8/21/2022

Nicole Penner

nicole.penner@snhu.edu

8-1 Assignment: Final Reflection

<https://www.youtube.com/watch?v=GV8FfeS12YY>

**Experiences and Strengths**

Throughout this course I have learned about how to create a full stack application using AWS services. In turn, that means I have learned about DevOps processes. I have learned how to successfully use containers and orchestration to hold and run applications. I feel comfortable with the idea of using both Docker and Docker Compose to run containers together in a singular application. I was able to gain extensive knowledge about using various AWS services to contain a cloud application. I feel comfortable using Lambda Functions, API Gateways, S3 buckets, DynamoDB database, as well as IAM policies. I feel that all of these skills I have gained, make me marketable to work on the frontend of an application, the backend of an application, or in full stack development. I could realistically now work on any part of an application even though I am still a novice. This class will be extremely useful in the future, and I am excited to have learned so much about cloud development as I believe it is useful in the development of many types of applications. Other skills and strengths I have gained from this course is time management, problem solving skills, working with a team (my classmates) to solve issues that arise, and the knowledge to use AWS services for cloud development.

**Planning for Growth**

Micro-services are useful to for software developers. It allows for software to be broken apart in separate pieces or parts. Different micro-services can be used to handle tasks and brought together to run a full application. Serverless development is ideal for some applications mainly because you do not have to pay to maintain physical disk space or servers. This can be extremely time consuming and expensive to do. Instead, serverless architecture allows for developers to only pay to use the server space they actually need. The third-party that the developers choose to use are then responsible for maintenance and upkeep of the “server”. It serves as a more efficient and cost-effective option for application development

There are many benefits when using serverless designs. Scaling is typically handled by the provider so it isn’t necessary for developers to spend time working on it. . Developers are only concerned if there is error when writing the application to use the “server”, while any other issues are taken care of by the third-party service managing it. This means the developers have less control to handle errors if it is in the realm of the service provider and not their code. Cost of using a server-less design is extremely predictable. This is because you are only charged for the space you use. I would say that when it comes to predictability, server-less is more predictable based on the pay-for-use model. With containerization, because you are handling it yourself – there is less predictability of cost you’ll have to run it.

In deciding what solution to use there is many different factors that should be considered. One consideration is control of certain factors when comparing server-less to containerization on a traditional server. With server-less architecture, scaling of the application is handled for you which is extremely efficient. The con of this is that you do not have control over your server, the third-party host does. Containerization gives you that control but it requires more management and money from the development team. Serverless models follow the model of elasticity which is another pro. If you need more storage, the application will expand. If you need less storage it will contract, and you are left paying for only the storage space you use automatically. This can be a huge plus when considering going with serverless architecture. Pay-per-use wastes less resources and ensures that there is less effort being put into managing and maintaining a physical server. If you need to expand storage with traditional architecture, you have to have more physical space to do so. If for some reason you need less space, you are still paying to run and manage more physical storage – it is not elastic. Serverless is also a more environmentally friendly option if that is important to your company.

Overall, when choosing the solution that best fits your needs many factors should be considered to come to the best option. Containers allow for the development team to have control over the server, but tend to be less efficient and take more effort to upkeep. Serverless architecture are elastic, pay-per-use, are more efficient, and take less effort to upkeep. The con when it comes to the serverless design is that you give up a certain level of control. Overall, each project should be considered on its own as there is no “one size fits all” option.