

Instrumente de build



George Popa
Gemini Solutions

Instrumente de build

Build automation este procesul de creare automată a componentelor software pornind de la codul sursă și dependențe.

Fazele uzuale ale procesului de build:

- Instalare dependențe
- Compilare cod sursă în cod binar
- Împachetare cod binar
- Rulare aplicație
- Rulare teste automate
- Revenire la starea inițială

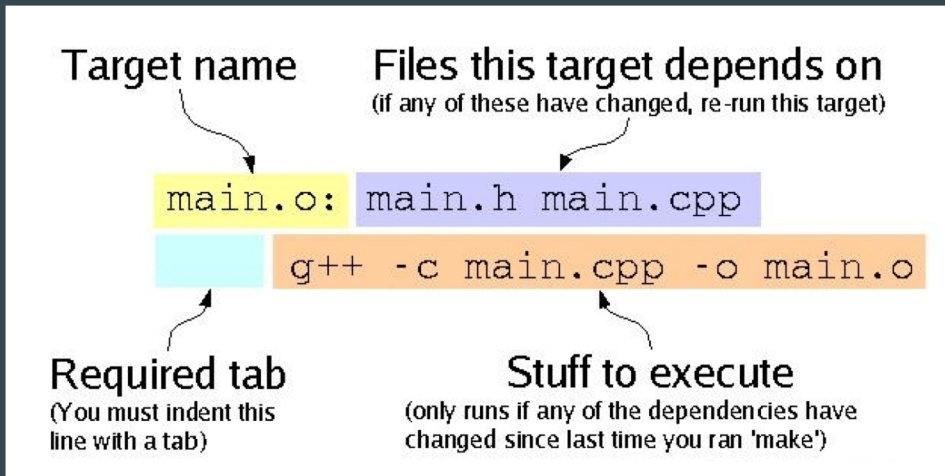
Instrumente de build

- ❖ GNU Make - Utilizat pe sisteme UNIX, de regulă pentru limbajele C/C++
 - <https://www.gnu.org/software/make>
- ❖ Apache Ant - Instrument de build Java
 - <http://ant.apache.org/>
- ❖ Apache Maven - Instrument de build, raportare și documentare a proiectului, utilizând plug-ins.
 - <https://maven.apache.org/>
- ❖ Gradle - Instrument de build cu spectru larg, utilizat în general pentru proiecte Java & Scala, C/C++ și Android (default build tool).
 - <http://gradle.org/>



GNU Make

- Instrument de automatizare a build-ului pentru aplicații C/C++
- Utilitarul `make` citește fișiere de tip **Makefile** pentru execuție **rules**
- Fiecare **rule** conține un **target**, o lista de dependențe și o secvență de comenzi de execuție:



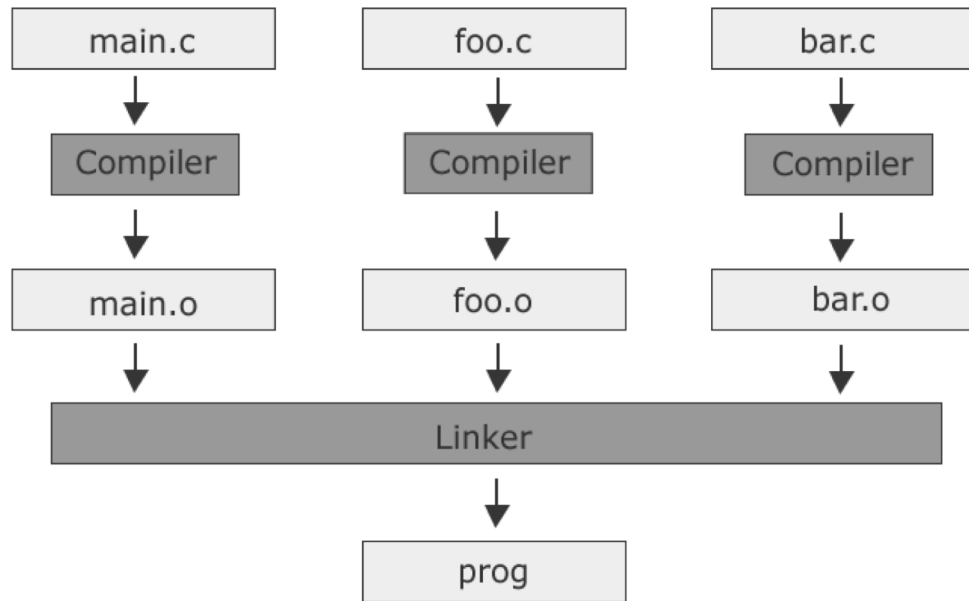
Makefile

```
prog: main.o foo.o bar.o
    gcc -o prog main.o foo.o bar.o

main.o: main.c
    gcc -c main.c

foo.o: foo.c
    gcc -c foo.c

bar.o: bar.c
    gcc -c bar.c
```



Makefile - calculator

```
all: compile run
```

```
compile:
```

```
    gcc -Wall -o calculator calculator.c add.c subtract.c
```

```
run:
```

```
    ./calculator
```

```
clean:
```

```
    rm -f calculator
```

- ❖ make compile
- ❖ make run
- ❖ make clean
- ❖ make

Gradle – Open Source Build Automation

- Construit utilizând conceptele Apache Ant și Apache Maven
- Folosește limbajul Groovy sau Kotlin, înlocuind fișierele XML (build.xml, pom.xml) - cu scopul de a ușura utilizatorilor umani înțelegerea build.gradle
- Compatibilitate cu fișierele Ant build.xml
 - `ant.importBuild 'build.xml'`
 - `gradle ant.target`

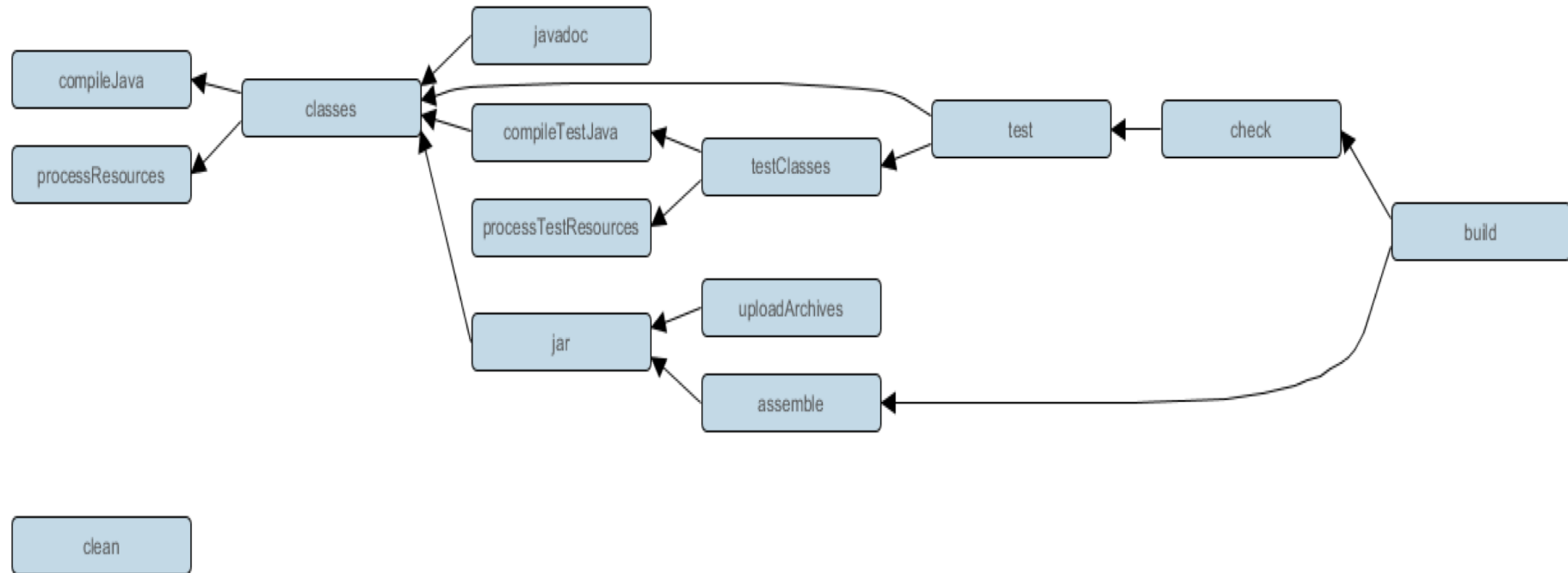
Gradle – Open Source Build Automation

Task-urile uzuale sunt deja definite in plug-ins:

- ❖ *Language plugins* (Java, Groovy, Scala, Antrl) - executate în JVM
- ❖ *Incubating language plugins* (Assembler, C, C++, Objective-C, Windows-resources)
- ❖ *Integration plugins* (application, ear, jetty, maven, war)
- ❖ *IDE plugins* (eclipse, idea, sonar)
- ❖ *Third party plugins*

Documentație: <http://gradle.org/getting-started-gradle-java/>

Gradle lifecycle



- `gradle -q viewLifecycle`

Let's have fun!

Gradle build automation tool

- gradle init
- gradle init --type java-library
- gradle tasks
- gradle clean build
- apply plugin: 'java'
- apply plugin: 'maven'
- gradle clean install

V. <https://guides.gradle.org/creating-new-gradle-builds/>
sau <https://www.jetbrains.com/help/idea/gradle.html>