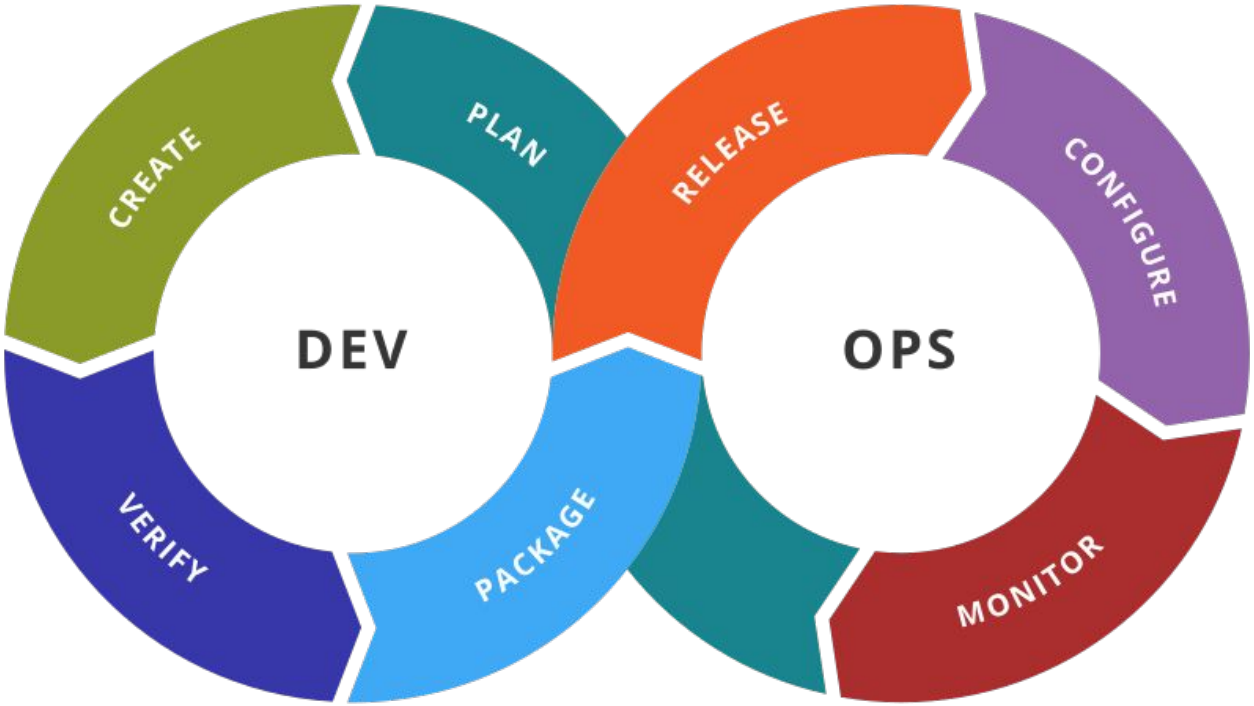
UKE 13
DevOps og håndtering av kode

IN1030 - Gruppe 1 & 3D

Plan for timen

- Hva er DevOps?
- Litt DevOps teori
- Ukesoppgaver



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**.

Nøkkelbegrep:

- DevelopmentOperations
- Automatisering
- Versjonshåndtering
- Distribusjon
- Kontinuerlig integrering
- Kontinuerlig testing

DevOps

Development & Operations

HVORFOR DET?

DevOps

- Vanligvis har produktutvikling og vedlikehold bestått av flere team:
  - Utvikling
  - Utgivelse
  - Support
  - Vedlikehold (om den jobben ikke er gjort av utviklingsteamet)

- Modellen tar høyde for at forskjellige team som ofte ikke har forståelse for andre teams problemer, kan resultere i en lengre periode før ny utgivelse kommer.

- 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 (development + operations) 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 - litt teori

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.

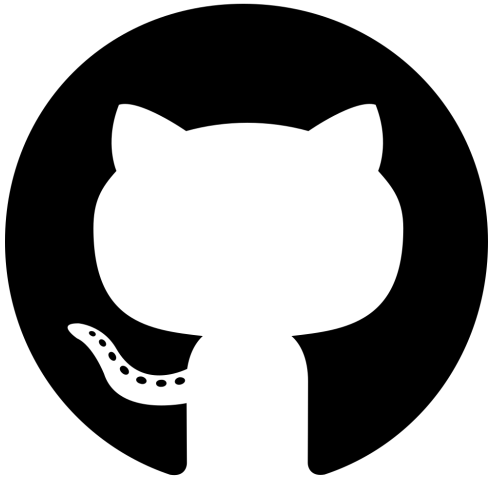
Fordeler

- Hurtigere distribusjon
- Minsket risiko
- Hurtigere bug-reparasjon
- Mer produktive team

Kodeadministrasjon

- **Definisjon**:
  Et sett med programvare-støttede praksiser for å administrere en inkrementelt voksende kodebase.

- **Kodeoverføring**
  - Utviklere tar kode inn i deres personlige fillager for å jobbe med det, så returnerer de det til det delte kodeadministrasjonssystemet
- **Versjonslagring 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



Automatisering

Kontinuerlig integrasjon - kodeendring, testing

Kontinuerlig levering - bruk-simulering og testing

Kontinuerlig utgivelse - oppdatering når master branch endres

(Jenkins)

Infrastruktur som kode - maskinprosesserbar kode, ingen manuell oppdatering av software

(Puppet)

Diskuter i grupper

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

10 min.

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 jo 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.

Mål

Prosessmetrikker:

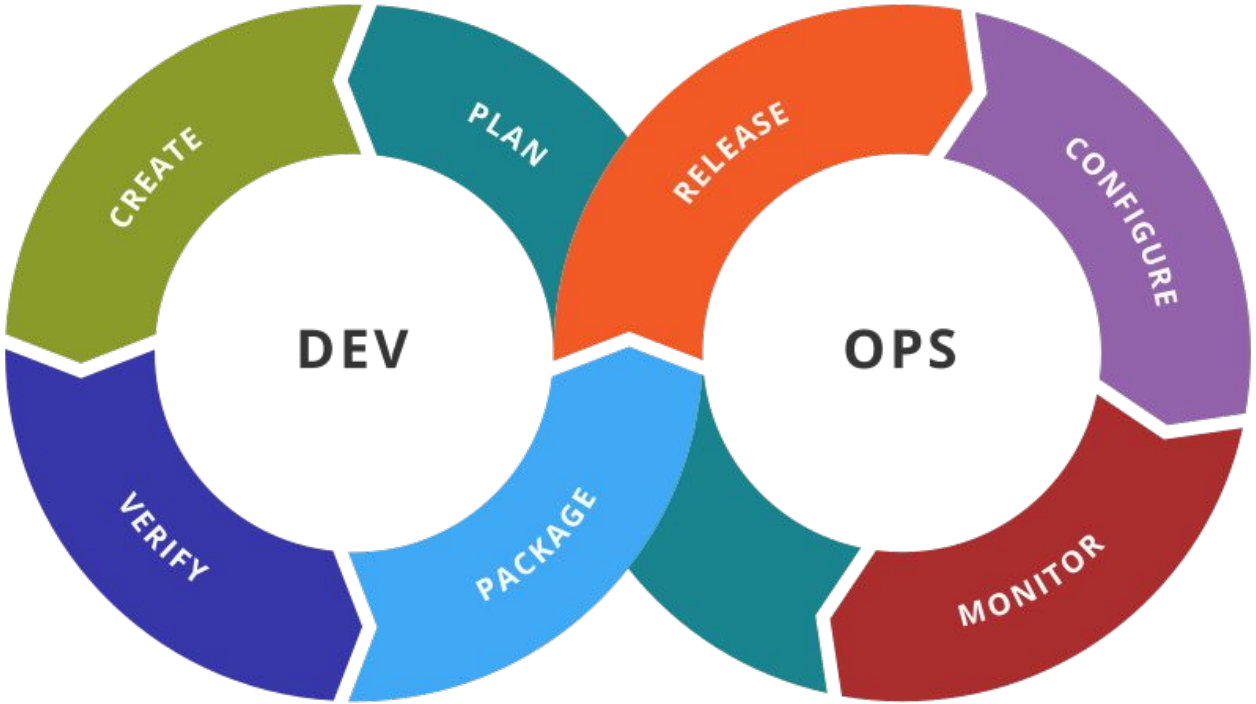
Gjennomsnittstid på gjenopprettelse

Prosentandel feilutgivelser

Utgivelsefrekvens

Endringsvolum

Forsinkelse fra utvikling til utgivelse


CREATE
PLAN
RELEASE
CONFIGURE
DEV
OPS
VERIFY
PACKAGE
MONITOR

Diskusjon

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

Summegrupper - 10 min

Kontinuerlig utgivelse - obsobs

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

Denne uken

Kapittel 10: DevOps and Code Management

(eksternt kapittel)

Neste uke - IT kontrakter

Foiler

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

jehank@uio.no
jamilakm@uio.no