# PGR301 DevOps eksamen 2021

# Oppgave – DevOps

* *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.*

For å få god kontroll på arbeidsprosessen kan man sette opp «Branch protection rules», i Settings > Branches > Add Rule.

«Require a pull request before merging» betyr at man ikke kan pushe rett til main, men at man må gå gjennom en annen branch og så lage pull requests. Man kan sikre at andre medlemmer har godkjent pull requesten ved å velge «Require approvals».

NB: Man må også velge «Include administrators» om det skal gjelde for absolutt alle.

Text

Description automatically generated

For å forhindre at man får feilende tester i main branchen, kan man sette opp å kreve status checks før man merger. I bildet under er det valgt at «build» is Github Actions skal bli godkjent før man kan merge en pull request. I eksempelet her vil koden prøve å builde en JAR med ‘mvn -B package –file pom.xml’. Om det failer, kan man ikke merge. Man kunne også *kun* kjørt tests, f.eks ‘npm test’ i et prosjekt som bruker npm, f.eks en nettside i React.

A screenshot of a computer

Description automatically generated with medium confidence

For å få en effektiv arbeidsflyt, burde nye oppgaver ha egne branches, som igjen er navngitt med beskrivende navn. Man kan også knytte potensielle github issues opp mot branch ved å bruke github issues ID i navnet på branchen. Her kan forskjellige firmaer ha forskjellige konvensjoner på hvordan de navngir branches, men et eksempel vil være «tests/added-tests-to-cover-doSomething/issueID». Ikke «fix-new-test».

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?*

Her trenger vi litt drøfting

# Oppgave – Feedback

Skjermbilde av dashboard i grafana (og influxdb?)

Kanskje noe tekst om det

# Oppgaver Terraform

Litt drøfting om hvorfor Jens sin state-fil

AWS CLI

*Sensor ønsker å lage sin egen bucket ved hjelp av CLI. Sensor har aws kommandolinje installer 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.*

Det første sensor må gjøre er å autentisere seg mot AWS ved hjelp av nøkler som er knyttet til brukerkontoen. Man trenger to nøkler; en Access Key og en Secret Access Key. Disse finner man I IAM -> Security Credentials -> Access keys for CLI, SDK, & API access -> Create Access Key.

For å autentisere seg kjører man kommandoen ‘aws configure’ i AWS CLI. Man fyller inn Access Key og Secret Access Key når man blir bedt om det, velger region eu-west-1 som default region name, og på output format kan det anbefales å velge json.

For å opprette en ny bucket fra CLI må sensor skrive: «aws s3api create-bucket --bucket pgr301-abc123-terraform --create-bucket-configuration LocationConstraint=eu-west-1»

# Terraform i Pipeline

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?

HER MÅ VI HA LITT TEKST OM HVA SENSOR MÅ ENDRE OSV

# Oppgave – Docker

*Hva vil kommandolinjer for å byggge et container image være?*

«docker build . -t containerimage»

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

«docker run -p 7777:8080 containerimage»

Man kan også spesifisere tag når man builder eller kjører, ved å legge på :tag etter containerimage, f.eks; «…8080 containerimage:latest».

*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 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?*

Man må bare kjøre den samme run kommandoen to ganger, der man endrer hvilken port docker lytter på mot maskinen.

«docker run -p 7777:8080 containerimage»

«docker run -p 8888:8080 containerimage»

* Workflowen (create\_image.yaml) kjører *kun* ved push til master, *ikke* ved pull request.
* Alle container-images har unik tag som identifiserer hvilken github commit som ble brukt som grunnlag. Det er også en latest-tag.
* ECR repository er det samme som lages automatisk av terraform.