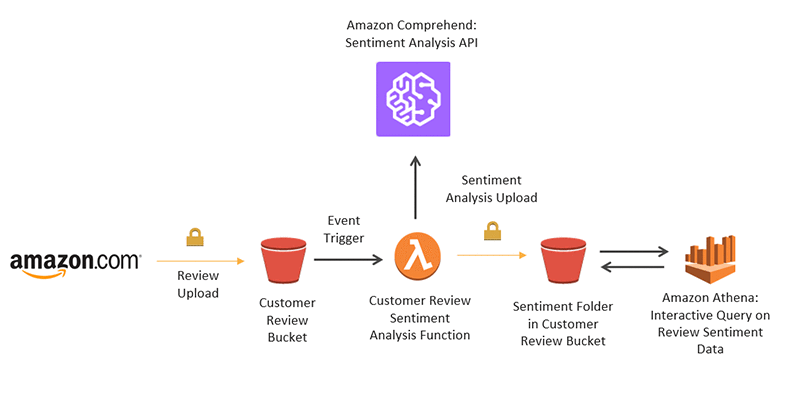
2021 AWS Senior Architect Interview Scenario Material

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 RDS to get ACID
2. Ownership  
   ETF bursting application (managed by another IT team) needed ETF total and estimated cash. Could have told them to add to their monolith codebase by querying the database directly, but took ownership of the effort to create an API to help modernize the application integration.   
   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. Exposed /total\_cash and /estimated\_cash endpoints using API Gateway, which called Lambda functions that calculate the total and estimated cash  
     
   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 GET 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)  
     
   API integration allows for much faster and mode efficient management of our codebase. We now deploy our API separately from the ETF bursting application code base. Allowing our code bases to achieve much more frequent deployment rates.
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. Invent and simplify:   
   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 ECS Fargate ETL job scheduler and use a Lambda function to restart jobs when needed. Our ETL jobs run as tasks within a Fargate cluster. 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 obsession:  
   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. Our 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. Insist on the highest standards:  
   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 responsible 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 0.25% change failure rate. Once the team exceeds 0.25% 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. Learn and Be Curious:  
   Our firm was spending an inordinate amount of time manually reconciling breaks with our trading partners, including verifying thousands of breaks daily 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 (AUS, 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 reduce dimensionality using principal component analysis. It was a great 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. This work also got me started on my journey to the Machine Learning Specialty certification.  
     
   Bulk exported Oracle tables to CSV on S3, then used EMR Spark jobs to transform the data (remove columns, merge 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.  
     
   Deployed Random Cut Forest (and XGBoost) based model using SageMaker. When new batch files arrive on our operational S3 bucket, a Lambda function is triggered via API Gateway to send the batch data to the inference endpoint.
8. Think Big  
   Portfolio managers need a way to profitably buy/sell futures contracts to keep their excess cash invested in the markets. They have been buying/selling based on simple spreadsheet based calculations. I proposed that they use a deep learning model to predict which side and how many contracts to buy. This required me to convince the PMs that a model could outperform their current process. Using training and verification results to compare against actual results in the market.  
   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. Will deploy the model, saved in protobuf format, using SageMaker Neo. Plan to put a REST API in front of the model using API Gateway and Lambda. Then build a SPA using Angular. Will use Cognito for authorization, Active Directory identity provider. Use JWT (OAuth 2.0) for API call authorization.  
     
   This approach has the potential to increase our excess cash earnings in all of our funds. We have hundreds of funds. A 1% increase in excess cash earnings will equate to several million in earnings per year across the investment management portfolios.
9. Deliver results   
   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.
10. ?  
    Lake Formation:  
    Use Glue Data Catalog as our Hive Metastore  
    Use Federated single sign-on to EMR (v5.31) Notebooks from enterprise identity provider (active directory) using (SAML) 2.0.  
    Enforce fine-grained, column-level access control to data tables based on policies defined in Lake Formation
11. Bias for action  
    We had a very manual process of analyzing our customer survey data. We receive thousands of customer calls daily to our phone associates where customers are looking to get help investing, open an account, change account information, as well as many other tasks. We survey our customers after these calls. We use the Medallia survey service, which sends us comments and feedback mentioned in the surveys. We ingest the survey data from Medallia into our S3 data lake as CSV files. We had analysts reviewing the CSV files and marking each survey comment/feedback as a complaint or not. We do this to improve our phone associate service as well as to meet regulatory compliance mandates.  
      
    I took the risk of proposing a machine learning solution to this problem, even though I encountered initial resistance from my internal stakeholders. I proposed that we improve the process by using a machine learning model to assess the comments/feedback records as a complaint or not. I knew we could easily revert back to our manual process if the idea didn’t work. I proposed that my development team use this architecture: when Medallia data arrives in our S3 bucket, trigger a Lambda function that uses the Comprehend API to perform sentiment analysis. The Lambda function writes the sentiment for each comment/feedback to a CSV file in S3. Athena is used to query the sentiment data on S3.  
      
    Our process allowed our analysts to process the survey comment feedback data much more quickly and with far less analyst time. We saved 10 hours per day of analyst time, which equates to around $300k per year.  
    [](https://aws.amazon.com/blogs/machine-learning/detect-sentiment-from-customer-reviews-using-amazon-comprehend/)
12. Are right, a lot  
    We have an internal web app that allows portfolio managers to analyze their fund tracking error over time. It was built as an Angular SPA, written in Typescript, hosted as a static website on S3. The Angular app makes REST API calls to Lambda function endpoints fronted by API Gateway. The Lambda functions store and retrieve the tracking error data from a DynamoDB table.  
      
    This web app was initially built as a prototype without authentication. I proposed that we use Amazon Amplify to layer authentication onto the web app. I argued that we could use Amplify to easily add authentication with very little change to the underlying Angular code. Developers and architects on the team wanted to follow the traditional path of coding the login authentication flow by creating our own Typescript components. I made the decision to have the team, myself included, add a login flow to our Angular app using Amplify with just simple Cognito user pool authentication and signup. The idea was that if this prototype was successful, we would then add an active directory SSO using SAML 2.0.  
      
    It took me less than a day to add a login and signup flow to our Angular SPA. I just ran the amplify add auth CLI command, then added the amplify-authenticator and amplify-sign-out components to our base component (around the router-outlet tag). Deployed our amplify stack and deployed the web app and it worked flawlessly.   
      
    Now my developers and architects are excited to use Amplify to graft on SSO to the app. I saved our developers a sprint of Typescript development.
13. Hire and develop the best  
    My development teams asked for a way to make their coding learning more interesting. They all spent time learning new development skills on their own, but they wanted a way to learn coding as a team. I proposed that I create a monthly Python coding contest where each month the winner would get $250 in cash. They loved the idea.  
      
    I used HackerRank to create monthly contests with challenges. I created several challenges per week. The developers earned points for successfully passing the test cases associated with the challenges. Some test cases were public, some hidden. Whoever accumulated the most points over the given month won. This made the weekly Python training sessions I held for the team very interesting. I used the Python Data Science Handbook by Jake VanderPlas as the training text.  
      
    By holding these training sessions and the contests I built a reputation around our larger IT department as a leader that made learning fun and spent the time to teach my developers. This has allowed me to easily cherry pick some of the best developers in our IT department to come work for me.
14. Frugality, constraints breed resourcefulness, self sufficiency, and invention  
    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 ECS Fargate ETL job scheduler, replacing the Control-M scheduler, and use a Lambda function to restart jobs when needed. Our ETL jobs run as tasks within a Fargate cluster. 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.  
      
    The constraints of the company’s 7-day lockdown policy led me to invent a process where we control our own job scheduling. 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.
15. Earn trust  
    I took over the leadership of a new team in September of 2020. The developers were still getting used to working in a completely remote environment. I needed to gain their trust as their new leader. All but one had never met me in person since I came from a different IT subdivision, so I had not worked with them before. The exiting manager gave me a spreadsheet with a row for each employee and contractor with a few notes on each developer’s performance. She then moved on so I had to learn the new software systems (analytics and machine learning) and the people on my own.   
      
    I established biweekly team meetings where I took my developers through everything I knew about their working environment, the IT division, and the company as a whole. I still do this today. I am very candid with them about their work environment and the management of the return to work strategy. I also hold weekly training sessions with them where I teach them Python for data analytics and machine learning using HackerRank and Kaggle. I never cancel these sessions so the messages are very consistent.  
      
    We just completed our company-wide employee survey where the employees are asked many questions about their engagement at work. For the question “My direct leader is a great role model for employees” I got 100% positive. For the question “My direct leader keeps me informed about what is happening” I also got 100%. These beat the company average by 12 and 9 points respectively.
16. Dive deep  
    I need to manage the change control of my software developers in a way that allows for speed of delivery. We have a manual approval process for all changes, even CI/CD changes. Therefore, I need to know what my developers are trying to elevate to production well before the manual approval stage arrives.  
      
    I stay connected to the details of my developers work. I use the Flow development metrics dashboard to track and audit my developer’s work. The tool gives me a dashboard that aggregates the version control (Bitbucket) and agile management (Jira) databases into a set of views. These views allow me to drill down on every commit, merge, and pull request for every developer. I use these views in one-on-ones and sprint retrospectives.   
      
    This approach has allowed me to approve every change record as soon as it arrives in my in-box. I have the team working in the “elite” status for delivery. We deliver far more changes (dozens per day) than any other team in our Client Experience department.
17. Have backbone; disagree and commit  
    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 responsible 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 originally proposed a solution to this problem in 2019. It was considered but ultimately our then CIO decided against it. Even though I disagreed with the decision, I committed to our CIO’s approach.   
      
    In 2021, I got the chance to participate in a skip level with our new 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 0.25% change failure rate. Once the team exceeds 0.25% 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.
18. Deliver results  
    In my AWS specialist training position with WhizLabs I build training material: videos and practice test questions for the AWS certification exams. WhizLabs asked me to deliver a video course for the AWS Machine Learning certification exam. They needed the complete production ready course material, which turned out to be over 70 video modules, in two months. This was a significant challenge for me to deliver on time. I needed to produce training material at the best level of quality that the given time allowed.   
      
    I set a schedule for myself of producing a module per night, 7 days per week. For each module I had to do the research for the subject, for example dimensionality reduction, produce the slide material, including diagrams drawn by hand using draw.io, record the video content, and edit the content. This took from 2 to 4 hours per night.  
      
    Halfway through the process, WhizLabs decided to change some elements of the slide content template. This meant I had to modify the recording I already made to meet the new requirement. I used the Camtasia layover feature to place the new content in the timeline of each of the videos and I re-recorded the voice over for several sections.  
      
    I was able to deliver the content on time. I consistently get 4.5 stars in the over 1,000 reviews from WhizLabs course customers for my Machine Learning course.
19. Strive to be the Earth’s best employer  
    In the pandemic work remotely environment I needed a way to make our work fun for my employees. We used to go to happy hours and do other things when we could be physically together.  
      
    I did some research and spoke to my employees in an attempt to create a fun environment while we’re all remote. I came up with using the Airbnb virtual experiences as a way to get together and have fun. I bought a “Create chocolate from cacao with a Jamaican cacao plantation owner” experience and a virtual escape room with a team from Krakow Poland. These were a tremendous hit with my employees and they only cost me $250 and $300 respectively.
20. Are right, a lot  
    Working with WhizLabs, I needed to create AWS Data Analytics certification questions that are challenging and that cover the appropriate material for the exam. Spent many hours researching and writing more than 150 questions. The questions were published and we started to get feedback. Received 4.5 stars average from over 1,000 reviewers. However, some write in comments called some questions and references for the answers into question. Reviewers stated that some questions and/or answer explanations were unclear or even incorrect.  
      
    I reviewed each feedback item and compared the comments to the questions, answers, and answer explanations/references. I found that most comments were easily explained as either a misunderstanding of the question, or a lack of knowledge of the reviewer. However, a few comments had genuine assertions about the correctness of the content. I made the point of accepting the feedback and quickly correcting the contant. My acceptance of the feedback and the action I took to correct the issues quickly allowed WhizLabs to show that they take proactive responsibility for their product. Rating stars continue to be high and one reviewer even made a video where he described how the questions and the associated video course, which I also created, really helped him pass his Data Analytics Specialty certification exam.  
      
    I was right about most of the content I produced, but when I was wrong, I took immediate action to correct the mistake.