UKE 13
DevOps og håndtering av kode

IN1030 - Gruppe 8

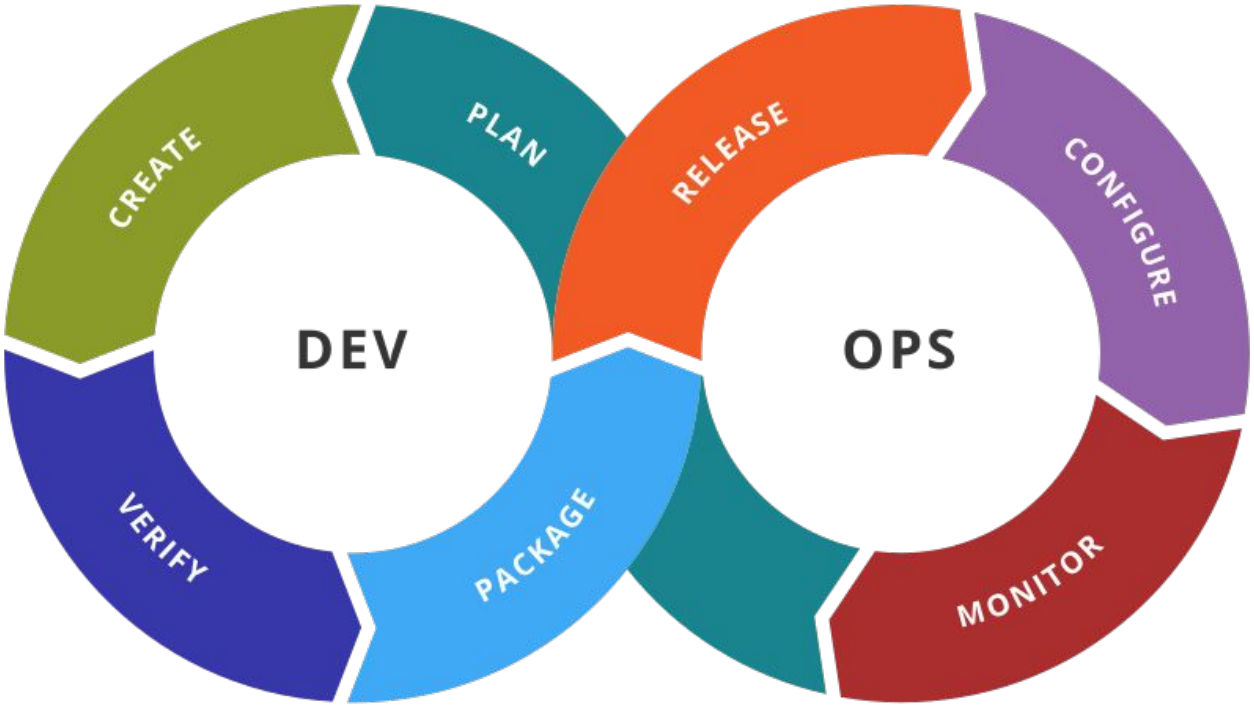
Plan for timen

Hva er DevOps

- Prinsipper
- Kode- og versjonshåndtering
- Automatisering
- Metrikker

Summeoppgaver

Jobbing med oblig 5



Læringsmål

...kjenner du til **ulike faser og aktiviteter** som inngår i systemutvikling.

...kan du anvende metoder og teknikker for kravhåndtering, utføre modellering ved hjelp av UML, og **vurdere fordeler og ulemper ved forskjellige metoder og teknologier for systemutvikling**.

DevOps

Development & Operations

Hva er DevOps?

- Vanligvis har produktutvikling og vedlikehold bestått av flere team:
  - Utvikling
  - Utgivelse
  - Support
  - Vedlikehold
- Problemer:
  - Forsinkelser i kommunikasjonen mellom teamene og av løsninger på problemer
  - Bruk av ulike verktøy, ulike ferdigheter og manglende forståelse av de andres problemer

- DevOps er en ***alternativ modell*** som integrerer alt ovenfor i et eneste team.
- Ingen fast definisjon fordi alle bedrifter implementerer det forskjellig
- Spesielt nyttig for skybaserte tjenester

DevOps-Prinsipper

DevOps-prinsipp	Forklaring
Alle ansvarlige for alt	Alle i teamet har delt ansvar for utvikling, utgivelse og vedlikehold/support av programvaren.
Alt som kan bli automatisert burde bli det	All testing-, utgivelse- og supportsaktiviteter bør bli automatisert når det er mulig. Legg opp til minst mulig manuelt arbeid med utgivelsen av programvaren.
Mål først, endre etterpå	DevOps burde bli drevet av målingsprogram hvor du samler data om systemet og operasjonene. Avgjørelser som angår endring i DevOps prosessen og verktøy, bør tas med denne dataen som grunnlag.

Sommerville, kap. 10, s. 297

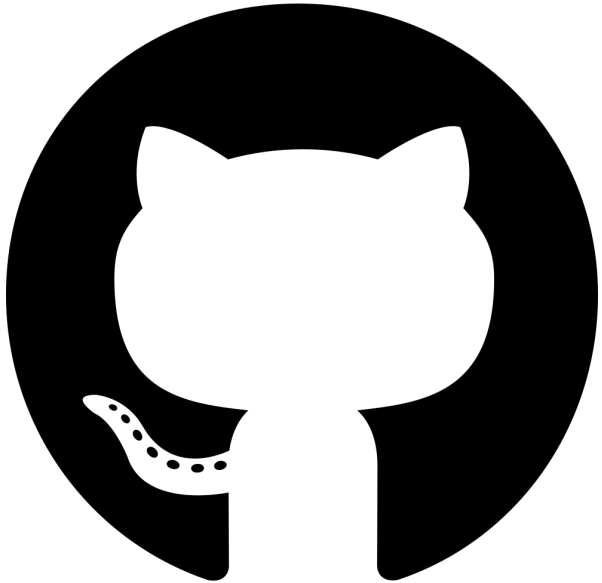
Fordeler ved DevOps

- **Raskere utgivelse/leveranse (Deploy)**
  - Fordi: kommunikasjonsdelay mellom teamene er blitt dramatisk redusert
- **Redusert risiko**
  - Fordi: inkrementet i hver leveranse er mindre, og dermed enklere å lokalisere kilden til problemet
- **Raskere reparasjon**
  - DevOps teamene jobber sammen for å få programvaren oppe å gå
  - Man trenger heller ikke å finne ansvarlig team og vente på at de fikser det dersom et problem oppstår
- **Produktive team**
  - Teamene blir gladere og mer produktive fordi de jobber sammen og ikke på separate områder

Sommerville, kap. 10, s. 298

Kodeadministrasjon og versjonshåndtering

- **Kodeoverføring**
  - Utviklere tar kode inn i deres personlige fillager for å jobbe med det, så returnerer de det til det delte kodeadministrasjonssystemet
- **Versjonlager og henting**
  - Filer kan bli lagret i forskjellige versjoner, og spesifikke versjoner kan bli hentet
- **Merging/Branching**
  - Parallellutviklingsgrener kan bli laget for å bli jobbet med samtidig. Endringer gjort av utviklere på forskjellige branches kan bli merget
- **Versjonsinformasjon**
  - Informasjon om de forskjellige versjonene holdt i systemet kan bli lagret og hentet



GitHub er et eksempel på verktøy som kan brukes

Aspekter av automatisering i DevOps

- **Kontinuerlig integrasjon (integration)** - kodeendring, testing.
  - *Ved integrasjon av kode til masterbranchen, lages en ny versjon av systemes som kan testes*
- **Kontinuerlig levering (delivery)** - simulering av brukssituasjon og testing
  - *Omgivelsene for systemet gjenskapes, og den nye versjonen testes heri.*
- **Kontinuerlig utgivelse (deployment)** - oppdatering når masterbranch endres, gjøres tilgjengelig for brukere
  - *Når systemet er testet og oppdatert, opdateres den utgivne versjonen*
- **Infrastruktur som kode** - maskinprosesserbar kode, ingen manuell oppdatering av software
  - *Oppsett av systemets infrastruktur som maskinlesbar kode.*

Diskuter i grupper

Hvilke fordeler får vi ved å automatisere og gjøre integrering, levering og utgivelse automatisk?

Fordeler med kontinuerlig utgivelse

1. **Redusert kostnad:** kontinuerlig utgivelse hurtigere og mer effektivt enn manuell - som også ofte feiler.
2. **Hurtigere problemløsning:** problem påvirker kun en liten del, fordi ting oppdateres kontinuerlig, enklere å finne årsak til feil.
3. **Hurtigere kundefeedback:** identifisere forbedring gjennom å stadig gi ut nye funksjonaliteter.
4. **A/B testing:** noen bruker gammel og andre ny versjon - kan sjekke hva som fungerer bra / best.

Sommerville, kap. 10, s. 318

Metrikker

DevOps - opererer utifra ønske om å kontinuerlig forbedre seg for å oppnå raskere utgivelse og bedre kvalitet på programvaren.

Data og analyse danner grunnlag for kontinuerlig forbedring av process og produkt
Til dette trenger vi ***metrikker*** vi kan måle på.

Prosessmetrikker:

Gjennomsnittstid på gjenopprettelse

Prosentandel feilutgivelser

Utgivelsefrekvens

Endringsvolum

Forsinkelse fra utvikling til utgivelse

Tjeneste-/produktmetrikker:

Prosent økning i sluttbrukere

Antall klager

Tilgjengelighet

Ytelse

Metrikker for bruk og bedrifts-drift nevnes også, men er ikke kjernen hos Sommerville (s. 322).

Diskuter i grupper

Hvilke tilpasninger kan man gjøre for å forbedre prosess?

Heri: redusere gjennomsnittstid for gjenopprettelse eller utgivelsesfrekvens?

Mål

Prosessmetrikker:

Gjennomsnittstid på gjenopprettelse

Prosentandel feilutgivelser

Utgivelsefrekvens

Endringsvolum

Forsinkelse fra utvikling til utgivelse

Prosessendringsforslag:

Revurdere de automatisere testene

Sjekke kodehåndtering

Ansette flere

Revurdere målsetting

mm.

Diskuter i grupper

Hvilke tilpasninger kan man gjøre for å forbedre produkt?

Heri: redusere klager? for å øke prosentandel nye sluttbrukere?

Mål

Tjeneste-/produktmetrikker:

Prosent økning i sluttbrukere

Antall klager

Tilgjengelighet

Ytelse

Tjenesteendringer:

Markedsføring

Universell utforming

Brukerundersøkelser

Kode for robusthet

Endre systemarkitektur

Diskusjon

Hva kan være dumt med kontinuerlig utgivelse av ny programvare? Gi gjerne eksempler.

Kontinuerlig utgivelse - potensielle ulemper

Ukomplette funksjonalitet - lekkasje av geniale ideer.

Irriterte kunder som ikke oppdaterer fordi de må gjøre det så ofte.

Eks.: Mac-oppdatering, Chrome-oppdatering.

Synkronisering av bedriftenes ønsker og dine utgivelser - husk på det.

Eks.: lønnssystem til UiO, Notion-oppdatering.

Nyttige linker:

IN2030 intro til versjonskontroll og Git

https://www.uio.no/studier/emner/matnat/ifi/IN2030/h20/timeplan/versjonskontroll.mp4

DevOps:

https://www.youtube.com/watch?v=_I94-tJlovg

Jobbing med Oblig 5

Denne uken

Kapittel 10: DevOps and Code Management

(eksternt kapittel)

Neste uke - IT kontrakter + Vi ser på gamle eksamensset

Foiler

Oblig 5, frist: fredag, 6. mai, kl. 23:59.

Takk for i dag!

Har du spørsmål, så send endelig en mail på: nhmoller@uio.no