Candidate 8498373

1.a. Currently our internal APIs are written in Python and the frontend is using React JS. We are experiencing performance issues, with some concern of the latency of the responses from our APIs, and/or performance of our UIs. We want to hear about the approach you would take to investigate potential bottlenecks, and which solutions you would propose to solve these issues.

Sure. The first thing I would do is to understand how the resources are deployed in the cloud. From the architecture shared separately it is not apparent, but the physical location of the resources in the cloud can impact negatively creating extra latency. It may be strategic to move some resources to be closer to each other, or to create more regional replicas/caches closer to the users.

Once that is taken care of, I would start gathering metrics on the current performance of both frontend and backend. For the React application I normally use the React Dev tools to identify unnecessary re-renders that could be affecting performance of the user interface. Implementing an internal policy to use React Strict Mode at development time can help with finding out these performance issues. Also, for the frontend I would look at the code and the build process to determine if techniques such as code splitting or lazy loading could make an improvement, depending on how large the codebase is.

For the API calls, I would identify frequently accessed endpoints and make sure proper caching plans are in place using well-known solutions like Redis. Next, I would look at the current load balancing solution to see if some improvements could be made, while also determining if the current backend could benefit from scaling up resources; it may be worthwhile here to consider flexible and automated scaling options to improve performance during peak consumption.

Other things to look at are database queries and how they could be optimized. I would propose to run profiling on them to identify issues with the queries, table, or view design. In my experience optimizing database queries or improving the overall design of the databases can impact the performance noticeably.

1.b. Now, our MLOPS process is very manual for instance with local model training. What technologies and services would you recommend to improve traceability, model versioning, results reproducibility of our ML operations? Describe the steps you would take for moving towards a more automated workflow.

To move towards a more automated flow, I would first propose to containerize the ML environment, to ensure consistency across different machines and platforms, if not done already. I would also propose to use Kubernetes technology, which brings scalability, fault-tolerance, and automation of the deployment process. For versioning and traceability, I would investigate existing tools in the market, either by our cloud provider or open-source solutions like MLflow. Full disclosure, my experience with MLOps is limited, so to guarantee the best

results I would support myself with in-house experts and/or with an IT consulting firm on the best practices and strategies, as a first step.

1.c. As the system is getting moved to another cloud provider, what approach would you take to ensure consistency and flexibility for the deployment of the related codebase. What systems and strategies would you put in place to make that process smooth for developers and keep downtime to a minimum for the end users?

Firstly, designing a plan with scenarios is critical to the success of any cloud migration process, including options for rolling back in case something does not go as expected. I would also suggest creating a DevOps taskforce (could be a temporary team if not already a DevOps team in-house) to assist with all the steps involved into the process. Most importantly, carefully defining roles and responsibilities for all the staff involved in the process is crucial.

To ensure consistency, I would propose tools like Terraform to make sure the resources in the new cloud environment replicate those we had before, and this allows versioning of the infrastructure. I would then look at the current CI/CD strategy and adapting it as needed for the new cloud, this will ensure a smooth transition for developers. To keep downtime to a minimum for end users, I would propose a Blue-Green deployment strategy, which involves a hybrid cloud approach; the codebase would be deployed to the new cloud and after proper testing, traffic could be switched to the green environment from the blue environment, thus ensuring minimum downtime and maximum flexibility for the migration process. If issues are detected with the new environment, end-users could be seamless switched back to the blue environment while issues are sorted out. Monitoring systems should also be put in place to make sure the new environment is performing as expected.

Finally, while all the above is important, I personally think it is crucial to consult with IT industry experts (e.g., Gartner, Forrester) to make sure the migration strategy is considering every aspect and it is up to date with current best practices and technology trends.

Thank you for your consideration.