Proposal For Cloud Deployment

Aditya Singh

Introduction

Deploying to the cloud 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 script. The JSONs are provided before the Docker image is build and run. To have a good and feasible solution for cloud deployment there are changes that would be required to the existing pipeline of the application as well as addition of modules to automate the deployment process.
We start with the changes required to the application followed by other modules required for cloud deployment.

1 Algorithm Modifications

As stated above, the current implementation builds and runs the docker image and requires the JSONs to be available while executing the docker container.
First obvious change would be to decouple input from the compilation. We can expose RESTful APIs such as *GET, POST* to the running container for sending JSON requests and the algorithm can then return a **True or False** response.
Alternatively, to reduce the overhead of reading the stocks, converting to a JSON and then providing to the algorithm we can point the algorithm directly to the database containing the stock information from where we can filter the recipes on requirements such as area, type of cuisine etc.

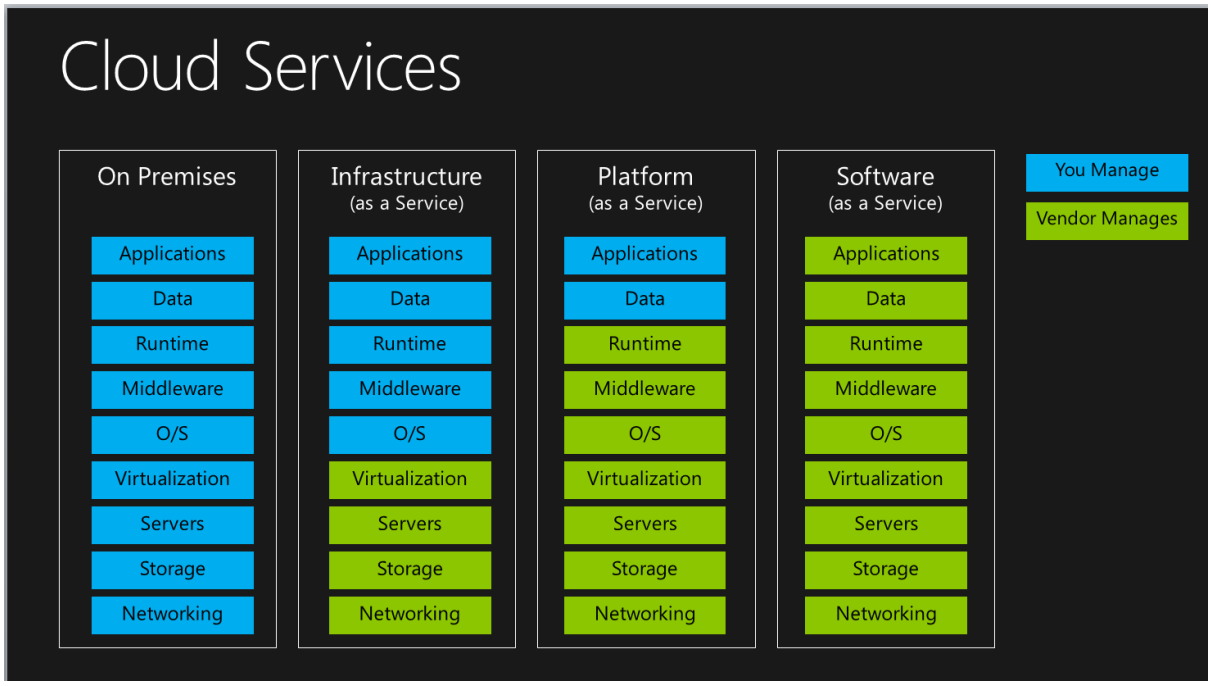

Figure 1: Cloud service categorization. (source)

2 Cloud Deployment Modules

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 we choose **PaaS**(platform as a Service) as we wish 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 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 number of recipes or portion size, which currently is hard-coded in the implementation can be controlled from outside the code. This will reduce the effort to re-deploy after a new field for recipe or portion size is added to the code. 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.