MISKOLCI EGYETEM
GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR

Szoftvertechnológia gyakorlatok

GEIAL316-B2

Continous Integration (CI) eszközök

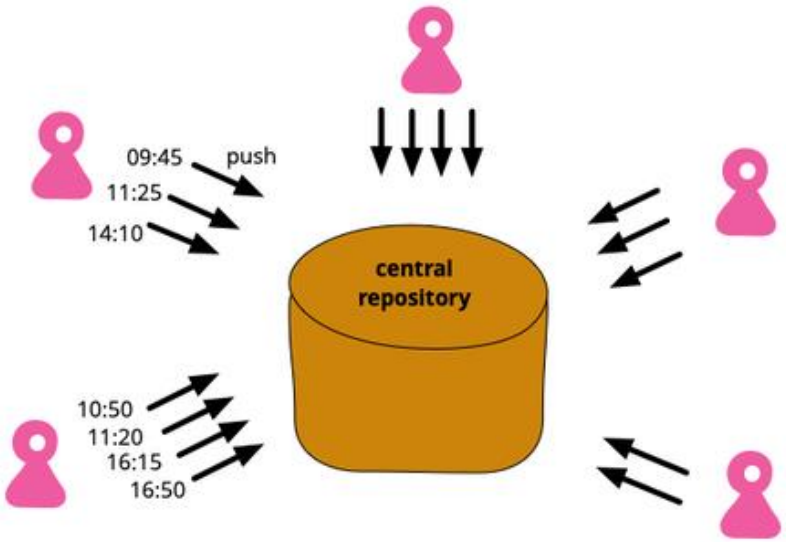
Dr. Tompa Tamás
egyetemi adjunktus
Általános Informatikai Intézeti Tanszék

Miskolc, 2025

Folytonos integráció

- „Continuous Integration is a **software development practice** where **each member of a team merges their changes into a codebase** together with their colleagues changes **at least daily**. Each of these integrations is **verified by an automated build** (including test) to detect integration errors as quickly as possible.”

Martin Fowler



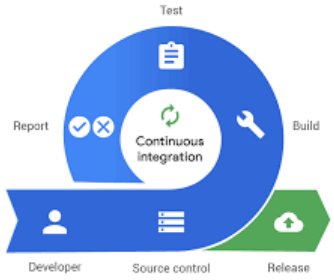
Forrás: https://martinfowler.com/articles/continuousIntegration.html

Folytonos integráció

- modern szoftverfejlesztési gyakorlat, amely **automatizálja a kód build-elését, tesztelését és telepítését**
- **gyakran** (naponta többször) **integrálják a kódváltoztatásokat a fő ágba** (main/master branch)
- gyors visszajelzés, hogy **az új kód nem rontotta el a rendszert**
- **kapcsolódó folyamat a f**olytonos szállítás (**Continuous Delivery** - CD)
  - a sikeresen tesztelt kód **automatikus telepítése** egy teszt vagy éles (production) környezetbe

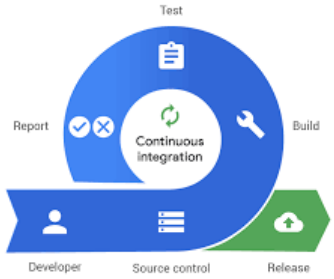
- **CI/CD eszközök**
  - Jenkins (nyílt forráskódú)
  - GitHub Actions (beépített CI/CD a GitHub-on)
  - GitLab CI/CD (GitLab saját integrált rendszere)
  - CircleCI (felhőalapú CI/CD)
  - Travis CI (GitHub-projektekkel integrálható)
  - Azure DevOps (Microsoft)

CI folyamatok



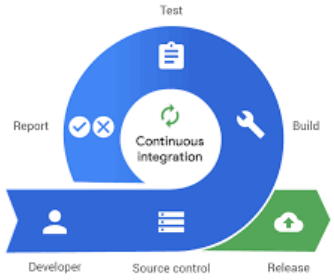
- **Kódellenőrzés**: automatikus kódellenőrzés futtatása a kódminőség javítása érdekében
  - szintaxis- és stílusellenőrzés (checkstyle)
- **Egységtesztek futtatása**: egységtesztek futtatása a kódváltoztatások után
- **Build folyamat**: a szoftver build folyamatának automatizálása
  - forráskód fordítása
  - futtatható fájlok vagy csomagok létrehozása

CI folyamatok



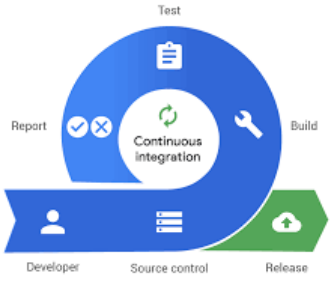
- **Dokumentáció generálása**: automatikus dokumentáció generálása a forráskódból
  - pl. API dokumentáció, felhasználói útmutatók
- **Kódlefedettség ellenőrzése**: az egységtesztek kódlefedettségének ellenőrzése a kód minőségének biztosítása érdekében
  - min. 80%
- **Funkcionális tesztek futtatása**: Az alkalmazás funkcionális tesztjeinek automatizálása a kódváltoztatások értékelésére.

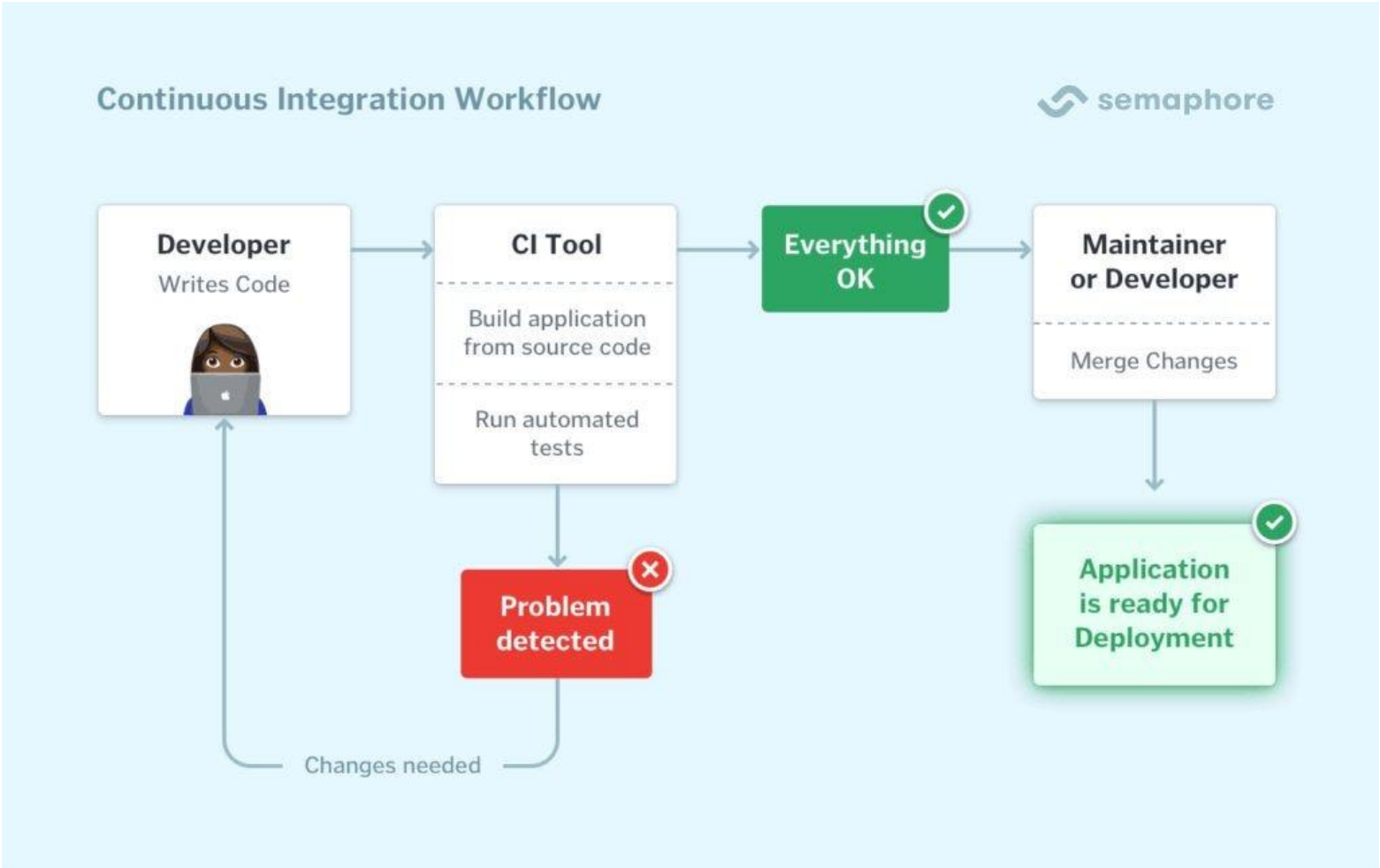
CI folyamatok



- **Teljesítménytesztek futtatása**: az alkalmazás teljesítményének automatizált tesztelése
- **Biztonsági ellenőrzések**: automatikus biztonsági ellenőrzések futtatása a potenciális biztonsági kockázatok azonosítása érdekében
- **Értesítések küldése**: értesítések automatikus küldése a CI rendszerben történő változásokról, például sikeres vagy sikertelen build-ekről, tesztek futtatásáról stb.

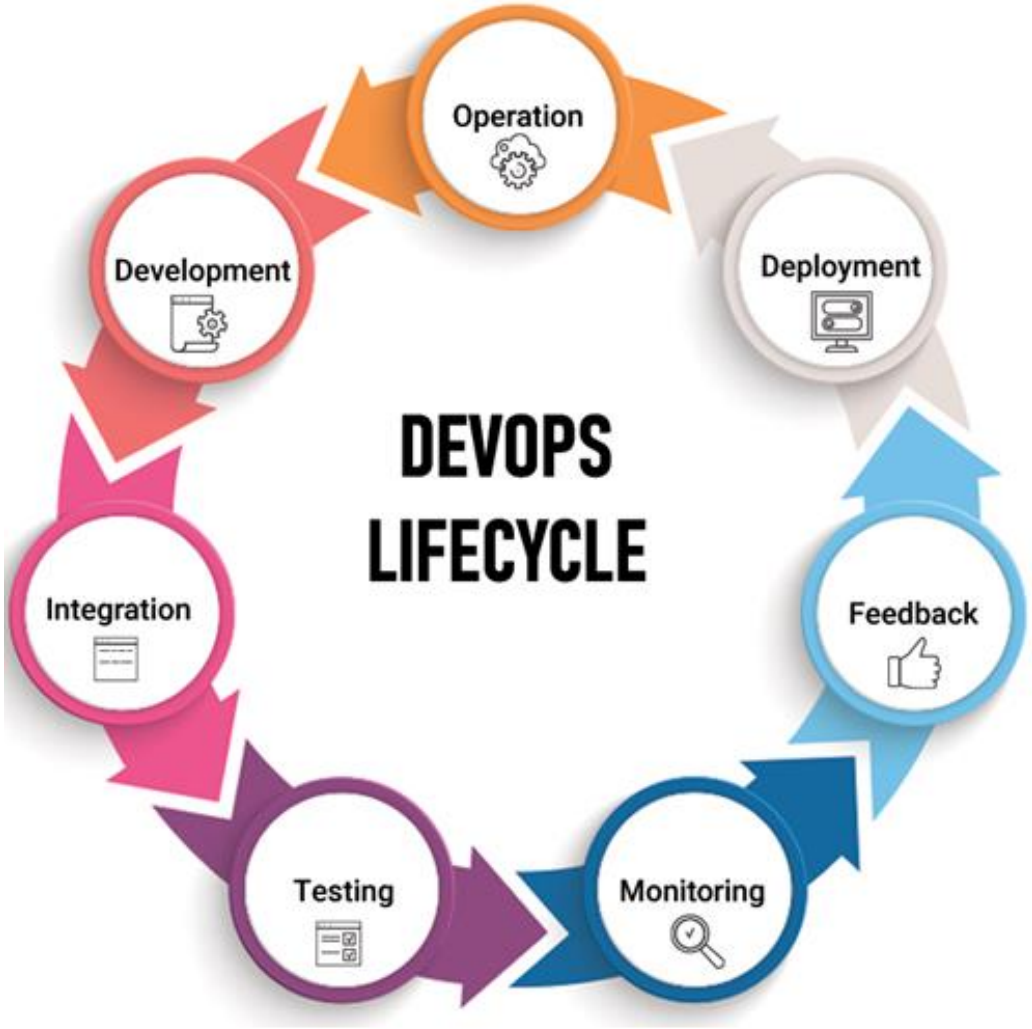
CI folyamatok





Forrás: https://medium.com/@AbbasPlusPlus/the-continuous-integration-flow-a-step-by-step-guide-7e32a08fd54c

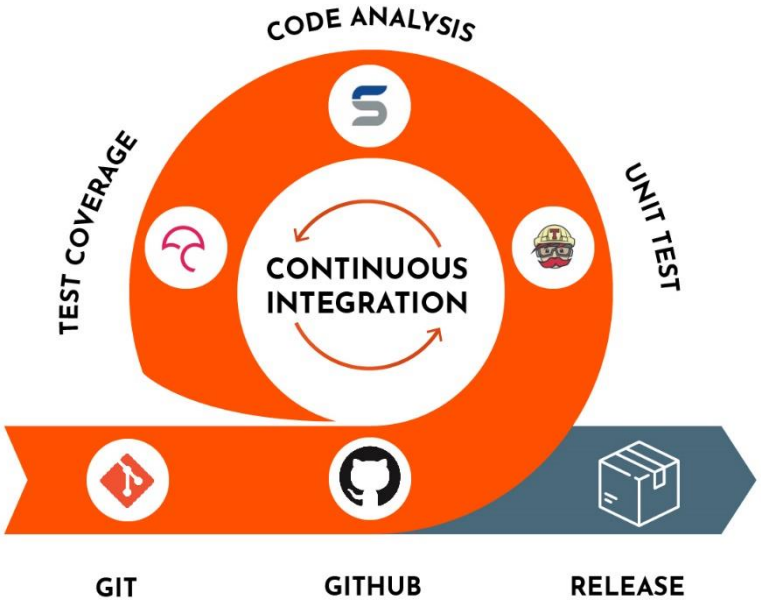
CI életciklus



Forrás: https://www.spiceworks.com/tech/devops/articles/what-is-devops-lifecycle/

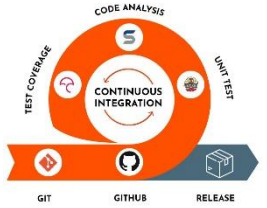
CI életciklus - folyamatok

1. Tervezés
2. Fejlesztés (kódolás)
3. Verziókövetés
4. Kódellenőrzés (review)
5. Build-elés
6. Egységtesztelés
7. Teszteredmény értékelés
8. Értesítés küldés
9. Javítás
10. Folyamat ismétlése



Forrás: https://devonblog.com/wp-content/uploads/2019/07/c-i-100.jpg

CI életciklus - folyamatok

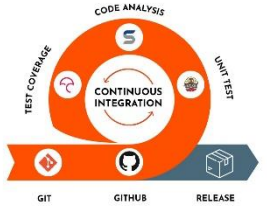


- A fejlesztők a **kódot verziókezelő rendszerben** (pl. Git, Mercurial, SVN) kezelik
  - az új kódváltoztatásokat branch-ekben (pl. feature-xyz) végzik
  - pull request (PR) vagy merge request (MR) segítségével integrálják a kódot a fő ágba (main / master)

- A fejlesztői környezetben a **CI rendszer** (pl. Jenkins, GitHub Actions, GitLab CI/CD) **érzékeli az új változtatásokat**
  - lefordítja a kódot

- CI rendszer különböző **teszteket futtat**, ha egy teszt meghiúsul, **a fejlesztő értesítést kap**

CI életciklus - folyamatok



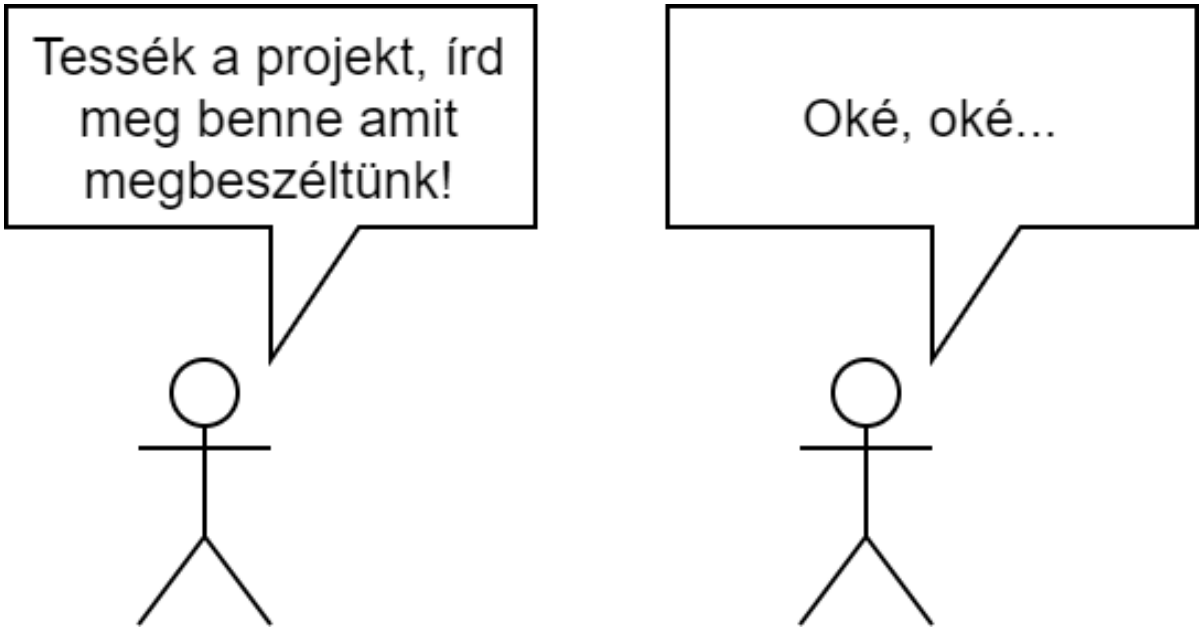
- **Kódminőség ellenőrzése** (Code Quality & Static Analysis)
  - linting: ellenőrzi a kódszabványokat (pl. ESLint, Prettier, Pylint, SonarQube)

- **Ha a build és a tesztek sikeresek**, a kimeneti **fájlokat** (binárisok, Docker image-ek, csomagok) **eltárolják** egy tárhelyen (pl. Artifactory, Nexus, Docker Registry)

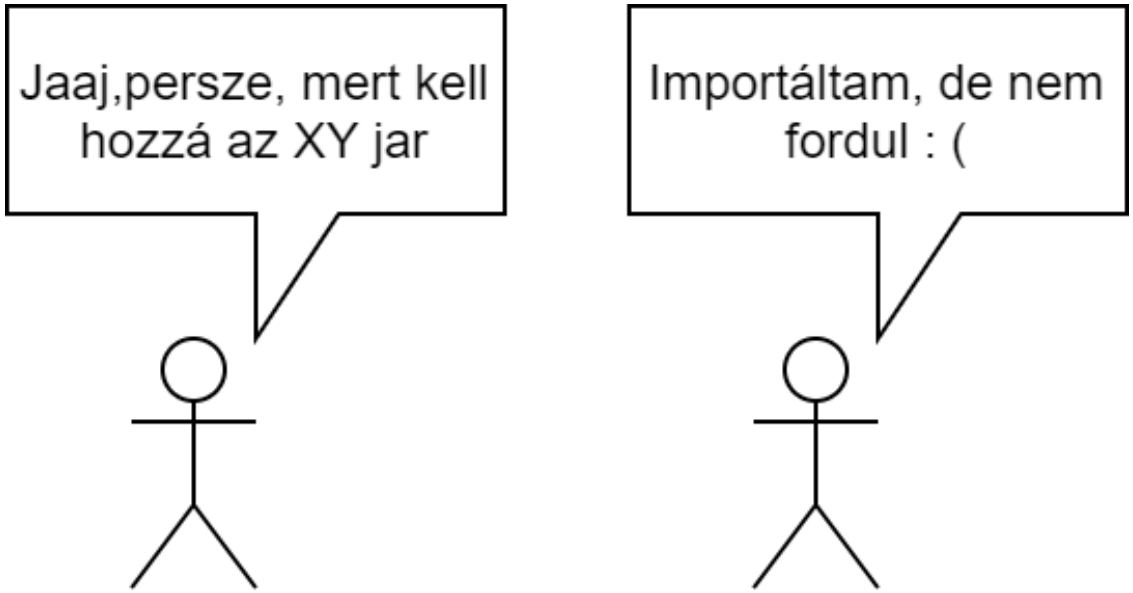
- A CI rendszer **e-mailben más csatornán értesíti** a fejlesztőket a **build és a tesztelés eredményeiről**

- **Ha a CI folyamat sikeres**, a rendszer **továbbítja a kódot a folytonos telepítés** (Continuous Deployment) szakaszba

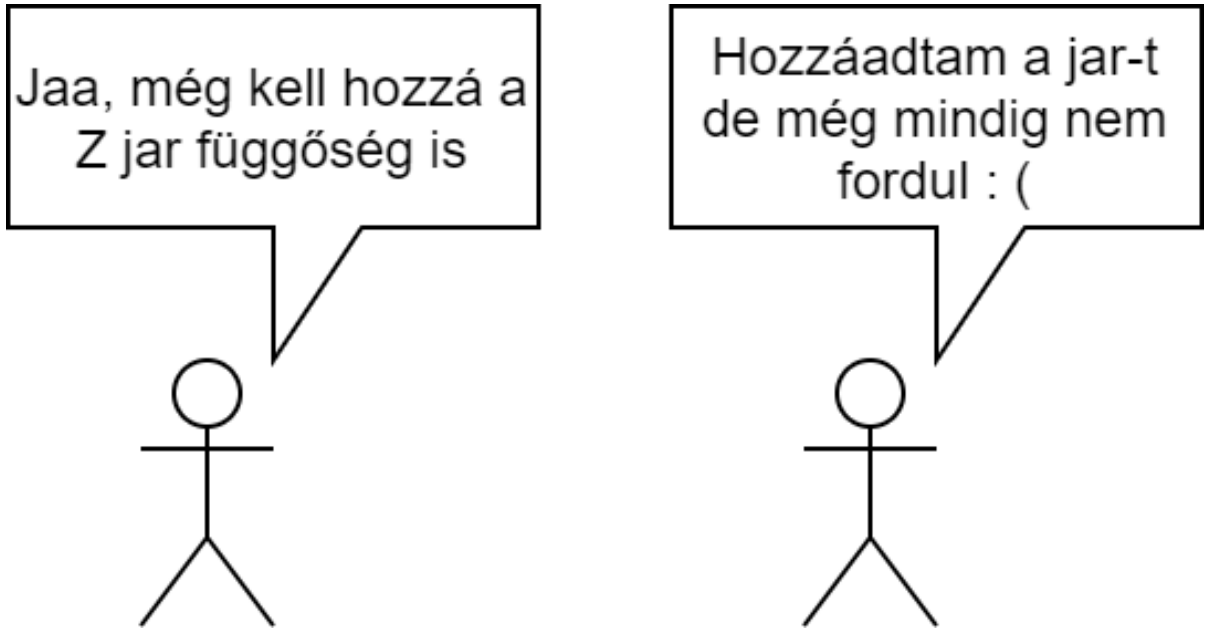
CI build eszközök – miért



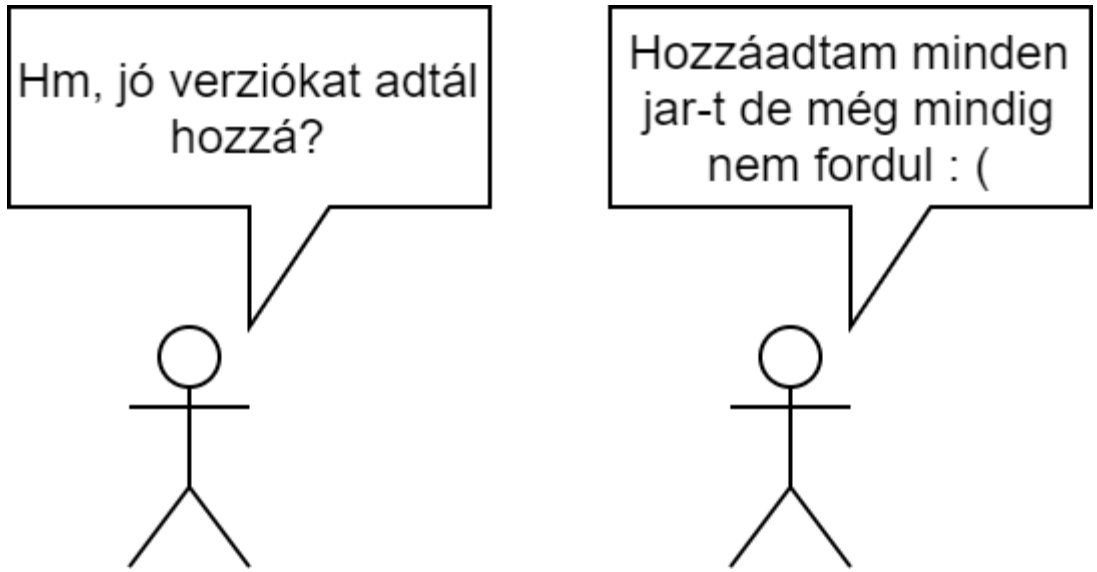
CI build eszközök – miért



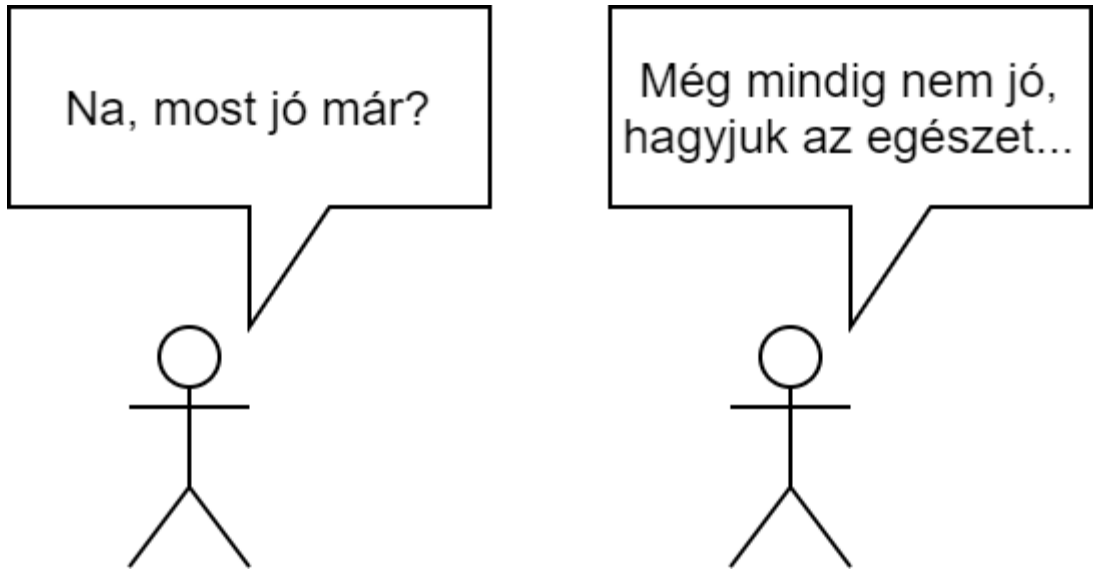
CI build eszközök – miért



CI build eszközök – miért



CI build eszközök – miért



CI build eszközök

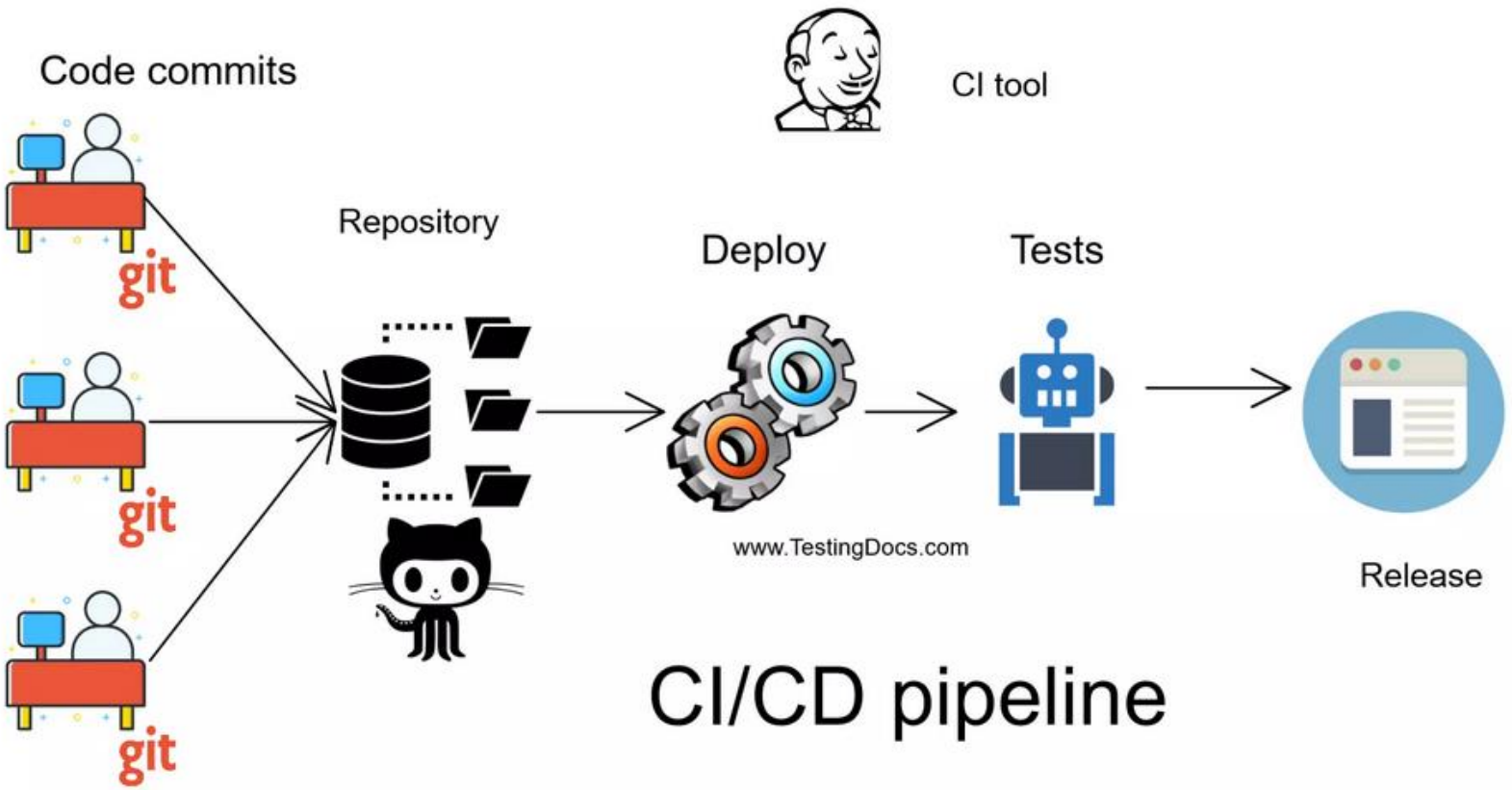
- **„A build tool is a tool that automates everything related to building the software project.”**

  Jakob Jenkov

  - forráskód generálás
  - dokumentáció generálás forráskód alapján
  - forráskód fordítás
  - lefordított kód csomagolása (Jar, Zip, War, stb.)
  - csomagolt kód telepítése szerverre, repo-ba (vagy máshová)

Forrás: https://jenkov.com/tutorials/maven/maven-tutorial.html

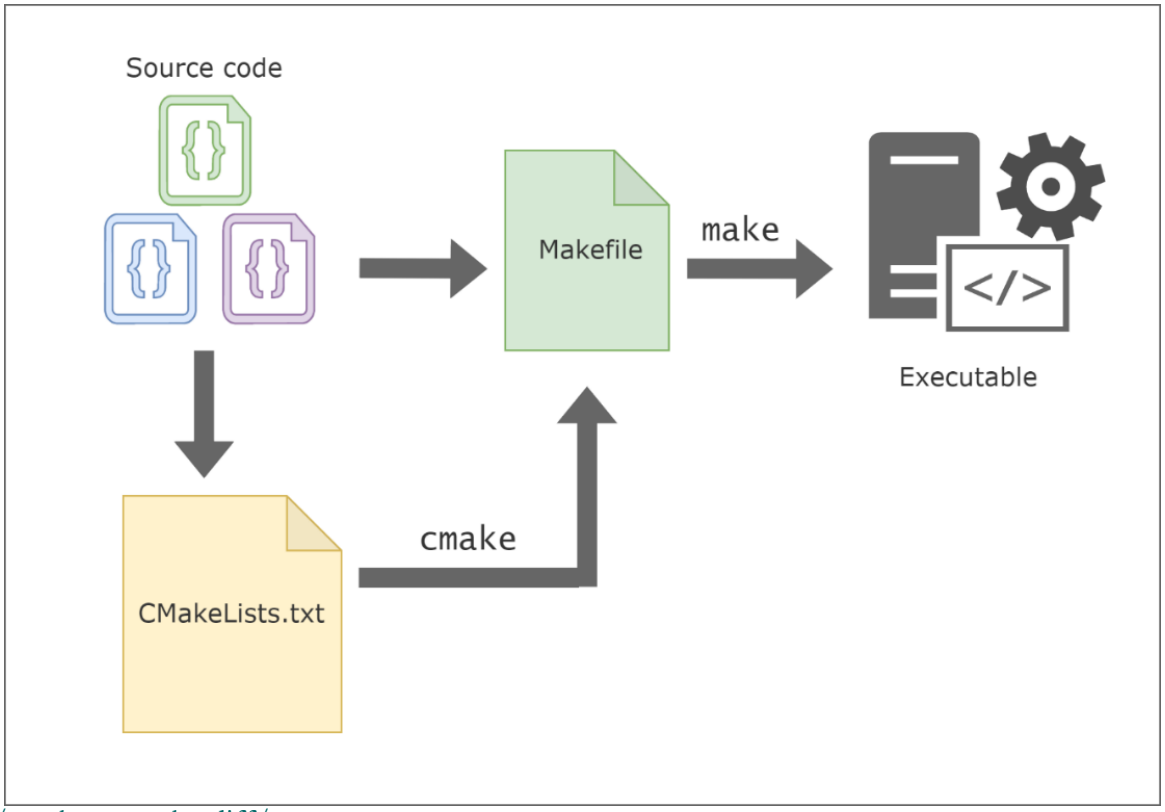
CI build eszközök



Forrás: https://www.testingdocs.com/continuous-integration-tools/

CI build eszközök - Make

- **Make**
  - C/C++
  - build-elés automatizálása, fordításhoz szükséges beállítások



Forrás: https://earthly.dev/blog/cmake-vs-make-diff/

CI build eszközök - Make

- **Make példa**

hellomake.c

```
#include <hellomake.h>

int main() {
  // call a function in another file
  myPrintHelloMake();

  return(0);
}
```

hellofunc.c

```
#include <stdio.h>
#include <hellomake.h>

void myPrintHelloMake(void) {

  printf("Hello makefiles!\n");

  return;
}
```

hellomake.h

```
/*
example include file
*/

void myPrintHelloMake(void);
```

Makefile 1

```
hellomake: hellomake.c hellofunc.c
	gcc -o hellomake hellomake.c hellofunc.c -I.
```


Forrás: https://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/

CI build eszközök – Java classpath

- **Java classpath**
  - **azon helyek a listája, ahol a JVM keresni fogja az osztályokat és más erőforrások**
  - lehetővé teszi a külső könyvtárak (jar fájlok) beágyazását
  - `javac -cp` parancs

```
javac -cp C:/.../jardir1/*;C:/.../jardir2/* class_with_main_method
```

```
javac -cp ".:/home/path/mail.jar:/home/path/servlet.jar;"
MyJavaFile.java
```

  - nehezen megvalósítható, macerás, hosszú parancsok
    - → build eszközök (Ant, Maven stb.) alkalmazása

CI build eszközök – Apache Ant

- **Apache Ant**
  - Make fájl Java-hoz
  - XML (fordítás, csomagolás, tesztek)
  - független modulok újrafelhasználása

Java-only	Ant

`md build\classes`
`javac`
`  -sourcepath src`
`  -d build\classes`
`  src\oata\HelloWorld.java`
`echo Main-Class: oata.HelloWorld>mf`
`md build\jar`
`jar cfm`
`  build\jar\HelloWorld.jar`
`  mf`
`  -C build\classes`
`  .`
 	
`java -jar build\jar\HelloWorld.jar`
	`<mkdir dir="build/classes"/>`
	`<javac`
	`  srcdir="src"`
	`  destdir="build/classes"/>`
	`<!-- automatically detected -->`
	`<!-- obsolete; done via manifest tag -->`
	`<mkdir dir="build/jar"/>`
	`<jar`
	`  destfile="build/jar/HelloWorld.jar"`
	
	`  basedir="build/classes">`
	`  <manifest>`
	`  <attribute name="Main-Class" value="oata.HelloWorld"/>`
	`  </manifest>`
	`</jar>`
	`<java jar="build/jar/HelloWorld.jar" fork="true"/>`

Forrás: https://ant.apache.org/manual/tutorial-HelloWorldWithAnt.html
https://en.wikipedia.org/wiki/Apache_Ant

CI build eszközök – Apache Maven

- **Apache Maven**

  - „Maven is a build tool, a project management tool, an abstract container for running build tasks.”
  - 2004.07.13. - első verzió
  - projektmenedzsment eszköz Java alapú projektek automatizálására és build folyamat kezelésére

  - Célkitűzések
    - projektmenedzsment egyszerűsítése (pom.xml)
    - központosított függőségkezelés (Maven repo: https://mvnrepository.com/)
    - modularitás és testreszabhatóság (plugin-ok)
    - közösségi támogatás (nyílt forráskód, Apache License v2)
    - egységes rendszer szoftverprojektek összeállításához

CI build eszközök – Apache Maven

- **Telepítése**
  - JDK szükséges
  - https://maven.apache.org/download.cgi
  - kicsomagolás után: `C:\apache-maven-3.9.5`
  - környezeti változó beállítása
    - Path-hoz hozzáadni: `C:\apache-maven-3.9.5`
      - vagy `export PATH=/opt/apache-maven-3.9.5/bin:$PATH`
    - ezután cmd-ben: `mvn --version` parancs

```
C:\Users\Tompa_Tamas>mvn --version
Apache Maven 3.9.5 (57804ffe001d7215b5e7bcb531cf83df38f93546)
Maven home: C:\apache-maven-3.9.5
Java version: 18.0.1.1, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-18.0.1.1
Default locale: hu_HU, platform encoding: UTF-8
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

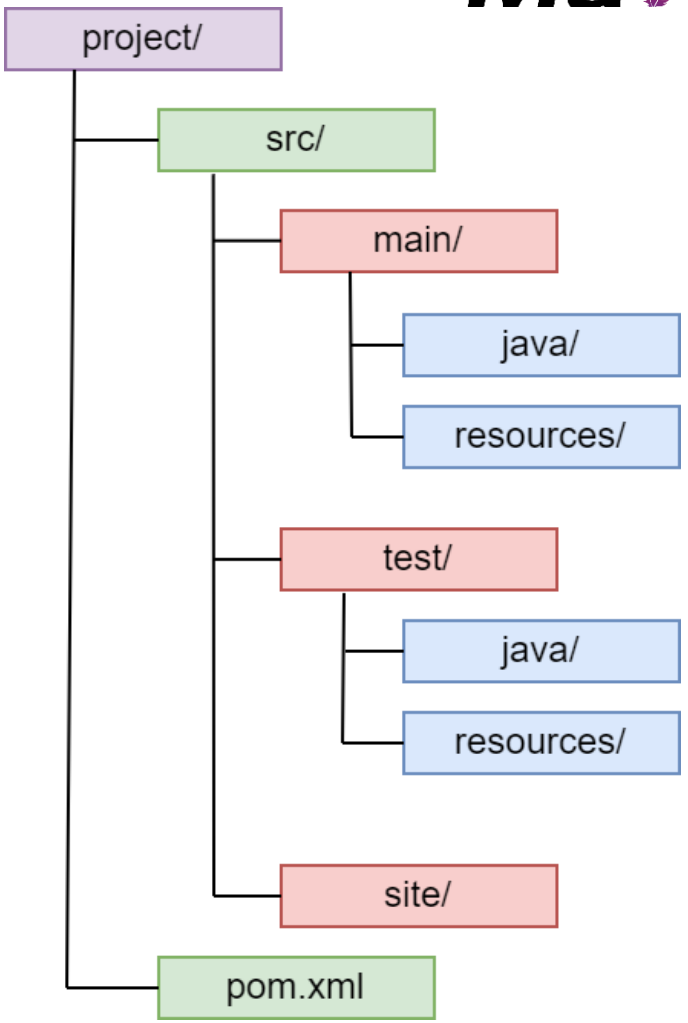
CI build eszközök – Apache Maven

- **IDE integráció**
  - IntelliJ IDEA 
    - beépített támogatás
  - Eclipse 
    - https://www.vogella.com/tutorials/EclipseMaven/article.html
    - Indigo verziótól felfelé már tartalmazza
  - Visual Studio Code 
    - Maven for Java plugin
    - https://code.visualstudio.com/docs/java/java-build

- **Hivatalos dokumentáció**
  - https://maven.apache.org/guides/index.html

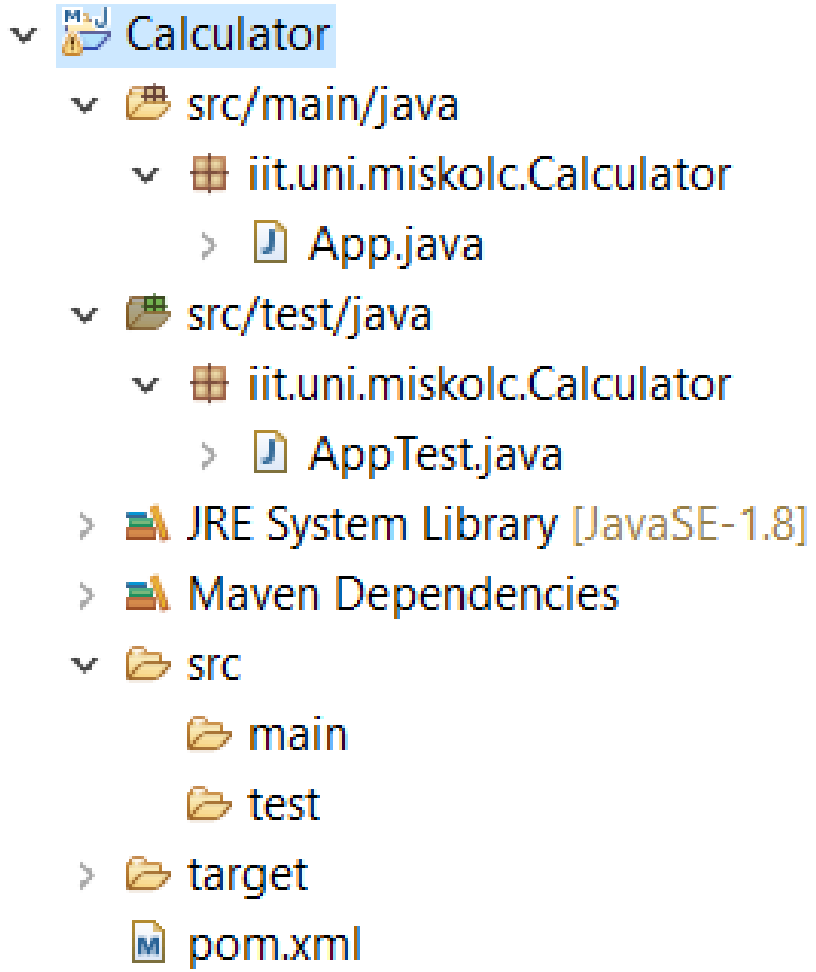
CI build eszközök – Apache Maven

- **Könyvtár szerkezet**
  - szabványos, fix struktúra
  - https://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html
  - **src/main/java**: forráskód
  - **src/main/resources**: konfigurációs fájlok, adatbázis-konfigurációk, nyelvi fájlok stb.
  - **src/test/java**: teszt forráskódja
  - **src/test/resources**: tesztadatok, konfigurációs fájlok, mock adatbázisok stb.
  - **pom.xml**: a projekt konfigurációs fájlja
  - **target**: build folyamat által generált kimenet



CI build eszközök – Apache Maven

- **Könyvtár szerkezet példa**



CI build eszközök – Apache Maven

- **pom.xml**
  - **Project Object Model**: a projeket leíró konfigurációs fájl
  - **Archetpye**: létrehozás mintából (pl. quickstart)
    - könyvtárszerkezetet, vázat létrehozza: `mvn archetype:generate`

```
<project>
      <modelVersion>4.0.0</modelVersion>
      <groupId>com.mycompany.app</groupId>
      <artifactId>my-app</artifactId>
      <version>1</version>
</project>

<dependencies>
       <dependency>
            …
       </dependency>
</dependencies>
```

CI build eszközök – Apache Maven

- **pom.xml**

  - `<project>`: a projekt gyökéreleme, amely tartalmazza az összes további elemet

  - Maven koordináták: `groupid:artifactid:version`
    - `<groupid>`: a projekt egyedi csoportazonosítója
      - `pl.: org.apache.maven`
    - `<artifactid>`: a projekt neve
      - `pl.:my-project`
    - `<version>`: a projekt verziószáma
      - `pl.: 3.4.5; 1.0-SNAPSHOT`
      - `snapshot:` aktív fejlesztés alatt ál, egy pillanatnyi (belső) állapota a projektnek
      - `release:` kiadható, stabli verzió

CI build eszközök – Apache Maven

- **pom.xml**
  - `dependencies:` a projekt függőségeit tartalmazó elem

```
<dependencies>
        <dependency>
            …
        </dependency>
</dependencies>
```

  - `<build>:` az építési konfigurációt tartalmazó elem
  - `<plugins>:` a Maven pluginok konfigurációját tartalmazó elem
  - `<repositories>:` a Maven repository-konfigurációját tartalmazó elem

CI build eszközök – Apache Maven

- **pom.xml példa**

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>hu.iit.unimiskolc</groupId>
    <artifactId>Calculator</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>jar</packaging>

    <name>Calculator</name>
    <url>http://maven.apache.org</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
    </properties>

    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>

        <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
        <dependency>
```

CI build eszközök – Apache Maven

- **settings.xml**
  - 2 opcionális `settings.xml` fájl:
  - Maven install jegyzék: `$M2_HOME/conf/settings.xml`
  - User home jegyzék: `${user.home}/.m2/settings.xml`
  - Local repo útvonalának megadása
  - Aktív build profil megadása
  - https://maven.apache.org/settings.html

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 https://maven.apache.org/xsd/settings-1.0.0.xsd">
  <localRepository/>
  <interactiveMode/>
  <offline/>
  <pluginGroups/>
  <servers/>
  <mirrors/>
  <proxies/>
  <profiles/>
  <activeProfiles/>
</settings>
```

CI build eszközök – Apache Maven

- **Modul-ok**
  - a projekt részekre bontása
    - külön fordítási egységek
    - egymástól függetlenül fordíthatók, tesztelhetők és csomagolhatók
    - minden modulnak saját pom.xml
    - pl.: üzleti logika és gui szétbontása
  - egy modul lehet egy könyvtár vagy egy funkcionális egység
  - az egyes modulok függőségként hivatkozhatnak más modulokra
    - `pom.xml`-ben megadhatóak

CI build eszközök – Apache Maven

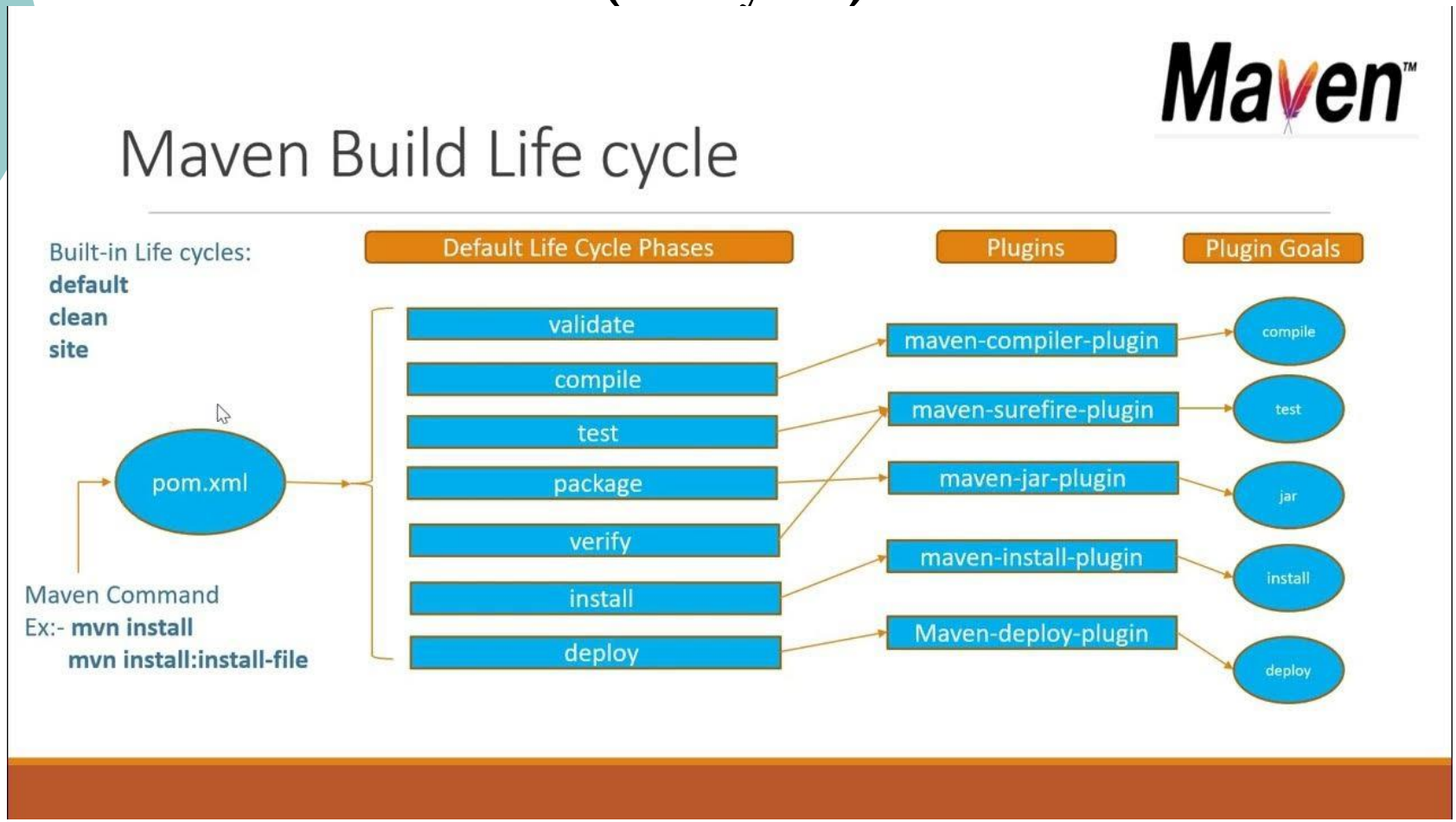
- **Modul-ok példa**

```
<project>
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.example</groupId>
    <artifactId>parent-project</artifactId>
    <version>1.0.0</version>
    <packaging>pom</packaging>

    <modules>
        <module>business-logic</module>
        <module>web-ui</module>
    </modules>
</project>
```

- szülő projekt: `parent-project`
- modulok: `business-logic` és `web-ui`

CI build eszközök – Apache Maven

- **Maven életciklus (lifecycle)**



Forrás: https://www.youtube.com/watch?app=desktop&v=Yw5KnoH0MSo

CI build eszközök – Apache Maven

- **Maven életciklus (lifecycle)**

  - **Clean (Tisztítás):** az előző build során létrehozott fájlok törlése. Ide tartozik a target mappa és a build fájlok törlése
  - **Validate (Érvényesítés):** projekt felépítésének, struktúrájának és konfigurációjának ellenőrzése.
  - **Compile (Fordítás):** forráskódok fordítása a megadott nyelvi szabványnak megfelelően, target mappa létrehozása
  - **Test (Tesztelés**): a projektben definiált tesztesetek futtatása
  - **Package (Csomagolás)**: a fordított forráskód és más szükséges erőforrások csomagolása adott formátumban (például JAR, WAR, stb).

CI build eszközök – Apache Maven

- **Maven életciklus (lifecycle)**

  - **Verify (Ellenőrzés):** a csomagolás során létrehozott fájlok ellenőrzése annak érdekében, hogy megfelelnek-e az előre meghatározott minőségi irányelveknek és szabályoknak
  - **Install (Telepítés):** a csomagolt fájlok telepítése a helyi Maven repository-ba (.m2). Ezek a fájlok azután hozzáférhetők más projektek számára a függőségként való használathoz
  - **Deploy (Közzététel):** A csomagolt fájlok más távoli repository-kba való publikálása,

CI build eszközök – Apache Maven

- **Maven életciklus (lifecycle) parancsok**

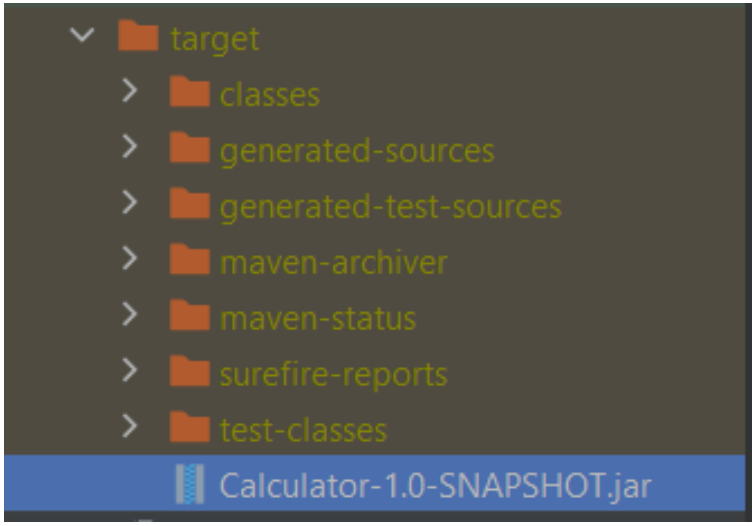
  - `mvn clean`: törli a target mappát (korábbi build fájlokat)
  - `mvn compile`: fordítja a forráskódot a src/main/java mappában
  - `mvn test`: futtatja a teszteket a src/test/java mappában
  - `mvn package`: csomagolja a fordított forráskódot egy futtatható állományba (pl. JAR)
  - `mvn install`: telepíti a csomagot a helyi Maven repository-ba (.m2)
  - `mvn deploy`: közzéteszi a csomagot adott távoli repository-ban

  - https://jenkov.com/tutorials/maven/maven-commands.html

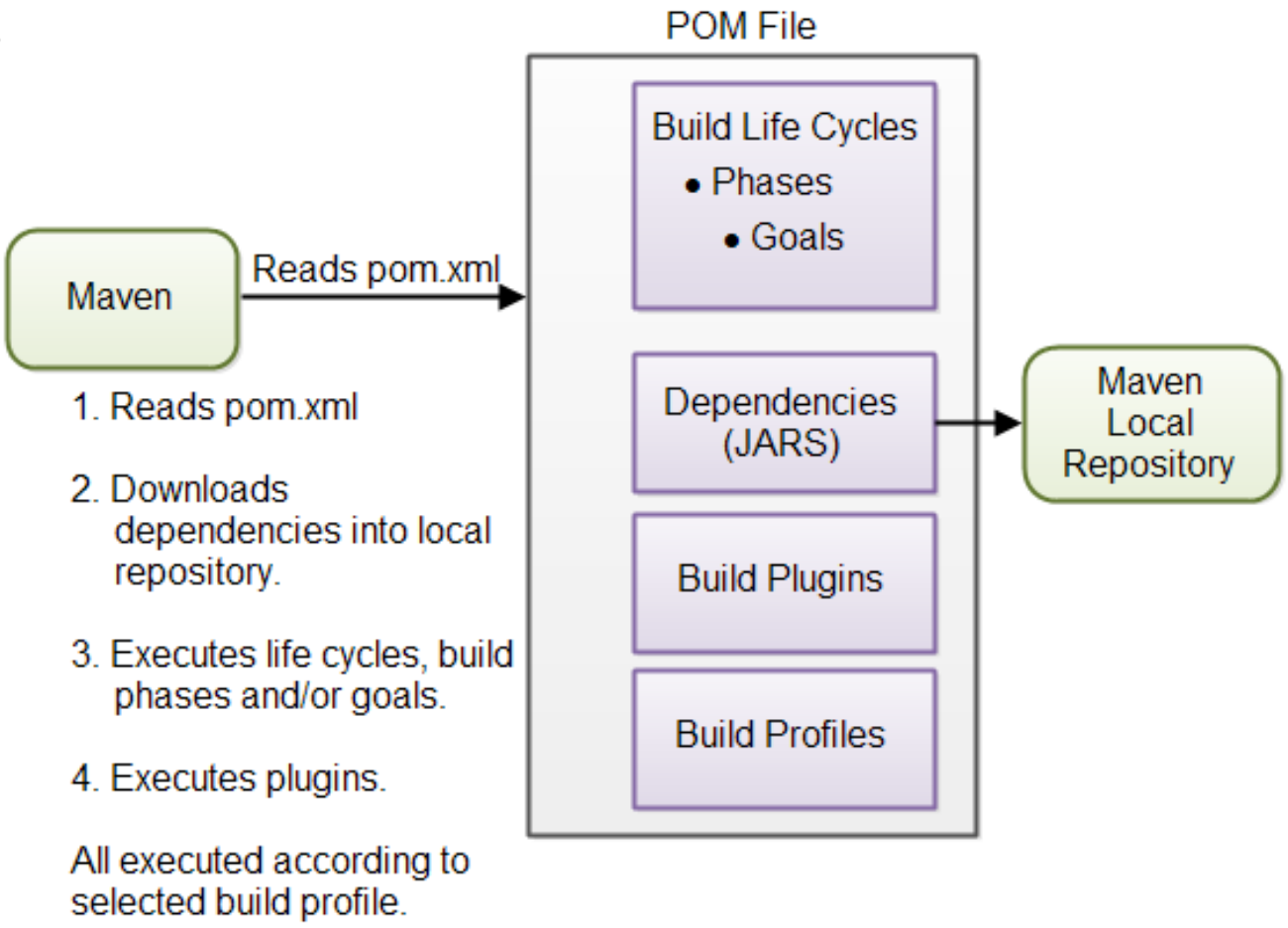
CI build eszközök – Apache Maven

- **Fat JAR**
  - Maven hozza létre
  - minden függőséget tartalmazó jar fájl
  - `pom.xml`-ben konfigurálható
    - `<build>` → `<plugins>` → `<configuration>`

  - Target mappában jön létre:



CI build eszközök – Apache Maven



Forrás: https://jenkov.com/images/maven/maven-overview-1.png

CI build eszközök – Apache Maven

- **Maven összefoglalás**
  - 3 fő dolgot biztosít
    - **Projektstruktúra, konfiguráció**
      - fix könyvtárszerkezet
    - **Függőségkezelés**
      - pom.xml `<dependency>`
    - **Build- és teszt automatizálás**
      - Maven életciklus

CI eszközök – Docker

- **Konténerizációs platform**, amely lehetővé teszi az **alkalmazások és függőségeik csomagolását**, szállítását és futtatását konténerekben
- Olyan, mintha egy mini **virtuális gép** lenne, amelyben az alkalmazás fut, de **függetlenül** attól, hogy **milyen az operációs rendszer**

- **Ha egyszer működik, mindenhol működik**
  - biztosítja, hogy az alkalmazás ugyanúgy működjön a fejlesztői gépen, tesztkörnyezetben és éles szerveren

- **Konténerizáció → Könnyebb telepítés és skálázás**
  - egy alkalmazás és annak összes függősége egyetlen konténerbe kerül, így könnyen áthelyezhető más gépekre vagy szerverekre

CI eszközök – Docker

- **Gyors és erőforrás-hatékony**
  - a konténerek kisebbek és gyorsabbak, mint a hagyományos virtuális gépek, mert nincs szükség külön operációs rendszerre minden egyes példányhoz
- **Mikroszolgáltatások támogatása**
  - több kis szolgáltatás külön konténerben futhat
  - pl. egy konténer a webalkalmazásnak, egy konténer az adatbázisnak

CI eszközök – Docker

- **Alapfogalmak**
  - **Docker Image**
    - egy alkalmazás és annak környezete előre elkészített csomagban
  - **Docker Container**
    - egy futó példánya egy Docker Image-nek
  - **Docker file**
    - egy fájl, amely meghatározza, hogyan kell egy Docker Image-et létrehozni
  - **Docker Hub**
    - egy nyilvános tárhely, ahonnan előre elkészített Docker Image-eket lehet letölteni

CI eszközök – Docker

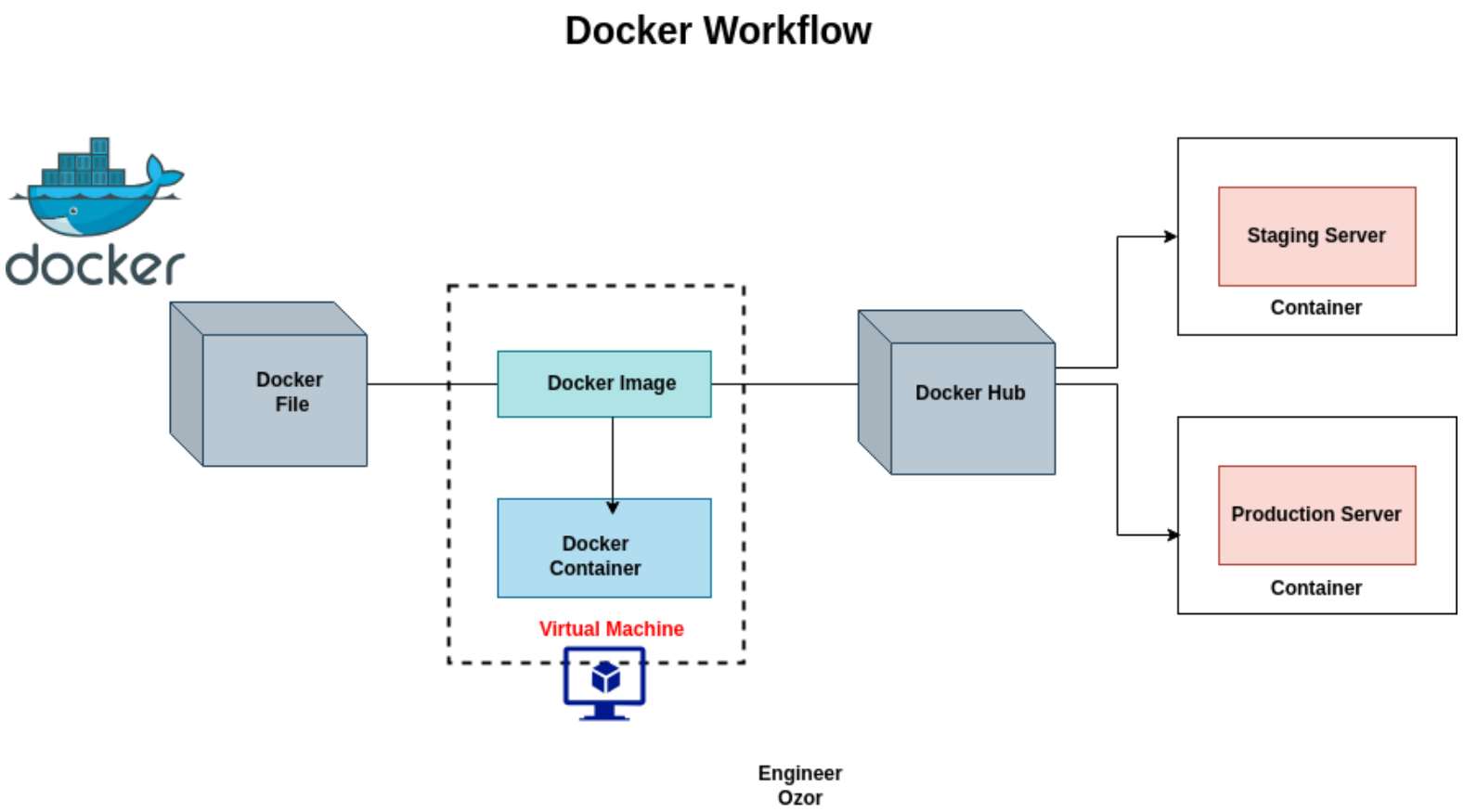
- **Példa (node.js)**

```
# Alap image (Node.js)
FROM node:18

# Munkakönyvtár beállítása
WORKDIR /app

# Csomagok bemásolása és telepítése
COPY package.json ./
RUN npm install

# Az alkalmazás fájljainak bemásolása
COPY . .

# Az alkalmazás futtatása
CMD ["node", "app.js"]

# Port megnyitása
EXPOSE 3000
```

```
docker build -t my-node-app .
```

```
docker run -p 3000:3000 my-node-app
```

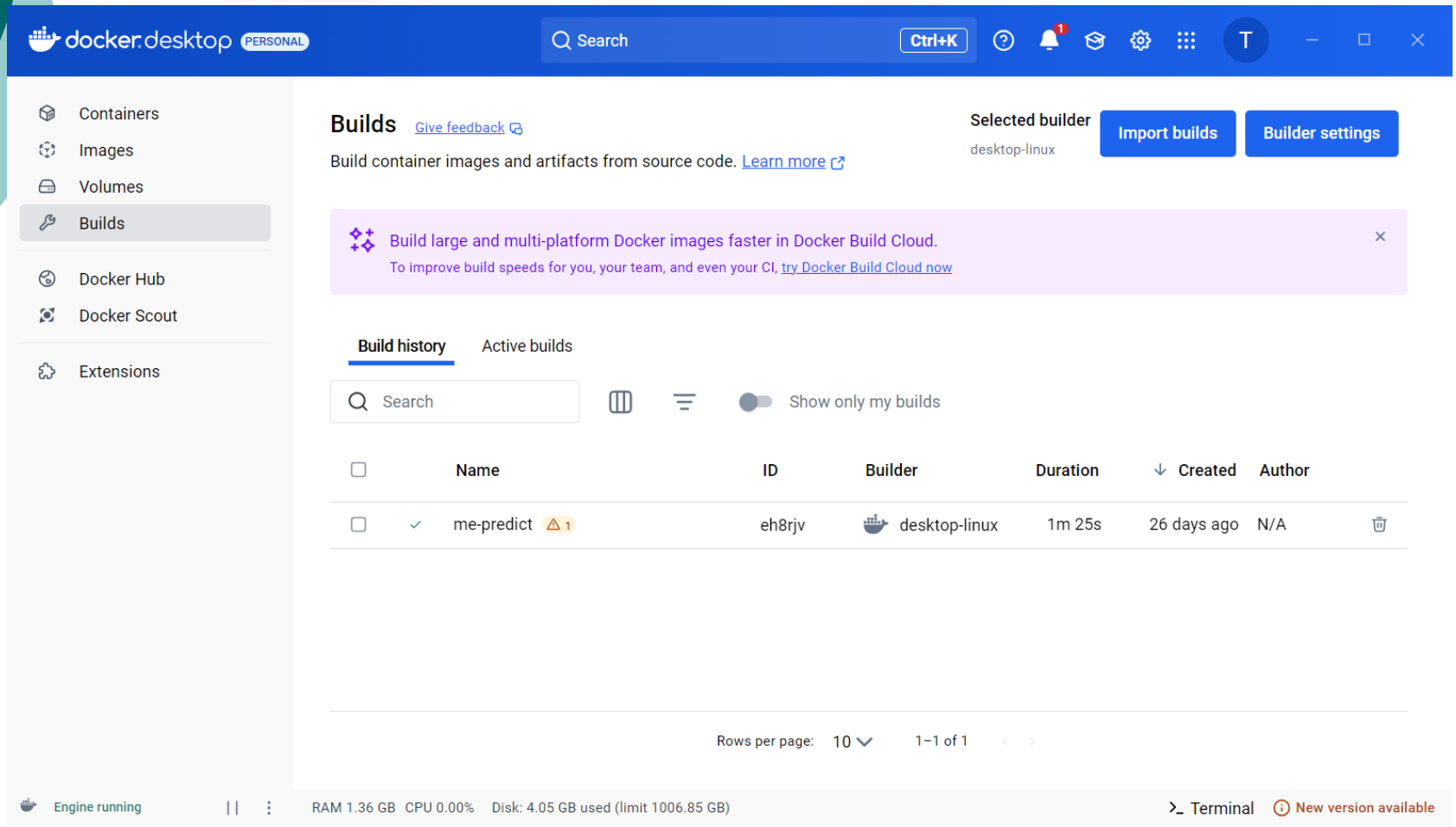
CI eszközök – Docker



Forrás: https://medium.com/@augustineozor/docker-workflow-b9fe71d32184

CI eszközök – Docker

- **Docker Desktop – Windows tool**



CI eszközök – JIRA

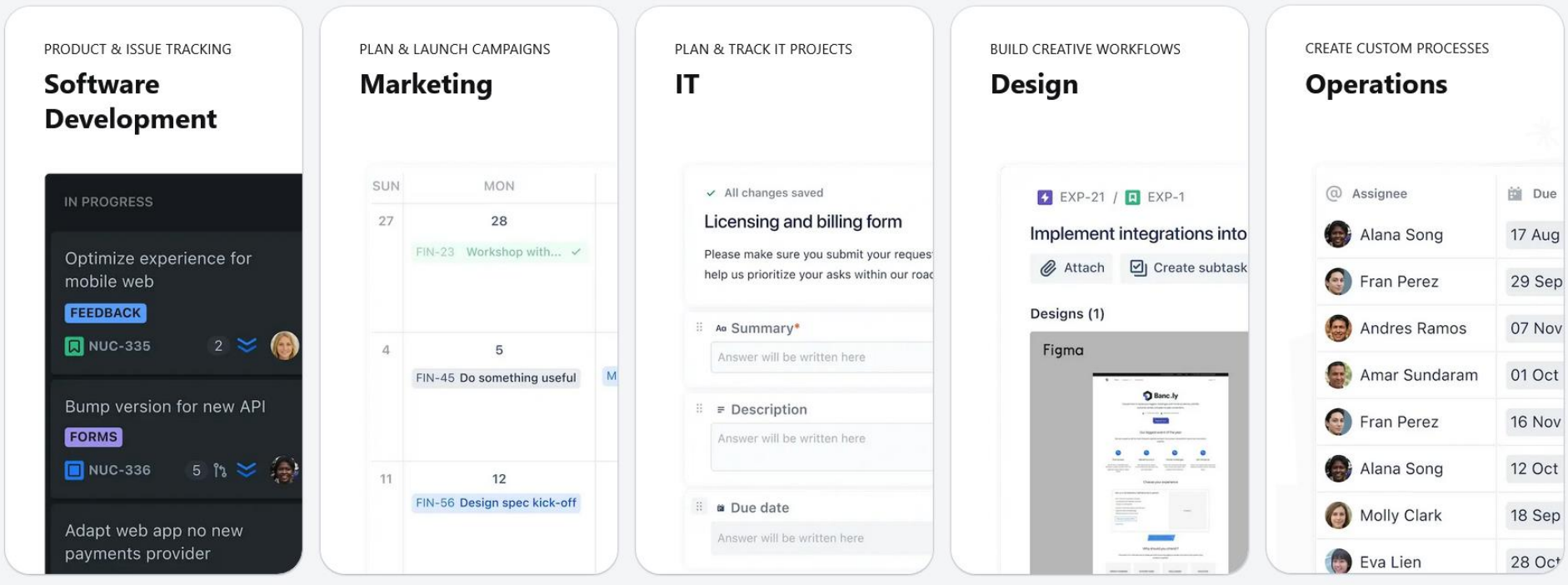
- **Projektmenedzsment eszköz**, amelyet az Atlassian fejlesztett ki
- Elsősorban agilis csapatok számára készült, hogy **segítsen a munkafolyamatok tervezésében**, nyomon követésében és kezelésében
- https://www.atlassian.com/software/jira

- Funkciók:
  - **Feladatkezelés:** lehetővé teszi a feladatok létrehozását, hozzárendelését, priorizálását és nyomon követését
  - **Agilis eszközök:** Támogatja a Scrum és Kanban táblákat, sprint tervezést, backlog kezelést és burndown chartokat

CI eszközök – JIRA

- **Integrációk:** Számos más eszközzel integrálható, mint például a Confluence, Bitbucket, és különböző CI/CD rendszerek
- **Testreszabhatóság:** testreszabható a csapat igényei szerint, beleértve a munkafolyamatokat, mezőket és jelentéseket
- **Jelentések és elemzések:** Részletes jelentéseket és elemzéseket biztosít a projekt előrehaladásáról és a csapat teljesítményéről

CI eszközök – JIRA

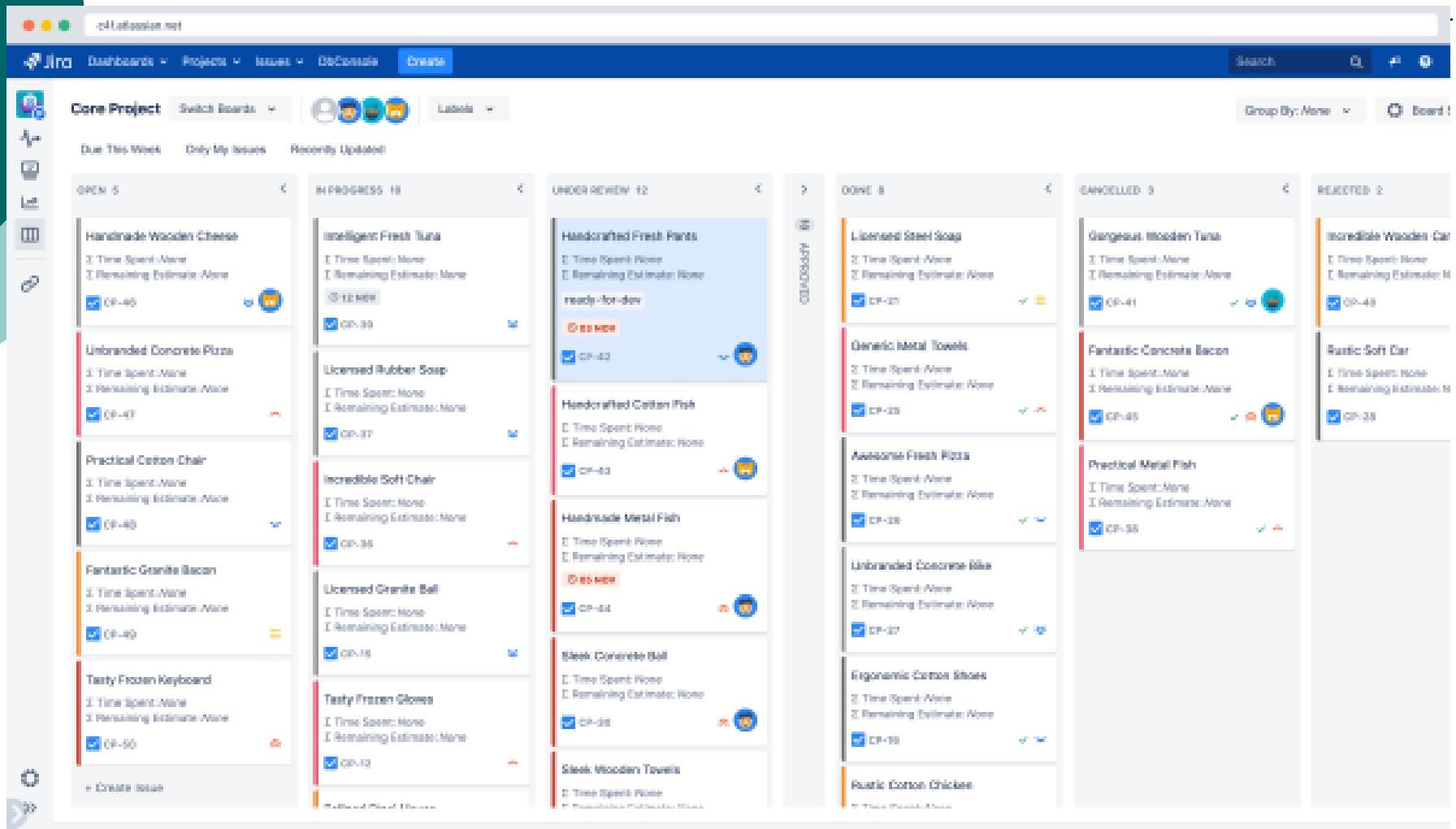


Forrás: https://www.atlassian.com/software/jira

CI eszközök – JIRA

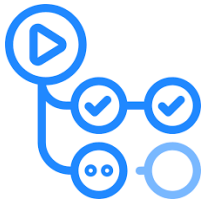
Forrás: https://www.atlassian.com/software/jira

CI eszközök – JIRA



Forrás: https://marketplace.atlassian.com/apps/1222860/work-management-for-jira?tab=overview&hosting=datacenter

CI eszközök – Github actions



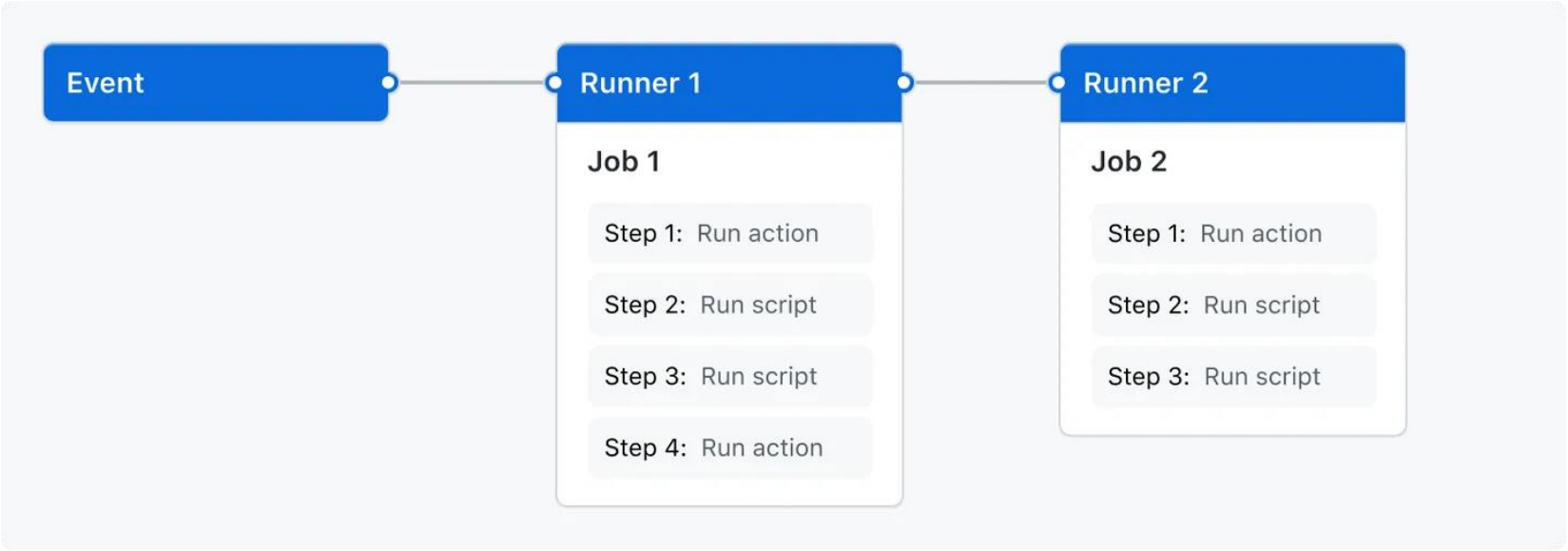
- **„Automate, customize, and execute your software development workflows right in your repository with GitHub Actions. You can discover, create, and share actions to perform any job you'd like, including CI/CD, and combine actions in a completely customized workflow”**

https://docs.github.com/en/actions

- GitHub Actions egy CI/CD platform, amely lehetővé teszi a build, tesztelési és telepítési folyamatok automatizálását közvetlenül a GitHub repositoryk-ban

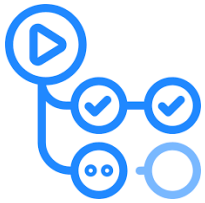
CI eszközök – Github actions

- Olyan munkafolyamatokat létrehozását teszi lehetővé, amelyek automatikusan lefutnak, bizonyos események bekövetkezésekor a repository-ban
  - pl. egy pull request létrehozása vagy egy commit pusholása esetében



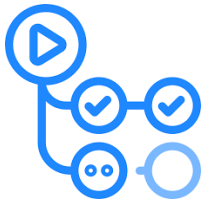
Forrás: https://docs.github.com/en/actions/about-github-actions/understanding-github-actions

CI eszközök – Github actions



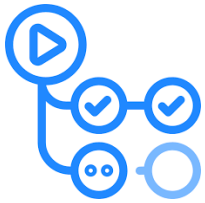
- **Funkciók:**
  - **Automatizálás:** lehetővé teszi a szoftverfejlesztési folyamatok automatizálását, beleértve a build-elést, tesztelést és telepítést
  - **CI/CD:** támogatja a folyamatos integrációt és folyamatos szállítást, így a kódváltozások gyorsan és megbízhatóan kerülhetnek a produkciós környezetbe
  - **Testreszabhatóság:** teljesen testreszabható munkafolyamatokat lehet létrehozni, amelyek különböző feladatokat végeznek el, például címkék hozzáadása új issue-khoz vagy alkalmazások telepítése minden egyes release után

CI eszközök – Github actions



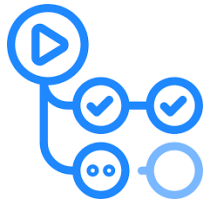
- **Funkciók:**
  - **Integráció:** számos más eszközzel és szolgáltatással integrálható, mint például a Docker, AWS, Azure stb.
  - **Könnyű használat:** a munkafolyamatokat YAML fájlokban definiálja, amelyek a repository .github/workflows könyvtárában találhatók
    - YAML: Yet Another Markup Language (vagy Ain't Markup Language), egy adat-szerializációs formátum, amely alkalmas konfigurációs fájlok létrehozására és adatcsere megvalósítására
      - egyszerű szintaxis, hierarchikus struktúra, adattípusok támogatása, sok támogatott platform/technológia

CI eszközök – Github actions



- **Definíciók:**
  - **Munkafolyamatok (Workflows):** egy konfigurálható automatizált folyamat, amely egy vagy több feladatot (job) futtat. A munkafolyamatok YAML fájlokban vannak definiálva, és különböző események hatására futnak, például egy pull request létrehozása vagy egy commit push-olása
  - **Események (Events):** olyan tevékenységek a repository-ban, amelyek kiváltják a munkafolyamat futását. Például egy pull request létrehozása vagy egy issue megnyitása
  - **Feladatok (Jobs):** egy lépéssorozat egy munkafolyamatban, amely ugyanazon a runner-en fut. Minden lépés egy shell script vagy egy action futtatása

CI eszközök – Github actions



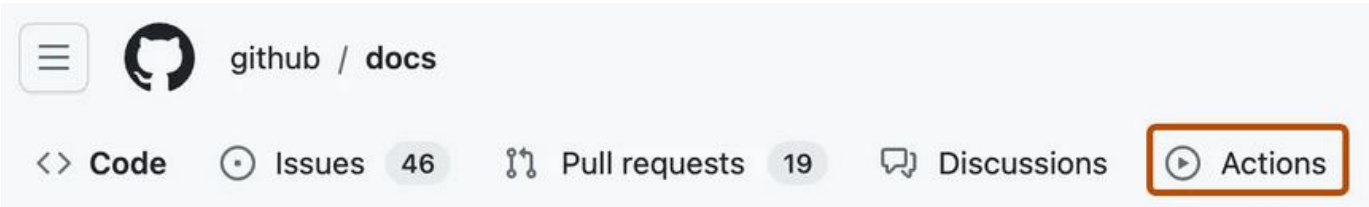
- **Példa:**

```
name: CI

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Set up Node.js
      uses: actions/setup-node@v2
      with:
        node-version: '14'
    - name: Install dependencies
      run: npm install
    - name: Run tests
      run: npm test
```

- Automatikusan lefut, amikor egy commit push-olásra kerül a repository-ba
  - ellenőrzi a kódot, telepíti a függőségeket és futtatja a teszteket

CI eszközök – Github actions

- **Példa Maven-el:**



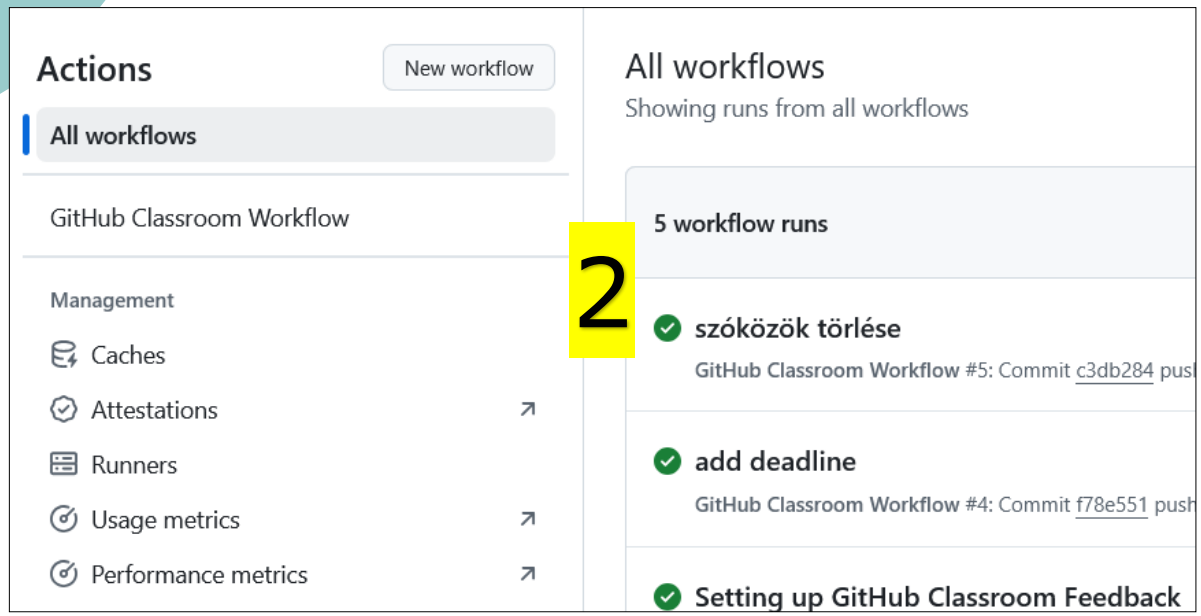

```
steps:
  - uses: actions/checkout@v4
  - uses: actions/setup-java@v4
    with:
      java-version: '17'
      distribution: 'temurin'
  - run: mvn --batch-mode --update-
snapshots verify
  - run: mkdir staging && cp
target/*.jar staging
  - uses: actions/upload-artifact@v4
    with:
      name: Package
      path: staging
```

- New workflow
- "Java with Maven"
- Configure
- Commit changes
- maven.yml fájl .github/workflows jegyzékben

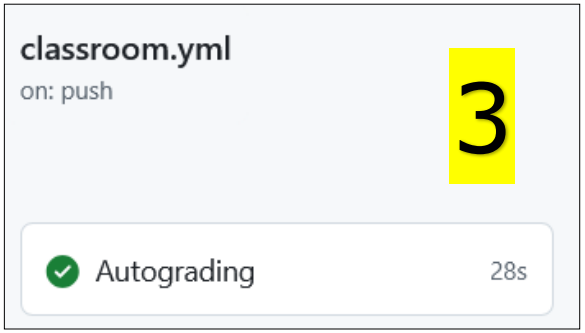
- automatikusan letölti a repository-t
- beállítja a Java környezetet
- futtatja a Maven teszteket
- csomagolja a build eredményét
- majd feltölti azt egy artifactként

CI eszközök – Github actions

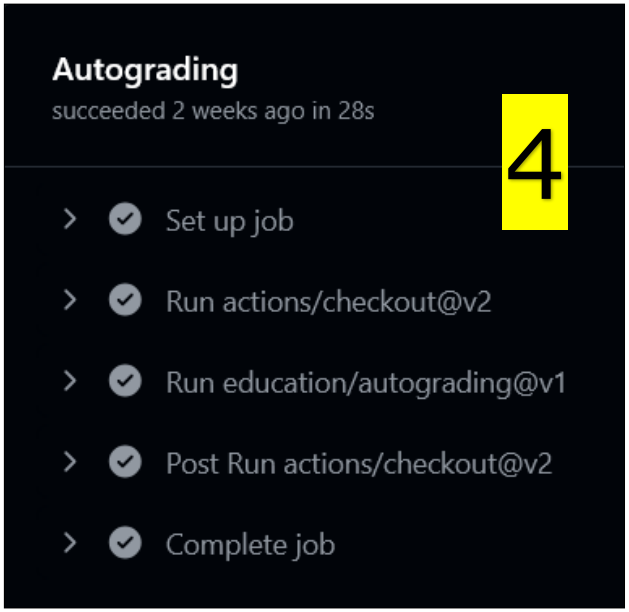
1



2



3



4

Köszönöm a figyelmet!