
- C/C++
- Elixir
- PHP

04

Building a Database

What is a “Query”?

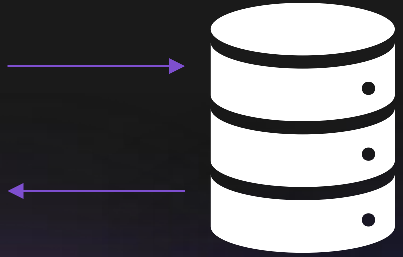
A request for data from a database

- Looks different for different types of databases
  - Popular languages include SQL, MQL, Cypher, etc.
- Managed by the backend – Database only stores data

Backend

Query = “I want all students with a GPA of 3.5 or higher”

Result



What is a database?

A way to store data

1

Relational Database

Tables with rows and columns – focuses on relations among rows

2

NoSQL Database

Tables with rows and columns – No relations

3

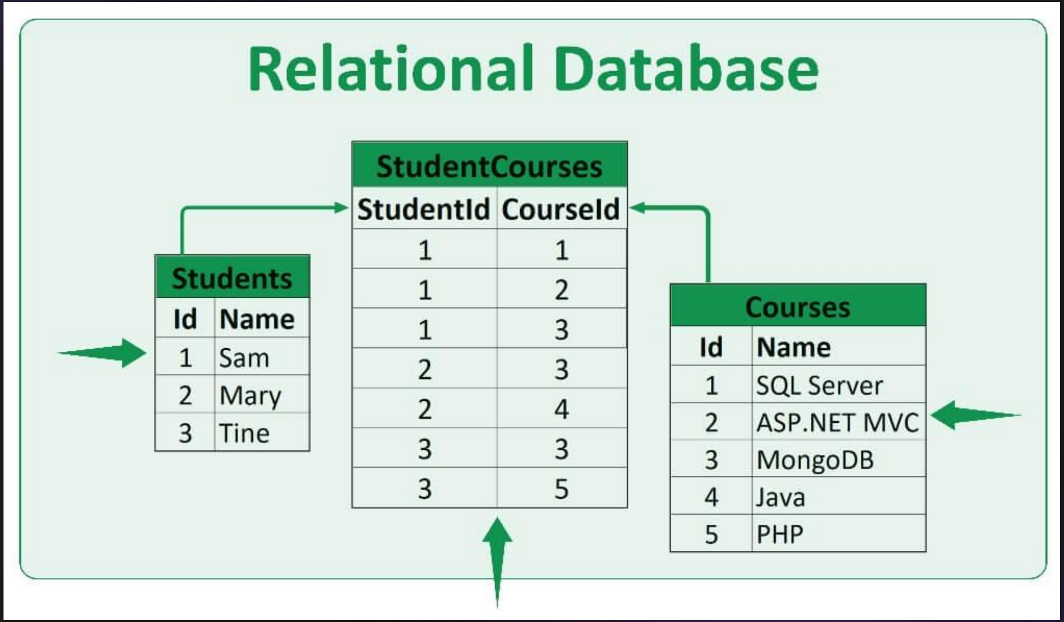
Others

For example, Graph, which Focuses on relationships and contextual meanings

Relational Databases (SQL)

Tabular with key/value pair relationships

uses SQL (Structured Query Language)



Relational Database

Students	
Id	Name
1	Sam
2	Mary
3	Tine

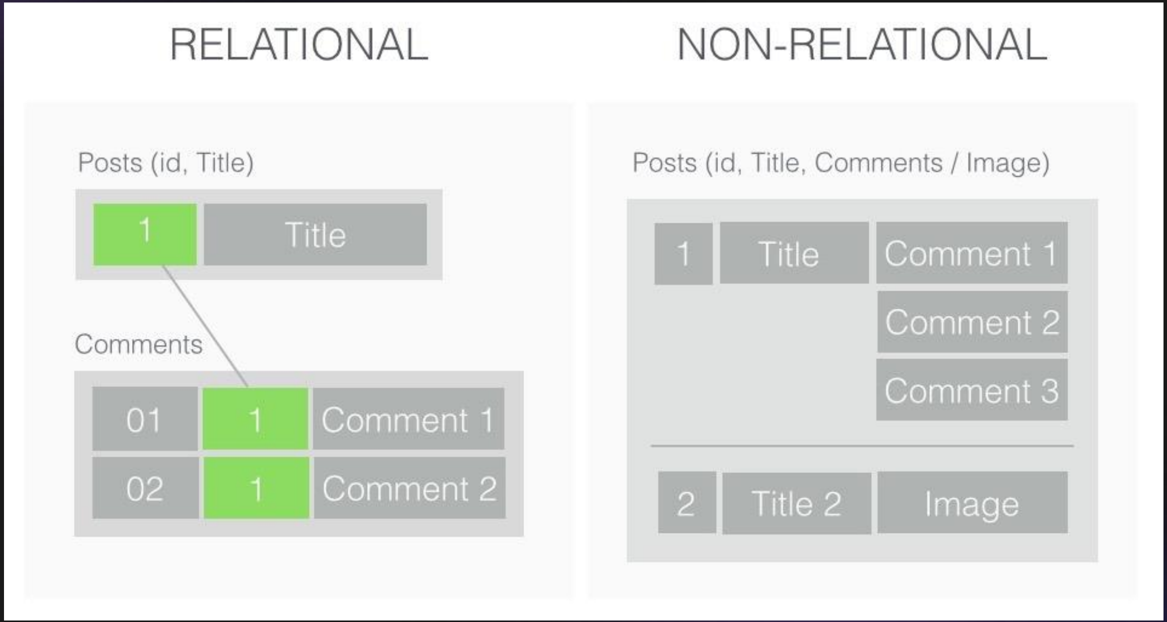
StudentCourses	
StudentId	CourseId
1	1
1	2
1	3
2	3
2	4
3	3
3	5

Courses	
Id	Name
1	SQL Server
2	ASP.NET MVC
3	MongoDB
4	Java
5	PHP

Non-Relational Databases (NoSQL)

Several Types of Models, often optimized for Scalability/Flexibility

No unified Language



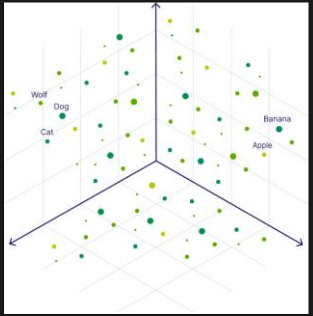
mongoDB

Cloud Firestore

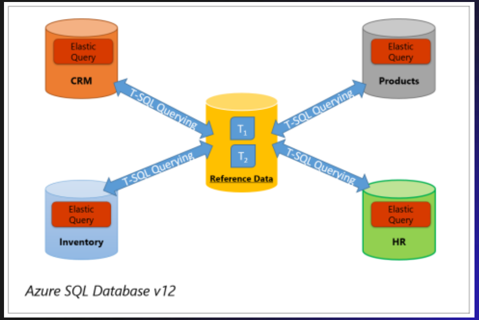
Other Types of Databases

Graph Database

Vector Database



Search/Text-based Database



neo4j

ArangoDB

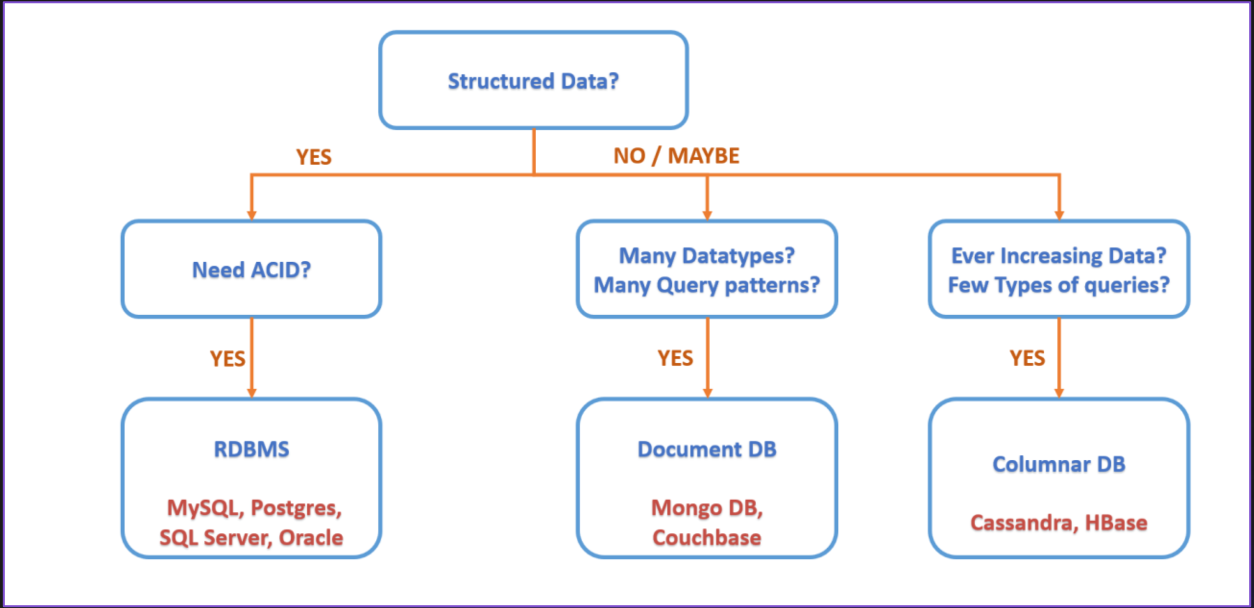
Pinecone

FAISS Vector Database

Solr

Which Database Do I Choose?

- Varies based on use-case and business constraints
- Five key factors: Scalability, Data Consistency, Query Patterns, System Resilience, and Cost

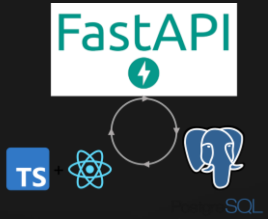

Structured Data?
YES
NO / MAYBE
Need ACID?
Many Datatypes? Many Query patterns?
Ever Increasing Data? Few Types of queries?
YES
YES
YES
RDBMS
MySQL, Postgres, SQL Server, Oracle
Document DB
Mongo DB, Couchbase
Columnar DB
Cassandra, HBase

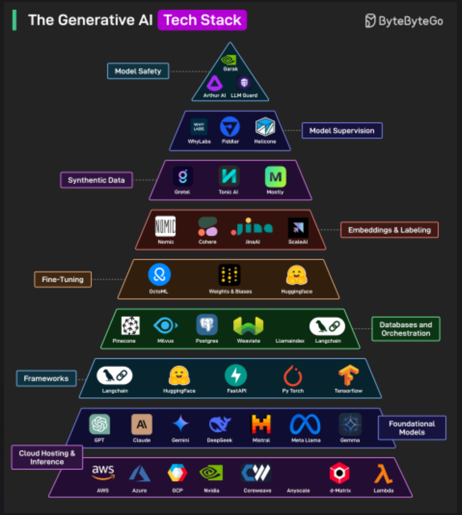
05

What is a “Stack”?

What is a “Stack”?

- This refers to the collection of technologies/frameworks used to build something
- Some popular stacks:
  - MERN
    - MongoDB (Database)
    - Express (Backend API Framework)
    - React (Frontend)
    - Node.js (Backend)
  - Modern AI
    - FastAPI (Backend API Framework)
    - PostgreSQL (Database)
    - Milvus (Database – Vector)
    - React + Next.js (Frontend)
    - Docker + Kubernetes (Infrastructure – Will discuss later)
  - LAMP
    - Linux (OS Host Environment)
    - Apache (Web Server, Backend-ish)
    - MySQL (Database)
    - PHP (Backend Logic + Frontend)

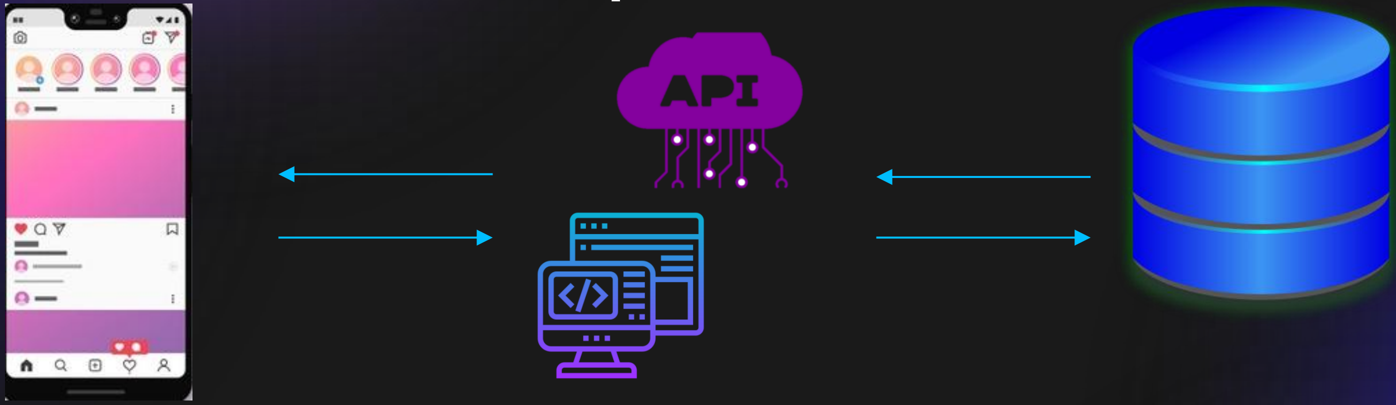




06

Example Revisited: Instagram

Revisited: **Instragram's** full-stack components



Frontend

JavaScript/HTML/CSS (React), Swift, Kotlin, React Native

Backend

Python (Django Framework), C++/Go (Performance-Critical Microservices), GraphQL API Layer

Database

PostgresSQL, Cassandra (NoSQL), Amazon S3

Simple Example

My MPI benchmark program thingy

GitHub

Flask

HTML CSS JS

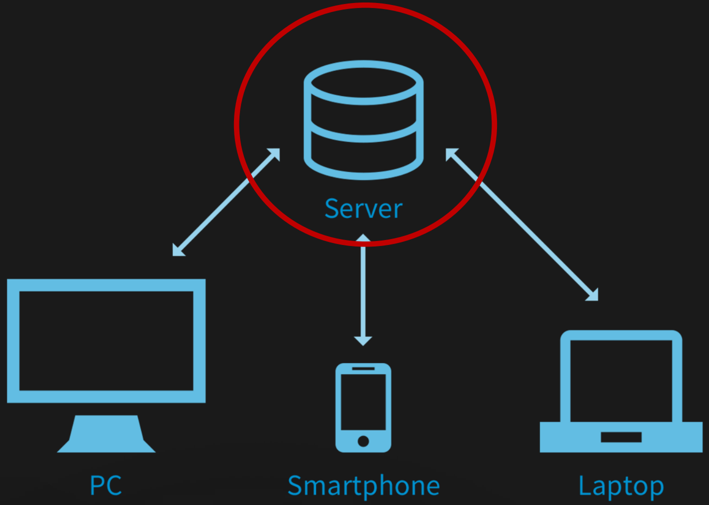
07

What is the "Cloud"?

Hosting the Server

If using your own server, must consider:

- Server must always be running
- Must be able to communicate to outside users
  - Must forward the server to the "DNS"
  - Must be able to communicate to all client devices through the internet
- Must scale with usage – could handle millions of requests per day



What Does Cloud Provide?

- Remote servers and infrastructure
- On-demand resources
- Abstracted everything pretty much

Google Cloud Platform
aws
ORACLE CLOUD
Microsoft Azure

The Big Cloud Services

AWS

Largest & Most Mature

- EC2 (Virtual Servers)
- S3 (Object Storage)
- RDS/Aurora (Databases)
- Lambda (Serverless Functions)

Microsoft Azure

- Virtual Machines
- SQL Databases
- Blob (Object) Storage
- Functions (Serverless)
- DevOps (will cover next)

Google Cloud Platform

- Compute Engine (VMs)
- Cloud Storage (Objects)
- BigQuery (Large Database)
- Cloud Run/Functions
- Vertex AI (ML Platform)

Additional Cloud Technologies (For Deploying)

- Vercel
  - Basically no configuration, super easy, optimized for React apps
- Netlify
  - Simple, good free-tier, instant rollbacks
- Railway
  - Little configuration, easy database configuration/management
- Supabase/Firebase
- Github Pages
  - Simple and free for static websites
  - Very limited but great for some uses

08

Additional **Concepts**

Foundational Developer Tools

VSCode

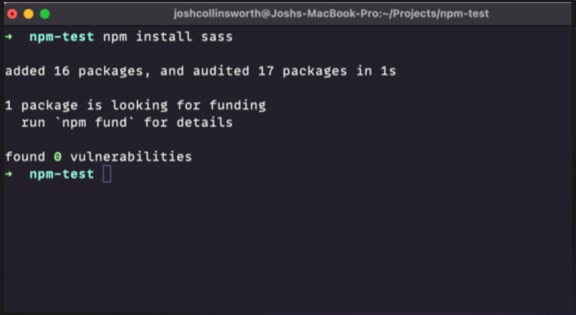
Industry-standard IDE, many others but most popular. Many built-in features

GitHub

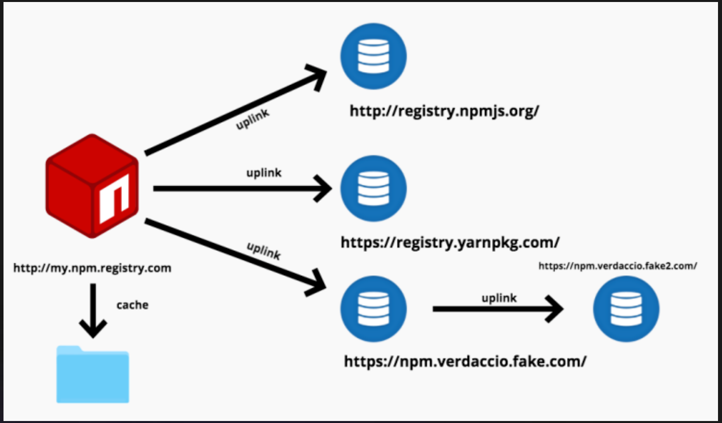
Version control, collaboration, hosting "repos" (repositories)

What is a **package manager?**

Installs, updates, and manages libraries/frameworks.

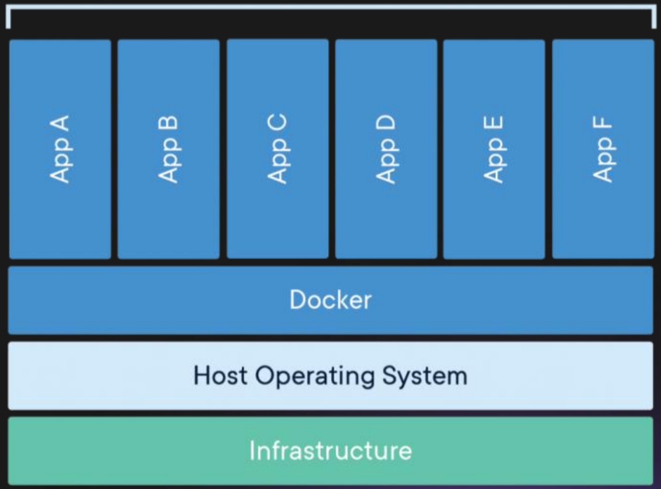

joshcollinsworth@Joshs-MacBook-Pro:~/Projects/npm-test

```
→  npm-test npm install sass

added 16 packages, and audited 17 packages in 1s

1 package is looking for funding
  run `npm fund` for details

found 0 vulnerabilities
→  npm-test
```


uplink
http://registry.npmjs.org/
uplink
https://registry.yarnpkg.com/
http://my.npm.registry.com
uplink
https://npm.verdaccio.fake2.com/
cache
uplink
https://npm.verdaccio.fake.com/

What is Docker?

- A tool that “Containerizes” an application
- Fixes the “works on my machine” problem
- Ensures each user runs on the same environment



What is a Build Tool?

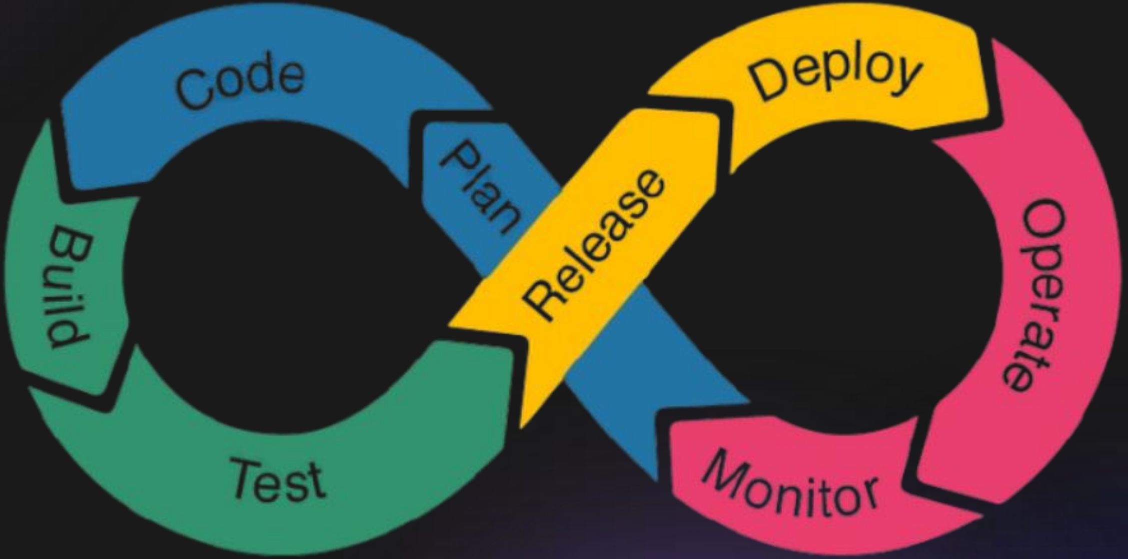
- Some frameworks, such as React, do not natively run by itself
- Build tools convert source code into executable code

What is CI/CD?

- Stands for “Continuous Integration and Continuous Deployment”
- A set of practices to automate everything
  - Building Code
  - Testing Code
  - Releasing Code
- Used in industry everywhere in some form, from startups to big tech
  - “DevOps Engineer”

What is Agile/DevOps?

"Development Operations", essentially just a philosophy used for big app deployments.



1. Plan
2. Code
3. Build
4. Test
5. Release
6. Deploy
7. Operate
8. Monitor

09

How to Start Building

Choosing a “Stack”

1. Get an idea (usually derived from a problem)
2. Think about what the “idea” actually does
3. Ask – What do I need to accomplish this? (Web-based frontend, Payment in the backend, real-time communication, etc.)
4. Research the best tools for the job (Do I need a framework for this frontend? What is the easiest way to process payments? How can I have real-time communication?)
5. Choose the easiest *combination* of tools to use

OR

1. Pick something you want to learn or think is cool
2. Use it

How to Learn How to Build Full-Stack Software

The best way to learn is by doing.

- Ask ChatGPT tons of questions
- Watch nerd YouTube Videos (Ex. Fireship)
- Browse through GitHub for other projects
- Build random stuff
- Go to Hackathons!!!
- Building stuff yourself is better than following tutorials

Thank you!

Any Questions?