Paul Yaginuma
MSDS434 Winter 2025

Reflection Paper

Being relatively new to most of these topics, this class did a good job introducing a lot of tools. That being said, at times is was a firehose. There were some weeks where the implementation and sample code was so complicated (largely because I was new to docker and GO) that it was about all I could do to mimic the lecture step-for-step and paste in the code. Especially weeks 3, 7, and 8 (elastic beanstalk, containerized micro service, and pub/sub). Some weeks I got a bit too caught up in trying top get the demo to work that I lost the bigger picture of how the tool might fit into a larger application or what the most common use cases might be. If anything was going to be replaced, I would replace the Kafka demo with either a module on hosted functions (e.g. lambda), spend time week giving a conceptual overview of the tools with examples or case studies of applications that use them, or cover roles/permissions/VPCs a bit more in depth at the beginning.

While I haven't seen the final layout for anyone's project, it seems like we largely landed in similar places where a database feeds a model, which serves predictions. It's been interesting to see how people swear by different tools when it comes to things like databases or even languages in general. It's also been interesting to see how versatile ec2 instances are and the various ways folks have used them. Overall, it looks like AWS is preferred more than GCP, which reflects actual industry.

One common hurdle that everyone seemed to face—including me—was issues with permissions and roles. I sort of blindly stumbled through figuring out what permissions to give various roles to get the demos to work, but I didn't have a good understanding of them until somewhat recently. I had initially thought roles were assigned by title—e.g. engineers have certain roles and sales have others—which can be the case, but I didn't realize roles can be both reused and assigned to tools. For the first several modules I hadn't realized that each instance of a tool had an associated role and simply was using the default role that was assigned at creation. Later in the class I figured out where to assign roles to specific tools, both in the GUI and programmatically, which made more sense and made it much easier to do things like having a lambda function spin up a sagemaker instance to train a model.

A few notable things I implemented for my project based on this class and feedback were a flask server on an ec2 instance, database integration, and environment variables for things like paths and db configs. I had previously hosted a web app using google's firebase hosting, but had never hosted a server on an ec2 instance, which offers more flexibility. For the database integration, I was only moderately successful—using it to keep track of the events list, but not the actual stats for each event—but I had never interacted with a database programmatically before so it's a useful skill to build on in the future. Finally, I used environment variables for various things like database configs, API keys, etc. just to follow best practices.