Deployment Strategy

Tools to be used

1. MLflow
2. Amazon Elastic Container Registry (ECR)
3. Amazon Sagemaker
4. Lambda Functions and AWS API Gateway

MLFlow

It is a one stop solution to manage code, artifacts, logging parameters tracking, helps to track project versions (v1.x, v2.x), helps in packaging ML models in multiple flavors such as TensorFlow, Python flavor and can be deployed to a Docker-based REST server by packaging it into a standard image and this image can then easily be deployed to any cloud services. (We will be using this service majorly). Also, we can use MLflow for model versioning, stage transitioning (from staging to prod), and annotations.

What we will be doing is creating a custom Python model, this will contain our inference or API logic, this complete pipeline of MLflow will be packaged into a custom Docker image built by MLflow. Why going all the way with this? Sagemaker behaves in a weird way where it requires the model to be fit to deploy and as such this only works when we create docker image via the MLflow path.

Code Snippets for building this inference pipeline here.

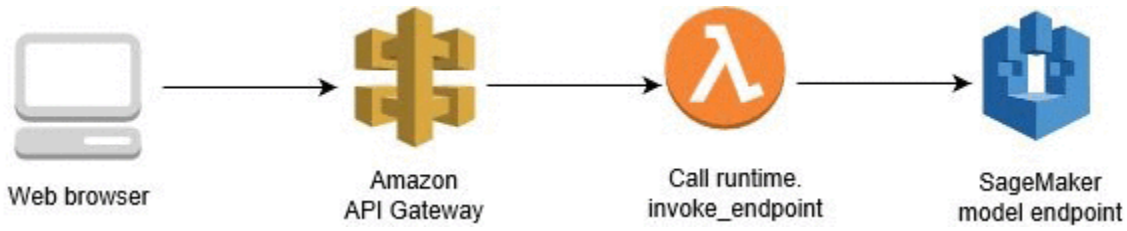
Amazon Elastic Container Registry (ECR)

It is a container registry and majorly being used to push docker image built using MLflow and then using this image to be called by the AWS Sagemaker. It behaves like a github but for Docker images and allows us to use our docker images for any AWS service.

Code snippet to automate the MLflow build, create docker image and push it to ECR is here.

SageMaker

SageMaker is one stop solution for everything MLOps. It can be used for training models, doing batch jobs as well as live inferencing. It allows us to achieve autoscaling whenever we have high loads of requests coming in.



We can host the docker image on Sagemaker either using the CLI or using the SageMaker SDK. Provide the required instance. We need to make sure that the model has been recognized and hosted properly.

Amazon SageMaker capability	Free Tier usage per month for the first 2 months
Amazon SageMaker Studio notebooks , On-demand notebook instances	250 hours of ml.t3.medium instance on Studio notebooks **OR** 250 hours of ml.t2 medium instance or ml.t3.medium instance on on-demand notebook instances
Amazon SageMaker Data Wrangler	25 hours of ml.m5.4xlarge instance
Amazon SageMaker Feature Store	10M write units, 10M read units, 25 GB storage
Training	50 hours of m4.xlarge or m5.xlarge instances
Inference	125 hours of m4.xlarge or m5.xlarge instances

Creation of API with Lambda and API Gateway

We then need to create a Lambda function that will invoke the sagemaker endpoint for us. Create the function and give the requisite IAM permissions. While editing the policy, make sure to add the JSON tag of "*sagemaker:InvokeEndpoint*" . To write the Lambda function, follow this template here.

For creating an API Gateway, follow the rest of the process here. API can then be tested using Postman and logs can be accessed using Cloudwatch.

The Sagemaker model can autoscale as and when the load increases. We can also define what scaling policy we need to follow according to our requirements. More info here.