Full stack Notes

Full stack Developer & CS Notes

Resources

Source: freeCodeCamp.org, YouTube Playlists, websites, books
Playlists: Back End Developer Learning Path (17 videos) • Popular Programming Courses (13 videos)
Last Updated: 10-05-2026
written by: Aayush Ate **Tags:** #backend #programming #cs #roadmap #learning-path #frontend #books

refer the website,books in the last page

Table of Contents

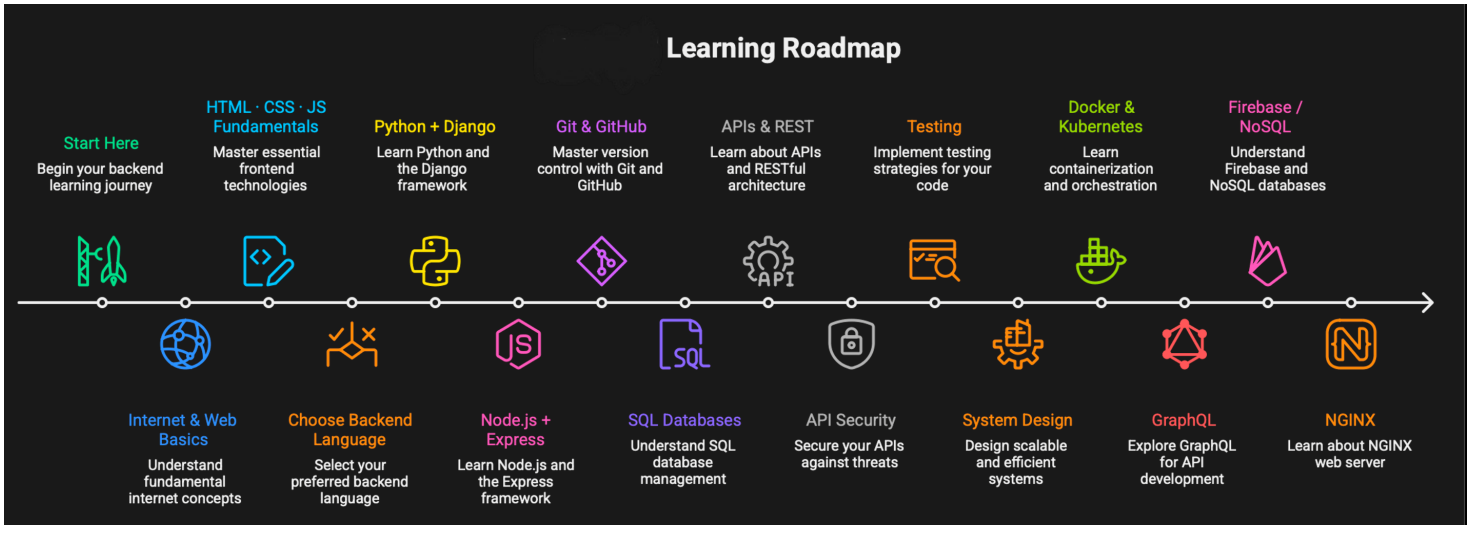
clickable videos and links given below -.-

- Learning Roadmap
- Playlist 1 — Back End Developer Learning Path
- Playlist 2 — Popular Programming Courses
- Core Concepts Master Index
- Technology Comparison Tables
- How Everything Connects

Learning Roadmap

Start Here

Follow this order for the most efficient backend learning journey.



📚 Playlist 1 — Back End Developer Learning Path

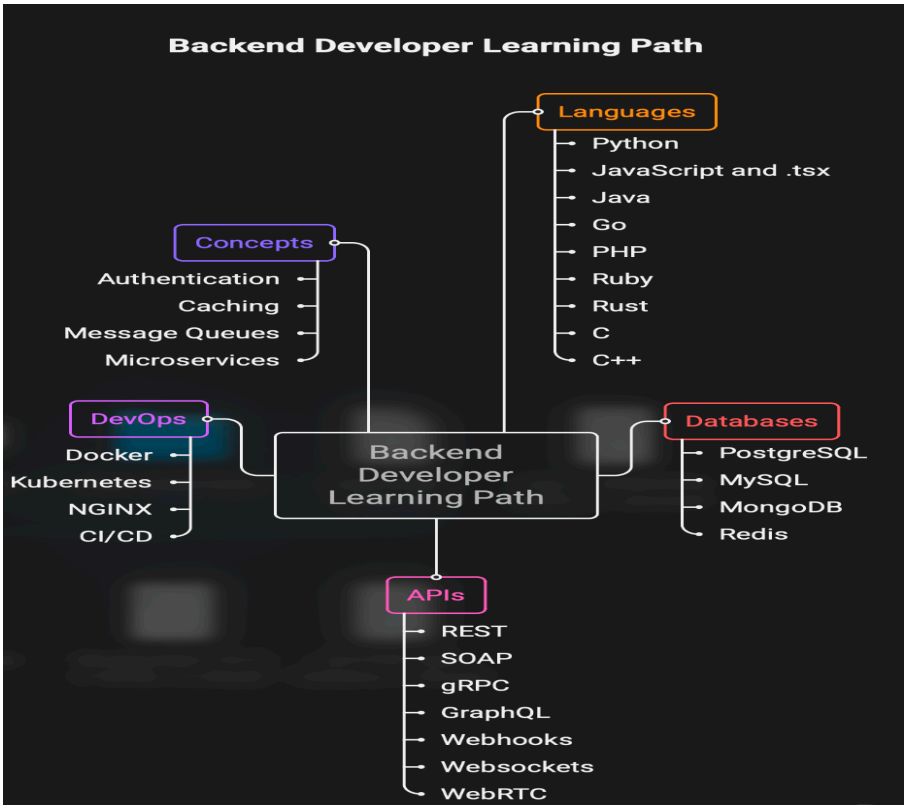
> **Channel:** freeCodeCamp.org | **Videos:** 17 | **Updated:** Mar 18, 2026

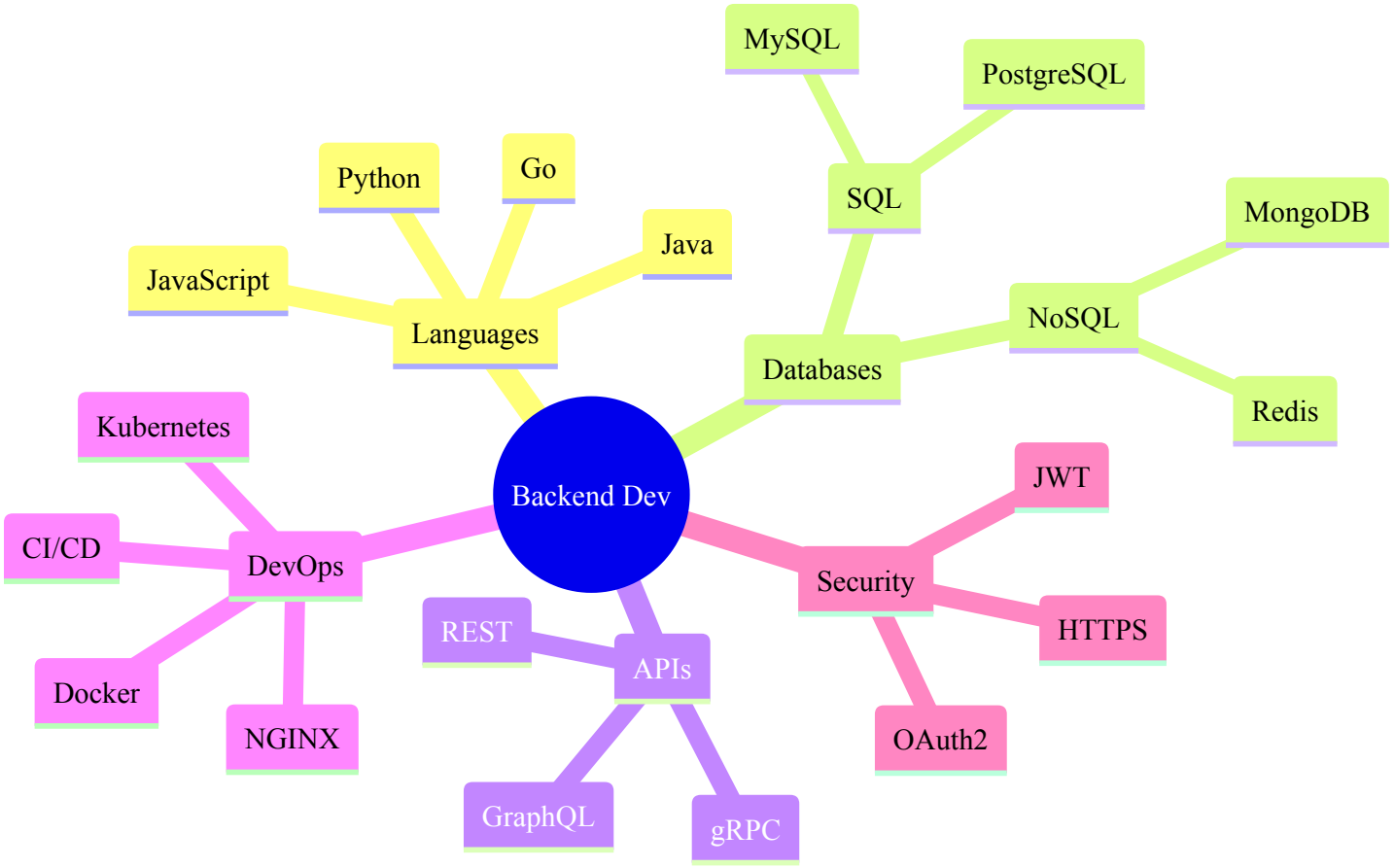
Video 1 — Back End Developer Roadmap `10:30`

📋 Overview

A bird's-eye view of the entire backend ecosystem — technologies, tools, and the order to learn them.

Key Technologies on the Roadmap:

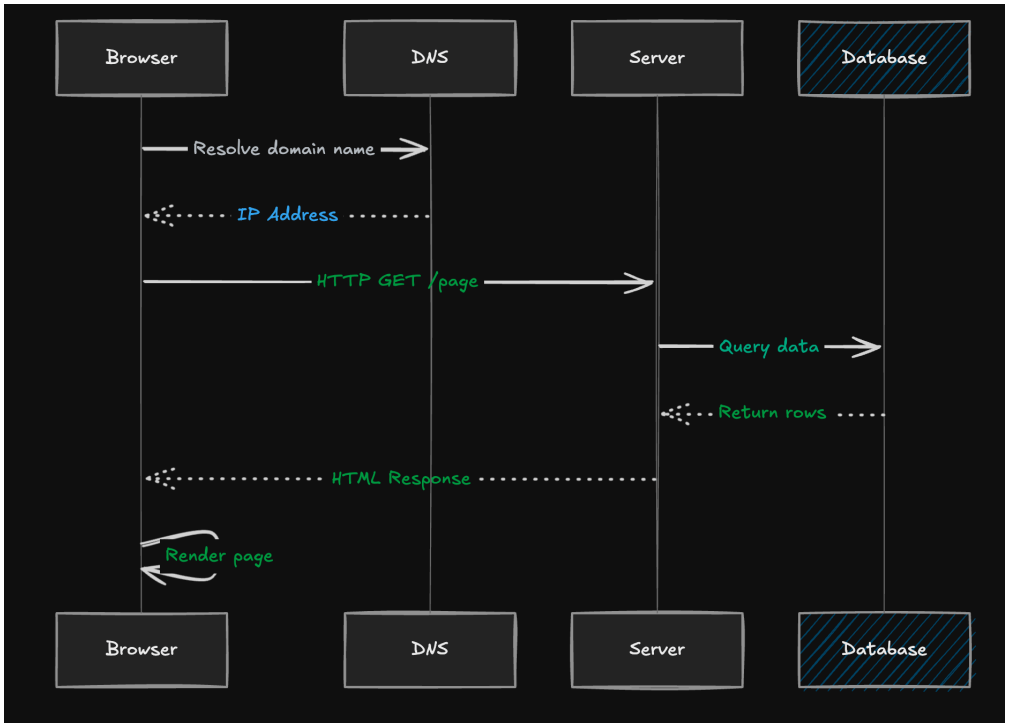




Video 2 — Full Stack Web Development for Beginners `7:29:12`

> **Tech Stack Covered**
>
> HTML · CSS · JavaScript · Node.js · MongoDB

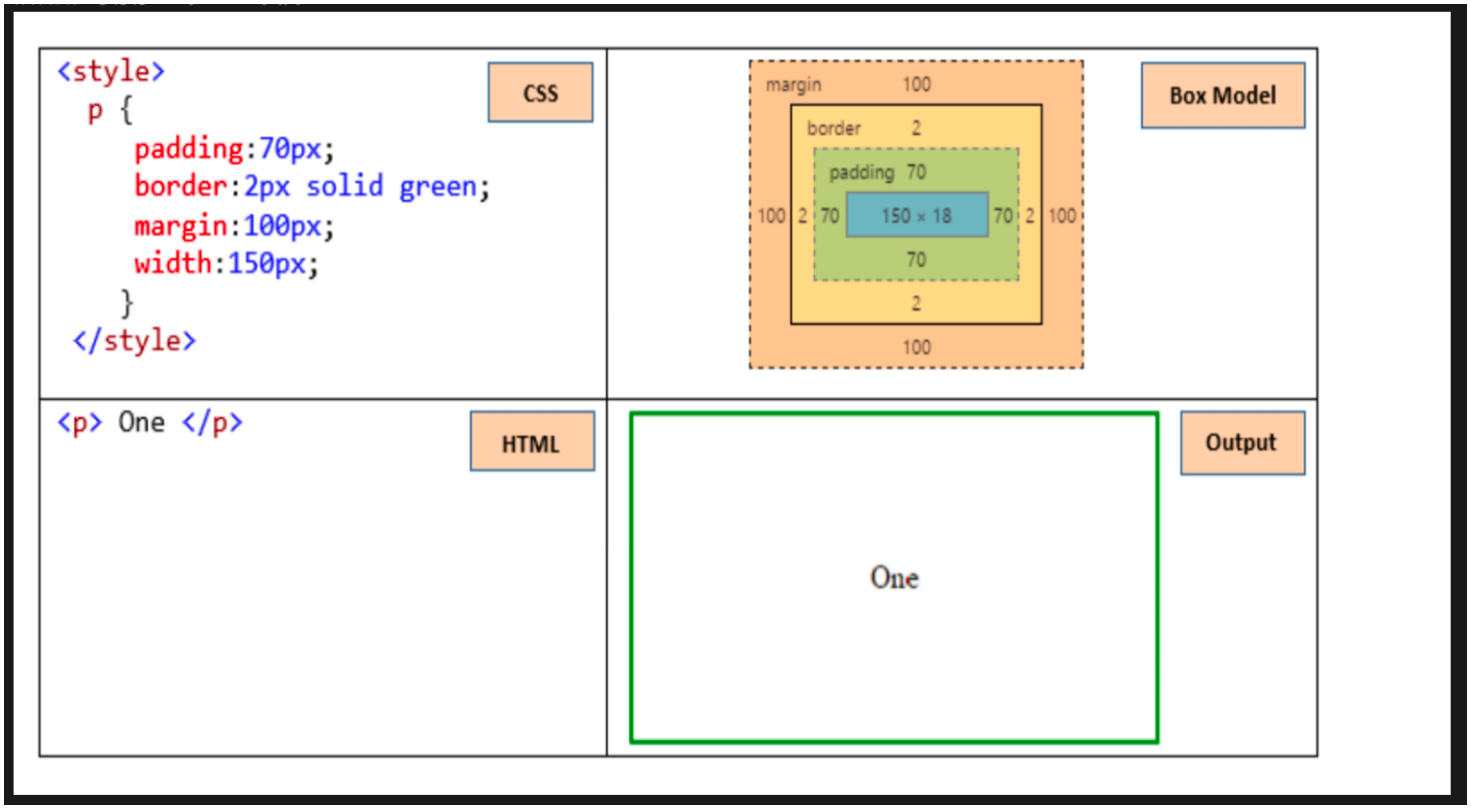
How the Web Works



HTML Essentials

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>My Page</title>
</head>
<body>
  <h1>Heading</h1>
  <p>Paragraph</p>
  <a href="/page">Link</a>
  <form action="/submit" method="POST">
    <input type="text" name="username">
    <button type="submit">Submit</button>
  </form>
</body>
</html>
```

CSS Box Model



Key CSS properties

- `display: flex | grid | block | inline`
- `position: relative | absolute | fixed | sticky`
- `box-sizing: border-box` ← always set this!

⚡ JavaScript Fundamentals

```javascript
// Variables
const name = "Alice";    // immutable binding
let age = 25;            // mutable

// Arrow Functions
const greet = (name) => `Hello, ${name}!`;
```

```
// Promises & Async/Await
async function fetchData(url) {
  try {
    const res = await fetch(url);
    const data = await res.json();
    return data;
  } catch (err) {
    console.error("Error:", err);
  }
}

// Array methods you MUST know
const nums = [1, 2, 3, 4, 5];
nums.map(n => n * 2);          // [2, 4, 6, 8, 10]
nums.filter(n => n > 2);       // [3, 4, 5]
nums.reduce((acc, n) => acc + n, 0); // 15
```

🟢 Node.js + Express Server

```
const express = require('express');
const app = express();

app.use(express.json());

// Routes
app.get('/api/users', async (req, res) => {
  const users = await User.find();
  res.json(users);
});

app.post('/api/users', async (req, res) => {
  const user = new User(req.body);
  await user.save();
  res.status(201).json(user);
});

app.listen(3000, () => console.log('Server running on port 3000'));
```

🍃 MongoDB + Mongoose

```
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
  name: { type: String, required: true },
  email: { type: String, unique: true },
  createdAt: { type: Date, default: Date.now }
});

const User = mongoose.model('User', userSchema);
```

SQL vs NoSQL

MongoDB is **schema-flexible** (good for prototyping). PostgreSQL is **schema-strict** (good for production with complex relations).

Video 3 — Python Tutorial for Beginners `8:41:54`

Comprehensive Python with mini-projects

🐍 Python Fundamentals

```python
# Data Types
name: str = "Alice"
age: int = 25
score: float = 98.5
is_active: bool = True
items: list = [1, 2, 3]
data: dict = {"key": "value"}
unique: set = {1, 2, 3}
coords: tuple = (10, 20)

# Functions
def greet(name: str, greeting: str = "Hello") -> str:
    return f"{greeting}, {name}!"

# List Comprehensions
squares = [x**2 for x in range(10)]
evens = [x for x in range(20) if x % 2 == 0]

# Lambda
double = lambda x: x * 2

# Classes
class Animal:
    def __init__(self, name: str, species: str):
        self.name = name
        self.species = species

    def speak(self) -> str:
        return f"{self.name} makes a sound"

    def __repr__(self) -> str:
        return f"Animal(name={self.name!r})"

class Dog(Animal):
```

```
    def speak(self) -> str:
        return f"{self.name} barks!"
```

🔄 Python Control Flow

```
# Match statement (Python 3.10+)
match command:
    case "quit":
        quit()
    case "start":
        start_game()
    case _:
        print("Unknown command")

# Context Managers
with open("file.txt", "r") as f:
    content = f.read()

# Generators
def fibonacci():
    a, b = 0, 1
    while True:
        yield a
        a, b = b, a + b

# Decorators
def timer(func):
    import time
    def wrapper(*args, **kwargs):
        start = time.time()
        result = func(*args, **kwargs)
        print(f"{func.__name__} took {time.time()-start:.2f}s")
        return result
    return wrapper
```

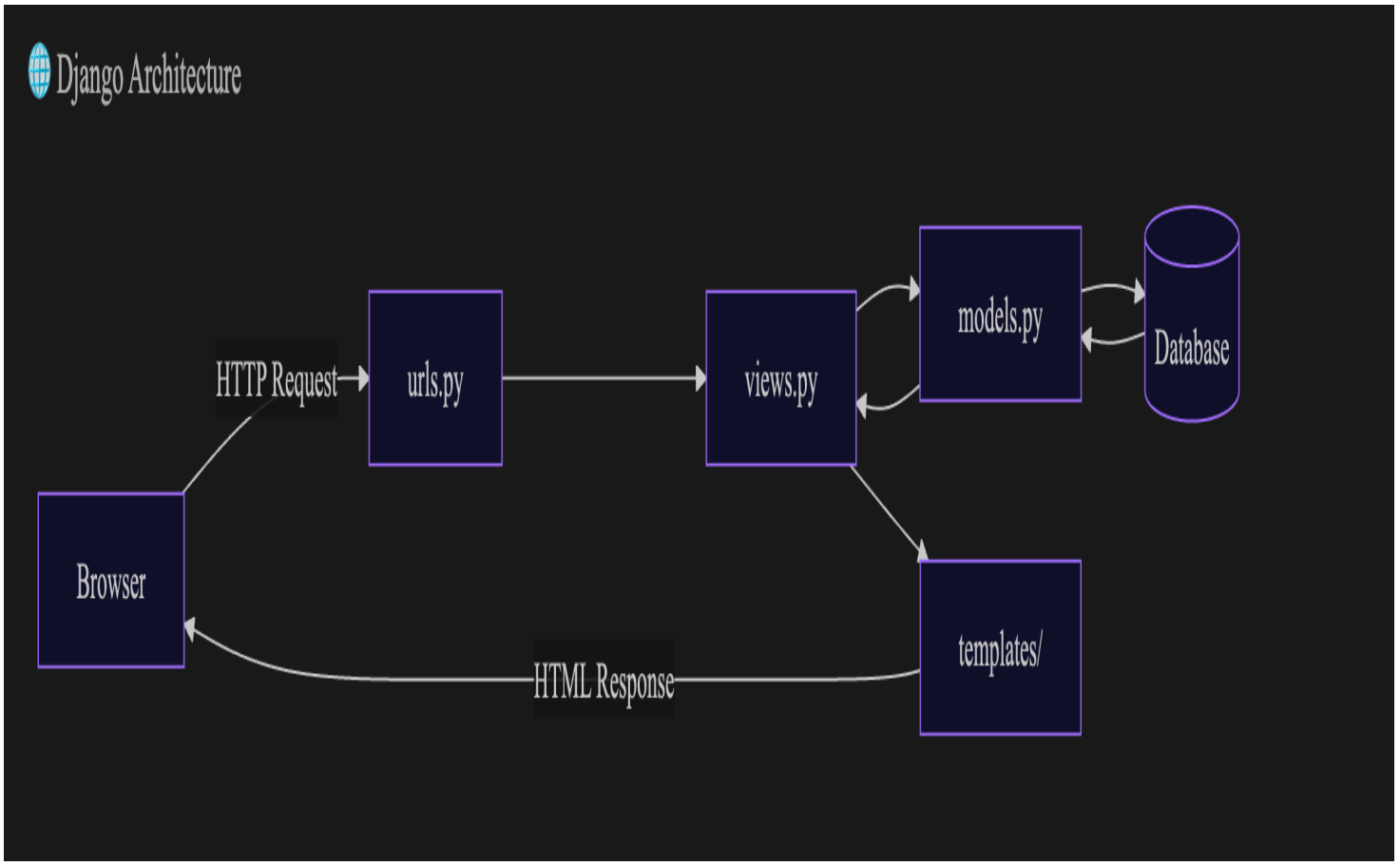
📦 Key Python Libraries

Library	Use Case	Install
`requests`	HTTP calls	`pip install requests`
`pandas`	Data analysis	`pip install pandas`
`numpy`	Math/arrays	`pip install numpy`
`pydantic`	Data validation	`pip install pydantic`
`pytest`	Testing	`pip install pytest`
`sqlalchemy`	ORM	`pip install sqlalchemy`
`uv`	Package manager	`pipx install uv`

Library	Use Case	Install
`ruff`	Linting/formatting	`uv tool install ruff`
`black`	Code formatting	`uv tool install black`
`mypy`	Type checking	`uv tool install mypy`

Video 4 — Learn Django by Building a Marketplace `2:23:45`

> 📋 Build a real-world Django application — marketplace with listings, auth, search



🏗️ Django Project Structure

```
myproject/
├── manage.py
├── myproject/
│   ├── settings.py       # Configuration
│   ├── urls.py           # Root URL routing
│   └── wsgi.py           # WSGI entry point
└── marketplace/          # App
    ├── models.py         # Database models
    ├── views.py          # Business logic
    ├── urls.py           # App URL routing
    ├── forms.py          # Form handling
```

```
    ├── admin.py          # Admin panel config
    └── templates/        # HTML templates
```

📝 Django Models

```python
from django.db import models
from django.contrib.auth.models import User

class Item(models.Model):
    name = models.CharField(max_length=255)
    description = models.TextField(blank=True)
    price = models.DecimalField(max_digits=10, decimal_places=2)
    seller = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='items')
    image = models.ImageField(upload_to='items/', blank=True)
    created_at = models.DateTimeField(auto_now_add=True)
    is_sold = models.BooleanField(default=False)

    class Meta:
        ordering = ['-created_at']

    def __str__(self):
        return self.name
```

🔗 Django Views & URLs

```python
# views.py
from django.shortcuts import render, get_object_or_404
from .models import Item

def item_list(request):
    query = request.GET.get('q', '')
    items = Item.objects.filter(is_sold=False)
    if query:
        items = items.filter(name__icontains=query)
    return render(request, 'marketplace/list.html', {'items': items})

# urls.py
from django.urls import path
from . import views

urlpatterns = [
    path('', views.item_list, name='item-list'),
    path('item/<int:pk>/', views.item_detail, name='item-detail'),
    path('item/new/', views.item_create, name='item-create'),
]
```

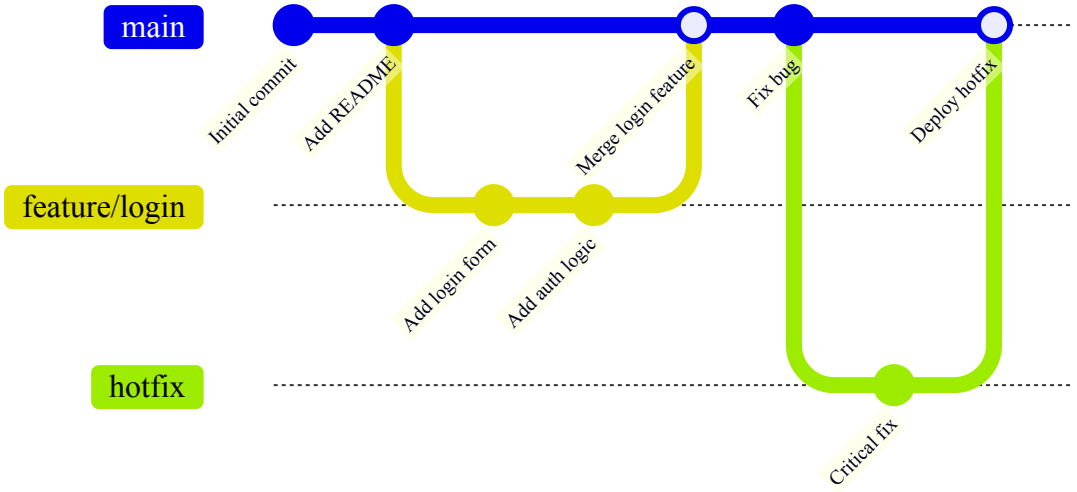
🔥 Django ORM Cheatsheet

```
Item.objects.all()                    # All items
Item.objects.filter(price__lt=100)    # Price < 100
Item.objects.exclude(is_sold=True)    # Not sold
Item.objects.order_by('-created_at')  # Newest first
Item.objects.select_related('seller') # JOIN optimization
```

Video 5 — Git and GitHub for Beginners `1:08:30`

> 📋 Version control from zero to confident

🌿 Git Core Concepts

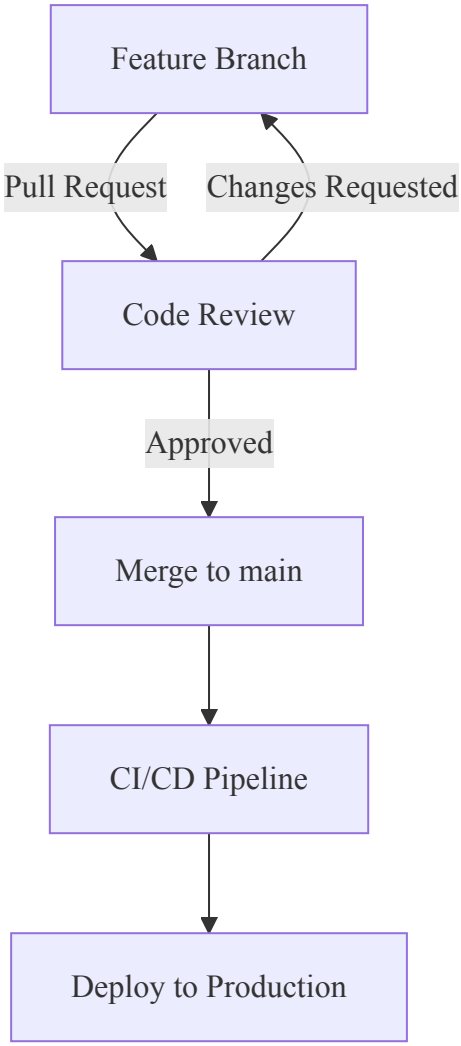


📌 Essential Git Commands

```
# Setup
git config --global user.name "Your Name"
git config --global user.email "you@email.com"

# Basic workflow
git init                    # Initialize repo
git clone <url>             # Clone remote repo
git status                  # Check changes
git add .                   # Stage all changes
git add <file>              # Stage specific file
git commit -m "message"     # Commit with message
git push origin main        # Push to remote

# Branching
git branch feature/new      # Create branch
git checkout feature/new    # Switch branch
git checkout -b feature/new # Create + switch
git merge feature/new       # Merge branch
git branch -d feature/new   # Delete branch
```

```
# Undoing
git restore <file>          # Discard working changes
git reset HEAD~1            # Undo last commit (keep changes)
git reset --hard HEAD~1     # Undo last commit (discard changes)
git revert <commit>         # Create undo commit (safe for shared)

# Inspection
git log --oneline --graph   # Visual history
git diff                    # Show unstaged changes
git stash                   # Temporarily save changes
git stash pop               # Restore stashed changes
```

🔀 Git Workflow Strategies



Video 6 — SQL Tutorial for Beginners `5:25:39`

> 📋 SQL from basics to advanced + interview prep

🗄️ Database Fundamentals

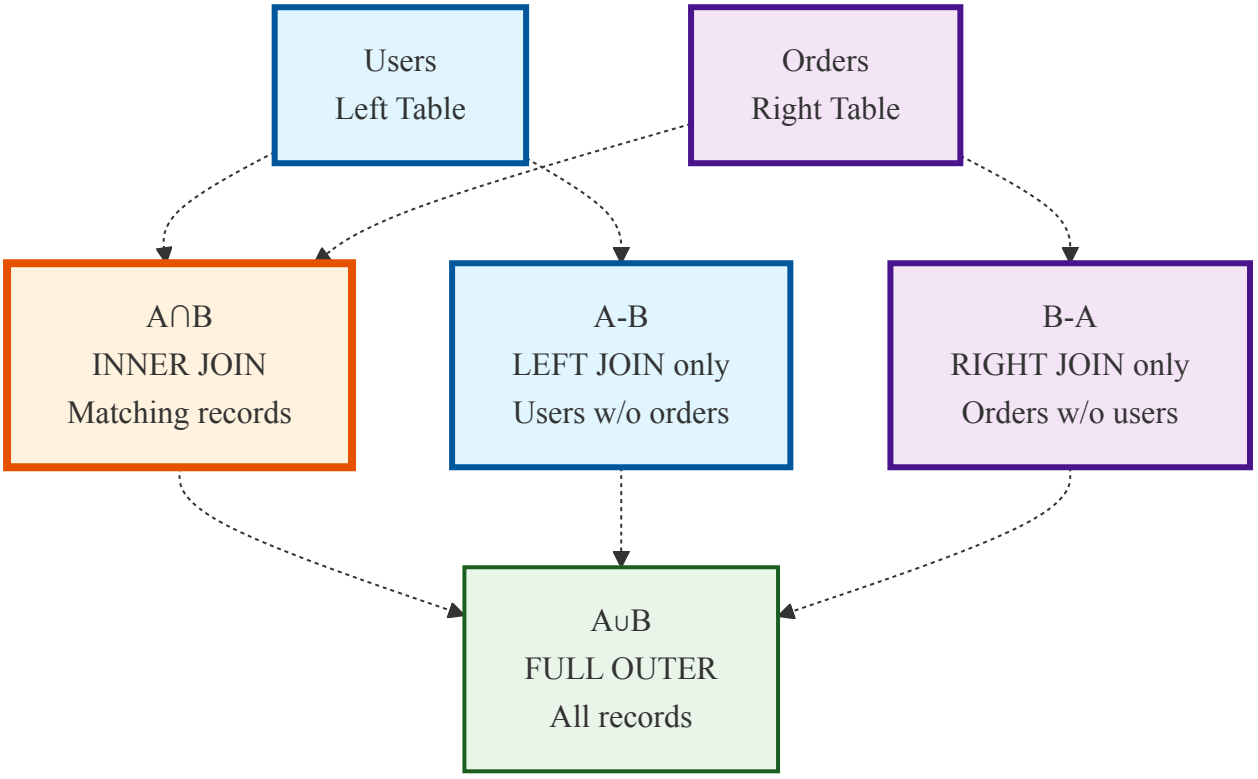
$$\text{Relation} = \{(a_1, b_1, c_1), (a_2, b_2, c_2), \ldots\}$$

where each tuple is a **row** and each attribute is a **column**.

📝 SQL Core Syntax

```sql
-- DDL: Data Definition Language
CREATE TABLE users (
    id          SERIAL PRIMARY KEY,
    email       VARCHAR(255) UNIQUE NOT NULL,
    name        VARCHAR(100) NOT NULL,
    age         INT CHECK (age >= 0),
    created_at  TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- DML: Data Manipulation Language
INSERT INTO users (email, name, age) VALUES ('alice@ex.com', 'Alice', 25);

UPDATE users SET age = 26 WHERE email = 'alice@ex.com';

DELETE FROM users WHERE id = 5;

-- DQL: Data Query Language
SELECT name, email FROM users WHERE age > 18 ORDER BY name ASC LIMIT 10;

-- Aggregations
SELECT
    department,
    COUNT(*) as count,
    AVG(salary) as avg_salary,
    MAX(salary) as max_salary
FROM employees
GROUP BY department
HAVING AVG(salary) > 50000;
```

🔗 SQL JOINs Visual



JOIN Type	Returns
`INNER JOIN`	Only matching rows in BOTH tables
`LEFT JOIN`	All left + matching right (NULL if no match)
`RIGHT JOIN`	All right + matching left (NULL if no match)
`FULL OUTER JOIN`	All rows from both tables
`CROSS JOIN`	Every combination (Cartesian product)

```sql
-- JOIN example
SELECT
    u.name,
    COUNT(o.id) as order_count,
    SUM(o.total) as lifetime_value
FROM users u
LEFT JOIN orders o ON u.id = o.user_id
WHERE u.created_at > '2024-01-01'
GROUP BY u.id, u.name
ORDER BY lifetime_value DESC;

-- Subquery
SELECT name FROM users
WHERE id IN (
    SELECT user_id FROM orders WHERE total > 1000
);

-- Window Functions
SELECT
    name,
    salary,
    RANK() OVER (PARTITION BY department ORDER BY salary DESC) as dept_rank
FROM employees;
```

🚀 SQL Interview Questions

⁉ Common SQL Interview Questions

1. What's the difference between `WHERE` and `HAVING`?
2. Explain the difference between `DELETE`, `TRUNCATE`, and `DROP`
3. What is a database index and when should you use one?
4. What is normalization? Explain 1NF, 2NF, 3NF
5. What is a stored procedure vs a view?

Normalization levels:

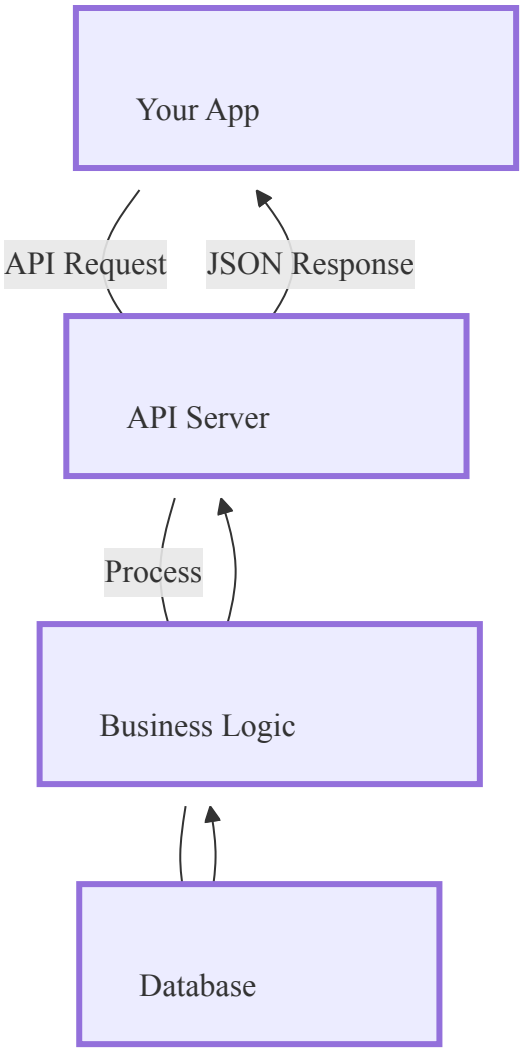
Normal Form	Rule
1NF	Atomic values, no repeating groups
2NF	1NF + no partial dependencies
3NF	2NF + no transitive dependencies
BCNF	Stricter 3NF

Video 7 — APIs for Beginners `3:07:07`

> 📋 What APIs are, how REST works, how to use them

🔌 What is an API?

> ⓘ Definition
>
> An **API** (Application Programming Interface) is a contract that defines how software components communicate. Think of it as a restaurant menu — you order (request) and the kitchen delivers (response), without needing to know how food is cooked.



🌐 REST API Principles

Principle	Description
Stateless	Each request is self-contained
Client-Server	Separation of concerns
Cacheable	Responses can be cached
Uniform Interface	Consistent API design
Layered System	Client doesn't know about backend layers

📋 HTTP Methods & Status Codes

```
CRUD → HTTP Methods:
Create → POST   /users
Read   → GET    /users  or  /users/:id
Update → PUT    /users/:id  (full replace)
       → PATCH  /users/:id  (partial update)
Delete → DELETE /users/:id
```

Status Code	Meaning
`200 OK`	Success
`201 Created`	Resource created
`204 No Content`	Success, no body
`400 Bad Request`	Invalid input
`401 Unauthorized`	Not authenticated
`403 Forbidden`	Not authorized
`404 Not Found`	Resource missing
`422 Unprocessable`	Validation error
`500 Server Error`	Backend crashed

🛠️ Fetch API in JavaScript

```
// GET request
const getUsers = async () => {
  const response = await fetch('https://api.example.com/users', {
    headers: {
      'Authorization': `Bearer ${token}`,
      'Content-Type': 'application/json'
    }
  });

  if (!response.ok) throw new Error(`HTTP ${response.status}`);
  return response.json();
};

// POST request
const createUser = async (userData) => {
  const response = await fetch('https://api.example.com/users', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify(userData)
  });
  return response.json();
};
```

Video 8 — Python API Development (`Comprehensive`)

> 📋 Build production-ready APIs with FastAPI

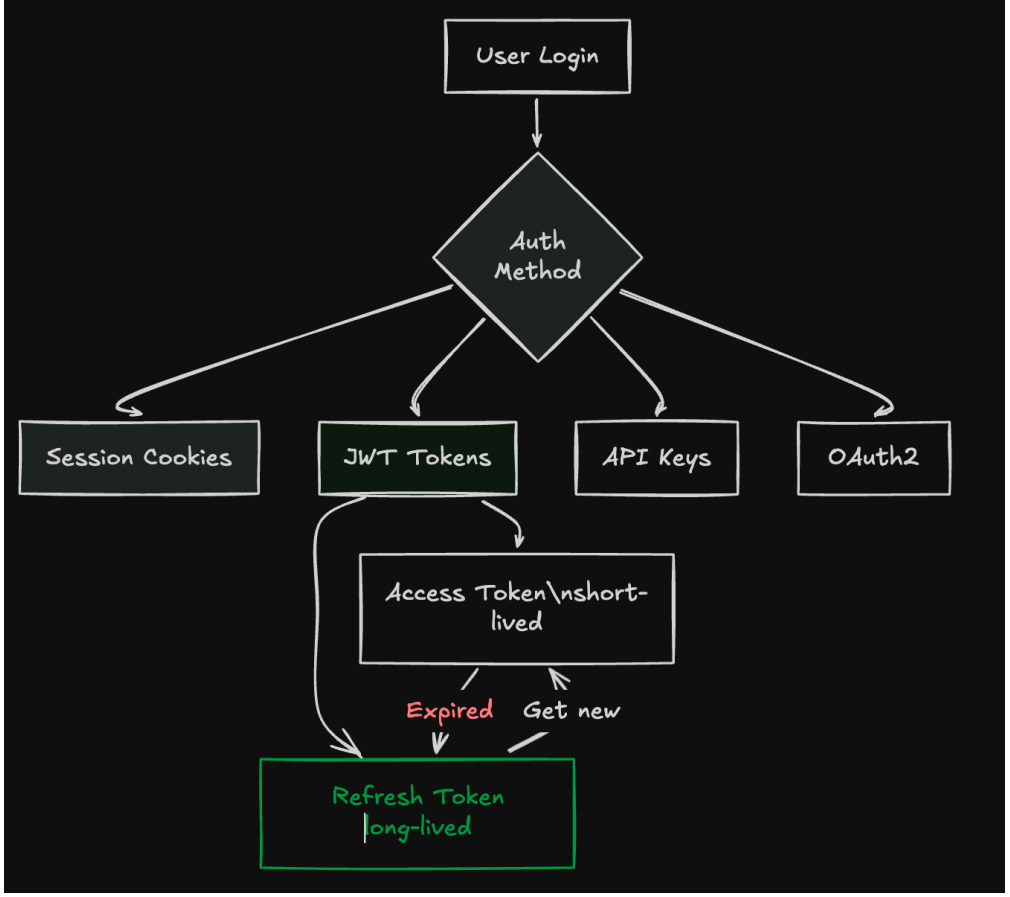
⚡ FastAPI Example

```python
from fastapi import FastAPI, HTTPException, Depends
from pydantic import BaseModel
from typing import Optional

app = FastAPI(title="My API", version="1.0.0")

class UserCreate(BaseModel):
    name: str
    email: str
    age: Optional[int] = None

class UserResponse(BaseModel):
    id: int
    name: str
    email: str

    class Config:
        from_attributes = True

@app.get("/users", response_model=list[UserResponse])
async def get_users(skip: int = 0, limit: int = 10, db = Depends(get_db)):
    return db.query(User).offset(skip).limit(limit).all()

@app.post("/users", response_model=UserResponse, status_code=201)
async def create_user(user: UserCreate, db = Depends(get_db)):
    db_user = User(**user.dict())
    db.add(db_user)
    db.commit()
    db.refresh(db_user)
    return db_user

@app.get("/users/{user_id}", response_model=UserResponse)
async def get_user(user_id: int, db = Depends(get_db)):
    user = db.query(User).filter(User.id == user_id).first()
    if not user:
        raise HTTPException(status_code=404, detail="User not found")
    return user
```

Video 9 — Secure Your APIs `1:27:01`

> 📋 API security: authentication, authorization, rate limiting, best practices

🔐 Authentication Methods



🔑 JWT (JSON Web Token)

A JWT has 3 parts separated by `.`:

$$\underbrace{\text{eyJhbGc....}}_{\text{Header}}.\underbrace{\text{eyJ1c2V....}}_{\text{Payload}}.\underbrace{\text{SflKxwR...}}_{\text{Signature}}$$

```
// Generate JWT (Node.js)
const jwt = require('jsonwebtoken');

const token = jwt.sign(
  { userId: user.id, email: user.email, role: user.role },
  process.env.JWT_SECRET,
  { expiresIn: '15m' }
);

// Verify JWT
const decoded = jwt.verify(token, process.env.JWT_SECRET);
```

🛡️ Security Best Practices

⚠ Never do these

- Store JWT in `localStorage` (use `httpOnly` cookies)
- Expose stack traces in production errors
- Use `SELECT *` with user-controlled input (SQL injection)
- Store plain-text passwords

```
// Password hashing with bcrypt
const bcrypt = require('bcrypt');

const hashPassword = async (password) => {
  const saltRounds = 12;
  return bcrypt.hash(password, saltRounds);
};

const verifyPassword = async (password, hash) => {
  return bcrypt.compare(password, hash);
};
```

🚦 Rate Limiting

```
const rateLimit = require('express-rate-limit');

const limiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutes
  max: 100,                 // 100 requests per window
  message: 'Too many requests, please try again later'
});

app.use('/api/', limiter);
```

Video 10 — JavaScript Testing with Jest `1:00:34`

> 📋 Unit testing, integration testing, mocking with Jest

🧪 Testing Fundamentals

```
Error parsing Mermaid diagram!

No diagram type detected matching given configuration for text: pyramid
    title Testing Pyramid
    "E2E Tests" : 10
    "Integration Tests" : 30
    "Unit Tests" : 60
```

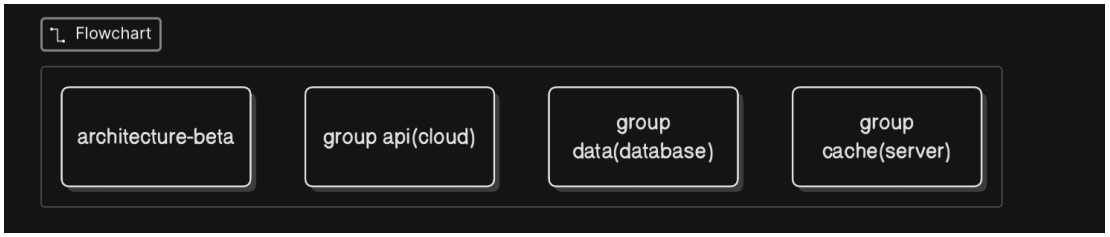
```
// Basic test structure
describe('UserService', () => {
  beforeAll(async () => { /* setup */ });
  afterEach(async () => { /* cleanup */ });
  afterAll(async () => { /* teardown */ });

  describe('createUser', () => {
    it('should create a user with valid data', async () => {
      const userData = { name: 'Alice', email: 'alice@test.com' };
```

```
      const user = await UserService.create(userData);

      expect(user).toBeDefined();
      expect(user.id).toBeTruthy();
      expect(user.email).toBe(userData.email);
    });

    it('should throw error for duplicate email', async () => {
      await expect(UserService.create({ email: 'exists@test.com' }))
        .rejects.toThrow('Email already exists');
    });
  });
});

// Mocking
jest.mock('../services/emailService');
const emailService = require('../services/emailService');
emailService.sendWelcome.mockResolvedValue({ success: true });
```

Video 11 — System Design for Beginners `1:25:07`

> 📋 How to design scalable systems — the concepts every backend dev needs

🏗️ System Design Components



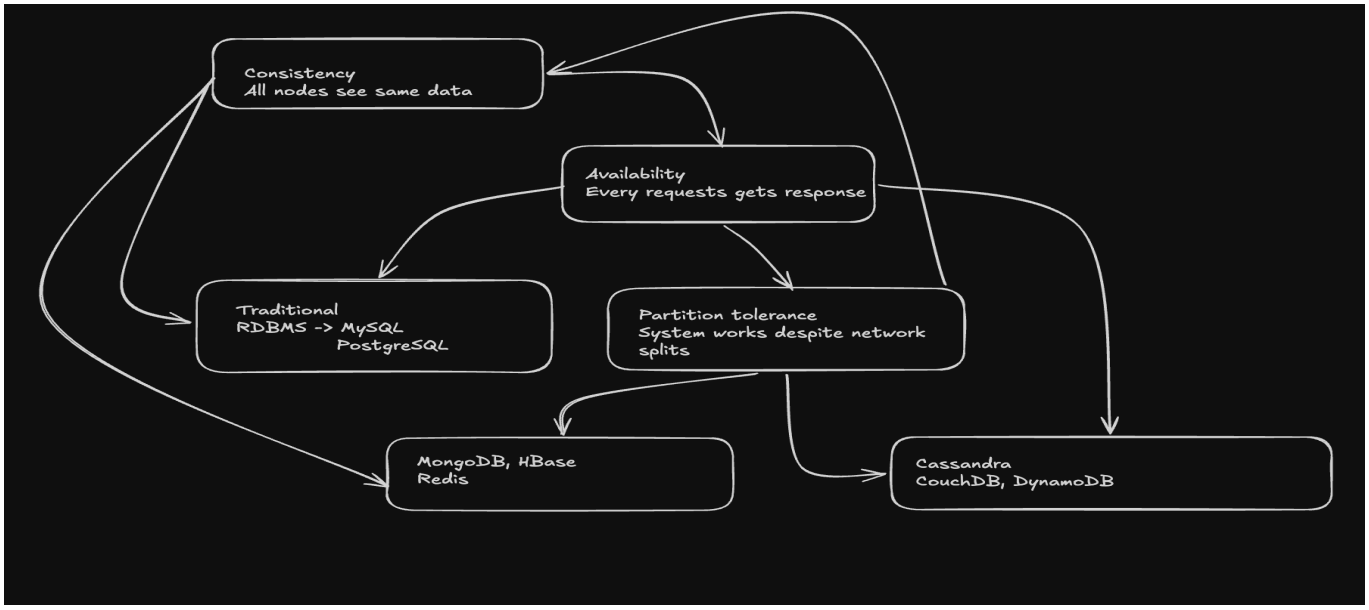
📐 Key System Design Concepts

Concept	Definition	When to Use
Horizontal Scaling	Add more servers	When CPU/memory is the bottleneck
Vertical Scaling	Upgrade server hardware	Quick fix, has limits
Caching	Store computed results	Frequent reads, expensive computations
Load Balancing	Distribute traffic	Multiple servers
CDN	Serve static assets from edge	Global users, static files
Message Queue	Async task processing	Long-running jobs, decoupling
Sharding	Split data across DBs	Massive datasets
Replication	Copy data to multiple DBs	High availability, read scaling

CAP Theorem

CAP theorem states that in a distributed system, you can only guarantee 2 out of 3 properties during network partitions: Consistency (C), Availability (A), or Partition Tolerance (P). The Three Properties • Consistency: Every read gets the latest write (all nodes show same data instantly) • Availability: Every request gets a response (system never says “unavailable”) • Partition Tolerance: System keeps working despite network splits between nodes

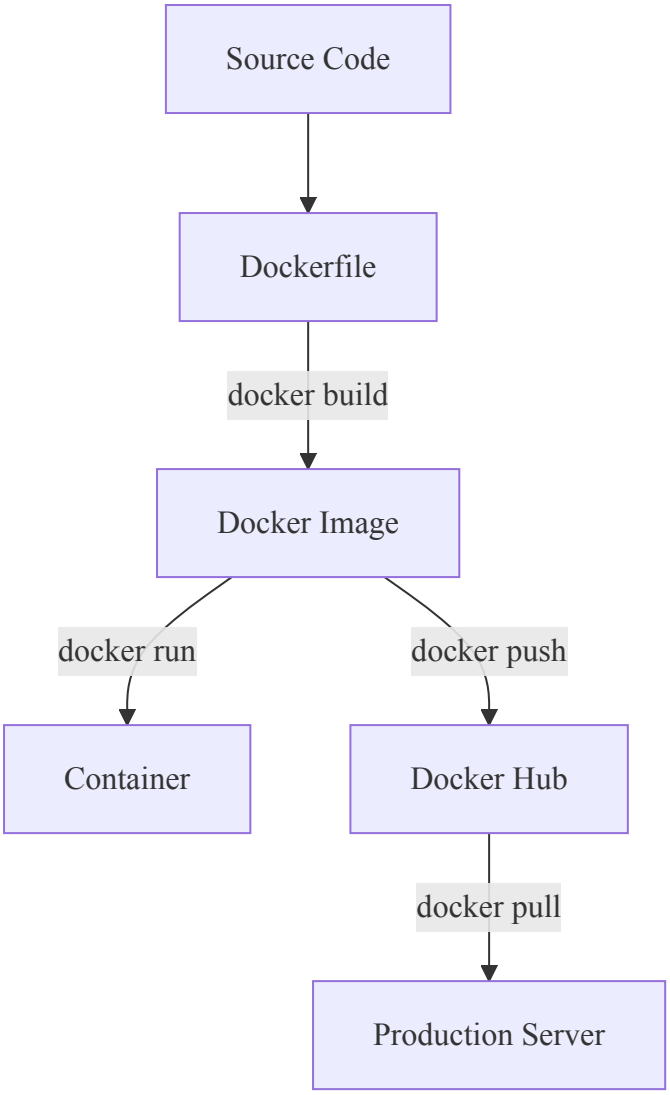
CAP: Choose 2 of 3



Video 12 — Docker & Kubernetes `5:56:37`

> Containerization from zero to orchestration

Docker Concepts



Dockerfile

```
# Multi-stage build for Node.js app
FROM node:20-alpine AS builder
WORKDIR /app
COPY package*.json ./
RUN npm ci --only=production

FROM node:20-alpine AS runtime
WORKDIR /app
COPY --from=builder /app/node_modules ./node_modules
COPY . .
EXPOSE 3000
USER node
CMD ["node", "server.js"]
```

Docker Compose

```
version: '3.8'

services:
  api:
    build: .
    ports:
      - "3000:3000"
```

```yaml
    environment:
      - DATABASE_URL=postgresql://user:pass@db:5432/mydb
      - REDIS_URL=redis://redis:6379
    depends_on:
      - db
      - redis
    restart: unless-stopped

  db:
    image: postgres:16-alpine
    volumes:
      - postgres_data:/var/lib/postgresql/data
    environment:
      POSTGRES_DB: mydb
      POSTGRES_USER: user
      POSTGRES_PASSWORD: pass

  redis:
    image: redis:7-alpine
    volumes:
      - redis_data:/data

volumes:
  postgres_data:
  redis_data:
```

Kubernetes Basics

```yaml
# Deployment
apiVersion: apps/v1
kind: Deployment
metadata:
  name: api-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: api
  template:
    metadata:
      labels:
        app: api
    spec:
      containers:
        - name: api
          image: myapp:v1.0.0
          ports:
            - containerPort: 3000
          resources:
            requests:
              memory: "128Mi"
              cpu: "250m"
            limits:
```

```
            memory: "256Mi"
            cpu: "500m"
---
# Service
apiVersion: v1
kind: Service
metadata:
  name: api-service
spec:
  selector:
    app: api
  ports:
    - port: 80
      targetPort: 3000
  type: LoadBalancer
```

Essential kubectl commands:

```
kubectl get pods                      # List pods
kubectl get services                  # List services
kubectl describe pod <name>           # Pod details
kubectl logs <pod-name>               # View logs
kubectl exec -it <pod> -- /bin/sh     # Shell into pod
kubectl apply -f deployment.yaml      # Apply config
kubectl scale deployment api --replicas=5
kubectl rollout undo deployment/api   # Rollback
```

Video 13 — GraphQL Course for Beginners `1:29:00`

> 📋 REST alternative: query exactly what you need

🔷 REST vs GraphQL

Feature	REST	GraphQL
Endpoints	Multiple (`/users` , `/posts`)	Single (`/graphql`)
Over-fetching	Common	Eliminated
Under-fetching	Requires N+1 requests	Single request
Type system	No standard	Built-in schema
Versioning	URL versioning	Schema evolution

📝 GraphQL Schema

```graphql
type User {
  id: ID!
  name: String!
  email: String!
  posts: [Post!]!
  createdAt: String!
}

type Post {
  id: ID!
  title: String!
  content: String!
  author: User!
  tags: [String!]!
}

type Query {
  user(id: ID!): User
  users(limit: Int, offset: Int): [User!]!
  posts(authorId: ID): [Post!]!
}

type Mutation {
  createUser(name: String!, email: String!): User!
  updateUser(id: ID!, name: String): User!
  deleteUser(id: ID!): Boolean!
  createPost(title: String!, content: String!, authorId: ID!): Post!
}

type Subscription {
  postCreated: Post!
  userOnline(userId: ID!): User!
}
```

🔍 GraphQL Queries

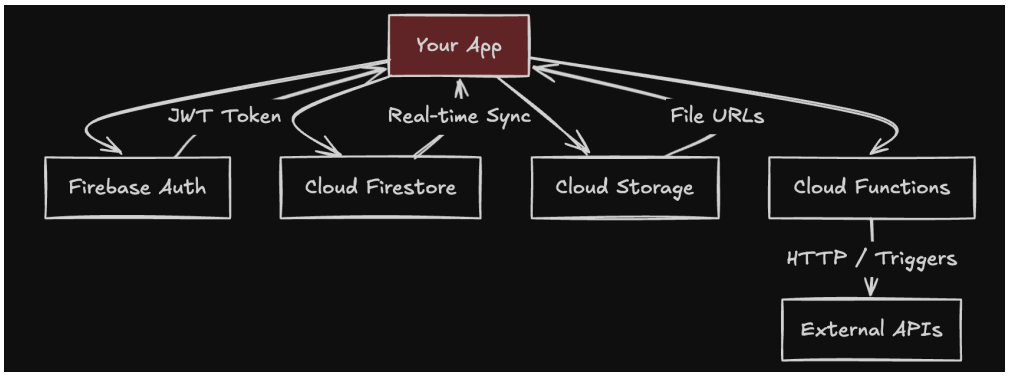
```graphql
# Query exactly what you need
query GetUserWithPosts {
  user(id: "123") {
    name
    email
    posts {
      title
      tags
    }
  }
}

# Mutation
mutation CreateUser {
  createUser(name: "Alice", email: "alice@example.com") {
    id
```

```
    name
  }
}

# Variables
query GetUser($userId: ID!) {
  user(id: $userId) {
    name
  }
}
```

Video 14 — Firebase Full Course 3:44:51

> 📋 Google's BaaS: Firestore, Auth, Storage, Functions

🔥 Firebase Architecture



📝 Firestore CRUD

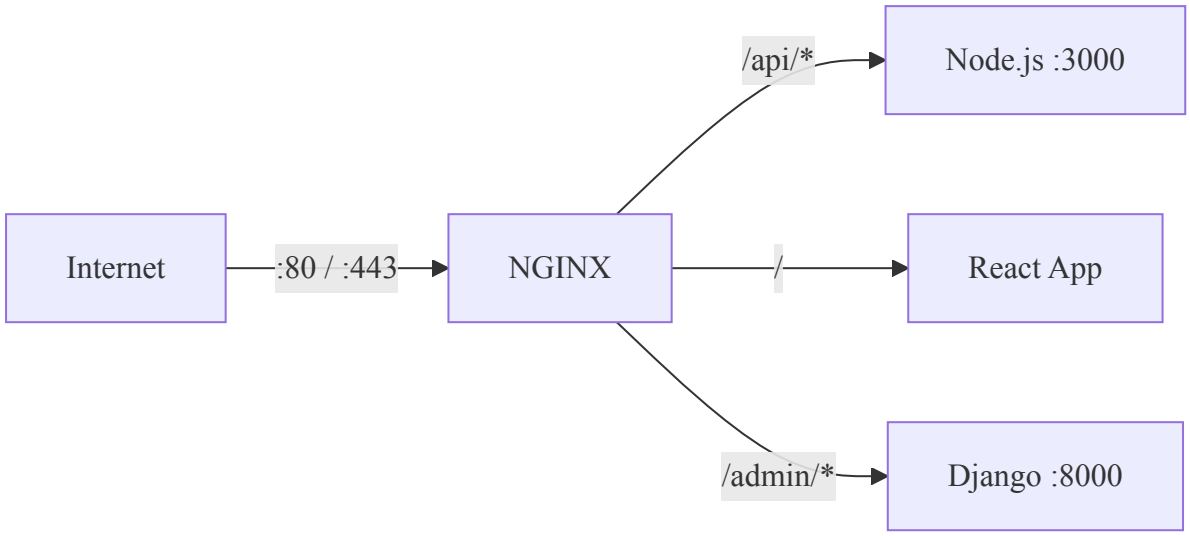
```
import { initializeApp } from 'firebase/app';
import { getFirestore, collection, doc, addDoc,
         getDoc, updateDoc, deleteDoc, query, where, getDocs } from
'firebase/firestore';

const db = getFirestore(app);

// Create
const docRef = await addDoc(collection(db, 'users'), {
  name: 'Alice',
  email: 'alice@example.com',
  createdAt: new Date()
});

// Read
const docSnap = await getDoc(doc(db, 'users', userId));
if (docSnap.exists()) console.log(docSnap.data());

// Query
const q = query(
  collection(db, 'users'),
```

```
  where('age', '>=', 18),
  orderBy('name')
);
const snapshot = await getDocs(q);
const users = snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));

// Real-time listener
onSnapshot(collection(db, 'messages'), (snapshot) => {
  snapshot.docChanges().forEach(change => {
    if (change.type === 'added') console.log('New message:', change.doc.data());
  });
});
```

Video 15 — NGINX Tutorial for Beginners `51:03`

> 📋 Web server, reverse proxy, load balancer

🔀 NGINX as Reverse Proxy



⚙️ NGINX Configuration

```
# /etc/nginx/sites-available/myapp

# Redirect HTTP to HTTPS
server {
    listen 80;
    server_name myapp.com www.myapp.com;
    return 301 https://$server_name$request_uri;
}

# Main HTTPS server
server {
    listen 443 ssl http2;
    server_name myapp.com www.myapp.com;
```

```
    ssl_certificate     /etc/letsencrypt/live/myapp.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/myapp.com/privkey.pem;

    # Security headers
    add_header X-Frame-Options DENY;
    add_header X-Content-Type-Options nosniff;
    add_header Strict-Transport-Security "max-age=31536000";

    # Static files
    location / {
        root /var/www/myapp/dist;
        try_files $uri $uri/ /index.html;
    }

    # API proxy
    location /api/ {
        proxy_pass http://localhost:3000;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

# Load balancing
upstream backend {
    least_conn;
    server backend1:3000;
    server backend2:3000;
    server backend3:3000;
}
```

Video 16 — Redis Course (In-Memory Database)

> 📋 Redis: caching, sessions, pub/sub, queues

⚡ What is Redis?

> ℹ️ Info
>
> Redis = **Re**mote **Di**ctionary **S**erver. An in-memory data structure store used as a cache, message broker, and database. Operations are $O(1)$ for most commands.

🗂️ Redis Data Types

Type	Commands	Use Case
String	`SET` , `GET` , `INCR`	Caching, counters
Hash	`HSET` , `HGET` , `HGETALL`	User sessions, objects
List	`LPUSH` , `RPOP` , `LRANGE`	Queues, recent activity
Set	`SADD` , `SMEMBERS` , `SINTER`	Tags, unique visitors
Sorted Set	`ZADD` , `ZRANGE` , `ZRANK`	Leaderboards, rate limiting

```
const redis = require('redis');
const client = redis.createClient({ url: process.env.REDIS_URL });

// Caching pattern
const getCachedUser = async (userId) => {
  const cacheKey = `user:${userId}`;

  // Try cache first
  const cached = await client.get(cacheKey);
  if (cached) return JSON.parse(cached);

  // Cache miss: fetch from DB
  const user = await db.findUserById(userId);

  // Store in cache with 1 hour TTL
  await client.setEx(cacheKey, 3600, JSON.stringify(user));

  return user;
};

// Session storage
await client.hSet(`session:${sessionId}`, {
  userId: user.id,
  email: user.email,
  loginAt: Date.now()
});
await client.expire(`session:${sessionId}`, 86400); // 24h

// Pub/Sub
const subscriber = client.duplicate();
await subscriber.subscribe('notifications', (message) => {
  console.log('Received:', JSON.parse(message));
});
await client.publish('notifications', JSON.stringify({ type: 'new_order', id: 123
}));
```

📚 Playlist 2 — Popular Programming Courses

Channel: freeCodeCamp.org | **Videos:** 13 | **Views:** 2M+

Video 1 — Harvard CS50 Full Course `24:51:37`

> The gold standard intro to CS — 13M+ views

CS50 Core Topics

Big O Notation

Complexity	Name	Example
$O(1)$	Constant	Array index access
$O(\log n)$	Logarithmic	Binary search
$O(n)$	Linear	Linear search
$O(n \log n)$	Linearithmic	Merge sort
$O(n^2)$	Quadratic	Bubble sort
$O(2^n)$	Exponential	Recursive Fibonacci
$O(n!)$	Factorial	Traveling salesman

Visual growth comparison:

$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(2^n) < O(n!)$$

🔄 Sorting Algorithms

```c
// Bubble Sort - O(n²)
void bubble_sort(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
            }
        }
    }
}

// Binary Search - O(log n)
int binary_search(int arr[], int n, int target) {
    int left = 0, right = n - 1;
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == target) return mid;
        if (arr[mid] < target) left = mid + 1;
        else right = mid - 1;
    }
    return -1;
}
```

🧩 Data Structures (C Implementation)

```c
// Linked List Node
typedef struct Node {
    int value;
    struct Node *next;
} Node;

// Hash Table
typedef struct {
    Node *buckets[BUCKET_COUNT];
} HashTable;

// Stack (array-based)
typedef struct {
    int items[MAX_SIZE];
    int top;
} Stack;

void push(Stack *s, int item) {
    if (s->top < MAX_SIZE - 1)
        s->items[++s->top] = item;
}
```

```c
int pop(Stack *s) {
    if (s->top >= 0) return s->items[s->top--];
    return -1; // empty
}
```

Video 2 — C++ Programming 31:07:30

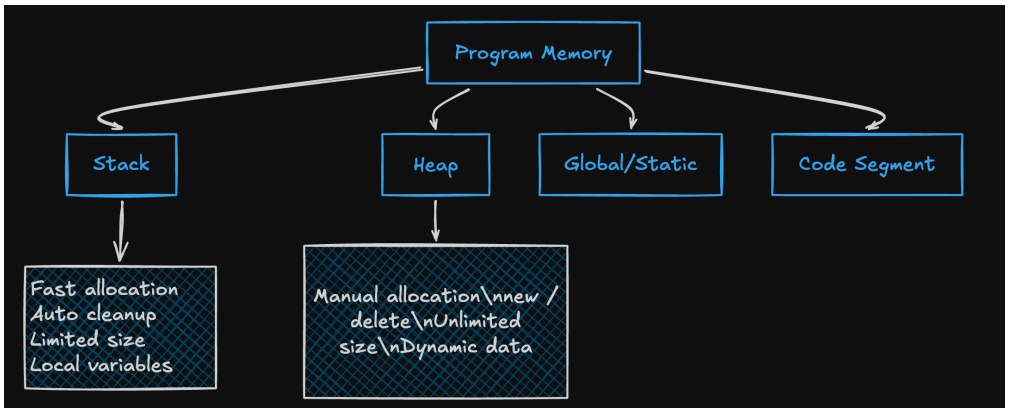
> Beginner to Advanced C++ — 7.5M views

⚡ C++ Key Concepts

```cpp
#include <iostream>
#include <vector>
#include <memory>
#include <algorithm>

// Modern C++ (C++17/20)

// RAII with smart pointers
auto ptr = std::make_unique<int>(42);
auto shared = std::make_shared<std::string>("hello");

// Templates
template<typename T>
T maximum(T a, T b) {
    return (a > b) ? a : b;
}

// STL Containers
std::vector<int> vec = {5, 2, 8, 1, 9};
std::sort(vec.begin(), vec.end());

// Lambda expressions
auto multiply = [](int x, int y) { return x * y; };
std::transform(vec.begin(), vec.end(), vec.begin(),
               [](int x) { return x * 2; });

// Classes with RAII
class Buffer {
    char* data;
    size_t size;
public:
    explicit Buffer(size_t sz) : data(new char[sz]), size(sz) {}
    ~Buffer() { delete[] data; }                    // RAII cleanup
    Buffer(const Buffer&) = delete;                  // No copy
    Buffer& operator=(const Buffer&) = delete;
    Buffer(Buffer&&) noexcept = default;             // Move OK
};
```

```cpp
// Structured bindings (C++17)
std::map<std::string, int> scores = {{"Alice", 95}, {"Bob", 87}};
for (const auto& [name, score] : scores) {
    std::cout << name << ": " << score << "\n";
}
```

🧠 Memory Model



Video 3 — Frontend Web Dev Bootcamp `21:14:42`

> 📋 HTML, CSS, JavaScript — 3.6M views

🎨 CSS Flexbox vs Grid

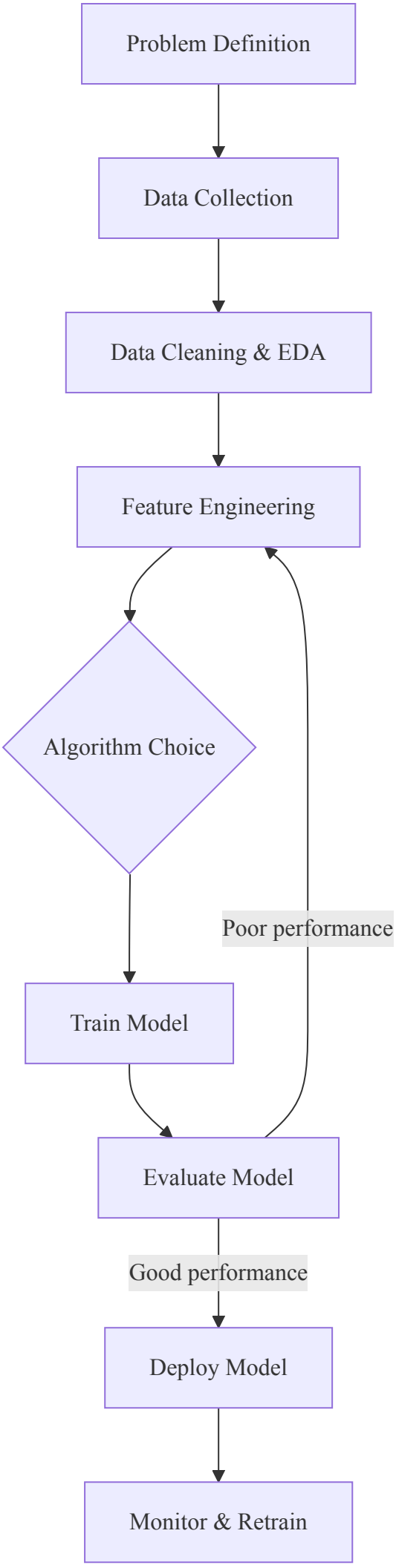
```css
/* Flexbox — 1D layout */
.flex-container {
  display: flex;
  flex-direction: row;          /* row | column */
  justify-content: space-between; /* main axis */
  align-items: center;          /* cross axis */
  flex-wrap: wrap;
  gap: 1rem;
}

.flex-item {
  flex: 1 1 200px; /* grow | shrink | basis */
}

/* Grid — 2D layout */
.grid-container {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
  grid-template-rows: auto;
  gap: 1.5rem;
}

/* Named grid areas */
.layout {
  display: grid;
```

```
  grid-template-areas:
    "header  header "
    "sidebar content"
    "footer  footer ";
  grid-template-columns: 250px 1fr;
}

.header  { grid-area: header; }
.sidebar { grid-area: sidebar; }
.content { grid-area: content; }
.footer  { grid-area: footer; }
```

Video 5 — Machine Learning for Everybody `3:53:53`

> 📋 ML fundamentals without heavy math — 9.8M views

🤖 ML Workflow



ML Algorithm Cheat Sheet

Type	Algorithm	Use Case
Classification	Logistic Regression, Random Forest, SVM	Spam, fraud detection
Regression	Linear Regression, XGBoost	Price prediction
Clustering	K-Means, DBSCAN	Customer segmentation
Dimensionality Reduction	PCA, t-SNE	Visualization
Neural Networks	CNN, RNN, Transformer	Images, NLP

```python
# Scikit-learn workflow
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix

# Split data
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Scale features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train_scaled, y_train)

# Evaluate
y_pred = model.predict(X_test_scaled)
print(classification_report(y_test, y_pred))
```

🧠 Neural Network Basics

$$\text{Output} = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

where σ is the activation function, w_i are weights, x_i are inputs, b is bias.

Common activation functions:

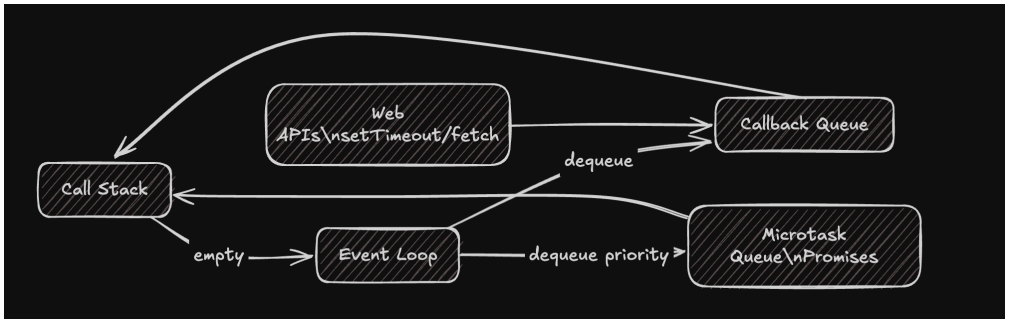
Function	Formula	Use
ReLU	$\max(0, x)$	Hidden layers
Sigmoid	$\frac{1}{1+e^{-x}}$	Binary output

Function	Formula	Use
Softmax	$\frac{e^{x_i}}{\sum e^{x_j}}$	Multi-class output
Tanh	$\tanh(x)$	Hidden layers (RNNs)

Video 7 — Learn JavaScript Full Course `3:26:43`

> JavaScript from scratch — 20M views

JS Event Loop

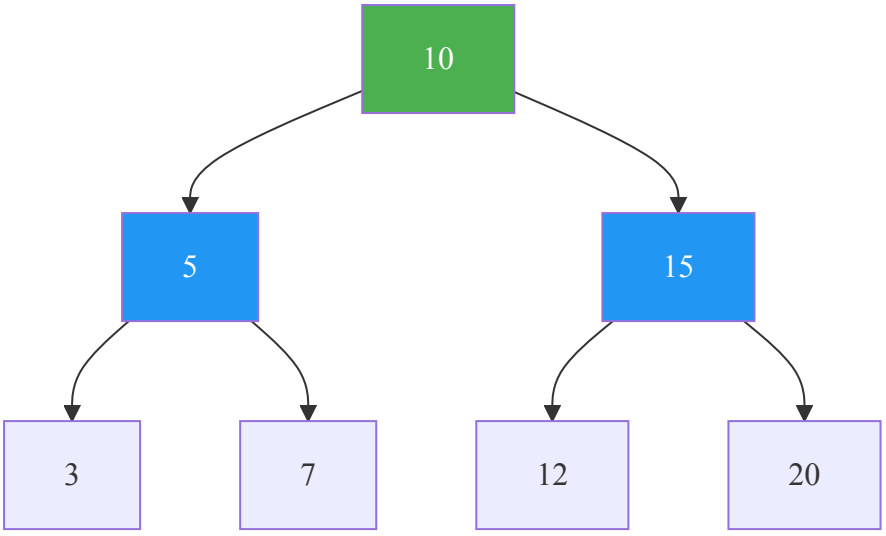



```
// Prototype & Classes
class EventEmitter {
  #listeners = new Map();

  on(event, callback) {
    if (!this.#listeners.has(event)) {
      this.#listeners.set(event, []);
    }
    this.#listeners.get(event).push(callback);
    return this;
  }

  emit(event, ...args) {
    this.#listeners.get(event)?.forEach(cb => cb(...args));
    return this;
  }

  off(event, callback) {
    const cbs = this.#listeners.get(event);
    if (cbs) this.#listeners.set(event, cbs.filter(cb => cb !== callback));
    return this;
  }
}
```

Video 8 — Data Structures `8:03:17`

📋 From basics to advanced — Google engineer — 7.3M views

🌳 Tree Data Structures



BST Operations complexity:

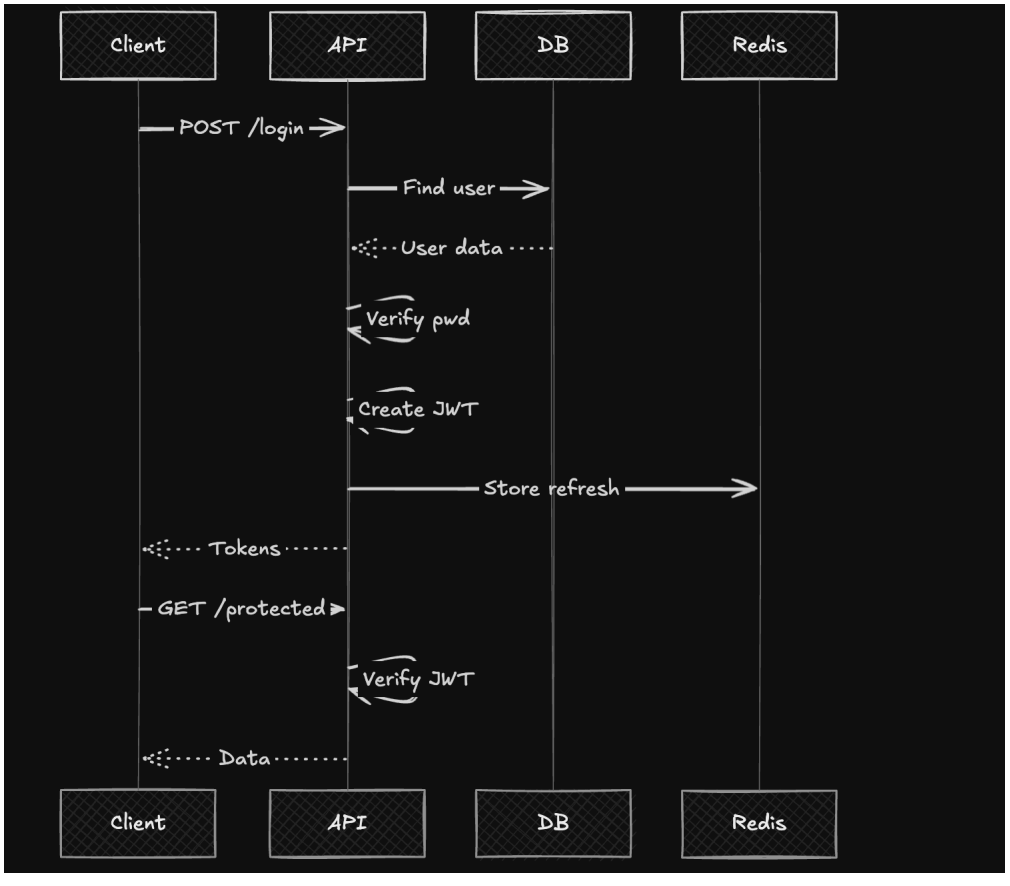
Operation	Average	Worst Case
Search	$O(\log n)$	$O(n)$
Insert	$O(\log n)$	$O(n)$
Delete	$O(\log n)$	$O(n)$

```
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None

class BST:
    def __init__(self):
        self.root = None

    def insert(self, val):
        def _insert(node, val):
            if not node:
                return TreeNode(val)
            if val < node.val:
                node.left = _insert(node.left, val)
            elif val > node.val:
                node.right = _insert(node.right, val)
            return node
        self.root = _insert(self.root, val)

    def inorder(self):
```

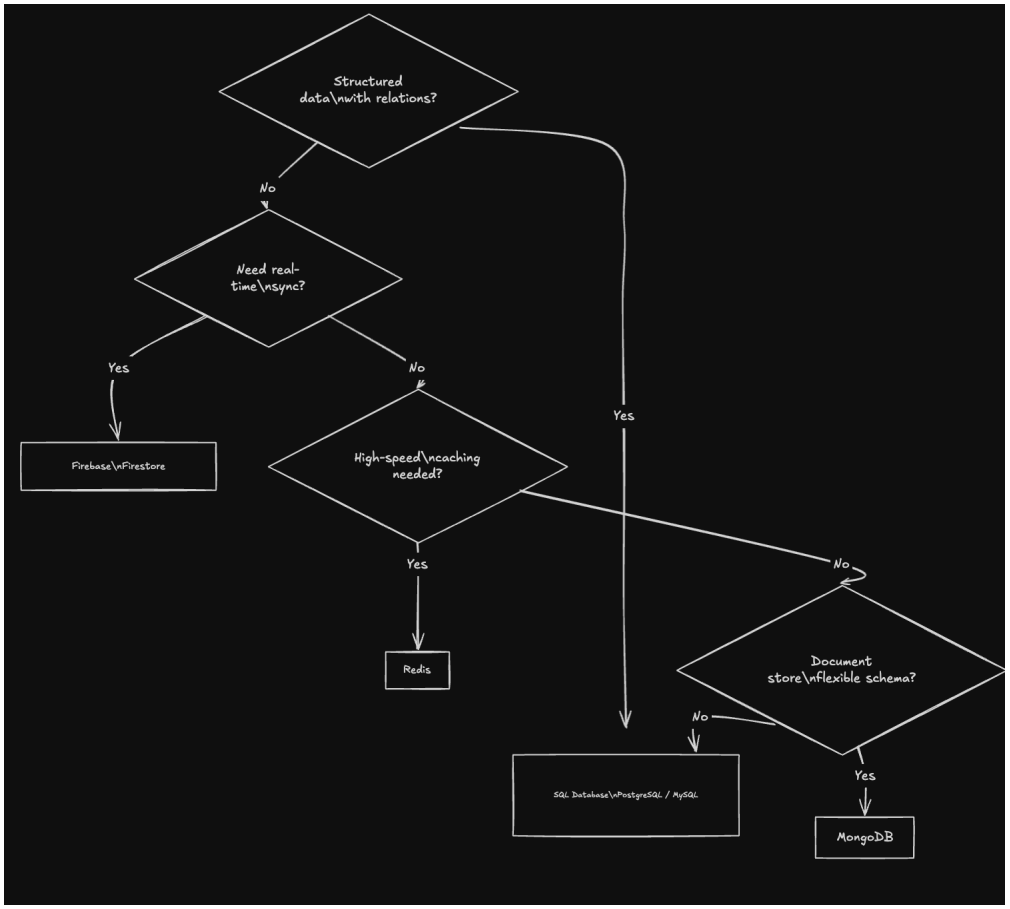
```
        result = []
        def _inorder(node):
            if node:
                _inorder(node.left)
                result.append(node.val)
                _inorder(node.right)
        _inorder(self.root)
        return result  # Returns sorted array!

# Graph BFS
from collections import deque

def bfs(graph, start):
    visited = {start}
    queue = deque([start])
    order = []

    while queue:
        node = queue.popleft()
        order.append(node)
        for neighbor in graph[node]:
            if neighbor not in visited:
                visited.add(neighbor)
                queue.append(neighbor)
    return order
```

🧠 Core Concepts Master Index

🔐 Authentication Flow



📊 Database Selection Guide



📊 Technology Comparison Tables

Backend Frameworks

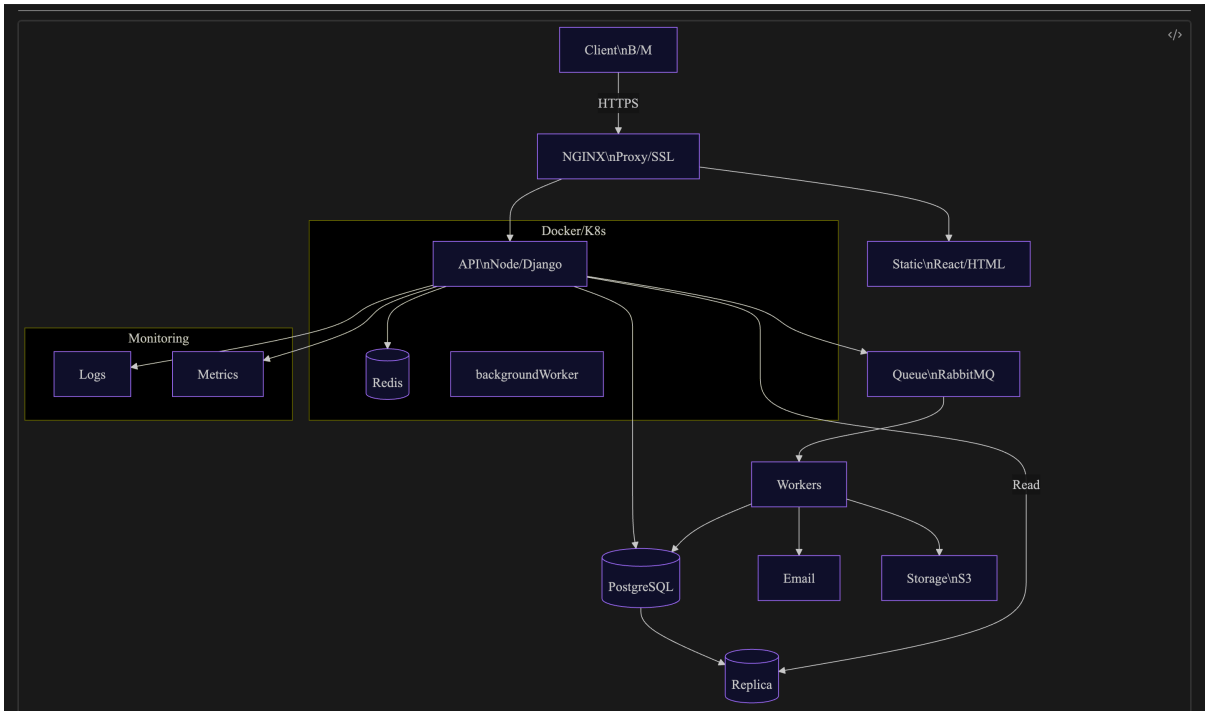
Framework	Language	Best For	Learning Curve
Django	Python	Full-stack, admin panels	Medium
FastAPI	Python	Modern APIs, async	Easy
Express	Node.js	Flexible APIs, real-time	Easy
Spring Boot	Java	Enterprise	Hard
Laravel	PHP	Rapid development	Medium
Rails	Ruby	Startups, prototyping	Medium

Database Comparison

Database	Type	Best For	Scaling
PostgreSQL	SQL	Complex queries, ACID	Vertical + Read replicas
MySQL	SQL	Web apps, WordPress	Vertical
MongoDB	Document	Flexible schema	Horizontal sharding

Database	Type	Best For	Scaling
Redis	In-memory	Caching, sessions	Cluster
Firestore	NoSQL	Real-time, mobile	Auto
Cassandra	Wide-column	Time-series, IoT	Horizontal

🔗 How Everything Connects



📝 Quick Reference Cheatsheets

https://devhints.io/

Git Cheatsheet

```
git init | clone | add | commit | push | pull | fetch | merge | rebase
git branch | checkout -b | merge | rebase | cherry-pick
git stash | stash pop | stash list
git log --oneline --graph --all
git reset --soft HEAD~1   # undo commit, keep staged
git reset --hard HEAD~1   # undo commit, lose changes
git revert <hash>          # safe undo for shared branches
```

Docker Cheatsheet

```
docker build -t myapp:v1 .        # Build image
docker run -p 3000:3000 myapp:v1  # Run container
docker ps -a                       # List all containers
docker exec -it <id> /bin/sh       # Shell into container
docker logs -f <id>                # Follow logs
```

```
docker compose up -d              # Start all services
docker compose down -v            # Stop + remove volumes
docker image prune -a             # Clean unused images
```

SQL Cheatsheet

```
SELECT, FROM, WHERE, GROUP BY, HAVING, ORDER BY, LIMIT
JOIN: INNER, LEFT, RIGHT, FULL OUTER, CROSS
FUNCTIONS: COUNT, SUM, AVG, MAX, MIN, COALESCE, NULLIF
STRING: UPPER, LOWER, LENGTH, TRIM, LIKE, ILIKE
DATE: NOW(), DATE_TRUNC, EXTRACT, AGE
WINDOW: ROW_NUMBER(), RANK(), DENSE_RANK(), LAG(), LEAD()
```

✓ You've Got This! till here

This roadmap represents ~100 hours of free high-quality content. Stay consistent: **30 minutes/day beats 5 hours on weekends.** now since you have got till here now just give your feedback -😀 😉

> ***refer this website also*** https://devhints.io/ https://overapi.com/ https://cheatography.com/ https://bytebytego.com/ https://en.algorithmica.org/hpc/ https://cp-algorithms.com/tags.html https://theartofhpc.com/ https://jenkov.com/ https://people.freebsd.org/~lstewart/articles/cpumemory.pdf

📚 Full-Stack Book Resources — Free PDFs & Reputed Sources

> Matched **topic-by-topic** to your `Full_stack_Notes.md` Sources: O'Reilly (free PDFs), GoalKicker, official docs, author-released books Written by: Aayush Ate | Last updated: 2026-05-10

🗂️ Master Free Book Hubs (Bookmark These First)

Source	Link	What You Get
Free O'Reilly PDFs (official list)	https://github.com/mohnkhan/Free-OReilly-Books	Direct `.pdf` download links from oreilly.com
GoalKicker	https://books.goalkicker.com	Free "Notes for Professionals" PDFs for every language & tool
EbookFoundation	https://github.com/EbookFoundation/free-programming-books	The biggest legal free book list on GitHub (⭐ 350k+)
Searchable version	https://ebookfoundation.github.io/free-programming-books-search/	Search the above by topic

Source	Link	What You Get
O'Reilly 10-day free trial	https://learning.oreilly.com/register/	Full library access — download before it expires

---

🗺️ Video 1 — Backend Roadmap

> Topics: Python · JavaScript · Go · Java · PostgreSQL · MongoDB · Redis · REST · GraphQL · gRPC · Docker · Kubernetes · JWT · OAuth2

Book	Source	Free Link
Backend Developer Roadmap (visual)	roadmap.sh	https://roadmap.sh/backend
Kubernetes (O'Reilly free PDF)	O'Reilly official	http://www.oreilly.com/webops-perf/free/files/kubernetes.pdf
Docker Security (O'Reilly free PDF)	O'Reilly official	http://www.oreilly.com/webops-perf/free/files/docker-security.pdf
Microservices (O'Reilly free report)	O'Reilly official	http://www.oreilly.com/programming/free/microservices.csp
Distributed Development Stack (O'Reilly)	O'Reilly official	http://www.oreilly.com/webops-perf/free/files/distributed-development-stack.pdf

---

🌐 Video 2 — Full Stack Web Dev (HTML · CSS · JS · Node.js · MongoDB)

HTML & CSS

Book	Source	Free Link
HTML5 Notes for Professionals	GoalKicker	https://books.goalkicker.com/HTML5Book/
CSS Notes for Professionals	GoalKicker	https://books.goalkicker.com/CSSBook/
CSS — The Definitive Guide (free sample)	O'Reilly	https://learning.oreilly.com/library/view/css-the-definitive/9781098117580/

Book	Source	Free Link
Learning Web Design (free sample)	O'Reilly	https://www.oreilly.com/library/view/learning-web-design/9781491960196/

JavaScript

Book	Source	Free Link
Eloquent JavaScript (full book free)	Author-released	https://eloquentjavascript.net/
You Don't Know JS (series, full free)	GitHub	https://github.com/getify/You-Dont-Know-JS
JavaScript.info (The Modern JS Tutorial)	Open	https://javascript.info/
JavaScript Notes for Professionals	GoalKicker	https://books.goalkicker.com/JavaScriptBook/
JavaScript: The Good Parts (O'Reilly)	O'Reilly free	http://www.oreilly.com/programming/free/files/the-good-parts-of-javascript-and-the-web.pdf
DOM Enlightenment (full book free)	Author-released	https://domenlightenment.com/

Node.js & Express

Book	Source	Free Link
Node.js Notes for Professionals	GoalKicker	https://books.goalkicker.com/NodeJSBook/
The Node Beginner Book (free part)	Author	https://www.nodebeginner.org/
Express.js Guide	GitHub	https://github.com/azat-co/expressjs-guide
Node.js Design Patterns (free chapter)	Packt/O'Reilly	https://www.nodejsdesignpatterns.com/

MongoDB

Book	Source	Free Link
MongoDB Notes for Professionals	GoalKicker	https://books.goalkicker.com/MongoDBBook/

Book	Source	Free Link
The Little MongoDB Book	Author-released	https://github.com/karlseguin/the-little-mongodb-book
MongoDB Manual (official)	MongoDB	https://www.mongodb.com/docs/manual/

🐍 Video 3 — Python Tutorial

Book	Source	Free Link
Automate the Boring Stuff with Python	Author-released	https://automatetheboringstuff.com/
Python Notes for Professionals	GoalKicker	https://books.goalkicker.com/PythonBook/
Think Python (2nd ed.)	Author-released	https://greenteapress.com/wp/think-python-2e/
Dive Into Python 3	Author-released	https://diveintopython3.problemsolving.io/
Fluent Python (official code)	O'Reilly GitHub	https://github.com/fluentpython/example-code-2e
Python Data Science Handbook	Author-released	https://jakevdp.github.io/PythonDataScienceHandbook/
Python for Everybody (free PDF)	Dr. Chuck	https://www.py4e.com/book
Real Python (free articles)	Open	https://realpython.com/
Introduction to Python (O'Reilly free)	O'Reilly	http://www.oreilly.com/programming/free/files/introduction-to-python.pdf
Python in Education (O'Reilly free)	O'Reilly	http://www.oreilly.com/programming/free/files/python-in-education.pdf

---

🏗️ Video 4 — Django

Book	Source	Free Link
Django Official Tutorial	Django	https://docs.djangoproject.com/en/stable/intro/tutorial01/
Django Girls Tutorial (free)	Open	https://tutorial.djangogirls.org/
Django Notes for Professionals	GoalKicker	https://books.goalkicker.com/DjangoBook/
Two Scoops of Django (free sample)	Author	https://www.feldroy.com/books/two-scoops-of-django-3-x
Test-Driven Development with Django (free sample)	O'Reilly	https://www.obeythetestinggoat.com/pages/book.html
Django for Beginners (free chapter)	Author	https://djangoforbeginners.com/

🌿 Video 5 — Git & GitHub

Book	Source	Free Link
Pro Git (full book free — official)	Author-released	https://git-scm.com/book/en/v2
Git Notes for Professionals	GoalKicker	https://books.goalkicker.com/GitBook/
Git Internals (O'Reilly free PDF)	O'Reilly	http://www.oreilly.com/programming/free/files/git-internals.pdf
Learn Git Branching (interactive)	Open	https://learngitbranching.js.org/
Oh Shit, Git! (quick ref)	Open	https://ohshitgit.com/

🗄️ Video 6 — SQL

Book	Source	Free Link
SQL Notes for Professionals	GoalKicker	https://books.goalkicker.com/SQLBook/
MySQL Notes for Professionals	GoalKicker	https://books.goalkicker.com/MySQLBook/
PostgreSQL Notes for Professionals	GoalKicker	https://books.goalkicker.com/PostgreSQLBook/
Use The Index, Luke (SQL performance)	Author	https://use-the-index-luke.com/

Book	Source	Free Link
Select Star SQL (interactive)	Author	https://selectstarsql.com/
SQLZoo (exercises)	Open	https://sqlzoo.net/
The Art of PostgreSQL (free intro)	Author	https://theartofpostgresql.com/
Learning SQL (O'Reilly — official code)	GitHub	https://github.com/ishitashah23/sql-practice-code

---

🔌 Video 7 — APIs for Beginners (REST · HTTP)

Book	Source	Free Link
HTTP Notes for Professionals	GoalKicker	https://books.goalkicker.com/HTTPBook/
APIs: A Strategy Guide (O'Reilly free)	O'Reilly	http://www.oreilly.com/web-platform/free/files/apis-strategy-guide.pdf
RESTful Web APIs (free sample)	O'Reilly	https://learning.oreilly.com/library/view/restful-web-apis/9781449359713/
REST API Design Rulebook (O'Reilly)	O'Reilly	https://learning.oreilly.com/library/view/rest-api-design/9781449317904/
HTTP: The Definitive Guide (O'Reilly)	O'Reilly	https://learning.oreilly.com/library/view/http-the-definitive/1565925092/
Postman Learning Center	Open	https://learning.postman.com/

⚡ Video 8 — FastAPI (Python API Development)

Book	Source	Free Link
FastAPI Official Docs (best tutorial)	FastAPI	https://fastapi.tiangolo.com/
FastAPI (O'Reilly book page)	O'Reilly	https://www.oreilly.com/library/view/fastapi/9781098135492/
FastAPI Cookbook (O'Reilly + Packt)	Packt	https://www.oreilly.com/library/view/fastapi-cookbook/9781805127857/

Book	Source	Free Link
Building Python Web APIs with FastAPI	Packt/O'Reilly	https://www.oreilly.com/library/view/building-python-web/9781801076630/
Building Python Microservices with FastAPI	Packt/O'Reilly	https://www.oreilly.com/library/view/building-python-microservices/9781803245966/
Full Stack FastAPI (free template + docs)	GitHub	https://github.com/fastapi/full-stack-fastapi-template
Pydantic Docs	Open	https://docs.pydantic.dev/
SQLAlchemy Docs	Open	https://docs.sqlalchemy.org/

🔐 Video 9 — API Security (JWT · OAuth2 · bcrypt · Rate Limiting)

Book	Source	Free Link
API Security in Action (O'Reilly)	O'Reilly	https://learning.oreilly.com/library/view/api-security-in/9781617296024/
OAuth 2 in Action (O'Reilly)	O'Reilly	https://learning.oreilly.com/library/view/oauth-2-in/9781617293276/
Web Application Security (O'Reilly free PDF)	O'Reilly	http://www.oreilly.com/webops-perf/free/files/web-application-security.pdf
OWASP Top 10 (free PDF)	OWASP	https://owasp.org/www-project-top-ten/
JWT.io Introduction	Open	https://jwt.io/introduction
The Web Application Hacker's Handbook	Open reference	https://github.com/OWASP/wstg

---

🧪 Video 10 — JavaScript Testing (Jest)

Book	Source	Free Link
JavaScript Testing Best Practices	GitHub	https://github.com/goldbergyoni/javascript-testing-best-practices

Book	Source	Free Link
Testing JavaScript (free articles)	Kent C. Dodds	https://testingjavascript.com/ (some free)
Jest Official Docs	Open	https://jestjs.io/docs/getting-started
Effective Testing with RSpec (O'Reilly)	O'Reilly	https://learning.oreilly.com/library/view/effective-testing-with/9781680502763/
Test-Driven Development — Kent Beck	O'Reilly reference	https://learning.oreilly.com/library/view/test-driven-development/0321146530/

🏗️ Video 11 — System Design

Book	Source	Free Link
Designing Data-Intensive Applications (DDIA) — references	GitHub	https://github.com/ept/ddia-references
System Design Primer (free, ⭐ 280k)	GitHub	https://github.com/donnemartin/system-design-primer
Grokking the System Design Interview (free notes)	GitHub	https://github.com/lei-hsia/grokking-system-design
Kubernetes for Full-Stack Developers (free PDF)	DigitalOcean	https://www.digitalocean.com/community/books/digitalocean-ebook-kubernetes-for-full-stack-developers
Site Reliability Engineering (free online)	Google	https://sre.google/sre-book/table-of-contents/
The Site Reliability Workbook (free online)	Google	https://sre.google/workbook/table-of-contents/

Book	Source	Free Link
Unsung Tools of DevOps (O'Reilly free PDF)	O'Reilly	http://www.oreilly.com/webops-perf/free/files/unsung-tools-of-devops.pdf
Modern Web Operations (O'Reilly free PDF)	O'Reilly	http://www.oreilly.com/webops-perf/free/files/modern-web-operations.pdf
ByteByteGo System Design	Alex Xu	https://bytebytego.com/ (your note already has this!)

🐋 Video 12 — Docker & Kubernetes

Docker

Book	Source	Free Link
Docker Notes for Professionals	GoalKicker	https://books.goalkicker.com/DockerBook/
Docker Security (O'Reilly free PDF)	O'Reilly official	http://www.oreilly.com/webops-perf/free/files/docker-security.pdf
Docker for Java Developers (O'Reilly free PDF)	O'Reilly official	http://www.oreilly.com/programming/free/files/docker-for-java-developers.pdf
Docker Official Docs	Open	https://docs.docker.com/
Play with Docker (interactive labs)	Open	https://labs.play-with-docker.com/
Docker Deep Dive (free chapter)	Nigel Poulton	https://nigelpoulton.com/books/

Kubernetes

Book	Source	Free Link
Kubernetes (O'Reilly free PDF)	O'Reilly official	http://www.oreilly.com/webops-perf/free/files/kubernetes.pdf

Book	Source	Free Link
Kubernetes for Java Developers (O'Reilly free PDF)	O'Reilly official	http://www.oreilly.com/programming/free/files/kubernetes-for-java-developers.pdf
Kubernetes for Full-Stack Developers (free PDF)	DigitalOcean	https://www.digitalocean.com/community/books/digitalocean-ebook-kubernetes-for-full-stack-developers
Kubernetes Up & Running (O'Reilly — official code)	GitHub	https://github.com/kubernetes-up-and-running/examples
Kubernetes Official Docs	Open	https://kubernetes.io/docs/home/
Kubernetes Deployment & Security Patterns (O'Reilly free PDF)	O'Reilly	https://www.oreilly.com/library/view/kubernetes-deployment-and/9781492056775/

🔷 Video 13 — GraphQL

Book	Source	Free Link
GraphQL Notes for Professionals	GoalKicker	https://books.goalkicker.com/GraphQLBook/
The GraphQL Guide (free intro)	Author	https://graphql.guide/
GraphQL Official Docs	Open	https://graphql.org/learn/
Fullstack GraphQL (free sample)	Author	https://www.graphql.college/fullstack-graphql/
Learning GraphQL (O'Reilly code)	GitHub	https://github.com/MoonHighway/learning-graphql

---

🔥 Video 14 — Firebase

Book	Source	Free Link
Firebase Official Docs	Google	https://firebase.google.com/docs
Firebase Fundamentals (free codelab)	Google	https://firebase.google.com/codelabs/firebase-web

Book	Source	Free Link
Firebase for Beginners (free guide)	Google	https://firebase.google.com/docs/guides

---

🤖 Video 15/16 — Machine Learning & Neural Networks

Book	Source	Free Link
Hands-On ML with Scikit-Learn, Keras & TF (code)	O'Reilly GitHub	https://github.com/ageron/handson-ml3
Dive into Deep Learning (full book free)	Author-released	https://d2l.ai/
Neural Networks and Deep Learning	Author-released	http://neuralnetworksanddeeplearning.com/
Pattern Recognition & Machine Learning — Bishop	Microsoft Research	https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/
Probabilistic ML — An Introduction — Murphy	Author-released	https://probml.github.io/pml-book/book1.html
Reinforcement Learning: An Introduction — Sutton & Barto	Author-released	http://incompleteideas.net/book/the-book-2nd.html

Book	Source	Free Link
Machine Learning from Scratch — Friedman	Author-released	https://dafriedman97.github.io/mlbook/
Python Machine Learning Projects	DigitalOcean free PDF	https://www.digitalocean.com/community/books/pythonmachinelearningproj
Getting Started with AI (O'Reilly free PDF)	O'Reilly	http://www.oreilly.com/data/free/files/getting-started-with-artificial-intelligence.pdf

---

🔄 General Full-Stack / DevOps

Book	Source	Free Link
Flask Web Development (official code)	GitHub	https://github.com/miguelgrinberg/flasky
Full Stack Python	Author-released	https://www.fullstackpython.com/
CI/CD with Docker and Kubernetes (free PDF)	SemaphoreCI	https://semaphoreci.com/resources/cicd-docker-kubernetes
CI/CD for Monorepos (free PDF)	SemaphoreCI	https://semaphoreci.com/resources/monorepos
The Twelve-Factor App	Author	https://12factor.net/
NGINX Cookbook (O'Reilly free PDF)	O'Reilly	https://www.nginx.com/resources/library/complete-nginx-cookbook/

🧠 Your Note's Reference Sites — Expanded Resources

> From the bottom of your `Full_stack_Notes.md`

Site (already in your notes)	What to Add From It
https://devhints.io/	Cheatsheets for every tool in your notes
https://overapi.com/	API method quick reference
https://bytebytego.com/	System design diagrams — pair with **System Design Primer** on GitHub
https://cp-algorithms.com/	Competitive programming algorithms — pair with **Open Data Structures** PDF
https://en.algorithmica.org/hpc/	HPC algorithms — pair with **Is Parallel Programming Hard?** free PDF
https://jenkov.com/	Java + concurrency — pair with **Java Concurrency in Practice** (O'Reilly)

Additional book for each site:

Site	Best Matching Free Book
bytebytego.com	**System Design Primer** https://github.com/donnemartin/s
cp-algorithms.com	**Competitive Programmer's Handbook** (free PDF) https
en.algorithmica.org/hpc	**Is Parallel Programming Hard?** https://mirrors.edge.kernel.org/pub/linux/kernel/people/pa
jenkov.com	**Java Notes for Professionals** https://books.goalkicker.co
people.freebsd.org/~lstewart/articles/cpumemory.pdf	Already a free PDF ✅ (What Every Programmer Should

---

📦 All O'Reilly Official Free PDFs (Direct Download Links)

> From the official O'Reilly free books list — these are `.oreilly.com` hosted, permanent links

Web & DevOps

```
Docker Security:
http://www.oreilly.com/webops-perf/free/files/docker-security.pdf

Kubernetes:
http://www.oreilly.com/webops-perf/free/files/kubernetes.pdf

Modern Web Operations:
http://www.oreilly.com/webops-perf/free/files/modern-web-operations.pdf

Distributed Development Stack:
http://www.oreilly.com/webops-perf/free/files/distributed-development-stack.pdf

Unsung Tools of DevOps:
```

```
http://www.oreilly.com/webops-perf/free/files/unsung-tools-of-devops.pdf

Web Application Security:
http://www.oreilly.com/webops-perf/free/files/web-application-security.pdf
```

Python & Programming

```
Introduction to Python:
http://www.oreilly.com/programming/free/files/introduction-to-python.pdf

Python in Education:
http://www.oreilly.com/programming/free/files/python-in-education.pdf

Microservices:
http://www.oreilly.com/programming/free/files/microservices.pdf

Kubernetes for Java Developers:
http://www.oreilly.com/programming/free/files/kubernetes-for-java-developers.pdf

Docker for Java Developers:
http://www.oreilly.com/programming/free/files/docker-for-java-developers.pdf

Building Reactive Microservices in Java:
http://www.oreilly.com/programming/free/files/building-reactive-microservices-in-
java.pdf

JavaScript: The Good Parts (related):
http://www.oreilly.com/programming/free/files/the-good-parts-of-javascript-and-
the-web.pdf

Git Internals:
http://www.oreilly.com/programming/free/files/git-internals.pdf
```

Data & AI

```
Getting Started with Artificial Intelligence:
http://www.oreilly.com/data/free/files/getting-started-with-artificial-
intelligence.pdf
```

🔗 Topic → Book Quick-Reference

Your Notes Topic	Best Free Book	Link
HTML/CSS	HTML5 + CSS Notes for Professionals	https://books.goalkicker.com/HTML5Book/
JavaScript	Eloquent JavaScript	https://eloquentjavascript.net/

Your Notes Topic	Best Free Book	Link
JS Deep	You Don't Know JS	https://github.com/getify/You-Dont-Know-JS
Node.js	Node.js Notes for Professionals	https://books.goalkicker.com/NodeJSBook/
MongoDB	The Little MongoDB Book	https://github.com/karlseguin/the-little-mongodb-book
Python	Automate the Boring Stuff	https://automatetheboringstuff.com/
Django	Django Girls Tutorial	https://tutorial.djangogirls.org/
FastAPI	FastAPI Official Docs	https://fastapi.tiangolo.com/
Git	Pro Git (official, free)	https://git-scm.com/book/en/v2
SQL	SQL Notes for Professionals	https://books.goalkicker.com/SQLBook/
REST APIs	APIs: A Strategy Guide (O'Reilly free)	http://www.oreilly.com/web-platform/free/files/apis-strategy-guide.pdf
API Security	OWASP Top 10	https://owasp.org/www-project-top-ten/
Testing (Jest)	JS Testing Best Practices	https://github.com/goldbergyoni/javascript-testing-best-practices
System Design	System Design Primer	https://github.com/donnemartin/system-design-primer
Docker	Docker Security (O'Reilly free)	http://www.oreilly.com/webops-perf/free/files/docker-security.pdf
Kubernetes	Kubernetes (O'Reilly free)	http://www.oreilly.com/webops-perf/free/files/kubernetes.pdf
GraphQL	The GraphQL Guide	https://graphql.guide/
Firebase	Firebase Official Docs	https://firebase.google.com/docs
ML / Neural Nets	Dive into Deep Learning	https://d2l.ai/
Algorithms	Competitive Programmer's Handbook	https://cses.fi/book/book.pdf

All PDF links are legally free — official author releases, O'Reilly free tier, or open-access publications. and if links not working search title of it

Notes compiled from freeCodeCamp.org YouTube Playlists and books. All code examples are original summaries for learning purposes. This the interesting things i have seen in youtube from freeCodeCamp, youtube and books there are other channels like bytebytego and geeks which i am very interested in there content and knowledge ***--Aayush Ate***