Proposal For Production

Aditya Singh

Introduction

Deploying to production in an efficient manner ensures the scalability and availability of the approach. Automated deployment environment provides consistency to the deployment and spawning machines as per requirement curbs the cost to the company.
Currently the implemented solution runs as a docker container. The client makes a **JSON GET** request
We start with the changes required to the application followed by other modules required for cloud deployment.

1 Algorithm Modifications

Currently the container requires the trainign data to be available locally inorder to ensure the model can be created in case the pre-trained model is not accessible. This can be changed by using a database which can be queried for data incase the pre-trained model is not available.
This can be further improved by sharing pre-trained models across instances.

2 Cloud Deployment

2.1 Cloud Services

Fig 1 shows the options available for choosing a cloud deployment strategy. The cost to company increases from **On Premises** to **SaaS** whereas, the complexity and effort to manage different resources decrease from left to right. In this proposal I have chosen **PaaS**(platform as a Service) to only focus on the Application and Data thereby keeping the pricing reasonable with very less effort in maintaining the application online.
Provider: AWS: Fargate is recommended by comparing the ease of use

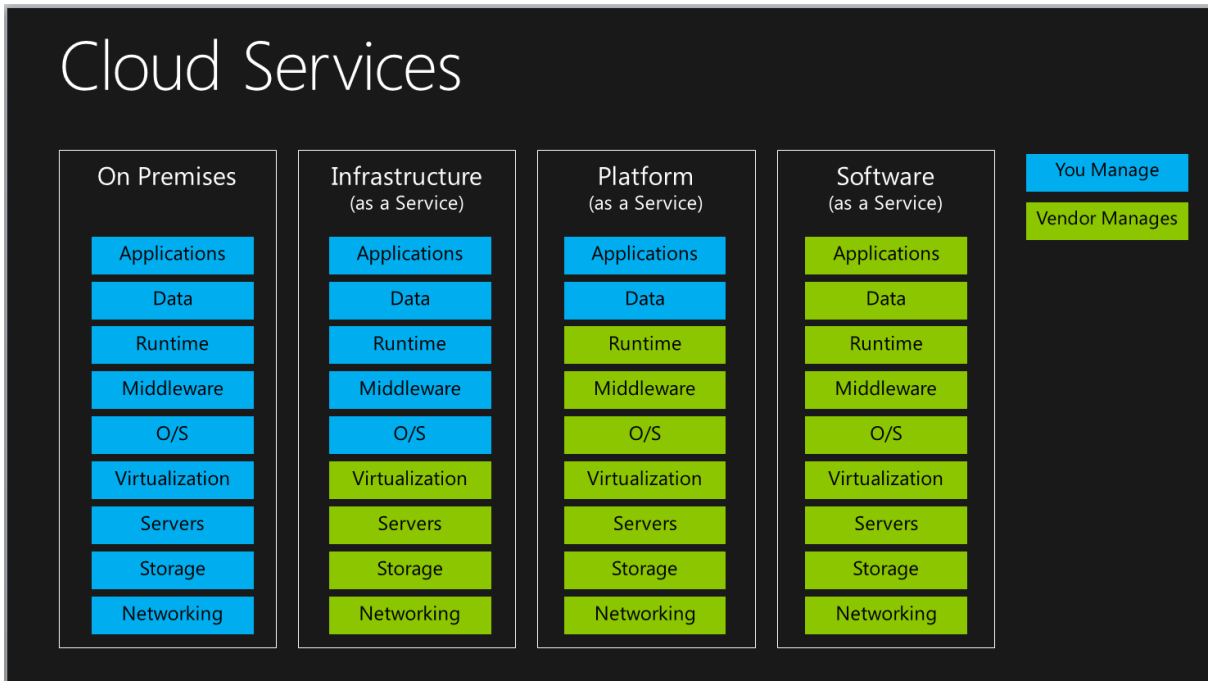

Figure 1: Cloud service categorization. (source)

with Dockers, automatic scaling and pricing among different providers such as Heroku, Google App Engine, and DigitalOcean. For a more detailed analysis one can read the blog post

2.2 Version Control

GitLab is one viable option for version control. The free tier contains the option for creating private repositories as well as 2000 minutes of Continuous Integration per month.

2.3 Continuous Integration / Continuous Deployment

AWS and Gitlab merge seamlessly for CI and CD purposes. GitLab CI provides docker engine to facilitate the testing and deployment of an application.

2.4 Configuration Management

Cloud configuration management is required to keep the implementation segregated from the constants which might change for example, the text categories(Cancel, Status, Others).
The configuration management can be implemented in several ways like

reading from files, databases or updating via an API call all of which can be further cached to improve performance.

2.5 Log Management / Performance Visualization

Post-deployment it is important to have a constant monitoring of the system in place. With proper logging it is easier to catch and fix bugs in the code as well as monitor the efficiency of the deployed application.
ELK-stack serves as a multipurpose tool to meet the above stated requirements. The price with a low-end requirement on AWS can be brought down to 22$ a month.