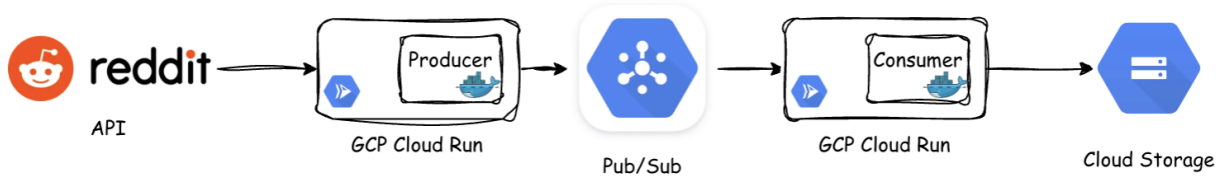
1. laboratorijska vježba

Uvod

Tijekom laboratorijskih vježbi stvorit ćete svoju **platformu za unos i obradu podataka** na Google Cloud Platformi (GCP). Cilj ove vježbe je kreirati **Data Ingestion Pipeline** koji omogućuje dohvat podataka s Reddit API-ja, njihovu distribuciju kroz **Pub/Sub topic** te konzumiranje podataka pomoću consumer skripte.

Vaše aplikacije bit će dockerizirane i deployane na **Cloud Run**, što omogućuje skalabilan deployment u oblaku.

Arhitektura pipeline-a u ovoj vježbi prikazana je na slici 1:



Slika 1. Arhitektura Data Ingestion Pipeline-a: Reddit API - Python Producer - Pub/Sub - Python Consumer - Console log

Glavni ciljevi vježbe:

- Razumjeti osnovne koncepte, **GCP** i njegove servise - posebno **Pub/Sub**.
- Napraviti skripte za **Producer** i **Consumer**.
- Dockerizirati aplikacije i deployati ih na GCP **Cloud Run**.
- Testirati pipeline i analizirati ponašanje poruka u Pub/Sub-u.

> **Napomena:** Kod možete pisati u bilo kojem jeziku/programskom okruženju, ali preporučamo **Python**.

Google Cloud Platform

Google Cloud Platform (GCP) je skup cloud servisa i alata koji omogućuju izgradnju, implementaciju i upravljanje aplikacijama, infrastrukturom i podacima u oblaku. Glavni servisi koje ćemo koristiti:

- Pub/Sub - distribuirani sustav za slanje i primanje poruka. (**Message Broker**)
- Cloud Run - platforma za deploy kontejneriziranih aplikacija.

Glavni servisi i pojmovi na GCP-u

Servis / pojam	Opis	Primjer uporabe
Compute Engine	Virtualne mašine za širok spektar uporabe.	Pokretanje custom server aplikacije.
Cloud Storage	Objektna pohrana podataka (JSON, CSV, slike).	Pohrana raw podataka s Reddit API-ja ili logova.
BigQuery	Analitička platforma (DWH - Data Warehouse) za velike podatke, SQL upiti nad TB/PB podataka.	Analiza sakupljenih podataka iz Pub/Sub-a.
Pub/Sub	Distribuirani servis za razmjenu poruka (event-driven arhitektura).	Kroz njega šaljemo poruke od Producer-a do Consumer-a.
Cloud Run	Deploy kontejneriziranih aplikacija u managed okruženju.	Deployment Python producer/consumer aplikacija.
Dataflow	Obrada podataka u streamu ili batch-u (temeljen na Apache Beam).	Napredna obrada podataka iz Pub/Sub-a.
Dataproc	Managed Hadoop i Spark clusteri.	Big Data batch obrada (nije dio laba, ali korisno znati).
Cloud Functions	Serverless funkcije, triggere po događajima.	Mogu zamijeniti Consumer u jednostavnim scenarijima.
Cloud Scheduler	Cron-job servis za planiranje zadataka u oblaku.	Pokretanje periodičnih zadataka, npr. producer skripte koje se automatski izvršavaju svakih 5 sekundi.
IAM (Identity & Access Management)	Upravljanje pristupom i privilegijama korisnika i servisa.	Postavljanje prava za Pub/Sub topic i Cloud Run servis.

Kreiranje GCP projekta

Za rješavanje laboratorijskih vježbi bit će vam potreban Google Cloud Platform (GCP) projekt. Novi korisnici Google Clouda dobivaju $300 besplatnih kredita koji vrijede 90 dana. Ti krediti omogućuju besplatno korištenje mnogih usluga, uključujući Cloud Run, Compute Engine, BigQuery i druge.

Tijekom registracije potrebno je unijeti podatke o plaćanju, ali **ne brinite - dok ne aktivirate plaćeni račun, neće biti naplate**, a resursi koji će vam biti potrebni za laboratorijske vježbe u konačnici neće ni približno potrošiti vaš besplatni kredit.

Upute za otvaranje Free Trial-a možete pronaći ovdje: Google Cloud Free Trial

> **Napomena**: Ukoliko ste već potrošili free credite, javite nam se - imamo alternativu u tom slučaju. Samo vas molimo da nam se javite na vrijeme, ne zadnji dan (ako se ipak javite zadnji dan, ništa strašno, moći ćete na drugom labosu donijeti rješenja prva dva labosa.)

> **Napomena**: Kod kreiranja samog projekta te za svaki resurs koji ćete stvoriti potrebno je definirati regiju u kojoj se nalazi. **VAŽNO** je koristiti **istu regiju za SVE resurse** (Pub/Sub, Cloud Run, Artifact Registry) kako bi se smanjila latencija i troškovi prijenosa podataka. Za ovu vježbu predlažemo regiju: `europe-west1`.

Pub/Sub

Pub/Sub je asinkroni servis za razmjenu poruka na Google Cloud Platformi koji omogućuje pouzdanu komunikaciju između različitih aplikacija i servisa.

Ključni koncepti

- **Publisher**: šalje poruke u topic.
- **Subscriber**: prima poruke iz topic-a.
- **Topic**: kanal u koji publisher šalje poruke.
- **Subscription (Consumer grupa)**: određuje tko i kako prima poruke. Svaki subscription dobiva kopiju svih poruka poslanih u topic nakon što je subscription kreiran.
- **Message attributes**: opcionalni atributi za filtriranje ili grupiranje poruka, omogućuju različitim aplikacijama da neovisno konzumiraju iste podatke.

Consumer grupe u Pub/Sub-u

Svaki **subscription** predstavlja **zasebnu grupu**. Svaki subscription prima **sve poruke** objavljene nakon njezinog kreiranja. Kada postoji više subscription-a, svaki od njih čita **sve poruke neovisno o ostalima**, dok u slučaju jednog subscriptiona, **svaka poruka** se pročita **samo jednom** i **uklanja** iz topica nakon što je **acknowledged**. Poruke koje nisu acknowledged mogu se **ponovno** povući. Dodatno, moguće je koristiti atribut key za grupiranje poruka ili filtriranje po određenim kriterijima, što omogućuje različitim aplikacijama da neovisno konzumiraju iste podatke.

Primjer toka poruke:

- Producer šalje poruku u Pub/Sub topic.
- Svaki subscription povezan s tim topicom dobiva kopiju poruke.
    - Ako se više subscribera spoji na isti subscription, oni zajedno čine consumer grupu i poruke se raspoređuju među njima (svaki subscriber dobiva različite poruke).
    - Ako postoji više subscription-a za isti topic, svaki subscription dobiva vlastitu kopiju poruke.
- Subscriber obrađuje poruku i, kada je obrada uspješna, šalje ack (potvrdu).
- Ako obrada ne uspije ili subscriber ne pošalje ack, poruka se vraća u red i može se ponovno procesirati.

Primjer iz stvarnog života:

Zamislimo da u uredu postoji centralni poštanski sandučić u koji stižu sve pošiljke za različite odjele.

- Subscription A je osoba koja pregledava i sortira poštu za odjel financija.
- Subscription B je osoba koja pregledava i sortira poštu za odjel marketinga.

Obje osobe primaju sve obavijesti o dolasku pošte, jer svaka treba provjeriti koje pošiljke su relevantne za njen odjel. Ako bi postojao samo jedan pregledatelj pošte koji uklanja obavijesti, drugi odjeli ne bi saznali što je stiglo.

Da bi svaka osoba vidjela samo relevantne pošiljke, možemo koristiti etikete na pošti (npr. “financije” ili “marketing”) kao ključ (key). Na taj način, obje osobe dobivaju sve poruke, ali svaka može filtrirati samo one koje su joj potrebne, dok cijeli sustav i dalje prima sve informacije.

Upute za postavljanje okoline

Pristup postojećem GCP projektu

Prijava i autentifikacija

- Otvorite Google Cloud Console
- Prijavite se svojim Google računom

Instalacija i postavljanje gcloud CLI

- Instalirajte gcloud CLI:
  - Windows: putem ovog linka
  - macOS:

```
curl https://sdk.cloud.google.com | bash
exec -l $SHELL
gcloud init
```

  - Linux:

```
sudo apt update && sudo apt install apt-transport-https ca-
certificates gnupg -y
echo "deb [signed-by=/usr/share/keyrings/cloud.google.gpg]
https://packages.cloud.google.com/apt cloud-sdk main" \
  | sudo tee /etc/apt/sources.list.d/google-cloud-sdk.list
curl https://packages.cloud.google.com/apt/doc/apt-key.gpg \
  | sudo gpg --dearmor -o /usr/share/keyrings/cloud.google.gpg
sudo apt update && sudo apt install google-cloud-cli -y
```

  i pokrenite Cloud Shell u gornjem desnom kutu sučelja vašeg GCP projekta.

> **Napomena:** Ako ne želite instalirati gcloud CLI lokalno, sve naredbe se mogu izvršavati i direktno u Cloud Shell konzoli u pregledniku. Time dobivate potpuno konfiguriran terminal povezan s vašim GCP projektom bez dodatne instalacije.

- Autentificirajte se i odaberite projekt:

```
gcloud auth login
gcloud config set project <PROJECT_ID>
```

- Zamijenite <PROJECT_ID> s ID-jem vašeg projekta

Provjera aktivnog projekta

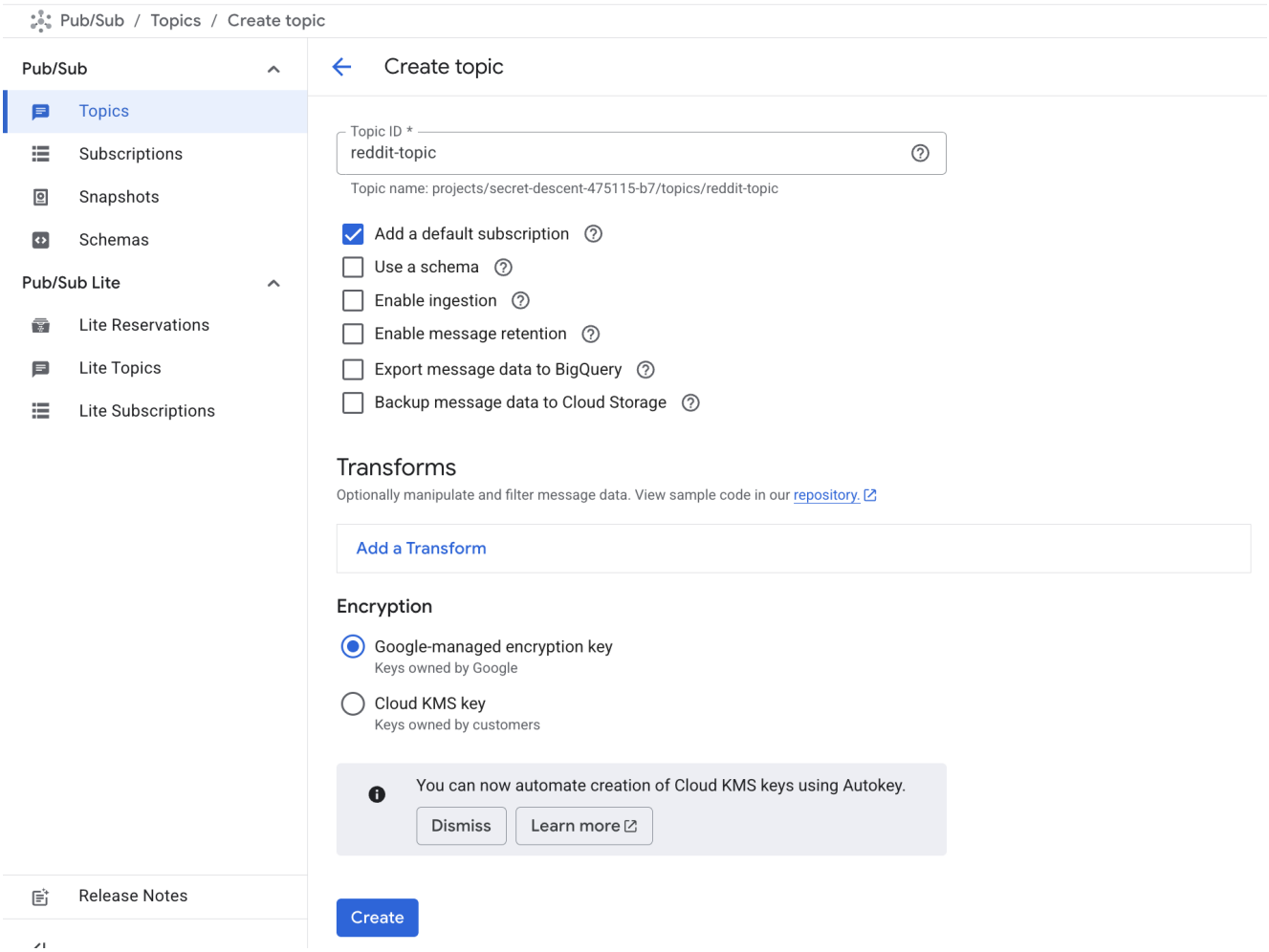
```
gcloud config list project
```

Kreiranje i provjera Pub/Sub resursa

Kreiranje topic-a:

```
gcloud pubsub topics create <TOPIC-NAME>
```

- Kako bi ispravno nazvali topic i ostale resurse pogledajte poglavlje naming conventions, te se podsjetite koju regiju odabrati (napomena s uputama nalazi se u poglavlju vezanom uz kreiranje GCP projekta)
- Topic možete kreirati i ručno unutar Google Cloud Console

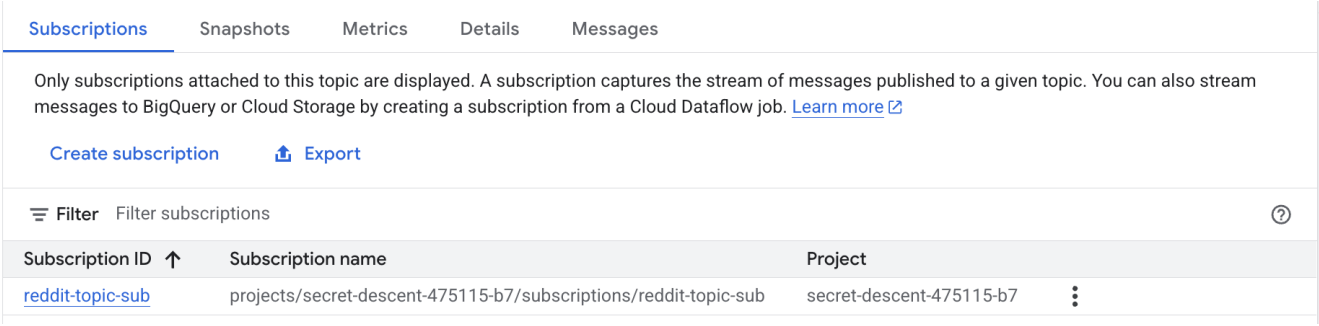


Slika 1. Topic

Kreiranje subscription-a:

```
gcloud pubsub subscriptions create <SUBSCRIPTION-NAME> --topic=<TOPIC-
NAME>
```

- Imenujte subscription ovako: TOPIC-NAME-sub
- Ukoliko ručno kreirate topic, GCP će za vam automatski kreirati i subscription na taj topic (možete kasnije dodati još subscriptiona)



Slika 2. Subscription

Postavljanje environment varijabli

```
export PROJECT_ID=<PROJECT_ID>
export PUBSUB_TOPIC=<TOPIC-NAME>
export PUBSUB_SUBSCRIPTION=<SUBSCRIPTION-NAME>
export GCP_REGION=europe-west1
```

- Environment varijable vašoj skripti i gcloud alatima govore koji projekt, topic, subscription i regija da se koriste, bez da te podatke hardkodirate u kod.
- Primjer korištenja:

```
import os

project_id = os.getenv("PROJECT_ID")
topic = os.getenv("PUBSUB_TOPIC")
subscription = os.getenv("PUBSUB_SUBSCRIPTION")
region = os.getenv("GCP_REGION")
```

```
gcloud pubsub topics publish $PUBSUB_TOPIC --message="Message"
```

- U sljedećim primjerima nećemo koristiti environment vrijable radi lakšeg razumijevanja

Testiranje Pub/Sub konekcije

Listanje topic-a

```
gcloud pubsub topics list
```

Listanje subscription-a:

```
gcloud pubsub subscriptions list
```

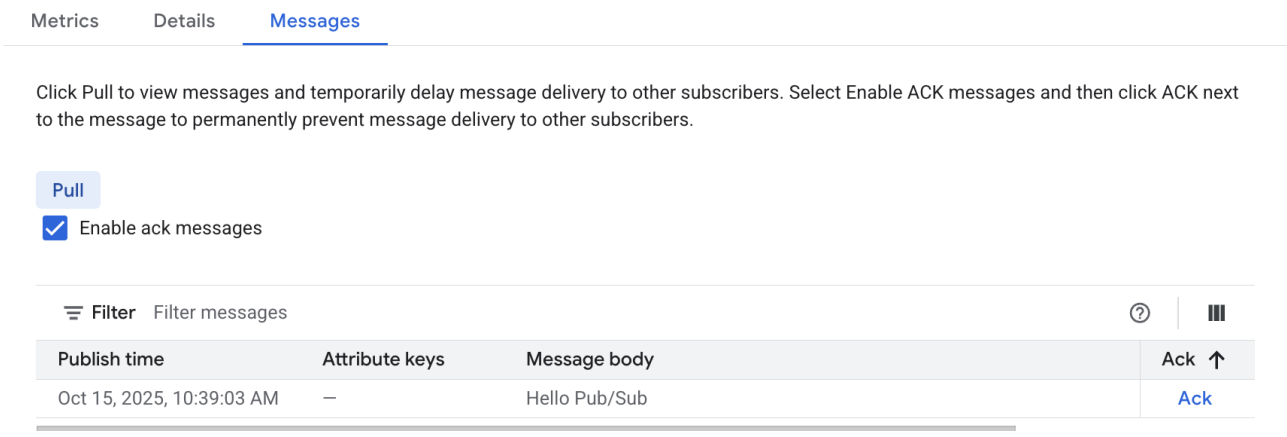
Slanje testne poruke:

```
gcloud pubsub topics publish <TOPIC-NAME> --message="Hello Pub/Sub"
```

Povlačenje poruke kroz subscription:

```
gcloud pubsub subscriptions pull <SUBSCRIPTION-NAME> --auto-ack
```

- Poruke možete povući i na GCP-u unutar vašeg subscriptiona za potrebe testiranja



Slika 3. Pull Messages

Imenovanje resursa

Best practices za nazivanje resursa

Dobar naming convention pomaže da se resursi lakše identificiraju i koriste:

- Koristite **opisne i konzistentne nazive**, npr. `reddit-producer`, `reddit-consumer`, `reddit-topic`.
- Za razdvajanje riječi koristite `-` (kebab-case) ili `_` (snake_case), ali budite konzistentni.

- U imenu uključite tip resursa ili funkcionalnost, npr. `cloudrun-producer-service`, `artifact-reddit-images`.
- Izbjegavajte znakove koji nisu podržani (razmaci, velika slova, specijalni znakovi).

Primjeri po resursima

Resurs	Primjer naziva (po studentu)	Standardno ime / konvencija GCP-a
GCP projekt	student-jmbag-projekt	[project-id]
Pub/Sub topic	reddit-topic-jmbag	[topic-name]
Pub/Sub subscription	reddit-subscription-jmbag	[subscription-name]
Cloud Storage bucket	reddit-bucket-jmbag	[bucket-name]
Cloud Run / App Engine service	reddit-producer-jmbag	[service-name]

API - aplikacijsko programsko sučelje

Aplikacijska programska sučelja (**Application Programming Interface or API**) standardni su način razmjene podataka između sustava. Klijentski programi koriste aplikacijska programska sučelja za komunikaciju s web uslugama. Općenito govoreći, API izlaže skup podataka i funkcija kako bi olakšao interakciju između računalnih programa i omogućio im razmjenu informacija. Web API je sučelje web usluge koje izravno sluša i odgovara na zahtjeve klijenata.

Reddit i Reddit API

Reddit je internetska platforma na kojoj korisnici objavljuju, komentiraju i pregledavaju sadržaje na različitim tematskim *subredditima*. Svaki subreddit bavi se određenom temom, a korisnici mogu objavljivati tekstualne i vizualne sadržaje te sudjelovati u raspravama.

Putem Reddit API-ja, moguće je pristupiti različitim funkcionalnostima platforme, kao što su čitanje i pisanje postova, komentiranje, upravljanje korisničkim računima te dobivanje informacija o sadržaju i korisnicima. Pruža programski način za automatizaciju interakcija s Redditom, što je korisno za izradu aplikacija, alata za analizu sadržaja, praćenje trendova i mnoge druge svrhe.

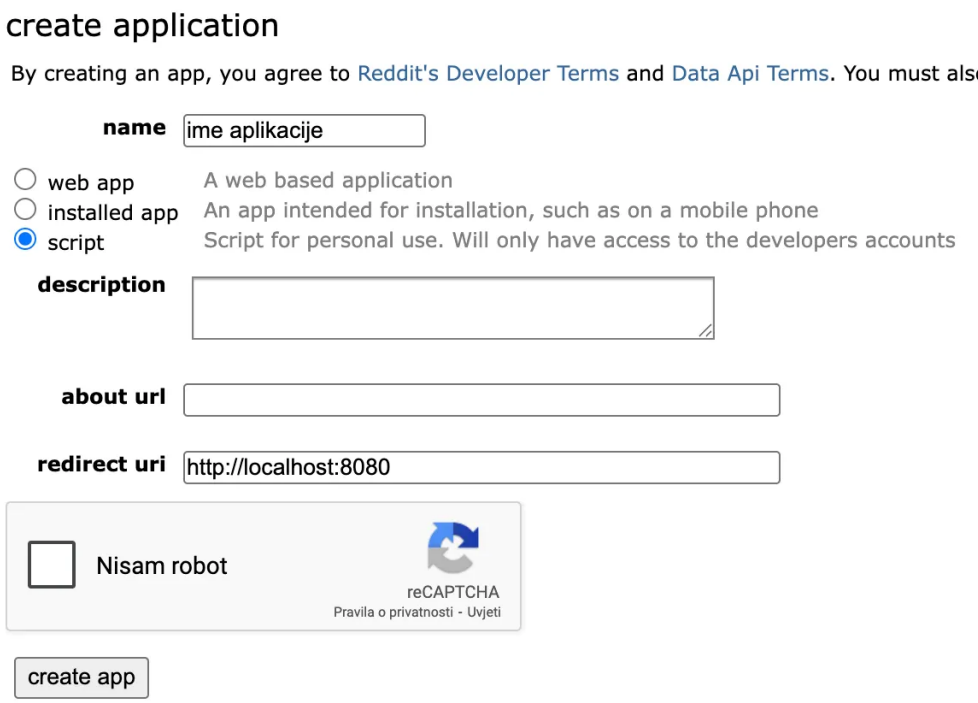
Tijekom vještine, Reddit API služit će kao izvor podataka koje ćete obrađivati kroz GCP.

Reddit API upute

Za korištenje Reddit-ovog API-ja potrebna je registracija jer Reddit ograničava direktne zahtjeve prema svom API-ju iz sigurnosnih i anti-abuse razloga. Ako aplikacija nije registrirana, Reddit će blokirati ili ograničiti pozive, što znači da nećete moći dohvatiti podatke. Registracijom dobivate Client ID i Client Secret, koji identificiraju vašu aplikaciju i omogućuju autorizirani pristup podacima.

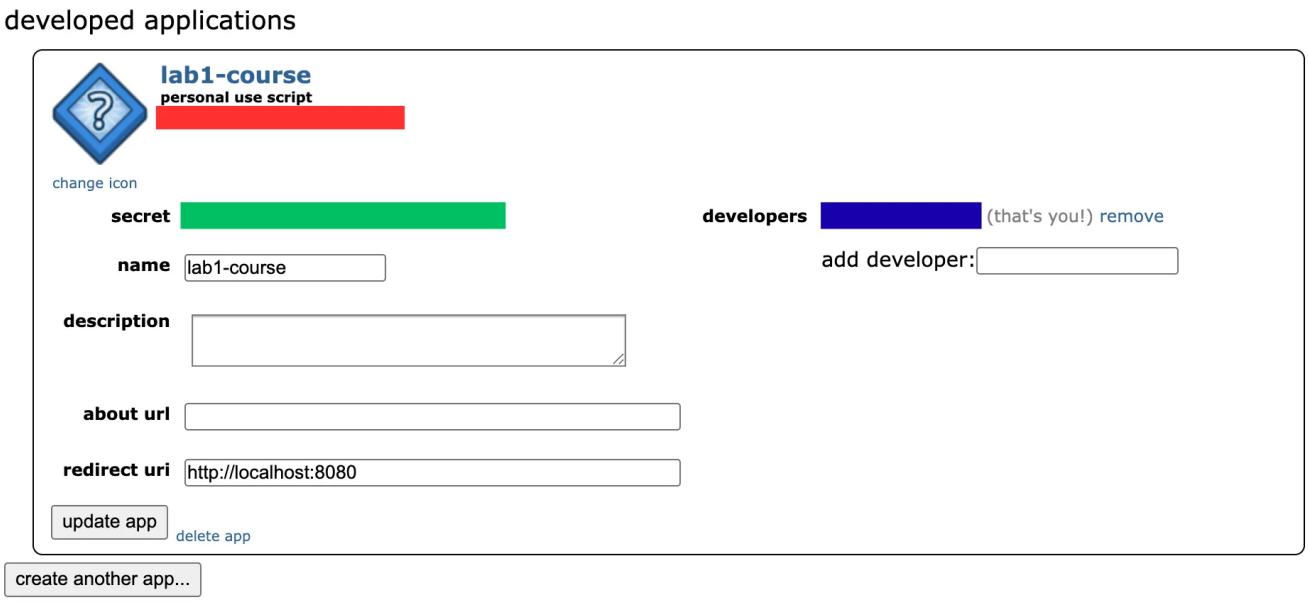
1. Posjetite https://www.reddit.com/prefs/apps
2. Kliknite "Create App" ili "Create Another App"
3. Popunite formular:

- **Name**: Ime vaše aplikacije
- **App type**: Odaberite "script"
- **Redirect uri**: http://localhost:8080 (za development)



Nakon kreiranja aplikacije, dobit ćete:

- **Client ID** (ispod "personal use script" crvena boja)
- **Client Secret** (pored "secret" - zelena boja)
- **Vaš Reddit username** (pored “developers” - plava boja)



Client ID, Client Secret i Reddit username pohranit ćete u Google Cloud Secret Manager servis, što će detaljnije biti objašnjeno u sljedećem poglavlju.

Dockerizacija aplikacija

Docker Desktop

Potrebno je instalirati određene alate kako biste lakše izvršavali ovu laboratorijsku vježbu:

1. **Docker Desktop** - ako ga već nemate, možete ga preuzeti na sljedećem linku: https://www.docker.com/products/docker-desktop
2. Nakon instalacije Docker Desktop-a, provjerite da je Docker pokrenut i da imate pristup terminalu za izvršavanje Docker komandi.

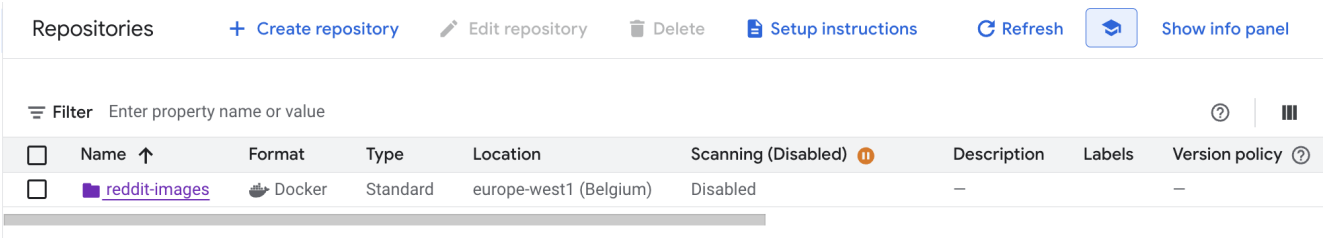
U ovoj laboratorijskoj vježbi ćete:

- koristiti Docker Desktop za buildanje i testiranje Docker image-a producera i consumera
- pohraniti Docker image-e u GCP **Artifact Registry**
- pripremiti image-e za deployment na **Cloud Run**

Artifact Registry

Nakon što ste izgradili Docker image, potrebno ga je pohraniti u GCP unutar Artifact Registry-a. Artifact Registry je upravljana usluga za pohranu i verzioniranje Docker image-a i drugih paketa, što omogućuje jednostavno dijeljenje, distribuciju i kontrolu pristupa. Pohranjivanjem image-a u Artifact Registry olakšava se deploy na Cloud Run, jer se Cloud Run direktno povezuje s registriranim image-ima i automatski ih povlači prilikom pokretanja servisa.

- Otvorite Google Cloud Console i odaberite svoj projekt.
- U lijevom izborniku odaberite Artifact Registry -> Repositories -> Create Repository.
- Unesite sljedeće podatke:
    - Name: npr. reddit-images
    - Format: Docker
    - Location: europe-west1 (ili regija vašeg projekta)
- Kliknite Create.



Slika 4. Kreiran Artifact Registry

- Nakon što je repozitorij kreiran, Docker image možete tagirati i pushati u Artifact Registry (Docker na temelju taga prepoznaje odredište tj. URL repozitorija u Artifact Registryju i automatski zna kamo treba poslati image):

1. za **producera**

```
docker tag <LOCAL_PRODUCER_IMAGE> <REGION>-
docker.pkg.dev/<PROJECT_ID>/<REPOSITORY>/<PRODUCER_IMAGE>:<TAG>
docker push <REGION>-
docker.pkg.dev/<PROJECT_ID>/<REPOSITORY>/<PRODUCER_IMAGE>:<TAG>
```

- Npr. docker tag reddit-producer europe-west1-docker.pkg.dev/<PROJECT_ID>/reddit-images/reddit-producer:latest i docker push europe-west1-docker.pkg.dev/<PROJECT_ID>/reddit-images/reddit-consumer:latest

2. za **consumera**

```
docker tag <LOCAL_CONSUMER_IMAGE> <REGION>-
docker.pkg.dev/<PROJECT_ID>/<REPOSITORY>/<CONSUMER_IMAGE>:<TAG>
docker push <REGION>-
docker.pkg.dev/<PROJECT_ID>/<REPOSITORY>/<CONSUMER_IMAGE>:<TAG>
```

Cloud Run

Cloud Run je fully managed serverless servis na Google Cloud Platformi (GCP) namijenjen za deploy kontejneriziranih aplikacija. Omogućuje pokretanje Docker imaga u potpuno upravljanom okruženju, što znači da se korisnici ne moraju brinuti o konfiguraciji virtualnih mašina ili infrastrukturnom održavanju. Cloud Run automatski skalira aplikacije prema prometu, od nule do tisuća instanci, što ga čini idealnim za aplikacije s varijabilnim opterećenjem ili mikroservisnu arhitekturu.

U ovom zadatku ćemo Cloud Run koristiti za dockeriziranje naše aplikacije, što znači da ćemo aplikaciju pakirati u Docker container i deployati je na Cloud Run. Time osiguravamo konzistentno okruženje za izvršavanje i jednostavan, brz proces deploya.

Svaki Cloud Run servis dobiva HTTPS endpoint i podržava različite programske jezike i frameworke. Također omogućuje povezivanje s drugim GCP servisima, sigurnu konfiguraciju i naplatu samo za resurse koje aplikacija koristi, što ga čini fleksibilnim i ekonomičnim rješenjem.

Prije nego što krenemo s kreiranjem Cloud Run servisa i Job-a, za pokretanje istih potreban nam je Service Account. **Service Account** je posebni nalog koji koriste servisi umjesto korisnika. Nećemo ulaziti u detalje, samo ćemo ga koristiti za deploy naših Cloud Run resursa.

Kreirajte Service Account te mu dodjelite rolu:

```
gcloud iam service-accounts create cr-deploy-sa \
    --description="Service Account for Cloud Run Service and Job
deployment" \
    --display-name="CR Deploy SA"
```

```
gcloud projects add-iam-policy-binding PROJECT_ID \
    --member="serviceAccount:cr-deploy-
```

```
sa@PROJECT_ID.iam.gserviceaccount.com" \
    --role="roles/run.admin"
```

- PROJECT_ID zamijenite id-jem svog projekta

Za kreiranje Cloud Run servisa:

- U Google Cloud Console odaberite Cloud Run -> Create Service.
- Odaberite Deployment platform: Cloud Run (fully managed).
- Unesite podatke:
  - Service name
  - Region: *regija vašeg projekta*
  - Container image URL: unesite svoj Artifact Registry image:

```
<REGION>-docker.pkg.dev/<PROJECT_ID>/<REPOSITORY>/<IMAGE>:<TAG>
```

- Kliknite Create.

Cloud Run automatski postavlja endpoint za vašu aplikaciju. Možete testirati pristup servisu i pratiti logove unutar Cloud Run konzole.

Backend (Flask)

Budući da naš consumer "sluša na vratima" (čeka poruke iz Pub/Sub-a), važno je da Cloud Run servis ima backend endpoint koji može primati dolazne HTTP pozive ili push poruke iz Pub/Sub-a. Za to ćemo koristiti Flask; lagani Python web framework koji omogućuje jednostavno definiranje backend ruta i rukovanje HTTP zahtjevima. U sklopu zadatka laboratorijske vježbe dobit ćete Flask template koji već sadrži osnovnu strukturu aplikacije i endpoint koji prima Pub/Sub poruke. Vaš zadatak bit će samo implementirati logiku obrade pristiglih poruka.

Cloud Run Job

Cloud Run Job je varijanta Cloud Run-a namijenjena jednokratnim ili periodičnim batch poslovima. Za razliku od servisa, Job nema stalan endpoint, pokreće se ručno ili po rasporedu i izvršava zadatak do kraja (npr. dohvaćanje podataka, obrada ili slanje u drugi servis). U ovoj vježbi ćemo producer aplikaciju postaviti kao Cloud Run Job, jer dohvaća poruke s Reddit API-ja samo jednom i zatim ih šalje u Pub/Sub topic. Na taj način smanjujemo troškove i pojednostavljujemo arhitekturu - Job se pokrene, izvrši posao i ugasi se.

> **Napomena**: U praksi bi se ovaj Job mogao periodično pokretati pomoću Cloud Scheduler-a (npr. svakih 10 minuta).

Za kreiranje Cloud Run Job-a:

- U Google Cloud Console odaberite Cloud Run -> Jobs -> Create Job.

- Odaberite Deployment platform: Cloud Run (fully managed).
- Unesite podatke:
  - Service name
  - Region: *regija vašeg projekta*
  - Container image URL: unesite svoj Artifact Registry image:

```
<REGION>-docker.pkg.dev/<PROJECT_ID>/<REPOSITORY>/<IMAGE>:<TAG>
```

- Kliknite Create.

Deploy na Cloud Run

Budući da su Docker image-i već buildani i pushani u Artifact Registry, u ovom koraku ćemo ih samo deployati na Cloud Run. Consumer će biti Cloud Run servis s backend endpointom, a producer Cloud Run Job koji dohvaća poruke jednom i šalje ih u Pub/Sub topic.

1. Consumer

```
gcloud run deploy <CONSUMER_SERVICE_NAME> \
  --image <REGION>-
docker.pkg.dev/<PROJECT_ID>/<REPOSITORY>/<CONSUMER_IMAGE>:<TAG> \
  --region <REGION> \
  --platform managed \
  --allow-unauthenticated \
  --service-account cr-deploy-sa@PROJECT_ID.iam.gserviceaccount.com
```

- Npr. gcloud run deploy consumer
  --image europe-west1-docker.pkg.dev/<PROJECT_ID>/docker-images/your-consumer:latest
  --region europe-west1
  --platform managed
  --allow-unauthenticated

2. Producer

```
gcloud run jobs deploy <PRODUCER_JOB_NAME> \
  --image <REGION>-
docker.pkg.dev/<PROJECT_ID>/<REPOSITORY>/<PRODUCER_IMAGE>:<TAG> \
  --region <REGION> \
  --service-account cr-deploy-sa@PROJECT_ID.iam.gserviceaccount.com
```

- Pokretanje Cloud Run Job-a:

```
gcloud run jobs execute <PRODUCER_JOB_NAME> --region <REGION>
```

Secret Manager

Google Cloud Secret Manager je servis za sigurno pohranjivanje i upravljanje osjetljivim podacima poput API ključeva, lozinki i tokena. Glavne prednosti korištenja Secret Manager-a su:

- osjetljive podatke ne pohranjujete u Git ili druge javne repozitorije,
- imate kontrolu pristupa na razini servisa ili korisnika,
- moguće je verzioniranje tajni i njihovo jednostavno rotiranje.

U ovoj vježbi, svi podaci koji su tajni, tj. ne smiju biti javno vidljivi na GitHubu, a potrebni su za deploy aplikacija na Cloud Run, pohranjuju se u Secret Manager.

Kreiranje tajne:

```
gcloud secrets create <SECRET_NAME> --replication-policy="automatic"
```

- Tajnu možete kreirati i pridodati joj vrijednost i ručno unutar Secret Manager-a na Google Cloud Console (+ Create Secret)

Dodavanje vrijednosti tajne:

```
gcloud secrets versions add <SECRET_NAME> --data-
file="path/to/local/keyfile.txt"
```

Ovdje "path/to/local/keyfile.txt" je lokalna datoteka koja sadrži vaš API ključ

Korištenje tajni u Cloud Run-u

- Možete povezati tajnu direktno s environment varijablom Cloud Run servisa:

```
gcloud run deploy <SERVICE_NAME> \
  --image=<IMAGE_URL> \
  --update-secrets=<ENV_VAR>=<SECRET_NAME>:latest
```

Korištenje u aplikaciji

- u aplikaciji dohvatite tajnu putem environment varijable:

```
import os
api_key = os.getenv("API_KEY")
```

> **Napomena:** Na ovaj način vaša aplikacija dobiva tajne vrijednosti u runtime-u, nikada ih ne pohranjujte u kod ili Git repozitorij.

Logs Explorer

Logs Explorer je alat u Google Cloud Console-u koji omogućuje pregled i analizu logova svih GCP resursa. Pomoću njega možete pratiti aktivnosti i greške u Cloud Run servisima, slanje i primanje poruka u Pub/Sub topicima, te događaje vezane za deploy i promjene resursa kroz Audit Logs. Logove je moguće filtrirati prema resursu, servisu ili razini loga (INFO, ERROR), a mogu se pratiti i u realnom vremenu, što je korisno za debugging i praćenje pipeline-a. Logs Explorer omogućuje uvid u tijek poruka i izvršenje servisa te pomaže u prepoznavanju problema i praćenju događaja u projektu.

U ovoj laboratorijskoj vježbi, Logs Explorer će vam pomoći da pratite dolazak poruka u Cloud Run consumer, provjerite jesu li poruke uspješno poslati i primljene, te identificirate eventualne greške prilikom deploya ili izvršenja servisa.

1. Zadatak

Tema ove laboratorijske vježbe je praćenje Reddit postova. Ako vam je ovo prvi susret s API-jima i Pub/Subom preporučujemo da odaberete Python. Dobro proučite dokumentaciju te pronađite zahtjev(e) koji će vam pomoći u ostvarenju tog zadatka. Prije nego što krenete programirati možete se poslužiti Postmanom kako biste ustvrdili kakav zahtjev trebate poslati te kojim podacima želite pristupiti.

Idući korak vašeg zadatka je stvaranje proizvođača (producera) i potrošača (consumera).

Stvaranje producera

- Odaberite i uvezite potrebne biblioteke
- Spojite se na prethodno stvoreni Pub/Sub Topic
- Stvorite API fetcher koji će podatke o objavama subreddita **r/dataengineering** slati na taj topic.
- Dohvatite prvih 10 objava iz kategorije "**Top**" (of All Time)
    - Za dohvaćanje podataka koristite običan HTTP request
    - Podatke s API-ja ne filtrirajte, pošaljite sve dostupne podatke (sva polja iz API odgovora) o svakoj objavi
    - Nakon što se poslali svih 10 objava na Pub/Sub Topic na kraju programa napišite beskonačnu petlju

> **Napomena** Ovo inače nije praksa, ali s obzirom na to da se GCP resursi naplaćuju ovime ćete smanjiti mogućnost velikih troškova.

Stvaranje consumera

- Spojite se na prethodno stvoreni Pub/Sub Topic
- Kada poruke pristignu, pročitajte ih u JSON formatu te ispišite, u stvarnom vremenu
- Proučite polja i podatke dostupne o objavi

Koristeći upute iz prethodnog labosa pohranite producera i consumera na GitHub repozitoriju.

Dockerizacija aplikacija producera i consumera

- Napravite Dockerfile datoteke koje određuju bazne slike, instaliraju potrebne ovisnosti i kopiraju izvorni kod aplikacija u kontejnere
- Izgradite Docker image iz Dockerfile datoteka
- Pokrenite kontejnere iz Docker imagea pazeći pritom na kojoj mreži se nalaze

2. Zadatak

Nakon što ste u prethodnom zadatku razvili producera i consumera te pripremili Dockerfile-ove, sada je vrijeme da cijeli sustav povežete s Google Cloud Platformom. Prvo je potrebno izgraditi Docker Image-e za obje aplikacije, pažljivo ih tagirati i pohraniti u Artifact Registry, čime osiguravate da su spremni za deployment unutar Cloud Run okoline.

Sljedeći korak je deploy consumera kao **Cloud Run** servisa. U ovom servisu, **Flask template** ili vlastiti kod mora biti konfiguriran tako da može primati poruke s Pub/Sub topica i obrađivati ih u stvarnom vremenu. Primjer minimalnog Flask template-a:

```
from flask import Flask
import os

app = Flask(__name__)

@app.route('/listening')
def listening_check():
    return 'Consumer service is listening', 200

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=8080) # Cloud Run port
```

Producer-a postavljate kao **Cloud Run Job** koji, kada se pokrene, šalje prve poruke u Pub/Sub topic. Važno je da u kodu pravilno dohvatite environment varijable za projekt, topic i subscription, a eventualne tajne poput API ključeva povežete preko Secret Manager-a kako bi aplikacije mogle sigurno koristiti osjetljive podatke.

Tijekom rada pipeline-a pratite **Logs** u Cloud Run konzoli ili Logs Exploreru kako biste provjerili dolazak i obradu poruka. Na taj način testirate cijeli workflow, od slanja podataka preko producera do konzumiranja i obrade kroz consumera, i uvjeravate se da je sustav pravilno konfiguriran i funkcionalan.