Greatest Contribution

My contribution that delivered the most value this year is efficiently integrating eligibility service (ELIS) with more AWS technologies like S3, Lambda, SQS, Event Bridge, Eventing, Aurora setup/migration, and Disaster Recovery.

Eventing will make fundamental architectural improvement in eligibility 2.0 journey by adopting to Events, triggered from Monolith eligibility check, and passing over many hops such as Collector message bus, lambdas, SQS, event bridges. This change on one hand will help surface Eligibility benefits in more performant way over the time and on other hand will unlock many other potential workflows where dependent workflows can subscribe to these events and take near real time decision making.

Successfully migrating regular RDS to Aurora RDS helps in many aspects, such as out-of-the-box failover in case of disaster recovery, lesser maintenance, better performance, and saving some expense. This is also a milestone in Disaster Recovery Initiative.

My main goal for this year is to become more familiar and hands on with AWS technologies while making improvements in ELIS so it can be model for others to follow. By integrating ELIS with more AWS services, I completed tasks like running AWS deployments with various CICD pipelines, troubleshooting Terraform failures on the go, closely collaborating with MDIP team for Snowflake integration, creating various AWS resources to work together for ELIS, and contributing to IAC teams in Athena to benefit COLTEC along with other AWS product engineering teams like COLTEC. This leads to achieving my goals because completing these tasks required me to have precise understanding of AWS services with IAC and resulted in quality improvements in ELIS.

Here are some examples of the tasks I completed for this contribution.

1. Eventing
   a. [COLTEC-995] Create Eli-S3 bucket to store benefits data - Jira (athenahealth.com)
   b. [COLTEC-1000] Lambda Infra and message processing - Jira (athenahealth.com)
   c. [COLTEC-1002] Insert the received track data into PSQL tables - Jira (athenahealth.com)
   d. [COLTEC-1035] Create new PSQL Track table - Jira (athenahealth.com)
   e. [COLTEC-1039] Optimize Lambda DB connection during initialization - Jira (athenahealth.com)
2. Aurora & Snowflake
   a. [COLTEC-454] Drain production data into snowflake - Jira (athenahealth.com)
   b. [COLTEC-1093] Come up with Aurora deployment plan - Jira (athenahealth.com)
   c. [COLTEC-1094] Do the actual Aurora migration [dev] - Jira (athenahealth.com)
   d. [COLTEC-1096] Do the actual Aurora migration [prod] - Jira (athenahealth.com)
   e. [COLTEC-1148] Onboard Aurora to MDIP - Jira (athenahealth.com)
   f. [COLTEC-1163] Eventing Lambda Point at Aurora - Jira (athenahealth.com)
3. DR & Others
   a. [COLTEC-1144] Create SNS Topic datafabirc_xmatters - Jira (athenahealth.com)
   b. [COLTEC-1184] Contribute to IAC Teams - Jira (athenahealth.com)
   c. [COLTEC-1254] Gating Eventing & DR in Develop Branch - Jira (athenahealth.com)
   d. [COLTEC-1297] Rebuild Integration - Jira (athenahealth.com)

Growth

Over the course of the year, I have grown in areas like initiating technical designs and enforcing engineering excellence while steadily contributing to COLTEC initiatives.

I have initiated a lot more technical designs comparing to last year. The range of the topics have also expanded, not only includes product engineering solutions, but also includes AWS integration plans, CICD pipelines, domain entity modelling, and code base optimization. When constructing solutions, I always look for more than one options to solve a problem with Pros/Cons. When creating design documents, I always write checklist based repetitive processes so others can follow. As the result of my technical design, I make sure COLTEC has a consolidated solution, a document of implementation procedure to follow, and corresponding Jira tasks created. Leading technical designs have helped me to gain more domain knowledge and more experience in approaching problems.

I have created POCs ranging from inserting a new Nimbus React app in monolith to integrating a new stack of AWS services with ELIS. POCs aim to verify the feasibility and foresee the drawbacks of a big technical solution before the team has invested too much into it. Since COLTEC works on multiple platforms within the same project, it is very difficult to arrive at a consolidated solution at the beginning that guarantees every piece can come together. It would be catastrophic if the team works on a solution for a long time and discovers major flaws when it is almost releasing time. POCs can reduce complexities in product conversations as well as guide the upcoming architectural changes. Creating POCs have helped me to gain more robust understanding of every technical aspect related to COLTEC and the dependencies between them.

Here are some examples of my technical design documents.

1. Prototyping POCs
   a. [SPIKE] Fetch All Bus Call Improvements - Collector R&D - Confluence (athenahealth.com)
   b. [SPIKE] Eligibility Eventing Lambda - Collector R&D - Confluence (athenahealth.com)
   c. RDS to Aurora Migration Plan - Collector R&D - Confluence (athenahealth.com)
   d. Copay Nimbus App - Collector R&D - Confluence (athenahealth.com)
   e. Microservice Endpoints - Collector R&D - Confluence (athenahealth.com)
2. Engineering Excellence
   a. [SPIKE] Where Should Eligibility Data Live in the Future - Collector R&D - Confluence (athenahealth.com)
   b. Code Optimization Items - Collector R&D - Confluence (athenahealth.com)
3. CICD
   a. [SPIKE] Future of Lambda Deployment - Collector R&D - Confluence (athenahealth.com)
   b. Spike - Future of ELIS Deployment - Collector R&D - Confluence (athenahealth.com)
4. Product Research & Solutions
   a. SPIKE: API for Eligibility Header to use 2.0 - Collector R&D - Confluence (athenahealth.com)
   b. Spike: Grace Period - Collector R&D - Confluence (athenahealth.com)
   c. ELID Troubleshoot - Collector R&D - Confluence (athenahealth.com)
   d. [SPIKE] What Logic Drives Copay Type? - Collector R&D - Confluence (athenahealth.com)
   e. Copay Tech Design Discussion - Collector R&D - Confluence (athenahealth.com)

This year I spent great time and effort into enforcing engineering excellence for COLTEC. I fully participated in the SDLC to enhance ELIS and Patient Eligibility Details (ELID). I proactively seek code reviews by well qualified developers and perform thorough code reviews for others to help improve robustness, performance, security, and maintainability of ELIS and ELID. I also proactively share any useful information with the team to avoid surprises for anyone.

I have initiated the creation of LMM Dashboarding with Prometheus and Grafana for ELIS. Although we could see a few default metrics in the AWS console, it is crucial for us to be able to read more meaningful data from our microservice in production. I learned a lot more about the metrics that are generated by our microservice, EC2 instance, and ECS cluster with a new set of query language called Prometheus Query Language or PQL. This process helped me to grow towards a more experienced full stack cloud developer by making me focus on monitoring the detailed performance of microservice and AWS.

I have initiated the progress of significantly improving performance in the monolith Fetch All API. Monolith Fetch All API 95th percentile latency was hovering 2.5+ seconds and more, but after my two rounds of improvements, it has come down to range of 1 second to 1.5 seconds, that is overall improvement of 40-50%. This initiation required engineering precision, understanding of system, monolith code grasp, speed of change, quality of work, backing up thoughts with investigation, comparing results using profiling tools, and determination to make change for betterment. This process has helped me to grow into the mindset of always looking for new ways to improve the performance and the value Athena delivers.

I have initiated the practice of building modular code that can be readily leveraged and maintained over time. I have been monitoring the team's accumulation of tech debt and attempting to ensure adequate payoff of debt. Developing new projects while carrying tech debt not only slows down the velocity of the team but also greatly increases the chances of bugs. I have created Confluence pages to document the items in COLTEC code base that do not adhere to common design patterns or programming idioms. I have also created and enforced the execution of code clean up epic for COLTEC. This process has helped me to grow towards a more experienced full stack cloud developer by making me more aware of clean code, best programming practices, and best design practices.

1. Prometheus & Grafana
    a. [COLTEC-429] Create grafana dashboards for our custom prometheus metrics - Jira (athenahealth.com)
    b. [COLTEC-526] Add Alert for MS Container memory usage - Jira (athenahealth.com)
    c. [COLTEC-578] Review Grafana Prod dashboard to surface latency (avg), throughput and errors - Jira (athenahealth.com)
    d. [COLTEC-580] Surface Percentile metrics and add to Grafana dashboard - Jira (athenahealth.com)
    e. [COLTEC-670] Grafana Prod is not showing option to surface prod data - Jira (athenahealth.com)
    f. [COLTEC-888] Modify Grafana Panel "JVM Memory Heap Usage" - Jira (athenahealth.com)
2. Fetch All Improvements

    a. [COLTEC-827] [Fetch All Improvements] Update FetchAllEligibilityInfo to use GetEnhancedEligibilityHistory - Jira (athenahealth.com)
    b. [COLTEC-828] [Fetch All Improvements] Create app layer function GetEligibilityBenefitsFromTrack - Jira (athenahealth.com)

3. ELID Cleanup
    a. Code Optimization Items - Collector R&D - Confluence (athenahealth.com)
    b. [COLTEC-842] Cleanup service folder - Jira (athenahealth.com)
    c. [COLTEC-843] Clean up types.ts - Jira (athenahealth.com)
    d. [COLTEC-849] TSX cleanup in ServiceBenefit - Jira (athenahealth.com)
    e. [COLTEC-850] TSX cleanup in EligibilitySection - Jira (athenahealth.com)

Furthermore, I contributed to all aspects of SDLC and full stack cloud development. I conducted DB design, conducted core system design, created multiple backend Java APIs in microservice, enhanced existing React app, conducted production bug analysis, fixed ELID bugs that reduced error count to 10%, and assisted PO, Analyst, UX in product level discovery spikes to make informed decisions. The successful GA of ELID and ELIS is a milestone and reflection of my steady contributions while setting up a path for the next generation of changes in Athena. This process has helped me to gain more understanding of agile values and principles by holding myself accountable for the outcome of COLTEC work.

1. ELID Enhancements (First Round)
    a. [COLTEC-465] Back End Support for 'Service Type Sent' - Jira (athenahealth.com)
    b. [COLTEC-474] API for Eligibility Header & Summary from 2.0 - Jira (athenahealth.com)
    c. [COLTEC-486] Identify benefits that are empty - Jira (athenahealth.com)
2. ELID Revamp (First Round)
    a. [COLTEC-668] Prototype EliD (1.5) - Jira (athenahealth.com)
    b. [COLTEC-696] Service Types Tab - Individual STC Cards Back End - Jira (athenahealth.com)
    c. [COLTEC-701] Individual STC Cards Front End/UI - Jira (athenahealth.com)
    d. [COLTEC-702] Move date selection to single conslidated dropdown - Jira (athenahealth.com)
    e. [COLTEC-706] Favoriting User Experience: General - Jira (athenahealth.com)
    f. [COLTEC-709] Favoriting User Experience: Shuffle - Jira (athenahealth.com)
    g. [COLTEC-713] Default ordering of suggested service types - Jira (athenahealth.com)
    h. [COLTEC-714] Error cases while favoriting - Jira (athenahealth.com)
    i. [COLTEC-723] Add 'PCP' to Header - Jira (athenahealth.com)
    j. [COLTEC-733] Achieve parity with red text warnings - Jira (athenahealth.com)
    k. [COLTEC-735] Displaying Information Not Returned from Payer - Jira (athenahealth.com)
    l. [COLTEC-783] Maximize Header Insurance Package Line Length Before Wrap - Jira (athenahealth.com)
    m. [COLTEC-812] Force users again and again to use EliD instead of legacy elig page - Jira (athenahealth.com)
    n. [COLTEC-815] Different survey links based on nimbus version/alpha wave - Jira (athenahealth.com)

    o. [COLTEC-816] Have Clickable DOS in History Tab for a Custom Check Take the User Directly to the Requested STC's Card - Jira (athenahealth.com)
    p. [COLTEC-818] Defect - Spinner keeps going on - Jira (athenahealth.com)
3. ELID Enhancements (Second Round)
    a. [COLTEC-920] Update Navigation from Custom Elig Check to go to Service Types tab for the Date of Service Requested - Jira (athenahealth.com)
    b. [COLTEC-1101] User favorites are not getting saved with EliD 1.3.2 version - Jira (athenahealth.com)
    c. [COLTEC-1155] EliD 1.3.3 throwing error intermittently - Jira (athenahealth.com)
    d. [COLTEC-1196] Investigate switch-to-legacy bugginess - Jira (athenahealth.com)
4. ELID Revamp (Second Round)
    a. [COLTEC-1253] Make ELID "DOS Dropdown Oriented" - Jira (athenahealth.com)

Aspirations

My career aspiration is to become a successful and experienced full stack cloud developer.

This year I have gained experience in implementing AWS services, running Terraform deployments, and improving quality of microservice. This experience both strengthened my cloud developing skills and my SDLC understanding, therefore it sat me up to become a more experienced full stack cloud developer.

For next steps, I wish I can further hone my AWS skills as well as my data structure design skills. I hope to gain more opportunities in designing cloud architecture, creating POCs, and enforcing best technical practices. Especially with supporting Copay work with Evented data, I will aim to focus more on logging, monitoring, and performance improving as they will help me to better assess the quality of COLTEC cloud stack.

Shortfalls

When COLTEC worked on Eventing, Aurora migration, and DR, we have caused decent amounts of AWS build failures and environments dependency issues. We ended up spending a lot more sprint capacity than we originally planned to resolve these problems. The cause of this shortfall is due to the sudden expansion of AWS resources in ELIS and the lack of mature dependency planning within the team.

What I learned from this experience is the importance of following best infrastructure practices, adopting extensible design, and starting clean at the beginning. Although there is always more than one correct way of best practices, the discussion of what is more optimized should become a habit for the team. Going forward, I will be more alert with enforcing best practices and picking up crucial infrastructure tasks myself. I will also seek to improve my skill with attention calling and idea messaging.