Avtomatizacija gradnje aplikacij

Matjaž B. Jurič

Gradniki okolja CI

- Repozitorij izvorne kode
- Repozitorij programskih artefaktov
- Repozitorij vsebnikov Docker
- Orodja za avtomatizacijo buildov
- Orodja za CI
- Orodja za upravljanje projektov in zahtevkov
- Orodja za analizo kode
- Orodja za testiranje
- Orodja za pripravo in namestitev dokumentacije

Repozitorij izvorne kode

- Sistemi z upravljanje z izvorno kodo
- Upravljanje z verzijami
- Distribuirani ali centralizirani model
- Upravljanje s sočasnostjo sprememb
- Najpogosteje uporabljeni:
  - GIT
  - Concurent Versions System (CVS)
  - Subversion (SVN)
  - Microsoft Team Foundation Server
  - IBM ClearCase, Synergy
  - Mercurial

Repozitorij programskih artefaktov

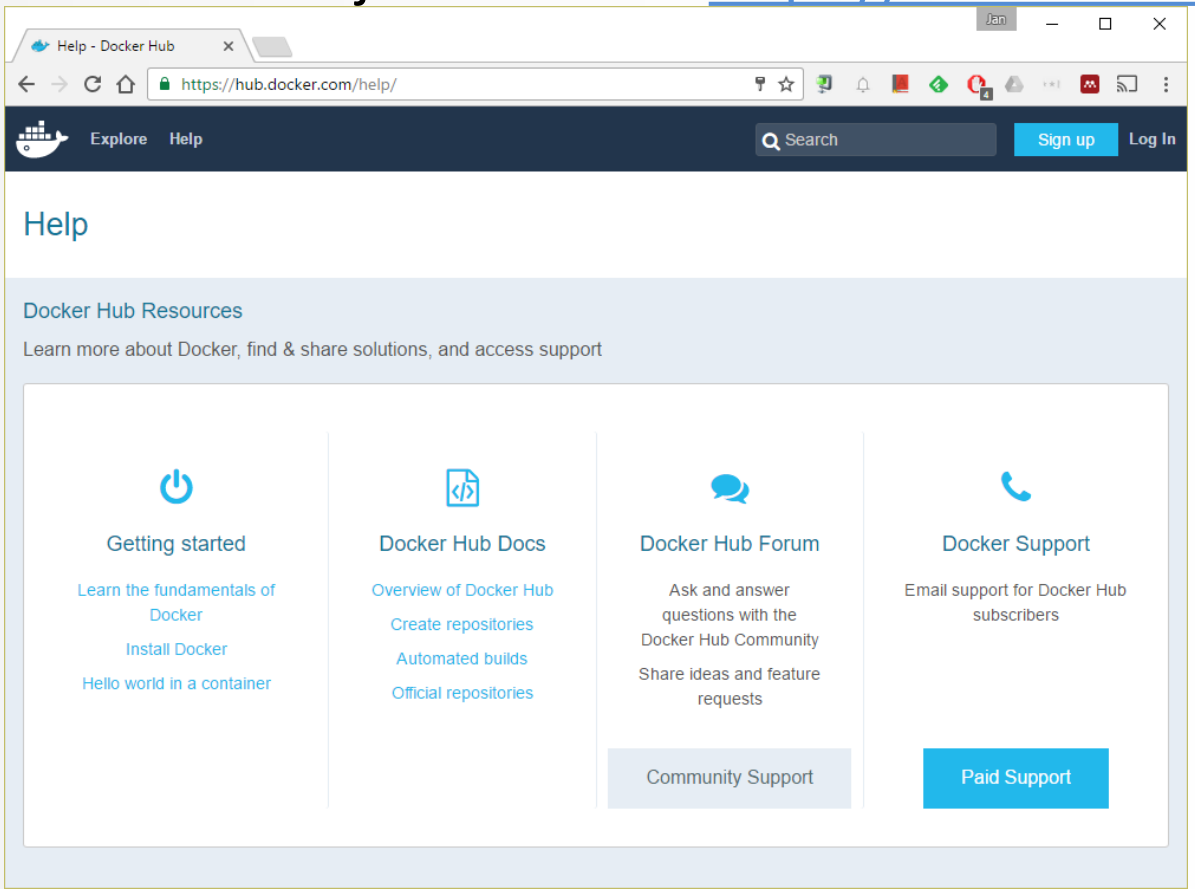
- Hrani in streže pakete programske kode, knjižnic, dokumentacije in drugo
- Javni in zasebni
- Upravljanje odvisnosti
- Upravljanje verzij
- Najpogosteje uporabljeni v kombinaciji z Java/Maven:
  - Sonatype's Nexus
  - JFrog's Artifactory
  - Apache Archiva

Repozitorij slik Docker – Docker Hub

- Oblačni repozitorij za shranjevanje in deljenje Docker slik
- Omogoča upravljanje z verzijami slik in kolaboracije
- Gosti mnogo uradnih certificiranih repozitorijev organizacij
  - npr. Canonical, Oracle, Red Hat
- Razvijalcem omogoča, da svoje slike gradijo na množici obstoječih slik, pripravljenih za različne namene
  - npr. postgres, java + maven, java + postgres
- Brezplačen račun za gostovanje javnih repozitorijev

Repozitorij slik Docker – Docker Hub

- Slike v repozitorij nalagamo in do njih dostopamo z uporabo ukazov
  - *docker run* - prenese sliko iz repozitorija in jo zažene
  - *docker login* - prijava v repozitorij
  - *docker push* - naloži sliko iz računalnika v repozitorij
- Spletna konzola se nahaja na naslovu https://hub.docker.com



Orodja za avtomatizacijo buildov

- Prevajanje programske kode
- Upravljanje z odvisnostmi
- Pakiranje programske kode
- Izvajanje testov
- Namestitev paketov
- Priprava dokumentacije
- Najpogosteje uporabljeni:
  - Apache Maven
  - Gradle
  - Apache Ant
  - Make
  - Pants
  - SBT

Apache Maven

- Orodje na katerem temelji avtomatizirano grajenje (build) projektov

- Prednosti uporabe Maven
  - Izboljšana vidnost in transparentnost razvojnega procesa
  - Apliciranje splošno sprejetih dobrih praks (npr. verzioniranje)
  - Standardizacija (npr. enotna struktura projektov)
  - Izboljšano upravljanje z odvisnostmi (binarne odvisnosti)
  - Lažja izmenjava kreiranih artefaktov (JAR, WAR, EAR, RAR)
  - Ponovna uporaba
  - Samodejno generiranje spletne strani in dokumentacije
  - Zmanjšan čas vpeljave novih razvijalcev

Apache Maven

- Srce vsakega Maven projekta je datoteka pom.xml
- POM (Project Object Model)
- pom.xml definira:
  - Naziv projekta
  - Verzijo
  - Odvisnosti
  - Cilje (goals)
  - Vtičnike (plugins)
  - Razne metapodatke

- V pom.xml so obvezne samo 4 vrstice (groupId, artifactId, version in modelVersion)
- Pri definiranju pom datotek lahko uporabimo koncept dedovanja
- Vsak pom.xml deduje od t.i. super POM-a

Apache Maven

- Primer preproste pom.xml datoteke:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0">
      <modelVersion>4.0.0</modelVersion>
      <groupId>com.demo</groupId>
      <artifactId>projektA</artifactId>
      <packaging>war</packaging>
      <version>0.0.1-SNAPSHOT</version>

      <dependencies>
            <dependency>
                  <groupId>junit</groupId>
                  <artifactId>junit </artifactId>
                  <version>4.8.1</version>
                  <scope>test</scope>
            </dependency>
      </dependencies>
</project>
```

Privzeta vrednost je jar, v tem primeru lahko izpustimo. Druge opcije: war, ejb, rar, ear, pom in drugi (custom).

Verzija

Seznam odvisnosti

Naziv artefakta, ki se kreira ob build-u:

projektA-0.0.1-SNAPSHOT.war

Orodja za CI

- CI orodja in build strežniki
  - Izvaja pogosto dnevno integracijo paketov programske kode
  - Avtomatizirani buildi
  - Avtomatizirani testi enot in integracij
  - Avtomatizirana analiza kode
  - Namestitev v repozitorije programskih artefaktov ter v integracijska in strežniška okolja
- Podpora upravljanja odvisnosti v izvorni kodi
- Inkrementalno procesiranje buildov
- Poročanje o buildih, statusih, testih ter namestitvah
- Izgradnja, namestitev in streženje dokumentacije

Orodja za avtomatizacijo CI cikla

- Podpora razširitvam v obliki vtičnikov
- Podpora združevanju več CI strežnikov v gruče
- Korelacija projektnih zahtevkov z buildi
- Najpogosteje uporabljena orodja:
  - Hudson in Jenkins
  - Travis CI
  - Microsoft Team Foundation Server
  - Atlassian Bamboo
  - TeamCity
  - Apache Continuum
  - ElectricFlow (Cloud)
  - CruiseControl
  - AnthillPro

Jenkins

Visual Studio Team Foundation Server

Bamboo

TeamCity by JetBrains

Electric Cloud

cruisecontrol

Orodja za avtomatizacijo CI cikla

- Poskrbijo, da se ob objavi sprememb v sistemu za verzioniranje izvorne kode, sproži nov cikel zvezne integracije, ki ga sestavljajo:
  - Prevajanje in pakiranje (build)
  - Izvedba testov enot
  - Nameščanje artefaktov (npr. objava JAR-ov v Nexus)
  - Gradnja vsebnikov Docker
  - Objava vsebnikov v repozitorij (npr. DockerHub)
  - Nameščanje vsebnika v testno okolje

- Cikel se proži samo ob spremembah na vnaprej določenih vejah, npr. na veji *master*

Travis CI

- Orodje za avtomatizacijo integracijskega cikla
- Integrirano s sistemom za verzioniranje izvorne kode GitHub
- Cikel zvezne integracije opišemo v datoteki *.travis.yml*, ki jo skupaj z izborno kodo objavimo v repozitoriju
- Cikel vsebuje dva glavna koraka:
  - **install**: namestitev potrebnih odvisnosti
  - **script**: koraki, ki opisujejo izvedbo glavnine integracijskega cikla
- Ukaze je mogoče izvajati tudi pred in po glavnih korakih z uporabo *before_install*, *before_script*, *after_script*, *after_success*, *after_failure*

Orodja za upravljanje projektov in zahtevkov

- Orodja za upravljanje s projekti (planiranje, organiziranje, izvajanje)
- Orodja za sledenje zahtevkom in napakam (ITS - Issue Tracking System, Request Management, Bug Tracking System)
- Centraliziran pregled zahtevkov PO
- Upravljanje z izdajami
- Integracija z repozitorijem izvorne kode in CI orodji
- Integracija dokumentacije
- Poročanje
- Podpora prilagoditvam delovnih tokov

Orodja za upravljanje projektov in zahtevkov

- Najpogosteje uporabljena orodja:
  - Atlassian JIRA
  - GitHub
  - GitLab
  - IBM Rational Team Concert
  - Google Code Hosting
  - HP Quality Center
  - Youtrack
  - Bugzilla
  - Trac
  - Mantis BT

Merjenje pokritosti (kakovosti) programske kode

- Pomemben vidik testiranja programske kode.
  - Enostavna ocenitev, kako dobro je programska koda **pokrita s testi**.
  - Razvijalce **spodbuja** k pisanju testov.
- Omogočajo **namenska orodja** (Code Coverage tools).
- Različne **metrike** merjenja:
  - Pokritost po **metodah**
  - Pokritost po posameznih **vejah** izvajanja
  - Pokritost po **vrsticah**
  - Itd.
- Dobra praksa je, da vodja projekta določi **minimalno pokritost** za vsak projekt (npr. 80 %).
- Zahteve glede minimalne pokritosti programske kode so pogosto del nefunkcionalnih zahtev projekta (**SLA**).

Aktualna orodja za merjenje pokritosti kode

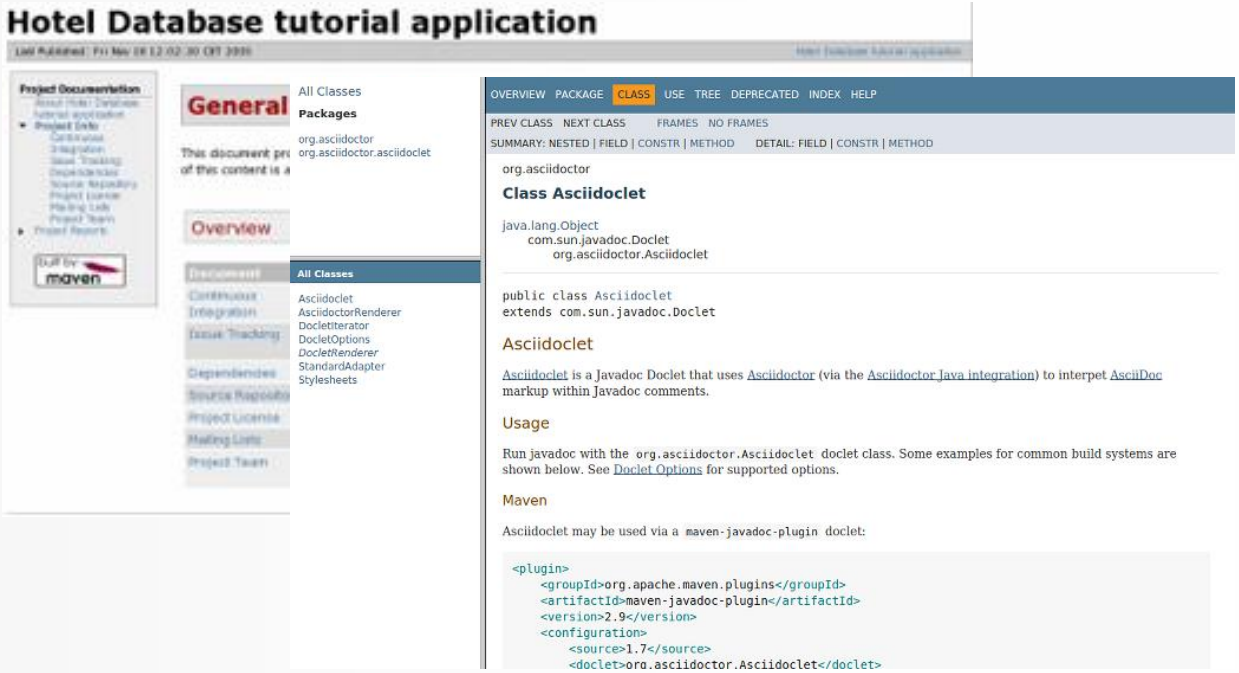
- **SonarQube**:
  - C#
  - Java
  - JavaScript
  - PL/SQL
  - Web (HTML, JSP/JSF)
- **Atlassian Clover**
  - Java
  - Groovy
- **Cobertura**
  - Java
  - Groovy
- **JaCoCo**
  - Java
  - Groovy
- **Emma**
  - Java

Orodja za testiranje

- CI orodja izvajajo testiranja enot in integracijsko testiranje ob buildu komponent
- Integracijsko testiranje se izvaja v integracijskem okolju
- Testiranje sprejemljivosti se izvaja na testnem strežniku
- Testiranje zagotovi konsistentnost delovanja programskih komponent
- Najpogosteje uporabljena orodja:
  - JUnit
  - SoapUI, LoadUI
  - Apache JMeter
  - Selenium
  - IBM Rational Functional Tester

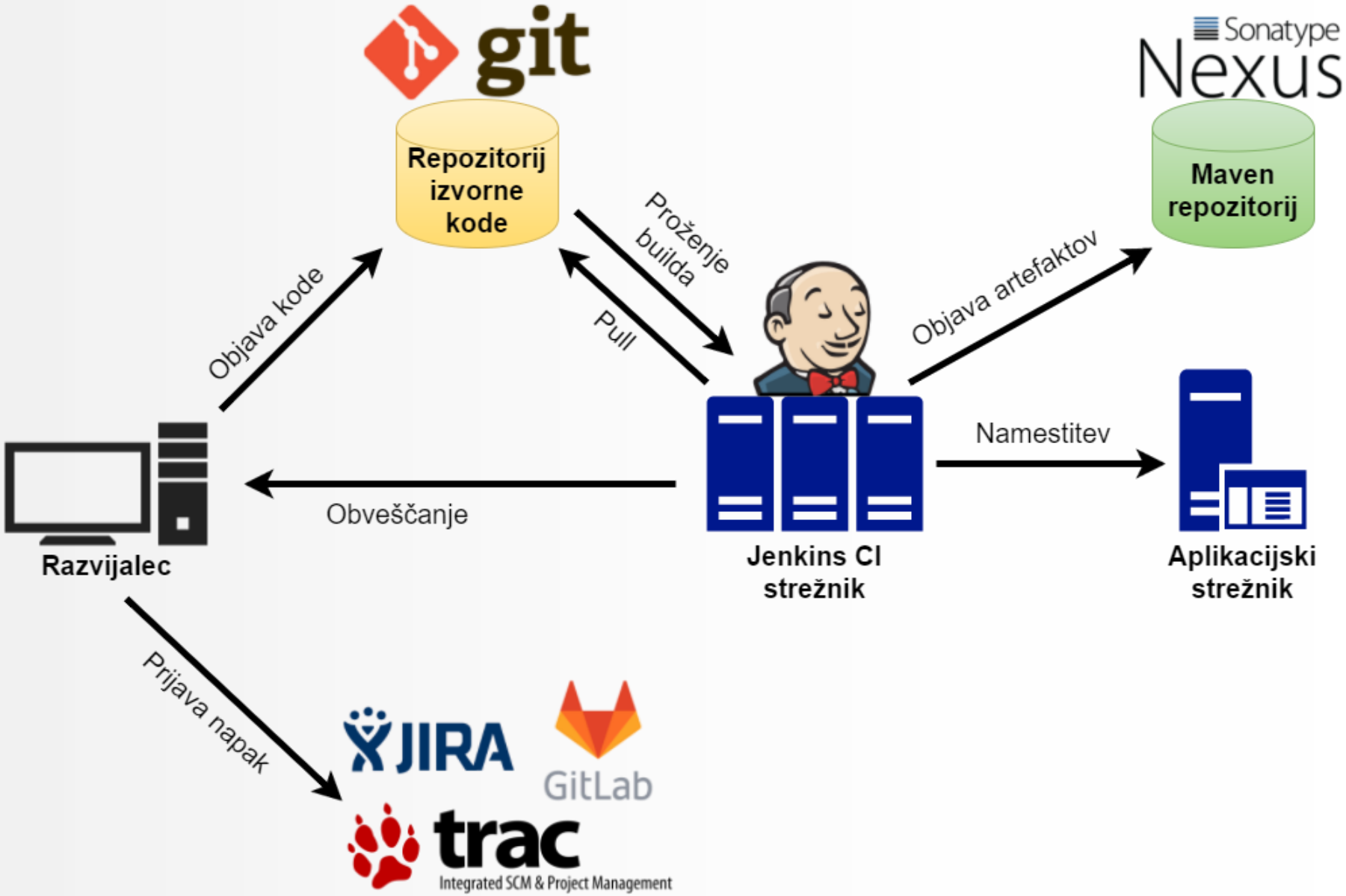
Orodja za pripravo in namestitev dokumentacije

- Za dokumentiranje PO v Javi uporabimo Javadoc
- Dokumentiranje komponent odvisno od uporabe build orodja
- Build strežnik avtomatizira proces izgradnje paketov dokumentacije, namestitev v repozitorij ter namestitev v dokumentarni sistem
- Najpogosteje uporabljena orodja:
  - Javadoc
  - Doxygen
  - XSDDoc
  - SandCastle
  - Maven Site



Primer cikla CI

- Tipičen scenarij za razvoj v Java EE:



AVTOMATIZACIJA GRADNJE APLIKACIJ

Zvezna integracija *(Continuous integration)*

- Zvezna integracija je razvijalska praksa, pri kateri se vse objavljene spremembe v izvorni kodi avtomatično prevedejo v novo verzijo namestitvene enote (npr. vsebnika Docker)

- Sprotno prevajanje in objavljanje novih verzij omogoča, da je vsem udeležencem razvoja vedno na volja zadnja delujoča verzija aplikacije

- Vključuje avtomatizacijo izvajanja testov enot *(unit testing)* kot del integracijskega cikla

- Poskrbi za ustrezno pripravo in namestitev izdelka v testno okolje

- Glavne komponente sistema za zvezno integracijo so
  - Orodja za avtomatizacijo buildov (Maven, Gradle, …)
  - Orodja za avtomatizacijo integracijskega cikla (Travis CI, Jenkins, …)
  - Repozitoriji artefaktov (Nexus, …)
  - Repozitoriji vsebnikov (DockerHub, Nexus, …)

Orodja za avtomatizacijo integracijskega cikla

- Poskrbijo, da se ob objavi sprememb v sistemu za verzioniranje izvorne kode, sproži nov cikel zvezne integracije, ki ga sestavljajo:
  - Prevajanje in pakiranje (build)
  - Izvedba testov enot
  - Nameščanje artefaktov (npr. objava JAR-ov v Nexus)
  - Gradnja vsebnikov Docker
  - Objava vsebnikov v repozitorij (npr. DockerHub)
  - Nameščanje vsebnika v testno okolje

- Cikel se proži samo ob spremembah na vnaprej določenih vejah, npr. na veji *master*

Travis CI

- Orodje za avtomatizacijo integracijskega cikla
- Integrirano s sistemom za verzioniranje izvorne kode GitHub
- Cikel zvezne integracije opišemo v datoteki *.travis.yml*, ki jo skupaj z izborno kodo objavimo v repozitoriju
- Cikel vsebuje dva glavna koraka:
  - **install**: namestitev potrebnih odvisnosti
  - **script**: koraki, ki opisujejo izvedbo glavnine integracijskega cikla
- Ukaze je mogoče izvajati tudi pred in po glavnih korakih z uporabo *before_install*, *before_script*, *after_script*, *after_success*, *after_failure*

Travis CI

- Primer datoteke *.travis.yml*, ki izvede *Maven build*, dobljen arhiv *JAR* zapakira v vsebnik *Docker* in ga objavi v repozitorij DockerHub

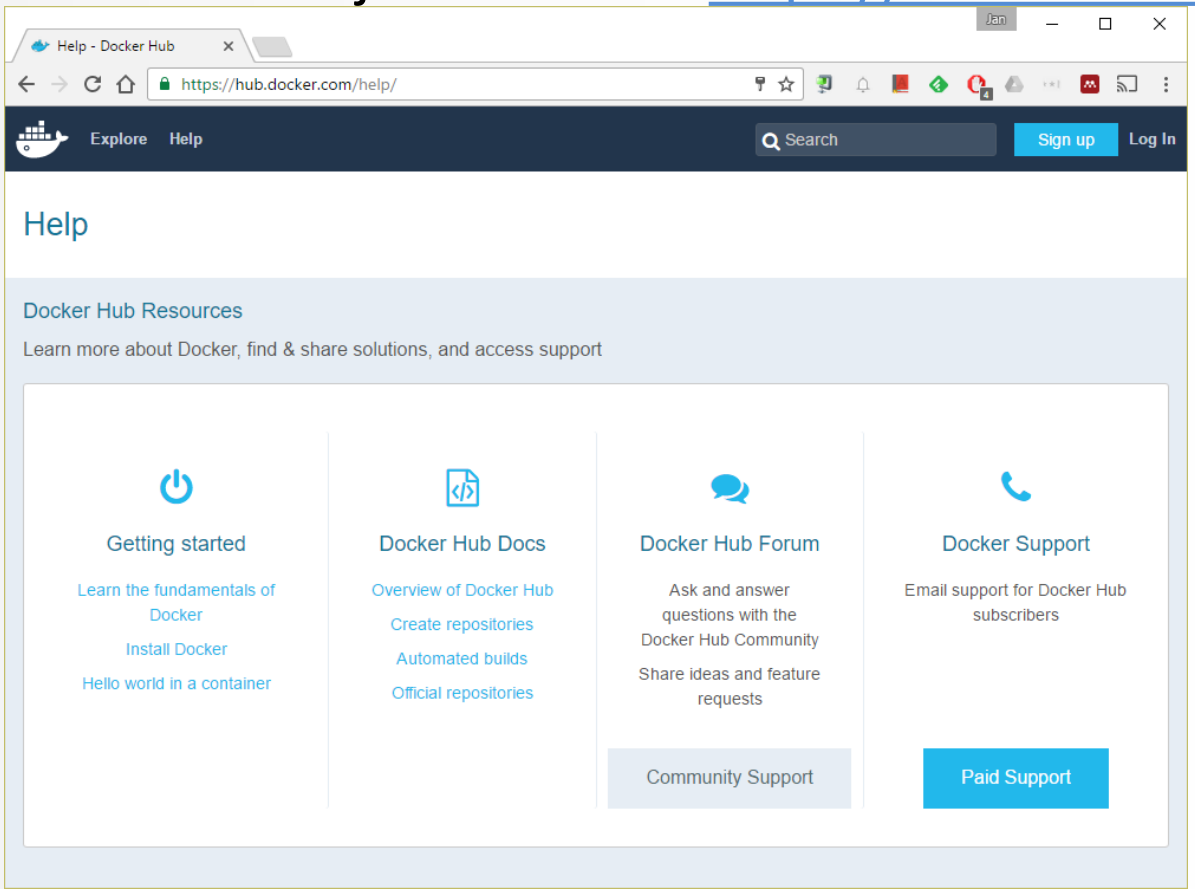
```
sudo: required

services:
  - docker

language: java
dist: trusty

jdk:
- openjdk8

cache:
  directories:
  - $HOME/.m2

script:
  - mvn clean package -U
  - docker build -t moj-repozitorij/rso-customers .

after_success:
  - docker login -u="$DOCKER_USERNAME" -p="$DOCKER_PASSWORD"
  - docker push moj-repozitorij/rso-customers
```

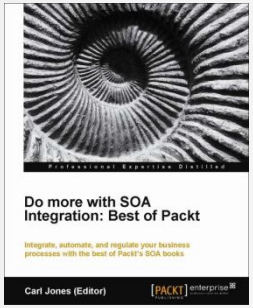
Docker Hub

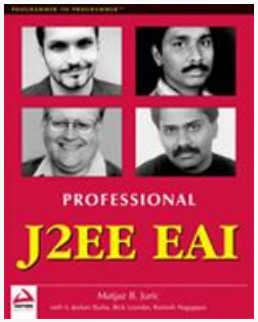
- Oblačni repozitorij za shranjevanje in deljenje Docker slik
- Omogoča upravljanje z verzijami slik in kolaboracije
- Gosti mnogo uradnih certificiranih repozitorijev organizacij
  - npr. Canonical, Oracle, Red Hat
- Razvijalcem omogoča, da svoje slike gradijo na množici obstoječih slik, pripravljenih za različne namene
  - npr. postgres, java + maven, java + postgres
- Brezplačen račun za gostovanje javnih repozitorijev

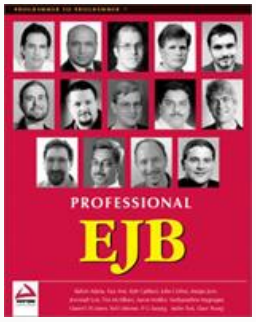
Docker Hub

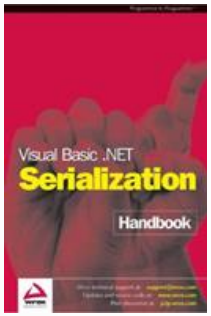
- Slike v repozitorij nalagamo in do njih dostopamo z uporabo ukazov
  - *docker run* - prenese sliko iz repozitorija in jo zažene
  - *docker login* - prijava v repozitorij
  - *docker push* - naloži sliko iz računalnika v repozitorij
- Spletna konzola se nahaja na naslovu https://hub.docker.com

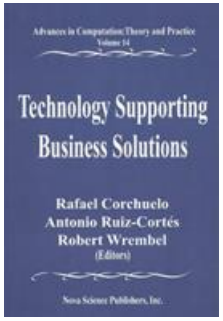


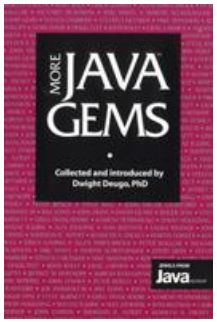












HVALA!

Java Champions

ORACLE ACE Director

IBM CHAMPION

ORACLE Certified Expert

Certified for IBM SOA technology

Certified for IBM CLOUD COMPUTING technology