01)

1. Export the model.
2. Load the model, Load weights, Preprocess the input -Run the inference script-Deliver the output
3. Build the REST API with fast api
4. Dockerize the app.
5. Deploy it in AWS ECS

02)

Model versioning is handled by **MLflow.** This is a open source application. It has four components Tracking, Projects, Models, and Model Registry. For model versioning and monitoring **Model Registry** is used. For this after training model is logged using MLflow library. Then the model is registered using register_model function. Then several stages are used to control the lifecycle. All the new models are first deployed to the Staging state and after test/shadow deployments if the metrics are good they are promoted to production. Specialty is **MLFlow handles version automatically** the API logic remains the same. The MLflow logs consist of training metrics such as accuracy,loss,F1 also model parameters and drift reports. For **drift detection Evidently AI** is used which will periodically compare live data with training data.

03)

Converting to torchscript will make the model faster. Also using GPU will reduce the latency.

04)

To monitor data drift **Evidently AI** can be used exclusively. This detects changes in statistical distributions of input features compared to the training dataset. These shifts include numerical and **categorical feature distributions, detecting outliers and detecting invalid data**. Also, when the ground truth is visible the prediction accuracy can also be monitored.

05)

For the entire lifecycle

1. Train and Validate-Train the PyTorch model, log it with MLflow, and validate metrics.
2. Quality checks-Run data and prediction drift checks using Evidently AI
3. Register and Promoting (Using MLFlow)- Register the model in MLflow Model Registry, promote to Staging → Production after validation.

4. Deploy-Serve the model via MLflow in Docker
5. Monitor-Continuously track model performance, latency, and data quality.