Mitchell Van Braeckel (1002297)

CIS*4010: Cloud Computing

Assignment 2

(Extended) Nov 6th, 2020

Part 2 – Alternative Existing AWS Services to EC2

Question:

"For the tasks that you were asked to do in Part 1, suggest what existing AWS services could be used to do the similar tasks and compare the capabilities and easy of use for your programs and the AWS services."

"The answer to the question posed in Part 2 should appear in a file in your A2 repo and should be named Part2.txt or Part2.pdf depending on its format. This will be graded outside of your Zoom grading session. It should not be any larger than 500 words."

Answer: (See next page – word count = 478)

Using a deployment provider like Amazon ECS (Elastic Container Service), including AWS Fargate, in combination with AWS CodePipeline and integrating it with AWS CodeBuild to automate workflows and AWS Service Role in IAM to access these.

Besides the obvious benefits of security, reliability, and availability, using these services makes supporting container-based applications and services much easier, including a CD pipeline that can be used for automation. For instance, this may consist of the following steps: choose a source (eg. GitHub repository), build project (via CodeBuild), add deployment provider (such as: AWS CodeDeploy, AWS Elastic Beanstalk, AWS CloudFormation, or Amazon ECS) and deploy code, then finally set an AWS Service Role in IAM (to give CodePipeline permissions). Note that for CodeBuild to deploy ECS, an image definition JSON is required to add instructions for pre-build, build, and post-build phases.

To summarize, EC2 allows launching of individual instances (for whatever purpose), whereas ECS launches instances that will be ready to launch container applications. In other words, installing Docker isn't required because it's ready when compared to an EC2 instance initially. The main distinction is that each instance is managed separately via EC2, where someone deploys applications and maintains connections themselves. In contrast, ECS allows one to launch a cluster of machines to serve as deployment ground of container apps, in turn allowing them to treat all instances in the cluster as one big instance available for container workload. Particularly, an ECS cluster with no instances can be started, but nothing can be run on it, so once an EC2 instance is registered inside, containers are ready to run in it.

Building and deploying instances and containers manually is slow and prone to errors. For example, I wrote the program and I highly doubt that it is better / more bulletproof than existing services or programs written by professionals (like the AWS Services). Using these as an alternative is more secure because it has been extensively tested and is therefore more reliable. Furthermore, CodePipeline offers much more flexibility while also minimizing complexity for developers with regards to management of and operations on instances and containers. In comparison to the programs in A2, it clearly accommodates many more services and an even greater variation of setups, especially because it supports integration with other AWS services my programs do not (i.e. the AWS services I listed above excluding AWS EC2).

In conclusion, using a combination of alternative AWS services, such as CodePipeline, that possesses the ability to integrate multiple AWS services is a better choice instead of only using EC2 because it offers increased security, reliability, and the ability to automate a CD pipeline for the project. Also, by using a deployment provider like ECS and putting EC2 instances inside, it removes our necessary step to install Docker on each instance, as well as the need to deploy application and maintain connections ourselves.