**1. Docker: Basic to Advanced (Week 1)**

**Learning Goals:**

* Understand containerization concepts
* Learn Docker commands and best practices for building, deploying, and managing containers
* Advance knowledge with multi-stage builds, volumes, networks, and performance tuning

**Plan:**

**1.1 Docker Basics**

* **Introduction to Docker and Containers:**
  + What is Docker?
  + Difference between containers and VMs.
  + Benefits of using containers.
* **Hands-On:**
  + Installing Docker on your machine (Windows/Mac/Linux).
  + Exploring Docker CLI commands (docker run, docker ps, docker stop, docker rm).
  + Understanding container lifecycle.
* **Exercises:**
  + Run a simple Nginx container.
  + Create and run your own custom image using a simple Python or Node.js application.

**1.2 Dockerfile and Docker Compose**

* **Introduction to Dockerfile:**
  + Syntax and structure of a Dockerfile.
  + Building images using Dockerfile.
  + Multi-stage builds, caching, and optimization.
* **Hands-On:**
  + Create a Dockerfile for a Python web application.
  + Practice multi-stage builds for reducing image size.
* **Introduction to Docker Compose:**
  + What is Docker Compose and its use case.
  + Structure of docker-compose.yml.
  + Setting up multi-container environments.
* **Hands-On:**
  + Use Docker Compose to run a web app and database (e.g., Python app + MySQL).

**1.3 Docker in Advanced**

* **Advanced Docker Concepts:**
  + Networking in Docker (bridge, host, overlay networks).
  + Managing volumes and persistent data.
  + Multi-Stage to optimize dockerfile.
* **Performance Tuning and Optimization:**
  + Optimizing Docker images for production.
  + Limiting CPU, memory, and other resource management options.
* **Hands-On:**
  + Set up a complex Docker environment with a front-end, back-end, and database using Docker Compose.
  + Implement custom networking between services.
  + Configure resource limits for containers.

**1.4 Presentation: Docker**

* Presentation on Docker, covering basic to advanced topics.
* Include a demo showing the deployment of a web app using Docker Compose.
* Highlight real-world use cases (e.g., microservices architecture).

**2. Triton Server for Model Serving (Week 2)**

**Learning Goals:**

* Deploy machine learning models at scale with Triton Inference Server
* Understand performance optimization techniques
* Master model ensemble and GPU utilization

**Plan:**

**2.1 Introduction to Model Serving**

* **Overview of Model Serving:**
  + What is model serving?
  + Challenges in deploying models at scale.
* **Hands-On:**
  + Simple local model serving using Flask or FastAPI to understand the concept.

**2.2 Triton Server: Model Deployment**

* **Introduction to Triton Inference Server:**
  + What is Triton Server and why is it useful?
  + Supported model formats (TensorFlow, PyTorch, ONNX, etc.).
  + Key features (concurrent execution, multi-framework serving).
* **Hands-On:**
  + Install and set up Triton Inference Server locally or on a cloud platform.
  + Deploy a simple model (e.g., image classification) using Triton.

**2.3 Triton: Batching**

* **Understand how batching works in Triton Server:**
  + Understand how batching works in Triton Server.
  + Learn different batching mechanisms (static and dynamic).
  + Implement and optimize batching to improve model throughput and resource utilization.
* **Hands-On:**
  + Set up a simple Triton Server with a model (e.g., an image classification model) and configure basic static batching to group multiple requests together.
  + Explore model optimization and batching strategies.

**2.4 Triton Server: Inference Acceleration and Model Ensemble**

* **Model Ensemble:**
  + What is model ensemble and why is it useful?
  + Using Triton to chain models together.
* **Inference Acceleration:**
  + Techniques for faster inference (CUDA, TensorRT).
* **Hands-On:**
  + Set up model ensemble with Triton to combine multiple models for an inference pipeline.
  + Experiment with GPU inference acceleration using TensorRT.

**2.5 Presentation + Demo: Triton**

* Presentation on Triton Server, focusing on real-world use cases.
* Demo a complete pipeline with multiple models deployed on Triton with GPU acceleration and resource optimization.
* Using Locust for Testing Throughput and Latency

**3. BentoML (Week 3)**

**Learning Goals:**

* Deploy machine learning models using BentoML
* Explore advanced features like GPU inference and adaptive batching
* Understand worker scaling and deployment strategies

**Plan:**

**3.1 Introduction to BentoML**

* **What is BentoML?**
  + Overview of BentoML for serving machine learning models.
  + Comparing BentoML with other serving solutions (e.g., Flask, Triton).

**3.2 BentoML Tutorial: Model Deployment and Worker**

* **Model Deployment:**
  + Deploy a model with BentoML using simple APIs.
  + Understand the model store and versioning.
* **Worker Scaling:**
  + Explore how BentoML manages workers and scale horizontally.
* **Hands-On:**
  + Build and deploy a machine learning model using BentoML on a local machine.
  + Experiment with different worker configurations and observe performance.

**3.3 BentoML GPU Inference**

* **GPU Inference with BentoML:**
  + Enabling GPU support for models.
  + How BentoML integrates with NVIDIA GPUs for faster inference.
* **Hands-On:**
  + Deploy a GPU-enabled model (e.g., a deep learning model) using BentoML.
  + Observe performance improvements with GPU inference.

**3.4 BentoML Tutorial: Model Composition and Adaptive Batching**

* **Model Composition and Adaptive Batching:**
  + Serving multiple models together with BentoML (composition).
  + Explore the adaptive batching feature for improving throughput.
* **Hands-On:**
  + Deploy multiple models in BentoML with composition and batching.
  + Test adaptive batching by sending concurrent requests.

**3.5 Presentation + Demo: BentoML**

* Prepare a 15-minute presentation covering BentoML’s key features and how it compares with other frameworks.
* Demo deploying and scaling a model, showcasing worker scaling and GPU inference.

**4. Kubernetes: (Week 4)**

**Learning Goals:**

* Understand Kubernetes concepts like pods, services, deployments, and scaling
* Learn to set up an EKS cluster and deploy applications
* Explore scaling strategies for applications

**Plan:**

**4.1 Kubernetes Basics**

* **Introduction to Kubernetes:**
  + What is Kubernetes and its architecture?
  + Key components (pods, nodes, services, deployments, etc.).
* **Hands-On:**
  + Install and set up Minikube or kind to run Kubernetes locally.
  + Deploy a simple application (e.g., Nginx) to the local Kubernetes cluster.

**4.2 Creating an EKS Cluster**

* **Amazon EKS:**
  + What is Amazon EKS and how it simplifies Kubernetes management?
  + Understanding AWS resources (VPC, EC2, IAM) needed for EKS.
* **Hands-On:**
  + Create an EKS cluster using the AWS Console or CLI.
  + Set up kubectl to interact with the EKS cluster.

**4.3 Deploy and Scale an App on EKS**

* **Deploying Applications:**
  + Creating Kubernetes manifests for your application (deployment, service, ingress).
  + Exposing your application with a LoadBalancer or Ingress.
* **Scaling and Monitoring:**
  + Horizontal Pod Autoscaler (HPA) to scale based on CPU/memory usage.
  + Monitoring resources with Prometheus.
* **Hands-On:**
  + Deploy a containerized web application to EKS.
  + Set up auto-scaling with HPA based on resource utilization.