## Web Warriors ⚔️

Objective: Learn the building blocks of the web, how the internet works, and how the protocols make communication happen.

#### 1. How the Internet Works

* ●Introduction to the concept of the Internet
* ●Overview of the World Wide Web (WWW) and its connection to the internet
* ●How data is transferred across networks
* ●Understanding IP addresses, domain names, and routing
* ●Key Concepts: Internet Service Providers (ISPs), Routers, DNS

#### 2. DNS Magic and Internals

* ●What is DNS (Domain Name System)?
* ●How DNS resolves domain names to IP addresses
* ●Types of DNS records: A, CNAME, MX, and more
* ●DNS Hierarchy: Root DNS servers, TLDs, and authoritative DNS servers
* ●How browsers query DNS servers to load websites
* ●Key Concepts: Recursive queries, Caching, TTL (Time-to-Live)

#### 3. Server-Client Architecture

* ●What is a Client-Server model?
* ●Differences between Client and Server in web applications
* ●HTTP request-response cycle
* ●How browsers act as clients that request resources from web servers
* ●Introduction to web servers and web hosting
* ●Key Concepts: Web servers (Apache, Nginx), Client-side vs Server-side, Request headers, Response codes

#### 4. Internet Protocols

* ●Introduction to network protocols: What are they, and why are they needed?
* ●The role of protocols in ensuring data transfer integrity and reliability
* ●Overview of key protocols: HTTP, HTTPS, FTP, and SMTP
* ●How secure protocols like HTTPS ensure encrypted communication
* ●Key Concepts: Protocol stacks, Handshakes, Encryption

#### 5. TCP/IP

* ●What is TCP/IP and why it is fundamental for data transmission
* ●How TCP ensures reliable data transfer via packet acknowledgment and retransmission
* ●The role of IP in addressing and routing data packets across networks
* ●Understanding ports and sockets in TCP/IP communication
* ●Key Concepts: Three-way handshake, Packet loss, Routing tables

#### 6. UDP (User Datagram Protocol)

* ●What is UDP, and how does it differ from TCP?
* ●Understanding when and why UDP is used (e.g., in video streaming or gaming)
* ●Comparison of TCP and UDP performance (reliability vs speed)
* ●Key Concepts: Datagram-based transmission, Low overhead, Connectionless communication

#### 7. TCP Handshakes and 3-Way Handshakes

* ●What is a 3-way handshake in TCP?
* ●Detailed breakdown of the 3 phases: SYN, SYN-ACK, and ACK
* ●The purpose of the handshake: Establishing a reliable connection
* ●How data is transmitted after the handshake is complete
* ●Key Concepts: Reliable connection establishment, Sequence numbers, Acknowledgments

#### 8. HTTP & HTTPS Protocols

* ●Introduction to HTTP (HyperText Transfer Protocol) and HTTPS (Secure version of HTTP)
* ●What happens during an HTTP request-response cycle?
* ●Understanding status codes: 200 OK, 404 Not Found, 500 Internal Server Error
* ●Introduction to SSL/TLS and how it secures data during transmission
* ●Key Concepts: Request methods (GET, POST, PUT, DELETE), HTTPS handshake, Certificate Authorities

## Spider-Man 🕸️

Objective: Learn the foundations of web design by mastering HTML and CSS, creating the structure and styling that powers the web.

#### 1. HTML Basics – The Web’s Skeleton

* ●Introduction to HTML (HyperText Markup Language) and its role in web development
* ●Understanding HTML tags and elements
* ●Building a simple webpage using HTML: html, head, body, header, footer
* ●Working with text elements: h1, p, a, ul, ol, li
* ●Structuring content with semantic HTML: section, article, nav, main, aside
* ●Key Concepts: Elements, Tags, Nesting, Attributes, Semantic HTML

#### 2. HTML Forms and Inputs – User Interaction

* ●Creating forms with form, input, textarea, select, button
* ●Understanding form submission, GET and POST methods
* ●Using input types (text, email, number, password)
* ●Validating form data with attributes like required, min, max
* ●Key Concepts: Form elements, Input validation, Method types, Accessibility in forms

#### 3. CSS Basics – Styling the Web

* ●Introduction to CSS (Cascading Style Sheets) and its purpose
* ●Understanding the CSS box model: margin, border, padding, content
* ●Styling text, colors, and fonts with color, font-family, font-size, line-height
* ●Understanding the concept of specificity and how CSS selectors work
* ●Applying styles to HTML elements using selectors, classes, and IDs
* ●Key Concepts: Box model, Selectors, Specificity, Inline vs external CSS

#### 4. CSS Layouts – Building Responsive Pages

* ●Introduction to layout techniques in CSS: display, position, float, flexbox, grid
* ●Building layouts with Flexbox: Aligning items, creating rows and columns
* ●Introduction to CSS Grid: Creating complex grid-based layouts
* ●Media queries: Making websites responsive to different screen sizes
* ●Key Concepts: Flexbox, Grid layout, Responsive design, Mobile-first approach

#### 5. Advanced CSS Styling

* ●Using pseudo-classes and pseudo-elements: :hover, :focus, :nth-child
* ●Animations and transitions: Making elements move or change on user interaction
* ●Styling links, buttons, and forms for better UX
* ●Key Concepts: Pseudo-classes, Pseudo-elements, Animations, Transitions

#### 6. CSS Frameworks – Speeding Up Development

* ●Introduction to popular CSS frameworks: Bootstrap, TailwindCSS
* ●How to use a CSS framework to quickly style pages
* ●Customizing and overriding default styles in a framework
* ●Key Concepts: Grid systems, Utility-first design, Responsive frameworks

#### 7. HTML5 & CSS3 Features

* ●HTML5 semantic elements: header, footer, main, article, section
* ●New input types in HTML5: date, email, tel, range, color
* ●CSS3 properties: border-radius, box-shadow, gradient, transitions
* ●Key Concepts: Modern HTML5 elements, Advanced CSS3 techniques, Cross-browser compatibility

## JavaScript Surgeons 🧑🏻‍⚕️

Objective: Master the fundamentals and advanced concepts of JavaScript, as well as the DOM, in scripting dynamic web pages.

#### 1. JavaScript Basics – The Language of the Web

* ●Introduction to JavaScript and its role in web development
* ●Understanding variables and data types: string, number, boolean, object, array, null, undefined
* ●Declaring variables with var, let, const
* ●Understanding scope (Global, Local, Block Scope) and hoisting
* ●Basic operators: +, -, \*, /, %, ==, ===
* ●Control flow statements: if, else, else if, switch, ternary operator
* ●Key Concepts: Variables, Data Types, Operators, Conditionals

#### 2. JavaScript Functions – Reusable Code

* ●What are JavaScript functions? Why are they important?
* ●Declaring functions with the function keyword
* ●Understanding parameters, arguments, and return values
* ●Arrow functions: A modern way to define functions
* ●Function scope and closures
* ●Key Concepts: Function Declaration, Function Expression, Parameters vs Arguments, Return Value, Scope, Closures

#### 3. JavaScript Loops – Automating Repetitive Tasks

* ●Introduction to loops: for, while, and do...while loops
* ●Looping through arrays
* ●Loop control statements: break and continue
* ●Using for...in and for...of loops
* ●Key Concepts: Iteration, Loop control, Array iteration

#### 4. JavaScript Arrays – Working with Collections of Data

* ●What is an array?
* ●Creating arrays, accessing elements by index
* ●Array methods: push, pop, shift, unshift, splice, slice, indexOf, etc.
* ●Using loops to iterate through arrays
* ●Multi-dimensional arrays
* ●Key Concepts: Array indices, Array methods, Array iteration

#### 5. JavaScript Objects – Working with Key-Value Pairs

* ●What is an object?
* ●Creating objects with literal notation and constructor functions
* ●Accessing object properties
* ●Adding, updating, and deleting object properties
* ●Object methods
* ●Key Concepts: Object properties, Object methods, Object iteration

#### 6. DOM Manipulation – Interacting with the Webpage

* ●Introduction to the DOM (Document Object Model)
* ●Selecting elements in the DOM: getElementById, getElementsByClassName, querySelector, querySelectorAll
* ●Modifying element content: innerHTML, textContent
* ●Modifying element attributes
* ●Creating and appending new elements to the DOM
* ●Key Concepts: DOM tree, Element selection, Content manipulation, Dynamic HTML

#### 7. JavaScript Events – Responding to User Actions

* ●Introduction to events: click, mouseover, keypress, submit, etc.
* ●Adding event listeners to elements
* ●Event handlers and callback functions
* ●Event bubbling and capturing
* ●Preventing default event behavior
* ●Key Concepts: Event listeners, Event handlers, Event propagation

#### 8. Asynchronous JavaScript – Handling Delays

* ●Introduction to asynchronous programming and why it's important
* ●Callbacks and callback hell
* ●Promises: a better way to handle asynchronous operations
* ●Async/Await syntax: A cleaner way to handle promises
* ●Making API requests with fetch
* ●Key Concepts: Asynchronous operations, Callbacks, Promises, Async/Await

#### 9. ES6+ Features – Modern JavaScript

* ●Introduction to ECMAScript 6 (ES6) and beyond
* ●Let and const: improved variable declarations
* ●Arrow functions: concise function syntax
* ●Template literals: string interpolation
* ●Destructuring: extracting values from arrays and objects
* ●Spread and Rest operators
* ●Classes: a more object-oriented approach
* ●Modules: organizing code
* ●Key Concepts: Modern JS syntax, improved code organization, new language features

#### 10. Working with APIs – Fetching Data from Servers

* ●What are APIs?
* ●Introduction to RESTful APIs
* ●Making API requests with fetch
* ●Handling API responses
* ●Working with JSON data
* ●Key Concepts: API endpoints, HTTP methods, JSON parsing, Asynchronous data fetching

#### 11. Error Handling – Debugging JavaScript

* ●What are JavaScript errors?
* ●Try...catch statements: handling errors gracefully
* ●Throwing custom errors
* ●Debugging tools and techniques: console.log, browser debugger
* ●Key Concepts: Error types, try...catch, Debugging

#### 12. JavaScript Testing – Ensuring Code Quality

* ●Introduction to JavaScript testing
* ●Unit testing basics
* ●Testing frameworks: Jest, Mocha
* ●Writing tests for functions and modules
* ●Key Concepts: Unit tests, Test frameworks, Code Quality

#### 13. JavaScript Best Practices and Patterns

* ●Code organization and structure
* ●Writing clean and maintainable code
* ●Design patterns: common solutions to recurring problems
* ●Avoiding common mistakes
* ●Key Concepts: Code maintainability, Design patterns, Code quality

## Frequently

## Node Ninja 🥷

Objective: Master backend development with Node.js, learning how to create powerful server-side applications, handle HTTP requests, and connect with databases.

#### 1. Introduction to Node.js – The Power of JavaScript on the Server

* ●What is Node.js and how it differs from traditional server-side languages
* ●Understanding the Node.js runtime environment
* ●The event-driven, non-blocking I/O model in Node.js
* ●Setting up a simple Node.js application
* ●Installing and using Node.js with npm (Node Package Manager)
* ●Key Concepts: Event-driven architecture, Non-blocking I/O, npm, Modules

#### 2. Understanding the Event Loop – Node.js Architecture

* ●What is the event loop and how does Node.js handle concurrency
* ●How Node.js uses the event loop to process requests asynchronously
* ●Blocking vs Non-blocking code execution
* ●The importance of callbacks and promises in managing asynchronous code
* ●Key Concepts: Event loop, Asynchronous processing, Callbacks, Promises

#### 3. Creating a Basic HTTP Server

* ●How to create a basic HTTP server with Node.js using the `http` module
* ●Setting up routes to handle different HTTP requests (GET, POST, PUT, DELETE)
* ●Sending and receiving data with the server
* ●Working with request and response objects
* ●Key Concepts: HTTP server, Request/Response objects, Routing

#### 4. Express.js – Simplifying Backend Development

* ●Introduction to Express.js and how it simplifies Node.js backend development
* ●Setting up an Express app and defining routes
* ●Handling dynamic data with URL parameters and query strings
* ●Middleware in Express: What is middleware and how to use it
* ●Built-in Express middleware functions (e.g., body-parser, cookie-parser)
* ●Key Concepts: Express.js, Routing, Middleware, Request handling

#### 5. RESTful API Design – Building APIs with Express

* ●What is a RESTful API and how to structure it
* ●Designing endpoints and handling HTTP methods (GET, POST, PUT, DELETE)
* ●Using query parameters and request bodies for passing data
* ●Returning JSON data and handling status codes in API responses
* ●Key Concepts: REST API design, CRUD operations, Status codes, JSON responses

#### 6. Working with Databases – Connecting Node.js to Databases

* ●Introduction to databases (SQL vs NoSQL)
* ●Using MongoDB with Node.js (Setting up MongoDB, connecting via Mongoose)
* ●Working with CRUD operations in MongoDB (Create, Read, Update, Delete)
* ●Introduction to SQL databases (using MySQL/PostgreSQL with Node.js)
* ●Using ORMs (Object Relational Mappers) like Drizzle, Prisma to interact with SQL databases
* ●Key Concepts: MongoDB, SQL, NoSQL, CRUD, Mongoose, Sequelize, ORMs

#### 7. Authentication and Authorization – Securing Your Application

* ●Introduction to user authentication and authorization concepts
* ●Using JWT (JSON Web Tokens) for stateless authentication
* ●Setting up user login and registration endpoints
* ●Password hashing with bcrypt.js
* ●Role-based access control and securing routes with middleware
* ●Key Concepts: Authentication, Authorization, JWT, bcrypt, Role-based access control

#### 8. Working with File Systems – Reading and Writing Files

* ●Introduction to the Node.js `fs` module for file system operations
* ●Reading files asynchronously and synchronously
* ●Writing files to the server’s file system
* ●Handling file uploads (e.g., using `multer` for handling multipart forms)
* ●Key Concepts: File system module, File reading/writing, File uploads

#### 9. Building Real-time Applications – WebSockets with Socket.io

* ●What are WebSockets and how they enable real-time communication
* ●Setting up a WebSocket server using the `ws` module or Socket.io
* ●Sending and receiving real-time data between the client and server
* ●Use cases for real-time apps (e.g., chat applications, live notifications)
* ●Key Concepts: WebSockets, Real-time communication, Socket.io

#### 10. Deploying Your Node.js Application

* ●Introduction to deployment options for Node.js applications
* ●Deploying Node.js apps on cloud platforms (Heroku, AWS, DigitalOcean, etc.)
* ●Setting up environment variables for different environments (production, development)
* ●Configuring reverse proxies with Nginx or Apache
* ●Key Concepts: Deployment, Cloud services, Reverse proxies, Environment variables

#### 11. API Rate Limiting – Protecting Your Endpoints

* ●What is API rate limiting and why it's important
* ●Implementing rate limiting to prevent abuse of your APIs (e.g., using `express-rate-limit`)
* ●Configuring custom rate limiters for different endpoints
* ●Handling rate limit exceeded errors and responses
* ●Key Concepts: Rate limiting, Throttling, API protection, express-rate-limit

#### 12. Logging & Monitoring – Tracking Application Health

* ●Introduction to logging and monitoring in Node.js applications
* ●Using logging libraries like Winston and Morgan for structured logging
* ●Setting up logging levels (info, warn, error) and storing logs in files or external services
* ●Integrating monitoring tools like PM2 for process management and performance monitoring
* ●Key Concepts: Logging, Winston, Morgan, PM2, Monitoring, Application health

#### 13. GraphQL

* ●Introduction to GraphQL
* ●What is GraphQL and how it differs from REST
* ●GraphQL architecture: schema, queries, mutations, and subscriptions
* ●Setting up a GraphQL server with Apollo Server
* ●Writing simple GraphQL queries and mutations
* ●Writing GraphQL Queries, Mutations, and Subscriptions
* ●Creating complex queries and mutations
* ●Understanding resolvers and schema design
* ●Subscriptions for real-time data updates
* ●Handling errors in GraphQL

#### 14. Monitoring with PM2

* ●Introduction to PM2 for Node.js process management
* ●Setting up PM2 for application monitoring
* ●Auto-restarting Node.js apps on crashes
* ●Monitoring performance with PM2 logs and stats
* ●Log rotations

## React Alchemist 🧙🏻

Objective: Master the art of building dynamic, interactive, and scalable web applications using React.js, Next.js, Tailwind CSS, ShadCN, and other modern technologies. Learn best practices, performance optimizations, and advanced patterns for building professional-grade React applications.

#### 1. Introduction to React – The Modern JavaScript Library

* ●What is React and why it’s the go-to library for building UIs
* ●Understanding the virtual DOM and how React improves performance
* ●Setting up a React project using `create-react-app` or Vite
* ●JSX: A syntax extension for JavaScript that allows writing HTML in JS
* ●Rendering elements and basic React components
* ●Key Concepts: React, JSX, Virtual DOM, Components

#### 2. Components and Props – The Building Blocks of React

* ●Understanding functional and class components
* ●Passing data between components using props
* ●How to use children and default props
* ●Breaking down UI into smaller reusable components
* ●Key Concepts: Components, Props, Reusability, State vs Props

#### 3. State Management – React's Core Mechanism

* ●Understanding state in React and how it drives component re-renders
* ●Managing state within functional components using `useState`
* ●Lifting state up to parent components for sharing data
* ●Conditional rendering based on component state
* ●Key Concepts: State, `useState`, Re-rendering, Lifting state up

#### 4. React Lifecycle Methods – Understanding Component Lifecycles

* ●Introduction to component lifecycle in class components
* ●Exploring React's lifecycle methods (e.g., `componentDidMount`, `componentWillUnmount`)
* ●Using `useEffect` hook for side effects in functional components
* ●How React’s lifecycle methods help manage data fetching, cleanup, and DOM manipulation
* ●Key Concepts: Lifecycle methods, `useEffect`, Mounting, Unmounting, Side effects

#### 5. Event Handling – Interactivity in React

* ●Handling events like clicks, form submissions, and user input
* ●Binding event handlers in React components
* ●Using `event.preventDefault()` and `event.stopPropagation()` for event flow control
* ●Creating controlled and uncontrolled form components
* ●Key Concepts: Event handling, Forms, `event.preventDefault()`, `event.stopPropagation()`

#### 6. React Hooks – Bringing Functionality to Components

* ●Introduction to React Hooks and their importance in functional components
* ●Using `useState` for state management and `useEffect` for side effects
* ●Exploring other hooks: `useContext`, `useReducer`, `useCallback`, `useMemo`
* ●Best practices for working with hooks
* ●Key Concepts: Hooks, `useState`, `useEffect`, `useContext`, `useReducer`, `useCallback`

#### 7. React Router – Navigating Between Pages

* ●Introduction to React Router for client-side routing
* ●Setting up React Router for multiple views (pages) in a single-page application (SPA)
* ●Using `Link` and `Route` to navigate between components
* ●Dynamic routing with URL parameters and query strings
* ●Key Concepts: React Router, `Link`, `Route`, Dynamic routing, SPA

#### 8. State Management with Context API – Global State for Your App

* ●What is the Context API and when to use it for global state management
* ●Creating a context, providing it, and consuming it in components
* ●Using `useContext` to access and update global state
* ●Avoiding prop drilling with the Context API
* ●Key Concepts: Context API, Global state, `useContext`, Prop drilling

#### 9. Forms in React – Building Dynamic Forms

* ●Controlled vs uncontrolled forms in React
* ●Handling form submissions and form validation
* ●Building complex forms with multiple input fields
* ●Using third-party libraries like Formik or React Hook Form for easier form management
* ●Key Concepts: Forms, Controlled inputs, Validation, Formik, React Hook Form

#### 10. Styling in React – From CSS to Styled Components

* ●Styling React components using traditional CSS, CSS Modules, and styled-components
* ●Introduction to CSS-in-JS libraries like Emotion and styled-components
* ●Managing responsive designs in React apps
* ●Best practices for CSS architecture in React (BEM, CSS Modules)
* ●Key Concepts: CSS, CSS-in-JS, styled-components, Responsive design

#### 11. Performance Optimization – Making Your App Fast

* ●Understanding React’s rendering behavior and performance bottlenecks
* ●Techniques to optimize performance in React apps (e.g., memoization, lazy loading)
* ●Using React's `React.memo`, `useMemo`, and `useCallback` hooks
* ●Code splitting and lazy loading with React Suspense
* ●Key Concepts: Performance, Memoization, `React.memo`, `useMemo`, `useCallback`, Code splitting

#### 12. Deploying Your React Application – Going Live

* ●Deployment options for React apps: Netlify, Vercel, Heroku, AWS, etc.
* ●Configuring environment variables for different deployment environments
* ●Building the React app for production using `npm run build`
* ●Setting up continuous deployment for automatic updates
* ●Key Concepts: Deployment, Continuous integration, Production build, Environment variables

#### 13. React Advanced Patterns – Enhancing Your Skills

* ●Introduction to higher-order components (HOCs) and render props
* ●Understanding compound components for reusable logic
* ●Custom hooks: Creating your own hooks for code reuse
* ●Using context providers and consumers for state management
* ●Key Concepts: Higher-order components, Render props, Custom hooks, Compound components

#### 14. Building Scalable React Applications – Architecture and Design

* ●Structuring large-scale React applications using component-based design
* ●Organizing components, hooks, and utilities for maintainability
* ●Breaking the application into features for better scalability
* ●Using state management tools like Redux or Zustand for advanced state management
* ●Key Concepts: Scalability, Component-based design, Architecture, Redux, Zustand

#### 15. NextJS - The React Framework for Full-Stack Applications

* ●Introduction to Next.js and why it’s an essential tool for React developers
* ●Setting up a Next.js project and understanding its file-based routing
* ●Static Site Generation (SSG) and Server-Side Rendering (SSR) in Next.js
* ●API Routes in Next.js for backend functionality within a React app
* ●Dynamic routing and how Next.js handles URL parameters
* ●Key Concepts: Next.js, File-based Routing, SSR, SSG, API Routes

## Deploy and AI Squad 🚀

Objective: Master the deployment of web applications to the cloud, ensuring scalability, security, and high availability. Learn about cloud infrastructure using AWS services like EC2, ECS, CloudFront, Load Balancers, Docker, and more to bring applications from development to production. After that we will move towards AI. What is AI, AI powered applications, Vector Databases (Pinecone etc), RAG based systems, Image background removal, image generation, text summarization and creative ideas with AI

#### 1. Introduction to Cloud Deployment – The Basics of Scaling Apps

* ●What is cloud deployment and why it’s essential for modern applications
* ●Overview of different cloud providers: AWS, Azure, Google Cloud
* ●Understanding high availability, scalability, and fault tolerance in the cloud
* ●Benefits of cloud computing in terms of flexibility, cost, and performance
* ●Key Concepts: Cloud deployment, Scalability, High availability, Fault tolerance

#### 2. AWS EC2 – Virtual Servers in the Cloud

* ●What is Amazon EC2 (Elastic Compute Cloud)?
* ●Setting up an EC2 instance to host your application
* ●Understanding EC2 instance types, regions, and availability zones
* ●Connecting to EC2 instances using SSH and setting up security credentials
* ●Key Concepts: EC2 Instances, Regions, Availability Zones, SSH

#### 3. Configuring EC2 Security Groups – Controlling Access to Your Instance

* ●What are security groups and why they are critical for security?
* ●Setting inbound and outbound rules for EC2 instances
* ●Restricting access to your EC2 instance using security group configurations
* ●Best practices for EC2 security group management
* ●Key Concepts: Security groups, Inbound and outbound rules, Port management

#### 4. Load Balancers – Ensuring High Availability and Reliability

* ●What is a Load Balancer and why it’s important for scalability and availability?
* ●Introduction to Elastic Load Balancing (ELB) in AWS
* ●Configuring an Application Load Balancer (ALB) for HTTP/HTTPS traffic
* ●Setting up health checks to monitor instance health
* ●Key Concepts: Load Balancing, Application Load Balancer (ALB), Health checks

#### 5. AWS CloudFront – Content Delivery Network for Faster Load Times

* ●Introduction to AWS CloudFront and why it’s crucial for performance optimization
* ●Configuring CloudFront distributions to serve static assets (images, scripts, stylesheets)
* ●Understanding caching, edge locations, and how CloudFront speeds up content delivery
* ●Integrating CloudFront with your S3 bucket for static website hosting
* ●Key Concepts: CloudFront, Caching, Edge locations, Content delivery

#### 6. Docker – Containerization for Consistency and Portability

* ●Introduction to Docker and its benefits for application deployment
* ●Creating Docker images and running containers locally
* ●Understanding Dockerfiles and how to write them for your application
* ●Setting up multi-container applications using Docker Compose
* ●Key Concepts: Docker, Containers, Dockerfiles, Docker Compose

#### 7. AWS ECS – Elastic Container Service for Running Docker Containers

* ●What is AWS ECS and why it’s used for deploying Docker containers?
* ●Setting up an ECS cluster to run Docker containers on EC2 instances
* ●Creating ECS tasks and services to manage containerized applications
* ●Configuring ECS with Application Load Balancer (ALB) for traffic distribution
* ●Key Concepts: ECS, Containers, ECS Tasks, ECS Services, ALB

#### 8. AWS ECR – Elastic Container Registry for Storing Docker Images

* ●What is Amazon ECR and how it works with ECS and Docker?
* ●Pushing and pulling Docker images to/from Amazon ECR
* ●Securing your ECR repository with IAM permissions and access control
* ●Best practices for managing container images in ECR
* ●Key Concepts: ECR, Container registry, IAM, Pushing and pulling Docker images

#### 9. Target Groups – Directing Traffic to the Right Containers

* ●What are Target Groups and how do they work with Load Balancers?
* ●Configuring Target Groups in AWS to route traffic to ECS services
* ●Setting up health checks for Target Groups to ensure only healthy containers receive traffic
* ●Understanding the concept of weighted routing and path-based routing in Target Groups
* ●Key Concepts: Target Groups, Routing, Health checks, Weighted routing

#### 10. Security Rules & IAM Roles – Managing Permissions for Security

* ●Introduction to AWS Identity and Access Management (IAM) for managing permissions
* ●Setting up IAM roles and policies to grant permissions to EC2, ECS, and other services
* ●Best practices for securing your AWS infrastructure using IAM
* ●Configuring VPC security groups and network ACLs to restrict access
* ●Key Concepts: IAM, Roles, Policies, Permissions, VPC Security Groups, Network ACLs

#### 11. Scaling and Auto Scaling – Adjusting Resources Based on Demand

* ●Introduction to Auto Scaling and how it helps scale EC2 instances based on demand
* ●Setting up Auto Scaling groups with EC2 instances and configuring scaling policies
* ●Using ECS Auto Scaling to scale Docker containers in response to traffic spikes
* ●Key Concepts: Auto Scaling, Scaling policies, Load balancing, Scaling EC2 and ECS

#### 12. Continuous Deployment (CI/CD) – Automating the Deployment Pipeline

* ●Introduction to CI/CD pipelines and why they’re essential for modern web apps
* ●Setting up CI/CD with GitHub Actions
* ●Automating Docker container builds and deployment to ECS/ECR
* ●Rolling updates and blue/green deployments with ECS for zero downtime
* ●Key Concepts: CI/CD, Code Pipeline, Code Build, GitHub Actions, Docker deployment, Blue/Green deployments

#### 13. Monitoring and Logging – Keeping Your Application Healthy

* ●Introduction to AWS CloudWatch for monitoring EC2, ECS, and other resources
* ●Setting up CloudWatch alarms for resource utilization and application health
* ●Using AWS Cloud Trail for logging and auditing API calls in your AWS account
* ●Integrating logging libraries (e.g., Winston, Morgan) into your Docker containers
* ●Key Concepts: CloudWatch, Monitoring, Cloud Trail, Logging, Winston, Morgan

#### 14. Cost Optimization – Managing Your AWS Resources Efficiently

* ●Best practices for managing AWS costs and avoiding unnecessary expenses
* ●Using AWS Trusted Advisor and Cost Explorer to monitor and optimize costs
* ●Setting up AWS Budgets and alerts to track spending
* ●Key Concepts: Cost optimization, AWS Budgets, Trusted Advisor, Cost Explorer

#### 15. Security Best Practices – Keeping Your Application Secure in the Cloud

* ●Setting up Web Application Firewalls (WAF) to protect against common attacks
* ●Using SSL/TLS certificates for secure communication in your application
* ●Regularly auditing IAM roles and security policies to minimize the risk of unauthorized access
* ●Securing Docker containers and images with scanning tools and security patches
* ●Key Concepts: Security, WAF, SSL/TLS, Docker security, IAM auditing