5 dana u oblacima – Hackaton

Opis zadatka

Glavni cilj na Hackaton-u jeste da se rešenje deploy-uje na AWS Cloud i da ispunjava funkcionalne zahteve iz Challenge faze.

Dodatno, tim će moći ostvariti bonus bodove ukoliko:

- Implementira dodatni funkcionalni zahtev
- Unapredi tehničko rešenje

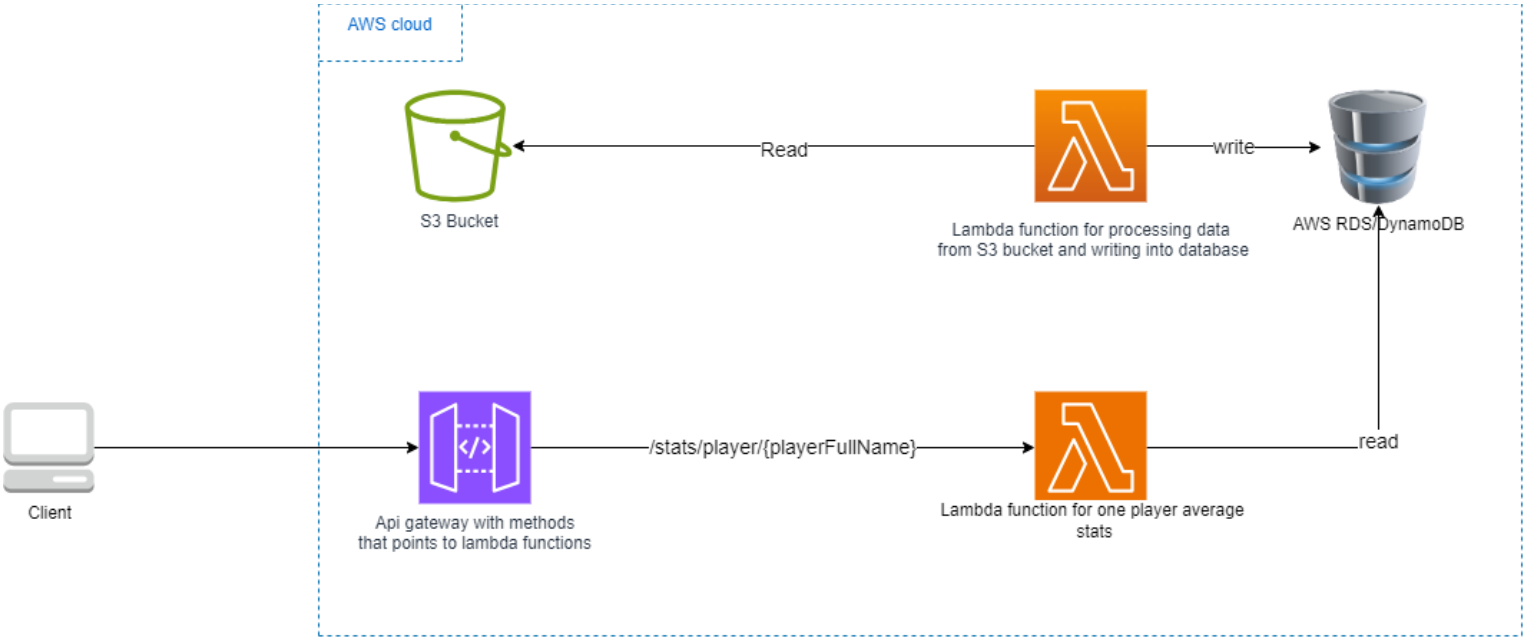
Za potrebe ove faze timovi će koristiti input fajl sa proširenim setom podataka (struktura fajla ne sadrži promene). Fajl možete preuzeti ovde.

AWS Account-i

- Svaki tim će raditi na odvojenom AWS account-u
- Budžet od 50 USD
- Region: eu-central-1 (Frankfurt)

Izvršavanje rešenja na AWS Cloud

Omogućiti da se rešenje iz Challenge faze izvršava na Cloud-u. Predložena arhitektura je data na slici ispod. Napominjemo da arhitektura rešenja ne mora biti ista kao predložena i da se mogu koristiti i drugi AWS servisi.



Koraci za implementaciju predloženog rešenja:

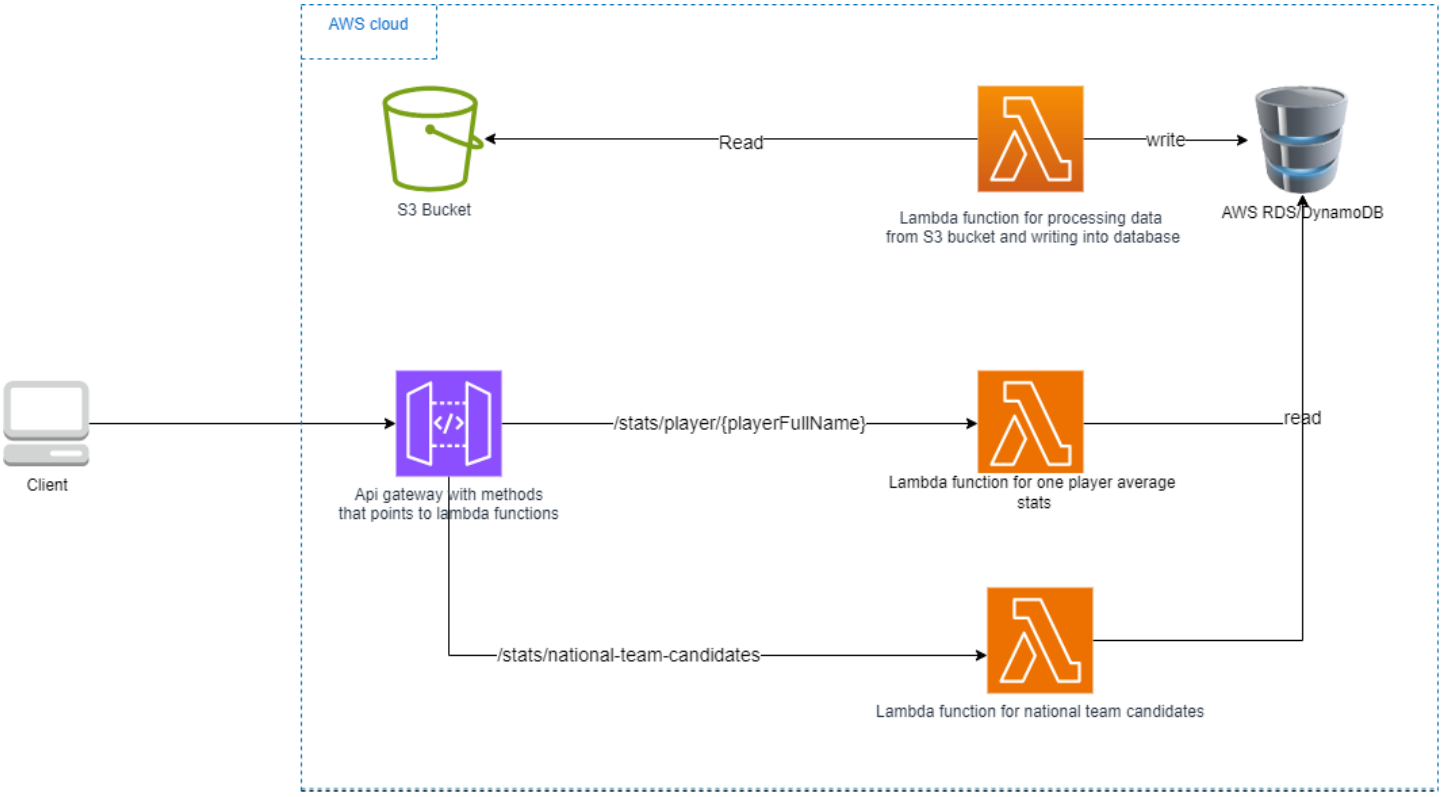
- Ulazni fajl smestiti na S3 Bucket
- Podesiti odgovarajuću bazu (NoSQL ili SQL)
- Implementirati Lambda funkciju za parsiranje fajla i smeštanje podataka u bazu
- Implementirati Lambda funkciju koja sadrži logiku za endpoint */stats/player/{playerFullName}*
- Konfigurisati API Gateway kako bi se lambda za */stats/player/{playerFullName}* mogla pokrenuti putem HTTP zahteva

Dodatni funkcionalni zahtev

Dodati endpoint koji će vraćati kandidate za nacionalni tim, tako što će za svaku poziciju pronaći po 3 najbolja igrača koji zadovoljavaju sledeće uslove:

- Najmanje 5 odigranih utakmica
- Uslovi za pozicije:
    - PG: Igrači sa najvećim hAST%, eFG%
    - SG: Igrači sa najvećim TS%, 3P%
    - SF, PF: Igrači sa najvećim TS%, REB
    - C: Igrači sa najvećim REB, BLK
- Napomena: igrače je potrebno prvo sortirati po prvo-navedenom parametru, a potom po drugo-navedenom
    - Ukoliko više od 3 igrača zadovoljavaju uslove za određenu poziciju, dodatno sortiranje uraditi po broju odigranih utakmica, a potom i leksikografski.

Arhitektura rešenja bi izgledala kao na slici ispod:



Koraci za implementaciju predloženog rešenja:

- Implementirati Lambda funkciju koja sadrži logiku za endpoint */stats/national-team-candidates*
- Konfigurisati API Gateway kako bi se lambda mogla pokrenuti putem HTTP zahteva

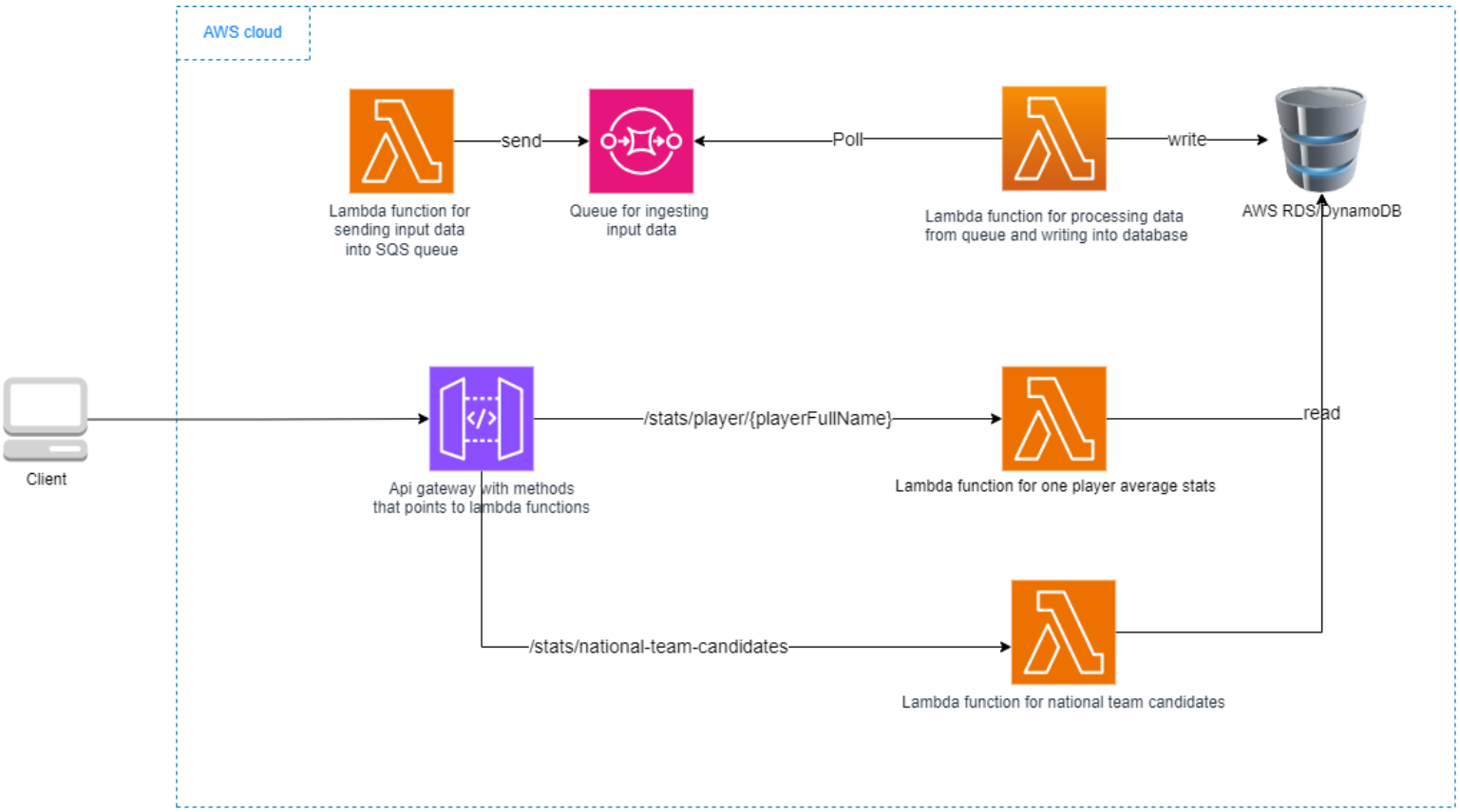
Specifikacija endpoint-a je data u tabeli ispod.

Path	/stats/national-team-candidates
Request method	GET
OK response status code	200
Response body	{ "pointGuards": [{

```
      "playerName": " Kodzo Danso",
      "gamesPlayed": 6,
      "hollingerAssistRatio": 5.6,
      "effectiveFieldGoalPercentage": 50.6
    }
  ],
  "shootingGuards": [
    {
      "playerName": " Rashid Kwami",
      "gamesPlayed": 5,
      "trueShootingPercentage": 45.6,
      "averageThreePointsPercentage": 30.6
    }
  ],
  "smallForwards": [
    {
      "playerName": " Monyyak Negus",
      "gamesPlayed": 7,
      "trueShootingPercentage": 66,
      "rebounds": 5.1
    }
  ],
  "powerForwards": [
    {
      "playerName": " Ekon Tariku",
      "gamesPlayed": 6,
      "trueShootingPercentage": 50.6,
      "rebounds": 12.1
    }
  ],
  "centers": [
    {
      "playerName": " Adom Runako",
      "gamesPlayed": 5,
      "rebounds": 12.1,
      "blocks": 4.3
    }
  ]
}
```

Tehničko unapređenje

Tehničko unapređenje podrazumeva da ulazni podaci ne budu učitavani iz fajla, već da ih rešenje preuzima sa SQS queue-a. Arhitektura rešenja je data ispod.



Koraci za implementaciju predloženog rešenja:

- Kreirati SQS queue i dozvoliti upis na queue sa account-a sa id-em: 967910360152
    - Nazivi queue-va po timovima:
        - 5-dana-u-oblacima-ankle-breaker-queue
        - 5-dana-u-oblacima-alley-oop-queue
        - 5-dana-u-oblacima-and-1-queue
        - 5-dana-u-oblacima-pick-and-roll-queue
        - 5-dana-u-oblacima-triple-double-queue
- Implementirati Lambda funkciju koja čita ulazne podatke sa SQS queue-a i smešta ih u bazu

Napomena: Lambda funkcija za smeštanje podataka na queue je implementirana od strane organizatora (kao i sam queue).

Poruka koja će biti smeštana na queue će biti u JSON formatu i predstavljaće učinak igrača na jednom meču, primer:

{
"player": "Monyyak Negus",
"position": "C",
”FTM": "2",
"FTA": "2",
"2PM": "7",
"2PA": "8",
"3PM": "0",
"3PA": "0",

```
        "REB": "5",
        "BLK": "0",
        "AST": "2",
        "STL": "0",
        "TOV": "0"
}
```


Bodovanje i prezentacija

Maksimalan broj bodova koji će tim moći da ostvari je 100, raspoređenih na sledeći način:

- Maksimalno 60 bodova za osnovni funkcionalni zahtev koji se izvršava na Cloud-u
    - Broj bodova će biti izračunat na osnovu tačnih asertacija od strane automatskih testova
- Maksimalno 10 bodova za dodatni funkcionalni zahtev
    - Granulacija na nivou pozicije – za svaku poziciju za koju je vraćen očekivani rezultat se dobija 2 boda; na primer – 3 pozicije dobar rezultat, 2 pozicije pogrešan rezultat - tim dobija 6 bodova
- Maksimalno 10 bodova za tehničko unapređenje
    - 10 bodova ukoliko rešenje uzima ulazne podatke sa SQS queue-a i prolazi sve asertacije automatskih testova; 0 bodova inače
- Maksimalno 10 bodova za kvalitet rešenja
    - U obzir će se uzimati stvari poput code quality, infrastructure as code (CDK)
- Maksimalno 10 bodova za prezentaciju

Trenutni plasman će se moći videti na Leaderboard-u.

Prezentacija

- Prezentacija treba da bude urađena u Powerpoint-u
- Ograničena na 15 minuta

Korisni linkovi

- AWS CLI - https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html
- AWS CLI with IAM identity - https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-sso.html
- Lambda with S3 example - https://www.freecodecamp.org/news/read-csv-file-from-s3-bucket-in-aws-lambda/
- Lambda with Java - https://docs.aws.amazon.com/lambda/latest/dg/lambda-java.html
- Lambda with Node.js - https://docs.aws.amazon.com/lambda/latest/dg/lambda-nodejs.html
- Lambda with Typescript - https://docs.aws.amazon.com/lambda/latest/dg/lambda-typescript.html
- Lambda with C# - https://docs.aws.amazon.com/lambda/latest/dg/lambda-csharp.html
- Lambda with Python - https://docs.aws.amazon.com/lambda/latest/dg/lambda-python.html
- AWS RDS - https://docs.aws.amazon.com/rds/
- AWS RDS PostgreSQL - https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.CreatingConnecting.PostgreSQL.html
- AWS RDS with Lambda - https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/rds-lambda-tutorial.html
- AWS RDS tutorials and sample codes - https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Tutorials.html
- AWS DynamoDB - https://docs.aws.amazon.com/dynamodb/
- AWA DynamoDB with Lambda - https://docs.aws.amazon.com/lambda/latest/dg/with-ddb.html
- AWS DynamoDb with Lambda code samples - https://docs.aws.amazon.com/lambda/latest/dg/with-ddb-create-package.html
- AWS SQS API Reference - https://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/Welcome.html
- AWS SQS Tutorials - https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-other-tutorials.html
- AWS Lambda with SQS - https://docs.aws.amazon.com/lambda/latest/dg/with-sqs.html
- Lambda with SQS tutorial - https://docs.aws.amazon.com/lambda/latest/dg/with-sqs-example.html
- SQS function code examples - https://docs.aws.amazon.com/lambda/latest/dg/with-sqs-create-package.html#with-sqs-example-deployment-pkg-java