Eksamen PGR301

Bakgrunn

Du er leid inn til SkalBank AS, som trenger desperat hjelp.

Banken har brukt flere år og hundretalls milioner på å utvikle et moderne kjernesystem for bank og et "fremoverlent" API som
nesten tilfredsstiller Directive (EU) 2015/2366 of the European Parliament and of the Council on Payment Services in the Internal Market, published 25 November 2016 også kjent som PSD.

Dette er en viktig satsning innen området "Open Banking" for SkalBank.

Arkitekturmessig består systemet av to komponenter.

- Et API, implementert ved hjelp av Spring Boot. Applikasjonen og noe infrastrukturkode ligger i dette repoet.
- Et kjernesystem som utfører transaksjoner med andre banker, avregner mot Norges bank osv. Dere kan late som metodekall som gjøres mot klassen ReallyShakyBankingCoreSystemService, kommuniserer med dette systemet.
- NB! Dere skal IKKE gjøre endringer i klassen *ReallyShakyBankingCoreSystmeService*

Finanstilsynet et litt imponert over den delvise etterlevelsen av direktivet, men ikke imponert over stabiliteten til SkalBanks API. Banken har en en rekke tekniske problemer

- Applikasjonen er veldig ustabil.
- Det er vanskelig å si om problemet ligger i API eller kjernesystemet. Alle driver faktaløs "Blamestorming" uten å fokusere på å finne ut av hvor problemet ligger.
- Responstidene er veldig variabel, og Applikasjonen feiler med sporadiske "BackEndException".
- Det er umulig å se hva som faktisk er feil. Applikasjonen lager lite logger, og gir ikke fra seg noe form for telemetri.

SkalBank har også problmer med sin systemutviklingsprosess for API teamet.

- Mellom fem og ti utviklere committer til *main* branch kontinuerlig, uten nødvendigvis å kompilere koden og kjøre
  tester. Dette skaper naturligvis det komplette kaos.
- Det finnes et team i SkalBank som jobber med manuelle tester og drift; "*Team Dino*". Teamet er på ca 100 ansatte og SkalBank vurderer å

rekruttere ytterligere for å øke kvaliteten på leveransene som har vært fallende siden lansering.
- Hver gang en ny versjon av API skal releases, lager tech lead "Jens" en JAR han gir til "Team Dino"".
- "Team Dino" tester applikasjonen grundig- og setter den nye versjonen i drift.
- "Team Dino" gjør overfladisk overvåkning, og starter applikasjonen på nytt hver natt eller etter behov.

Krav til leveransen

- Eksamensoppgaven er gitt på GitHub repository ; https://github.com/PGR301-2021/eksamen_2021
- Du skal IKKE lage en fork av dette repositoryet, men kopiere innholdet til et nytt repository. Årsaken til dette er at sensor vil lage en fork av ditt repo, og arbeidsflyten blir lettere for sensor om har har et frittstående repo.
- Du kan velge å kode i et privat eller public repo.
- Du kan jobbe i et privat repo, og deretter gjøre det public noen timer etter innleveringsfrist hvis du er bekymret for plagiat fra medstudenter.

Når sensor evaluerer oppgaven vil han/hun se på

- Ditt repository og "Actions" fanen i GutHub for å bekrefte at Workflows faktisk virker
- AWS miljøet i klassens AWS konto for å bekrefte at oppgaver som beskrevet er utført
- Vurdere drøftelsesoppgavene. Det anbefales å lage en egen "Readme/Markdown" for disse i ditt repo.
- Lager en fork av ditt repo og tester ut pipelines med egen AWS bruker/github bruker.

Ved innlevering via WiseFlow, kan dere lage et tekstdokument som kun inneholder link til deres repository

Evaluering

- DevOps prinsipper & Pipeline - 20 poeng
- Feedback & Telemetri -30 poeng
- Terraform og IAC i Pipeline - 30 poeng
- Docker - 20 poeng

Litt om GitHub free tier

- I oppgaven blir du bedt om å lage GitHub actions workflows.

- Med GitHub "Free tier" har du 2,000 minutter med gratis byggetid per måned, dersom du bruker et privat repo.
- Dersom dere i en ekstrem situasjon skulle trenge mer byggetid, kan dere gjøre repository public. Da er byggetiden ubegrenset.
- Hvis dere da er bekymret for at andre skal kopiere arbeidet deres, kan dere lage en ny GitHub bruker med et tilfeldig navn.

OBS!

- I "Free" planen til GitHub er "branch protection" ikke tillat når et repository er privat. Det vil si at dere ikke kan konfigurere GitHub til å hindre push mot for eksempel *main* branch direkte, eller konfigurere regler for godkjenning før merge av pull request osv.
- I denne oppgaven blir dere bedt om å beskrive *hvordan* dette kan gjøres, men dere trenger altså ikke konfigurere dette for repoet dere leverer.

Beskrivelse av API

APIet eksponerer tre endepunkter, dere kan ikke endre på disse

- /account/{fromAccount}/transfer/{toAccount} [POST] - Overfører penger mellom to kontoer. Oppretter kontoer for både mottaker ov avsender dersom de ikke finnes. Se modellpakken for beskrivelse av Payload
- /account [POST] - Oppdaterer en konto - Se modellpakken for beskrivelse av Payload
- /accoount/{accountId} [GET] - Returnerer kontoopplysninger

Oppgave - DevOps

Med DevOps som arbeidsmåte i tankene- Hvilke forbedringer kan teamet gjøre med fokus på måten de jobber med kildekode og versjonskontroll?

Drøft

- Beskriv med ord eller skjermbilder hvordan man kan konfigurere GitHub på en måte som gir bedre kontroll på utviklingsprosessen. Spesielt med tanke på å hindre kode som ikke kompilerer og feilende tester fra å bli integrert i *main* branch.
- Beskriv med ord eller skjermbilder hvordan GitHub kan konfigureres for å sikre at minst ett annet medlem av teamet har godkjent en pull request før den merges.
- Beskriv hvordan arbeidsflyten for hver enkelt utvikler bør være for å få en effektiv som mulig utviklingsprosess, spesielt hvordan hver enkelt utvikler bør jobbe med Brancher i Github hver gang han eller hun starter en ny oppgave.

Drøft

SkalBank har bestemt seg for å bruke DevOps som underliggende prinsipp for all systemutvikling i banken. Er fordeling av oppgaver mellom API-teamet og "Team Dino" problematisk med dette som utgangspunkt? Hvilke prinsipper er det som ikke etterleves her? Hva er i så fall konsekvensen av dette?

Oppgave Pipeline

Skriv minst en enhetstest. Enhetstester er ikke Team API sin sterkeste side.
Lag en GitHub actions workflow som ved hver *push* mot `origin/main` branch gjør følgende ved hjelp av Maven, og pom.xml filen som ligger i dette repoet.

- Kjører enhetstester
- Kompilerer koden
- Bygger artifakt (JAR)

Sensor vil

- Brekke testen med vilje, og se at pipelinen feiler.
- Forsøke å pushe kode som ikke kompilerer til main.

Oppgave - Feedback

Når noe går galt, noe som stort sett er hele tiden, kan ikke "Team Dino" gjøre noe annet enn å starte prosessen på nytt, de har ingen innsikt i applikasjonen sin
tilstand mens den kjører.

Det har vært opphetede debatter om hvor problemene oppstår.

Teamet som lager Kjernesystemet peker på APIet, og API teamet peker på kjernesystemet.

Et av de største problemene er de lange, og lite konsekvente responstidene.

Endre applikasjonen slik at den gir fra seg telemetri

Gjør nødvendige endringer i Spring Boot applikasjonen, slik at den produserer metrics med Micrometer rammeverket og leverer disse til Influx DB på lokal maskin.

- Legg til kode som registrerer målepunkter i applikasjonen, slik at dere kan bevise med

fakta hva som forårsaker de dårlige responstidene.
- Det ønskelig å måle hvor ofte APIet faktisk kaster "BackEndException"
- Dere kan anta at sensor kjører InfluxDB på sin maskin.
- Sensor vil gjøre flere kall mot endepunktene i applikasjonen med for eksempel Curl eller Postman

Bruk valgfrie mekanismer fra Micrometer rammeverket (Timer, Counter, Gauge, LongTaskTimer, DistributionSummary osv) for å samle data fra hvert av endepunktene i APIet.

- Hvilke spørring(er) kan sensor gjøre mot InfluxDB for å analysere problemet? For eksempel noe i retning av;

```
select * from my_timer_metric_name
```

- Start Grafana på lokal maskin ved hjelp av Docker. Bruk InfluxDB som en datakilde og legg ved et skjermbilde av et Dashboard du har laget som viser en Metric fra InfluxDB som er produsert av Micrometer rammeverket.

NB.

Dersom du løser oppgaven på Linux operativsystem, vil jeg anbefale å bruke "host" basert nettverksmodus for Docker slik at Spring boot applikasjonen, Influx DB og Grafana kan kommunisere fritt på "localhost".

I denne oppgaven vektlegges det at du har klart å bruke Micrometer rammeverket til å identifisere problemområdet til applikasjonen.

Oppgave Terraform

Terraformkoden ligger i *infra* katalogen i dette repoet.

Teamet som har laget SkalBank sitt API, started med høye ambisjoner om å lage terraform kode for all infrastruktur, men de møtte raskt på problemer, og ga opp.

Hver gang en utvikler kjører `terrafomrm apply` fra sin maskin, dukker det opp en *terraform.tfstate* fil i samme mappe som de kjører terraform fra. Og prosessen feiler.

Dette fungerte, i følge teamet, på "Jens" sin maskin minst en gang for lenge siden, og bucket ble opprettet i S3. Men, det feiler for alle andre. Det feiler også for Jens nå, etter han "ryddet" på maskinen sin og slettet den mystiske *terraform.tfstate* filen.

Nå får alle samme feilmelding!

```
aws_s3_bucket.mybucket: Creating...

Error: Error creating S3 bucket: BucketAlreadyOwnedByYou: Your previous request to
    status code: 409, request id: R17AHHV5ACY29JYQ, host id: fRoCEBp3sHGQ3ci3eYP9O+

  on bucket.tf line 1, in resource "aws_s3_bucket" "mybucket":
   1: resource "aws_s3_bucket" "mybucket" {
```

Drøft

Hvorfor funket terraformkoden i dette repoet for "Jens" første gang det ble kjørt? Og hvorfor feiler det for alle andre etterpå, inkludert Jens etter at han ryddet på disken sin og slettet *terraform.tfstate* filen?

Viktig!

Slett *infra/bucket.tf* fra repoet ditt. Du skal ikke ha med denne filen videre i din egen infraoppgave.

Lag en S3 bucket i klassens AWS konto

Bruk påloggingsinforasjonen gitt i Canvas for å logge på klassens AWS konto. Bruk AWS Console (UI) eller CLI til å lage en S3 bucket. Denne skal ha følgende egenskaper.

- Navn skal ha pgr301-!identifikator!-terraform
- Region skal være eu-west-1

Identifikator kan for eksempel være kandidatnummer for eksamen. Eller studentnavnet ditt.

AWS CLI

Sensor ønsker å lage sin bucket ved hjelp av CLI. Sensor har aws kommandolinje installert på sin lokale maskin. Hva må sensor gjøre for å konfigurere AWS nøkler/Credentials? Anta at Sensor sin AWS bruker ikke har nøkler/credentials fra før.

Fullfør

```
aws s3api ...
```

AWS brukeren du har fått utlevert har ingen nøkler. Ved hjelp av Console (UI) Lag en Access Key som du kan bruke videre i oppgaven.

Endre Terraform provider til å bruke en S3 backend for state.

Gjør nødvendige endringer i Terraform kode for å bruke en Backend som lagrer state i et objekt i S3 bucketen du opprettet i forrige oppgave.

Terraform kode

Lag Terraform kode som oppretter følgende ressurser i klassens AWS konto, i region *eu-west-1*

- ECR repository. Navnet på repository skal være studentnavn eller en annen unik identifikator, for eksempel kandidatnummer for eksamen.

ECR (Elastic Container Reigstry) brukes for å lagre Docker container images. Vi skal bruke dette ECR Repoet i en senere oppgave.

Terraform i Pipeline

- Implementer en workflow med GitHub actions som kjører `Terraform init & apply` for hver endring av kode i *main* branch.
- Implementer en workflow med GitHub actions som kjører `Terraform init & plan` for hver pull request som lages mot main branch slik at de som gjør en code review kan se hva konsekvensen av å godta endringen vil være.
- Pipeline skal feile dersom Terraformkode som pushes til main ikke er riktig formatert.
- Pipeline skal bare kjøre dersom det er endringer in *ifra/* katalogen.

Sensor vil å lage en fork av ditt repo

- Beskriv hva sensor må gjøre etter han/hun har laget en fork for å få pipeline til å fungere for i sin AWS/gitHub konto.
- Hvilke verdier må endres i koden?
- Hvilke hemmeligheter må legges inn i repoet. Hvordan gjøres dette?

Oppgave - Docker

Teamet har bestemt seg for å ta i bruk Docker. En av årsakene til dette er at medlemmer av

"Team Dino" ofte ikke har Java SDK og Maven installert på sine maskiner.

Obs!

Når dere starter applikasjonen i Docker, eller via Spring Boot, uten at InfluxDB kjører vil dere få mange feilmeldinger av denne typen i terminalen

```
java.net.ConnectException: Connection refused (Connection refused)
    at java.base/java.net.PlainSocketImpl.socketConnect(Native Method) ~[na:na]
    at java.base/java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl
    at java.base/java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSoc
    at java.base/java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.j
    at java.base/java.net.Socket.connect(Socket.java:609) ~[na:na]
```

Dette er fordi Spring Boot applikasjonen ikke har nettverksforbindelse til InfluxDB. Disse feilmeldingene kan dere ignorere, og Spring Boot applikasjonen fungerer som forventet til tross for feilmeldinger.

Dockerfile

"Jens" som er Tech lead er *veldig* skeptisk til Endringer. Applikasjonen har er utviklet for Java 11, og det ønsker han å fortsette med i all fremtid. Derimot sier mageføelsen hans, som han stoler mye på, at applikasjonen sitt *kjøremiljø* bør være basert på det splitter nye `openjdk:18-jdk-alpine3.14` container imaget. Ta høyde for dette i Dockerfilen du skriver.

- Skriv en `Dockerfile` og legg den til i ditt repository. Docker prosessen skal både kompilere og bygge Spring Boot applikasjonen, og kjøre den.

Hva vil kommandolinje for å *bygge* et container image være? Fullfør …

```
docker .....
```

Hva vil kommando for å *starte* en container være? Applikasjonen skal lytte på port 7777 på din maskin. Fullfør…

```
docker .....
```

Medlemmer av "Team Dino" har av og til behov for å kjøre to ulike versjoner av applikasjonen

lokalt på maskinen sin, *samtidig* .Hvordan kan de gjøre dette uten å få en port-konflikt? Hvilke to kommandoer kan man kjøre for å starte samme applikasjon to ganger, hvor den ene bruker port 7777 og den andre 8888?

```
docker .....
```

```
docker .....
```

Lag en GitHub Actions workflow som bygger et Docker image av Spring Boot applikasjonen.

- GitHub Workflowen skal kjøres ved hver push til *main* branch.
- Hvert Container image skal ha en unik tag som identifiserer hvilken commit i GitHub som ble brukt som grunnlag for å bygge container image.
- Container image skal pushes til ECR repository som ble laget i Terraform oppgaven.
- Hvis du ikke har fått til Terraform oppgaven, kan du lage et ECR repository manuelt via AWS console (UI), og du får ikke poengtrekk i denne oppgaven dersom du gjør dette.

Lykke til og ha det moro!