Høgskulen på Vestlandet

BACHELOROPPGAVE

Verktøy for monitorering av kantenheter

Tool for edge monitoring

Gruppe D39
Egil Stahl Løvold
Eirik Myreng Hauge
Jaran Vasstveit

DAT191

Fakultet for ingeniør- og naturvitenskap
Institutt for datateknologi, elektroteknologi og realfag

Veileder: Harald Soleim

Innleveringsdato: 22.Mai 2023

Rapportens tittel Verktøy for monitorering av kantenheter	Dato: 22.05.2023
Forfatter(e): Egil Stahl Løvold, Eirik Myreng Hauge, Jaran Vasstveit	Antall sider u/vedlegg: 50 Antall sider vedlegg: 74
Studieretning: Dataingeniør	Antall disketter/CD-er:
Kontaktperson ved studieretning: Harald Soleim	Gradering: Ingen

Oppdragsgiver: Counting Hero	Oppdragsgivers referanse:
Oppdragsgivers kontaktperson: Ruben Patel	Telefon:

Sammendrag:

Kontroll over distribuerte enheter blir stadig mer relevant. Med den økende bruken av kantprosessering har det å oppnå effektiv monitorering og behandling av data blitt en nøkkelutfordring. Counting Hero utvikler spesialiserte systemer for deteksjon og klassifisering av video- og lyddata, primært fra sensorer som brukes til trafikkovervåkning langs veiene. I denne sammenhengen er det avgjørende med kontinuerlig oversikt over sensorenes helse og dataene som samles inn slik at tiltak kan iverksettes umiddelbart for å minimere nedetid.

Bachelorgruppen har fått i oppgave å lage et verktøy for dette. Denne rapporten beskriver problemstillingen, design av prosjektet, detaljert løsning, resultater, diskusjon og konklusjon.

Abstract:

Control over edge devices is becoming increasingly relevant. With the growing use of edge processing, achieving efficient monitoring and processing of data has become a key challenge. Counting Hero develops specialized systems for detection and classification of video and audio data, primarily from sensors used for traffic monitoring along roads. In this context, it is crucial to have continuous oversight of the sensors' health and the data being collected, enabling immediate action to minimize downtime.

The Bachelor group has been assigned the task of creating a tool for this purpose. This report describes the problem statement, project design, detailed solution, results, discussion, and conclusion.

Stikkord:

Sensor	Edge	Monitorering

Høgskulen på Vestlandet, Fakultet for ingeniør- og natuvitskap
Postadresse: Postboks 7030, 5020 BERGEN Besøksadresse: Inndalsveien 28, Bergen
Tlf. 55 58 75 00 Fax 55 58 77 90 E-post: post@hvl.no Hjemmeside: http://www.hvl.no

FORORD

Denne bacheloroppgave er gjennomført ved IDER fakultet ved HVL i sammarbeid med Counting Hero. Rapporten beskriver arbeidet gjort i prosjektet "Verktøy for monitorering av kantenheter" og er skrevet av Eirik Myreng Hauge, Jaran Vasstveit og Egil Stahl Løvold.

Gruppen er takknemlig for å kunne delta i en bedrift som Counting Hero der fokuset er utvikling av nye løsninger. Gjennom vårt bidrag håper vi at Counting Hero kan bruke mindre tid på feilretting og mer tid på utvikling av morgendagens løsninger. Samtidig har prosjektet vært enormt god læring og vi takker for muligheten og oppfølgingen vi har fått og en trivelig bachelorplass.

Videre ønsker vi å takke vår veileder Harald Soleim for gode tilbakemeldinger og meningsfulle innspill i løpet av prosjektperioden. Vi ønsker også å takke vår eksterne veileder Ruben Patel for hjelp, bidrag og tilbakemeldinger som har formet utviklingen.

ORDLISTE

ORD	Forklaring/Definisjon
Alerts Manager	Monitoreringsløsning i InfluxDB
Application Programming Interface (API)	Gjør det mulig for ulike programvarer å kommunisere med hverandre, for eksempel dataoverføring.
Bucket	Lokasjon for lagring i InfluxDB
CH	Counting Hero
Dashboard	Ulike sider i Grafana
Flux	Funksjonelt spørrespråk i InfluxDB
Grafana	Visualiseringsverktøy
Hypertext Transfer Protocol (HTTP)	En protokoll for kommunikasjon mellom klienter og servere på internett
InfluxDB	Tidsseriedatabase
Kantenhet (Edge)	Ytterste delen av et system eller nettverk der data skapes og behandles.
Kontekstdata	Samlet data fra flere sensorer over et lengre tidsperspektiv.
Line Protocol	InfluxDB dataformat
Measurement	Del av Line Protocol
Panel	Foranderlige visualiseringstyper i Grafana
Tidsserie, Tidsseriedata	Tidsmerket data

FIGURLISTE

Innholdsfortegnelse

1 Innledning

De senere år har bruken av kantprosessering i distribuerte systemer økt betydelig (Khan, et al., 2023). Distribuerte systemer består av to eller flere maskiner med kommunikasjon over et nettverk. Fordeling av arbeidsmengde og ansvarsområde bidrar til effektiv ressursbruk og logisk atskillelse av prosesser (IBM, 2021). I motsetning til skytjenester der data behandles sentralt, kan data prosessers i kant nærmere opprinnelsessted. Behovet for dataoverføring minsker og resulterer i frigjøring av ressurser og raskere responstid i applikasjoner. Når enheter står i kant, oppstår utfordringer tilknyttet feilhåndtering og feilsøking. Kompleksiteten av oppdagelse og håndtering av feilsituasjoner øker med antall enheter i systemet (Lin & Chen, 1997), prosjektgruppen har forsøkt å forenkle dette. I dette kapitelet introduseres bakgrunnen for prosjektet. Delkapitlene beskriver kontekst, motivasjon, prosjektets eier og problembeskrivelse og mål.

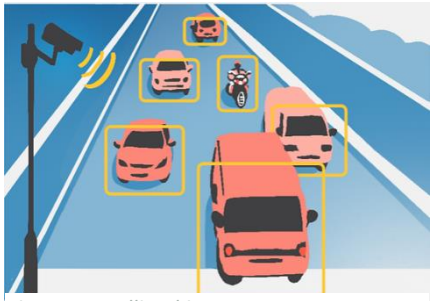
1.1 Counting Hero

Counting Hero (CH) er et software-as-a-service-selskap etablert i 2019 og eier bachelorprosjektet. CH leverer systemer for klassifisering og deteksjon i data fra sensorer langs vei, hovedsakelig i bildedata. Maskinlæringsmodeller tilgjengeliggjør informasjon øyeblikkelig med lav feilmargin og uten behov for å stoppe trafikken.

Tjenestene CH leverer deles i tre underkategorier:

- Personteller til ferge (figur 1.1):
    - Siden betalingsløsningen på fergene ble automatisert strever fergeoperatører med telling av personer om bord. Dette er fergeoperatørene er av sikkerhetsmessige årsaker pålagt å vite. Ved bruk av kameraer og maskinlæring gir systemet til CH fullstendig oversikt for operatørene ved av- og påstigning.
- Telling kjøretøy (figur 1.2):
    - Klassifisering og deteksjon av kjøretøy gir innsikt i verdifull statistikk. CH teller ikke bare kjøretøy, men også annen nyttig informasjon som lengde og type.
- Dekkavlesning (figur 1.3)
    - Feil eller mangler på dekk kan være livsfarlig. Feil dekk til årstid, feil type dekk til type kjøretøy og generell slitasje går utover trafikksikkerheten. CH sine systemer leser automatisk informasjonen og varsler ved feil, slik reduseres behovet for manuell inspeksjon.

Figur 1.1: Personteller til ferge



Figur 1.1: Telling kjøretøy

Figur 1.3: Dekkavlesning

Tjenestene avhenger av lav responstid, dataprosesseringen gjøres derfor på enheter plassert i felten, fysisk tilgang er derfor begrenset.

1.2 Motivasjon

Motivasjonen for prosjektet er å automatisere monitorering av enheter i kant. CH er avhengig av oppetid på kantenheter. Fungerer ikke enhetene som de skal, fungerer ikke tjenestene. Utviklere i CH ser at feil på kantenheter tar tid å oppdage og ofte kunne vært unngått ved monitorering av verdier som minnebruk, prosessorbruk og temperatur. Manuell overvåkning av verdier for enkelte enheter er gjennomførbart, men er tidkrevende, lite oversiktlig og ikke skalerbart.
Motivasjonen for prosjektet er å automatisere monitorering av enheter i kant. CH er avhengig av oppetid på kantenheter. Fungerer ikke enhetene som de skal, fungerer ikke tjenestene. Utviklere i CH ser at feil på kantenheter tar tid å oppdage og ofte kunne vært unngått ved monitorering av verdier som minnebruk, prosessorbruk og temperatur. Manuell overvåkning av verdier for enkelte enheter er gjennomførbart, men er tidkrevende, lite oversiktlig og ikke skalerbart.

1.3 Problembeskrivelse og mål

Problembeskrivelse
I nåværende system må utviklere i CH fjernstyre maskiner for å få tilgang på enhetsdata. Monitorering og feilsøking krever tilgang på data og gjennomføres ved analyse av tidsbegrensede systemlogger (100 minutter). Prosessen er tidkrevende og kvaliteten svekkes av manglende kontekst for dataene. Kontekstdata inkluderer data fra andre sensorer på enhet over et lengre tidsperspektiv. Kontekstdata synliggjør og øker forståelse for hvorfor feil oppstår (Sisense Inc, n.d.). Manuell datainnhenting og manglende kontekst resulterer i uforholdsmessig tidsbruk på feilsøking.

Hvordan kan innsamling, visualisering og monitorering av enhetsdata forhindre nedetid?

Mål
Målet med oppgaven er å utvikle et verktøy som reduserer nødvendig tid for feilsøking på enheter. Verktøyet skal samle data fra forskjellige enheter i en webapplikasjon. Applikasjonen skal tillate visning av tilstand for enheter og systemet som helhet. Utvikler skal kunne filtrere på enheter, tid, verdier i enhetsdata og attributter. Utforming av webapplikasjonen skal være gjort i samråd med oppdragsgiver for at verktøyet dekker reelle behov. Oppdragsgiver skal kunne bygge videre på gruppens arbeid, verktøyet skal derfor være skalerbar og ha høy foranderlighet.

1.4 Oppbygging av rapporten

Rapporten er skrevet med utgangspunkt i at leser har teknisk forståelse relevant til feltet. Det antas at leser har kjennskap til utrykk og begreper tilknyttet programmering og distribuerte system. Ruben Patel er både prosjekteier og bruker, men refereres til som prosjekteier for å unngå forvirring. Oppgaven er delt inn i ni kapitler. Under er oversikt for innhold i hvert kapittel:

Kapittel 2 inneholder prosjektets bakgrunn, avgrensninger for oppgaven, ressurser. I kapittelet informeres leser om eiers behov og hvordan det dekkes.

Kapittel 3: inneholder prosjektets design. En gjennomgang av forslag til løsning, valgt løsning, valg av verktøy, prosjektmetodikk og evalueringsplan skal gi leser innsikt i beslutninger tatt underveis med begrunnelser og vurderinger.

Kapittel 4: inneholder en detaljert beskrivelse av den tekniske løsningen. InfluxDB, Grafana og dataoverføringen fra kant.

Kapittel 5: inneholder resultatene fra evaluering i henhold til evalueringsplanen i kapittel 3.

Kapittel 6: inneholder diskusjon av hvilke konsekvenser arbeidsprosess og verktøy har hatt på resultatene i kapittel 5.

Kapittel 7: inneholder konklusjon for grad av måloppnåelse med bakgrunn i problembeskrivelse og mål fra kapittel 1 med forslag til videre arbeid.

Kapittel 8: inneholder bibliografi.

Kapittel 9: inneholder vedlegg til rapporten.

2 Prosjektbeskrivelse

CH hadde en åpen oppgave med frihet til valg av problemområde. Agenda for de første møtene var i hovedsak kartlegging av mulige problemstillinger. Prosjekteier hadde forslag til forskjellige områder:

- Trening/oppdatering av maskinlæringsmodeller
- Videreutvikling av brukergrensesnitt i interne systemer
- Monitorering av kantenheter

Dette kapittelet beskriver prosessen fra et åpent prosjekt til en spesifikk problemstilling og generell løsning. Prosjektets bakgrunn og eiers daværende system utdypes. Avgrensninger og ressurser presenteres. Til slutt får leser en kort introduksjon i monitorering og tidsseriedata.

2.1 Prosjektets bakgrunn og avgrensninger

Systemet til CH består av flere komponenter i samhandling med hverandre. I startfasen av prosjektet måtte gruppen identifisere problemstillinger innenfor de forskjellige komponentene. Tiden brukt på kartlegging av problemstilling står tydeligere beskrevet i timelistene og GANTT-skjemaet i prosjekthåndboken. De overnevnte områdene hadde fordeler og ulemper i kontekst av en bacheloroppgave.

I tabellen nedenfor (tabell 2.1) er hvert område vurdert på fem kriterier:

- **Tidligere erfaring:** gruppens erfaring med liknende teknologi. Her vektlegges fag og gruppens egne prosjekter.
- **Fleksibilitet i evaluering:** hvilke alternativer har gruppen i evaluering.
- **Tilpasningsbehov i dagens system:** i hvilken grad må løsningen tilpasse seg dagens system. Dette innebærer frihet i teknologivalg og om CH har tidligere arbeid på område.
- **Skaleringspotensiale:** oppgaven kan begrenses eller utvides basert på prosjektets framgang.
- **Nytte CH:** anslått nytteverdi prosjektet har for CH. Nytteverdien baseres på samtaler med prosjekteier og eksisterende løsninger i CH.

Tabell 2.1 Vurdering valg av oppgave:

Kjerneområdene	Tidligere erfaring	Fleksibilitet evaluering	Tilpasningsbehov system	Skaleringspotensiale	Nytte CH
Trening/Deployment modeller					
Videreutvikling grensesnitt					
Monitorering					

Det er brukt fargekoding, der grønn er fordel, gult er middels og rød er ulempe. Gruppen har gjennom valgfag tidligere erfaring med alle områdene.

CH har allerede løsninger i eget system for trening og utrulling av modeller. Nytteverdien ville vært noe begrenset fordi målet er forbedring av eksisterende løsning. Hovedfordelen med modellarbeid var evalueringsprosessen. Endring i modellytelse er konkret og målbart, enten økt presisjon eller responstid. Det eksisterer utallige evalueringsmetrikker med forskjellige formål (Zhou, et al., 2021). Prosjekteier anbefalte ikke modellarbeid med hensyn på skalering. Trening og evaluering av modellene er tidkrevende. Hvis gruppen startet dårlig, kunne endring av tilnærming hvert vanskelig.

Både videreutvikling av grensesnitt og monitorering er mindre konkret å evaluere sammenlignet med kvantitative evalueringsmetoder for modellarbeid. Hvordan evaluerer gruppen om en monitoreringsløsning eller et brukergrensesnitt er bra? Ettersom CH manglet monitoreringsløsning kunne gruppen bestemme verktøy og utforming av løsning selv, men videreutvikling av grensesnitt måtte gjøres på eksisterende løsning. Fordelene med den ene var svakhetene til den andre. Fritt valg av verktøy medfører risikoen at gruppen gjør en feilvurdering i valg av verktøy, som påvirker resultatet negativt fordi verktøyet ikke passer til problemstillingen. Ved valgt verktøy tar løsningen utgangspunkt i teknologi som allerede brukes.

Grunnet høy frihet i utforming, godt skaleringspotensiale og høy nytteverdi for CH, valgte gruppen monitorering.

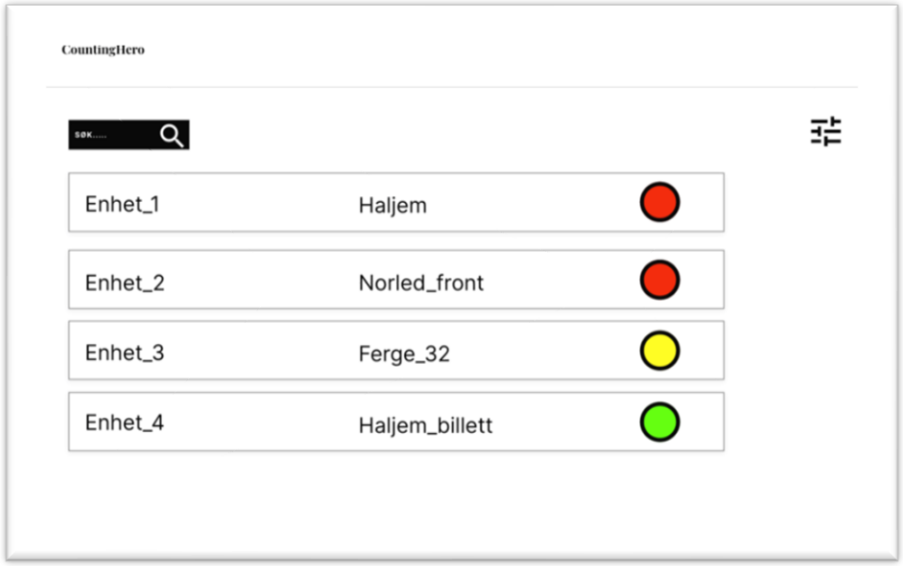
2.1.1 Initiale krav

CH trenger innsikt i helsen på enhetene og metrikker. Ved feil kan CH raskt iverksette tiltak og dermed redusere nedetid. Funksjonelle egenskaper til systemet:

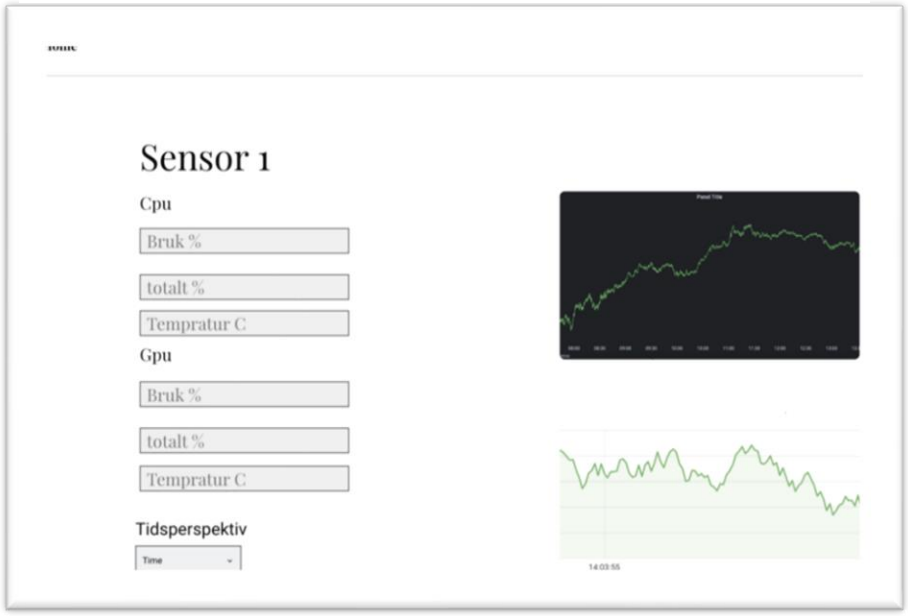
- Lagre måledata
- Varsle ved feil
- Varsle ved måledata utenfor verdiområder
- Søkefunksjon for enhet
- Visualisere metrikker
- Skaleringsmulighet (flere enheter og metrikker.

2.1.2 Initial løsnings-ide

Gruppen hadde frihet til å velge utviklingsmetode og verktøy i henhold til funksjonelle egenskaper for prosjektet beskrevet i visjonsdokumentet. De funksjonelle kravene ble delt inn i frontend (webgrensesnittet) og backend (datalagring og monitorering). Figur 2.1 og 2.2 er fra prototype versjon 1. Webgrensesnittet skulle vise enhetene med status som i figur 2.1 og metrikker for enkeltenheter som i figur 2.2



Figur 2.2: Prototype ver.1, forside



Figur 2.1: Prototype ver.1, metrikker for enkeltenhet

2.2 Ressurser

Gruppen var avhengig av eksisterende ressurser for gjennomføring av prosjektet. Gruppen benyttet ressursene i en av to sammenhenger; prosjektplanlegging og systemutvikling. I prosjektplanlegging inngår møteaktivitet, ansvarsfordeling og kommunikasjon. Systemutvikling refererer til det praktiske arbeidet tilknyttet løsningen.

Prosjektplanlegging
Innkalling til møter ble gjort med Microsoft Teams eller e-post og hovedsakelig gjennomført fysisk. Ved sykdom deltok medlem via zoom. Gruppen kommuniserte og organiserte møter gjennom Facebook Messenger og discord. Kommunikasjon med veileder og bedrift tok sted ved fysiske møter eller over e-post. Gruppen benyttet Microsoft Word til rapportskriving og utforming av støttedokumenter.

Gruppen brukte Trello til organisering og tildeling av arbeidsoppgaver underveis i prosjektet. Trello er et nettbasert Kanbanverktøy som gir enkel oversikt over status, ansvar, tidsperspektiv og beskrivelse for individuelle oppgaver (Al-Baik & Miller, 2014).

Ved behov for hjelp eller veiledning vedrørende rapportskriving og prosjektplanlegging kunne gruppen henvende seg til intern veileder, Harald Soleim.

Systemutvikling
Gruppen hadde egne datamaskiner for utvikling av programvare. Eksperimentering og utvikling av det grafiske grensesnittet og datainnsamlingen ble utført her. Gruppen fikk tilgang til testmaskin hos oppdragsgiver. Maskinen ble benyttet i testing og utvikling av ny programvare. Et lukket testmiljø forenkler utviklingsprosessen betydelig fordi endringer kan testes innenfor kontrollerte rammer. Med testmaskinen fikk gruppen arbeidsplass på kontorene til CH.

Ved behov for hjelp eller veiledning vedrørende utvikling henvendte gruppen seg til prosjekteier og ekstern veileder, Ruben Patel. Veiledning tok sted under de faste fredagsmøtene hos bedriften.

2.3 Monitorering og tidsseriedata

En overvåkningsløsning for distribuerte systemer består av fem forskjellige elementer; lag, hendelser, terskler, innsamlingsintervall og datalagring (Kufel, 2016). Lagene beskriver hvor dataen sendes fra (opprinnelse), hvordan den sendes (nettverk) og hvordan den analyseres og presenteres (monitorering). Hendelser er tidstemplede data fra en kantenhet og inneholder identifikator og metrikker. Når hendelse er oversendt kan systemet varsle om uønskede resultater. Uønskede resultater kan være manglende data eller verdier utenfor definert terskelområde, eksempelvis om en harddisk snart er 99% full. Innsamlingsintervallet er hvor ofte kantenheter oversender hendelser.

Tidsserier representerer en kategori av tidsbestemte dataobjekter (hendelser), med en uerstattelig rolle i konteksten av overvåking av distribuerte systemer (Fu, 2010). En særegen egenskap ved tidsseriedata er dens kontinuerlige natur. Dataene må betraktes som en helhet heller enn som adskilte numeriske felt. Alle datapunktene har en relasjon til hverandre i form av tid. Minnelekkasjer er for eksempel et problem i distribuerte applikasjoner, noe som reduserer ytelse, selv om de kjører på

plattformer med automatisk minnehåndtering (Sor & Narayana Srirama, 2011). I tidsserier kan feiloppdagelser forenkles, fordi minnebruken kan visualiseres på tid og den gradvise økningen er enklere å oppfatte.

3 Design av prosjektet

Dette kapitelet inneholder en vurdering av tilgjengelige verktøy, drøfting av valg for verktøy, prosjektmetodikken og evalueringsplan.

3.1 Forslag til løsning

Siden dataene er tidsseriedata, var det naturlig å velge en tidsseriedatabase til lagring. Relasjonsdatabaser kan også lagre tidsseriedata, men ved høyere volum er både spørringer og transformasjoner adskillig (Musa, et al., 2019). De fleste tidsseriedatabaser kommer også med innebygget monitoreringsfunksjonalitet.

Til grensesnittet var det naturlig å undersøke eksisterende verktøy istedenfor å umiddelbart starte utviklingen fra bunnen av.

Resten av Kapittel 3.1 inneholder et sammendrag av de aktuelle verktøyene (3.1.1) med drøfting av styrker og svakheter (3.1.2)

3.1.1 Ulike verktøy

Databaser
DB-Engines (DBE) er et initiativ for å samle og presentere informasjon om databasesystemer (DB-Engines, 2023). Systemene får en rangering basert på fem metrikker:

- Generell interesse - frekvensen av Google-søk.
- Tekniske diskusjoner - antall relaterte spørsmål og interesserte brukere på Stack Overflow og DBA Stack Exchange.
- Jobbtilbud - antall tilbud der systemet er nevnt på Indeed og Simply Hired.
- Profesjonelle profiler - antall LinkedIn-profiler der systemet er nevnt.
- Sosiale nettverk - antall Twitter-tweets der systemet er nevnt

Figur 3.1 viser DBE-rangeringen for tidsseriedatabaser per mai 2023, rangeringen var hovedsakelig samme ved prosjektstart. Siden verken gruppen eller prosjekteier hadde tidligere erfaring med tidsseriedatabaser tok gruppen utgangspunkt i denne. Det er en naturlig antagelse at systemene med høy rangering også brukes mer fordi de er av høyere kvalitet. Dette i kombinasjon med at gruppen ikke hadde mulighet for å undersøke alle verktøy, ble de fire høyeste undersøkt nærmere. Siden KDB ikke er gratis og Graphite ikke tillater overføring av flere metrikker per enhet, sto valget mellom InfluxDB og Prometheus.

Rank			DBMS	Database Model	Score		
May 2023	Apr 2023	May 2022			May 2023	Apr 2023	May 2022
1.	1.	1.	InfluxDB	Time Series, Multi-model	29.90	+1.31	+0.35
2.	2.	2.	Kdb	Time Series, Multi-model	8.03	-0.44	-0.95
3.	3.	3.	Prometheus	Time Series	7.43	+0.44	+1.29
4.	4.	4.	Graphite	Time Series	6.26	-0.05	+0.80
5.	5.	5.	TimescaleDB	Time Series, Multi-model	4.73	+0.36	+0.03
6.	6.	↑ 7.	RRDtool	Time Series	3.61	+0.42	+1.11
7.	↑ 8.	↑ 9.	DolphinDB	Time Series, Multi-model	3.42	+0.81	+1.77
8.	↓ 7.	↓ 6.	Apache Druid	Multi-model	3.07	+0.31	+0.06
9.	9.	↑ 15.	TDengine	Time Series, Multi-model	2.95	+0.34	+2.04
10.	↑ 11.	↓ 8.	OpenTSDB	Time Series	2.50	+0.32	+0.67

Figur 3.1: DBE-populæritetsrangering av tidsseriedatabaser.

Visualiseringsverktøy

Både InfluxDB og Prometheus har innebygd visualisering, men dedikerte verktøy har mer funksjonalitet. InfluxDB støtter verktøyene Grafana og Chronograf og konfigurasjonen dekkes i dokumentasjonen. Prometheus har kun innebygd funksjonalitet med Grafana.

3.1.2 Diskusjon av alternativene

InfluxDB vs Prometheus

Det er lite som skiller InfluxDB og Prometheus. Begge er tidsseriedatabaser med open-source og cloud versjoner. Dermed kan løsningen utvikles gratis og flyttes til skyen om det skulle være behov. Begge støtter linux, inkluderer REST-API for tilgang, innebygd monitoreringsfunksjonalitet, klient-biblioteker til dagens mest brukte programmeringsspråk og liknende dataformat.

De nevneverdige forskjellene er:

- InfluxDB er push-basert, data må sendes fra kilden. Prometheus er pull-basert, data hentes fra kilden.
- InfluxData tilbyr også datainnsamlingsverktøyet Telegraf. Telegraf kan installeres på kantenhet, er enkelt å konfigurere og sender enhetsdata til InfluxDB.
- Begge har SQL-liknende spørrespråk. InfluxDB har i tillegg Flux, et funksjonelt spørrespråk som forsøker å utbedre manglende til SQL versjonen.
- InfluxDB inkluderer grafisk brukergrensesnitt for administrasjon av databasen.
- Dokumentasjonen til InfluxDB er oversiktlig og inkluderer eksempelkode. Prometheus har ikke like oversiktlig dokumentasjon, og mindre eksempelkode.

Grafana vs Chronograf
Grensesnittene i Grafana og Chronograf har lik struktur. Begge består av dashboard (websider) og mindre paneler (ruter) med forskjellige visualiseringsalternativer. Chronograf har kun støtte for InfluxDB, Grafana har til sammenligning støtte med flere titalls andre datakilder. Grafana har mer funksjonalitet og større brukerbase. (Bobriakov, 2022)

3.2 Valgt løsning

Basert på forskjellene i beskrevet i kapittel 3.1.2 valgte gruppen InfluxDB og Telegraf til backend og Grafana til frontend.

To spørrespråk gir høyere fleksibilitet. Telegraf kan ta av seg innsamling av data. Grafisk brukergrensesnitt og bedre dokumentasjon gjør læringskurven bratte. Brattere læringskurve er viktig for gruppen, men også CH om de skal kunne videreutvikle løsningen.

Grafana støtter adskillig flere datakilder, deriblant fire av de fem mest populære tidsseriedatabasene. Gruppen valgte dermed Grafana med hensyn på lav kostnad av å bytte databasesystem gjennom prosjektet eller om CH velger å bytte senere.

3.3 Prosjektmetodikk

Dette kapitellet består av utviklingsmetodikk, prosjektplan og risikoanalyse.

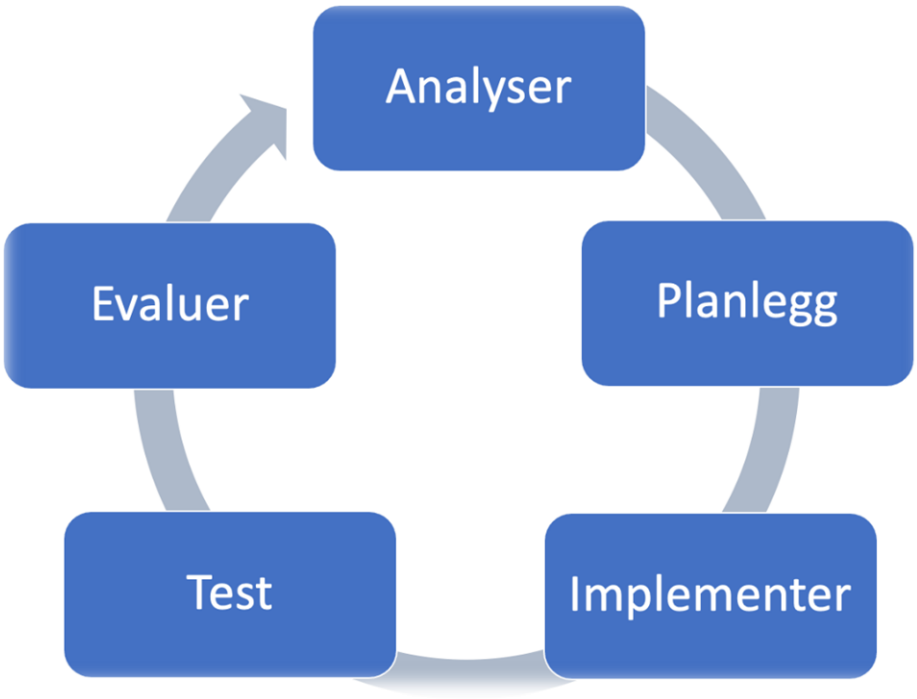
3.3.1 Utviklingsmetodikk

Gruppen har brukt elementer fra den agile utviklingsmetoden Scrum. Scrum er et prosessrammeverk designet for å utvikle og levere komplekse produkter gjennom iterative sykluser, kjent som "sprints". Sprints er tidsbestemte intervaller hvor bestemte oppgaver skal fullføres. Denne tilnærmingen gir gruppen mulighet til å konsentrere seg om enkelte funksjoner i produktet, samtidig som justeringer kan gjøres underveis. Beslutningstakingen i denne prosessen er empirisk, noe som betyr at den er basert på faktiske observasjoner og erfaringer (Schwaber & Sutherland, 2011). Scrum-metodikken er et godt valg for gruppen, fordi den åpner for regelmessige møter og tilbakemeldinger fra prosjekteier. Denne strukturen bidrar til kontinuerlig kommunikasjon og effektiv prosjekthåndtering.

Figur 3.2 viser gruppens iterative arbeidsflyt gjennom prosjektet. Flyten består av fem deler:

- Del 1 (Analyser): Avdekk behovene til prosjekteier i de ukentlige møtene.
- Del 2 (Planlegg): Hvilke endringer kan dekke behovene fra del 1 og hvordan kan de implementeres i InfluxDB/Grafana.
- Del 3 (Implementer): Utfør endringer i løsningen i henhold til vurderingene i del 2.
- Del 4 (Test): Sjekk at endringene fungerer som forventet. Gjennomføres ved å sende testdata til databasen.

- Del 5 (Evaluer): Prosjekteier brukertester løsningen. Erfaringer, tilbakemeldinger og tekniske feil overføres til del 1 for å avdekke nye behov.

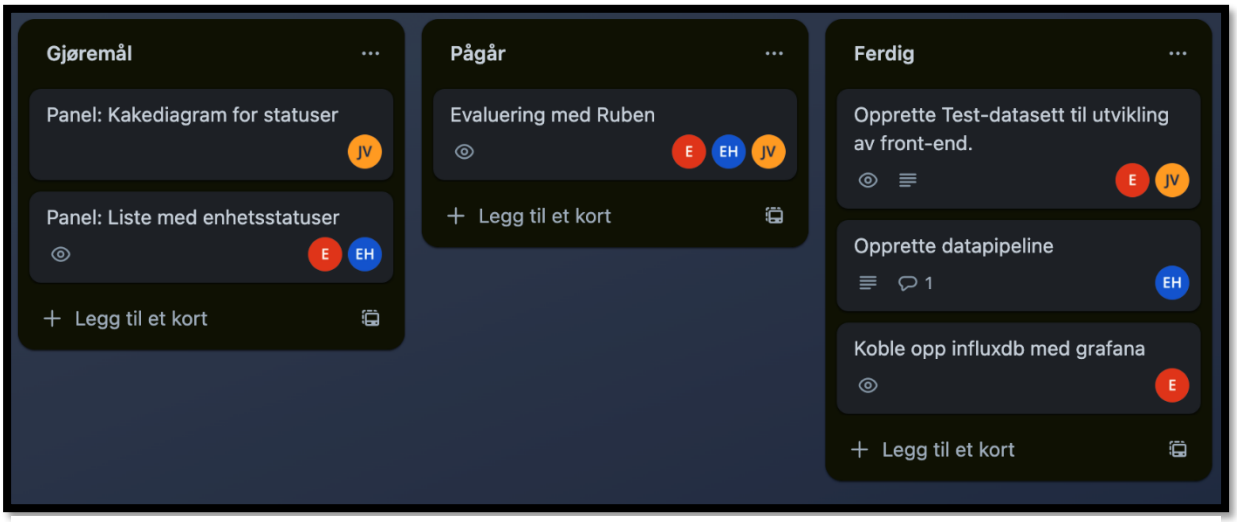


Figur 3.2: Illustrasjon av arbeidsflyt

Gruppen har brukt Kanban i utviklingsprosessen. Kanban kan øke effektiviteten i utviklingsprosjekt ved å visualisere arbeidsflyten for større oppgaver i mindre, håndterbare deler (Powell J, 2018).

Identifisering av oppgaver: Større oppgaver deles inn i mindre, håndterbare oppgaver. Oppgavene er individuelle elementer som må fullføres for å dekke brukerbehov.

Figur 3.3 viser visualiseringen av arbeidsflyten. Oppgavene visualiseres på en Kanban-tavle. Tavlen er delt inn i forskjellige kolonner, som representerer forskjellige stadier av arbeidsflyten (Gjøremål, Pågår, Ferdig).



Figur 3.3: Kanban tavle

Styring av flyt: Når en oppgave flyttes fra en kolonne til en annen, blir det en ledig plass i den forrige kolonnen. Dette signaliserte til gruppen at det er rom for en ny oppgave i denne fasen av arbeidsflyten. Dette sikrer en jevn flyt av arbeid gjennom systemet.

Ved identifisering av oppgaver, visualisering og styring av arbeidsflyt, har Kanban hjulpet gruppen med å balansere arbeidsbelastningen og forbedre effektiviteten.

3.3.2 Prosjektplan

Gruppen benyttet GANTT-diagram (se prosjekthåndboken figur 1.6) for prosjektplanlegging. GANTT-diagram er et prosjektstyringsverktøy som illustrerer framgangen i et prosjekt og aktivitetene over tid (MEARDON, 2023).

Diagrammet består av to hoveddeler; dokumentasjon og utvikling. Dokumentasjon dekker det akademiske arbeidet i prosjektet, inklusiv støttedokumentasjon og hovedrapport. Utvikling dekker de tre sprintene: systemoppsett, grensesnitt i Grafana og evaluering. Hver sprint er ytterligere delt i mindre mer konkrete deler.

3.3.3 Risikoanalyse

Risikoanalyse er grunnlag for er å identifisere, evaluere og prioritere potensielle risikoer, som kan være tekniske, organisatoriske eller menneskelige (Aven, 2023).

En effektiv risikoanalyse omfatter trinn for identifisering, vurdering, og håndtering av risiko. Identifisering inkluderer forståelse av mulige risikoer og deres kilder. Vurdering innebærer å bestemme sannsynligheten for at en risiko vil oppstå, og dens mulige innvirkning på prosjektet. Håndtering av risiko kan innebære forebygging, reduksjon, overføring eller aksept.

Å ha en solid risikoanalyseprosess på plass fører til robust systemutvikling og reduserer potensielle negative feilkilder til prosjektet.

Risikovurdering for prosjektet ligger vedlagt i prosjekthåndboken figur 2.1.

3.4 Evalueringsplan

Gruppen benyttet seg av to evalueringsmetoder; kontinuerlig brukertesting/enhetstesting og avsluttende intervju. Metodene er kvalitative. Gruppen vurderte kvantitative metoder som SUS (System Usability Scale). Majoriteten av kvantitative metoder avhenger av en viss brukermasse (Ibekwe, 2022) og viser ikke nødvendigvis hvilke problemer brukeren støtte på (Budiu, 2017).

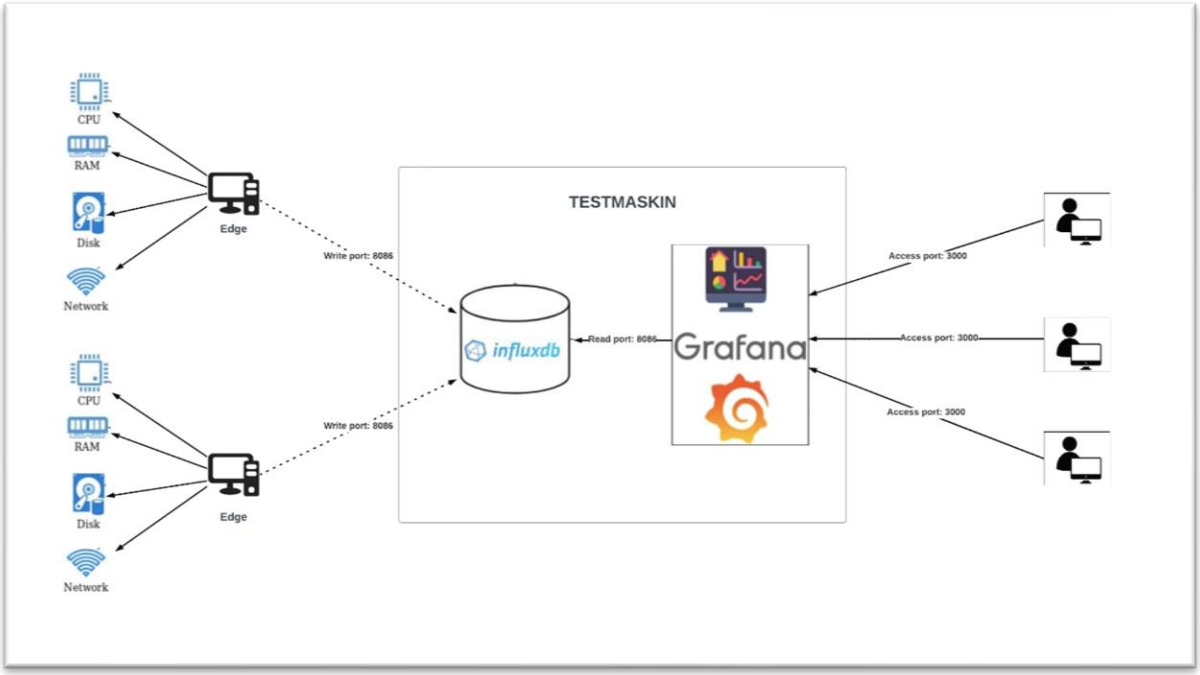
Som beskrevet i kapittel 3.3.1 mottar gruppen kvalitative tilbakemeldinger fra prosjekteier under statusmøtene i utviklingsfasen. Når endringer i løsningen planlegges og utføres basert på kvalitativ analyse, virker det mot sin hensikt å gjennomføre en kvantitativ sluttevaluering.

4 Detaljert løsning

Dette kapittelet inneholder en beskrivelse av den endelige løsningen. Hvordan enhetsdata fra kant sendes til testmaskinen der det lagres, monitoreres og visualiseres. Beskrivelsen deles i følgende: overordnet systemarkitektur, innhenting av data og administrasjon av InfluxDB (databasen) og Grafana (visualiseringsverktøyet). Overordnet systemarkitektur følger dataene i systemet fra opprinnelse til visualisering. I resten av delkapitlene beskrives komponentene i detalj.

4.1 Overordnet Systemarkitektur

Systemet består av kantenheter som sender data til testmaskinen med InfluxDB og Grafana instansene og maskiner som kobler seg til denne. Kantenhetene og testmaskinen er tilkoblet VPN til CH, dermed er dataene kryptert og IP-adressen til sentralenheten er statisk. Figur 4.1 viser hvordan kantenheter i CH samler og sender data til InfluxDB på port 8086 og deretter lagres. InfluxDB varsler på manglende verdier eller verdier utenfor brukerdefinerte områder. Grafana leser data fra port 8086 og visualiserer enhetsdataene med tilhørende varsler. Brukerne i CH kan koble seg til Grafana på port 3000.



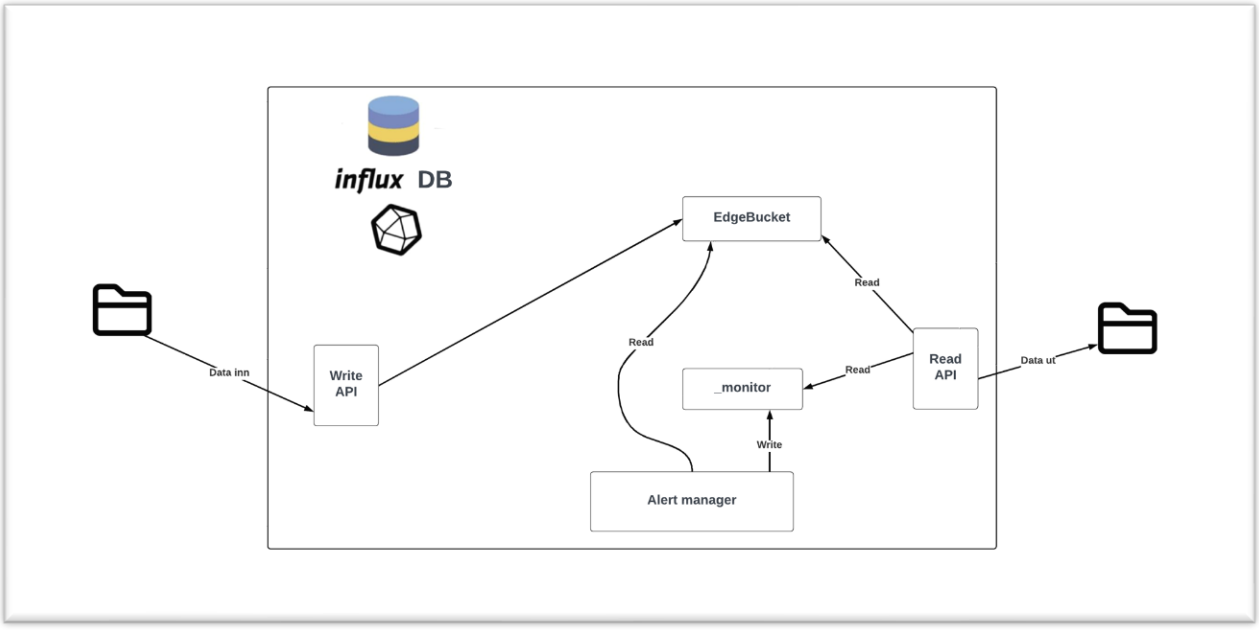
Figur 4.1: Overordnet systemarkitektur

4.2 InfluxDB

InfluxDB instansen mottar og lagrer data fra kantenhetene. I dette delkapittelet dekkes komponenter og funksjoner relevant for prosjektets løsning.

- **Line Protocol**: Line Protocol er InfluxDBs eget dataformat. Alle datapunkt som mottas, lagres og sendes ut er på dette formatet.
- **Buckets:** navngitt område hvor dataene lagres.
- **Flux**: Et funksjonelt skriptspråk designet for å spørre, behandle, skrive og analysere data fra diverse datakilder. Flux er tett integrert mot InfluxDB og er også utviklet av InfluxData.
- **GUI:** et webbasert brukergrensesnitt som gir enkel og intuitiv administrasjon og interaksjon med InfluxDB-installasjonen.
- **Data Explorer:** del av GUI for å utforske og visualisere data med flux-spørringer på en brukervennlig måte.
- **Alert manager:** innebygd system for overvåking av data og generering av alarmer basert på manglende data og brukerdefinerte og terskler.
- **API:** RESTFUL HTTP-grensesnitt for skriving til og lesing fra InfluxDB.

Figur 4.2 viser fra venstre til høyre hvordan data skrives til InfluxDB og leses ut til Grafana. Data mottas på write-endepunktet og lagres i bucket «EdgeBucket». Alert manager leser fra Enhetsdata og varsler basert på brukerdefinerte sjekker. Resultatet fra sjekkene skrives til bucket «_monitor». Varslene og data leses av Grafana via read-endepunktet.



Figur 4.2: Databehandling i InfluxDB

4.2.1 Dataformat (Line Protocol)

Figur 4.3 viser Line Protocols fire deler; measurement, tag set, field set og timestamp. Med hensyn på measurement, tag set og timestamp er hvert datapunkt unikt. Det er dermed ingen datapunkter som har samme verdi på measurement, tag set og timestamp.

```
measurementName,tagKey=tagValue fieldKey="fieldValue" 1465839830100400200
-------------- -------------- --------------------- -------------------
      |               |                 |                    |
  Measurement       Tag set          Field set            Timestamp
```

Figur 4.3: Line Protocol format

Figuren 4.4 viser et enkelt datapunkt på Line Protocol.

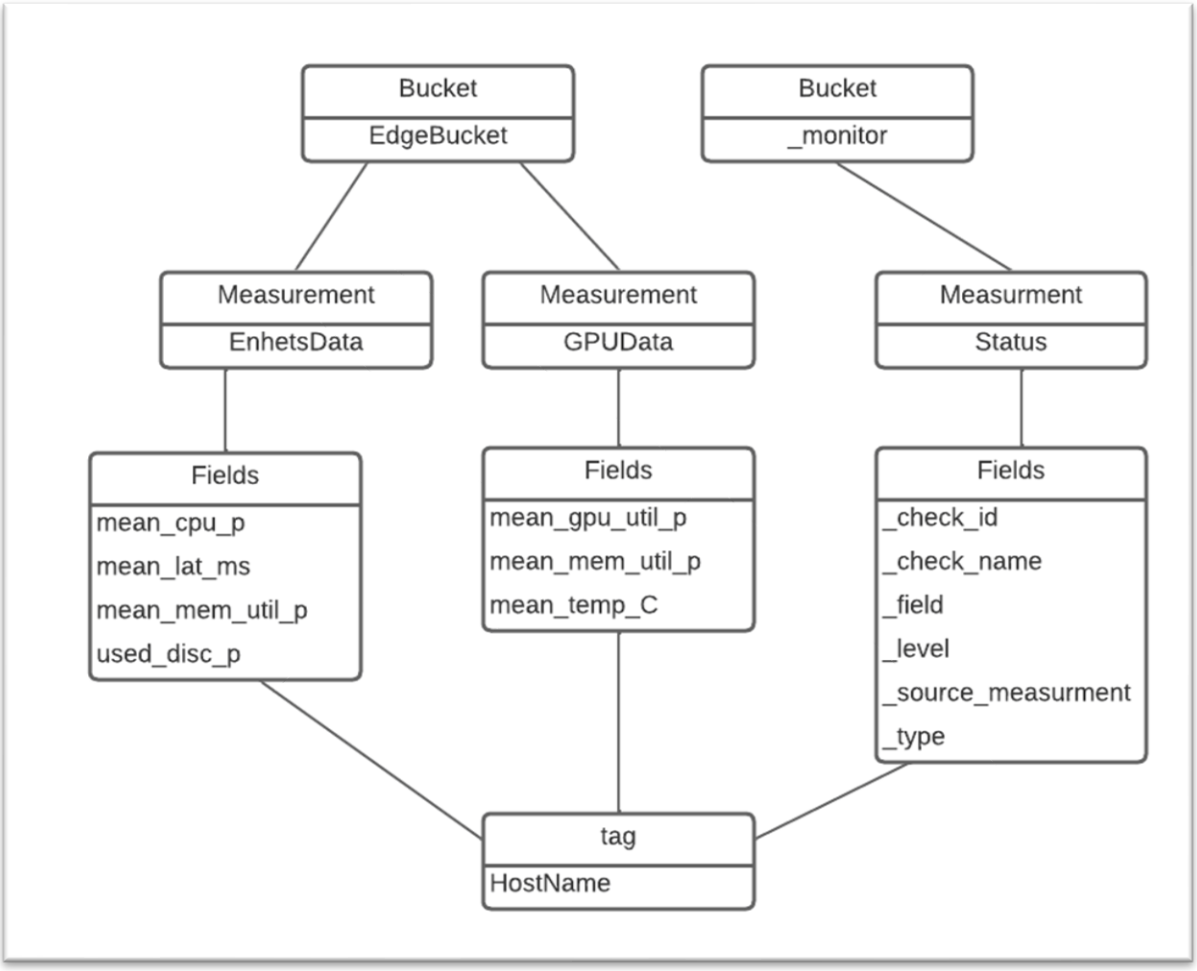
- Measurement: Hvert datapunkt inneholder nøyaktig et measurement. Measurement identifiserer dataen og hvor den lagres. «Enhetsdata» i figur 4.4.
- Tag Set: Tag set er en gruppe nøkkel-verdi par med tags. Det er ikke påkrevd i Line Protocol, men brukes som identifikator og til å indeksere data, all data med samme tag indekseres sammen. Dette effektiviserer spørringer og gir logisk atskillelse i datastrukturen. «HostName=Whips» i figur 4.4.
- Field Set: Field set inneholder alle nøkkel-verdi par for fields. Fields er de reelle måledataene. I figur 4.4 er «RAMUtil» og «CPUUtil» nøklene med de respektive verdiene i prosent 67.8 og 7.2.
- Timestamp: Tidsstempel i UNIX-format. Kan sendes med dataen eller legges til av InfluxDB instansen når dataen ankommer.

```
Enhetsdata,HostName=Whips RAMUtil=67.8,CPUUtil=7.2
-------------- -------------- --------------------
      |               |                 |
  Measurement       Tag set          Field set
```

Figur 4.4: Eksempel på Line Protocol

Datastruktur endelig løsning:

Figur 4.5 illustrerer den logiske datamodellen i løsningen. Hierarkiet følger figuren nedover. Tag tilhører fields som tilhører measurement som tilhører bucket. EdgeBucket har to measurements, «Enhetsdata» og «GPUData». Datapunktene inneholder henholdsvis fire og tre metrikker (Fields), «HostName» som eneste tag. Status fra sjekkene ligger under measurement «Status» i «_monitor» bucket. Metrikkene her er standard for alle sjekker i InfluxDB, tag er også her «HostName».



Figur 4.5: Logisk datamodellen av løsningen

4.2.2 Buckets

Som vist i den logiske datamodellen er buckets nivået over measurements i databasen. Alle buckets kan konfigureres med hensyn på navn og bevaring av data. Ut fra behov kan en bucket bevare data for ønskede tidsperioder. Endelig løsning består av en bucket (EdgeBucket) som bevarer data i 30 dager fordi prosjekteier ikke hadde behov for eldre data.

InfluxDB har egen «systembucket» for lagring av data tilknyttet instansens egne prosesser, _monitor. _monitor inneholder data tilknyttet InfluxDBs interne varslingssystem, og forklares nærmere i 4.2.6.

4.2.3 Flux

I løsningen brukes Flux-spørringer for å hente ut og transformere data til visualisering. Spørringene følger denne flyten:

1. Hent en spesifisert mengde data fra en kilde.
2. Filtrer data basert på tid eller kolonneverdier.
3. Behandle og forme data til ønsket resultat.
4. Returner resultatet.

Figuren 4.6 viser et eksempelskript fra testfasen.
Spørringen forklart med utgangspunkt i flyten beskrevet ovenfor:

1. (Linje 1 – Linje 2) Henter all data fra bucket «GrafanaViz» mellom kl. 12:49:13 og kl. 12:49:17.
2. (Linje 3 – Linje 5) Filtrer på measurement lik «Enhetsdata» så HostName lik «Whips» så _field lik «RAMUtil» verdier.
3. (Linje 6 – Linje 7) Verdiene for «RAMUtil» er i prosent. Her gjøres alle verdiene om til promille. InfluxDB inkluderer kolonner med fra og til tidspunkt for spørringen, disse er overflødige og fjernes i linje 7.
4. (Linje 8) Returnerer resultat.

```
1  from(bucket: "GrafanaViz")
2    |> range(start: 2023-05-04T12:49:13Z, stop: 2023-05-04T12:49:17Z)
3    |> filter(fn: (r) => r["_measurement"] == "Enhetsdata")
4    |> filter(fn: (r) => r["HostName"] == "Whips")
5    |> filter(fn: (r) => r["_field"] == "RAMUtil")
6    |> map(fn: (r) => ({r with _value: r._value * 10.0}))
7    |> drop(columns: ["_start", "_stop"])
8    |> yield()
```

Figur 4.6: Flux-skript (InfluxDB)

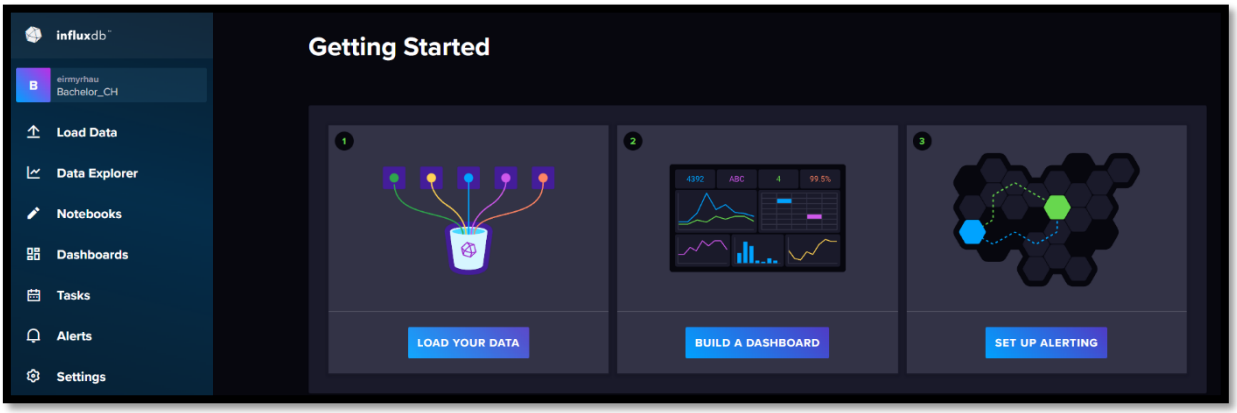
Tabell 4.1 viser resultatet fra spørringen.

_measurement group string	_field group string	_value no group double	_time no group dateTime:RFC3339	HostName group string
Enhetsdata	RAMUtil	757	2023-05-04T12:49:13.000Z	Whips
Enhetsdata	RAMUtil	758	2023-05-04T12:49:14.000Z	Whips
Enhetsdata	RAMUtil	766	2023-05-04T12:49:15.000Z	Whips

Tabell 4.1: Resultat av Flux-spørring

4.2.4 GUI

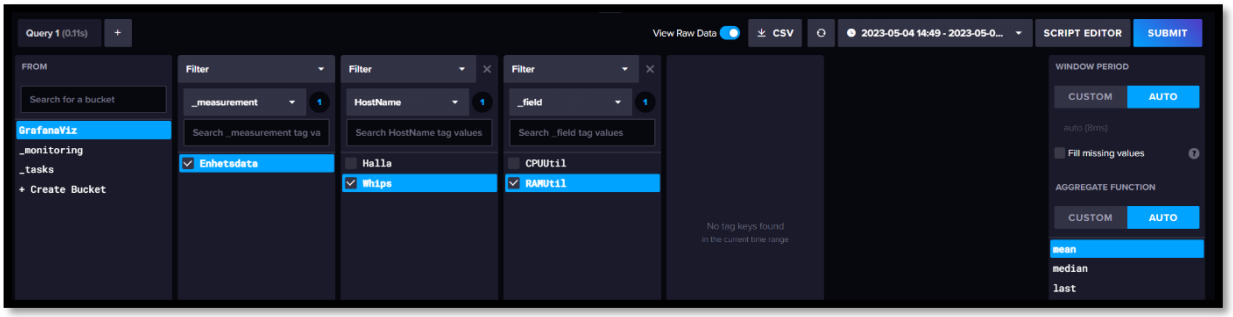
InfluxDB kommer med eget GUI (Grafisk Brukergrensesnitt) tilgjengelig på port 8086.
Figur 4.7 er skjermbilde fra hjemmesiden i grensesnittet. Knappene i panelet på venstre side leder videre til forskjellige sider. Grensesnittet brukes i løsningen til å administrere buckets og API-nøkler (Load Data), lage skript med «Script Builder» (Data Explorer) og opprette varslinger (Alerts).



Figur 4.7: Hjemmeside InfluxDB

4.2.5 Data Explorer

«Data Explorer» består av tre komponenter; «Script Builder», «Script Editor» og visualiseringsområde. «Script Builder» er en visuell måte å bygge flux-spørringer på. Figuren 4.8 viser script builder i bruk. Filtrering gjøres ved å trykke på metrikkene brukeren ønsker og builderen produserer automatisk skriptet. Skriptet kan så hentes ut ved å trykke «Script Editor» øverst til høyre. Builderen produserer samme skript som eksempelskriptet i 4.2.3 (figur 4.6) med unntak av omgjøringen fra prosent til promille og fjerning av tidskolonnene.



Figur 4.8: Data Explorer: Script builder

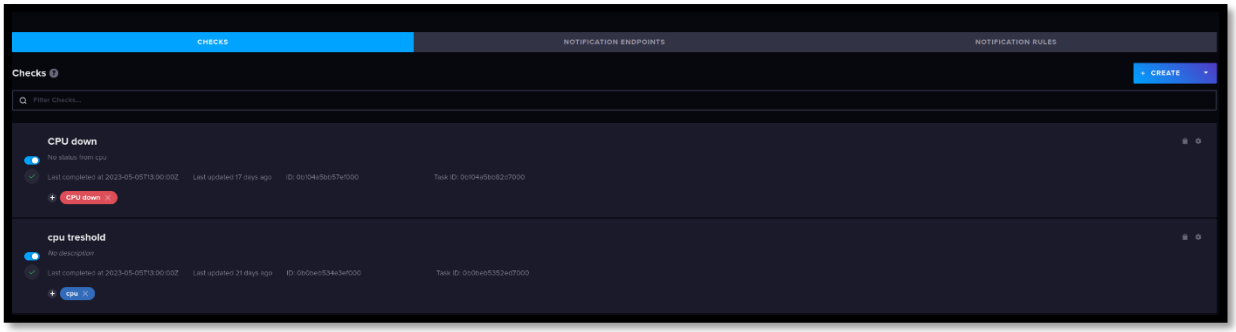
Resultatet av skriptet visualiseres så i et felt rett over (figur 4.9).

Figur 4.9: Data Explorer: Visualisering

4.2.6 Alert manager

Under «Alerts» ligger InfluxDB sitt innebygde varslings- og monitoreringssenter. Herunder kan sjekker opprettes, endres eller slettes. InfluxDB tilbyr to forskjellige sjekker; «deadman» og «threshold». De to typene er noe annerledes, men prinsippet er det samme; varsle når verdier oppfyller brukerdefinerte kriterier. Alle sjekker skriver resultatet av sjekken til «_monitor» bucket. Sjekkene kan returnere fire forskjellige resultater: «crit», «warn», «info» og «ok».

Deadman:
En deadman-sjekk varsler når en tidsserie ikke får nye verdier innenfor et brukerdefinert tidsrom. Hver tidsserie er unik med hensyn på measurement og tag set. I figur 4.10 er «CPU down» av typen deadman.

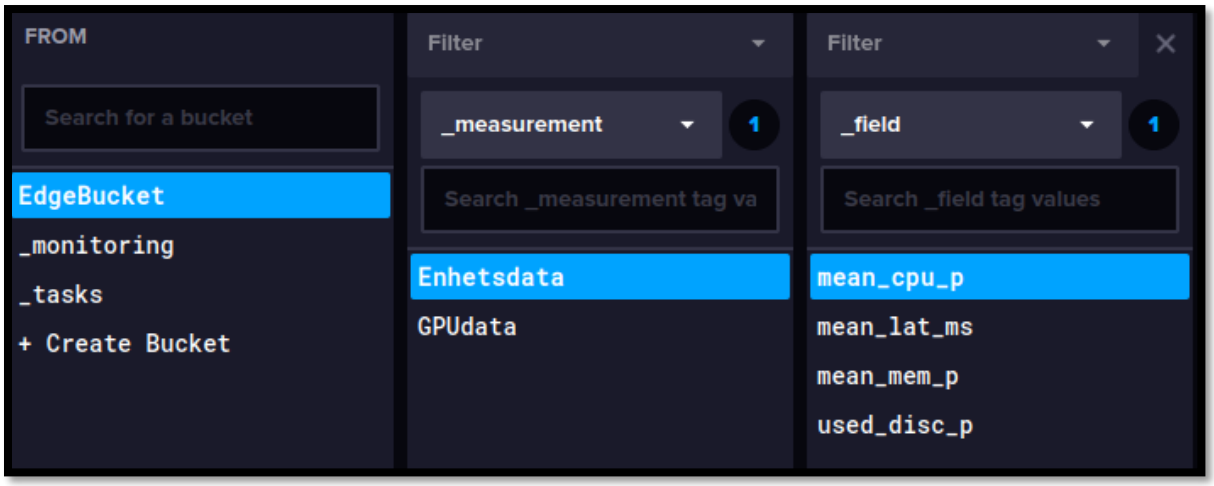


Figur 4.10: Alert manager

Figurene (figur 4.11-4.13) viser konfigurasjonen for sjekken.

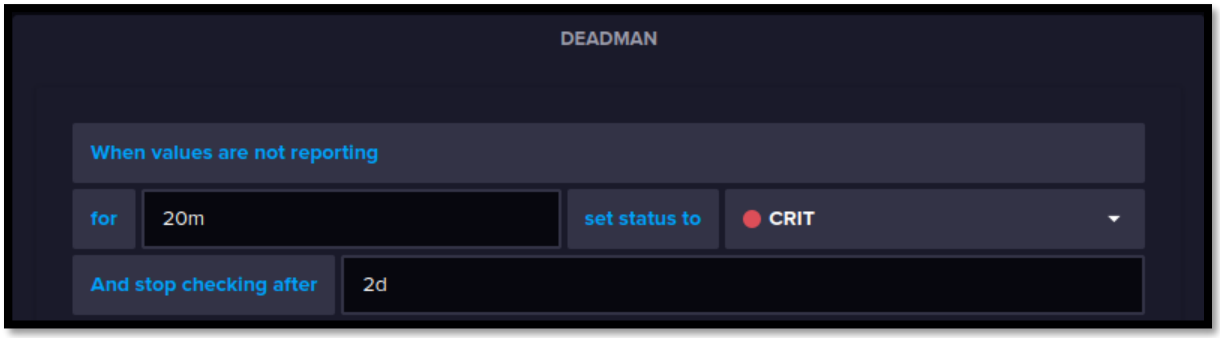
- Figur 4.11: Definerer hvilken tidsserie som monitoreres, measurement «Enhetsdata» og field «mean_cpu_p» i dette tilfellet.
- Figur 4.12: Definerer hvor ofte sjekken gjennomføres. Her hvert 15. minutt.
- Figur 4.13: Definerer hvor lenge sjekken «venter» før den returnerer «crit», her 20 minutter. «And stop checking after» er hvor lang tid før sjekken slutter å monitorere en tidsserie med status «crit», her to dager.

Siden «HostName» er tag i datastrukturen, vil sjekken monitorere hver enkelt kantenhet som sender data.



Figur 4.11: Konfigurasjon av Deadman-sjekk: Script Builder

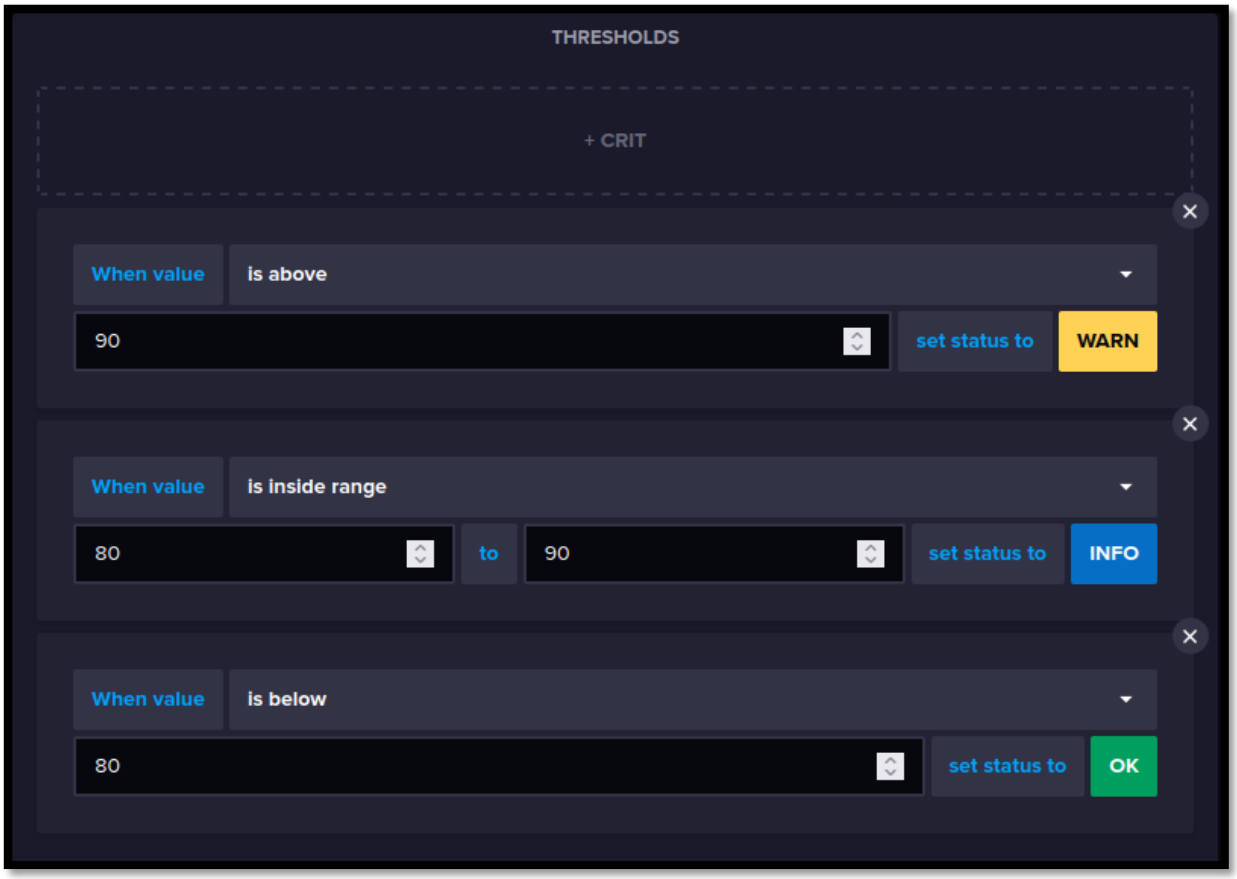
Figur 4.12: Konfigurasjon av Deadman-sjekk: Scheduler



Figur 4.13: Konfigurasjon av Deadman-sjekk: Time to alert

Threshold:
En threshold-sjekk varsler når en tidsserie får nye verdier som er innenfor/utenfor et brukerdefinert intervall. «cpu threshold» i figur x er av typen threshold. Sjekkene fungerer og konfigureres nesten på samme måte som deadman.

Figur 4.14 er ekvivalent med konfigurasjonen for deadman vist i figur 4.13. Sjekken returnerer «ok» for alle verdier under 80%, «info» for verdier mellom 80% og 90% og «warn» for verdier over 90%. Threshold kan konfigureres til å returnere «crit» men er forbeholdt deadman i endelige løsningen for å unngå forvirring.



Figur 4.14: Konfigurasjon av Threshold-sjekk: Definere terskelverdier

4.2.7 API

InfluxDB API brukes i løsningen til skriving og lesing av data. Databasen administrere tilgang til endepunktene med API-nøkler

Write:
Skriving av data til InfluxDB API-et gjøres ved en HTTP-forespørsel på /api/v2/write-endepunktet. Forespørselen bruker POST-metoden og må oppfylle kravene i figuren ovenfor (figur x). Bruk av write-endepunktet i løsningen beskrives i 4.3.

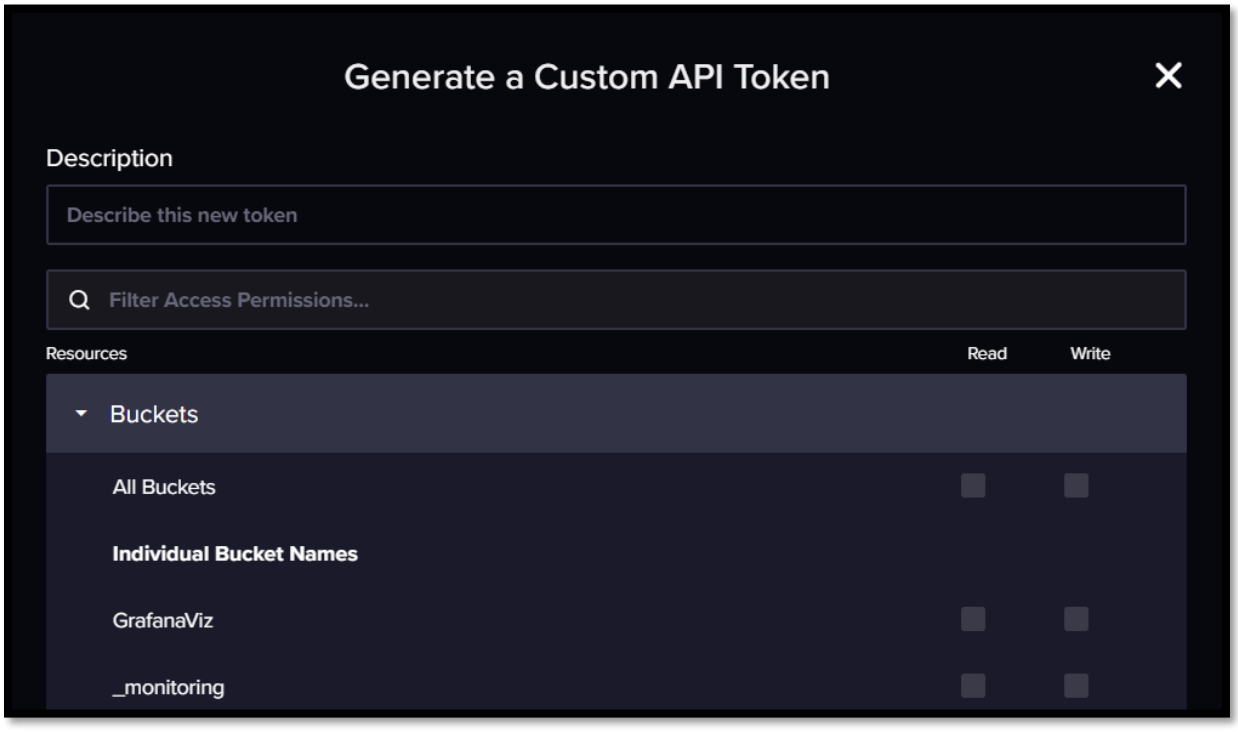
Query:
Lesing av data fra InfluxDB API-et gjøres ved en HTTP-forespørsel på /api/v2/query-endepunktet. I løsningen gjøres dette i grafana og beskrives i kapittel 4.4.1.

Nøkler:
InfluxDB bruker API-nøkler for å administrere tilgang på API-endepunktene. Nøkler kan opprettes, endres, slettes, aktiveres og deaktiveres i GUI.

Requirement	Include by
Organization	Use the `org` query parameter in your request URL.
Bucket	Use the `bucket` query parameter in your request URL.
Timestamp precision	Use the `precision` query parameter in your request URL. Default is `ns`.
API token	Use the `Authorization: Token YOUR_API_TOKEN` header.
Line protocol	Pass as plain text in your request body.

Figur 4.15: Krav til http-forespørsmål på write-endepunktet

Figur 4.16 viser grensesnittet for å opprette en ny API-nøkkel. Nøkkelen kan konfigureres til å ha lese/skrive-rettigheter til alle eller valgte buckets.



Figur 4.16: Grensesnitt opprettelse API-nøkkel

4.3 Oversending av data

Innhenting av data kan deles i tre; autentisering og måling/sending av data. Enhetsdata samles inn og aggregeres lokalt på hver enhet og enheten må autentisere seg mot databasen ved dataoverføring. Koden for dette er implementert i programvaremoduler hos CH. CH har skrevet modulene selv, men de er basert på eksempelskriptet i figur 4.17 som prosjektgruppen lagde. Oversending av data forklares med utgangspunkt i eksempelskriptet. Den eneste praktiske forskjellen er at data måles momentant i eksempelet, men aggregeres over måleperioden i den faktiske løsningen. De tre delene forklares i den rekkefølge de oppstår i skriptet.

Figur 4.17: Eksempelskript i python for måling og overføring av data

4.3.1 Autentisering

InfluxDB verifiserer forespørsler på «token» parameteret. «Token» inneholder en API-nøkkel. Ved mangel på nøkkel eller manglende rettigheter avvises forespørselen.

4.3.2 Måling

«Psutil» er et open-source python bibliotek for innhenting av informasjon om prosesser og systemutnyttelse (Rodola, 2023), og brukes i scriptet for innhenting av minne- og CPU-utnyttelse. Samme biblioteket blir brukt i den endelige løsningen sammen med andre bibliotek for måling på flere metrikker. Dataene samles i en streng «msg» som er på «Line Protocol» formatet.

4.3.3 Overføring

Dataoverføringen gjennomføres med urllib3. Urllib3 er en brukervennlig HTTP klient for python. CH brukte allerede dette for dataoverføring. Scriptet sender en HTTP POST forespørsel til write-endepunktet til influxDB APIet. Forespørselen autentiseres med «token» og «msg» skrives inn i influxDB.

4.4 Grafana

Grafana leser og visualiserer data fra InfluxDB i et webgrensesnitt. Tilkoblingen til InfluxDB settes opp i Grafana og trenger en API-nøkkel med leserettigheter. Grensesnittet består av forskjellige websider (dashboards). Hvert dashboard inneholder paneler (panels) som består av flux-spørring og visualisering. Noen av panelene inneholder navn på enhetene til CH. Navnene inneholder informasjon om enheten og hvor den står, av sikkerhetsmessige hensyn er dette skjult i figurene. I visualiseringene brukes fargekoder og tekst for å vise statuser. Fargekodene følger samme mønster som i InfluxDB; grønn=ok, blå=info, gul=warn og rød=crit. Grafana lagrer dashboard som JSON-filer, alle fire er vedlagt i systemdokumentasjonen.

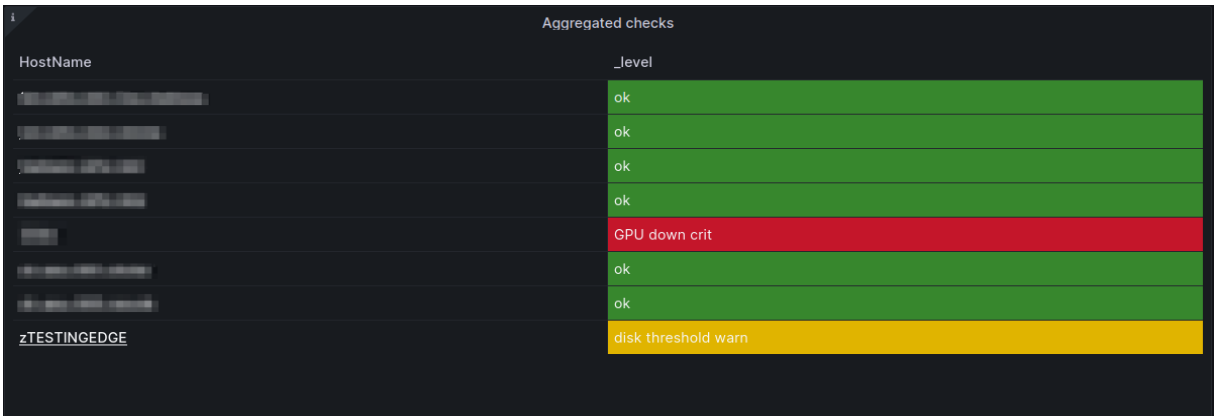
4.4.1 Panel

Paneler er den minste komponenten av grensesnittet i Grafana. Et panel inneholder en flux-spørring og visualisering. Figur 4.18 viser konfigurasjonen for paneler. Visualiseringen (rødt) avhenger av innstillingene (gult) og flux-spørringen (grønt). Grafana kommer med 23 forskjellige visualiseringstyper, seks av de brukes i løsningen. Under vises en av hver type visualisering med forklaring. I paneler som inneholder statuser vises alltid verste for hver kant, hvis en kant har «warn» og «info» vises derfor «warn». Øverst til venstre og høyre er henholdsvis «Host Name» variabelen og valgt tidsrom, hvilken innvirkning disse har på panelene forklares i 4.4.2.

- Tidsseriegraf (figur 4.18 -rødt).
  Figuren viser:
  - «RAMUtil» verdier for en enhet i tidsrommet 16:09-16:45.

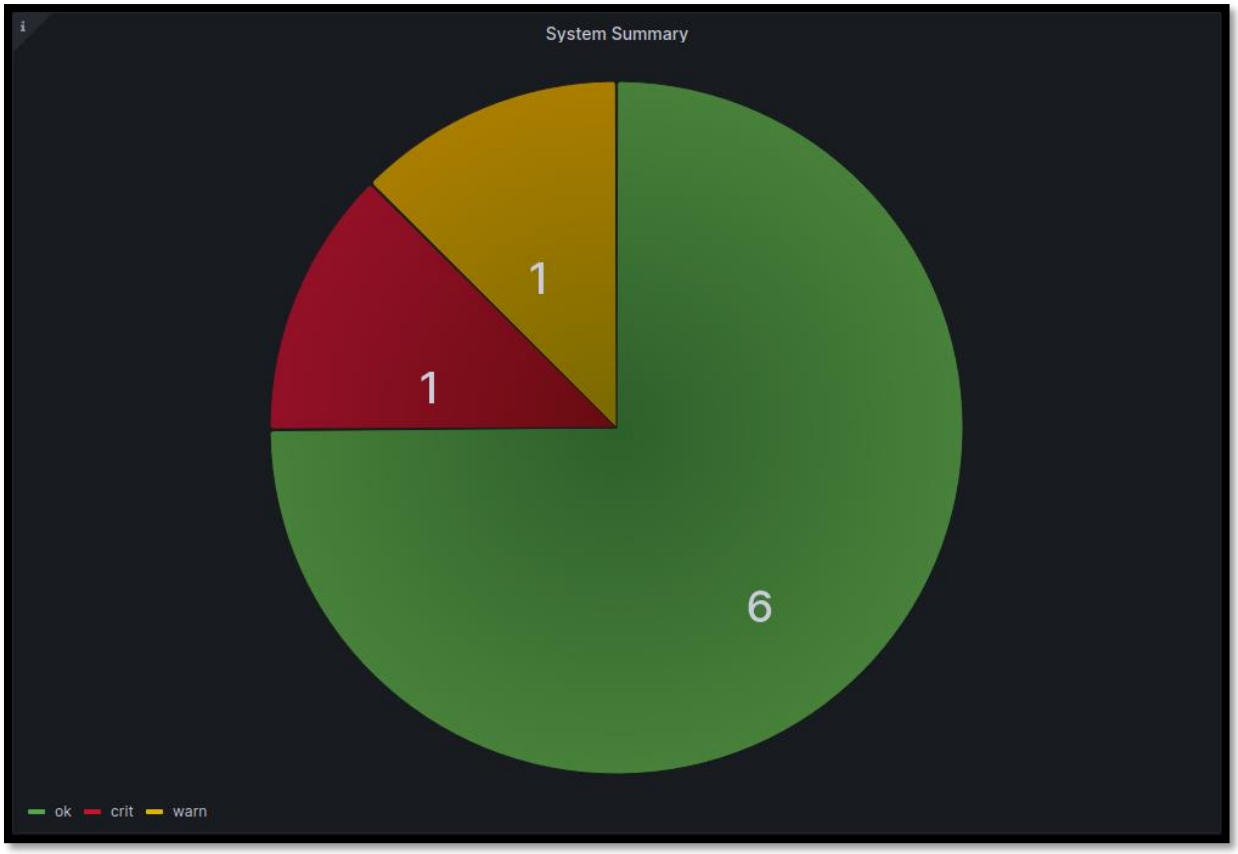
Figur 4.18: Panelkonfigurasjon tidsseriegraf

- Tabellvisualisering (figur 4.19) med oversikt med alle kantenheter og tilhørende status. Figuren viser:
    - 1 enhet er nede
    - 1 enhet har status «warn» på disk threshold sjekk.
    - 6 enheter med «ok» på alle sjekker.



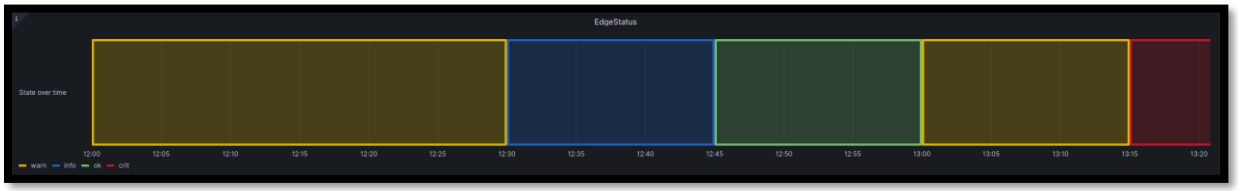
Figur 4.19: Panel, tabell

- Kakediagram (figur 4.20). Systemsammendrag av statuser. Hver enhet er representert en gang. Diagrammet viser:
    - 1 enhet er nede
    - 1 enhet med «warn» på minst en sjekk.
    - 6 enheter med «ok» på alle sjekker.



Figur 4.20: Panel, Kakediagram

- State timeline (figur 4.21). Viser tidslinje for status til en enhet.
  Figuren viser:
  - (12:00-12:30) Minst en sjekk med status «warn».
  - (12:30-12:45) Minst en sjekk med status «info».
  - (12:45-13:00) Alle sjekker med statuser «ok».
  - (13:00-12:30) Minst en sjekk med status «warn».
  - (13:15-) Enhet nede.



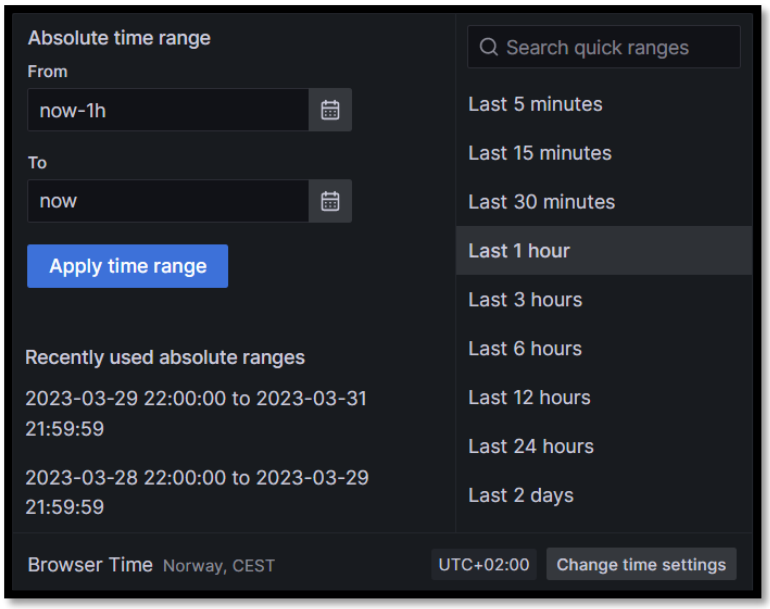
Figur 4.21: Panel, State Timeline

4.4.2 Tidsrom og variabler

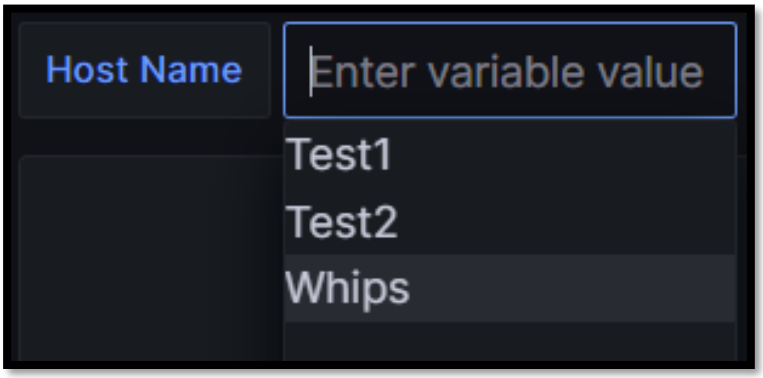
I Grafana kan brukeren selv endre tidsrom og «Host Name» variabel. Flux-spørringen i figur 4.22 er samme som i figur 4.18 i 4.4.1. I linje 2 defineres tidsrommet for spørringen. «v.timeRangeStart» og «v.timeRangeStop» verdiene oppdateres automatisk basert på tidsrommet brukeren velger i «Time Range» - menyen i figur 4.23. Linje 3 av spørringen filtrerer på ønsket kantenhet. Grafana bytter ${HostName} med valgt verdi for «Host Name» variabelen som vist i figur figur 4.24. Alternativene er alle unike «Host Name» verdier for de siste 30 dagene. Listen er derfor dynamisk og inneholder alle enheter i systemet. Når brukeren endrer variabelen i menyen endres dermed spørringer på alle panelene og visualiseringene oppdateres.

```
1  from(bucket: "GrafanaViz")
2    |> range(start: v.timeRangeStart, stop: v.timeRangeStop)
3    |> filter(fn: (r) => r["HostName"] == "${HostName}")
4    |> filter(fn: (r) => r["_field"] == "RAMUtil")
5    |> aggregateWindow(every: 5s, fn: mean, createEmpty: true)
```

Figur 4.22: Flux spørring (Grafana)



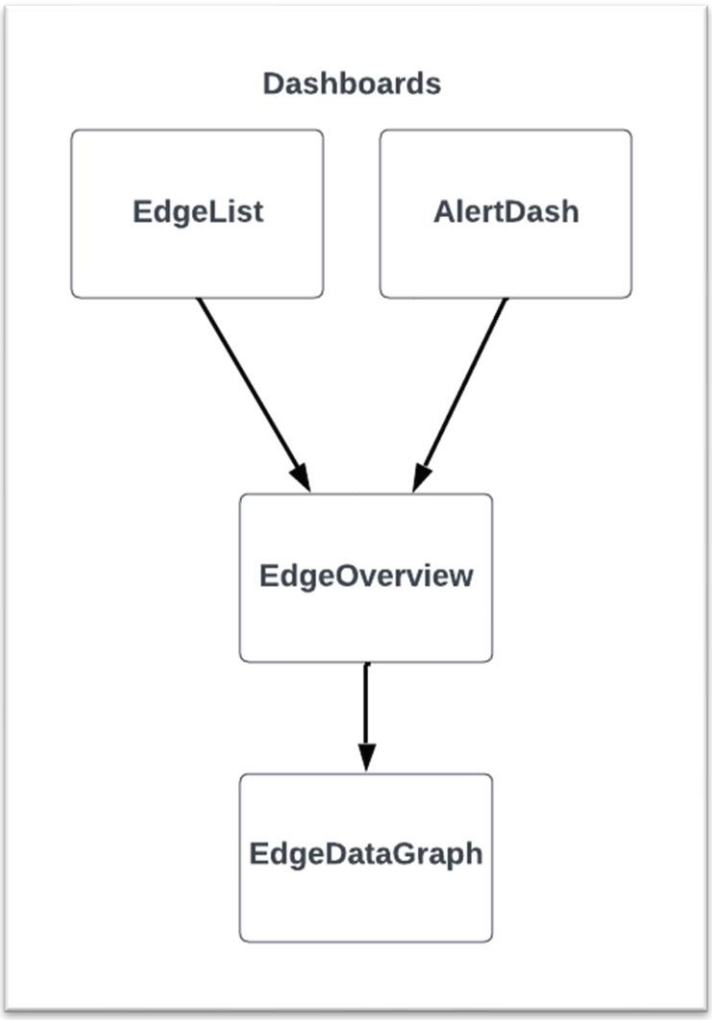
Figur 4.24: Time Range meny



Figur 4.23: Valg av «Host Name» variabel

4.4.3 Dashboard

Løsningen inneholder fire dashboard med inntil fem paneler. Dashboardene varierer i detaljnivå og beskrives i dette delkapittelet. Figuren 4.25 viser hierarkiet for dashboard. «EdgeList» og «AlertDash» er hovedsidene og detaljnivået på dataen øker for hvert steg «nedover» i figuren. «EdgeList» og «AlertDash» inneholder statusinformasjon (feil og varsler) for hele systemet. «Edge overview» og «EdgeDataGraphs» inneholder henholdsvis statusinformasjon og tidsseriegrafer for en enkelt kant. Pilene symboliserer klikkbare lenker. Ved å trykke på navnet til en kantenhet i «EdgeList» eller «AlertDash» navigeres brukeren til «EdgeOverview» med valgte kantenhet som variabel. Trykker de igjen på enhetsnavnet føres de til «EdgeDataGraph».



Figur 4.25: Dashboardhierarki

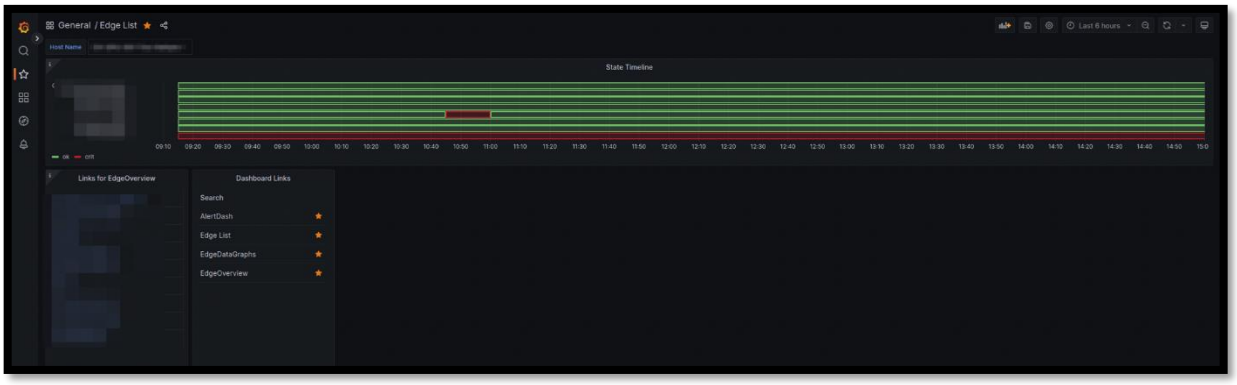
Alert Dash

Figur 4.26 viser AlertDash. De to panelene «System Summary» (venstre) og «Aggregated checks» (høyre) viser statuser fra sjekkene i InfluxDB. Hver kant er representert med en verdi i hvert panel. I eksempelet har en av kantenhetene ikke sendt data på 15 minutter, og har derfor status «crit» i panelene.

Figur 4.26: Dashboard, AlertDash

EdgeList

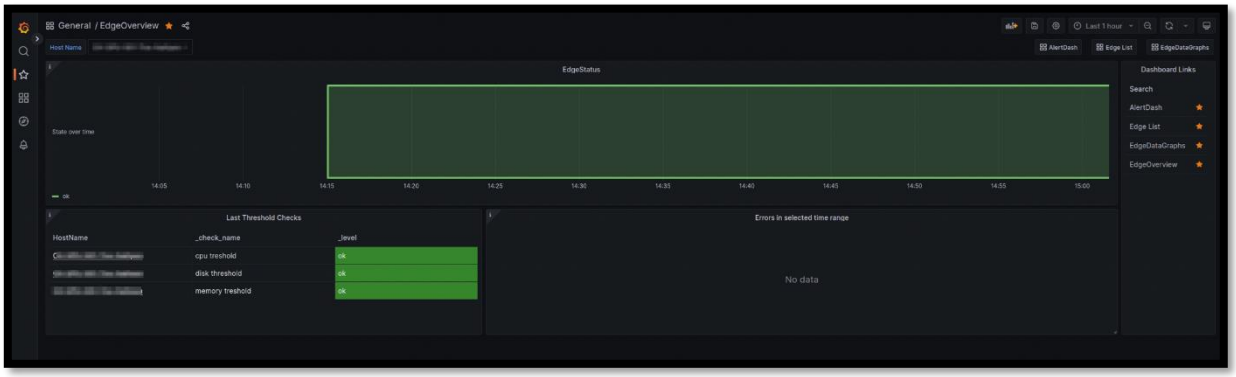
Figur 4.27 viser Edgelist og består av tre paneler. Klikkbare lenker i «Links for EdgeOverview» og «Dashboard Links». «State Timeline» panelet viser status over tid for alle enhetene i systemet, der hver enhet har sin egen tidslinje.



Figur 4.27: Dashboard, EdgeList

EdgeOverview

«EdgeOverview» inneholder de tre panelene: «EdgeStatus», «Last Threshold Checks» og «Errors in selected time range». «EdgeStatus» er allerede vist i kapittel 4.4.1 (figur 4.21). «Last Threshold Checks» inneholder siste status fra alle threshold-sjekker. «Errors in selected time range» viser alle feil eller varsler på enheten i valgt tidsrom. Ved ingen feil vises «No Data».



Figur 4.28: Dashboard, EdgeOverview

EdgeDataGraphs

«EdgeDataGraphs» inneholder fem paneler. De fire øverste panelene i figur 4.29 viser grafer for enkelte metrikker, nederste viser grafer for alle metrikker på valgt enhet.

Figur 4.29: Dashboard, EdgeDataGraphs

5 Evaluering

Resultatene evalueres ut fra evalueringsplanen beskrevet i kapittel 3.5. Gruppen har gjennomført kontinuerlig evaluering i form av enhets- og brukertesting og sluttevaluering i form av intervju. Formålet med evaluering er å kunne validere om løsningen fungerer slik den skal, og løser problemene presentert i kapittel 1.3.

5.1 Brukertesting

Prosjekteier hadde tilgang på løsningen fra da testmaskinen ble satt opp i midten av mars og kunne kontinuerlig teste funksjonalitet og brukergrensesnittet i Grafana.

5.2 Enhetstesting

På slutten av hver sprint skrev gruppen egne datapunkter til InfluxDB med formål å avdekke uventet adferd. Tiden løsningen var i bruk hadde ingen av kantenhetene sendt verdier utenfor «ok» verdiområdet i threshold-sjekkene, innsending av testdata var derfor høyst nødvendig med hensyn på håndtering av status fra sjekkene. Gruppen brukte eksempelskriptet i figur 4.17 med hardkodede verdier til å utløse varsler på flere av sjekkene samtidig for å se hvordan det påvirket visualiseringen i Grafana.

5.3 Intervju

Avslutningsvis gjennomførte gruppen intervju med prosjekteier. Prosjekteier ble forelagt spørsmål tilknyttet funksjonelle egenskaper beskrevet i visjonsdokumentet og oppgavens problembeskrivelse og mål.

Gruppen tok lydopptak av intervjuet og har transkribert i etterkant. Spørsmålene er delt inn i kategoriene tilgjengelighet på enhetsdata, innsamling av data, visualisering av data, skalerbarhet og generelt. Transkribert svar er i tekstboks rett under spørsmål.

Eier er forelagt noen «definisjoner» brukt i oppgaven. Definisjonene konkretiserer spørsmålene og reduserer muligheten for tolkning.

- InfluxDB: Databaseinstansen.
- Grafana: visualiseringsplattformen.
- Enhetsdata: for eksempel temperatur og minneutnyttelse.
- Kontekstdata: Metrikker over større tidsperspektiv og i sammenheng med hverandre.
- Horisontalt skalerbar: Legge til nye enheter.
- Vertikalt skalerbar: Legge til nye metrikker på enhet og nye monitoreringsutløsere.

5.4 Resultater

I dette delkapittelet deles resultatet fra de ovennevnte evalueringsmetodene delt inn i kontinuerlig og slutt.

5.4.1 Kontinuerlig evaluering

Gruppen brukte både brukertestingen og enhetstestingen til utbedring av løsningen. Under står noen av feilene med beskrivelse, årsak og løsning.

Enhetstesting
Enhetstesting resulterte i to feiloppdagelser:

«Unknown» status:

- Beskrivelse: En av threshold-sjekkene returnerte status «unknown».
- Årsak: Sjekken var konfigurert slik at noen verdier ikke hadde definert status se figur 4.x. Verdier under 80% hadde status «ok» og verdier mellom 80%-90% hadde status «info». Når sjekken mottok en verdi på 80% returnerte den derfor «unknown».
- Løsning: Endret «info» til å inkludere endeverdiene.

Aggregering av sjekker:

- Beskrivelse: Ved flere varsler på enkelt enhet skal mest alvorlige vises. «Info» ble vist i flere paneler til tross for at enheten hadde «crit» på deadman.
- Årsak: Logikken i Flux-skriptene. Ved aggregering av sjekker ble resultatet fra siste av sjekkene i alfabetisk rekkefølge
- Løsning: Implementerte ytterligere logikk for aggregering ved hjelp av pivot() og map(). Vises i figur 4.x.

Brukertesting
Prosjekteier hadde flere tilbakemeldinger på grensesnittet underveis i utviklingen. Flere av tilbakemeldingene gjaldt enklere endringer som fargekoding eller størrelse på paneler andre gjaldt større endringer. Figur 4.25 som viser hierarkiet for dashboard og er utformet basert på tilbakemeldingene fra brukertestingen. Eier ønsket at hvert dashboard har hver sin funksjon. «AlertDash» og «EdgeList» viser status for systemet som helhet, henholdsvis momentant og over tid. «EdgeOverview» og «EdgeDataGraphs» viser informasjon for en enkelt enhet over tid, henholdsvis statuser/feil og enhetsdata.

5.4.2 Sluttevaluering

Under er spørsmålene fra intervjuet med påfølgende svar i tekstbokser.

Tilgjengelighet på enhetsdata:

- Hvordan er tilgjengeligheten på loggdata i dagens system sammenlignet med tidligere?

Svært begrenset tidligere. Måtte tidligere logge inn på enhet og bruke operativsystemets «system monitoring» med tidsbegrensning på 100 minutter. Mye høyere tilgjengelighet nå, trenger kun å logge inn i Grafana.

- Hvordan er tilgjengeligheten på kontekstdata i dagens system sammenlignet med tidligere?

Mye høyere, hadde ikke tilgang på kontekstdata tidligere.

Innsamling av data:

- Hvor enkelt/vanskelig har det vært å overføre data til InfluxDB? Gjerne forklar noe av prosessen.

Ganske enkelt, tatt utgangspunkt i eksempelscriptet prosjektgruppen produserte. Laget ny modul i internt system, ferdig modul er enkel å legge til på nye enheter. Gir høy foranderlighet, kan lage nye moduler ved behov for monitorering på nye metrikker.

- Har dere måtte gjøre noen tilpasninger i deres systemer, eventuelt hvilke?

Utelukkende å skrive moduler for ønskede data, lite tidkrevende prosess.

Visualisering av data:

- I hvilken grad dekker visualiseringen i «AlertDash» og «Edge List» behovet for overordnet monitorering av systemet deres (på implementerte metrikker)?

Dekker absolutt behovet. Gir en momentan oversikt på enhetsstatuser (AlertDash) og enhetsstatuser over tid (Edge List) og gjør det enkelt å oppfatte feil.

- I hvilken grad dekker «EdgeOverview» og «EdgeDataGraphs» behovet for innsikt på loggdata for hver enhet (på implementerte metrikker)?

«Edge overview» tillater for annotering av nyttige hendelser på state timeline og tilstand over tid. «EdgeDataGraphs» gjør det enkelt å oppdage endringer over tid, eksempelvis om minnebruken stiger gradvis selv om den ikke skal det.

Skalerbarhet:

- Horisontalt skalerbar:
    - Hvordan oppleves prosessen å legge til nye enheter i løsningen? Nevn gjerne utfordringer/fordeler.

Svært enkel. Legger til modul på ny enhet.

- Vertikalt skalerbar:
    - Hvordan oppleves prosessen å legge til nye metrikker i løsningen? Nevn gjerne utfordringer/fordeler.

Noe mer komplisert enn å legge til ny enhet, men fremdeles enkelt. Eksempelvis om man ønsker overvåkning GPU. Når en modul først er laget, kan den legges til på alle ønskede enheter.

    - Hvordan oppleves prosessen å legge til nye monitoreringsutløsere i løsningen? Nevn gjerne utfordringer/fordeler.

Enkelt. Måtte ha noe veiledning i starten med tanke på struktur på sjekkene men ikke mye.

Generelt:

- Kan du forklare kort om hvordan/hvor ofte dere bruker løsningen i dag? Gjerne inkluder spesifikt tilfelle.

Bruker den flere ganger om dagen. Brukes rutinemessig; morgen, midt på dagen og slutten av dagen for å se at alle enheter er oppe. Brukes mye til å kunne observere «mean latency», er tre enheter med relativt kort avstand mellom hverandre med høy variasjon. Før løsningen var det antatt at problemet gjaldt for hele området, men så fort at det ikke var tilfellet når det ble enkelt å sammenligne data fra enhetene. Dette er også nyttig for å feilsøking og sjekke om tiltak fungerer. Brukes også til å sjekke om datainnsamling fra enheten fungerer som den skal ved å sjekke «disc_usage». Dette er noen eksempler på foreløpig bruk.

- Tror du løsningen kan forenkle feilsøking på enheter? Hvorfor/hvorfor ikke?

Se forrige svar

- Tror du løsningen kan bidra til lavere nedetid på enheter? Hvorfor/hvorfor ikke?

Ja, både preventiv og reaktivt. Nå får vi beskjed om at enheten er nede innen kort tid, og kan iverksette tiltak med en gang. «Threshold» sjekkene gjør at en kan gripe inn før feilen oppstår, for eksempel varsling på økende temperatur på enheter.

- Har du noen ønskede endringer til slik løsningen er i dag?

Vanskelig å si, løsningen har vært i utvikling/i bruk parallelt over siste måneden. Trenger nok mer tid (noen måneder) med ferdig løsning for å vite sikkert. De forskjellige fremstillingene i Grafana (panelene) er enkeltvis gode, mulig det burde gjøres endringer med tanke på flyt (navigering mellom dashboards).

- Hvilke ønsker hadde dere hatt om gruppen skulle fortsatt prosjektarbeidet?

- Tettere integrasjon mot dagens system (med tanke på infrastruktur), står foreløpig separat (på testmaskinen).
- Mer intelligent varsling (eksempelvis predikere feil som kommer til å oppstå)
- Implementere visning av systemlogger for hver maskin.
- Mulighet for å «kvittere» ut feil med signatur. Slik at varslingen fjernes og viser signatur til personen som gjorde det.

- I hvilken grad føler du prosjektgruppen har gjort tilpasninger basert på tilbakemeldinger underveis? Gjerne inkluder eksempel.

I høy grad. Prosjektgruppen har sittet mye på kontoret gjennom utviklingsfasen. Har vært lite behov for å spørre etter endringer, fordi prosjektgruppen har spurt effekten/nytten av endringer kort tid etter implementasjon. Tett dialog med prosjektgruppen har gjort at «views» (dashboards) i Grafana dekker behovet.

6 DISKUSJON

I denne delen av oppgaven diskuteres konsekvensene av valgene prosjektgruppen har tatt underveis med hensyn på valg av oppgave, avvik fra initial løsnings-ide, valgt arbeidsprosess og verktøy. Til slutt oppsummeres løsningens styrker og svakheter ikke allerede dekket i kapitelet.

6.1 Valg av oppgave og avgrensinger

Det kommer fram i GANTT-skjemaet og timelistene i prosjekthåndboken at gruppen brukte mesteparten av tiden i januar-februar på valg av oppgave og avgrensing.

Valg av oppgave:
Kartleggingen av mulige oppgaver er fremstilt i tabell 2.1. Tabellen kan grovt oppsummeres i to trekk; i hvilken grad er prosjektet gjennomførbart (Tidligere erfaring og tilpasningsbehov system og skaleringspotensiale) og hva er den forventede nytteverdien av løsningen.

Middels erfaring fra feltet resulterte i at gruppen brukte mye tid på å kartlegge mulige løsninger og vurdere utviklingsverktøy, dette kommer også fram i timelistene i prosjekthåndboken.

Fleksibilitet i evalueringen viste seg å være bedre enn forventet. Siden løsningen ble implementert i mars, kunne prosjekteier brukerteste løsningen i nesten to måneder.

Avgrensinger:
Prosjektgruppen avgrenset hvilke enhetsdata løsningen skulle inneholde til CPU-utnyttelse, minneutenyttelse og temperatur. Hensikten med avgrensingen var å prioritere arbeidet med å gjøre løsningen skalerbar for CH. Dersom gruppen skulle ha gjennomført dette selv, ville det ha krevd tid for å fastslå hvilke metrikker som skulle brukes og hvordan dataene skulle organiseres. Prosjekteier, som har bedre kjennskap til CHs system, har bedre forutsetninger for å avgjøre hvilke metrikker som er hensiktsmessige å monitorere. Under «Vertikalt skalerbar» i intervjuet kommer det fram at løsningen trengte liten tilpasning for å legge til flere metrikker og nye sjekker. I intervjuavsnittet Under «Vertikalt skalerbar» kommer det fram at løsningen trengte lite tilpasninger for å legge til flere metrikker og nye sjekker.

6.2 Avvik fra initial løsnings-ide

I starten av utviklingsfasen planla prosjektgruppen at Telegraf skulle håndtere måling og overføring av data. Ulempen med Telegraf er at det må installeres og administreres på hver enkelt enhet. Siden gruppen avdekket at prosjekteier ønsket minst mulig tilpasninger på kantenhetene, bestemte gruppen seg for at sending skulle gjøres manuelt til InfluxDB API-et. Sending av data via HTTP POST-forespørsler med Urllib3 var også adskillig enklere fordi CH allerede brukte dette i eksisterende moduler.

6.3 Arbeidsprosess og valgte verktøy

Riktig valg av arbeidsprosess og verktøy var nødvendig om gruppen skulle oppnå målene fra kapittel 1.3. Verktøyene måtte lagre og monitorere nødvendig informasjon og visualiseringen måtte tilgjengeliggjøre denne informasjonen for bruker i webgrensesnittet. Hvis begge disse kriteriene er oppfylt kan bruker raskt oppdage feil eller forhindre de før de oppstår. Eneste måten å kontrollere dette, er tilbakemeldinger fra prosjekteier.

Prosjekteier indikerte i siste svaret fra intervjuet at gruppen har utført meningsfulle endringer i løsningen, basert på tilbakemeldinger. Valget av en agil iterativ arbeidsmetodikk med kontinuerlig brukertesting ser derfor ut til å ha påvirket resultatene positivt. Eksempelvis er organiseringen av dashboard og panelene direkte konsekvens av tilbakemeldinger fra brukertest og refereres til i samme spørsmålet som «behovdekkende».

Videre indikerte prosjekteier i svarene fra «Skalerbarhet» at utvidelse av metrikker, monitorering og enheter var forholdsmessig enkel. Tilbakemeldingen tyder på at valget av InfluxDB og Line Protocol har positivt påvirket skalerbarheten i løsningen.

6.4 Styrker og svakheter i endelig løsning

Den endelige løsningen har både styrker og svakheter. Ettersom både InfluxDB og Grafana har brukervennlige grensesnitt, vil prosjekteier antageligvis ikke trenge mye tid på å bli kjent med løsningen. På en annen side er funksjonalitet prisgitt verktøyene. Hvis visualiserings- eller datalagringsbehovet endrer seg over tid og Grafana/InfluxDB ikke kan dekke det behovet, må hele den delen av løsningen byttes. Datagrunnlaget og dataflyten vil derimot være bevart og vil antageligvis ha noe overføringsverdi.

Per nå kjører både InfluxDB og Grafana på testmaskinen hos CH, den må være tilkoblet 24/7 for at løsningen skal fungere. Prosjektgruppen har tatt back-up av nødvendige filer i tilfelle noe skulle skje med maskinen, men det vil fremdeles ta tid å sette opp på nytt og data vil kunne gå tapt. Når/hvis testmaskinen ikke lenger dekker de fysiske kravene til løsningen, må den enten oppgraderes eller løsningen må flyttes på skytjenester og vil med dermed ikke lenger være gratis.

Ytterligere svakheter avdekkes i spørsmålet om ønsket videre arbeid for prosjektet. Løsningen er kun delvis integrert i CH sine systemer og varslinger baserer seg utelukkende siste datapunkt og kan dermed ikke predikere fremtidige feil. Videre mangler løsningen visning av systemlogger og prosjekteier kan ikke kvittere ut feil av visualiseringen.

Når modulene til CH aggregerer data over 15 minutters perioder, er dette både en styrke og svakhet. Behovet for datalagring og antall forespørsler mot databasen blir mindre, samtidig «forsvinner» verdier inn i aggregeringen. Flere datapunkter er positivt ved dataanalyse.

7 KONKLUSJON OG VIDERE ARBEID

I dette kapittelet presenteres konklusjoner om grad av måloppnåelse basert på problembeskrivelse og mål i kapittel 1, resultater i kapittel 5 og diskusjon i kapittel 6. Gruppen fremlegger forslag til videre arbeid basert på egne erfaringer og tilbakemeldinger fra prosjekteier.

7.1 Måloppnåelse

Problembeskrivelsen i kapittel 1.3 viser hvordan CHs mangel på enhetsdata resulterer i nedetid og dermed negativt påvirker tjenestene de leverer. Feiloppdagelser kan ta tid, og kunne mulig vært avverget ved monitorering. Gruppen stilte derfor spørsmålet:

Hvordan kan innsamling, visualisering og monitorering av enhetsdata forhindre nedetid?

Målet med oppgaven ble å utvikle et verktøy. Verktøyet skulle:

- Lagre enhetsdata og visualisere det i en webapplikasjon.
- Bruker skulle kunne observere tilstand for enkeltenheter og systemet som helhet.
- Utforming av verktøyet skulle gjennomføres i samråd med oppdragsgiver for å sikre dekning av reelle behov.
- Oppdragsgiver skulle kunne bygge videre på verktøyet.

Basert på sluttevalueringen i kapittel 5 og diskusjonen i kapittel 6 konkluderer gruppen med høy måloppnåelse.

- Verktøyet lagrer enhetsdata i InfluxDB og visualiserer det i webapplikasjonen i Grafana.
- Basert på svarene i intervjuet kan bruker ved hjelp av dashboard enkelt observere tilstand for enkeltenheter og systemet som helhet.
- Fra kapittel 6.3 kommer det tydelig fram at valgt arbeidsprosess resulterte i at utformingen er gjort i samråd med oppdragsgiver.
- Svarene under skalerbarhet i intervjuet viser at oppdragsgiver opplevde prosessene; legge til nye metrikker, legge til nye enheter og monitorere nye verdier som enkle og lite tidkrevende.

Siden prosjekteier selv sier i intervjuet at løsningen bidrar til å forhindre nedetid, både preventivt og reaktivt konkluderer gruppen at:

Ved å samle og monitorere enhetsdata i InfluxDB, så visualisere dataene i Grafana bidrar løsningen til å forhindre nedetid.

7.2 Videre arbeid

Gruppen har tre forslag til videre arbeid, disse baserer seg på svakhetene diskutert i kapittel 6.4.

Integrere verktøyet i CH-infrastruktur

Instansene av InfluxDB og Grafana kjører lokalt på testmaskinen. Om noe skulle skje med den, mister CH tilgang på verktøyet og all innsamlet data. InfluxDB/Grafana må settes opp og konfigureres på nytt basert på systemdokumentasjonen.

Ved å flytte verktøyet over på CH nåværende infrastruktur kan den enklere skaleres ved behov, datatap kan unngås og tilgjengeligheten avhenger ikke av en enkelt maskin.

Videreutvikling av Grafana

Flere av delene i webgrensesnittet kan utbedres. Både flyten mellom og innhold i dashboard. Prosjekteier ønsker eksempelvis samling av loggmeldinger og mulighet for å «kvittere» ut feil.

Databehandling og feilpredikering
All data lagret i databasen er nå aggregert over 15 minutters intervaller. CH ønsker og en mer intelligent metode for å varsle feil, om mulig predikering av fremtidige feil. Hvis løsningen skal predikere feil er det nærliggende å tro at data aggregert over kortere tidsintervall vil være nyttig. InfluxDB støtter automatisk aggregering av data. Tidsserier med kort intervall kan derfor brukes i dataanalyse, så aggregeres ved en viss alder for å bruke mindre lagring.

8 REFERANSER

Bibliografi

Al-Baik, O. & Miller, J., 2014. a systematic review. *The kanban approach, between agility and leanness:*, 11 Desember, pp. 1861-1897.

Aven, T., 2023. *Store Norske leksikon.* [Internett]
Available at: https://snl.no/risikoanalyse
[Funnet 20 Mai 2023].

betterstack, 2023. Prometheus vs Grafana. *The Key Differences to Know*, 3 Mars.

Bobriakov, I., 2022. *metricfire.* [Internett]
Available at: https://www.metricfire.com/blog/grafana-vs-chronograf-and-influxdb/
[Funnet 20 Mai 2023].

Brooke, J., 1996. *SUS: A quick and dirty usability scale.* United kingdom: s.n.

Budiu, R., 2017. Quantitative vs. Qualitative Usability Testing. 1 Oktober.

Budiu, R., 2018. Quantitative vs. Qualitative Usability Testing. 1 Oktober.

DB-Engines, 2023. *DB-Engines.* [Internett]
Available at: https://db-engines.com/en/ranking/time+series+dbms
[Funnet 19 Mai 2023].

Fu, T.-c., 2010. A review on time series data mining. *Engineering Applications of Artificial Intelligence. vol 24*, 10 Oktober, pp. 164-181.

Ibekwe, U., 2022. Usability studies: How many participants are enough?. 23 Mars.

Ibekwe, U., 2022. Usability studies: How many participants are enough?. 23 Mars.

IBM, 2021. Distributed computing. *CICS Transaction Server for z/OS*, 3 Mars.

influxdata, 2022. *What is Chronograf?.* [Internett]
Available at: https://www.influxdata.com/time-series-platform/chronograf/
[Funnet 19 Mai 2023].

influxdata, 2023. *Use Grafana with InfluxDB OSS.* [Internett]
Available at: https://docs.influxdata.com/influxdb/v2.6/tools/grafana/
[Funnet 19 Mai 2023].

Khan, W. Z. et al., 2023. Edge computing: A survey. *Future Generation Computer Systems. volum 96*, 8 Mars, pp. 219-235.

Kufel, Ł., 2016. Tools for Distributed Systems Monitoring. *Foundations of Computing and Decision Sciences. vol 41*, 13 Desember, pp. 237-260.

Lin, M.-S. & Chen, D.-J., 1997. The computational complexity of the reliability problem on distributed systems. *Information Processing Letters. vol 64*, 19 Mai, pp. 143-148.

MEARDON, E., 2023. *atlassian.* [Internett]
Available at: https://www.atlassian.com/agile/project-management/gantt-chart
[Funnet 8 Mars 2023].

Musa, E., Delač, G., Šilić, M. & Vladimir, K., 2019. Comparison of Relational and Time-Series Databases for Real-Time Massive Datasets. 20 Mai, pp. 1060-1070.

Olan, M., 2003. Unit testing: Test early, test often. *Journal of Computing Sciences in Colleges*, 2 Januar, pp. 319-328.

Powell J, D., 2018. Kanban for Lean Production in High Mix, Low Volume Environments. *IFAC-PapersOnLine. vol 51*, 6 September, pp. 140-143.

Prometheus , 2022. *Prometheus.* [Internett]
Available at: https://prometheus.io/docs/prometheus/latest/querying/basics/
[Funnet 19 Mai 2023].
Rodola, G., 2023. *psutil 5.9.5.* [Internett]
Available at: https://pypi.org/project/psutil/
[Funnet 13 Mars 2023].
Sargeant, J., 2012. Qualitative Research Part II: Participants, Analysis, and Quality Assurance. *Journal of graduate medical education*, 4 Mars, pp. 1-3.
Sisense Inc, u.d. *Contextual Data.* [Internett]
Available at: https://www.sisense.com/glossary/contextual-data/
[Funnet 8 februar 2023].
Sor, V. & Narayana Srirama, S., 2011. A Statistical Approach for Identifying Memory Leaks in Cloud Applications. *International Conference on Cloud Computing and Services Science*, 1 Januar, pp. 623-628.
Zhou, J., Gandomi, A., Chen, F. & Holzinger, A., 2021. A Survey on Methods and Metrics. *Evaluating the Quality of Machine Learning Explanations:*, 1 Mars, p. 593.

9 VEDLEGG

Følgende dokumenter er vedlagt:

- Prosjekthåndbok
- Visjonsdokument
- Kravdokument
- Systemdokumentasjon
- ZIP av dashboard-JSON-modeller