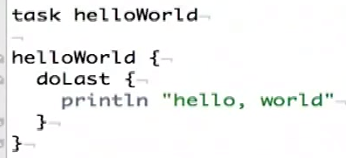
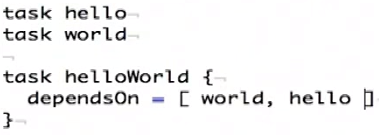
# Approach

declarative  
imperative

pro  
-> less code  
con  
-> conventions -> strong opinion  
-> adaptability  
  
gradle  
-> base convention  
-> own process  
-> tooling -> eclipse  
  
tasks  
-> basic unit  
-> object

# Runtime

gradle helloWorld -> imperative  
-> 1) declare, 2) configure  
-> closure -> “doLast” auto-call method

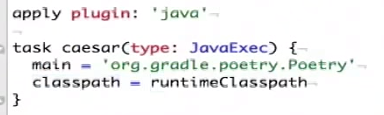
dependencies  
-> dependsOn -> built-in method -> imperative -> small

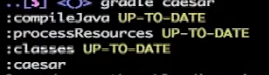
# plugin

-> standard approach, e.g. for building java -> run it



1) inherit from parent type  
2) specify main  
3) cp auto-derived from gradle-compiled classes

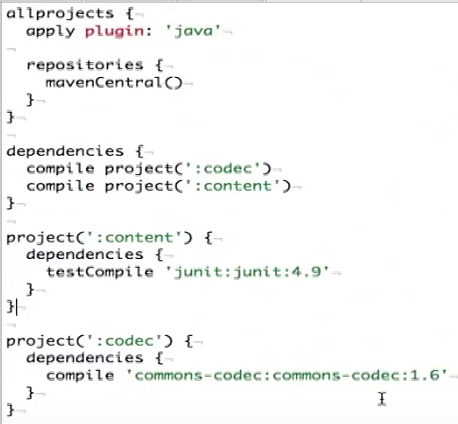


4) if up-to-date -> no re-run -> time saved

5) just in case

6) expose declarative format and access it in imperative sense

7) multi-project







**长号**

# **Git**

git remote add origin https…  
git push -u origin master

remote “origin”, default local branch “master”  
-u tells Git -> remember parameters -> simply run git push

synch:  
git pull origin master  
git diff HEAD  
git reset -> use to undo “add” (before commit)  
git checkout -> get rid of all changes  
git rm -> delete from repository

fetching pushed commits -> git pull  
1) git fetch  
2) (git creates branch “origin/master”)  
3) git merge origin/master  
git add & commit & push -> “merge commit”

delete remote branch  
git push origin :nam

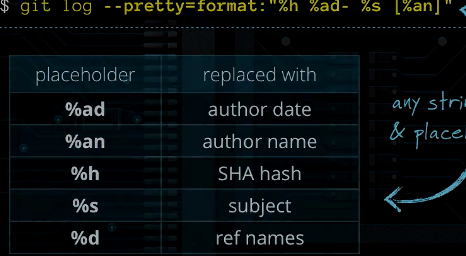
show locally available branches  
git remote show origin  
clean up all local branches, that have been deleted on central repo  
git remote prune origin

heroku -> auto-deploy -> deploys everything, that is pushed to it …

## Rebase

merge master changes with local changes without merge commits!!  
-> no log of the merges (shorter, but a bit dirty)

git fetch -> get latest without merging  
move your master (i.e. “local”/master) commits after the commits from origin/master  
git rebase

Log Format

Exclude & Ignore -> .gitignore file  
-rm --cached -> remote from repository, but not locally