5 DANA U OBLACIMA - HACKATHON

ZADATAK

Zadatak za Hackaton sastoji se od dva dela:

- Dodatne funkcionalne izmene na rešenje iz individualnog dela
- Izvršavanje rešenja na AWS Cloud-u sa kontejnerskim rešenjima (ECS/EKS)

FUNKCIONALNE IZMENE

1. Dodati novi domenski entitet User. Entitet User sadrži atribute: id, email, first name, last name.
2. Dodati endpoint za kreiranje User-a

Path	/user
Request method	POST

Request body	<pre>{
	  "id": 1,
	  "email": "test@test.com",
	  "firstName": "Test",
	  "lastName": "Test"
	}</pre>

Response status code	201 – User uspešno kreiran

Response body	<pre>{
	  "id": 1,
	  "email": "test@test.com",
	  "firstName": "Test",
	  "lastName": "Test"
	}</pre>

3. Izmeniti endpoint za reset-ovanje stanja, tako da se pored order i trade rekorda obrišu i svi user rekordi.
4. Proširiti entitet Order sa referencom na User-ov id (userId).

5. Nakon što se order 100% realizuje i pređe u CLOSED status, potrebno je User-u poslati email sa sledećim sadržajem:

   "Your ${orderType} order ${orderId} is completely realized."

6. Podrška za više različitih valuta
   a. Pored BTCUSD, od sada platforma podržava i ETHUSD. Lista podržanih valuta može biti fiksirana u vašoj aplikaciji.
   b. Endpoint za Orderbook od sada zahteva i path varijablu currencyPair, tako da će path sada izgledati */orderbook/BTCUSD* ili */orderbook/ETHUSD*

IZVRŠAVANJE REŠENJA NA AWS CLOUD-U

Omogućiti da se rešenje individualnog dela takmičenja izvršava na AWS Cloud-u uz primenu kontejner servisa kao što su ECS (Elastic Container Service) ili EKS (Elastic Kubernetes Service).

Zahtevi:

1. Potrebno je Docker-izovati aplikaciju i deploy-ovati Docker image na AWS ECR (Elastic Container Registry).
2. Kreirati ECS/EKS cluster i kreirati odgovarajuće resurse neophodne za startovanje aplikacije u okviru cluster-a: VPC, Internet Gateway, Subnet-i, ...
3. Endpoint-i implementirani u Challenge fazi su dostupni preko interneta. Da biste ovo omogućili potrebno je:
   a. Kreirati i konfigurisati Load Balancer koji će prihvatati request-e i prosleđivati ih vašoj aplikaciji na ECS/EKS cluster-u.
   b. Kreirati i konfigurisati API Gateway (REST API koji možete kreirati import-ovanjem Open API specifikacije navedene u dokumentu). Request-e koji dolaze na API Gateway, potrebno je proslediti na URL od Load Balancer-a. Takođe, za Orderbook endpoint, potrebno je enable-vati CORS (odnosno dodati OPTIONS metodu).
4. Sve podatke o Order-ima ili Trade-ovima potrebno je sačuvati u nekoj od AWS baza podataka kao što su RDS/Aurora (relacione) ili npr DynamoDb (NoSQL).

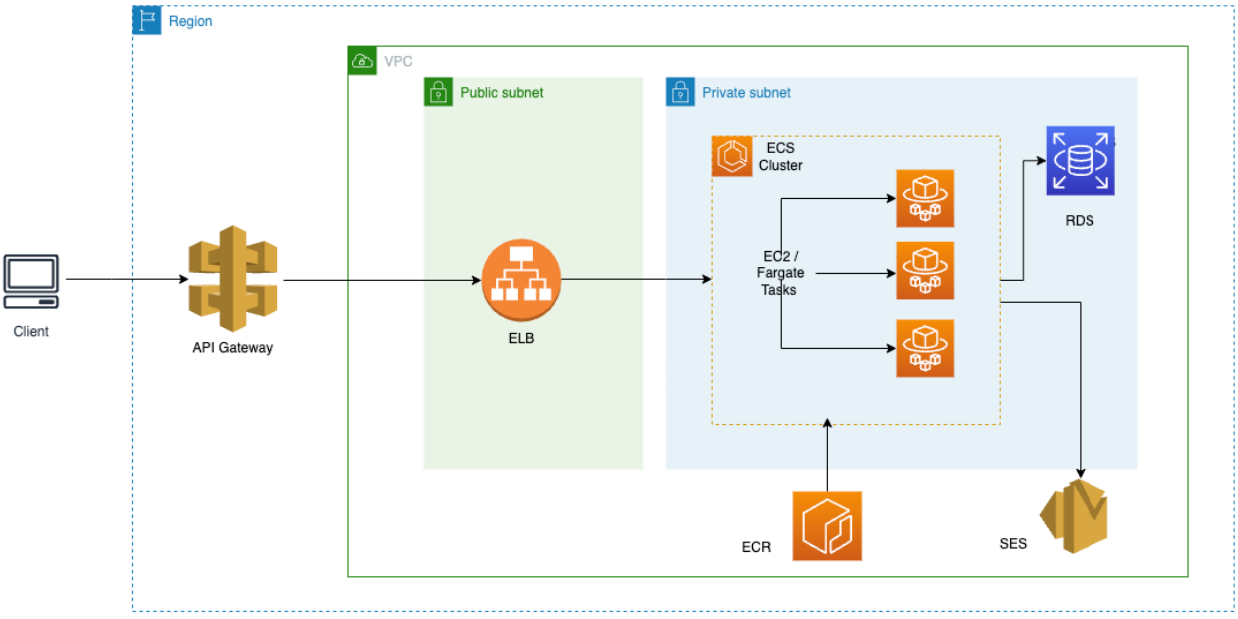
5. Email notifikaciju nakon kompletne realizacije order-a možete implementirati uz pomoć AWS SES (Simple Email Service) servisa.
6. Autoscale-ing - potrebno je podesiti autoscaling tako da u slučaju da CPU usage pređe 70% scale-out se izvrši, odnosno novi kontejner (task/pod) bude startovan.

BONUS:

- Infrastructure as a code
  Potrebno je sve resurse izvući u kod, kako bi se lakše moglo upravljati izmenama na resursima kao i budućoj automatizaciji i Continuous Delivery (CD) procesima.
  Možete iskoristiti AWS CDK da biste ovo ostvarili. CDK fajlove možete dodati na vaš GitHub projekat.

OpenAPI specifikacija koju Hackathon rešenje treba da implementira se nalazi na: https://5days-assets.s3.eu-west-1.amazonaws.com/openapi.json
Na osnovu ove specifikacije možete generisati API Gateway REST API. Možete je vizualizovati koristeći, na primer, https://editor.swagger.io/

PRIMER ARHITEKTURE REŠENJA



Napomena: prikazano rešenje je samo jedno od velikog broja mogućih rešenja. Rešenje koje Hackathon timovi budu kreirali može, a ne mora, da bude istovetno njemu. Timovi mogu, na primer, da koriste druge AWS servise ili da drugačije rasporede ili konfigurišu servise prikazane na dijagramu iznad.

NAPOMENE

- Svaki od Hackathon timova dobija pristup zasebnom AWS Cloud nalogu sa dostupnih 50 Dolara.
- Vaše AWS rešenje treba da se izvršava u EU-West-1 (Ireland) regiji.
- Leaderboard se nalazi na: https://dkfehfze6xvua.cloudfront.net

KORISNA DOKUMENTACIJA

- Docker, How to containerize your app https://www.baeldung.com/dockerizing-spring-boot-application#Dockerize
- Elastic Container Service - https://aws.amazon.com/ecs/faqs/
- Elastic Kubernetes Service - https://aws.amazon.com/eks/faqs/
- API Gateway https://aws.amazon.com/api-gateway/faqs/
- Load Balancers https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/what-is-load-balancing.html
- SES https://aws.amazon.com/ses/faqs/
- RDS https://aws.amazon.com/rds/faqs/
- RDS Aurora https://aws.amazon.com/rds/aurora/faqs/
- Dynamo DB https://aws.amazon.com/dynamodb/faqs/

PRAVILA PREZENTACIJE REŠENJA

- Prezentacija mora biti u Powerpoint formatu
- Trajanje prezentacije je ograničeno na 15 minuta
- Broj prezentera nije ograničen

PRAVILA BODOVANJA

- Kvalitet rešenja i koda – **10 poena**
- Arhitektura celokupnog Cloud rešenja – **40 poena**
    - Diverzitet korišćenih AWS komponenti – **20 poena**
    - AWS Security & Networking - **15 poena**
    - Autoscaling - **5 poena**
- Prezentacija rešenja – **10 poena**
- Test scenarija **30 poena**
    - Bodovanje je u nastavku, ukupno 14 TC (Test Case-ova)
- Bonus deo **10 poena**

Bodovanje testova

Validacioni testovi (**ukupno 6 poena**):

- Order is not created if price is negative (**1.5**)
- Order is not created if quantity is negative (**1.5**)
- Order is not created if type is not SELL or BUY (**1.5**)
- User is not created if email is not correct (**1.5**)

Funkcionalni testovi (ukupno 24 poena):

- Osnovni testovi (**ukupno 8 poena**)
    - 2 trades are created and BUY order is in status CLOSED when BUY order finds SELL order with less price and more amount **(4)**
    - Trade is created and email is sent to user when BUY order finds SELL order with same price and amount (**4**)

- Napredni testovi (**ukupno 16**)
    - Trade is created and BUY order is in status OPEN when BUY order finds SELL order with same price and less amount (**2**)
    - Trade is created and both orders are in status CLOSED when BUY order finds SELL order with same price and same amount (**2**)
    - Trade is created when BUY order finds SELL order with less price and with same amount (**2**)
    - Trade is not created when BUY order cannot finds SELL order with same or less price and same or less amount (**2**)

- Trade is created and SELL order is in status OPEN when SELL order finds BUY order with higher price and less amount (**2**)
- 2 trades are created and SELL order is in status CLOSED when SELL order finds BUY order with higher price and more amount (**2**)
- Trade is not created when BUY order finds SELL order with same price and amount but with different currency (**2**)
- Trade is created for ETHUSD and both orders are in status CLOSED when BUY order finds SELL order with same price and same amount (**2**)