Technical Implementation

To Implement Rocket Games Account management system below are technical stacks used for API and backend Database.

- AWS Cloud -API Gateway, Lambda, S3 Bucket, Secret Manager,IAM
- Snowflake - A SaaS Data warehouse hosted on AWS.

Snowflake Implementation:

Below are the technical steps involved to meet the designed data model.

1. Login to Snowflake Account and choose underlying infrastructure for snowflake as AWS
2. Run the script deploy_db_objects.sql provided in code package file /GameProject/build folder. This script is responsible for creating DB artefacts such as Database, schemas, warehouse, stages, snow-pipe and Tables etc.
3. For Snowpipe- require to map the notification channel to S3 bucket event.
4. Run the script deploy_roles.sql provided to create roles and assign the privileges roles.
5. Mock data provided in /GamesProject/data-ingest folder for setting up the data. This can be loaded into snowflake staging schema Player and Stuido-Game stage tables using snowflake copy command provided in deploy script. At first need to upload these files in S3 external staging bucket.
6. According to data model target tables are created for reporting purposes in wh schema and ddls provided for the same.
7. Written SQL based data-pipelines in snowflake to ingest the data into target tables according to data model. Data-pipeline scripts provided in /Games/data-pipelines folder.
8. These data pipe line can orchestrated through snowflake tasks or external orchestration methodology.
9. Streams are created on stage schema table to fetch the CDC/incremental records.
10. Tasks are written to schedule the jobs based on data model hierarchy.
11. Reporting queries are provided in /Games/User-stories folder according to user story requirements.
12. Below is the sequence of loading the data into Target tables using data-pipelines from staging tables.
    a. Player
    b. Studio
    c. Games
    d. Player_Games
13. Below are staging tables.
    a. Player_stg
    b. Studi_game_stg

AWS Services Implementation:

Based on API design model, have created below services to cater the Rocket Games API using AWS services.

1. Created the s3 bucket **rocket-games-account-management** and object keys for storing the ingestion data and reporting queries . And also uploaded the lambda layers package file for accessing python external libraries.
2. Created lambda function- **rocket_games_api_lambda_integration** (lambda). Lambda handler code is available in /GamesProject /scripts folder.
3. Lambda IAM role create for accessing API gateway,S3 bucket and secrets manager.
4. API gateway- **rocket-games-api** and below resources/methods created for this.
    a. /player-ingest - POST method For ingesting player data through this method.
    b. /studio-ingest - POST method For ingesting the studio and games data through this api method.
    c. /studio -GET method For getting the studio based reports/actions.
    d. /rg-owner -GET method For getting the publisher based reports/actions.
5. Above methods are integrated with created AWS Lambda (**rocket_games_api_lambda_integration**)
6. API Gateway stage created and deployed the API to get API endpoint. I assume this end point will be given to UI/WEB system to hit the API gateway through requests.
7. Snowflake DB details are stored in AWS secret manager and role provided to lambda for accessing the Snowflake DB via lambda. The pre-requisite for this set the **Trust relationship** policy b/w snowflake and AWS through IAM.
8. FYI, provided sample API request files for testing API methods in /GamesProject/scripts/api-input-parameters

AWS Infra

1. Provided terraform infra script for applying the AWS services. File is available in /GamesProject/aws-infra/infra.tf file.

FYI. I have just validated terraform file not tested fully.