Things to know before learning K8s

https://youtu.be/bjrlo9LwMJY?si=6fv_o_4xzvchP7KR

This Document has prerequisites you should know before learning Kubernetes.

Containerization (Docker)📦

Why to Learn?

Kubernetes is fundamentally a container orchestration tool, managing the deployment, scaling, and operation of application containers. it's important to understand the basics of containerization. You should be familiar with concepts such as images, containers, and container registries.

There are different software tools that can be used to build containers (container runtimes), but the ones that are supported by Kubernetes are Docker, containerd, CRI-O, and any other implementation of the Kubernetes CRI (Container Runtime Interface).

What to Learn:

1. **Docker Commands:**
    - Master key commands (`docker run`, `docker ps`, `docker stop`, etc.) for container management.
2. **Dockerfiles:**
    - Understand Dockerfile structure.
    - Write Dockerfiles to define custom container images.
3. **Docker Images:**
    - Grasp the concept of Docker images.
    - Learn to build and customize images.

4. **Docker Containers:**
    - Understand container lifecycle and benefits.
    - Manage networks, volumes, and environment variables.
5. **Docker Compose:**
    - Use Docker Compose for multi-container applications.
    - Configure services with `docker-compose.yml`.

Learning Resources:

- Docker Documentation: Official resource for comprehensive Docker knowledge.
- Docker Labs: Practical labs for hands-on experience.
- Docker Curriculum: Comprehensive guide with practical examples.
- https://www.ibm.com/topics/containerization: What is containerization?

Linux Commands 🐧

Kubernetes runs on Linux Operating System for managing container or running "kubectl" commands. Having Linux basics knowledge can help you create secure and performing applications on Kubernetes.

What to Learn in Linux:

1. **Basic Commands:**
    - Master essential commands for file management and navigation.
2. **File Permissions:**
    - Understand how to set and modify file permissions.
3. **Process Management:**
    - Learn commands for monitoring and managing processes.
4. **Networking Basics:**
    - Familiarize yourself with commands for network troubleshooting.
5. **Text Editors (e.g., Vim, Nano):**
    - Acquire proficiency in terminal-based text editors for quick edits.

6. **System Information:**
   - Explore commands for gathering insights into system resources.

Learning Resource:

https://youtu.be/lCq4mYQL0WY : Important Linux commands for DevOps Engineers

https://kubernetes.io/docs/reference/kubectl/cheatsheet/ Kubectl Cheatsheet.

Networking

In Kubernetes, understanding networking is super important. Without knowing about things like ports, load balancers, and firewalls, you can't successfully set up a Kubernetes application.

Crucial for apps that need to exchange data between backend applications or be accessible to users as a frontend web app or website. Networking is a big deal in Kubernetes, to understand how Deployments, Pods, and Services communicate with each other.

What to learn

- Gain foundational knowledge in networking.
- Understand IP addressing concepts, including IPv4 and IPv6.
- Familiarize yourself with routing principles and how data is directed between networks.
- Learn subnetting to efficiently allocate IP address ranges within a network.
- Grasp basic network protocols, such as TCP/IP and UDP, to comprehend data communication principles.
- Recognize the integral role of networking in facilitating communication within Kubernetes clusters.

Networking Fundamentals for DevOps Engineers | DevOps Networking

Networking fundamentals for DevOps Engineers | Networking for DevOps Networking Knowledge for DevOps Engineers - In this Devops tutorial, we will learn different networking fundamentals or different networking basic concepts

https://youtu.be/M9Kex1ID7GY?si=LwQmpBfuJKmNL380

Understanding Microservices

Microservices is architecture pattern followed by Kubernetes, where each service is deployed as a container. So having understanding of Microservices work is really important.

For instance, if an E-commerce application follows the microservice architecture, it will comprise the database service as a container, the frontend service in another container, and the backend service in another container, all working together to make the E-commerce application work flawlessly.

What to learn?

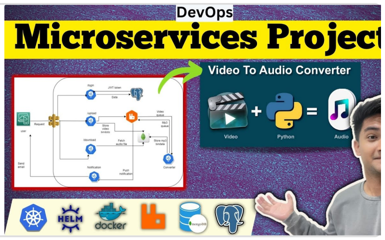
- **Microservices Basics:**
  - Understand Microservices as an architectural pattern for breaking down applications into small, independent services.
- **Service Communication:**
  - Learn how Microservices communicate, starting with an introduction to RESTful APIs.
- **Independence and Autonomy:**
  - Grasp the concept of each Microservice operating independently with its own database and functionality.
- **APIs:**
  - Gain a basic understanding of how Microservices expose APIs
- **Hands-On Projects:**
  - Apply knowledge through hands-on projects or small applications, reinforcing Microservices principles through practical experience.

Learning resource:

DevOps Project: Video to Audio Python Microservices App on Kubernetes

Kubernetes Project for Practice | DevOps project for resume | AWS Project | Python Microservices video to audio converter app | Kubernetes Microservices Deployment

https://youtu.be/jHlRqQzqB_Y?si=Q8iwx6XMeED9ba1O



Cloud Provider ☁️

Kubernetes is commonly deployed in the cloud.

Deploying Kubernetes in the cloud offers various advantages, Firstly, cloud platforms such as AWS, Azure, or GCP provide the necessary infrastructure for hosting Kubernetes clusters, including virtual machines, storage, and networking services.

Furthermore, cloud platforms offer features like auto-scaling, load balancing, and monitoring tools that align with Kubernetes' requirements for scalability and resilience.

What to learn?

- Cloud Fundamentals: On-demand provisioning, scalability, pay-as-you-go.
- Major Cloud Platforms: AWS, Azure, GCP. (Managed Kubernetes Cluster setup)
- Deployment Models: Public, private, hybrid clouds.
- Service Models: IaaS, PaaS, SaaS.
- Basic Networking: Virtual networks, subnets, security groups.
- IAM (Identity and Access Management): User roles, permissions.
- Storage Services: Object, block, file storage.
- Security Best Practices: Encryption, access controls, compliance.
- Cost Management: Monitoring, budgeting, optimization.
- Hands-On Practice: Experiment on a cloud platform for practical experience.

YAML

YAML - Yet Another Markup Language or “YAML Ain’t Markup Language”

YAML is to create manifest files responsible to create kubernetes object in declarative way. So having understanding of how to write in YAML is important!

What to learn?

- Basic syntax, key-value pairs, lists.
- Data types: strings, numbers, booleans.
- Handling lists, arrays, dictionaries, and nested structures.
- Practical use in configuration files (e.g., Kubernetes).
- Validation tools, best practices, error handling.

Learning resource:

IBM Developer

IBM Developer is your one-stop location for getting hands-on training and learning in-demand skills on relevant technologies such as generative AI, data science, AI, and open source.

https://developer.ibm.com/tutorials/yaml-basics-and-usage-in-kubernetes/

12 Factor App

The Twelve-Factor App

A methodology for building modern, scalable, maintainable software-as-a-service apps.

https://12factor.net/