UKE 14
DevOps

IN1030 - Gruppe 4

Hva skal vi i dag?

- **Litt info...**
- **DevOps**
- **Litt om IT-kontrakter**
- **Ukesoppgaver + fasit**

Gruppetimene videre

Dette er siste ordinære gruppetime! :0

Neste torsdag er det langhelg med 17. mai og Kristi himmelfartsdag og dermed ingen gruppetime.

Tirsdagsgruppetimene (gr 1 og gr 5, den 16. mai) skal være åpne som normalt, så de som ønsker gjennomgang av IT-kontrakter eller må levere oblig 5 på nytt kan gå på en av dem.

Det planlegges eksamensøving i gruppetimene før eksamen. Mer info om dette kommer…

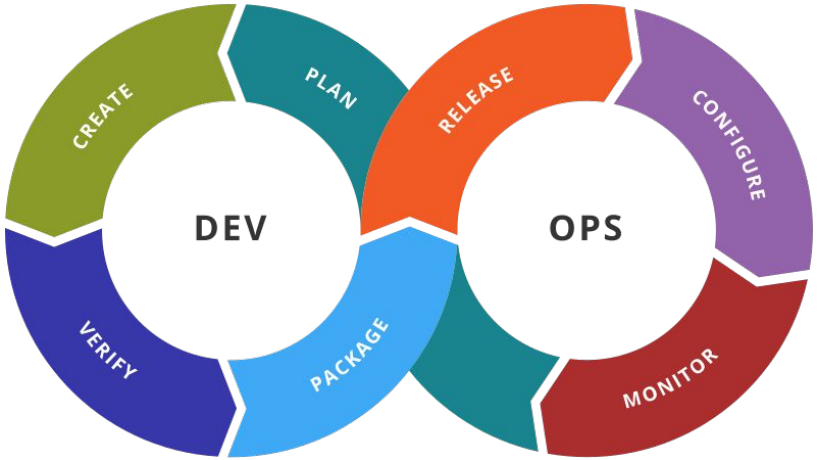
Læringsmål

...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
HVORFOR DET?



“Tradisjonelt” - Release and support

- Vanligvis har produktutvikling og vedlikehold bestått av tre team:
  - Utvikling, Utgivelse, Support, Vedlikehold (om den jobben ikke er gjort av utviklingsteamet)
- Modellen tar høyde for at de forskjellige teamene som oftest ikke har forståelse av problemene et annet team har møtt på som de må håndtere, som resulterer i en lenger periode før ny utgivelse kan komme ut

Derfor...

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 ta med denne dataen som grunnlag.

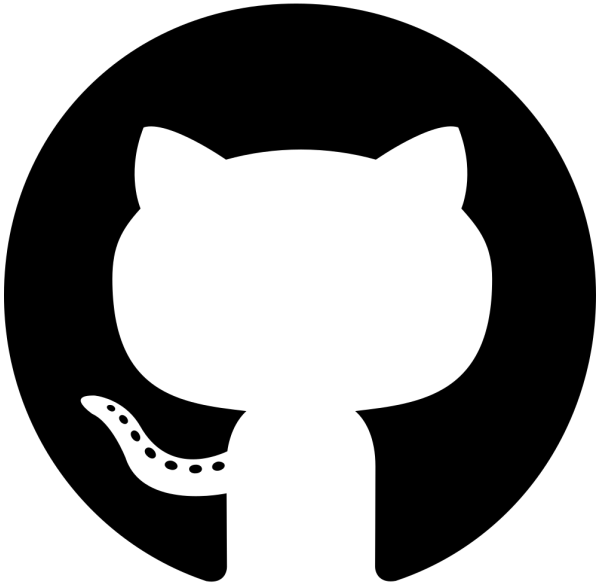
Fordeler

- **Raskere utgivelse/leveranse (Deploy)**
  - Fordi: kommunikasjons delay mellom teamene er blitt drastisk 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 går
  - 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

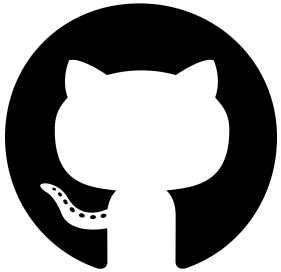
Kodeadministrasjon/ Code management

Sett med programvare-støttende praksiser for å administrere en inkrementelt voksende kodebase.

- Kodeoverføring
  - jobbe lokalt → kodeadministrasjonssystemet
- Versjonslager og henting
  - Spesifikke versjoner kan bli hentet
- Merging/branching
  - Paralell utviklingsgrener kan lages for å jobbe samtidig → merge endringene til samme løsning
- Versjonsinformasjon
  - Kontroll over endringer som blir gjort



Automatisering





Jenkins

Kontinuerlig integrasjon - Continuous integration

- kodeendring blir gjort → testing av endring implementert skjer
- enklere lokalisering av eventuelle feil
- systemet blir tilgjengelig for alle i teamet ved jevnlige oppdateringer
- Kodehåndtering - Github

Kontinuerlig levering - Continuous delivery

- Overordnet strategi
- Korte sykler som sørger for at programvaren alltid er funksjonell og klar for å slippes. Hurtig og ofte

puppet

Kontinuerlig utgivelse - Continuous deployment

- Oppdatering når master branch endres (Jenkins)
- Automatisk test → automatisk slipp

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 er 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 kunde feedback:** 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

DevOps - ønske om å kontinuerlig forbedre seg for å oppnå raskere utgivelse og bedre kvalitet på programvaren. Da trenger man å måle litt for å se hvordan ståa er…

Prosessmetrikker:

Gjennomsnittstid på gjenopprettelse

Prosentandel feilutgivelser

Utgivelsefrekvens

Endringsvolum

Forsinkelse fra utvikling til utgivelse

Diskuter i grupper

Hvilke tilpasninger kan man gjøre for å 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

Mål

Tjenestemetrikker:

Prosent økning i sluttbrukere

Antall klager

Tilgjengelighet

Ytelse

Diskuter i grupper

Hvilke tilpasninger kan man gjøre for å redusere klager? Eller for å øke prosentandel nye sluttbrukere?

Mål

Tjenestemetrikker:

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 - 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

For å oppsummere:

DevOps kan bidra til innovativ og raskere utvikling. Målet er økt samarbeid mellom operasjoner og ansatte som vil bedre kvalitet. Dette kan bidra til at man er mer responsiv til behov og krav.

Ved å implementere DevOps så kan man trenger ny mindset, skills og verktøy.

Spørsmål?

Nyttige linker:

IN2030 intro til versjonskontroll og Git

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

DevOps:

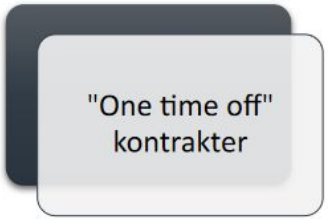
https://www.youtube.com/watch?v=_I94-tJlovg

IT-kontrakter

"En kontrakt er en avtale som mellom partene etablerer en bindende forpliktelse til å gjøre eller å unnlate å gjøre noe"

(Disse slidene inneholder kopier fra av fjorårets forelesning)

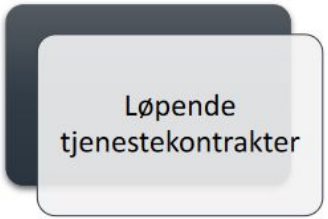
Generelle kontraktkategorier



Avtale der en part yter en bestemt service og den andre betaler én gang.



Avtale der en part yter en service der mengen er ubestemt og den andre betaler fortløpende etterhvert som de forespør mer av servicen.



Brukes ofte om driftstjenester, dersom man benytter en tredjepart til å gjøre dette istedenfor å ha alt 'in house'. Annet eks er skytjenester og kjøp av plattformtjenester.



Avtaler som inngåes når separate organisasjoner jobber sammen om noe. Hvem man har lov til å samarbeide med, fordeling av kostnader og utbytte etc...



Brukes av ofte om lokale bedrifter som faller under større forelderselskaper, der forelderselkapet kan garantere legitime tjenester for det lille selskapet og påta seg ansvar dersom de ikke oppfyller garantien.

Kontraktkategorier

Resultatansvar/ "Fossefall"

- Spesifisert resultat
- Fast pris
- Høy grad av forutberegnelighet, men begrenset fleksibilitet

Iterativgjennomføringsmodell / "Serie-fossefall"

- Definert gjennomføringsmodell og løpende spesifisering av delleveranser
- Målpris / Estimeringsmodell
- Forutberegneligheten øker og fleksibiliteten reduseres underveis i gjennomføringen

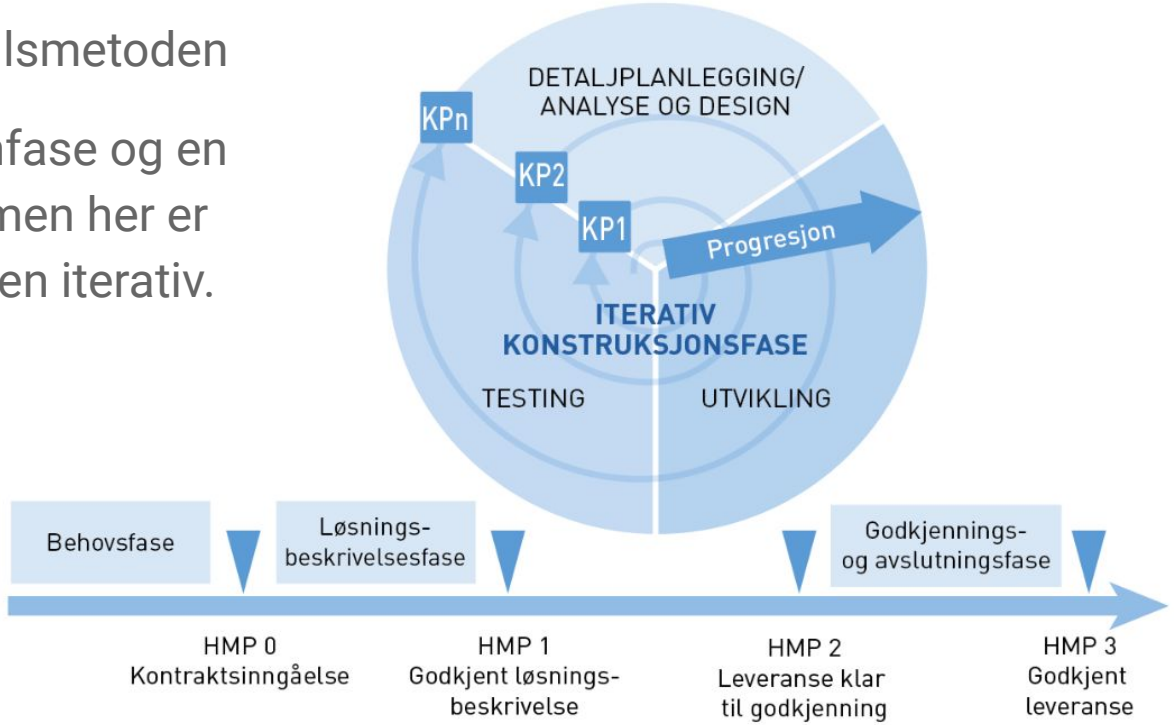
Bistandsforpliktelse / "Ressurskjøp"

- Avtalt hvilke ressurser og omfang de skal arbeide, men uten spesifisert resultat
- Betaling etter medgått tid og kostnader
- Høy grad av fleksibilitet, men begrenset forutberegnelighet

Seriefossefallsmetode

Likner på fossefallsmetoden

Begge har en planfase og en avslutningsfase, men her er den midterste fasen iterativ.



Hvilken kontraktmodell bør man velge?

Kjedelig svar, men det kommer litt an på…

"Fossefall"; når prosjekter innebærer mindre kompleksitet og ønsket resultat er klart for alle parter.

"Seriefossefall"; når resultatet kan defineres på overordnet nivå og partene er enige om å følge en avtalt gjennomføringsmodell hvor resultatet i hver delleveranse spesifiseres underveis.

"Ressurskjøp"; når det skal utvikles etter smidig metode eller resultatet av andre grunner ikke er klart definert

Kan også benytte en smidig kontraktmodell hvor leverandøren aldri overtar det kontraktsansvaret for funksjonaliteten, men er ansvarlig for ressurser, å følge avtalt gjennomføringsmodell og definerte ikke-funksjonelle krav.

Spørsmål?

Ukesoppgaver

UKESOPPGAVER

Fasit

Oppgave 10.1

Explain why adopting DevOps provides a basis for more efficient and effective software deployment and operation.

DevOps gir hurtigere og bedre utgivelser fordi – når alle er med på alt – reduserer det tiden man bruker på å kommunisere.
Man kan sette alle til å fikse bugs, alle i teamet har like stort eierskap og forståelse for hva som er lurt og ikke og små endringer i hver utgivelse gjør fatale feil mindre sannsynlig.
Automatiseringen av det meste gjør også at man kan jobbe effektivt med de viktigste arbeidsoppgavene.

Oppgave 10.2

Briefly explain why it is essential to use a code management system when several developers are involved in creating a software system. What are the benefits of using a code management system if only a single developer is involved?

Det er essensielt å bruke et kodehåndteringssystem når man er flere utviklere for å kunne kontrollere koden som styrer programvaren.
Med et håndteringssystem kan man jobbe parallelt, unngå å skrive over viktig kode, se endringshistorikk, legge til funksjonalitet uten å endre master-koden, håndtere ulike versjoner av koden, dobbeltsjekke og teste før man legger til endringer i masterkoden og håndtere lagring.

Fordelene med å bruke et kodehåndteringssystem når man jobber alene er de samme. Det er lurt å kunne se tidligere versjoner av sin egen kode og man kan da lettere backtracke, rette opp i og unngå feil.

Oppgave 10.4

What does creating a new branch in a code management system mean? What problems can arise when more than one developer is working on the same code and they try to merge their changes with the project master branch?

Når man lager en ny branch i kodehåndteringssystemet Git lager man en ny versjon av masterfilen/filen man brancher ut av. Dette gjør det mulig å endre på funksjonene i masterfilen uten å ødelegge/endre det som ligger i der.

Når man er fornøyd og har testet funksjonaliteten man har lagt til i branchen forsøker man gjerne å merge den inn i masterfilen. Dersom masterfilen da er annerledes enn den versjonen man lagde branchen sin av, skaper dette problem. Dette kan skje dersom en utvikler har merged sin branch inn i masterfilen i mellomtiden. Problemet oppstår fordi koden i din branch, som skulle vært lik med masterfilen, ikke er det lenger, og det kan oppstå konflikter som skaper nye bugs og feil. Du og den andre utvikleren må da gå gjennom koden manuelt og velge ut hva som skal være i masterfilen og hva som skal slettes.

Oppgave 10.9 (ekstraoppgave)

What are the differences between continuous integration, continuous delivery, and continuous deployment?

Svar del 1:
Continuous integration handler om kodehåndtering - at man automatisk integrerer ny, ferdig testet kode i programmet etterhvert som utviklere gjør seg ferdig med sine deler av koden. Det kan være fra ny funksjonalitet til bare en endring av variabler. Prinsippet er at koden skal være så oppdatert som mulig, godt integrert med alt man har av kode, hele tiden.

Continuous delivery handler om å sørge for at programmet alltid er klart for å slippes. Dette gjøres gjennom kontinuerlig testing, både av selve programmet, men og av funksjoner og kodesnutter som legges til. En masterbranch skal alltid være ferdig testet, klar for levering.

Oppgave 10.9 (ekstraoppgave)

What are the differences between continuous integration, continuous delivery, and continuous deployment?

Svar del 2:
Continuous deployment handler om å alltid kunne - i prinsippet - rulle ut programmet til brukerne. Man sørger for dette ved å kjøre automatiske tester av programvaren på maskinene og plattformene den er ment å fungere på.

Alle disse DevOps metodene har rot i automatisering, ved at automatisering av aktiviteter sørger for kontinuitet. Forskjellen ligger både i hvilke automatiseringverktøy man kan bruke for å oppnå kontinuitet, og hvilke steg i utvikllingsprosessen de spiller størst rolle. Continuous integration kan man sørge for med kodehåndteringsverktøy som Git, og er noe utviklerne må jobbe for å opprettholde (de må committe og pushe koden). Continuous delivery kan man sørge for med testeverktøy som SquareQube og Jenkins og som må få tilgang til master branchen i Git. Mens Continuous deployment må ha hardware/miljø å teste programvaren i - og har et høyere abstraksjonsnivå enn de to andre metodene.

En tidligere eksamensoppgave

Forklar hvorfor versjonshåndtering er viktig.

Versjonshåndtering er viktig fordi vi ofte gjør feil når vi koder. Versjonshåndtering gjør det mulig å se tilbake på hva som er blitt lagt til fra koden fungerte til den ikke fungerer lenger. På den måten kan vi lettere kontrollere og fikse feil og bugs.

akiac@uio.no