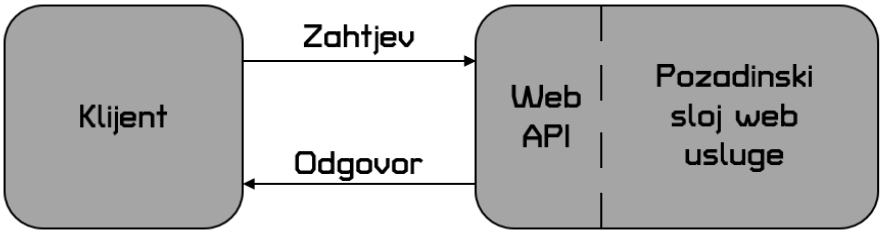
2. laboratorijska vježba

Uvod

Dobrodošli na drugu laboratorijsku vježbu. U ovom zadatku ćete naučiti jednu od temeljnih vještina potrebnih u ovom području – povezivanje na API-je te ćete do kraja ovog laboratorija moći razviti aplikaciju koja može učinkovito proizvoditi poruke na Kafku, kao i dizajnirati robusno i skalabilno rješenje za konzumiranje poruka s Kafke. Osim toga, istraživat će se i koncept kontejnerizacije i naučiti kako razviti aplikacije pomoću Dockera.

API - aplikacijsko programsko sučelje

Aplikacijska programska sučelja (APIs) standardni su način razmjene podataka između sustava. Klijentski programi koriste aplikacijska programska sučelja (APIs) 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. Kao što je prikazano na slici, Web API je sučelje web usluge koje izravno sluša i odgovara na zahtjeve klijenata.



REST API

Kao i API, REST je također akronim te označava *reprezentacijski prijenos stanja* i odnosi se na određeni stil izrade API-ja. REST arhitekturni stil često se primjenjuje u dizajnu API-ja za moderne web usluge, pa se tako i glavne web usluge poput Googlea, Facebooka i Twittera oslanjaju na REST za svoje API-je jer se REST temelji na HTTP-u - protokolu koji pokreće gotovo sve internetske veze. Osim toga, REST je lagan i fleksibilan te može lako podnijeti velike količine aktivnosti. Web API koji se pridržava REST arhitekturnog stila naziva se REST API, a upravo jedan takav ćemo koristiti za ostvarenje ove vježbe - CoinCap API 2.0.

CoinCap API 2.0

Uz CoinCap API 2.0 možete jednostavno pristupiti i integrirati podatke o kriptovalutama u svoje aplikacije pružajući vrijedan uvid u najnovija kretanja na tržištu kriptovaluta. Ovaj API ne zahtijeva upotrebu ključa za zahtjeve koje ćete upućivati. Odgovore vraća u JSON-u - laganom lako čitljivom standardnom formatu za razmjenu podataka.

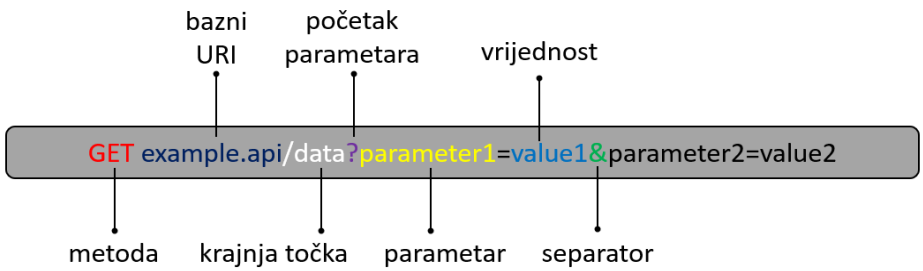
HTTP Metode

Interakcija s API-jem na webu uključuje sastavljanje HTTP zahtjeva s potrebnim zaglavljima i obradu HTTP odgovora, uključujući i pogreške. Klijenti određuju željenu metodu u HTTP zahtjevu. Najčešće metode s kojima ćete se susresti prikazane su u tablici.

Metoda	Upotreba
GET	pristupanje resursu u načinu rada samo za čitanje
POST	slanje novog resursa na poslužitelj
PUT	ažuriranje resursa
DELETE	brisanje resursa

URI - uniformni identifikator resursa

Uniform Resource Identifier (URI) je niz znakova koji jedinstveno identificira web resurse. REST API-ji koriste uniformne identifikatore resursa (URI) za adresiranje resursa. Prije početka korištenja API-ja potrebno je dobro proučiti dokumentaciju - što se šalje u zahtjevu te što se vraća u odgovoru. Prateći dokumentaciju API-ja naučit ćete kako uspješno komunicirati s API-jem i dohvatiti željene podatke preko poslanog URI-ja s odabranom metodom.



HTTP statusni kodovi

REST API-ji obavještavaju klijente o rezultatu njihovog zahtjeva preko statusnih kodova. Statusni kodovi su podijeljeni u pet kategorija prikazanih u tablici.

Kategorija	Opis
1xx: Informacija	Zahtjev zaprimljen, procesiranje se nastavlja
2xx: Uspjeh	Zahtjev zaprimljen, obrađen i akcija uspješno završena
3xx: Preusmjeravanje	Ukazuje da klijent mora poduzeti dodatnu akciju kako bi završio svoj zahtjev
4xx: Greška klijenta	Klijent je uzrok problema - npr. zahtjev nije sintaktički korektan
5xx: Greška poslužitelja	Zahtjev je ispravan, ali ga poslužitelj ne može ispuniti

Postman

Postman je popularni softverski alat koji koriste razvojni programeri za izradu zahtjeva za API (Application Programming Interface) i testiranje svojih API-ja. Pruža korisničko sučelje koje je jednostavno za upotrebu za stvaranje, slanje i upravljanje HTTP zahtjevima te omogućuje korisnicima da pregledaju i analiziraju odgovore iz svojih API poziva. Postman možete preuzeti na osobno računalo ili ga dodati kao ekstenziju u vašem pregledniku. Poželite li primjerice dobiti uvid u podatke o Kraken burzi kriptovaluta poslat ćete GET naredbu sa sljedećim URI-jem: `api.coincap.io/v2/exchanges/kraken` na što ćete povratno dobiti odgovor na vaš zahtjev u JSON formatu.

```
{
    "data": {
        "exchangeId": "kraken",
        "name": "Kraken",
        "rank": "6",
        "percentTotalVolume": "2.19918499618785718000000000000000000000",
        "volumeUsd": "648830895.6632081873880911",
        "tradingPairs": "343",
        "socket": false,
        "exchangeUrl": "https://kraken.com",
        "updated": 1677006553683
    },
    "timestamp": 1677006559816
}
```


Umrežavanje kontejnera

U Dockeru, možete koristiti most (bridge) kako biste omogućili kontejnerima povezivanje na istu mrežu pružajući pritom izolaciju od drugih kontejnera koji nisu na istoj mreži. Kako biste omogućili unutarmrežnu komunikaciju Docker kontejnera dodajte sljedeće retke na kraj vaše docker-compose.yaml datoteke.

```
networks:
  kafka_network:
    driver: bridge
```

Kada nanovo pokrenete docker compose primit ćete obavijest o kreaciji nove mreže.

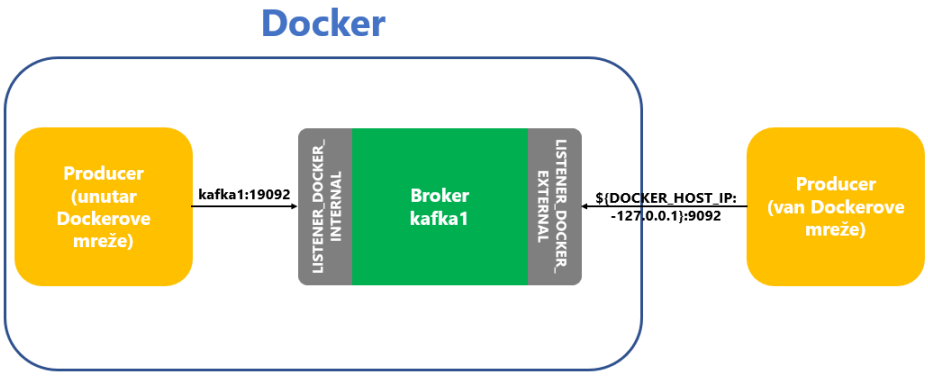
Vratimo se na YAML datoteku iz prošle laboratorijske vježbe preko koje ste pokrenuli Kafku u Dockeru.

```
KAFKA_ADVERTISED_LISTENERS:
LISTENER_DOCKER_INTERNAL://kafka1:19092,LISTENER_DOCKER_EXTERNAL://${DOCKER_H
OST_IP:-127.0.0.1}:9092
KAFKA_LISTENER_SECURITY_PROTOCOL_MAP:
LISTENER_DOCKER_INTERNAL:PLAINTEXT,LISTENER_DOCKER_EXTERNAL:PLAINTEXT
KAFKA_INTER_BROKER_LISTENER_NAME: LISTENER_DOCKER_INTERNAL
```

Polja predstavljaju redom:

- `KAFKA_ADVERTISED_LISTENERS` - lista listenera koji mogu uspostaviti konekciju s Kafka klijentom razdvojenih zarezom sa svojom IP adresom/hostnameom i portom.
- `KAFKA_LISTENER_SECURITY_PROTOCOL_MAP` - parovi ključ-vrijednost kojima označavamo koje sigurnosne protokole koriste listeneri
- `KAFKA_INTER_BROKER_LISTENER_NAME` - popis listenera koji se koriste za unutarnju komunikaciju

Komunicirati s Dockerom možete unutar iste Docker mreže ili van te mreže. Ako želite lokalno poslati podatke s API-ja na Kafku u Dockeru komunicirat ćete izvana, a nakon što dockerizirate svoje aplikacije morat ćete uspostaviti unutarmrežnu komunikaciju među kontejnerima.



Group ID

Group ID je konfiguracijska postavka u Kafka potrošačima. Ako želite da više potrošača podijeli radno opterećenje, dodijelit ćete im isti group ID; ako pak želite da jedan potrošač radi neovisno, dodijelit ćete mu jedinstveni group ID - group ID je ništa više od niza znakova. Kafka potrošači su izgrađeni za skaliranje na više strojeva, radeći pritom u grupama. Svaki potrošač govori Kafka brokeru kojoj grupi pripada, a zatim se izlazne poruke automatski ravnomjerno raspodjeljuju među članovima te grupe. Tako zamislite da imate temu “kupovine”. Sustav koji pišete trebao bi pokrenuti obavijest svaki put kada kupac nešto kupi. Slanje velikog broja emailova može biti sporo, pa možda imate pet strojeva koji konzumiraju s group ID-ja "email" kako bi razdijelili opterećenost. U isto vrijeme, možda želite sažeti te kupovine kako biste dobili podatke o prodaji, a to možda zahtijeva samo jedan stroj s vlastitim group ID-jem “prodaja”.

Zadatak

Dohvat podataka s API-ja

Tema laboratorijske vježbe je praćenje cijena kriptovaluta u stvarnom vremenu; odnosno, trebate prikazati kretanje cijene kriptovalute po slobodnom izboru u proteklih sto dana u intervalima od sat vremena u jeziku po izboru - ako vam je ovo prvi susret s API-jima i Kafkom preporučujemo da odaberete Python. Dobro proučite dokumentaciju te pronađite zahtjev koji će vam omogućiti ostvarenje tog zadatka. Prije nego što krenete programirati možete se poslužiti Postmanom kako biste ustvrdili kakav zahtjev trebate poslati te koje podatke te u kojem obliku želite izvući iz odgovora. Koristite HTTP biblioteke kako bi dohvatili podatke s API-ja u vašem kodu.

Stvaranje Kafka proizvođača (producera)

1. Odaberite i uvezite potrebne biblioteke - slobodan izbor, ali preporučamo Confluentove biblioteke

2. Kreiranje i odabir teme (topica) na koju se šalju podatci - nazovite je prema imenu kriptovalute koju ste odabrali
3. Podesite konfiguraciju Kafka proizvođača - obično IP adresa i port (prisjetite se konfiguracije listenera u YAML datoteci)
4. Kreiranje instance Kafka proizvođača
5. Stvorite petlju u kojoj ćete obrađene podatke API-ja slati kao poruke na Kafka posrednika (broker)
    1. Jedna poruka predstavlja cijenu kriptovalute u intervalu od sat vremena - šaljite jednu poruku po desetini sekunde
    2. Formatirajte sadržaj poruke - poruku pošaljite u obliku: vrijeme,cijena s vrijednostima u danom trenutku
    3. Objava poruka (publish) na Kafka temu

Stvaranje Kafka potrošača (consumera)

1. Odaberite i uvezite potrebne biblioteke - slobodan izbor, ali preporučamo Confluentove biblioteke
2. Podesite konfiguraciju Kafka potrošača - obično IP adresa, port i group id (prisjetite se konfiguracije listenera u YAML datoteci)
3. Kreiranje instance Kafka potrošača
4. Pretplaćivanje na temu na koju se šalju poruke iz prethodnog odjeljka
5. Provjera dolaska novih poruka na temu pomoću `poll()` metode te njihova obrada - ispišite sadržaj svake poruke na zaslon

Dockerizacija aplikacija proizvođača i potrošača

1. Napravite Dockerfile datoteke koje određuju bazne slike, instaliraju potrebne ovisnosti i kopiraju izvorni kod aplikacija u kontejnere
2. Izgradite Docker slike iz Dockerfile datoteka
3. Pokrenite kontejnere iz Docker slika pazeći pritom na kojoj mreži se nalaze

Napomena: Ne zaboravite vaše kodove pushati na Github.