



Szoftvertchnológia

Continuous integration & delivery

Cserép Máté

ELTE Informatikai Kar

2020.



Continuous integration & delivery

Folyamatos integráció

- ▶ A *folytonos integráció* (*continuous integration, CI*) egy olyan gyakorlati módszer, amely lehetővé teszi a programkódok ellenőrzésének és tesztelésének felgyorsítását
 - ▶ célja a lehetséges hibák, integrációs problémák azonnali, automatizált kiszűrése, visszajelzés a fejlesztőnek
 - ▶ a programkódok verziókezelő rendszer segítségével egy központi tárhelyre kerülnek, naponta többször
 - ▶ a tárhely tartalma minden módosítást követően automatikusan fordításra kerül (*build automation*), a fordítással pedig a lekódolt tesztek is végrehajtnak
 - ▶ az így ellenőrzött kódot további tesztelés követheti



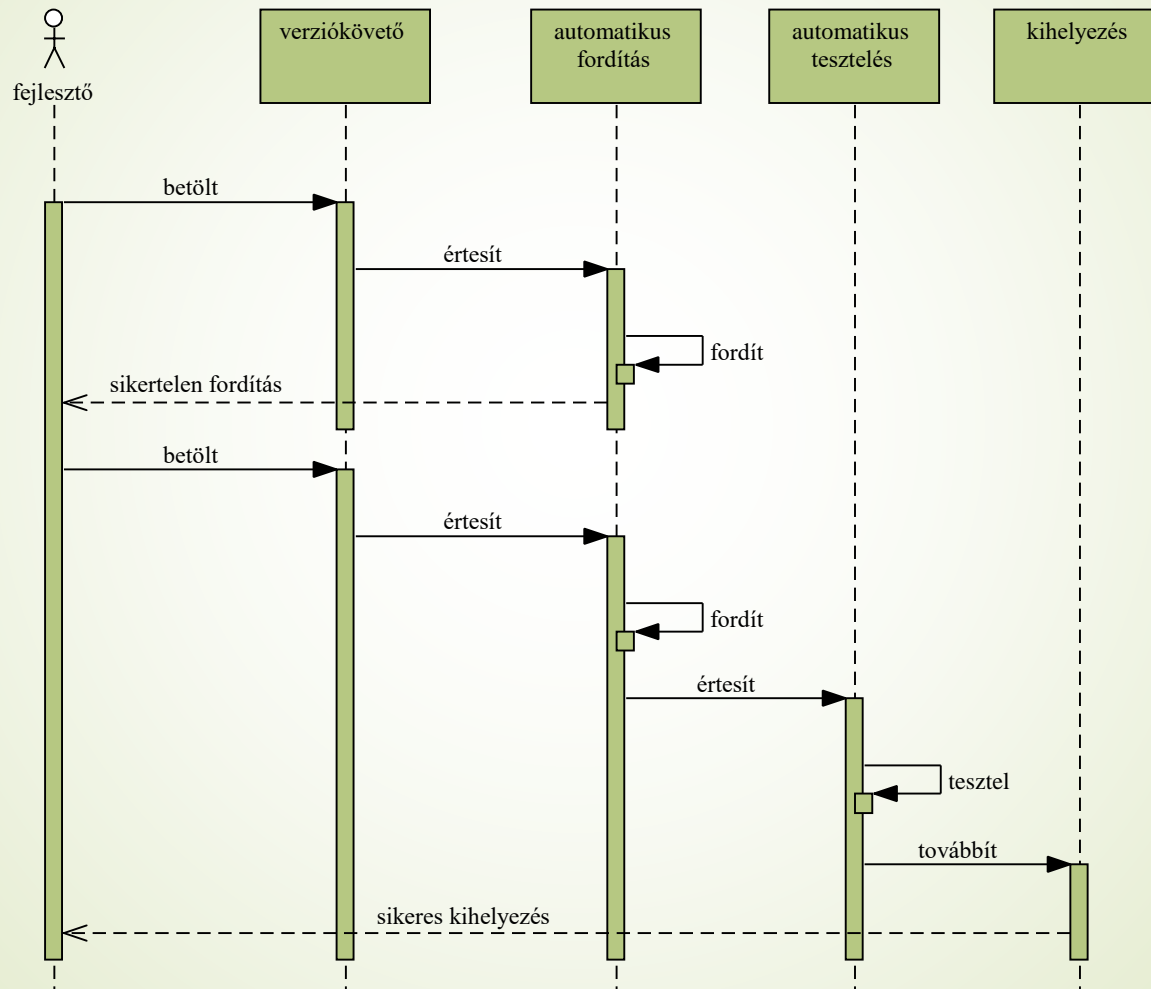
Continuous integration & delivery

Folyamatos teljesítés

- ▶ Az agilis szoftverfejlesztés (*agile software development*) célja a gyors alkalmazásfejlesztés megvalósítása, inkrementális alapon
 - ▶ a szoftver folyamatos fejlesztés és kiadás alatt áll (*continuous delivery*), a sebesség állandó, a változtatások minden lépésben beépíthetőek (*welcome changes*)
 - ▶ a működő szoftver az előrehaladás mérőeszköze, előtérben az egyszerűség, ugyanakkor folyamatos odafigyelés a megfelelő tervezésre, optimalizációra
 - ▶ a fejlesztést általában önszervező, kis csapatok végzik, megosztott felelősséggel, folytonos interakcióval, gyors visszajelzésekkel
 - ▶ a folyamatos kiadások automatizálhatók, ekkor *continuous deployment*-ről beszélünk

Continuous integration & delivery

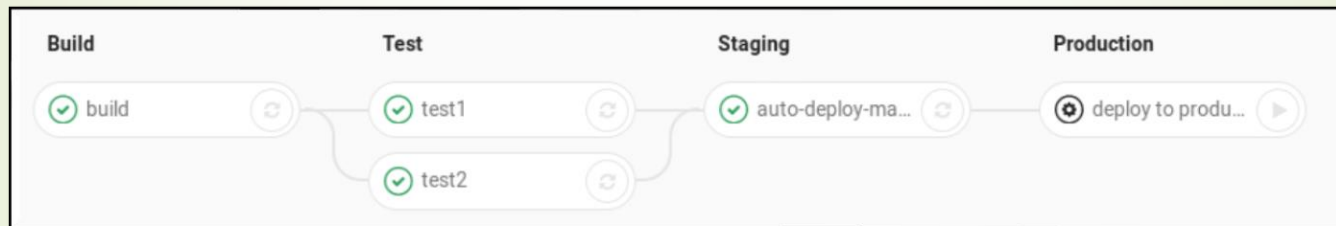
Folyamatos integráció és teljesítés



Continuous integration & delivery

Feladatok

- A folyamatos integráció és teljesítés lépéseit egymásra épülő feladatok (*jobs*) láncolataként (*pipelines*) definiálhatjuk

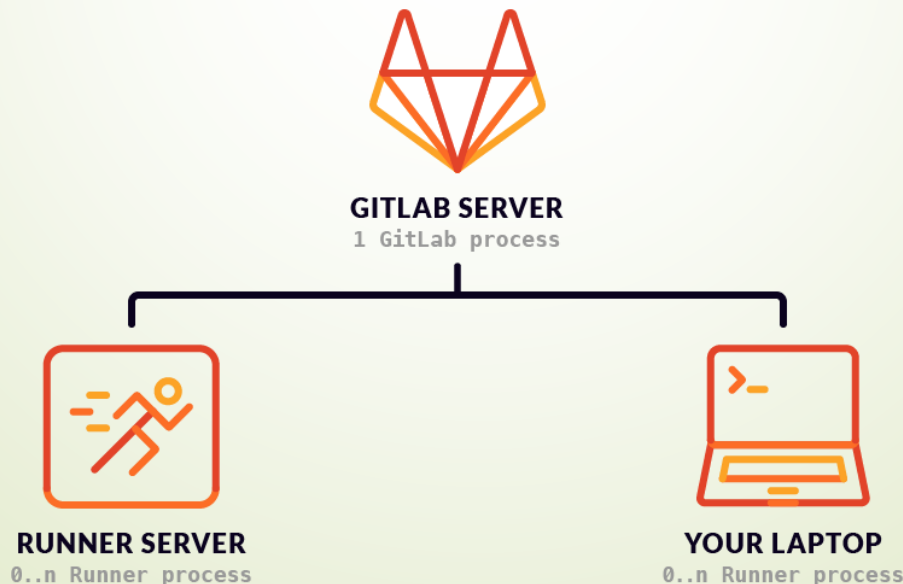


Status	Pipeline	Commit	Stages	Duration	Time Ago	Actions
running	#6453892 by [user] latest	3df39dd6 add data durability to Alex	✓ 🌙			▶ ⏴ ⏵ ⏶ ⏷ ⏸
passed	#6453883 by [user] latest	d0f228af Filled in content	✓ ✓	00:08:15	4 minutes ago	▶ ⏴ ⏵ ⏶ ⏷ ⏸
passed	#6453817 by [user] latest	06a01e18 De Wet Geo Expert	✓ ✓	00:08:50	8 minutes ago	▶ ⏴ ⏵ ⏶ ⏷ ⏸
passed	#6453004 by [user] latest	0c7954e5 Update index.html.md	✓ ✓	00:08:33	59 minutes ago	▶ ⏴ ⏵ ⏶ ⏷ ⏸

Continuous integration & delivery

GitLab Runners

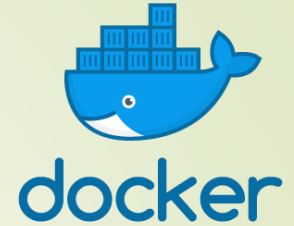
- ▶ A GitLab rendelkezik integrált, saját megoldással a folyamatos integráció és teljesítés támogatására
 - ▶ a feladatokat (*jobs*) a GitLab szervertől független ún. *GitLab Runner* példányok hajtják végre
 - ▶ Shell, SSH, VirtualBox, Docker, Kubernetes, stb.
 - ▶ a *runnerek* egyedileg konfigurálhatóak, lehetnek megosztottak vagy projekthez rendelvek



Continuous integration & delivery

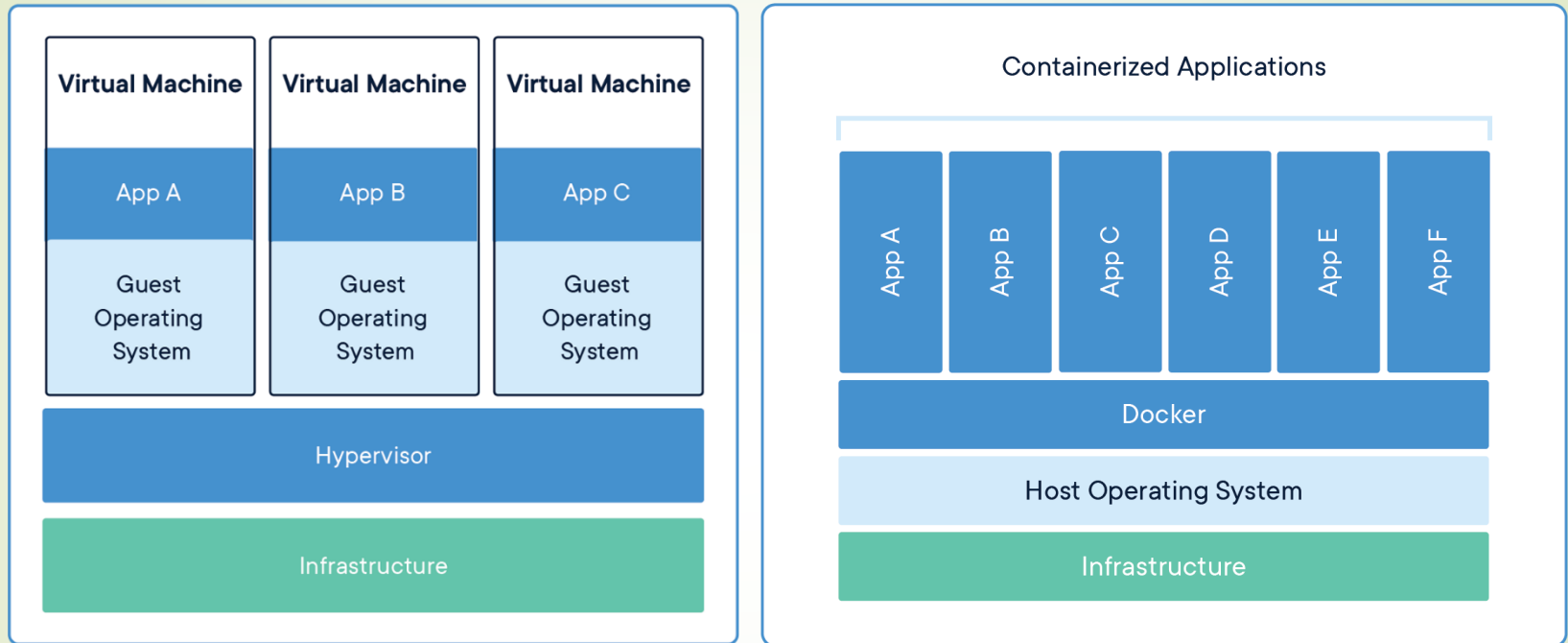
Docker

- ▶ A folyamatos integrációt egy izolált, reprodukálható környezetben érdemes végezni
- ▶ A Docker napjainkban a legelterjedtebb *container framework*
 - ▶ a *container* hasonlít a virtuális gépekhez (VM) olyan tekintetben, hogy egy teljesen elkülönített, virtualizált környezett biztosít, amelynek a gazdaszámítógép szolgáltat erőforrásokat
 - ▶ a fő különbség a *containerek* és a virtuális gépek között, hogy minden *container* osztozik a gazda kerneljén a többi *containerrel*, a virtualizált hardver és az OS nem része, csupán az alkalmazásunkhoz kötődő könyvtárak, binárisok és a felhasználói terület
 - ▶ a *containerek* ezáltal nagyságrendekkel kisebb *overheaddel* bírnak a VM-ekhez képest, így könnyebb súlyú megoldást nyújtanak a virtualizációra



Continuous integration & delivery

Virtuális gépek és *container*erek



Forrás: docker.com

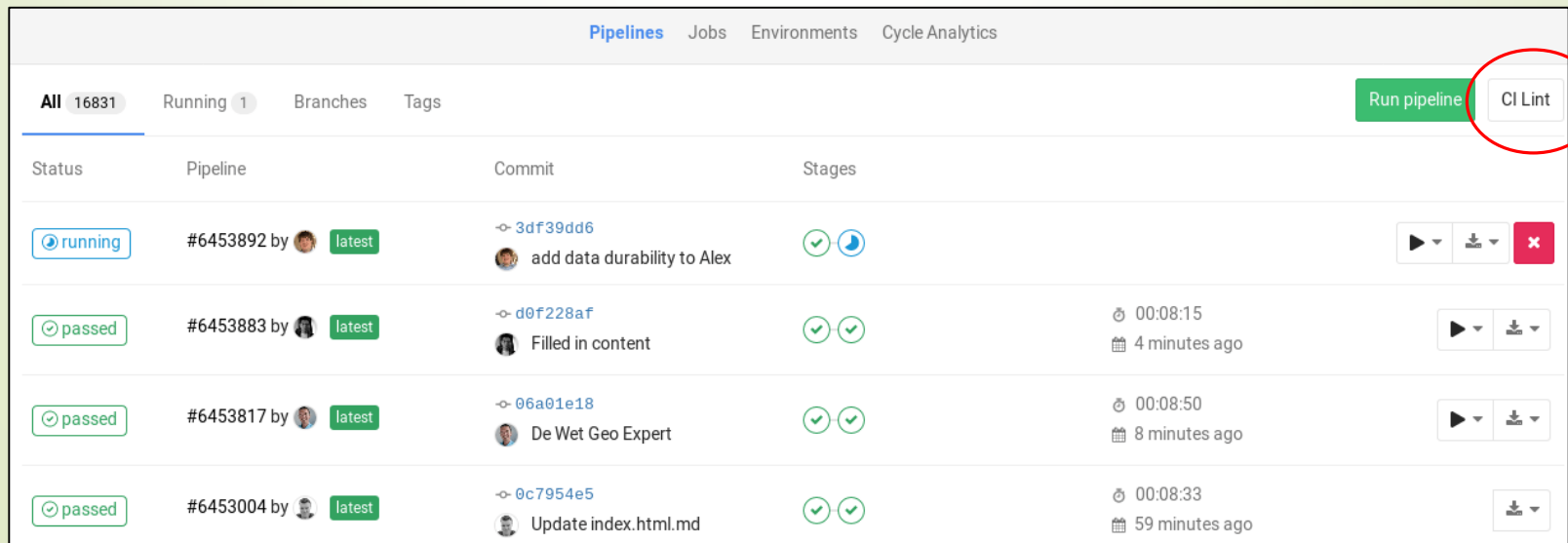
- a *container framework*ök lehetőséget adnak hordozható alkalmazások létrehozására és menedzselésére
- az alkalmazások modularizálhatóak és skálázhatóak, a komponensek külön containerben futhatnak



Continuous integration & delivery

GitLab CI/CD

- ▶ A folyamatos integráció konfigurációját a `.gitlab-ci.yml` fájl tartalmazza, YAML formátumban
 - ▶ a YAML (*YAML Ain't Markup Language*) egy emberi szemmel könnye(bbe)n olvasható strukturált leíró nyelv
 - ▶ <https://yaml.org/spec/1.2/spec.html>
- ▶ A GitLab webes felületén elérhető egy CI Lint funkció a formátum validálására beküldés előtt



The screenshot shows the GitLab Pipelines interface. At the top, there are tabs for 'Pipelines', 'Jobs', 'Environments', and 'Cycle Analytics'. Below the tabs, there are filters for 'All 16831', 'Running 1', 'Branches', and 'Tags'. On the right side, there are two buttons: 'Run pipeline' and 'CI Lint'. The 'CI Lint' button is circled in red. Below the buttons, there is a table with columns for 'Status', 'Pipeline', 'Commit', and 'Stages'. The table contains four rows of pipeline data.

Status	Pipeline	Commit	Stages
running	#6453892 by [user] latest	3df39dd6 add data durability to Alex	✓ 🔄
passed	#6453883 by [user] latest	d0f228af Filled in content	✓ ✓ 🕒 00:08:15 📅 4 minutes ago
passed	#6453817 by [user] latest	06a01e18 De Wet Geo Expert	✓ ✓ 🕒 00:08:50 📅 8 minutes ago
passed	#6453004 by [user] latest	0c7954e5 Update index.html.md	✓ ✓ 🕒 00:08:33 📅 59 minutes ago



Continuous integration & delivery

YAML szintaxis

► Példa YAML kód:

```
name: Gipsz Jakab # kulcs-érték párok
age: 42
details: # beágyazott kollekció
  givenname: Jakab
  familyname: Gipsz
  birthyear: 1978
languages: # értékek listája (tömb)
  - Hungarian
  - English
  - German
# több soros szöveg
intro_multi: |
  multiple line
  introduction
intro_single: >
  single line
  introduction
```



Continuous integration & delivery

GitLab CI/CD: jobs

- ▶ A `.gitlab-ci.yml` fájlban feladatokat (*jobs*) definiálhatunk, amelyekben megadhatjuk milyen utasításokat kell végrehajtaniuk (`script`).
- ▶ Pl.:

```
build_program:
```

```
  script:
```

- `apt-get update -qq`
- `apt-get install -yqq openjdk-11-jdk ant`
- `ant compile`
- `ant jar`



Continuous integration & delivery

GitLab CI/CD: multiple jobs

- ▶ Több feladat is definiálható, továbbá megadható egy globális `before_script` elem is, amelyet minden *job* előtt végre kell hajtani. (Felüldefiniálható az egyes feladatokban.)

```
before_script:
```

- apt-get update -qq
- apt-get install -yqq openjdk-11-jdk ant junit

```
build_program:
```

```
script:
```

- ant compile
- ant jar

```
test_program:
```

```
script:
```

- ant compile-test
- ant test



Continuous integration & delivery

GitLab CI/CD: stages

- ▶ A folyamatos integráció feladatait egymást követő szakaszokra (*stages*) oszthatjuk
 - ▶ alapértelmezetten 3 *stage* van: *build*, *test*, *deploy*
 - ▶ ez tetszőlegesen felüldefiniálhatjuk

```
stages:
```

```
- lint  
- build
```

- ▶ Egy *stage* feladatai egymástól függetlenül párhuzamosítva végrehajthatóak (több *runner* bevonásával)
 - ▶ a *stagek* egymásra épülnek, amennyiben egy *stage* valamely feladata hibával zárul, a rá épülő *stagek* nem kerülnek végrehajtásra

Continuous integration & delivery

GitLab CI/CD: stages

- ▶ A folyamat *stagekre* osztása:

```
before_script:
```

- apt-get update -qq
- apt-get install -yqq openjdk-11-jdk ant junit

```
build_program:
```

```
stage: build
```

```
script:
```

- ant compile
- ant jar

```
test_program:
```

```
stage: test
```

```
script:
```

- ant compile-test
- ant test

Continuous integration & delivery

GitLab CI/CD: artifacts

- ▶ A CI feladatok részeként előállított bináris vagy egyéb állományokat megőrizhetjük (*artifact*)

pdf:

```
script: pdftex paper.tex
```

```
artifacts:
```

```
paths:
```

```
- paper.pdf
```

```
expire_in: 1 month
```

- ▶ Az *artifactok* a GitLab webes felületéről könnyen letölthetőek

Job artifacts

The artifacts will be removed in 6 days

[Keep](#) [Download](#) [Browse](#)

Job details

Duration: 5 minutes 4 seconds

Finished: about 5 hours ago

Runner: #44028

[Raw](#) [Erase](#)

Continuous integration & delivery

GitLab CI/CD: artifacts

► *Artifactok* definiálása:

```
before_script:
```

- apt-get update -qq
- apt-get install -yqq openjdk-11-jdk ant

```
build_program:
```

```
stage: build
```

```
script:
```

- ant compile
- ant jar

```
artifacts:
```

```
paths:
```

- **dist/program.jar**

```
expire_in: 1 week
```



Continuous integration & delivery

GitLab CI/CD: dependencies

- Az egymástól függő programok (modulok) ellenőrzése könnyen redundáns végrehajtáshoz vezethet:

```
before_script: ...
```

```
build_program_A:  
  stage: build  
  script:  
    - cd module_A  
    - ant jar
```

```
build_program_B: # függ program_A.jar állománytól  
  stage: build  
  script:  
    - cd module_A  
    - ant jar  
    - cd ..  
    - cd module_B  
    - ant jar
```

Continuous integration & delivery

GitLab CI/CD: dependencies

► *Artifactok átadása jobok között:*

```
before_script: ...
```

```
build_program_A:
```

```
  stage: build
```

```
  script:
```

```
    - cd module_A
```

```
    - ant jar
```

```
  artifacts:
```

```
    paths:
```

```
      - module_A/dist/program_A.jar
```

```
build_program_B: # függ program_A.jar állománytól
```

```
  stage: build
```

```
  script:
```

```
    - cd module_B
```

```
    - ant jar
```

```
  dependencies:
```

```
    - build_program_A
```

Continuous integration & delivery

Docker images

- Docker container alapú GitLab Runner esetén megadhatjuk melyik *docker image*-ből kívánunk kiindulni:

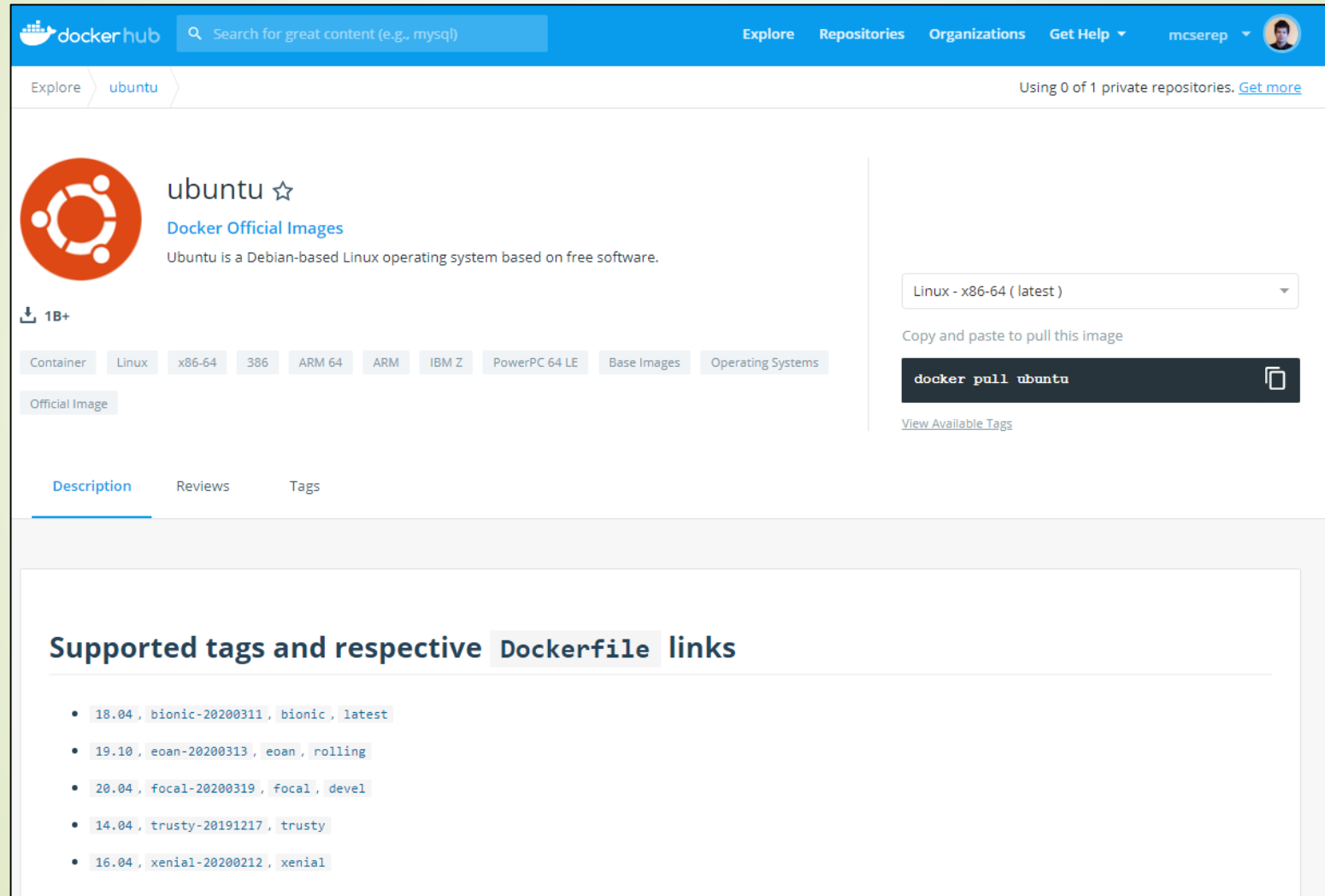
```
image: ubuntu:18.04
```

image név
tag név

- Docker image-t egy *docker registry*-ből kérhetünk:
 - Alapértelmezetten a publikus Docker Hub-ot használjuk, ahová saját image is feltölthető: <https://hub.docker.com/>
 - Használható privát docker registry is (pl. vállalati környezet)
image: mycompany.com:5000/custom:latest
- Ha nem adjuk meg, akkor a runner konfigurációja adja meg a használandó image-t
 - a szofttech.inf.elte.hu runnerjei az `ubuntu:18.04` imagere vannak konfigurálva

Continuous integration & delivery


Docker Hub



The screenshot shows the Docker Hub interface for the 'ubuntu' repository. The page includes a search bar, navigation links, and a pull button. The 'Supported tags and respective Dockerfile links' section is highlighted.

docker hub Search for great content (e.g., mysql) Explore Repositories Organizations Get Help mcserep

Explore ubuntu Using 0 of 1 private repositories. [Get more](#)

 **ubuntu** ☆
Docker Official Images
Ubuntu is a Debian-based Linux operating system based on free software.

1B+

Container Linux x86-64 386 ARM 64 ARM IBM Z PowerPC 64 LE Base Images Operating Systems

Official Image

Description Reviews Tags

Supported tags and respective Dockerfile links

- 18.04, bionic-20200311, bionic, latest
- 19.10, eoan-20200313, eoan, rolling
- 20.04, focal-20200319, focal, devel
- 14.04, trusty-20191217, trusty
- 16.04, xenial-20200212, xenial

Linux - x86-64 (latest)

Copy and paste to pull this image

```
docker pull ubuntu
```

[View Available Tags](#)

Continuous integration & delivery

GitLab CI/CD

- ▶ További lehetőségek (teljesség igénye nélkül):
 - ▶ `only`, `except`: CI jobok végrehajtásának feltételhez kötése (például csak a *master* branch-en futtatni)
 - ▶ `when`: CI jobok végrehajtásának feltételhez kötése (manuális vs. automatikus végrehajtás)

```
image: maven:latest
build_program:
  stage: build
  script:
    - mvn compile
test_program:
  stage: test
  script:
    - mvn test
deploy_program:
  stage: deploy
  script:
    - mvn deploy
only: master
when: manual
```



Continuous integration & delivery

GitLab CI/CD

- `cache`: fájlok, könyvtárak (tipikusan függőségek) megőrzése CI jobok és pipelineok között

```
image: maven:latest
```

```
cache:
```

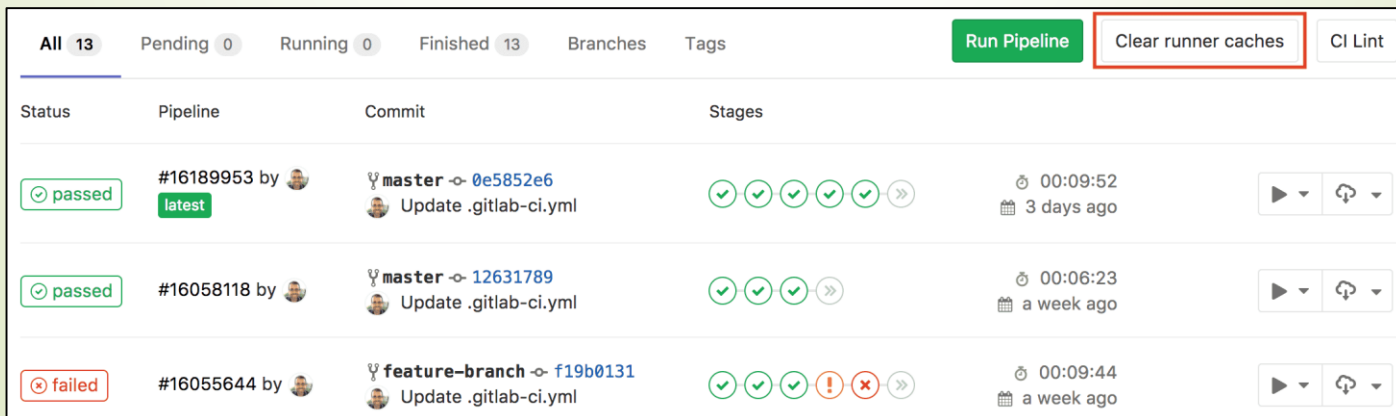
```
  paths:
```

- `.m2/repository`

```
build_program:
```

```
  ...
```

- a cache kiüríthető manuálisan:



The screenshot shows the GitLab CI/CD interface. At the top, there are tabs for 'All 13', 'Pending 0', 'Running 0', and 'Finished 13'. There are also buttons for 'Run Pipeline', 'Clear runner caches' (highlighted with a red box), and 'CI Lint'. Below the tabs, there is a table with columns for 'Status', 'Pipeline', 'Commit', and 'Stages'. The table lists three pipelines:

Status	Pipeline	Commit	Stages	Duration	Actions
passed	#16189953 by [user] latest	master -> 0e5852e6 Update .gitlab-ci.yml	✓ ✓ ✓ ✓ ✓ >>	00:09:52 3 days ago	▶ ⌵ ↺ ⌵
passed	#16058118 by [user]	master -> 12631789 Update .gitlab-ci.yml	✓ ✓ ✓ >>	00:06:23 a week ago	▶ ⌵ ↺ ⌵
failed	#16055644 by [user]	feature-branch -> f19b0131 Update .gitlab-ci.yml	✓ ✓ ✓ ! ✗ >>	00:09:44 a week ago	▶ ⌵ ↺ ⌵

Continuous integration & delivery

GitLab CI/CD

- ▶ `variables`: változók definiálása.
A futtató környezetre számos változó már előre definiált:
<https://docs.gitlab.com/ee/ci/variables/>
- ▶ `services`: szolgáltatások (pl. adatbázis motor) külön Docker containerben futtatása (*docker-compose*)

```
services:
```

```
  - mysql:latest
```

```
variables:
```

```
  MYSQL_DATABASE: my_db
```

```
  MYSQL_USER: my_user
```

```
  MYSQL_PASSWORD: very_secret_password
```

- ▶ a hosztnév `mysql` lesz (`alias` opcióval megadható más)

Continuous integration & delivery

GitLab CI/CD: terminals

- A folyamatos integráció feladatainak végrehajtását egy online terminál ablakon keresztül követhetjük a GitLab webes felületén

```
🔄 running Job #1394 triggered just now by 🌟 Administrator
```

```
Running with gitlab-runner 11.3.0~beta.694.gf4a3dadf (f4a3dadf)
  on shell-runner d8b80d51
Using Shell executor...
Running on Steves-MBP-2...
Fetching changes...
HEAD is now at 1aeb472 Update .gitlab-ci.yml
Checking out 1aeb4725 as master...
Skipping Git submodules setup
$ sleep 15
$ echo "Done"
Done
Terminal is connected, will time out in 30m0s...
```


Continuous integration & delivery

GitLab CI/CD: példa projekt

- Hálózati Pacman játék Java implementációval:

<https://szofttech.inf.elte.hu/mate/pacman-java/>

```
1 image: openjdk:11
2
3 stages:
4   - build
5   - test
6
7 before_script:
8   - apt-get update -yqq
9   - apt-get install -yqq ant junit4
10
11 # Build
12 build_game:
13   stage: build
14   script:
15     - ant compile
16     - ant jar
17
18 # Test
19 test_model:
20   stage: test
21   script:
22     - >
23     ant test
24     -Dlibs.junit_4.classpath=/usr/share/java/junit4.jar|
25     -Dlibs.hamcrest.classpath=/usr/share/java/hamcrest-core.jar
26
```

Continuous integration & delivery

GitLab CI/CD: példa projekt

- Maven build rendszerrel: `dev-maven-java13` fejlesztési ág

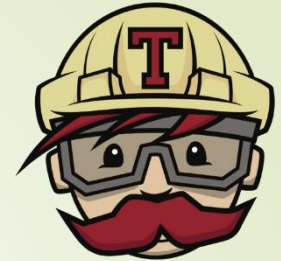
```
1 image: maven:latest
2
3 variables:
4   MAVEN_OPTS: "-Dmaven.repo.local=${CI_PROJECT_DIR}/.m2/repository"
5
6 cache:
7   key: maven-build
8   paths:
9     - .m2/repository
10
11 build:
12   stage: build
13   script:
14     - mvn compile
15 test:
16   stage: test
17   script:
18     - mvn test
```

- A cache egy kulccsal azonosítható, így pl. szabályozható, hogy mely fejlesztési ágak között kerüljön megosztásra.

Continuous integration & delivery

Travis CI

- ▶ Folyamatos integrációs szolgáltatás GitHub projektekhez
 - ▶ Nyílt forráskódú projektekhez ingyenesen használható
 - ▶ <https://travis-ci.com/>
- ▶ Támogatja a Linux, a Windows és a macOS operációs rendszereket, több előkészített környezettel (*image*)
 - ▶ A környezetet a használt programozási nyelvhez állíthatjuk be, az elterjedtebb fordító eszközökkel és könyvtárakkal
- ▶ A CI konfigurációt a `.travis.yml` fájlban adhatjuk meg



Travis CI

Continuous integration & delivery

Travis CI – GitHub integration

The image shows a Travis CI interface with a build history list and a detailed view of a successful build.

Build History:

- Commits on Apr 16, 2019:** Execute tests in Travis CI (377c81b) - Failed (red X)
- Commits on Apr 15, 2019:** Added Travis CI to build CodeCompass (3f52da6) - Passed (green checkmark)
- Commits on Apr 12, 2019:** (b402c4f) - Passed (green checkmark)

Build Details (Commit 3f52da6):

- Status:** **ci** Added Travis CI to build CodeCompass - **#29 passed**
- Restart build:** Button
- Commit:** 3f52da6
- Compare:** 7cc7cc1..3f52da6
- Branch:** ci
- Author:** Máté Cserép
- Language:** C++
- Run Time:** Ran for 19 min 53 sec
- Completed:** about 17 hours ago

Job Log:

- [View config](#)
- [Remove log](#)
- [Raw log](#)
- 1 Worker information (worker_info)
- 6 Build system information (system_info)
- 158
- 159 Adding APT Sources (apt)
- 816
- 817 \$ git clone --depth=50 --branch=ci https://github.com/Ericsson/CodeCompass.git (git.checkout 1.27s)
- 827

Continuous integration & delivery

Travis CI

► Például:

```
os: linux
dist: xenial
language: java

stages:
  - compile
  - test
  - deploy

# ...

jobs:
  include:
    - stage: compile
      - mvn compile
    - stage: test
      - mvn test

# ...
```

Continuous integration & delivery

AppVeyor CI



- ▶ Folyamatos integrációs szolgáltatás
 - ▶ Integrálható a GitHub, GitLab, BitBucket, Visual Studio Team Services platformokkal
 - ▶ Nyílt forráskódú projektekhez ingyenesen használható
 - ▶ <https://www.appveyor.com/>
- ▶ Támogatja a Linux, a Windows operációs rendszereket, több előkészített környezettel (*image*)
 - ▶ Kiemelt .NET és Visual Studio támogatás
- ▶ A CI konfigurációt a `appveyor.yml` fájlban adhatjuk meg

Continuous integration & delivery

AppVeyor CI – webes interfész

aegis

Current build History

Core: Added M-tree (#21).

a year ago by Rónai Péter (committed by Roberto Giachetta)

Core: Added M-tree (#21).

a year ago by Rónai Péter (committed by Roberto Giachetta)

Pull request #21 - M tree

Merge branch 'master' into M_Tree

a year ago by Roberto Giachetta (committed by GitHub)

Core: Added Hilbert R-tree (#20).

a year ago by Rónai Péter (committed by Roberto Giachetta)

Pull request #21 - M tree

Finished M-Tree

a year ago by Rónai Péter

1.0.38

a year ago in 6 min 37 sec

1.0.37

a year ago in 6 min 49 sec

1.0.36

a year ago in 6 min 30 sec

1.0.35

a year ago in 5 min 29 sec

Console

Messages 34

Tests

Artifacts

```
1 Build started
2 git clone -q --branch=master https://github.com/robertogiachetta/aegis.git C:\projects\aegis
3 git checkout -qf 64427e34f338989da19925565681efdb67e46c17
4 nuget restore src\AEGIS.sln
5 MSBuild auto-detection: using msbuild version '15.5.180.51428' from 'C:\Program Files (x86)\Microsoft Visual
  Studio\2017\Community\MSBuild\15.0\bin'.
6 Restoring NuGet package NUnit.3.6.1.
7 Restoring NuGet package Shouldly.2.8.2.
8 Restoring NuGet package StyleCop.Analyzers.1.0.0.
9 Restoring NuGet package Castle.Core.4.1.0.
10 Restoring NuGet package Moq.4.7.63.
11 Restoring NuGet package NUnit.ConsoleRunner.3.6.1.
12 Restoring NuGet package OpenCover.4.6.519.
13 Restoring NuGet package SimpleInjector.4.0.7.
14 Restoring NuGet package Microsoft.Win32.Primitives.4.3.0.
15 Restoring NuGet package System.Diagnostics.DiagnosticSource.4.3.0.
16 Adding package 'Microsoft.Win32.Primitives.4.3.0' to folder 'C:\projects\aegis\src\packages'
17 Adding package 'System.Diagnostics.DiagnosticSource.4.3.0' to folder 'C:\projects\aegis\src\packages'
18 Added package 'System.Diagnostics.DiagnosticSource.4.3.0' to folder 'C:\projects\aegis\src\packages'
```



Continuous integration & delivery

AppVeyor CI

► Például:

```
version: 1.0.{build} # Version format
image: Visual Studio 2017 # Build worker image
platform: Any CPU # Build platform
configuration: Debug # Build Configuration

# Execute script before build
before_build:
  - dotnet restore src\MyProject.sln

# Execute build script
build_script:
  - dotnet build src\MyProject.sln

# Execute test script
test_script:
  - dotnet test src\MyProject.sln
```


Continuous integration & delivery

Jenkins

- ▶ Nyílt forráskódú folyamatos integrációs szolgáltatás
 - ▶ Integrálható a GitHub, GitLab, és egyéb projektvezető szolgáltatásokkal
 - ▶ Nem nyújt hoszting szolgáltatást, de más vállalkozások kínálnak (pl. CloudBees)
 - ▶ <https://jenkins.io/>
- ▶ A CI konfigurációt a `Jenkinsfile` adja meg, például:

```
pipeline {
  agent { docker { image 'maven:3.3.3' } }
  stages {
    stage('build') {
      steps {
        sh 'mvn compile'
      }
    }
  }
}
```

