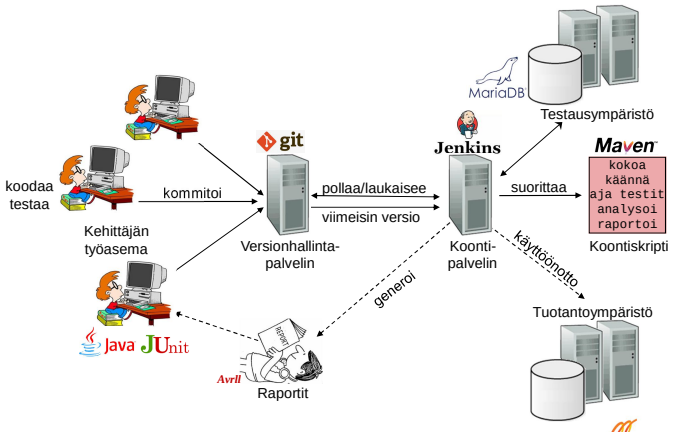
Jatkuva integrointi

Jenkins

Jatkuva integrointi

- CI, Continuous Integration
- Prosessi, jossa
  - sovelluksen uudet ja muuttuneet osat liitetään muuhun kokonaisuuteen
  - suoritetaan koonti
  - testataan koko kokonaisuutta
- Tarkoitus varmistaa, että komponenttien yhteenliittämisessä ei tule yllätyksiä
  - uudet osat / päivitykset eivät saa aiheuttaa taantumista
  - siis regressiotestataan

Jatkuva integrointi



Jatkuva integrointi

- Hakee versionhallinnasta viimeisimmät versiot
- Integroi omat viritykset mukaan
- Varmistaa, ettei omat muokkaukset riko integrointia
  - integroitava ja testattava ensin omalla koneella
  - tarkasta testitulokset: testasitko oikein? tarvitsetko uusia testejä?
- Vie testatut muutoksesi versionhallintaan tyyliin kerran päivässä
  - sekä koodi että testit, ja muuttuneet konfiguraatiot
  - jotta muulla tiimillä ne käytössä seuraavana päivänä
- Onnistunut integrointi = kaikki automatisoidut testit menevät läpi
- Seuraa integrointipalvelimen raportteja
  - ➡ epäonnistuneen integroinnin korjaaminen korkeimmalla prioriteetilla
  - tarkasta testitulokset
  - tarkasta staattisen ja dynaamisen analyysin tulokset

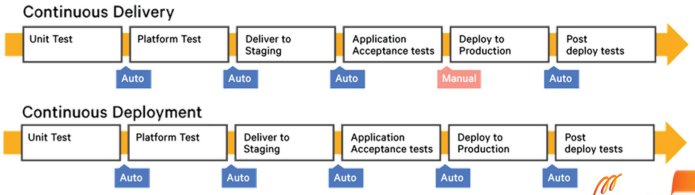
Jatkuva integrointi

- Käynnistys jokaisen commitin jälkeen
  - kun git laukaisee
- tai esim. kerran päivässä (nightly builds)
  - kun Jenkins pollaa
- Tee koonnista itsetestaava
  - suorituta myös testit automaattisesti
- Testaa tuotantopalvelimen kloonissa
  - näytteillepano, koekäyttö (staging)
- Tee raportit julkisiksi, anna nopea palaute
  - työtilassa tulosnäyttö: vihreä/punainen
  - jokainen näkee viime koonnin tulokset
  - nopea havainnointi, nopea korjaus

Voisi olla näytöllä tiimien työtilan seinällä

Jatkuva julkaisu

- Automatisoi päivitysten käyttöönotto
  - jatkuva toimitus (Continuous Delivery)
  - jatkuva tuotantoonvienti (Continuous Deployment)
- Jatkuva julkaisuputki, 24/7 = **DevOps** **GitOps**



Jenkins CI

- Jatkuvan integroinnin työkalu, open source
- Hallitsee myös jatkuvan tuotantoonviennin
- Osa käyttää mm. ant- tai maven-koontiskriptejä
  - build.xml tai pom.xml
- Cron-ajastettujen tehtävien suoritus ja tarkkailu
  - UNIX: cron, crontab
- Taipuu moniin moniin tehtäviin, laajennettavissa
  - yli 600 lisäosaa (plugin)
- Muita vastaavia työkaluja
  - GitLab CI, Travis CI, CruiseControl, ...

https://jenkins.io

Docker

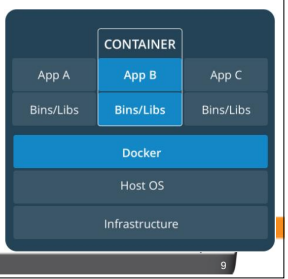
- OTP: Ajamme Jenkinsiä Docker-alustalla
  - Ryhmän omalla educloud-koneella
  - Ryhmällä itsellään admin-oikeudet
- Keino paketoida ja suorittaa sovelluksia
- Hyödyntää Container-virtualisointitekniikkaa
- Käytetään tavallisesti pilvipalveluissa

- PaaS, Platform as a Service
  - esim. kehittäjän itse pilveen käynnistämä virtuaalikone, jossa valmiiksi asennettuna ja konfiguroituna kaikki tarvittavat komponentit sekä integraatio kehitysympäristöön
  - voi tarjota työkalut, joilla koodi menee suoraan tuotantoon
- ~ CaaS, Container as a Service

https://docker.com/

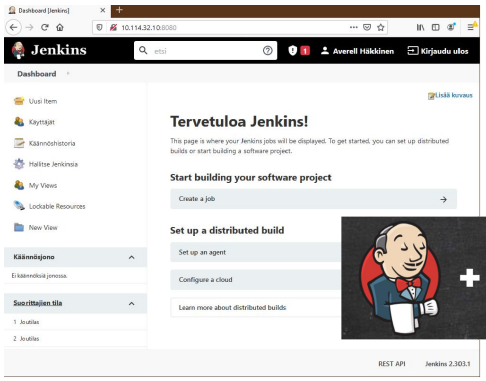
Container ~ Kontti

- Kontti sisältää kootusti kaiken tarvittavan KJ:n ydintä lukuun ottamatta
  - Ydin: keskeytyskäsittely, vuorottaminen, muistinhallinta, laitetason siirräntä
  - mukana tyypillisesti riisuttu versio KJ:n muista osista
  - tarvittavat ohjelmakirjastot ja alustapalvelut
  - itse ajettava sovellus
- Kontit ladataan suoritettavaksi Docker-alustalle
- Yhdessä virtuaalikoneessa voi ajaa useita rinnakkaisia sovelluksia
  - pysyvät täysin eristettyinä omissa konteissaan
  - jos yksi kontti sössii, ei vaikuta muihin
- Tehostaa DevOps-sykliä
  - standardoitu kontti nopea jakaa tuotantoon, siinä on mukana kaikki tarvittava
  - alustalle riittää, että osaa pyörittää Dockeria
- OTP: Kontissa pyöritetään Jenkinsiä

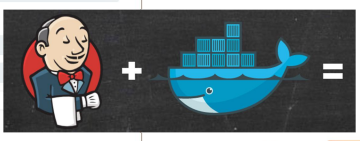


Jenkins

- Asennus- ja konfigurointitehtävä löytyy OMA:sta



Muuta käyttämään englanninkielistä käyttöliittymää



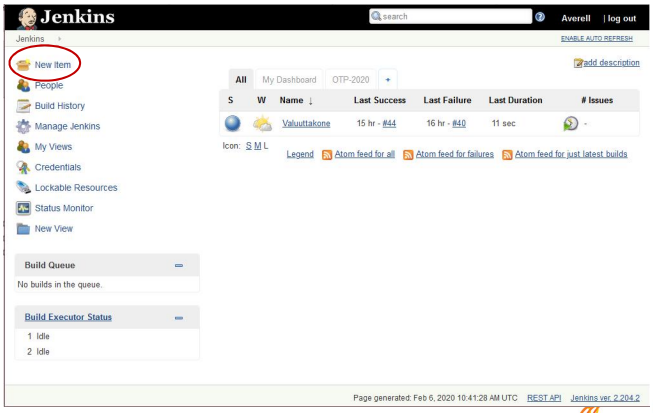
"Mr Jenkins, I need to build my application using Java 11."

- pom.xml:
  - `<maven.compiler.release>11</maven.compiler.release>`
- Kun Jenkinsiä ajetaan Docker-kontissa, niin oletus on Java 11
  - versiosta Jenkins 2.303.1 alkaen
- Jenkins itsessään käyttää siis Java/JDK 11:ta
- Projektin sovellustakaan varten ei tarvitse asentaa erikseen

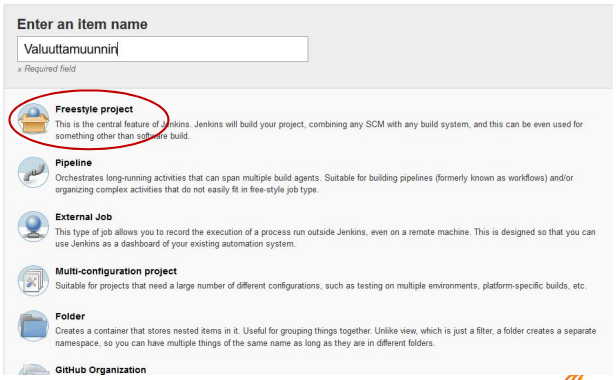
"Mr Jenkins, I want to use Maven."

- Valitse Manage Jenkins | Global Tool Configuration | Maven
  - Napsauta "Add Maven"
  - Valitse versionumero, esim. 3.8.2
  - Halutessasi voit täyttää myös Name-kentän, esim. "Maven 3.8.2"

Jenkins-työn luominen



- Jenkins-töitä voi luoda eri tarkoituksiin
- Usein vapaamuotoinen työ, joka määritetään itse

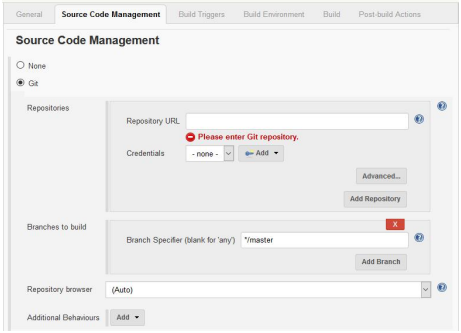


Jenkins-työn määrittely

1. Haku versionhallinnasta
   - Missä projektin lähdekieliset koodit ovat
2. Varsinainen koonti
   - Mikä koontityökalu (ant, maven, gradle,...)
   - Mistä koontiskripti löytyy
   - Mikä laukaisee koonnin
   - Mikä vaihe suoritetaan
     - validate, compile, test, package, verify,install, deploy
3. Koonnin jälkeiset toimenpiteet
   - Aja staattisen analyysin työkalut (esim. testikattavuus)
   - Julkaise raportit

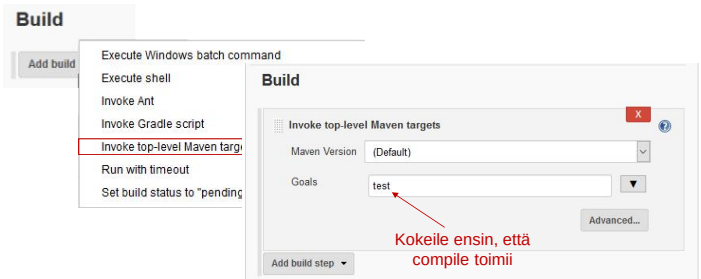
1. Haku versionhallinnasta

- Credentials: GitLab-tunnus, jolla vähintään Reporter-oikeudet
  - voitte tehdä Jenkinsiä varten oman tunnuksen
- Koodit haetaan yleensä develop-haarasta



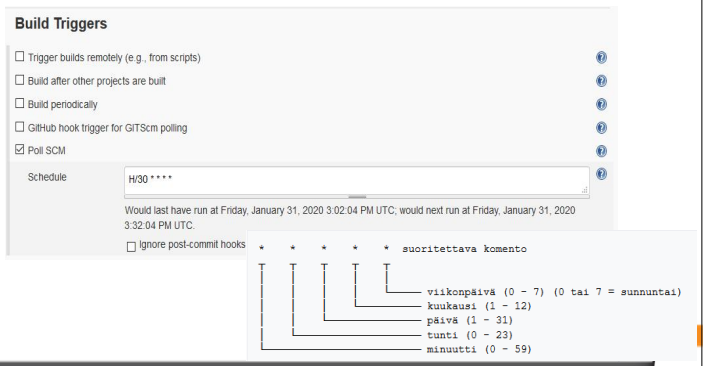
2. Varsinainen koonti

- Työn määrittelyyn voi liittää askelia, jotka tehdään koonnin aikana
- Koonnin yhteydessä voi suorituttaa myös skriptejä ja komentorivi-komentoja



Koonnin ajastaminen

- Poll SCM ajastaa lähdekoodien hakemisen versionhallinnasta
- Aikataulu annetaan kuten UNIXin crontabissa

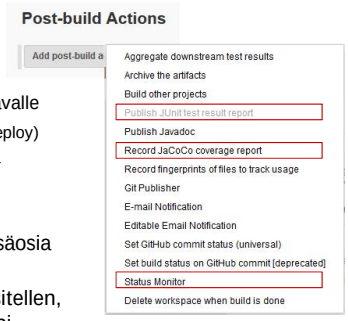


Webhook

- GitLabissa ja GitHubissa voi määritellä Webhookin, jonka avulla ne voivat kertoa Jenkinsille, että jotain on muuttunut
- Jenkins
  - valitse "`Trigger builds remotely (e.g., from scripts)`"
  - anna kohtaan "`Authentication token`" joku merkkijono
- tämän jälkeen voit käynnistää/laukaista (trigger) työn myös käsin urlista
  `/build?token=TOKEN_NAME`
- Käy asettamassa URL ja token gitissä, ja git voi käynnistää työn
  - GitLab
    - Settings | Webhooks
- GitLab.com ja educloud
  - Palomuuriasetukset eivät mahdollista tätä
- Onnistuu, kun käytössä sisäverkossa pyörivä Metropolian Gitlab

3. Koonnin jälkeiset toimet

- Esimerkiksi
  - testiraporttien julkaisu
  - staattinen analyysi
  - kuittaus koonnista esim. IRC-kanavalle
- Myös sovelluksen käyttöönoton (deploy) testipalvelimelle voi automatisoida
  - manuaaliset hyväksymistestit
- Viereinen lista sen mukaan mitä lisäosia (plugin) on itse asentanut
- Näitä kannattaa ottaa mukaan yksitellen, ja tsekata aina, että työ menee läpi



Esimerkki: Raportointi