# Deployment Considerations

- How would I deploy the model in a real-world scenario?

  I am quite familiar with the AWS cloud environment; I will be describing the process through an AWS lens.

  Since we would need to store data, train, load and deploy the custom Sentiment Analysis model for our customers and create a pipeline to store the incoming data and to train our model again to check for any case of data drifts or covariate shifts, here's the process I would go through in the AWS environment:

  - AWS CodeCommit for version controlling of the code.
  - AWS CodeBuild for designing, training and testing the model.
  - AWS S3 to store and manage the model weights and configuration files.
  - AWS CodePipeline which automates sequences above mentioned, including deployment step and re-training of model to fin-tune over data drift.
  - ECS/EKS, AWS Sagemaker for the Model Training, Fine-Tuning and Deployment.
  - AWS Lambda and Amazon CloudWatch for monitoring and automated testing.

- How would I address and/or handle potential challenges?
  - **Scalability, Latency and Cost:**
    - Let's suppose we train this model on one EC2 or **EKS instance** (Elastic Kubernetes Service), EC2 instance should suffice, we can then use a new EKS instance solely for Deployment.
    - Let's suppose we use g5.12xlarge instance as single node, multi pod, in the case of **g5.12xlarge it can be 4 pods**, (i.e.) 1 to 4.
    - When the pod is initialized, it will allocate less than 500 MB of space for our specific model, which gives us the rest of 3800 GB to work with and we can deploy it with an asynchronous framework, which enables us to increase our throughput for request without blockings.
    - EKS has something called **Horizontal Pod Autoscaler** (HPA) which enables us to scale the number of pods (and/or instances) in case of parallel processing due to increase of demand; it also downscales when the demand is low.
    - Each pod can handle **8-10 concurrent requests per second** (considering the **latency to be 100ms**) if optimized to run asynchronously, this translates to **480-600 requests per minute**.

If the instance is initiated with 4 pods, the requests are 4 folds per minute.

- The monthly cost of **g5.12xlarge** which handles 4*(480-600) requests per minute is **$3,280 US Dollars**. If we consider 480-600 requests per minute to be sufficient, we can use a smaller instance size like **g5.4xlarge** and the price is roughly **$820 US Dollars per month**. We can use HPA to increase or decrease the number of instances depending on demand.
- The above cost with respect to per million tokens if each token is of 3.75-character length and we have 10,000 tokens/second; for g5.12xlarge and g5.4xlarge, the **Cost per million tokens** is approx. **$0.1264** US Dollars.

- **What processes would I use to handle the updates?**
  - I would use **model registry** and integrate it with the CI/CD pipeline, in our case it will be the AWS CodePipeline. This will help us version control and roll out an updated one seamlessly, it also helps us monitor the performance and health.
  - We can also have **rolling updates** in EKS, in case we are using multiple pod instances, as we might encounter data drifts, for example a certain product like a beauty cream has either gone from a positive sentiment to negative or vice-versa. Kubernetes helps us roll out the updated model onto one pod and monitor the performance and eventually onto all other pods.