UNIVERSITÀ DEGLI STUDI
DI MILANO

Ingegneria del software

Lezione 6

Git Workflow

Come si usano i branch?

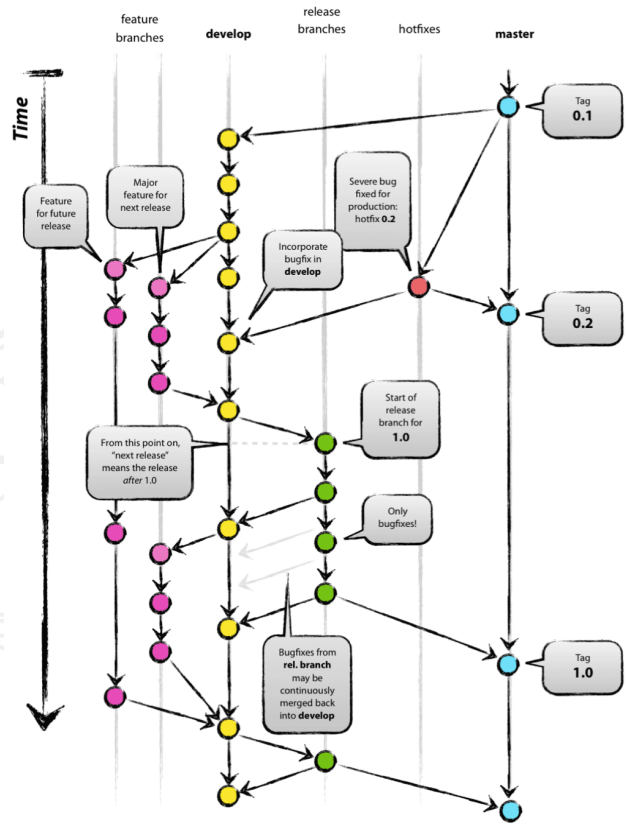
Praticamente impossibile non usarli!

- libertà completa
- seguendo delle linee guida
- all'interno di un workflow più grande

GitFlow [AVH]

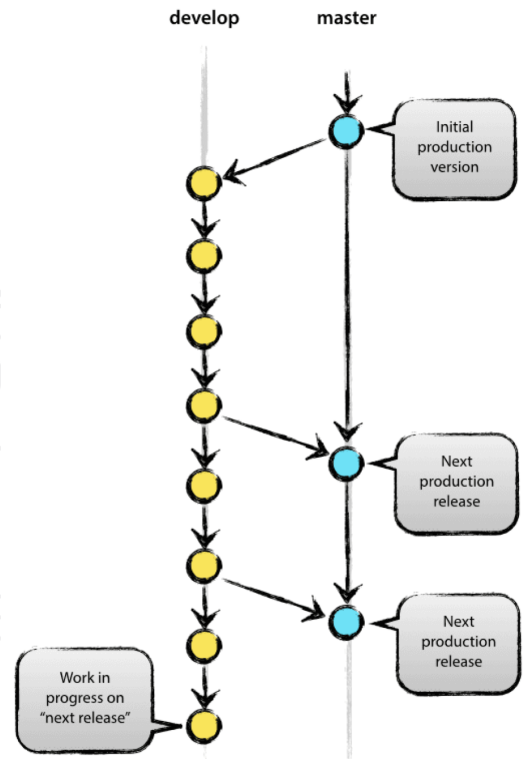
http://nvie.com/posts/a-successful-git-branching-model/

- Differenziazione tra tipi di branch
- nuove operazioni “guidate”
- non visibile nella figura, ma tratta anche i remote



master e develop

- Sono due rami con “vita infinita”
- **master** (e in particolare origin/master) contiene le versioni “stabili e pronte alla consegna”
- **develop** è il ramo di integrazione



https://github.com/petervanderdoes/gitflow-avh/blob/develop/git-flow-feature

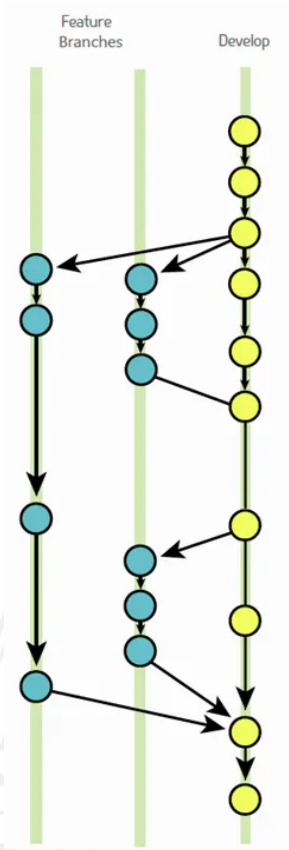
feature

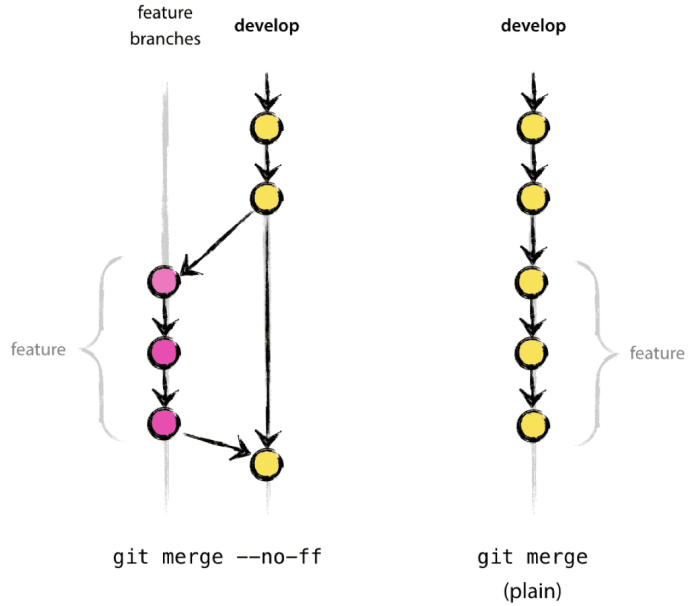
- git flow feature start *feat1*

```
git checkout develop
git branch feat1
git checkout feat1
```

- git flow feature finish *feat1*

```
git checkout develop
git merge --no-ff feat1
git branch -d feat1
```





https://github.com/petervanderdoes/gitflow-avh/blob/develop/git-flow-release

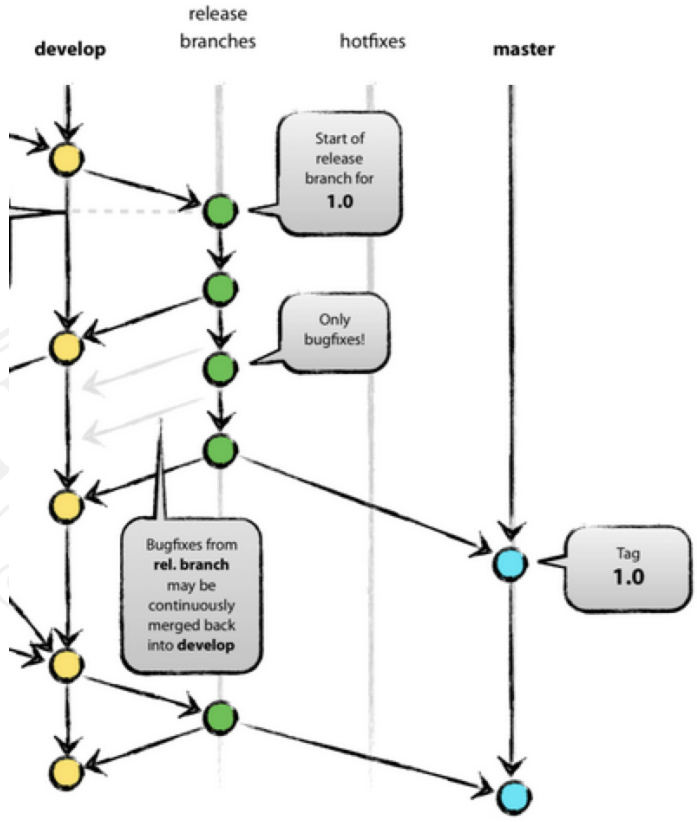
release

- git flow release start *ver*

```
git checkout -b ver develop
```

- git flow release finish *ver*

```
git checkout master
git merge --no-ff release-ver
git tag -a ver
git checkout develop
git merge --no-ff release-ver
git branch -d release-ver
```



hotfix

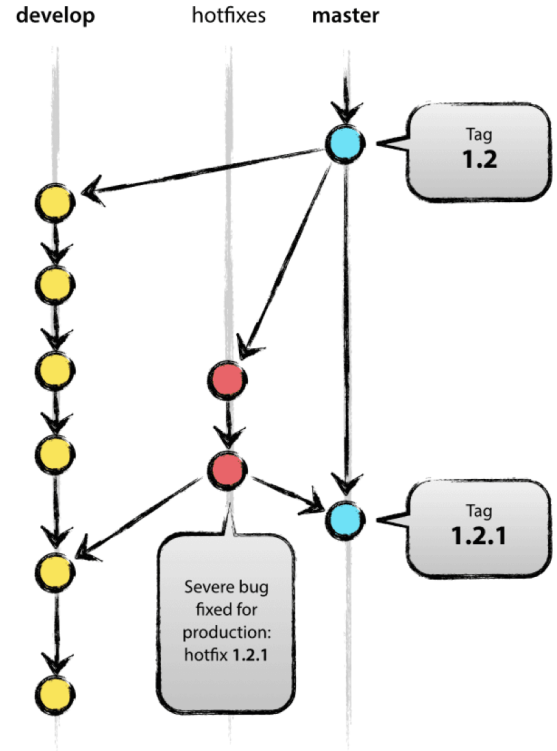
Riparazione veloce di difetti urgenti senza aspettare prossima release

- git flow hotfix start *ver*

```
git checkout -b ver master
```

- git flow hotfix finish *ver*

```
git checkout master
git merge --no-ff ver
git tag -a ver
git checkout develop    # uhm ???
git merge --no-ff ver
git branch -d ver
```



Limiti di git

- singolo livello di autorizzazione
- nessun livello di review

git request-pull

- git-request-pull - Generates a summary of pending changes

```
git request-pull  <start> <url> [<end>]
```

- Inizialmente git non era pensato per avere un ambiente di hosting centralizzato…
- l'interazione era stata pensata molto più *peer to peer*… o tramite vecchi canali quali ad esempio la posta.

Esempio

```
git request-pull  5a548dc git@bitbucket.org:emanuelemazzola/p2-lab05-2019.git  consegna_16:master
The following changes since commit 5a548dc21910dd1e7466323812282909c0e6fd2a:

  start (2019-11-13 08:22:34 +0100)

are available in the Git repository at:

  git@bitbucket.org:emanuelemazzola/p2-lab05-2019.git master

for you to fetch changes up to 7c2095152be13b030bb42804acf7d8bc7988b6ed:

  Risoluzione di alcuni errori: Codizione per la quale il mazziere gioca o meno (almeno un giocatore non deve aver sballato), errata condizione di gioco nella classe mazziere

----------------------------------------------------------------
Emanuele Nazzola (1):
      README.md edited

emanuele.mazzola (5):
      Prima versione
      Prima versione tutti i file
      Seconda versione Distribuzione delle carte iniziali. Modifiche visualizzazione punteggi
      Nome errato
      Risoluzione di alcuni errori: Codizione per la quale il mazziere gioca o meno (almeno un giocatore non deve aver sballato), errata condizione di gioco nella classe mazz

 .idea/gradle.xml                                          |  1 +
 README.md                                                 |  4 ++--
 src/main/java/it/unimi/di/prog2/blackjack/BlackJack.java  | 29 ++++++++++++++++++++++++++--
 src/main/java/it/unimi/di/prog2/blackjack/GiocatoreBJ.java |  3 ++-
 src/main/java/it/unimi/di/prog2/blackjack/MaggioreStrategy.java | 24 ++++++++++++++++++++++
 src/main/java/it/unimi/di/prog2/blackjack/Mazziere.java   | 23 ++++++++++++++++++++-
 src/main/java/it/unimi/di/prog2/blackjack/MinoreStrategy.java | 23 ++++++++++++++++++++++
 src/main/java/it/unimi/di/prog2/blackjack/MultiMazzo.java | 14 ++++++++++++-
 src/main/java/it/unimi/di/prog2/blackjack/RandomStrategy.java |  2 ++
 src/main/java/it/unimi/di/prog2/blackjack/RischioStrategy.java | 27 +++++++++++++++++++++++++++
 src/main/java/it/unimi/di/prog2/blackjack/Sfidante.java   | 34 ++++++++++++++++++++++++++++++++---
 src/main/java/it/unimi/di/prog2/blackjack/Strategia.java  |  6 ++++++
 12 files changed, 180 insertions(+), 10 deletions(-)
 create mode 100644 src/main/java/it/unimi/di/prog2/blackjack/MaggioreStrategy.java
 create mode 100644 src/main/java/it/unimi/di/prog2/blackjack/MinoreStrategy.java
 create mode 100644 src/main/java/it/unimi/di/prog2/blackjack/RischioStrategy.java
```

Progetti Open Source

- Soffrono in particolar modo per questi limiti
- Gli ambienti di hosting hanno cercato soluzioni alternative:
  - inventandosi nuovi meccanismi
  - provando a “imporre” nuovi workflow (GitHub Flow, GitLab Flow, etc)

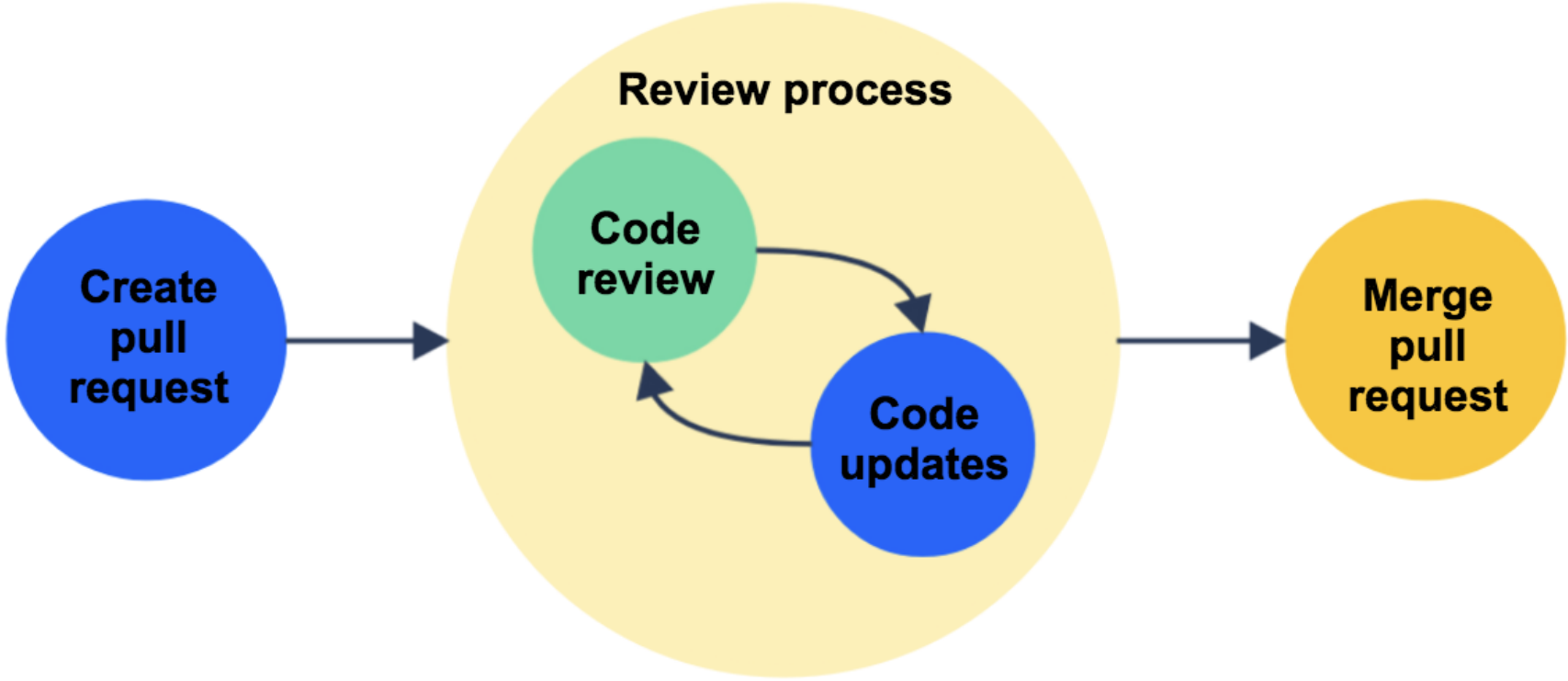
Fork

- Risolve un primo problema di autorizzazioni
  - permette di mantenere legami tra repository su sito di hosting ma con owner e autorizzazioni diversi
- Ottimizzazioni
  - condivisione dello spazio degli oggetti

https://github.com/torvalds/linux/blob/b4061a10fc29010a610ff2b5b20160d73samsung.c

https://github.com/torvalds/linux/tree/8bcab0346d4fcf21b97046eb44db8cf37

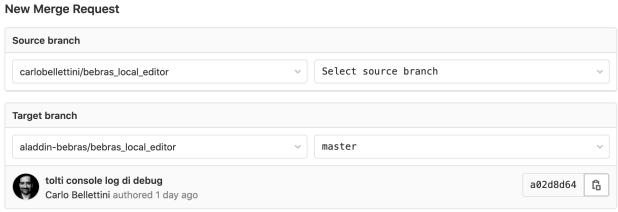
Tra creazione e deploy va prevista una fase di review

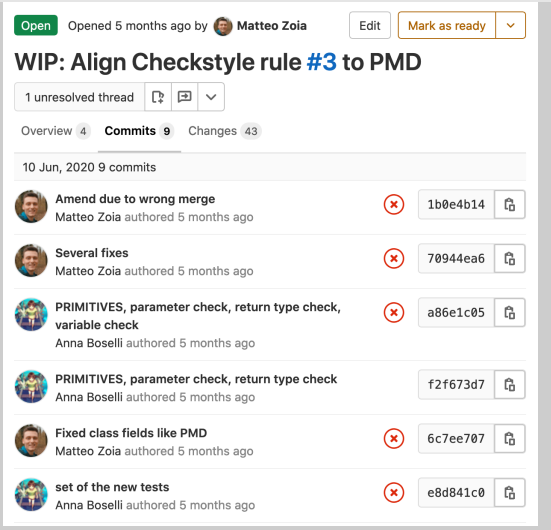


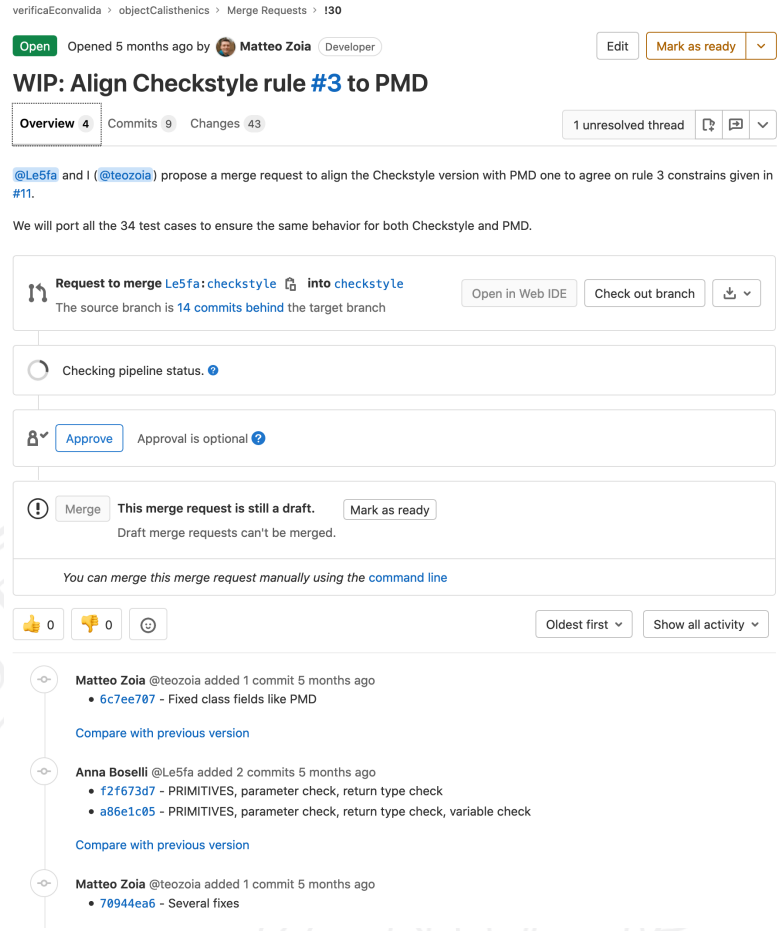
(fork &)Pull requests

https://help.github.com/articles/what-is-a-good-git-workflow/

Permette di gestire interazioni lasche tra sviluppatori mediate dal sito di hosting



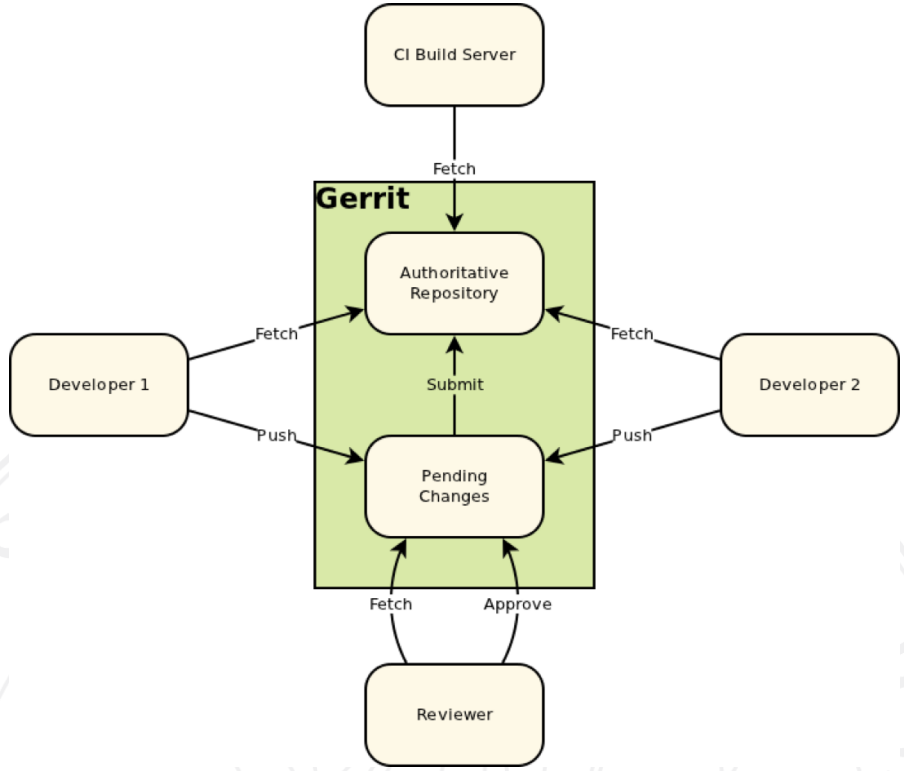




Gerrit

https://www.gerritcodereview.com/about.html
https://gerrit-review.googlesource.com/Documentation/
https://source.android.com/docs/setup/contribute/submit-patches

- progetto di Google
  - AOSP
- peer review delle sottomissioni
  - In grandi progetti non si può pensare che ci sia una persona singola a decidere…
  - Gerrit è al tempo stesso un server Git (due server git) e un sito web di peer review.



Verifier: Verifying a change

- If you are assigned to be the Verifier for a change, you need to do the following:
  - Patch the change into your local client using one of the Download commands.
  - Build and test the change.

Within Gerrit use Publish Comments to mark the commit as "Verified" or "Fails," and add a message explaining what problems were identified.

Approver: Reviewing a change

If you are assigned to be the Approver for a change, you need to determine the following:

- Does this change fit within this project's stated purpose?
- Is this change valid within the project's existing architecture?
- Does this change introduce design flaws that will cause problems in the future?
- Does this change follow the best practices that have been established for this project?
- Is this change a good way to perform the described function?
- Does this change introduce any security or instability risks?

If you approve of the change, mark it with LGTM ("Looks Good to Me") within Gerrit.

Build automation

- Come proteggersi da checkin di una versione non funzionante?
  - Automatizzando
    - la ricompilazione
    - il testing
- Molti tool disponibili
  - make
  - Ant
  - Gradle

Make

- Comandi di shell
- dipendenza (tra file)

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

```
CC=gcc
CFLAGS=-I.

%.o: %.c $(DEPS)
  $(CC) -c -o $@ $< $(CFLAGS)

hellomake: hellomake.o hellofunc.o
	$(CC) -o hellomake hellomake.o hellofunc.o -I.
```

generazione dei makefile

- A monte della costruzione, c'è la definizione di come farlo
  - differenze per le varie macchine su cui si fa deployment
    - hanno la tale libreria? che versione?
    - hanno la tale funzione?
    - hanno…
- Automake, autoconf
- imake

http://jakarta.apache.org/ant/index.html

Ant

- Ant nasce per supportare il progetto Tomcat
- Quando Tomcat entra in Jakarta viene capita l'utilità di Ant e viene reso indipendente (Luglio 2000)
- È scritto in Java per progetti Java
- Supporta CVS, Junit, FTP, JavaDOCS, JAR, etc…
  - Non solo compilazione, ma test e deploy

Concetti base di Ant

- Ogni progetto contiene più target :
  - compiling
  - deploying
- I target possono avere dipendenze da altri
- Targets contengono i task che fanno effettivamente il lavoro Si possono aggiungere nuovi tipi di task definendo (o trovando in giro) nuove classi Java

Un piccolo build file

```xml
<?xml version="1.0"?>
<project name="Hello" default="compile">
    <target name="clean" description="remove intermediate files">
        <delete dir="classes"/>
    </target>
    <target name="clobber" depends="clean" description="remove all artifact files">
        <delete file="hello.jar"/>
    </target>
    <target name="compile" description="compile the Java source code to class files">
        <mkdir dir="classes"/>
        <javac srcdir="." destdir="classes"/>
    </target>
    <target name="jar" depends="compile" description="create a Jar file for the application">
        <jar destfile="hello.jar">
            <fileset dir="classes" includes="**/*.class"/>
            <manifest>
                <attribute name="Main-Class" value="HelloProgram"/>
            </manifest>
        </jar>
    </target>
</project>
```

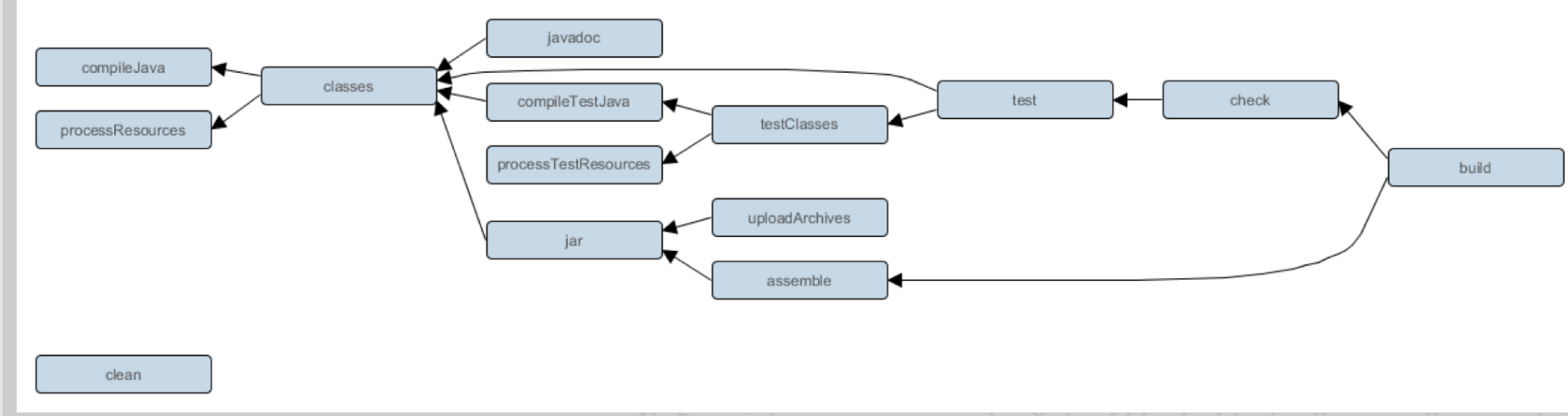
Gradle

- Usa linguaggi Groovy (o Kotlin)
- Approccio dichiarativo e fortemente basato su convenzioni (*build-by-convention*)
- Definisce un linguaggio specifico per le dipendenze
- Multi-project builds
- Estendibile tramite plugin

plugin Java

- Definisce
  - sourceSet
  - task

Directory	Meaning
src/main/java	Production Java source
src/main/resources	Production resources
src/test/java	Test Java source
src/test/resources	Test resources



Altri plugin

- Application (task: run, startScripts)
- FindBugs (http://findbugs.sourceforge.net/)
- PMD (http://pmd.sourceforge.net/)
- checkstyle (http://checkstyle.sourceforge.net/index.html)
- jacoco (http://www.eclemma.org/jacoco/)
- eclipse
- idea

Bug tracking

- Tiene traccia e gestisce tutte le segnalazioni sui difetti di un software
  - Un database dei bug
  - Un mezzo di comunicazione per la segnalazione degli stessi
  - È uno strumento di assegnazione compiti
  - Alcuni tool:
    - BugZilla - http://www.mozilla.org/bugs
    - Scarab - http://scarab.tigris.org/
    - GNATS - http://www.gnu.org/software/gnats
    - BugManager – http://www.bitmover.com
    - Mantis - https://www.mantisbt.org/

Bug workflow