**Leaderboard System Design**

Tech Stack Selection:

- Services: **Python.** Python is a reliable and lightweight language that is known for its speed when working with large amounts of data. Psycopg2 can be used as the db connector.
- Database: **Aurora PostgreSQL**. This database is great for high availability and high performance, and can easily be configured into multi-zone clusters for scalability, reliability, and availability.

Architecture Design:

- A user's request will first get authenticated before it invokes our API
- The request will get processed by our API and the corresponding lambda will be invoked
- The compute layer lambda functions will run business logic and send requests to the database
- The persistence layer will be a database cluster in multiple availability zones to increase scalability (load balancing), availability (more databases to access), and redundancy (multiple copies of the data across zones).

Infrastructure Design:

- For the API layer, API Gateway offers an edge-optimized, scalable, and serverless solution. To authenticate the calls, a dedicated Authorizer lambda function will run, connecting to whatever authentication method we are using.
- The API Gateway API can be set up with Route 53 to assign a domain alias
- Once authenticated, the corresponding lambda functions will run for the endpoint that was invoked. Lambda functions offer scalable, serverless benefits that will help the throughput of the API. Reserving concurrencies for any of the lambdas can offer latency reduction during scaling for an increased cost.
- Our database will be a cluster of Aurora Postgres instances across multiple availability zones. Using a cluster will ensure scalability due to load balancing and data redundancy due to replication across instances and zones. If one zone is unavailable, traffic will be routed to other zones and a new primary instance will be selected. Offering read replicas will also improve performance and decrease the work that the primary instance has to do.
- The tables and views will see performance boosts after indexing correctly.
- Everything behind the API Gateway will be part of a VPC for improved security.

Testing:

- After creating the new API endpoints, we can set up a **Postman** collection test suite that will ensure that an input event will always return the same response.
- Unit tests for the lambda services layer can be set up using **pytest**. This can be integrated into the CI/CD pipelines so that code is only deployed when the testing suite passes all tests.