2021 AWS Senior Architect Interview Scenario Material

1. Hosted a static website on an S3 bucket in the us-east-1 region. Made content available from a CloudFront origin pointing to the us-east-1 bucket. Performed cross-region replication to another S3 bucket in the eu-west-1 region for European users.   
     
   Needed to give greater avaIlability for all users of the website:
2. Created an additional CloudFront origin pointing to the eu-west-1 region
3. Set up a CloudFront origin group with the us-east-1 bucket as the primary and the eu-west-1 region as the secondary

Could have used Route 53 failover routing, but the CloudFront origin group is much more efficient.

1. Ran a web application using multiple Linux EC2 instances that stored data on EBS volumes.   
     
   We were looking for a solution to increase the resiliency of the application in case of a failure and to provide storage that complies with atomicity, consistency, isolation, and durability (ACID).   
     
   1. Created an Application Load Balancer using Auto Scaling Groups across multiple availability zones  
   2. Stored the application data on EFS and mounted a target to each EC2 instance  
     
   Could also have used DynamoDB to get ACID
2. Needed to build an API that automated inquiries for ETF total and estimated cash. The total and estimated cash amounts are the amount of excess cash in the fund; these numbers are used to calculate how much balancing cash will be required when doing a creation or redemption. We experience a larger number of inquiries during the market open and near market close that cause slower response times.   
     
   I needed to design a solution that was scalable and elastic.   
     
   1. Designed a REST API using API Gateway that accepted the total and estimated cash requests  
   2. Had the API Gateway call a Lambda function that calculates the total and estimated cash requests  
     
   Used [Lambda proxy integration](https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-lambda-proxy-integrations.html), which is a lightweight, flexible API Gateway API integration type that allows you to integrate an API method – or an entire API – with a Lambda function. I used the ANY method.  
   Refer to [Tutorial: Build a Hello World REST API with Lambda proxy integration](https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-create-api-as-simple-proxy-for-lambda.html)
3. I didn’t make the right decision/judgement.  
   I was working on a large development program where I was responsible for implementing the order management system for our firm’s global trading platform. I had to change our order management process from three regional trading books (US, EU, Australia) to a single global platform. I visited our Australian office (our smallest office) for three weeks to understand their operating environment from a directly personal level. They convinced me that they needed several features prioritized in our release schedule. I returned to the US and pushed my team to prioritize the AUS features. I successfully got their features prioritized, however we ended up spending important time implementing features for our smallest office while our larger, more productive offices (US and EU) waited for their required features.  
     
   I realized that I put our program behind schedule. I then brought together all of my peer internal technical teams responsible for completing the tasks needed to put our program back on schedule. I created an expedited plan and led the teams through our plan. We delivered our global trading platform on schedule. Our lead portfolio manager performed data analytics on our new trading platform and projected that the program would pay for itself in less than a year, which it did.
4. Bias for action:   
   Our organization imposes a 7 day lock down on certain kinds of production changes. Specifically, job scheduling changes that are orchestrated by a product called Control-M. However, other types of changes can use our “CI/CD” pipeline. We would have job failures on the weekends or at night and we would have to escalate and get many people on a conference call for hours to get our jobs to be rerun.   
     
   I decided that we should build a simple Lambda job scheduler. I designed a Lambda function that when a file is dropped into an S3 bucket, the Lambda function uses the name of the file to determine which job to restart. I designed the process to use Storage Gateway to drop the file into the S3 bucket.  
     
   We can now restart our jobs at will. We have saved hundreds of hours of developer, manager, and operations employee time since implementing this change.
5. Customer focus:  
   The analyst and machine learning specialists that are my team’s customers use EMR clusters to perform queries on their data in S3. Their data is typically in csv format. Many times EMR is overkill for the types of queries they need to run. Spinning up an EMR cluster in the morning can take upwards of 30 minutes before they can start their work. Also, the clusters are terminated if they are inactive for an hour. I needed to improve the analysts’ working experience.   
     
   I introduced Athena and Glue to simplify our analysts' work for these types of queries. Now we transform their data: convert it to parquet and partition it. Are analysts can now query their data directly from S3 using Athena. I also introduced the Athena plugin for Tableau so they can use Tableau to build reports using their Athena queries. This has saved our analysts hours each day. We have dozens of analysts.
6. Bias for action:  
   Our change management process requires a manual approval of every change record by the owning manager. When our developers merge to master for a production elevation, a change record must be created and then approved manually by the manager. My teams do dozens of production elevations each day. My peer managers have similar volume (there are hundreds of us). I took it upon myself to improve this process.  
     
   I got the chance to participate in a skip level with our CIO. I took him through this wasteful process and explained my proposed solution. He asked me to write my proposal up and send it to him. I did and he told his subordinates (department heads in IT) to work with me to implement it.  
     
   The solution: when a dev merges to master for a production elevation, auto-generate and auto-approve a CR. The change will automatically deploy to production. The auto-generated CR is marked as auto-approved by the responsible manager. The manager signs a contract with ops stating that his auto-approve CR process is contingent on remaining below 1% change failure rate. Once the team exceeds 1% change failure rate, their auto-approved CR process is revoked.  
     
   This process has saved our development team thousands of hours of waste time in our deployment cycle. It has also made me a hero among the devs :) I got 100% favorable on our recent anonymous employee survey.
7. Always learning:  
   Our firm was spending an inordinate amount of time manually reconciling breaks with our trading partners, including verifying thousands of breaks as auto-reconciliation eligible. We had a staff of reconcilers manually working every reconciliation break. The staff (approximately 30 FTE) across our three trading locations (Australis, EU, and US) worked these breaks every global trading day.   
     
   I learned of other implementations of machine learning, specifically classification algorithms, in our firm and I got the idea to use machine learning to eliminate the waste in our reconciliations department.  
     
   I led the effort to predict probability reconciliation breaks will acceptably auto-reconcile or need manual reconciliation. My team (specifically a co-op student I hired from Drexel University) and I used SageMaker and its Random Forest algorithm (and later XGBoost) to build a regressor. We also performed data engineering on our daily reconciliation data to remove dimensionality using principal component analysis. It was experience, we were able to move 10 FTE to more productive tasks, saving the firm approximately 2 million dollars annually. Also, the Drexel intern loved his assignment so much that I was able to hire him full time.   
     
   Bulk exported Oracle tables to CSV on S3, then used EMR Spark jobs to transform the data (remove columns, merger columns, etc) and place it in our ML S3 buckets.  
   Used Random Cut Forest with mean absolute error as the validation metric. Used a pipeline to preprocess (impute missing values in numerical data, encode categorical data) and fit the model. Then predict, and score the model.  
   I then used the XGBoost regressor using the mean absolute error metric for validation. I used number of estimators, early stopping to find the optimal number of estimators, and learning rate. Used large number of estimators and small learning rate. Took longer to train, but we had time to get our model right. Used cross-validation with mean absolute error for validation.
8. Building a deep learning model to predict how many contracts, and which side, of the S&P and/or NASDAQ bigs to keep our excess cash in our funds invested at the end of the trading day (overnight money). Using Keras Sequential class with three hidden layers and relu activation function. Using early stopping and dropout to correct overfitting. Loss Function: Mean Absolute Error, Optimizer: stochastic gradient descent - Adam.
9. Built an internal web app that allows traders to get a quick view of cash available in their fund at any time throughout the day. Ingests cash available from an internal system using Kinesis Data Firehose to S3 in batches. When the data batch is written to S3 a Lambda function is triggered. The Lambda function updates the data in a DynamoDB table. Wrote a static website hosted on S3 written in Typescript using Angular. The Angular SPA uses Angular services to call REST APIs written in python as Lambda functions behind API Gateway. Authentication through Cognito and Active Directory identity provider. Use JWT (OAuth 2.0) for API call authorization.