Verktøy for sentral loggføring i proton CT

Systemdokumentasjon

Versjon <6.0>

Dokumentet er basert på Systemdokumentasjon utarbeidet ved NTNU. Revisjon og tilpasninger til bruk ved IDER, DATA-INF utført av Carsten Gunnar Helgesen, Svein-Ivar Lillehaug og Per Christian Engdal. Dokumentet finnes også i engelsk utgave.

REVISJONSHISTORIE

Dato	Versjon	Beskrivelse	Forfatter
26/04/22	1.0	Begynt å fylle ut dokumentasjon for Loki Promtail og Grafan	Anders, Etkar og Nichlas
10/04/22	2.0	Fyllt ut dokumentasjon for Docker, Promtail og Grafana.	Anders, Etkar og Nichlas
24/04/22	3.0	Fylt ut for Servertjeneste	Anders
30/04/22	4.0	Fylt ut for database	Etkar
17/04/22	5.0	Fylt inn siste rest av dokumentasjon om de forskjellige instansene	Anders, Etkar og Nichlas
22/05/22	6.0	Siste finpuss før innlevering.	Anders, Etkar og Nichlas

INNHOLDSFORTEGNELSE

FIGURLISTE

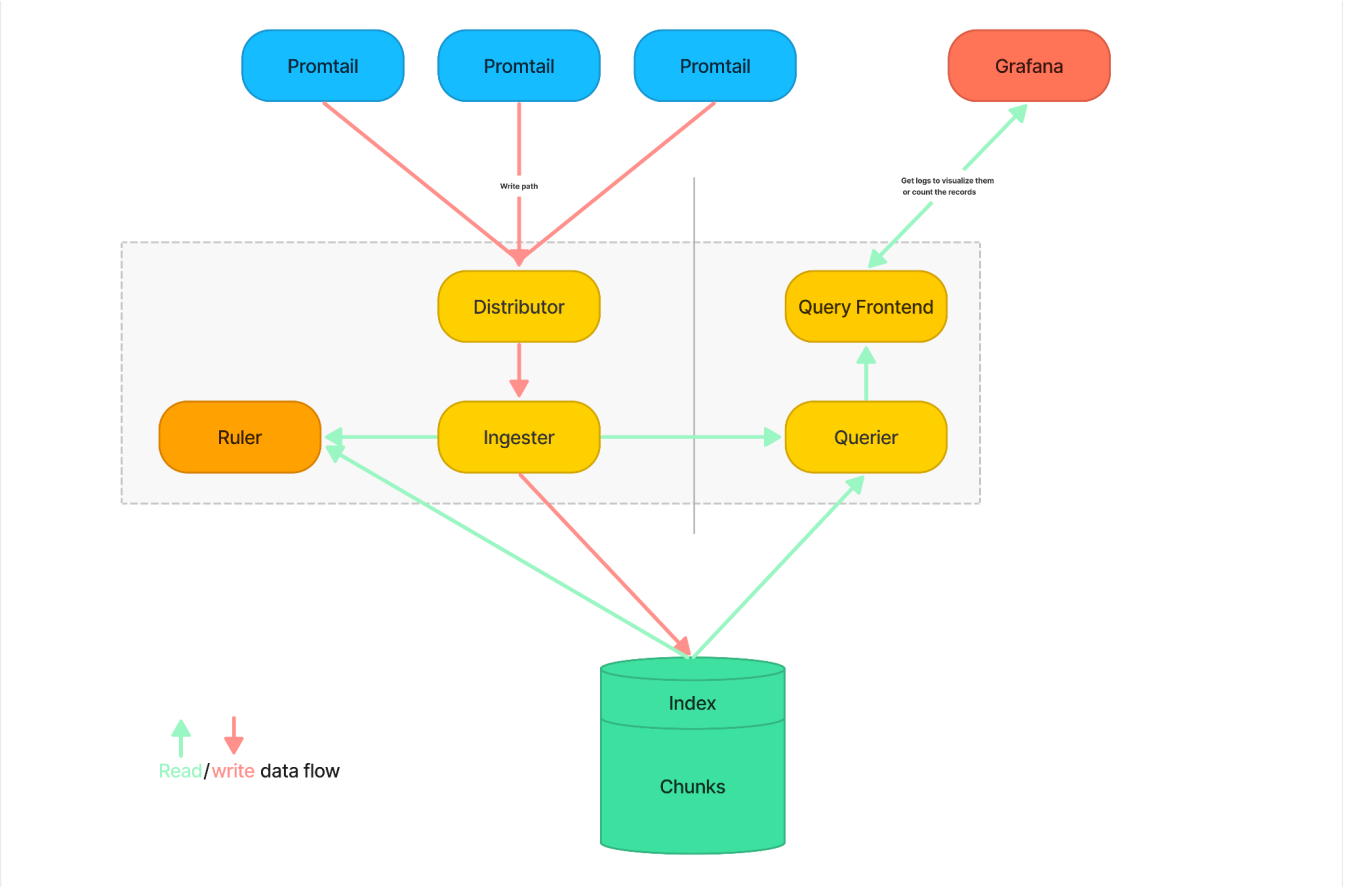
1 INNLEDNING

I dette dokumentet vil strukturen til prosjektet bli nærmere forklart. Det vil komme frem hvordan system instanser virker i det distribuerte systemet og kommunikasjon på tvers av disse. Det er beskrevet detaljerte løsninger på hvordan installasjon og kjøring av de forskjellige instansene gjøres.

2 Arkitektur

2.1 Kommunikasjonen mellom Promtail, Loki og Grafana

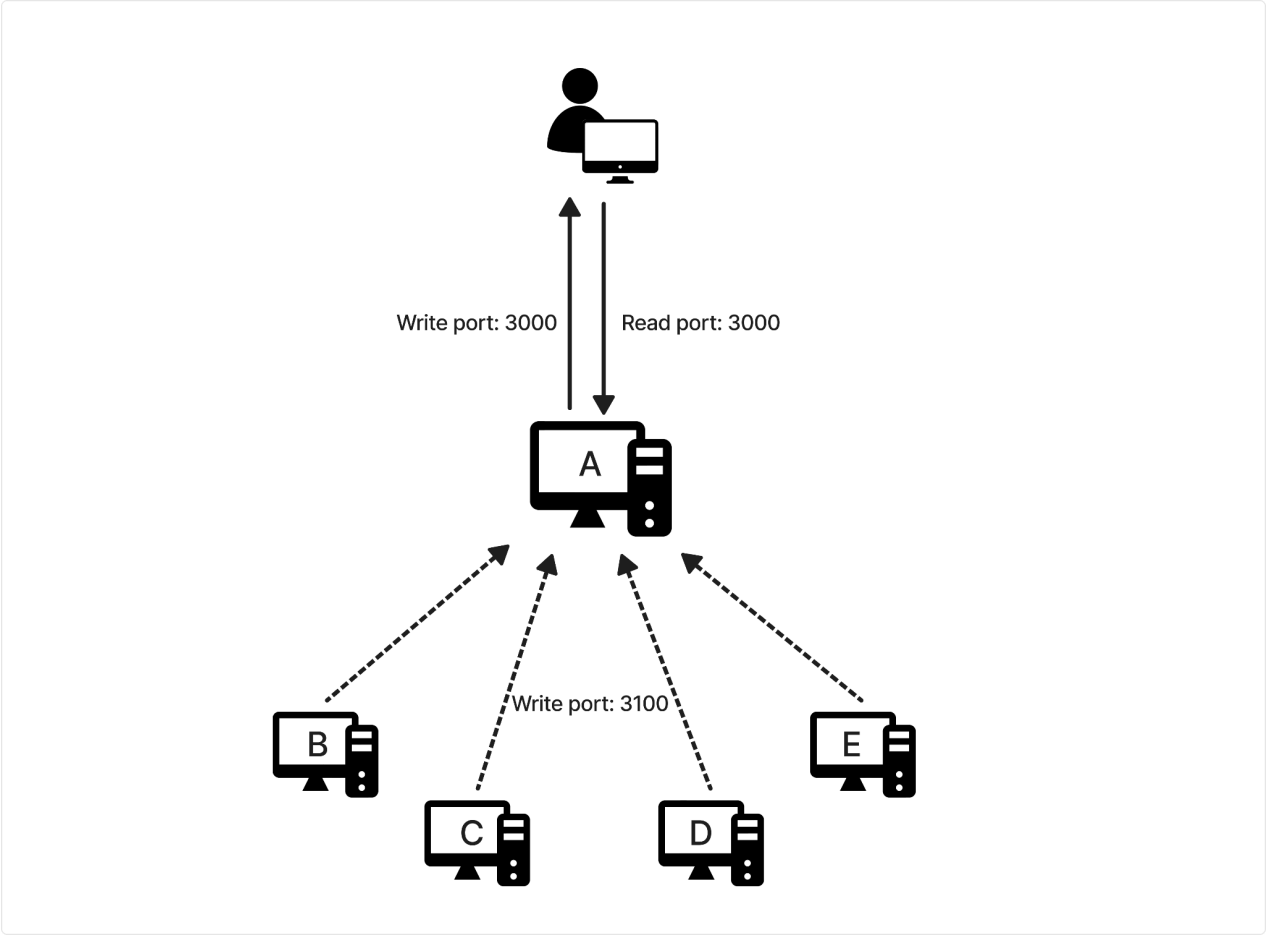
Promtail-agenter oppdager og skriver loggmeldinger til en distributør som er ansvarlig for å håndtere innkommende loggstrømmer. Hver strøm valideres før de porsjoneres i partier og blir sendt parallelt til mange Ingesters (Grafana. u.å.a). Ingestertjenesten har ansvar for å skrive loggdata til langtidslagring (Amazon DynamoDB og S3) og returnere loggdata for queries i minnet. En Querier henter loggmeldinger fra både Ingesters og databasen og returnerer dem til en Query Frontend for å akselerere lesehastigheten (Lewandowski, 2021). Herfra organiseres loggdata og blir visualisert ved ved hjelp av et egendefinert dashbord i Grafana.



Figur 1: Loki arkitektur med Promtail agenter og Grafana

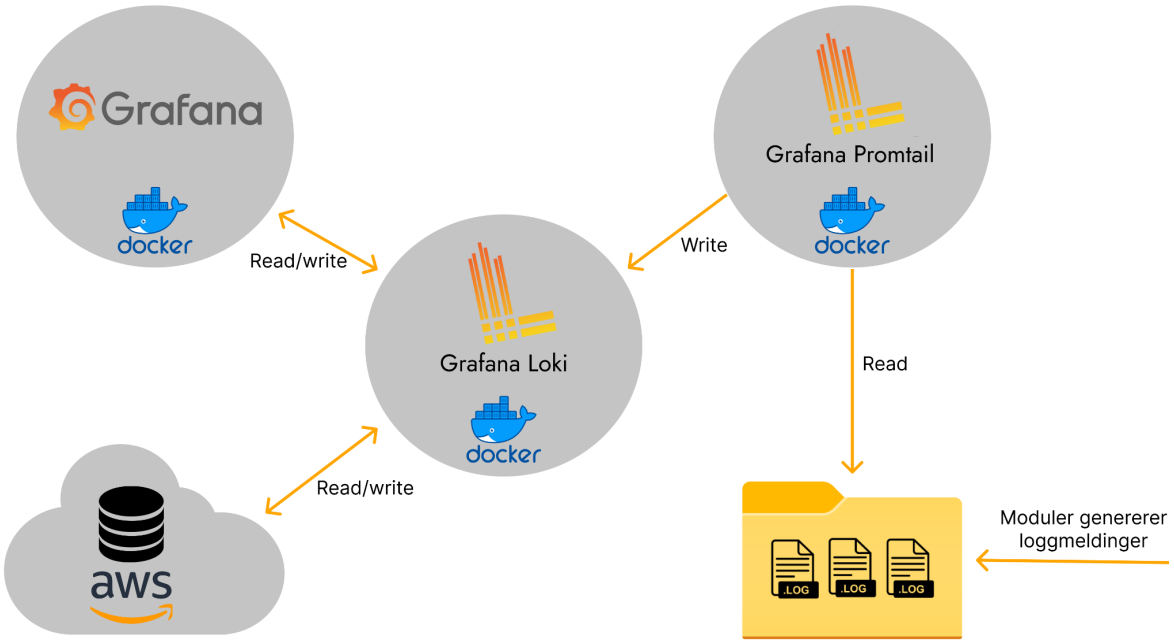
2.2 Kommunikasjon i det distribuerte systemet

Det distribuerte systemet inneholder en server (A) og et antall noder (B,C,D,E). På nodene genereres det loggmeldinger som monitoreres samtidig ved at en server sentraliserer tilgangen til logginstansene (Lukems, 2021). I figuren under, monitoreres server A av en ekstern bruker. Serveren kjører Grafana, Loki og Promtail i Docker-compose, mens hver av nodene kun kjører en Docker container med Promtail. Loggmeldingene blir pushet til Loki sitt API på port 3100 via Promtail. Der blir de prosessert før de kobles opp mot Grafana på port 3000, slik at brukeren enkelt kan monitorere systemet via en nettleser.



Figur 2: Sammenheng mellom server, klienter og ekstern bruker

2.3 Det overordnede systemet

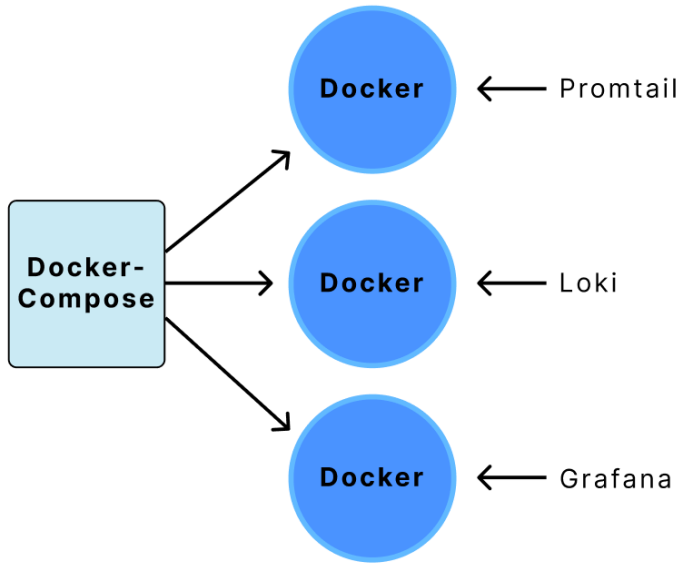


Figur 3: Det overordnede systemet med de viktigste komponentene

2.4 Docker Compose

De tre hovedkomponentene i loggføringsverktøyet (Grafana, Loki og Promtail) kjører i hver sine ferdig-konfigurerte Docker containere. Docker Compose er et verktøy som hjelper til med å definere og samle en multi-container applikasjon, slik som det sentrale loggføringssystemet (Docker, u.å, a). Med Docker Compose lages det en YAML fil som definerer de forskjellige containerne som skal kjøres, og med én enkelt kommando kjøre, samt avslutte disse. Dette gjør det praktisk, slik at man enkelt holder styr på alle container instansene, og slipper da i tillegg å starte opp en og en container manuelt. Docker Compose YAML filen kan også inneholde ekstra instrukser for hvordan forskjellige Containere skal kjøres.

For å kjøre en container med Loki og Promtail trengs et Docker image og en konfigurasjonsfil. Docker imaget inneholder all kode som trengs for å kjøre Docker containeren. Konfigurasjonsfilen gir Promtail og Loki, som kjører i hver sin container, instrukser på hvordan de skal oppføre seg.



Figur 4: Docker-compose illustrasjon

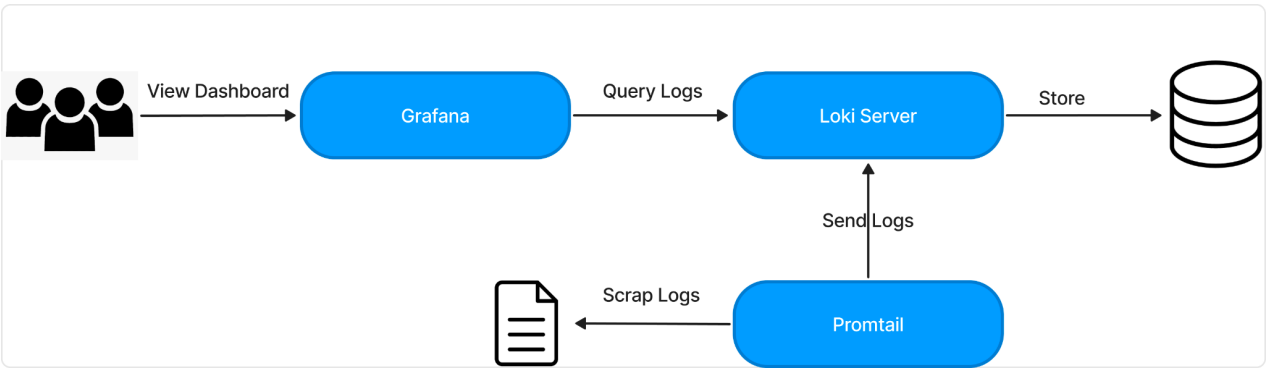
3 PROSJEKTSTRUKTUR (KOMMUNIKASJON)

Prosjektet kan deles inn i forskjellige strukturer etter dybdeforhold; Fra de helt grunnleggende hovedkomponentene i systemet, til kodestrukturen i enkelte komponenter. I dette kapittelet vil de mest relevante strukturene for prosjektet bli belyst:

- Hovedkomponentene
- GitLab
- Kodestruktur

3.1 Hovedkomponentene

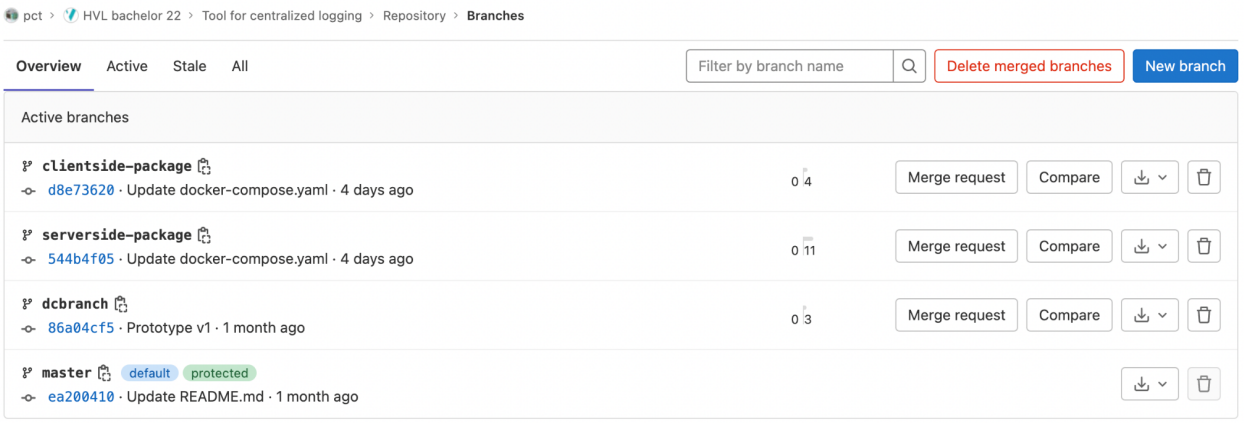
De mest sentrale komponentene i loggføringsverktøyet er organisert i Docker containere som er bygget sammen i Docker compose (se figur 4). I figuren nedenfor er kommunikasjonen mellom disse komponentene illustrert *(Spring Cloud, 2022)*.



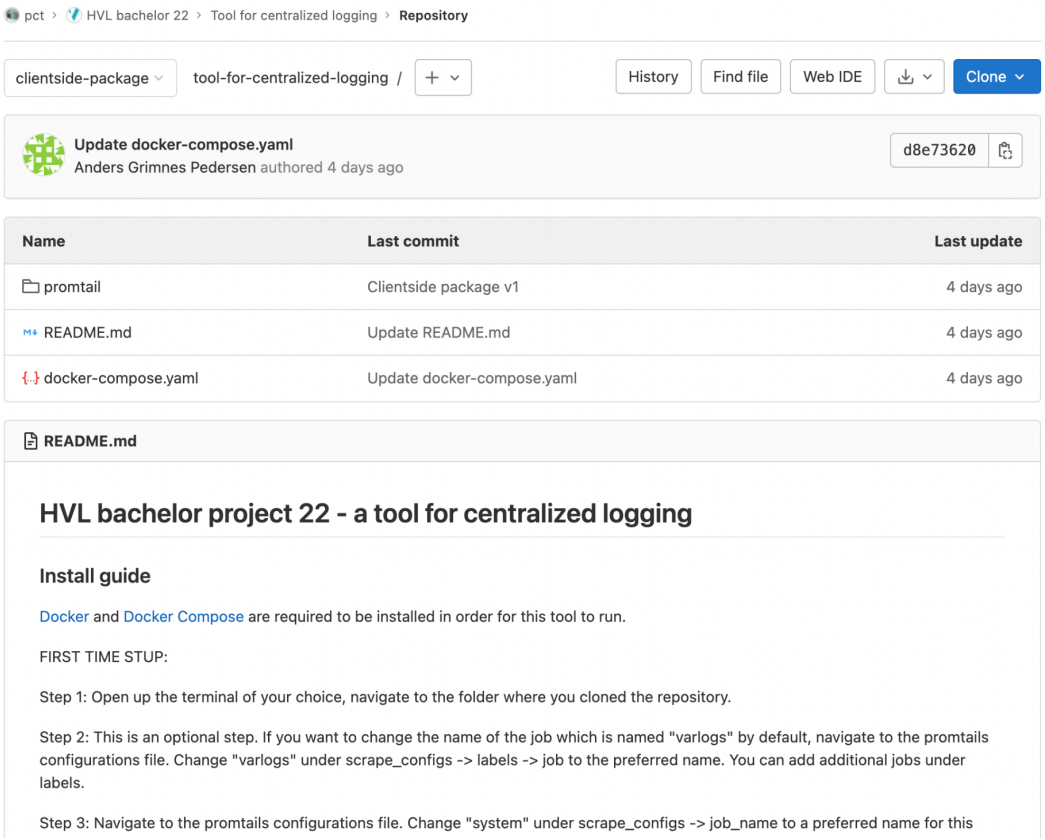
Figur 5: Kommunikasjonen mellom hovedkomponentene

3.2 GitLab

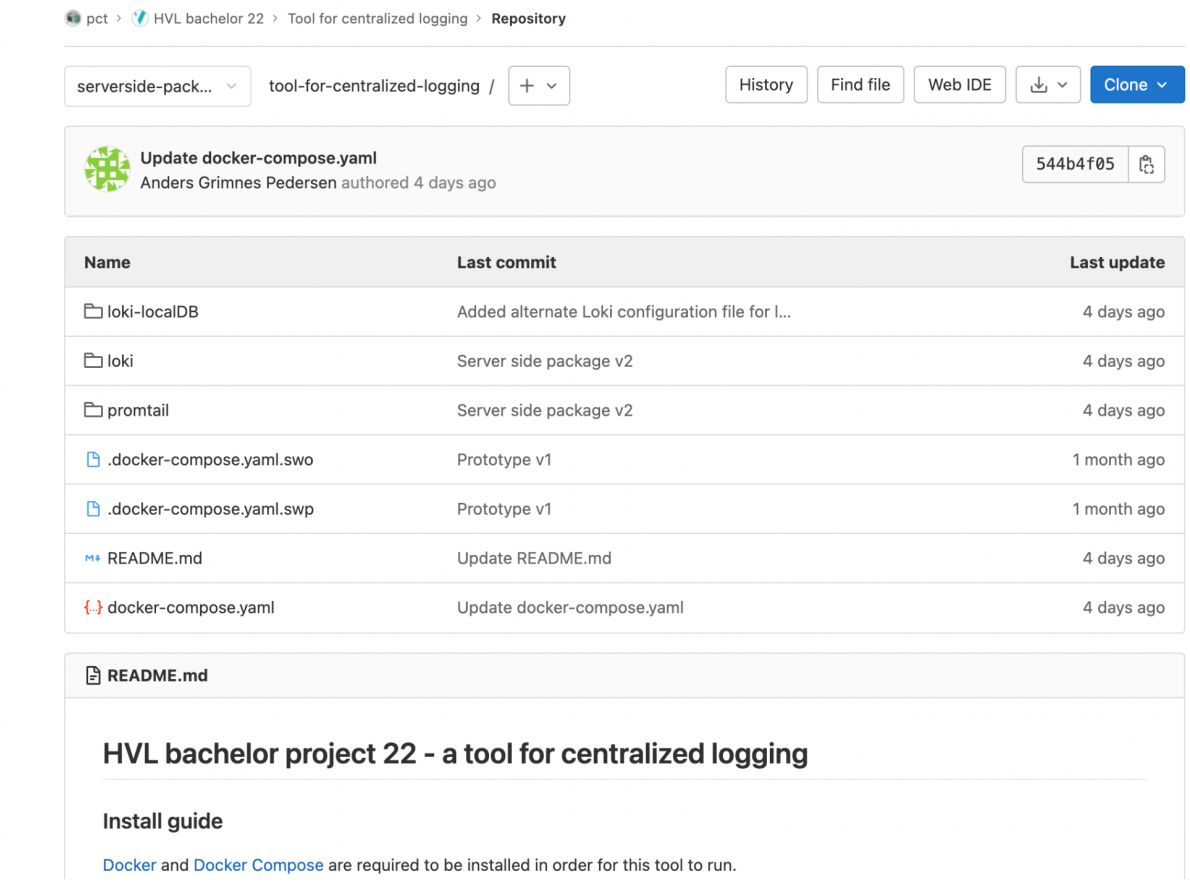
I GitLab finnes all kode og informasjon for det overordnede pCT prosjektet. Gruppen har en egnet mappe plassert inne i dette systemet med mappesti: *pct/HVL bachelor 22/Tool for centralized logging*. Der ligger all nødvendig informasjon for å bygge loggføringsverktøyet. Mer om dette i kapittel 8 - Dokumentasjon av kildekode.



Figur 6: GitLab repositorien for prosjektet med ulike branches



Figur 7: GitLab branch: clientside-package



Figur 8: GitLab branch: serverside-package

3.3 Kodestruktur

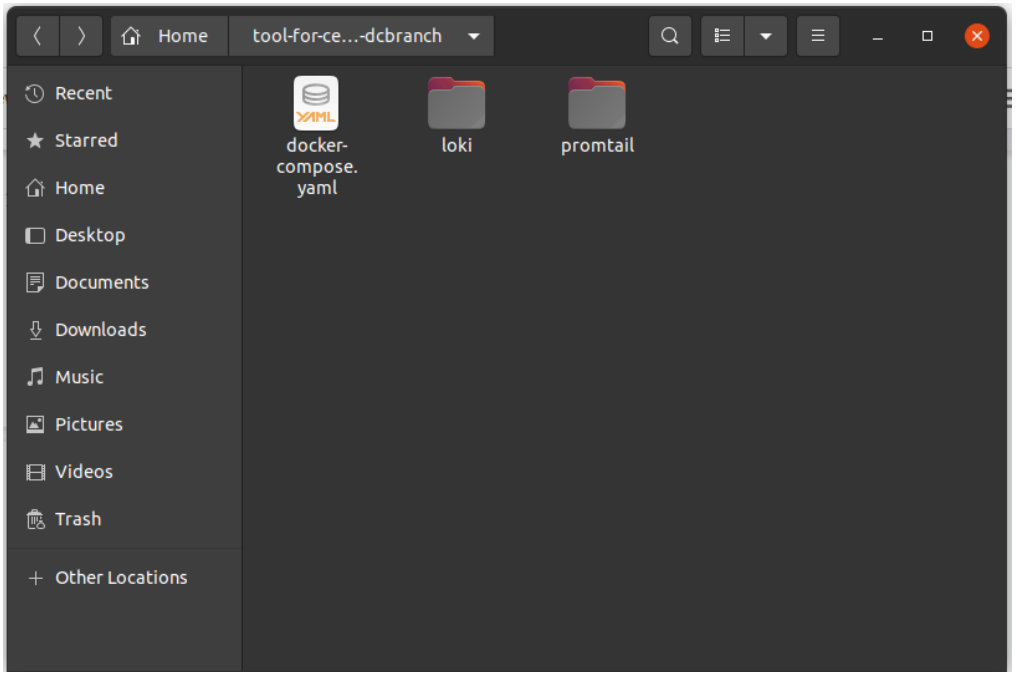
For at kommunikasjonen mellom hovedkomponentene skal fungere, trengs det konfigurasjonsfiler. Filene er av typen YAML og er plassert lokalt på maskinene i det distribuerte systemet. Disse kan hentes fra gruppens repository på GitLab. Nedenfor er YAML-filene som blir brukt i prosjektet.

- docker-compose.yaml
- promtail-config.yaml
- loki-config.yaml

Filene er fordelt på de ulike maskinene i det distribuerte systemet (se Figur 2) etter hvilket bruksområde de skal dekke. På servermaskinen er alle tre konfigurasjonsfilene plassert, mens nodene kun har konfigurasjonsfilen til Promtail.

3.3.1 Servermaskin

Mappen med de tre konfigurasjonsfilene er plassert lokalt på servermaskinen. Siden mappen kan hentes fra GitLab, er servermaskinen uavhengig og kan enkelt byttes ut ved behov. I undermappene /loki og /promtail ligger konfigurasjonsfilene til Loki og Promtail.



Figur 9: mappen som inneholder de tre konfigurasjonsfilene på servermaskinen

docker-compose.yaml

Docker-compose samler og kjører flere docker containere på en gang. De tre hovedkomponentene (Grafana, Loki og Promtail) finnes i ferdig konfigurerte docker containere før de blir bygget sammen i Docker-compose. Bildet under illustrerer hvordan docker-compose-config.yaml bygger disse containerne sammen.

```yaml
version: "3"

networks:
  loki:

#Defines all the services to run (the docker images to run), with a set of given runtime arguments
services:
  loki:
    image: grafana/loki:2.4.1  #Defines which image to run
    ports:
      - "3100:3100"  #Mapping a port to the container. Host IP, if not set, MUST bind to all network interfaces. Host and container MUST use equivalent ranges. Either specify both ports (HOST:CONTAINER), or just the container port. In the latter case, the Compose implementation SHOULD automatically allocate any unassigned host port. Always specified as a quoted string.
    volumes:
      - ./loki:/etc/loki  #Volumes are synchronised folders, shared between host and container. Host folder path is defined first, then volume folder path, as such; (HOST:CONTAINER)
    command: -target=all,table-manager
             -config.file=/etc/loki/loki-config.yaml  #overrides the the default command declared by the container image (i.e. by Dockerfile's CMD).
    networks:  #Defines the networks that service containers are attached to. Services communicate with each other through Networks. In this specification, a Network is a platform capability abstraction to establish an IP route between containers within services connected together
      - loki

  promtail:
    image: grafana/promtail:2.4.1
    volumes:
      - /var/log:/var/log
      - ./promtail:/etc/promtail
    command: -config.file=/etc/promtail/promtail-config.yaml
    networks:
      - loki

  grafana:
    image: grafana/grafana:latest
    volumes:
      - grafana-storage:/var/lib/grafana
    ports:
      - "3000:3000"
    networks:
      - loki


volumes:
  grafana-storage:
    external: true
```

Figur 10: illustrerer hvordan docker-compose.yaml er bygget

loki-config.yaml

Konfigurasjonsfilen til Loki inneholder informasjon om serveren og lagringsplassen.

```yaml
#Enables authentication through the X-Scope-OrgID header, which must be present if true.
#If false, the OrgID will always be set to "fake".
auth_enabled: false

#Defines the listening ports for Loki.
server:
  http_listen_port: 3100
  grpc_listen_port: 9096

#Sets common definitions to be shared by different components. This way, one doesn't have to replicate configs in multiple places.
common:
  path_prefix: /tmp/loki  #When this is defined this will be present in front of the endpoint paths.
  storage:
    filesystem:  #Configures storing the chunks on the local filesystem. Required fields only required when filesystem is present in config.
      rules_directory: /tmp/loki/rules
  replication_factor: 1  #How many times incoming data should be replicated to the ingester component.
  ring:  #A common ring configuration to be used by all Loki rings.
    instance_addr: 127.0.0.1 #IP address to advertise in the ring.
    kvstore:  #The key-value store used to share the hash ring across multiple instances.
      store: inmemory  #Backend storage to use for the ring. Supported values are; consul, etcd, inmemory, memberlist, multi.

#Configures the storage solution for the index store and the chunk store. These are seperate stores, unless defined different.
#Can be configured to use database storage, local file storage, or a combination
schema_config:
  configs:
    - from: 2021-04-26
      store: aws     #This is the store used for indexes. Indexes are used to access the chunk storage in the right storage space.
      object_store: s3  #This is the store used for chunks. Chunks are compressed log data.
      schema: v11
      index:
        prefix: loki_index
        period: 24h       #How long to store the indexes before deletion.

#This part is to tell Loki how and where to store indexes and chunks
storage_config:
  aws:
    s3: s3://********************** #Chunk storage access key for Amazon S3.
    dynamodb:
      dynamodb_url: dynamodb://******************* #Index storage access key for Amazon DynamoDB.

#Data transmission settings for database
table_manager:
  chunk_tables_provisioning:
    inactive_read_throughput: 1
    inactive_write_throughput: 1
    provisioned_read_throughput: 5
    provisioned_write_throughput: 5
  index_tables_provisioning:
    inactive_read_throughput: 1
    inactive_write_throughput: 1
    provisioned_read_throughput: 5
    provisioned_write_throughput: 5
  retention_deletes_enabled: false
  retention_period: 48h

#Ingester settings
ingester:
  chunk_idle_period: 5m #Define the idle period before a chunk is compressed and flushed to the database.
  chunk_retain_period: 30s #Define how long after the chunk is flushed, the chunk should be retained in memory.

ruler:
  alertmanager_url: http://localhost:9093 #Comma-separated list of Alertmanager URLs to send notifications to. Each URL is treated as a separate group in the configuration.
```

Figur 11: konfigurasjonsfilen til Loki

3.3.2 Servermaskin og klienter

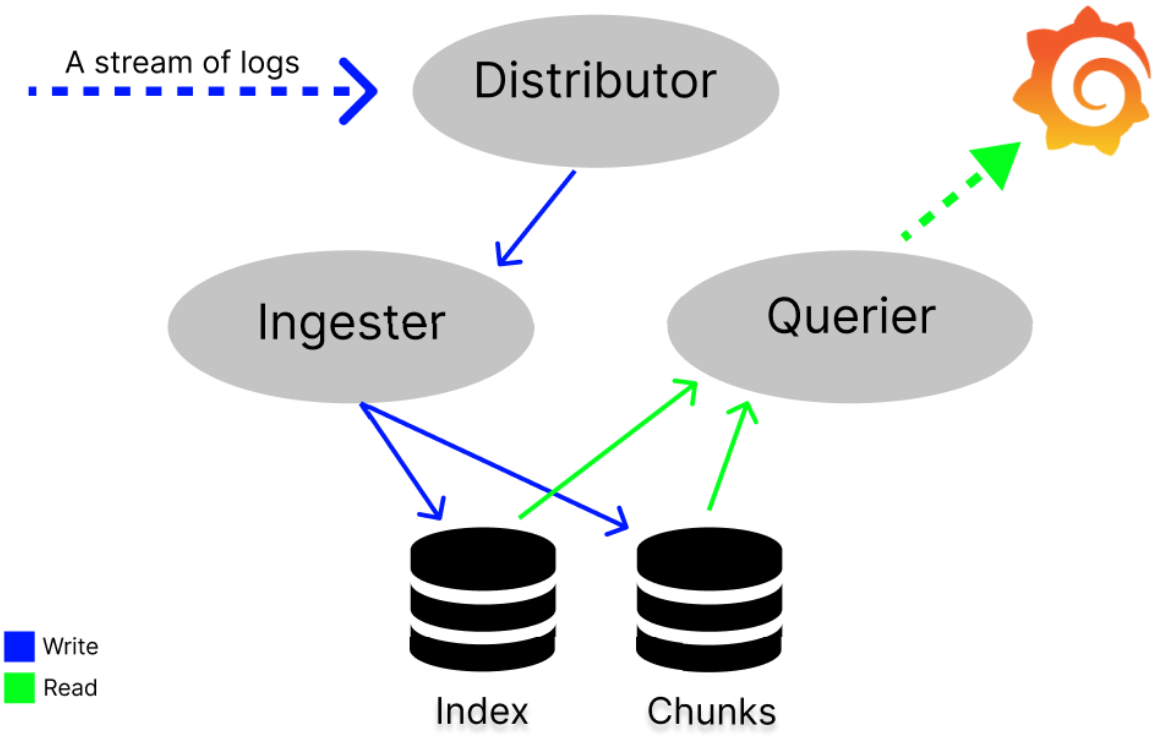
promtail-config.yaml

Enhver maskin i systemet har en Docker med Promtail innebygget. For at Docker containerne kan kjøre og skrive til APIet til Loki, trengs det en lokal konfigurasjonsfil på maskinen kalt promtail-config.yaml. Denne filen er identisk på alle maskinene.

```
#Define the ports for promatil. These are at default.
server:
  http_listen_port: 9080
  grpc_listen_port: 0

#Describes where to save read file offset to disk.
#Promtail will save a file indicating how far it has read into a file. It is needed for when Promtail
#is restarted to allow it to continue from where it left off.
positions:
  filename: /tmp/positions.yaml

#Define the url of the Loki instance you want promtail to forward the logs to. You can define multiple instances. You always connect to the Loki push API.
clients:
  - url: http://158.39.77.99:3100/loki/api/v1/push


#The scrape_configs block configures how Promtail can scrape logs from a series of targets using a specified discovery method
scrape_configs:
- job_name: system  #Name of this scrape config in the Promtail UI. This is at default
  static_configs:
  - targets:         # 'targets' can be discluded, it will automatically default to localhost, and
      - localhost    # should ONLY be defined as localhost if you chose to define it.
    labels:
      job: varlogs             #Define the label for a job. All logs retrieved from the '__path__' are labeled with 'varlogs' in this case.#In Grafana you can access all the log messages from this job if you query '{job="varlogs"}'. Every Logstream under this job will retrieve a label containing this job.
      __path__: /var/log/*log  #This is where you define the path to where you want to retrieve the log messages.
```

Figur 12: konfigurasjonsfilen til Promtail

4 DATABASEMODELL



Figur 13: illustrasjon av hvordan loggmeldingene blir lagt inn i databasen

Grafana Loki lagrer loggmeldinger i databasen i form av indekser og chunks. Loki tar inn loggmeldingene i separate streams der hver stream blir tildelt en stream ID og et sett med loggmeldinger. Disse streamene blir da komprimert og lagret som chunks. En merkelapp (label) blir opprettet for den chunken og blir lagret som en indeks. Indekser og chunks blir lagret i separate databaser (Grafana, u.å.b). Chunk-objektene blir lagret i Amazon S3, mens indeksene som inneholder labelen til chunkene blir lagret i Amazon DynamoDB.

5 Server-tjeneste

For å teste og kjøre det distribuerte systemet har gruppen satt opp en NREC server. NREC tilbyr infrastruktur som tjeneste (IaaS), også kalt skyinfrastruktur (NREC, u.å.a). NREC står for nettverket, serveren, lagringen og virtualiseringen, men gir brukeren tilgang til, samt håndteringen av disse instansene via skyen.

Server tjenesten og den virtuelle maskinen vi har benyttet, settes opp i NREC sitt dashboard, på deres nettside. Tilgangen og kontrollen av den ferdig konfigurerte serveren opprettes gjennom Linux terminalen (CLI). I Dette kapittelet beskrives oppsettet, tilgangen og hvordan serveren fungerer.

Alle bildene vist som illustrasjon i dette kapittelet er tatt i NREC dashboard (NREC, u.å.b)

SSH nøkkelpar

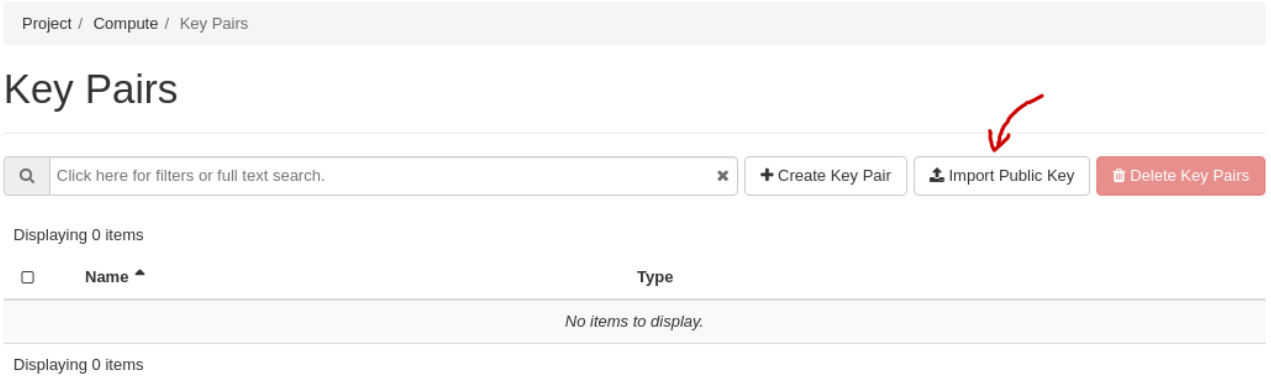
Virtuelle maskiner i NREC gis tilgang til gjennom et SSH nøkkelpar (NREC, u.å.c).

Opprett en lokal nøkkel i CLI:

```
$ cd ~
$ ssh-keygen -b 4096 -t rsa -f .ssh/id_rsa
```

Det trengs ikke å oppgi en passphrase selv om det blir prompted, bare trykk enter. Filene 'id_rsa' og 'id_rsa-pub' blir lagret i den skjulte mappen '.ssh'.

For å laste opp SSH nøkkelen til NREC profilen, i dashbordet, naviger til Project -> Compute -> Key Pairs. Velg "Import Public Key":

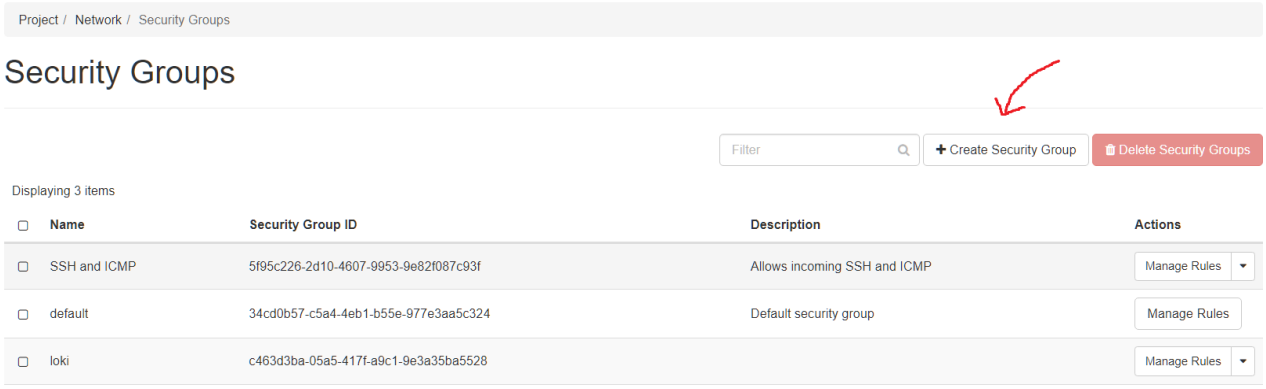


Fyll inn et valgfritt navn i "Key Pare Name". Velg SSH i "Key Type".
Lim inn innholdet til id_rsa.pub i "public Key", eller last opp dokumentet.

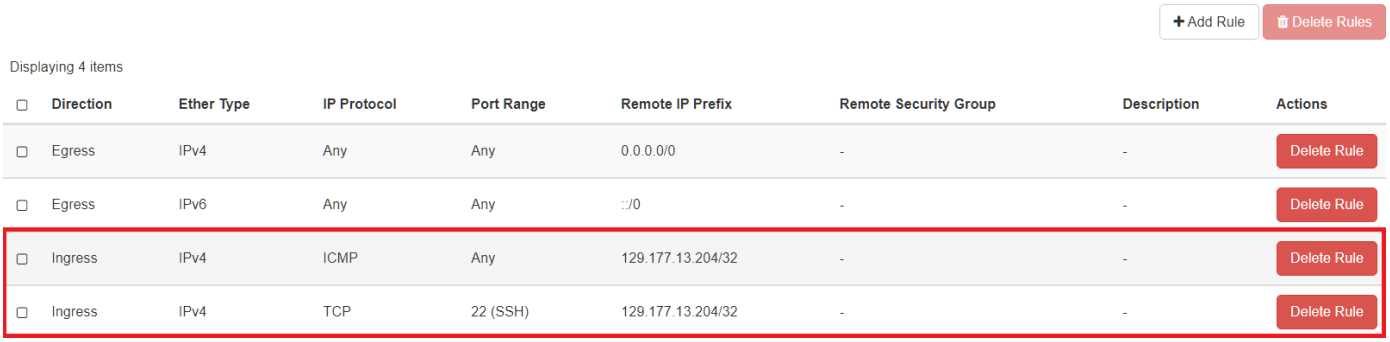
Opprett sikkerhetsgrupper

En sikkerhetsgruppe inneholder regler som definerer hvilken trafikk som er lov inn (ingress) og ut (egress) fra den virtuelle maskinen på serveren (NREC, u.å.c). Det følger med en standard sikkerhetsgruppe, men det trengs noen ekstra sikkerhetsgrupper for at alle instanser i loggføringsverktøyet skal kunne kommunisere med serveren. Det trengs også en sikkerhetsgruppe for kommunikasjon til UiB sitt påloggings API.

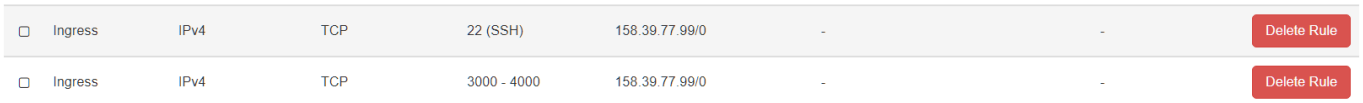
For å legge til en sikkerhetsgruppe, naviger til Project -> Network -> Security Groups i NREC dashboardet. Trykk på “Create Security Group”. Deretter gir du sikkerhetsgruppen et navn og en valgfri beskrivelse:



Det trengs SSH og ICMP (ping) tilgang fra host login.uib.no. Trykk på “Add rule” og legg til de to reglene som vist på bildet:



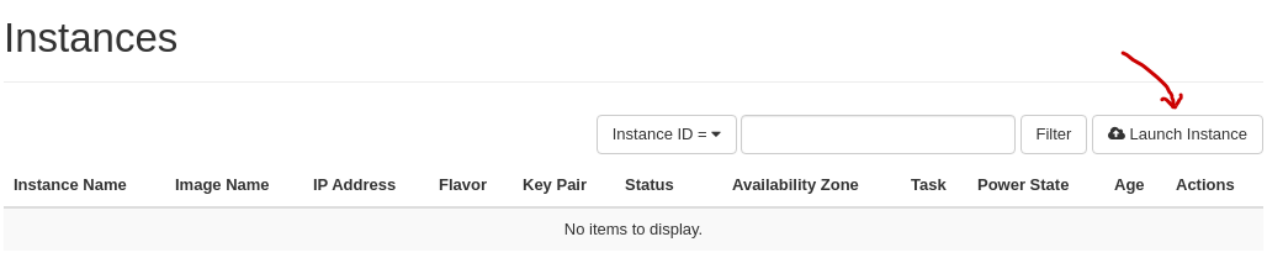
Det trengs å åpne for innkommende trafikk på port 3000 - 4000, slik at Promtail (kjørende fra eksterne maskiner) kan snakke med Loki (på serveren) sitt API og Grafana (på serveren). Trykk på “Add rule” og legg til de to reglene som vist på bildet:



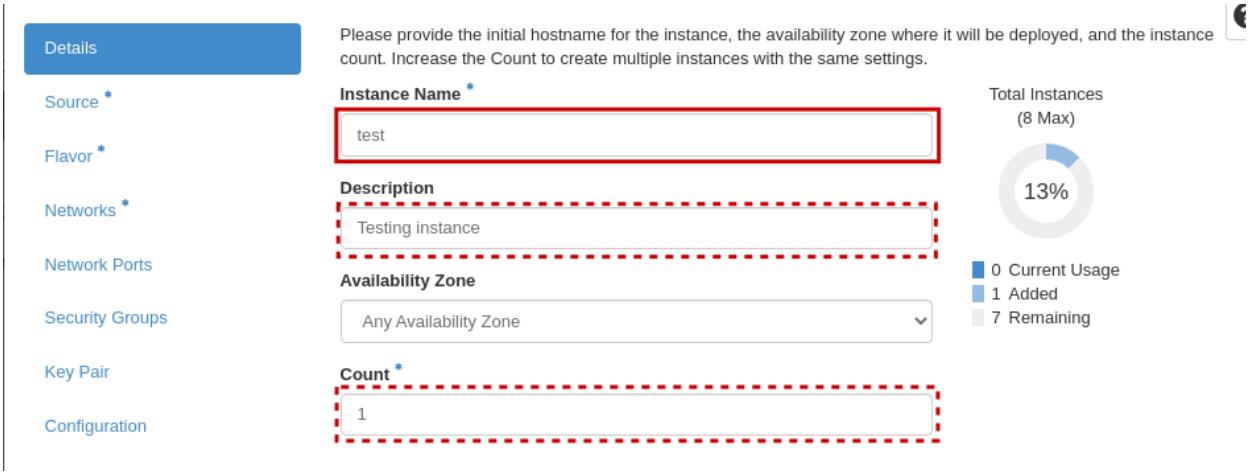
Opprette en virtuell linux maskin

For at serveren skal kunne tilby Grafana og Loki som tjeneste, trenger den et operativsystem (NREC, u.å.d). I dette tilfellet Ubuntu (linux-basert).

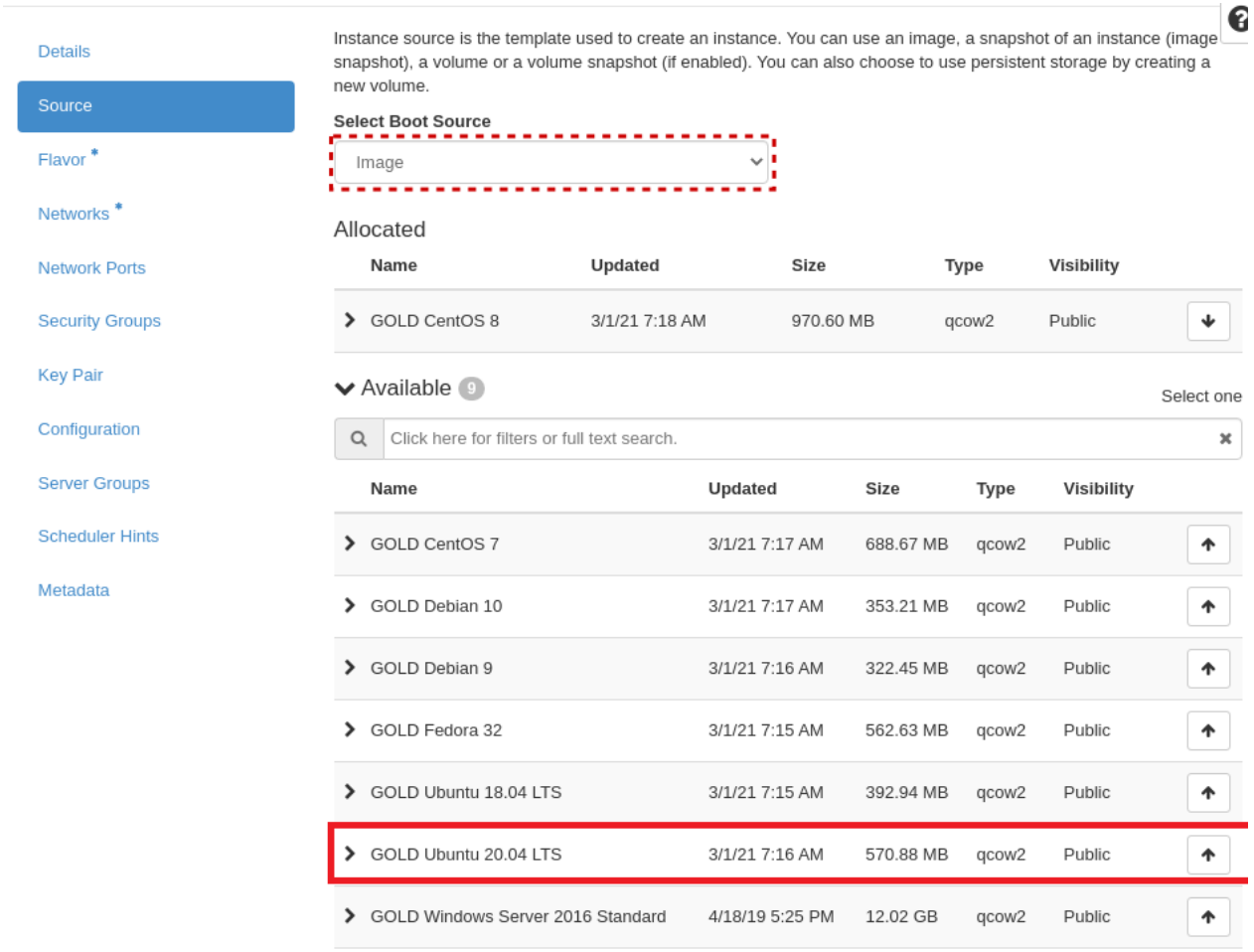
I NREC dashboardet naviger til Project -> Compute -> Instances. Trykk på “Launch Instance” :



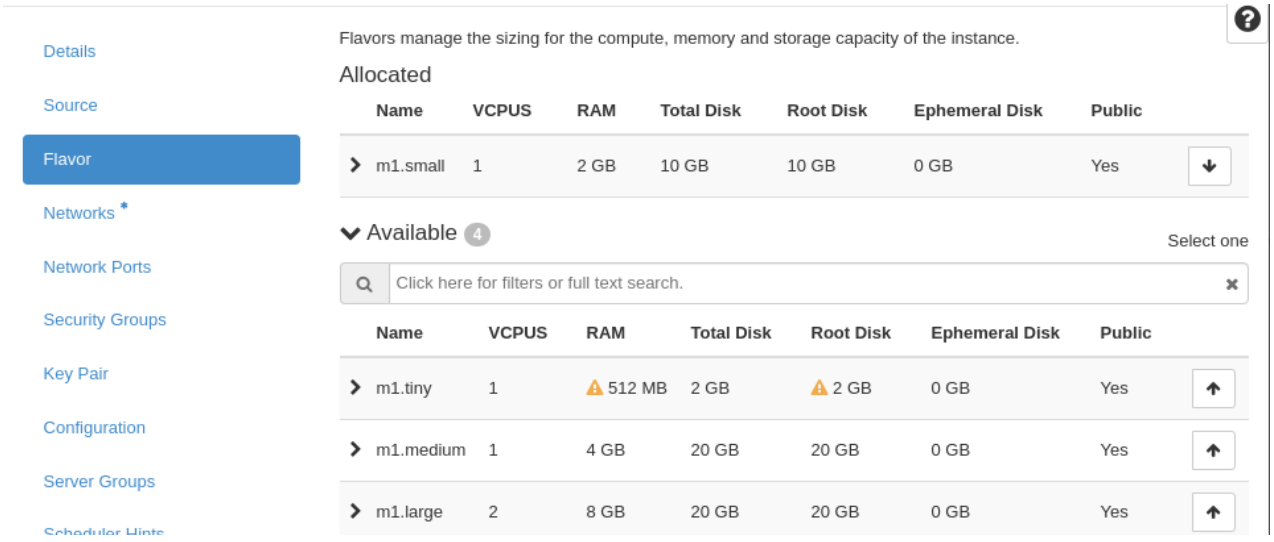
Oppgi et instans navn og en valgfri beskrivelse. La “availibility zone” feltet stå som den står, sist bestemmer “count” hvor mange virtuelle maskiner som ønskes å kjøre, la denne stå som 1 :



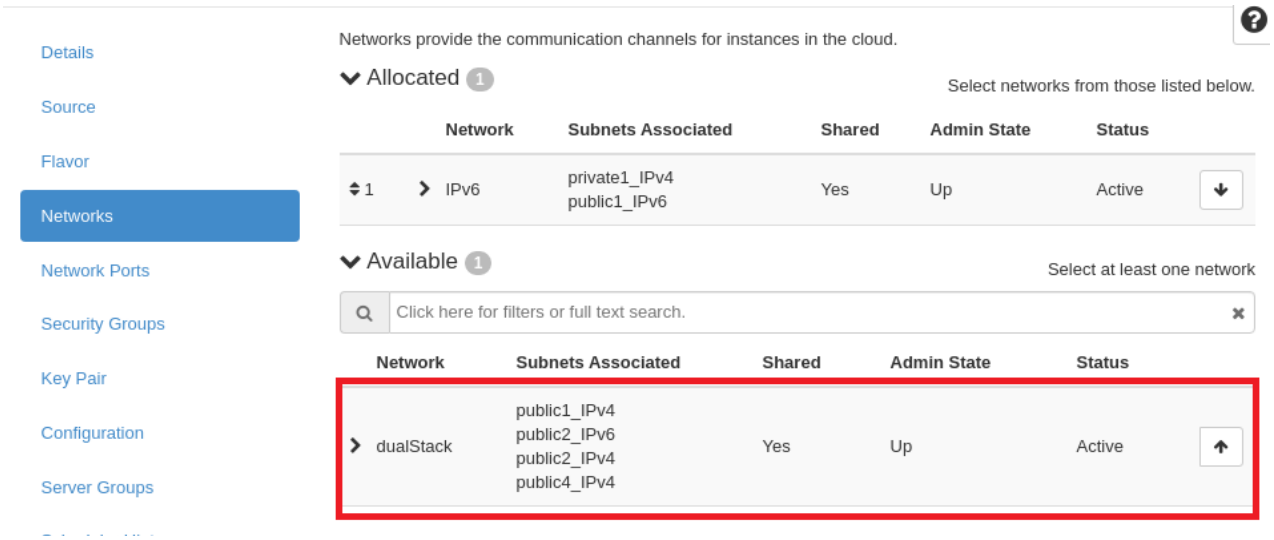
Naviger til “Source”, i menyen på venstre side. La “Select boot source” stå på “Image”. Deretter Velg “GOLD Ubuntu 20.04 LTS” imaget. Dette er operativsystemet den virtuelle maskinen kjører på.



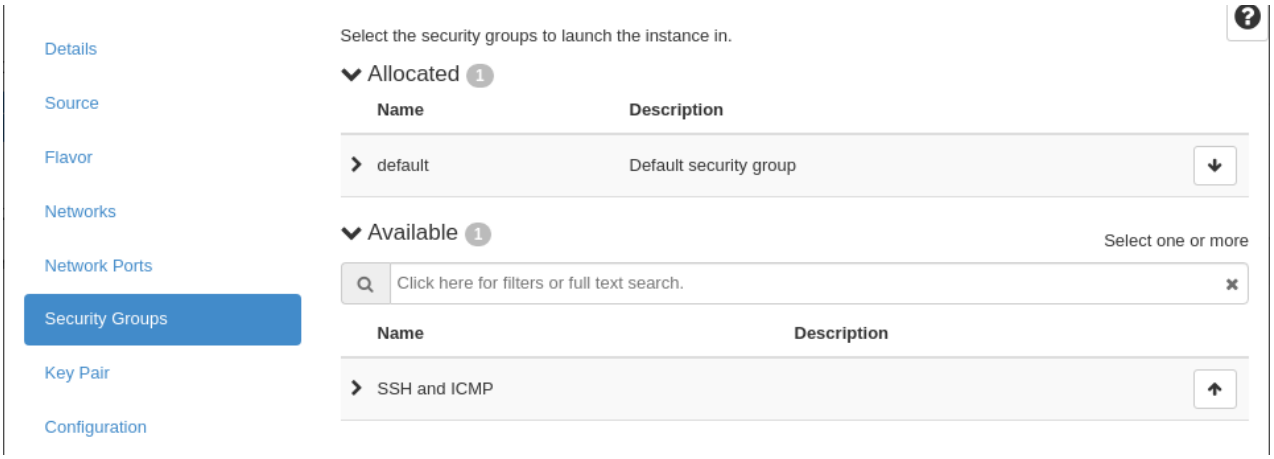
Naviger til “Flavor”, i menyen på venstre side. Velg “m1.small”, eller en flavor med mer prosessering, minne og lagringsplass enn det som kreves for Linux imaget.



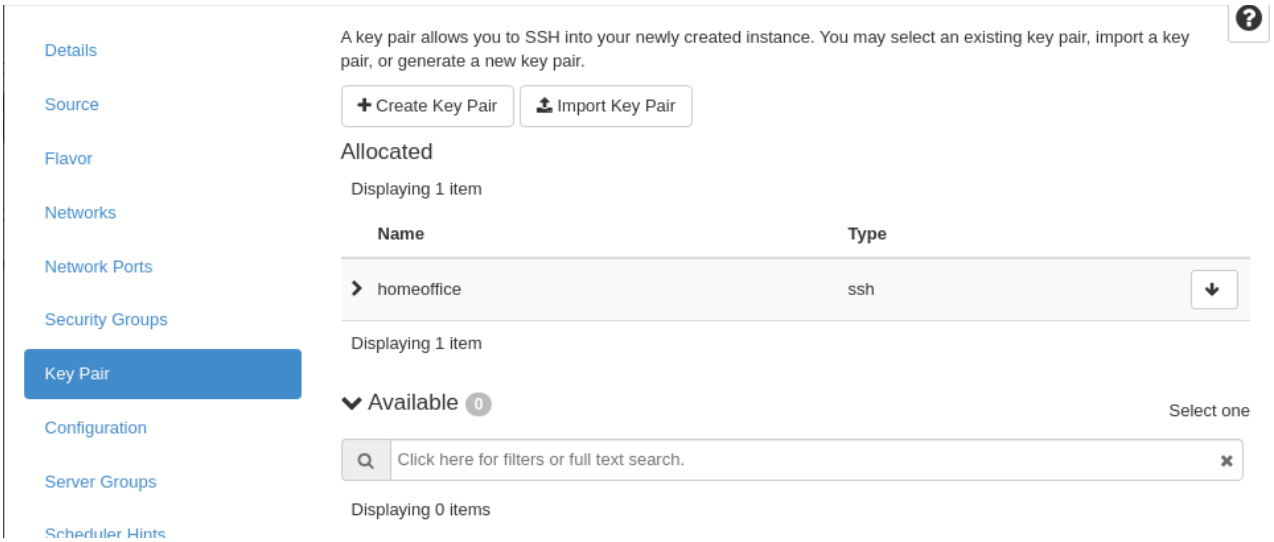
Naviger til “Networks”, i menyen på venstre side. Velg “dualStack”, som tilbyr en offentlige IPv4 og IPv6 adresse. Det er et problem i Linux som gjør at tilkobling via IPv6 ikke fungerer, derfor benyttes det en IPv4 adresse. Velg KUN dualStack, ikke både IPv6 og dualStack.



Naviger til “Security groups”, i menyen på venstre side. Legg til sikkerhetsgruppen som ble opprettet i forrige kapittel. La sikkerhetsgruppen “default” også være lagt til (dermed skal det totalt være 2 aktive sikkerhetsgrupper).



Naviger til “Key pair”, i menyen på venstre side. Legg til SSH nøkkelparet som ble opprettet i kapittelet ”SSH nøkkelpar”. Dette nøkkelparet kan IKKE endres etter at instansen er laget.



Trykk på “Launch Instance” for og fullføre.

Koble til serveren sin virtuelle maskin

All kommunikasjon med den virtuelle maskinen gjøres gjennom Terminal (CLI).

For å koble opp mot server:

```
$ ssh -J <UiB_brukernavn>@login.uib.no ubuntu@<serverIPv4_adresse>
```

Fyll inn Uib <brukernavn> og <IPv4 adresse>. Et eksempel:

```
ssh -J noen123@login.uib.no ubuntu@158.39.77.99
```

En prompt kommer opp, som etterspør passord. Dette er Uib passord til gjeldende bruker.

Den virtuelle maskinen på serveren kan nå opereres gjennom CLI.

Avslutt forbindelsen med kommandoen ‘exit’.

Kjøre Grafana, Loki og Promtail på serversiden

Følg installasjonsinstruksjoner for Docker og Docker Compose. Deretter følg installasjonsinstruksjonene av serverside-loggagregeringspakken. Denne pakken kan lastes ned i og kjøres fra alle disk plasseringer i den virtuelle maskinen (men ikke i Root). For å starte Grafana,

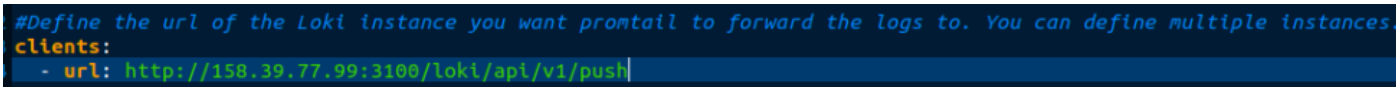
Loki og Promtail, naviger til mappen for docker-compose yaml-filen (Der du har lastet ned serverside-loggaggregeringspakken). Skriv inn i CLI for å kjøre alle instansene:

```
$ sudo docker-compose up
```

Når Grafana og Loki kjører på serveren, kan disse nås fra andre datamaskiner på IP 158.39.77.99 og portnumrene 3100 og 3000. Loki kjører på portnummer 3100 og Grafana på 3000.

For å komme seg inn i Grafana Dashboard på ekstern PC, skriv inn https://158.39.77.99/3000 i valgfri nettleser. En innloggingsside vil komme opp. Logg inn med admin bruker:
Brukernavn: admin
Passord: admin
Her vil du kunne se, behandle og søke i loggmeldinger fra alle noder koblet opp mot serveren.

Ved å laste ned klientside-loggagregeringspakken på en linux-maskin, vil Promtail allerede være ferdig konfigurert til å snakke med APIet til Loki på serveren. Dette er konfigurert i

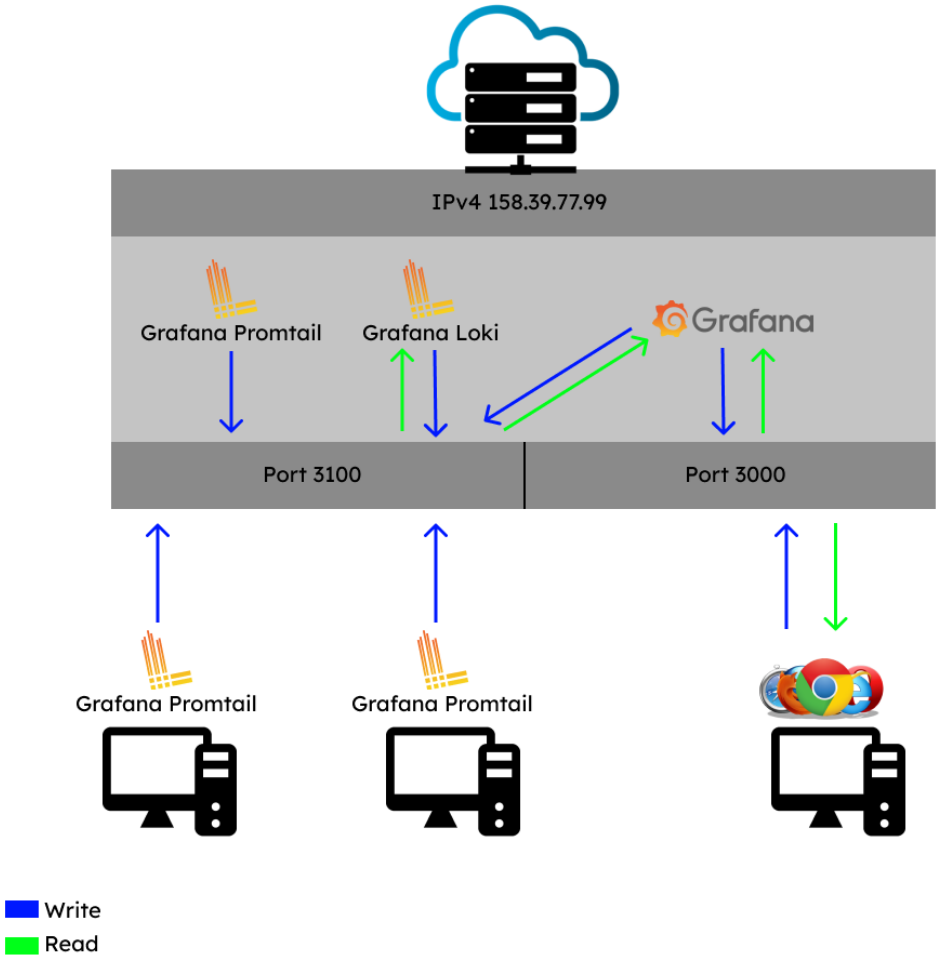


Figur 14: Kodesnutt 1 fra promtail-config.yaml filen

promtail-config.yaml filen:

Promtail vil sende loggmeldinger fra maskinen til serveren gjennom denne adressen. Ønskes det å konfigurere kommunikasjon med en annen server på en annen IP, bytt ut “158.39.77.99” med den ønskede servers IP.

Systemmodell for kommunikasjon via server



Figur 15: Serverkommunikasjon arkitektur

Kommunikasjon via server er illustrert i form av en modell. Det distribuerte systemet kommuniserer via serveren på IPv4 adresse 158.39.77.99. Serveren kjører en Promtail instans, som sender loggmeldinger til Grafana Loki sitt API på port 3100 internt på serveren. Serveren hoster Grafana Loki og Grafana. Grafana Loki kjører på port 3100 og leser alt av loggmeldinger som kommer inn på porten, både fra sin interne instans av Promtail og alle promtail instanser kjørende på alle noder koblet til serveren. Loki prosesserer disse loggmeldingene og pusher de ut igjen på port 3100.

Grafana kjører på port 3000 på serveren. Grafana innhenter de ferdig prosesserte loggmeldingene fra Loki på port 3100. Eksterne datamaskiner kan koble til serveren på port 3000 og visualisere dataene i Grafana, samt sende queries i grafana sitt brukergrensesnitt, som videreformidler disse queriene til Loki på port 3100.

6 INSTALLASJON OG KJØRING

De viktigste eksterne avhengighetene/programvarene som brukes i prosjektet er følgende:

- Docker og Docker-compose
- Grafana Loki
- Promtail
- Grafana
- Database

De fire første punktene er installert i et Linux miljø via command-line interfacet (CLI).

7.1 Docker og Docker Compose

Docker er en plattform der applikasjoner eller tjenester kan kjøres i containere. Her spesifiseres alt av nødvendig programvare for å kjøre tjenestene i en konfigurasjonsfil kalt Dockerfile. Dette er nyttig for å kjøre programmet på hvilken som helst maskin, uavhengig av operativsystem.

Docker-compose gjør det mulig å sette opp og kjøre flere containere samtidig. Da vil det fungere som en sammenhengende tjeneste bestående av flere mikrotjenester. Konfigurasjonen for alle containerne defineres i en YAML-fil som gjør at de kan startes med èn kommando (Docker, u.å.b).

Siden verktøyet for sentral loggføring i pCT består av flere tjenester er det gunstig å benytte Docker-compose for å samle de ulike containerne. For å laste ned Docker-compose må operativsystemet ha Docker Engine installert på Ubuntu. Begge lastes ned ved å skrive inn kommandoer i terminalen.

Docker Engine installasjon (Docker, u.å.c)**:**
Sett opp et oppbevaringssted

1. Oppdater *apt* pakkeindeksen og installer pakker for å tillate *apt* for å bruke et oppbevaringssted over HTTPS:

```
$ sudo apt-get update
$ sudo apt-get install \
                ca-certificates \
                curl \
                gnupg \
                lsb-release
```

2. Legg til Docker sin offisielle GPG key:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/usr/share/keyrings/docker-archive-keyring.gpg
```

3. Sett opp et stabilt oppbevaringssted:

```
$ echo \
  "deb [arch=$(dpkg --print-architecture)
signed-by=/usr/share/keyrings/docker-archive-keyring.gpg]
https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Installer Docker Engine

1. Oppdater *apt* pakkeindeksen og installer siste versjon av Docker Engine og containerd

   ```
   $ sudo apt-get update
   $ sudo apt-get install docker-ce docker-ce-cli containerd.io
   ```

2. Tester at Docker Engine er installert korrekt ved å kjøre *hello-world* image.

   ```
   $ sudo docker run hello-world
   ```

   Denne kommandoen søker lokalt etter 'hello-world' docker imaget, hvis det ikke finnes laster det ned et test image og kjører den i en container. Når containeren kjører, printer den ut en melding "Hello World!" og avsluttes.

Nå kan Docker Compose installeres.

Docker Compose installasjon (Docker, u.å.d)**:**

1. Last ned den siste stabile utgivelsen av Docker Compose:

   ```
   $ sudo curl -L
   "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
   ```

2. Legg til kjørbare tillatelser på binærfilen:

   ```
   $ sudo chmod +x /usr/local/bin/docker-compose
   ```

3. Test installasjonen:

   ```
   $ docker-compose --version
   ```

   Viser hvilken versjon av Docker Compose systemet har.

7.2 Installasjon av verktøyet for sentral loggføring

Installasjonsprosessen kan deles inn i to kategorier, serverside-installasjon og klientside-installasjon. Det kreves at både server og klient kjører i et linux miljø. En klient er en datamaskin (en node i det distrubuerte systemet) som genererer loggmeldinger.
Det kreves også at Docker og Docker Compose er installert både på server og klient.

Serverside-installasjon

Serverside-loggagregeringspakken kan lastes ned fra GitLab på URL:
https://git.app.uib.no/pct/hvl-bachelor-22/tool-for-centralized-logging/-/tree/serverside-package
Pakken kan installeres i en valgfri filplassering på serveren. Denne pakken er konfigurert til å kjøre på serveren slik at serveren hoster Grafana og Loki instansene, som alle nodene i det distribuerte systemet skal kobles opp mot. Denne pakken inneholder også en Promtail instans for loggovervåking av selve serveren.

Serverside konfigurasjon og oppstart

Naviger til filplasseringen der du lastet ned pakken. Det skal nå ligge en mappe der som heter "tool-for-centralized-logging". Denne mappen inneholder docker-compose.yaml, og igjen to mapper kalt "loki" og "promtail". Mappen "loki" inneholder "loki-config.yaml" filen. Mappen "promtail" inneholder "promtail-config.yaml" filen. Disse filene må ikke omplasseres (dette har med shared volumes å gjøre).

For og starte systemet åpne mappen "tool-for-centralized-logging" i terminal vinduet. Kjør kommandoen:

```
$ sudo docker-compose up
```

Promtail, loki og grafana vil automatisk starte.

I et førstegangsoppsett må det defineres en eksternt grafana lagringsplass. Kjør kommandoen:

```
$ sudo docker volume create --name=grafana-storage
```

Denne lagringsplassen blir brukt for å lagre oppsettet til Grafana Dashboard, slik at konfigurasjonen ikke går tapt når Docker Containeren avsluttes.

Du skal nå kunne gå inn i en nettleser på hvilken som helst PC, skrive inn serveren sin IPv4 adresse, samt port nummer 3000, for og få tilgang til Grafana. F.eks: https://158.39.77.99/3000.

Ved innloggingsforespørsel, er innloggingen:
Bruker: admin
Passord: admin

Klientside-Installasjon

klientside-loggagregeringspakken kan lastes ned fra GitLab på URL:
https://git.app.uib.no/pct/hvl-bachelor-22/tool-for-centralized-logging/-/tree/clientside-pack
Pakken kan installeres i en valgfri filplassering på maskinen. Denne pakken er konfigurert til å kjøre på en node (maskin) slik at den hoster en Promtail instans. Promtail instansen overvåker og leser ut loggmeldinger, som sendes videre til serveren.

Promtail er konfigurert til å lese ut loggmeldingene fra /var/log/ mappen, og alle filer som er av typen .log.

```
query '{job="varlogs"}'.
      __path__: /var/log/*log  #This is where you define the path to where you want to retrieve the log
messages.
```

Figur 16 Kodesnutt 2 fra promtail-config.yaml filen

En demo-test kan lastes ned fra GitLab på URL:
https://git.app.uib.no/pct/hvl-bachelor-22/tool-for-centralized-logging/-/tree/dcbranch

7.3 Grafana Loki og Promtail

Loki er et logg-aggregeringsverktøy som er laget for høye hastigheter med bruk av minst mulig ressurser (Rowe, 2020). Dette verktøyet er egnet for å evaluere loggdata i stor skala. Loki virker enten gjennom Grafana GUI eller med et command-line interface (CLI). Loggmeldingene blir tilegnet gjennom Promtail som sender til Loki for prosessering og lagring.

Promtail er verktøyet som overvåker og leser ut loggmeldinger fra en spesifisert lokal filkilde (f.eks /var/logs) og sender innholdet av lokale loggmeldinger til Loki (Grafana, u.å.c).

Installasjon av Loki og Promtail med Docker (Grafana, u.å.d)**:**

```
$ wget https://raw.githubusercontent.com/grafana/loki/v2.5.0/cmd/loki/loki-local-config.yaml
-O loki-config.yaml
docker run --name loki -d -v $(pwd):/mnt/config -p 3100:3100 grafana/loki:2.5.0
-config.file=/mnt/config/loki-config.yaml
$ wget
https://raw.githubusercontent.com/grafana/loki/v2.5.0/clients/cmd/promtail/promtail-docker-
config.yaml -O promtail-config.yaml
docker run --name promtail -d -v $(pwd):/mnt/config -v /var/log:/var/log --link loki
grafana/promtail:2.5.0 -config.file=/mnt/config/promtail-config.yaml
```

Nå er *loki-config.yaml* og *promtail-config.yaml* lastet ned på ønsket filplassering. Disse filene blir brukt til å kjøre Loki og Promtail i Docker containere.
Output kan ses på http://localhost:3100/metrics

Installasjon med Docker Compose

```
$ wget
https://raw.githubusercontent.com/grafana/loki/v2.5.0/production/docker-compose.yaml -O
docker-compose.yaml
docker-compose -f docker-compose.yaml up
```

Nå er *docker-compose.yaml* opprettet og kan brukes for å spesifisere de ulike containerne, slik det er illustrert på figur 7.

Den sammensatte løsningen i Docker Compose kjøres ved å skrive "docker-compose up" i terminalen. Da starter alle Docker containerne i ett og samme nettverk - loki (definert under "networks", se figur 7).

7.4 Grafana

For å visualisere innholdet i loggmeldingene brukes Grafana som et webgrensesnitt. Ved å spesifisere containeren til Grafana i Docker-compose (se figur 7), vil Grafana-imaget lastes ned og kjøres på localhost:3000.

7 REFERANSER

Docker (u.å.a) *Use Docker Compose.* Tilgjengelig fra: https://docs.docker.com/get-started/08_using_compose/ (Hentet 30. mars)

Docker (u.å.b) *Overview of Docker Compose.* Tilgjengelig fra: https://docs.docker.com/compose/ (Hentet: 30. mars 2022)

Docker (u.å.c) *Install Docker Engine on Ubuntu.* Tilgjengelig fra: https://docs.docker.com/engine/install/ubuntu/ (Hentet:30. mars 2022)

Docker (u.å.d) *Install Docker Compose.* Tilgjengelig fra: https://docs.docker.com/compose/install/ (Hentet: 30. mars 2022)

Grafana (u.å.a) *Components.* Tilgjengelig fra: https://grafana.com/docs/loki/next/fundamentals/architecture/components/ (Hentet: 26. april 2022)

Grafana (u.å.b) Storage. Tilgjengelig fra: https://grafana.com/docs/loki/latest/storage/ (Hentet 21. april)

Grafana (u.å.c) *Promtail.* Tilgjengelig fra: https://grafana.com/docs/loki/latest/clients/promtail/ (Hentet: 24. mars 2022)

Grafana (u.å.d) *Install Grafana Loki with Docker or Docker Compose.* Tilgjengelig fra: https://grafana.com/docs/loki/latest/installation/docker/ (Hentet: 24. mars 2022)

Lewandowski, A. (2021) *Logs analytics at scale in the cloud with Loki.* Tilgjengelig fra: https://getindata.com/blog/logs-analytics-scale-the-cloud-loki/ (Hentet: 26. april 2022)

Lukems (2021) *Server monitoring with Grafana, Prometheus and node_exporter*. Tilgjengelig fra: https://lukems.com/en/server-monitoring-with-grafana/ (Hentet: 04. april 2022)

NREC (u.å.a) *What is NREC*. Tilgjengelig fra: https://www.nrec.no/ (Hentet 24.04.2022)

NREC (u.å.b) *Dashboard.* Tilgjengelig fra: https://dashboard.nrec.no/dashboard/ (Bilder tatt 24.04.2022)

NREC (u.å.c) *Security Groups.* Tilgjengelig fra: https://docs.nrec.no/security-groups.html (Hentet 24.04.2022)

NREC (u.å.d) *Create a Linux Virtual Machine.* Tilgjengelig fra: https://docs.nrec.no/create-virtual-machine.html (Hentet 24.04.2022)

Rowe, W. (2020) *Follow this Grafana Loki tutorial to query IT log data.* Tilgjengelig fra: https://www.techtarget.com/searchitoperations/tutorial/Follow-this-Grafana-Loki-tutorial-to-query-IT-log-data (Hentet: 24. mars 2022)

Spring Cloud (2022) *SpringBoot integration of leightweight logging system loki- 2.* Tilgjengelig fra: https://www.springcloud.io/post/2022-02/springboot-loki-2/#gsc.tab=0 (Hentet: 26. april 2022)