# Continuous Integration og Continuous Delivery

# DevOps – Development + Operations

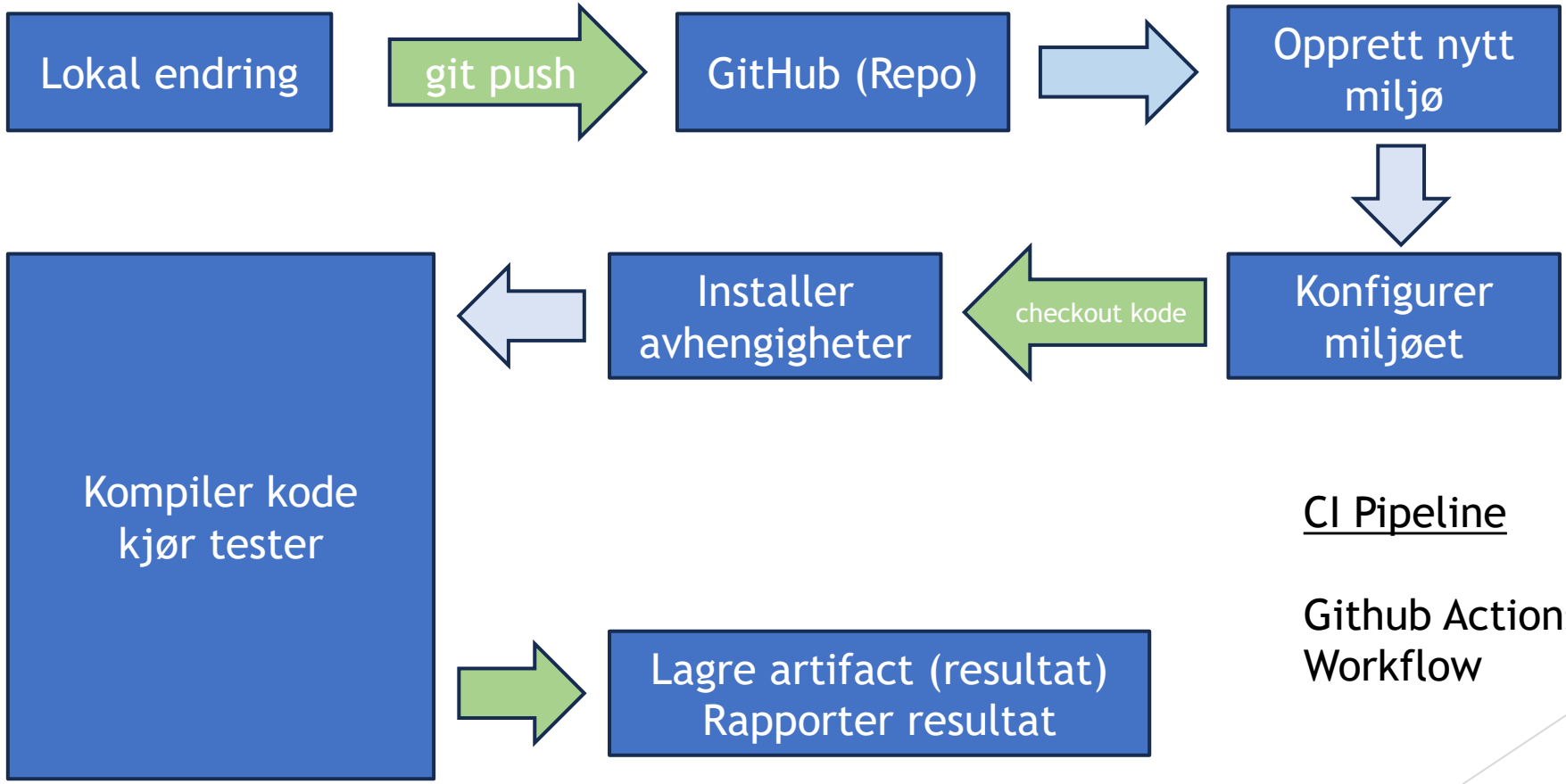
- DevOps - Både utvikling og "operasjoner" rundt produktet gjøres av ett team
  - Operasjoner – Deployment og Support
  - Reduserer tidsbruk (f.eks. kommunikasjon mellom teams)
- Grunnleggende Prinsipper i DevOps (Sommerville, 2021)
  - Alle er ansvarlige for alt
  - Alt som kan automatiseres bør automatiseres
  - Mål først endre senere
- Continuous Integration og Continuous Delivery er noen av teknikkene brukt i DevOps

# Continuous Integration

- Betyr at en integrert versjon av systemet blir bygget, kjørt og testet hver gang en endring blir pushet til remote repo-et
  - Kan gjøres for spesifikke branches
  - Skal gjøres kontinuerlig (1 eller flere ganger per dag)
  - Automatisert prosess

- Målet er å automatisk oppdage hvis endringen medbringer problemer for andre deler av systemet
  - Utviklere tester ofte bare det som er direkte relatert til endringen
  - Altså et «sikkerhetsnett» - Spesielt ved merging!
  - Varsler hvis noe er feil!

# Continuous Integration




CI Pipeline

Github Actions:
Workflow

(Mats Lindh)

# Continuous Delivery

- Continuous Delivery: Applikasjonen er alltid klargjort for leveranse (til «produksjon») uansett versjon
  - Krever at main-branchen alltid er produksjonsklar (ingen feil eller uferdig funksjonalitet)
  - Kan inkludere UI-testing, load testing, integrasjonstesting og systemtesting osv.
- Automatisert leveranse
  - Vi trykker på en «knapp» manuelt
  - Resten av prosessen er (hovedsakelig) automatisert
- Continuous Deployment – Et steg videre
  - Kode slippes automatisk til produksjon hver gang en endring blir gjort i main og alle tester passerer
  - Kult i prinsipp, men krever meget gode rutiner for å være trygt

# Arbeidsmetodikk og teamkultur

- Disse kontinuerlig teknikkene bygger på prinsippet av å gjøre hyppige og mindre endringer
- Det er vanskeligere å automatisere hvis endringer hoper seg opp (f.eks. kvartalvis deployment)
  - Stor sannsynlighet for at mye går galt på en gang
  - Større sannsynlighet for at noe ikke blir oppdaget
  - Teknologien (som CI/CD) kan ikke hjelpe oss underveis
- Altså viktig at alle utviklere følger disse prinsippene
  - Commit, push og merge ofte.
  - Skriv tester «samtidig» som koden

# Et knutepunkt

- Vi ser nå et knutepunkt mellom mange konsepter vi har vært gjennom
- Automatisering er kult! Men krever ...
  - Hyppige endringer – Git, branches, merging
  - Dekkende og automatiserbare tester – Testerammeverk, skriv underveis, low coupling, high cohesion (TDD)
  - Klart definerte avhengigheter – Byggeverktøy / avhengighetssystemer
  - Et team som understøtter alt dette

# Continuous Integration med Github Actions

- Github har en integrert mulighet for å sette opp CI workflows - Actions
  - Andre git-hosting tjenester har typisk sine egne løsninger
- Finnes mange ferdiglagde templates for workflows
  - Språk, byggeverktøy osv.
- Konfigureres via en fil i repo-et: .github/workflows/<task>.yml
- Kan brukes til å ...
  - Automatisk kjøre tester ved endringer (push, pull request osv.)
  - Gjemme hemmeligheter – apinøkler, innloggingsdetaljer, osv
    - Settes inn i prosessen når workflowen kjører
    - Blir lagret i Github, IKKE i selve repoet

# Eksempel YAML fil –CI med maven prosjekt

```yaml
name: Musicas CI

on:
  push:
    branches: [ "main", "develop" ]
  pull_request:
    branches: [ "main", "develop" ]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v4
    - name: Set up JDK 22
      uses: actions/setup-java@v4
      with:
        java-version: '22'
        distribution: 'temurin'
        cache: maven
    - name: Build with Maven
      run: |
        cd Forelesninger/Lost_Koblet_Kode/Forberedelse/musicas
        mvn --batch-mode --update-snapshots verify
```

Spesifiser prosjektet pom.xml, fra roten av git-repoet

# Mer om Github actions

- YAML syntax
- Dokumentasjon / Guide for CI med Java og Maven
  - Finnes for mange andre språk og byggeverktøy
- Deployment med Github Actions
- Bruke secrets

# maven-surefire-plugin

- For at testene skal blir kjørt ved CI må prosjektet (eller test-modulen) sin pom.xml ha lagt til maven-surefire-plugin
- Legg til det følgende innenfor den overordnede <project>-taggen

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-surefire-plugin</artifactId>
      <version>3.2.2</version>
    </plugin>
  </plugins>
</build>
```

# Krav for at maven-surefire-plugin skal funke

- For at maven-surefire-plugin skal gjenkjenne og kjøre testene må navnene på disse møte en av følgende krav ...
  - Starte eller slutte på «Test»
  - Slutte på «Tests»
  - Slutte på «TestCase»

- Det går også konfigurere maven-surefire-plugin ytterligere
  - Inkludere tester som ikke følger navnkonvensjonene over
  - Ekskludere tester
  - osv.
  - Se dokumentasjonen for mer info

# Demo – Enkel CI i GitHub

- Sette opp en GitHub Action for prosjektet musicas
- Konfigurere slik at endringer til main kjører testene
- Legge til maven-surefire-plugin i prosjektet