5 dana u oblacima – Hackathon

Opis zadatka

Glavni cilj na Hackathon-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

AWS Account-i

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

Osnovni funkcionalni zahtevi

Pre implementiranja osnovnih funkcionalnih zahteva za ovu fazu takmičenja potrebno je prilagoditi rešenje iz prve (challenge) faze takmičenja pošto se dodatni zahtevi oslanjaju na tu implementaciju. Kao i u challenge fazi svaki tim će dobiti test kolekciju koja sadrži dodatne testove na osnovu kojih treba prilagoditi rešenje iz prve faze takmičenja. Postman kolekcija i AWS nalozi će vam biti prosleđeni na email (po jednoj osobi iz svakog tima).

Nova funkcionalnost koju treba podržati je startovanje meča za igrače koji nemaju tim. Na primer ako se startuje matchmaking, igrači koji nemaju tim će biti uparani sa drugim igračima koji takođe nemaju tim.

Potrebno je omogućiti da igrači menjaju timove. Na primer ako je “Player1” dodeljen timu, on od sada može da napusti tim1, postane slobodan igrač (bez tima) i zatim se doda u tim2 prilikom kreiranja tima2.

Promene koje je potrebno napraviti u odnosu na individualni zadatak:

- Igrači mogu da menjaju timove (ako je igrač već u timu, igrač može da promeni tim)
- Timovi ne moraju da imaju tačno 5 igraca, šalje se teamSize kao query parameter u request-u za kreiranje timova
- Dodati endpoint za brisanje svih podataka iz baze **(Ovaj deo je bitan za pregledanje zadataka tako da bi bilo dobro ovaj i naredni endpoint uraditi što pre)**

  DELETE /players

- Dodati endpoint koji vraća sve igrače sa njihovim statistikama (**Isto kao i prethodni endpoint, koristi se za pregledanje pa bi bilo dobro napraviti ga pri početku izrade)**
  GET /players
  Response:


```
[
  {
    "id": "b769b730-d1b9-4a94-8d2d-2936e050722c",
    "nickname": "Player1",
    "wins": 2,
    "losses": 0,
    "elo": 58,
    "hoursPlayed": 120,
    "teamId": "76e7e1bd-628e-47b2-ac92-157c330669ef",
    "ratingAdjustment": 50
  },
  {
    "id": "34e7a9e7-df16-4082-92d5-cadb8fc087e5",
    "nickname": "Player2",
    "wins": 1,
    "losses": 1,
    "elo": 10,
    "hoursPlayed": 120,
    "teamId": "76e7e1bd-628e-47b2-ac92-157c330669ef",
    "ratingAdjustment": 50
  },
  {
    "id": "64f6186b-3a39-47b9-9313-3fcb8e4f06fd",
    "nickname": "Player3",
    "wins": 1,
    "losses": 1,
    "elo": 6,
    "hoursPlayed": 120,
    "teamId": "76e7e1bd-628e-47b2-ac92-157c330669ef",
```

```
    "ratingAdjustment": 50
  },
  {
    "id": "187b683c-1255-46a6-a709-60f8af0fd200",
    "nickname": "Player4",
    "wins": 0,
    "losses": 2,
    "elo": -41,
    "hoursPlayed": 120,
    "teamId": "76e7e1bd-628e-47b2-ac92-157c330669ef",
    "ratingAdjustment": 50
  },
  {
    "id": "f8987880-8869-472c-8335-83f60132b15d",
    "nickname": "Player5",
    "wins": 1,
    "losses": 1,
    "elo": 8,
    "hoursPlayed": 120,
    "teamId": null,
    "ratingAdjustment": 50
  },
  {
    "id": "2928b9d0-3a24-4919-99ae-c91d038b8063",
    "nickname": "Player6",
    "wins": 2,
    "losses": 0,
    "elo": 53,
    "hoursPlayed": 120,
    "teamId": null,
    "ratingAdjustment": 50
  },
  {
    "id": "912c9f8f-72e1-43f7-9186-0eb9430456a0",
    "nickname": "Player7",
    "wins": 0,
    "losses": 1,
    "elo": -21,
    "hoursPlayed": 60,
    "teamId": null,
    "ratingAdjustment": 50
  },
  {
    "id": "9db725b4-df4c-4e87-9e64-3f02c9bdf033",
    "nickname": "Player8",
    "wins": 1,
    "losses": 0,
    "elo": 28,
```

```
    "hoursPlayed": 60,
    "teamId": null,
    "ratingAdjustment": 50
  },
  {
    "id": "8386445c-1860-42cb-99bd-6383088255a6",
    "nickname": "Player9",
    "wins": 0,
    "losses": 1,
    "elo": -23,
    "hoursPlayed": 60,
    "teamId": null,
    "ratingAdjustment": 50
  },
  {
    "id": "e80e11bb-4c77-46e6-92a5-e55b9a6889b6",
    "nickname": "Player10",
    "wins": 0,
    "losses": 1,
    "elo": -24,
    "hoursPlayed": 60,
    "teamId": null,
    "ratingAdjustment": 50
  }
]
```


- Dodati validaciju prilikom pokretanja (beleženja) igre da oba tima moraju imati jednaki broj igrača i da je taj broj veći od 0
- Dodati endpoint za napuštanje tima za igrača
    - Ukoliko igrač nema trenutno tim servis treba da ima isto (idempotentno) ponašanje kao kada stvarno napušta tim

PUT /players/{player_id}/leave_team

Response:


```
{
  "id": "b769b730-d1b9-4a94-8d2d-2936e050722c",
  "nickname": "Player1",
  "wins": 2,
  "losses": 0,
  "elo": 58,
  "hoursPlayed": 120,
  "teamId": null,
  "ratingAdjustment": 50
}
```

- Dodati endpoint za matchmaking
  - Ovaj endpoint treba da kreira dva tima od N igrača
  - N se dobija kao queru parameter
  - Nakon zabeleženog meča igrači napuštaju ove timove
  - Za kreiranje timova uzima se prvih 2*N igrača iz baze koji nemaju tim, sortirano po elo igrača opadajuće
  - Kako bi timovi bili slične snage prvi tim dobija prvog (index 0) i poslednjeg (index 2 * N – 1), zatim drugi tim dobija drugog (index 1) i pretposlendjeg (index 2 * N – 2) i tako dalje. Ukoliko je N neparan prvi tim dobija N-tog igrača (index N-1) a drugi tim N+1-tog igrača (index N). Ukoliko je N paran broj nastavlja se osnovna logika dodele igrača pa drugi tim dobija i N-tog (index N – 1) i N+1-tog (index N) igrača
  - Dodati validaciju da ima 2*N igrača bez tima u bazi
  - Dodati validaciju da je N > 0

POST /teams/generate_teams?teamSize=N

Response:

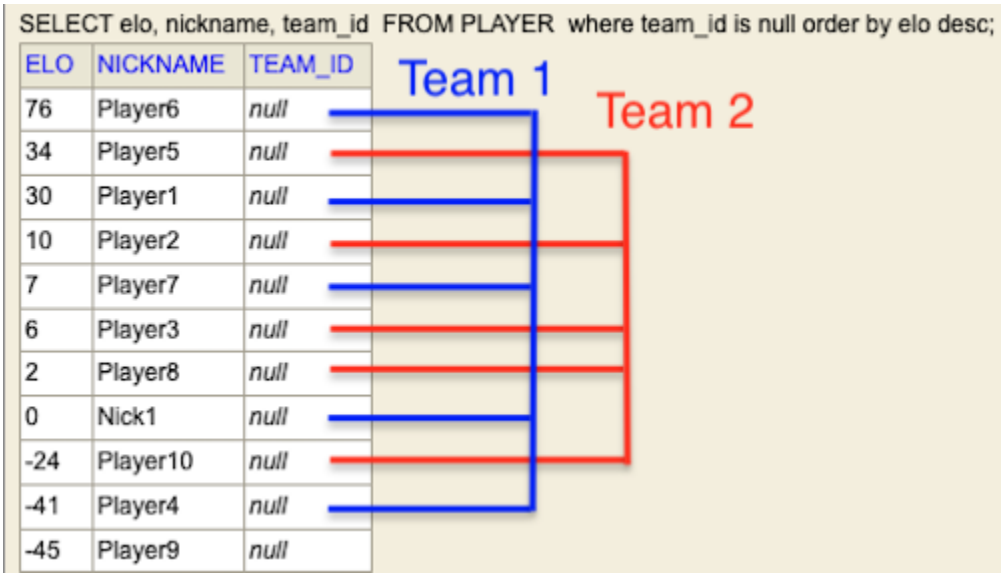

```
[
  {
    "id": "7e936b67-9fa1-45be-9940-e9c07bb8d3f8",
    "teamName": "470c3b08-8168-4a14-a72b-c5944f70456a",
    "players": [
      {
        "id": "b769b730-d1b9-4a94-8d2d-2936e050722c",
        "nickname": "Player1",
        "wins": 2,
        "losses": 0,
        "elo": 58,
        "hoursPlayed": 120,
        "teamId": "7e936b67-9fa1-45be-9940-e9c07bb8d3f8",
        "ratingAdjustment": 50
      },
      {
        "id": "8386445c-1860-42cb-99bd-6383088255a6",
        "nickname": "Player9",
        "wins": 0,
        "losses": 1,
        "elo": -23,
        "hoursPlayed": 60,
        "teamId": "7e936b67-9fa1-45be-9940-e9c07bb8d3f8",
        "ratingAdjustment": 50
      },
```

```
      {
        "id": "9db725b4-df4c-4e87-9e64-3f02c9bdf033",
        "nickname": "Player8",
        "wins": 1,
        "losses": 0,
        "elo": 28,
        "hoursPlayed": 60,
        "teamId": "7e936b67-9fa1-45be-9940-e9c07bb8d3f8",
        "ratingAdjustment": 50
      }
    ]
  },
  {
    "id": "ceb11457-018b-4bec-b943-46d007766664",
    "teamName": "78919ee6-9f54-4646-b21e-e03732086590",
    "players": [
      {
        "id": "2928b9d0-3a24-4919-99ae-c91d038b8063",
        "nickname": "Player6",
        "wins": 2,
        "losses": 0,
        "elo": 53,
        "hoursPlayed": 120,
        "teamId": "ceb11457-018b-4bec-b943-46d007766664",
        "ratingAdjustment": 50
      },
      {
        "id": "912c9f8f-72e1-43f7-9186-0eb9430456a0",
        "nickname": "Player7",
        "wins": 0,
        "losses": 1,
        "elo": -21,
        "hoursPlayed": 60,
        "teamId": "ceb11457-018b-4bec-b943-46d007766664",
        "ratingAdjustment": 50
      },
      {
        "id": "f8987880-8869-472c-8335-83f60132b15d",
        "nickname": "Player5",
        "wins": 1,
        "losses": 1,
        "elo": 8,
        "hoursPlayed": 120,
        "teamId": "ceb11457-018b-4bec-b943-46d007766664",
        "ratingAdjustment": 50
      }
    ]
  }
```

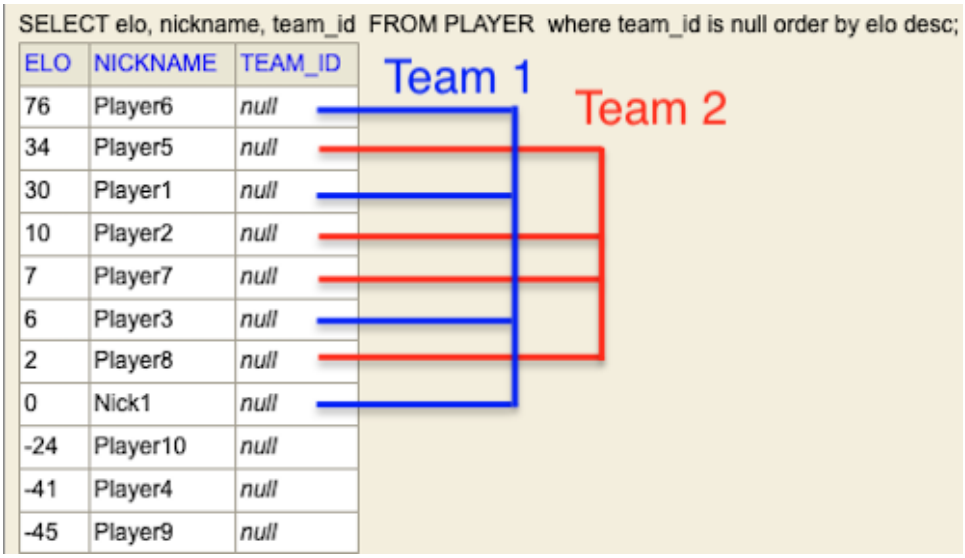
```
]
```

Ilustracija logike za matchmaking endpoint:

Za N = 5 timovi bi izgledali:



Za N = 4 timovi bi izgledali:



Dodatne napomene:

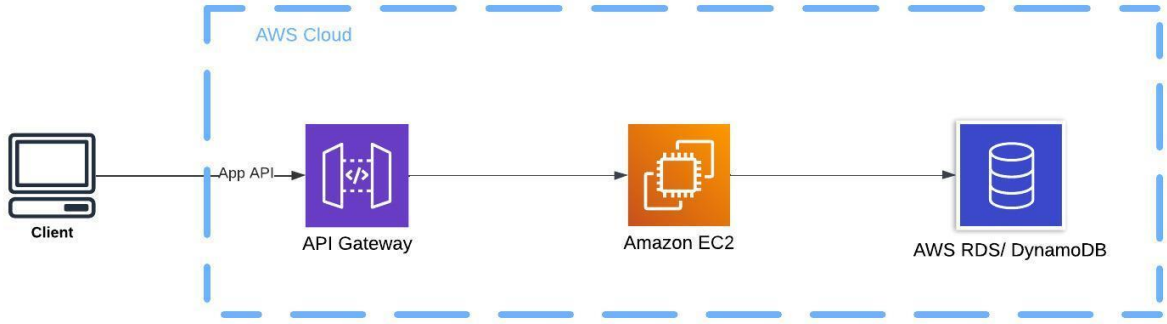
- Sve uspešne operacije treba da vrate HTTP response code 200
- Vrednosti elo, duration, wins, losses, hoursPlayed, ratingAdjustment su celi brojevi

Bonus funkcionalnost:

- Nakon odigranog meča potrebno je poslati email na adresu xxxx@xxx.xxx sa podacima o odigranom meču. Svaki tim će dobiti na koju email adresu treba da se šalju podaci o odigranom meču.
- Nakon odigranog meča smestiti rezultat meča u vidu .json file-a na s3.

Deploy aplikacija na AWS

Omogućiti da se rešenje iz Challenge faze izvršava na Cloud-u sa osnovnim funkcionalnim zahtevom. 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:

- Podići API Gateway i konfigurisati endpointe da bi klient mogao da komunicira sa aplikacijom
- Podići odgovarajuću bazu (RDS ili NoSQL)
- Podići EC2 instancu, odraditi deploy aplikacije na instancu i podesiti da aplikacija koristi bazu koja je podignuta na AWSu. Podatke o cenama za svaki tip EC2 instance možete naći na sledećem linku: https://aws.amazon.com/ec2/pricing/on-demand (za region izabrati Frankfurt)

Dodatni funkcionalni zahtev

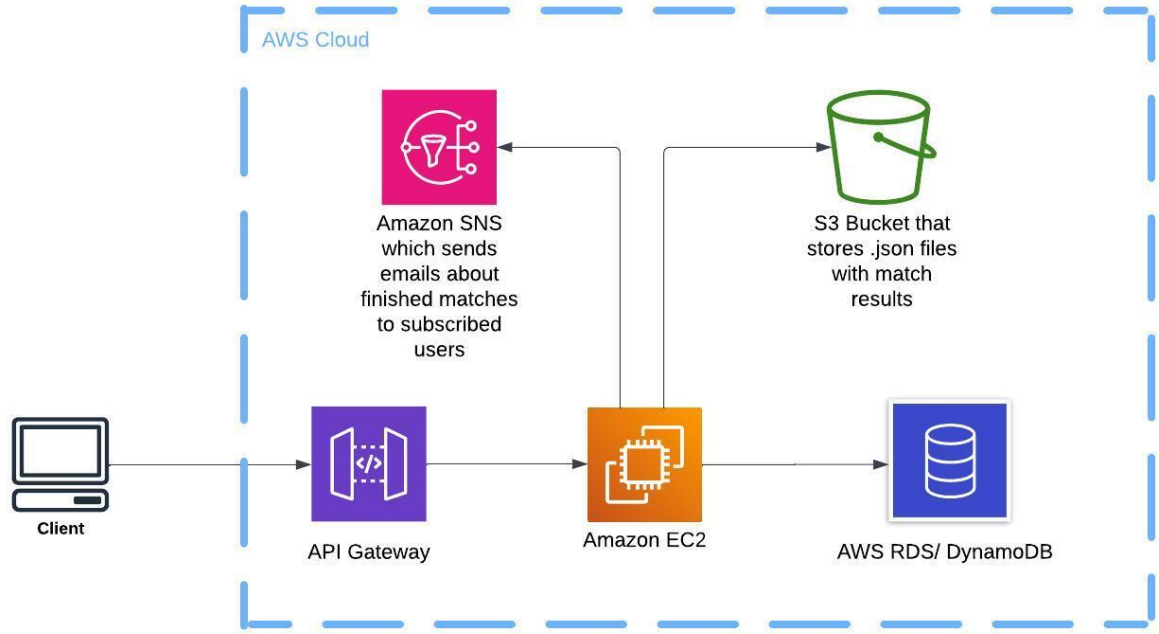
Pod dodatnim funkcionalnim zahtevom se smatra slanje odigračnog meča na email i čuvanje rezultata odigranog meča u vidu .json fajla na S3 bucket preko lambdi.

.json fajl koji se šalje na SNS i čuva u S3 predstavlja request za pokretanje meča. Primer:

```
{
   "team1Id" : "7e936b67-9fa1-45be-9940-e9c07bb8d3f8",
   "team2Id" : "ceb11457-018b-4bec-b943-46d007766664",
   "winningTeamId" : "ceb11457-018b-4bec-b943-46d007766664",
   "duration" : 60
 }
```

Fajl koji se čuva treba da ima jedinstven naziv.

Arhitektura sa dodatnim funkcionalnim zahtevom se nalazi na sledećoj slici:



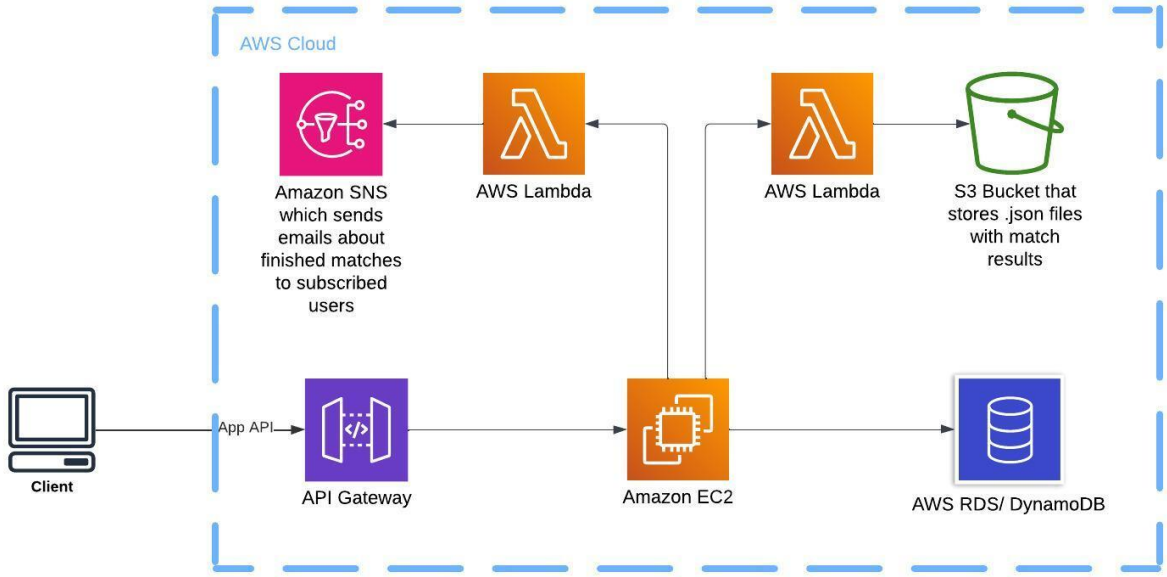
Koraci za implementaciju dodatnog funkcionalnog zahteva

- Kreirati S3 bucket i podesiti da aplikacija čuva .JSON rezultat meča
- Kreirati SNS topic i odraditi subscribe na email adresu radi slanja emaila i podesiti da aplikacije šalje notifikaciju na SNS topic

Tehničko unapređenje

Pod tehničkim unapređem se smatra dodatno kreiranje lambdi koje aplikacija okida za slanje notifikacije na SNS topic i smeštanje .json reporta o odigranom meču na S3 bucket.

Arhitektura:



Koraci za implementaciju tehničkog unapređenja:

- Napraviti lambdu za slanje notifikacije na SNS topic
- Napraviti lambdu za smeštanje fajla u S3 bucket
- Podešavanje aplikacije da umesto direktnog slanja na SNS topic i čuvanja na S3 bucket okida napravljene lambde

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
    - Slanje emaila preko SNSa - 5 bodova
    - Čuvanje json fajla na S3 bucket - 5 bodova
- Maksimalno 10 bodova za tehničko unapređenje
    - Kreiranje lambdi
        - Za slanje emaila preko SNSa - 5 bodova
        - Čuvanje json fajla na S3 bucket - 5 bodova
- Maksimalno 10 bodova za kvalitet rešenja
    - U obzir će se uzimati stvari poput code quality, infrastructure as code (CDK)
- Maksimalno 10 bodova za prezentaciju

Prezentacija

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

Dodatne napomene:

- Trenutni plasman možete videti na Leaderboard-u.
- Ukoliko želite da detaljno vidite asertacije nakon izvršavanja kolekcije testova u Postman-u koju ćemo vam proslediti, pri pokretanju testova označite opciju “Persist responses for a session”.

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