Many projects, one code

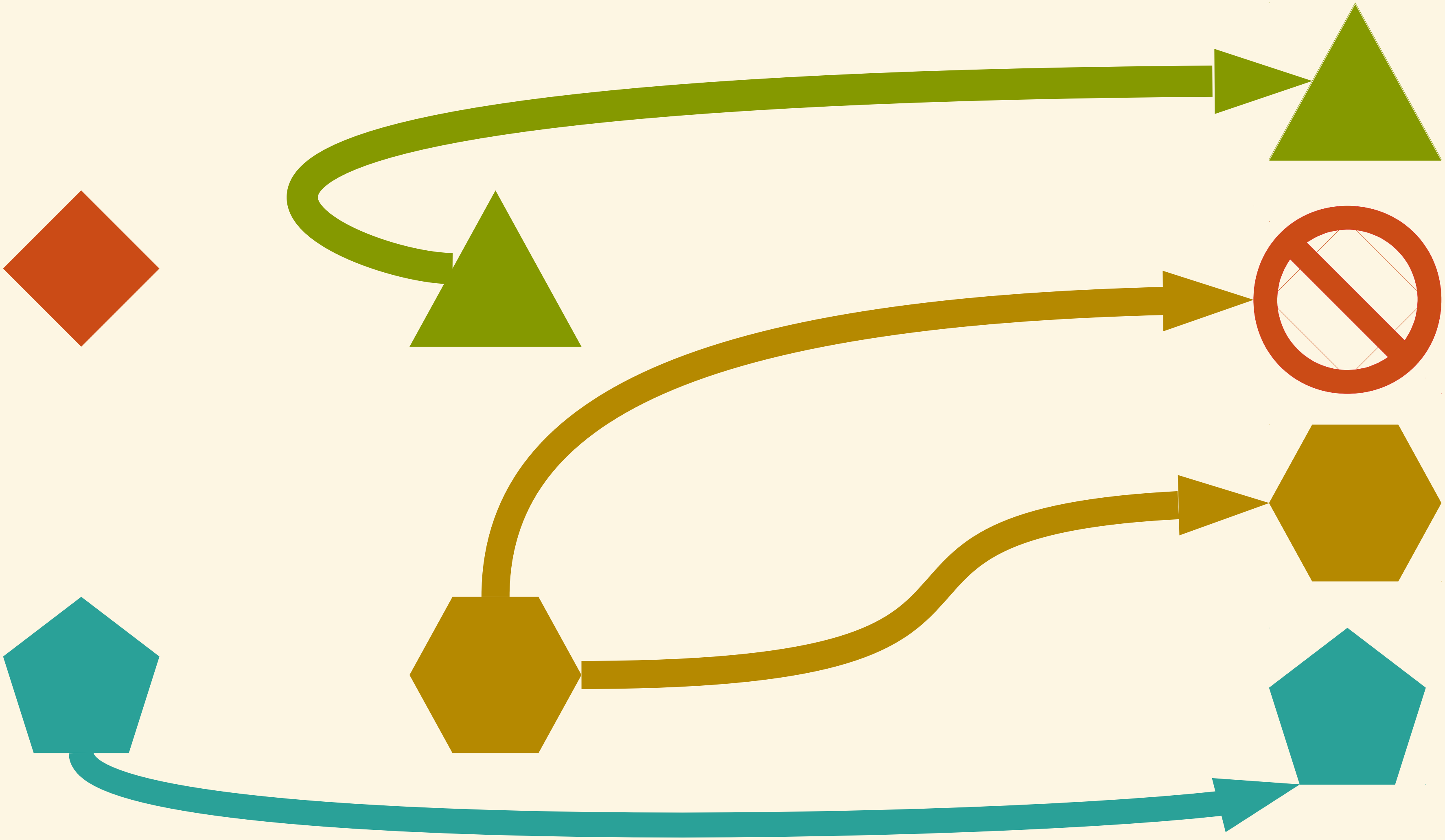
code repositories and deployment strategies for configuration management

GULCh
Gruppo Utenti Linux Cagliari h...?

Opera

Marco Marongiu (@brontolinux)

Do the right thing...

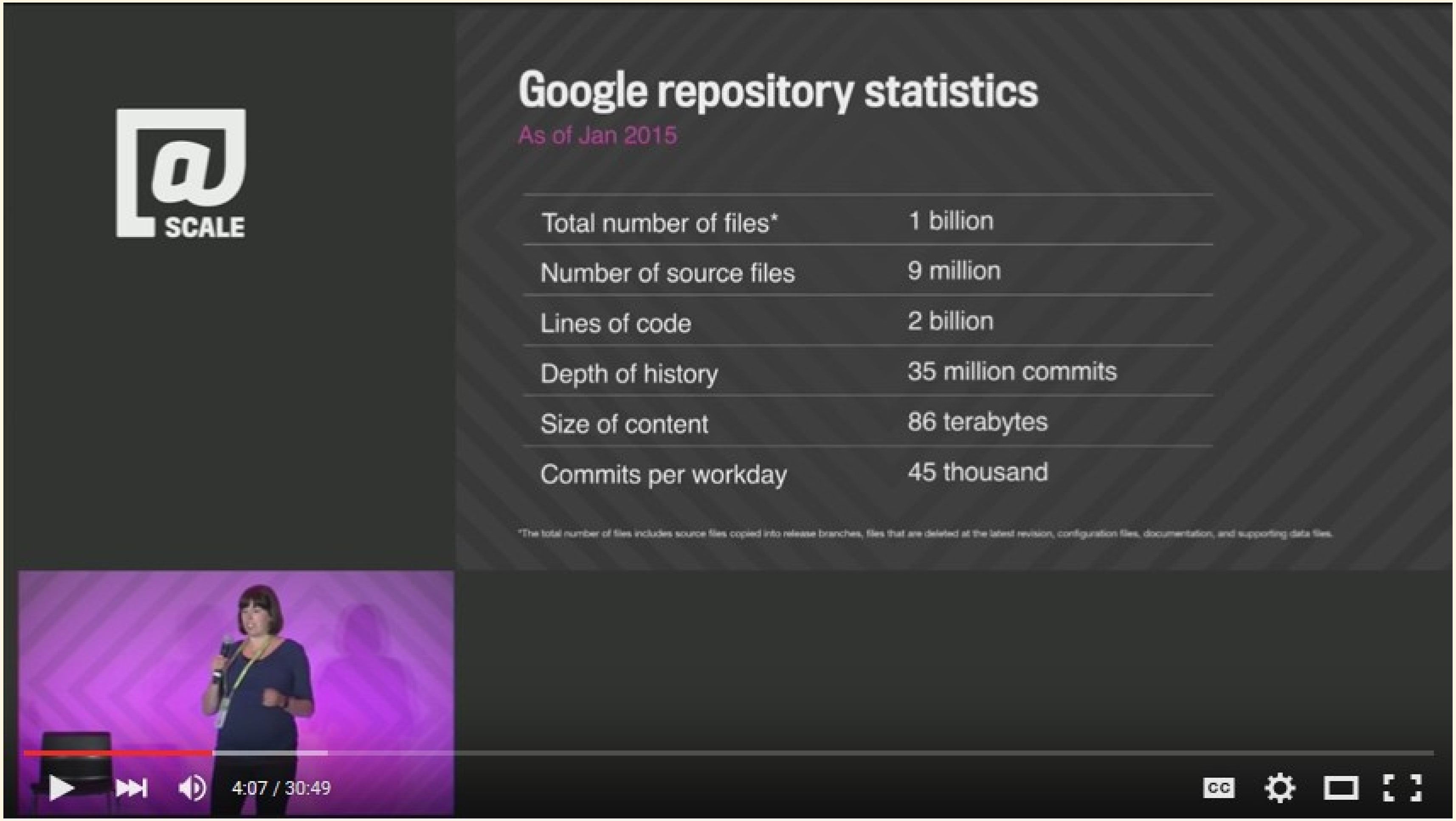


Do the right thing...

One repository per project or one for all?

- One per project + shared code
  - how to keep track of which version of the shared code was deployed at time t?
- One repository for all
  - how make the project code and the shared code merge gracefully?
- A mix?
  - e.g.: per-project branches plus shared code in master?

The answer is: *it depends...*



Google repository statistics

As of Jan 2015

Total number of files*	1 billion
Number of source files	9 million
Lines of code	2 billion
Depth of history	35 million commits
Size of content	86 terabytes
Commits per workday	45 thousand

*The total number of files includes source files copied into release branches, files that are deleted at the latest revision, configuration files, documentation, and supporting data files.

Knowledge and common sense

Not many good examples out there, buried in tons of *******.

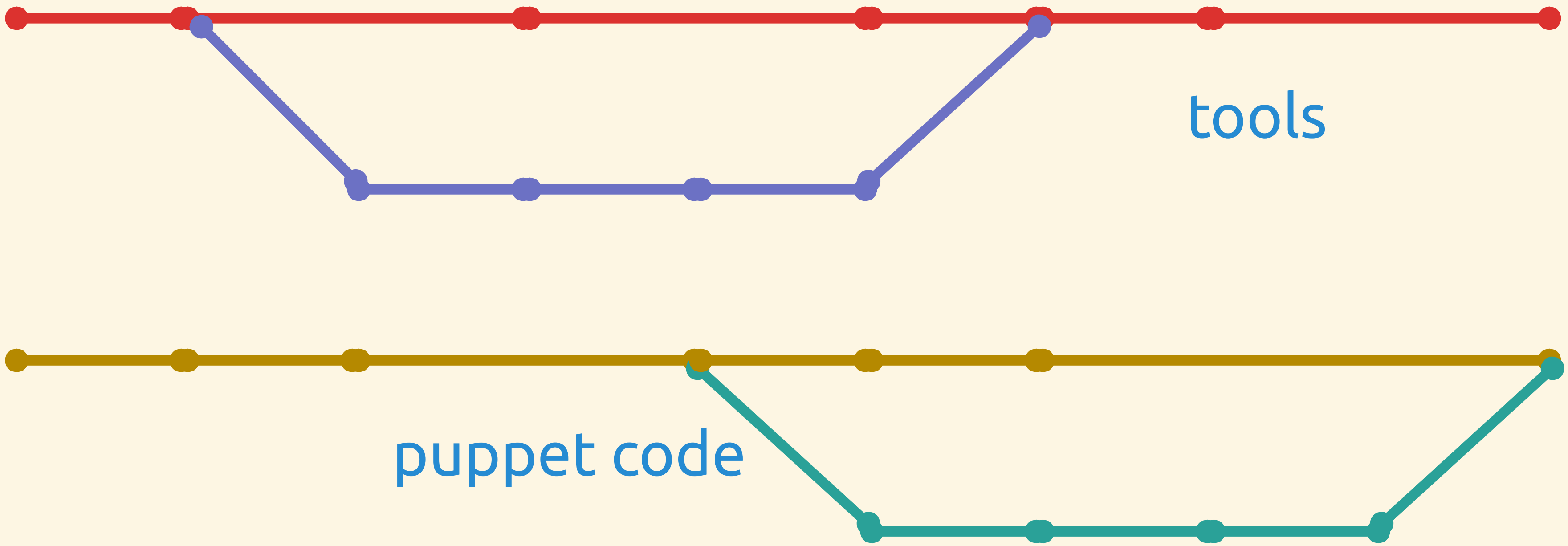
Knowledge and common sense is all you have to understand what fits best for you:

- the knowledge of your problem
- the knowledge of your VCS of choice
- common sense

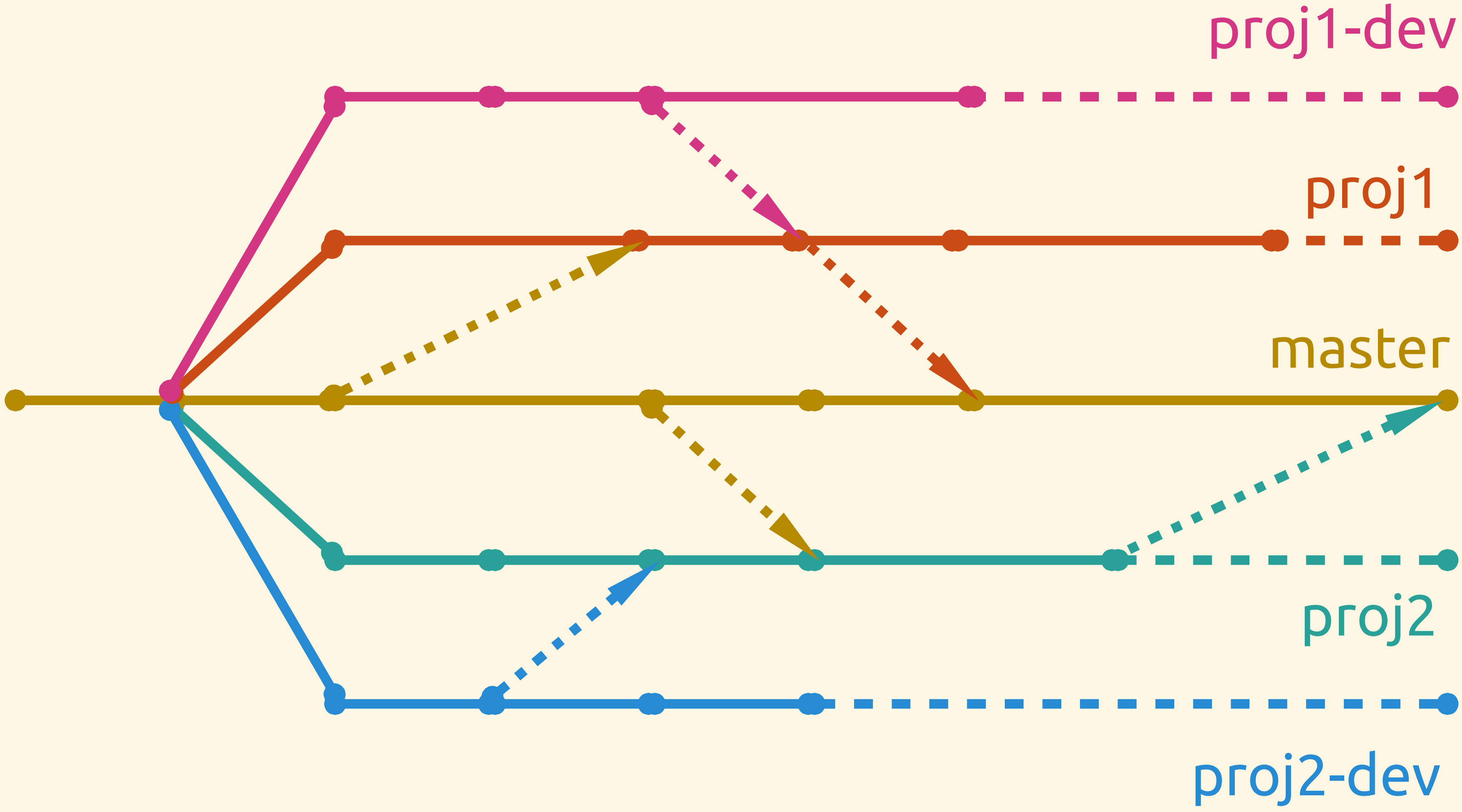
Experience can save you. If the problem is new for you, brace for impact...

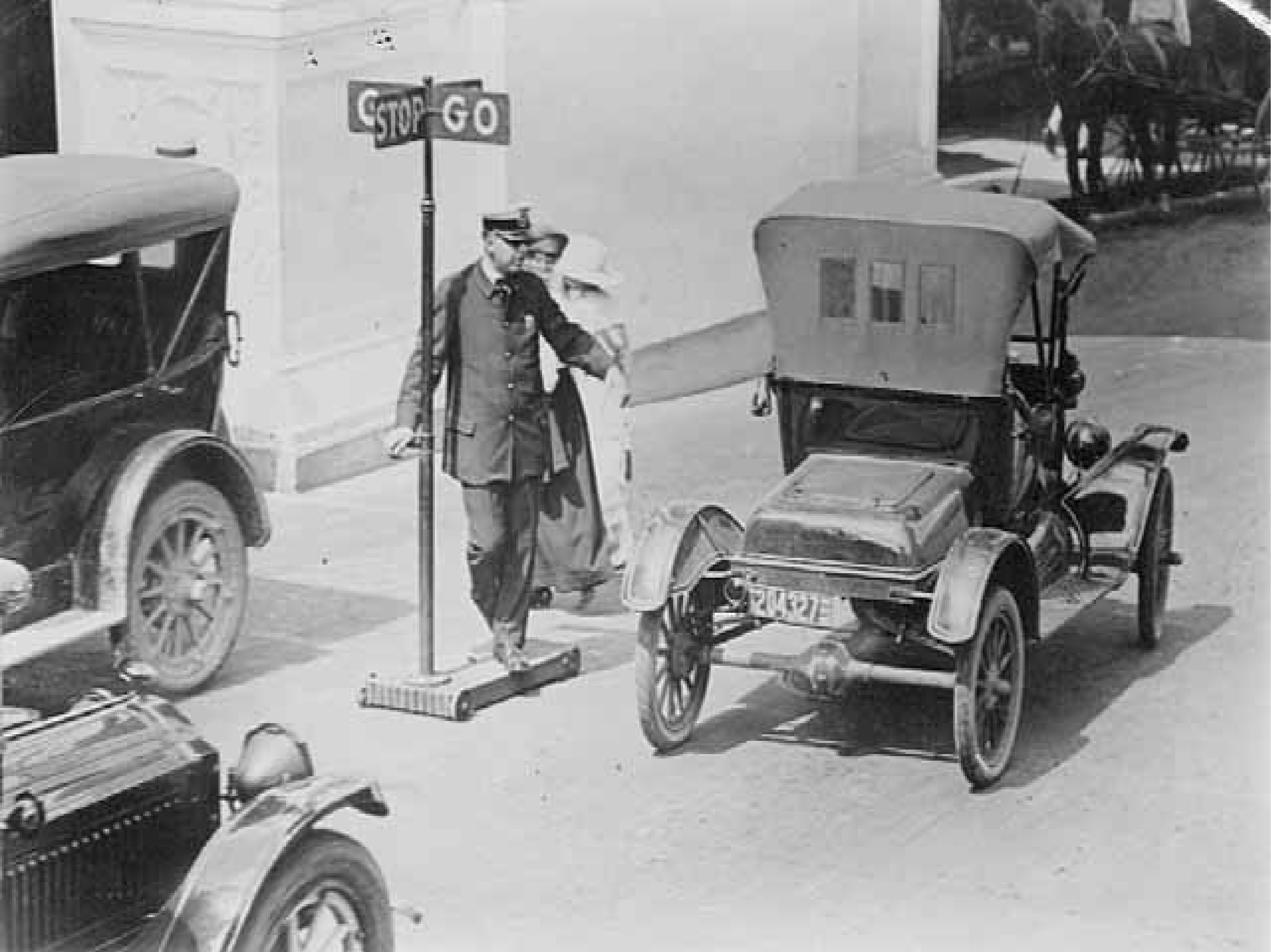
25

Separated lives

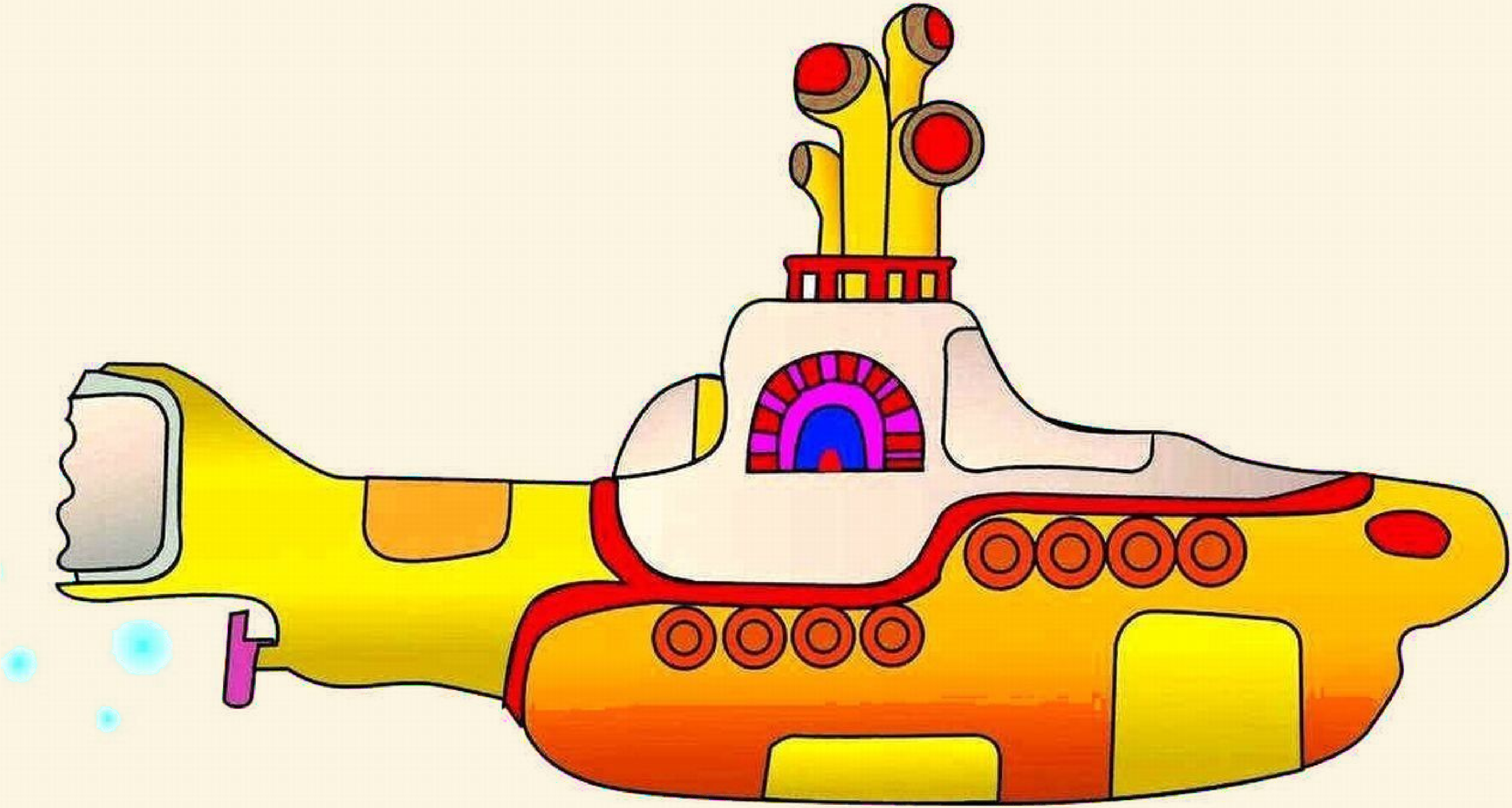


Separated lives



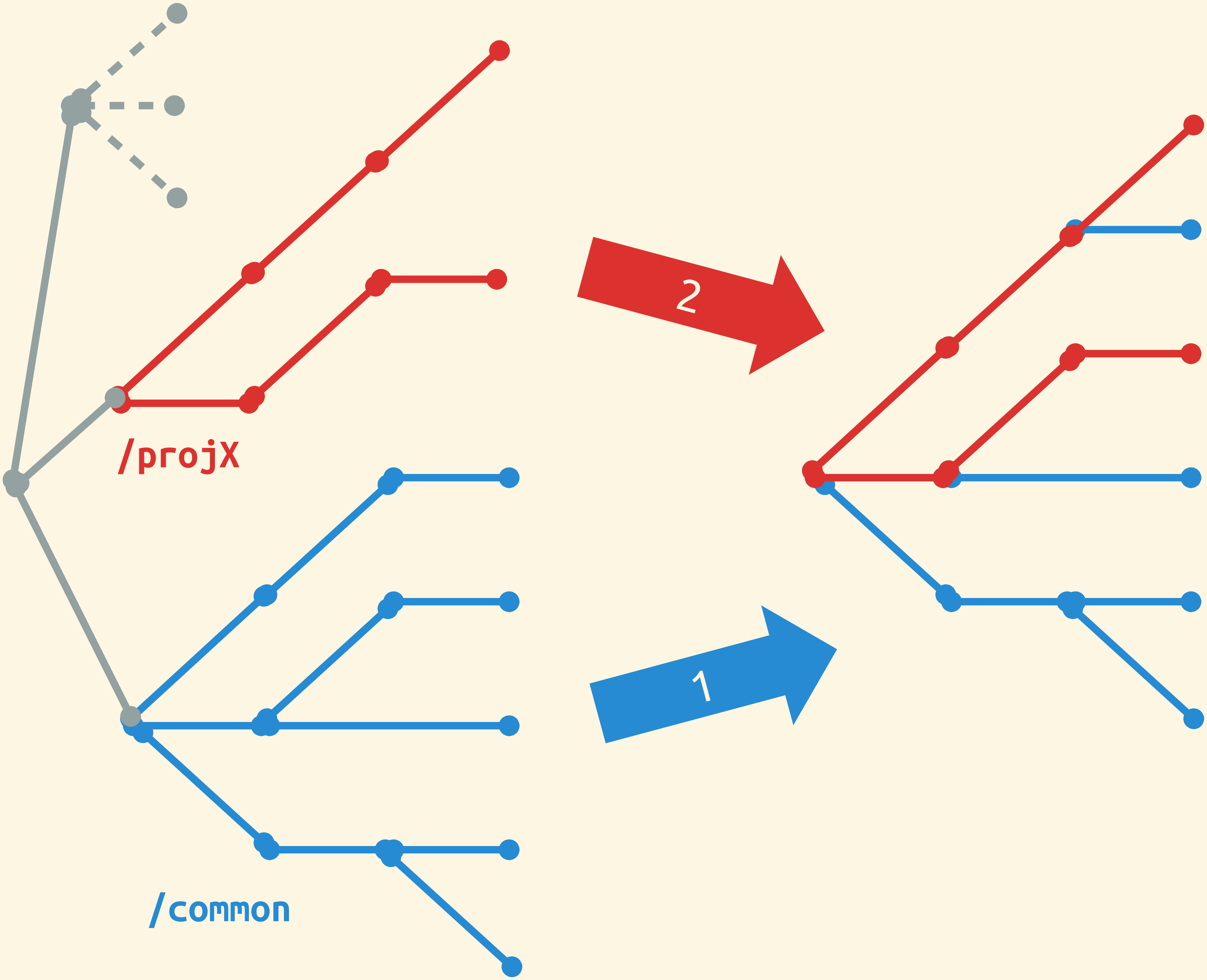

STOP
GO

We all live in a...



```
opera/
|-- controls
|   |-- cf_report.cf
|   |-- cf_runagent.cf
|   `-- cf_serverd.cf
|-- def.cf
|-- libraries
|   `-- site-opera.cf
|-- promises.cf
|-- services
...
```

```
common/
|-- controls
|   |-- cf_agent.cf
|   |-- cf_execd.cf
|   `-- cf_monitord.cf
|-- libraries
|-- modules
|-- services
|-- sources
|-- templates
|-- tools
`-- update.cf
...
```


/projX
/common
2
1

It works, but...

- command line long and ugly

```
make -C /var/cfengine/git/common/tools/deploy deploy
PROJECT=projX BRANCH=dev-projX-foo SERVER=projX-
testhub
```

- not optimized to deploy on more than one server at a time

```
for SERVER in projX-hub{1..10} ; do make -C
/var/cfengine/git/common/tools/deploy deploy
PROJECT=projX BRANCH=dev-projX-foo SERVER=$SERVER ;
done
```

- deploying on all the policy hubs required to remember all of the addresses/hostnames

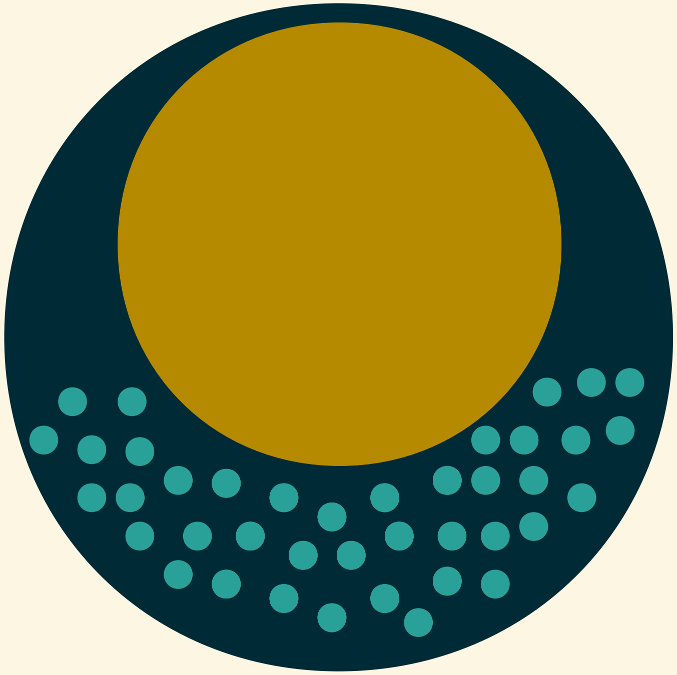
Meet cf-deploy

Front-end to `make` for deployments

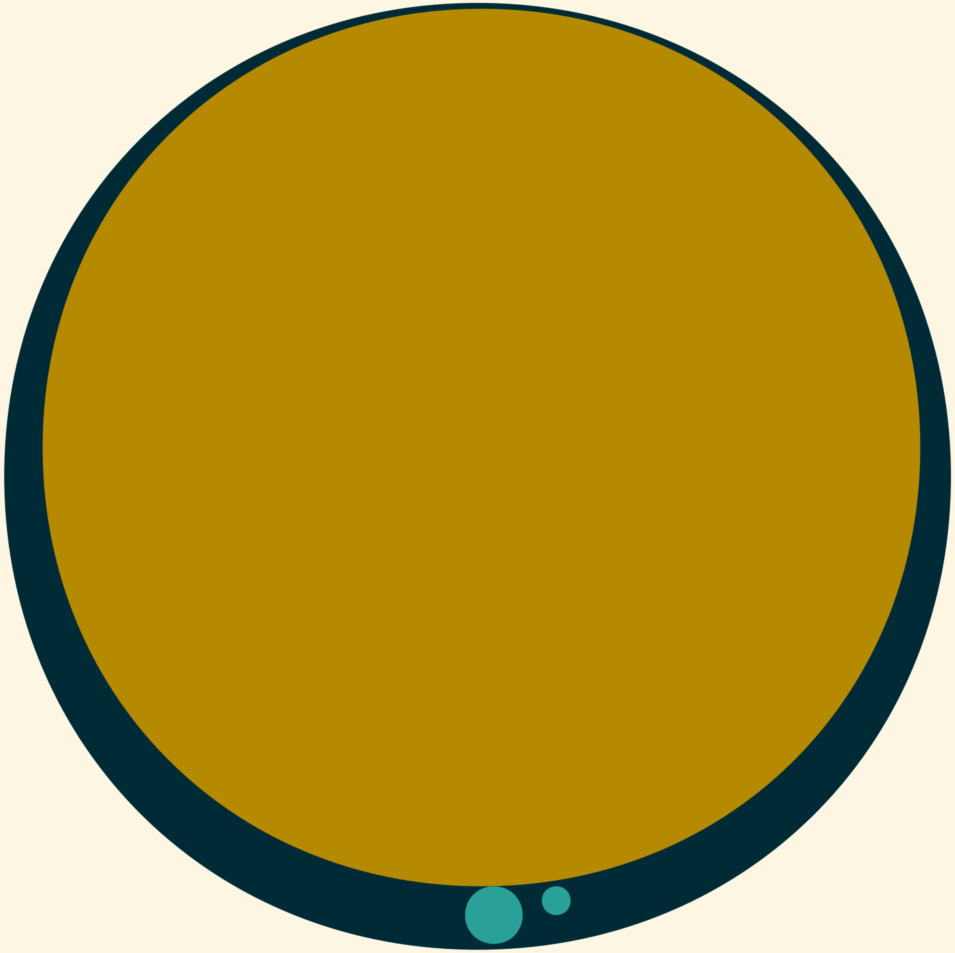
Initially a bash script, but...

  two configuration files per each (project, environment, location) triple

  it works, but...



165/231 lines of bash
40+ config files
348 lines in total



266/457 lines of perl
2 config files
493 lines in total

```
# project, directory, type
proj1,     project1,  remote
proj2,     project2,  remote
proj3,     project3,  remote
proj4,     project4,  remote
myownpc,   myownpc,   local
```

```
# Location, Project,  Environment, CNAME
Ashburn,    proj1,    prod,        proj1-us-cfengine.doma.in
Amsterdam,  proj2,    prod,        cfengine-ams.amsterd.am
Amsterdam,  proj2,    prod,        cfengine-ams.oursh.op
Ashburn,    proj2,    prod,        cfengine-ash.oursh.op
Thor,       proj3,    prod,        cfengine-proj3-prod.icela.nd
Thor,       proj3,    staging,     cfengine-proj3-stag.icela.nd
Oslo,       proj2,    prod,        cfengine.oursh.op
Seattle,    proj4,    prod,        cf-proj4-sea.doma.in
Wroclaw,    proj2,    prod,        cfengine.wrocl.aw
Oslo,       proj3,    test,        cf-test-v01.os.lo
Oslo,       proj4,    test,        cf-test-v06.os.lo
Oslo,       proj1,    test,        cf-test-v10.os.lo
Oslo,       proj2,    test,        cf-test-v12.os.lo
Oslo,       proj4,    test,        cf-test-v20.os.lo
Oslo,       proj2,    preprod,     pre-cfengine.os.lo
Seattle,    proj2,    preprod,     pre-cfengine-sea.oursh.op
Thor,       proj2,    preprod,     pre-cfengine-thor.oursh.op
Ashburn,    proj4,    preprod,     pre-cf-proj4-ash.doma.in
Seattle,    proj4,    preprod,     pre-cf-proj4-sea.doma.in
none,       myownpc,  prod,        /var/cfengine/inputs
```

How cf-deploy works

1. it reads from one configuration file which subdirectory should be deployed together with **`/common`** and the project type
   - remote: must rsync to a remote server to deploy
   - local: must rsync to a local filesystem
2. it reads the other configuration file to calculate the list of the hubs to deploy to
3. it runs the requested action.

How to deploy a project

Before (when the project has only one hub!):

```
make -C
/var/cfengine/git/common/tools/deploy
deploy PROJECT=projX BRANCH=master
SERVER=projX-hub
```

After (regardless):

```
cf-deploy deploy projX
```

or even shorter:

```
cf-deploy projX
```

How to preview a change

(which files will be modified by the deployment)

Before (one hub!):

```
make -C
/var/cfengine/git/common/tools/deploy
preview PROJECT=projX BRANCH=master
SERVER=projX-hub
```

After (regardless):

```
cf-deploy preview projX
```

How to preview a change

(diff the files)

Before:

```
make -C
/var/cfengine/git/common/tools/deploy
diff PROJECT=projX BRANCH=master
SERVER=projX-hub
```

After:

```
cf-deploy diff projX hub projX-hub
```

How to operate on a branch

other than master

Before:

```
make -C
/var/cfengine/git/common/tools/deploy
action PROJECT=projX BRANCH=name
SERVER=projX-hub
```

After:

```
cf-deploy action projX branch name
```

Operate on a specific environment

for a project, e.g. test

Before:

```
for SERVER in list hubs in test ; do
make -C
/var/cfengine/git/common/tools/deploy
action PROJECT=projX BRANCH=master
SERVER=$SERVER ; done
```

After:

```
cf-deploy action projX-test
```

Operate on a specific location

for a project

Before:

```
for SERVER in list hubs in location ; do
make -C
/var/cfengine/git/common/tools/deploy
action PROJECT=projX BRANCH=master
SERVER=$SERVER ; done
```

After:

```
cf-deploy action projX-location
```

List all hubs

Before: a separate list is needed, please remember to keep it updated

```
? ? ?
```

After: the list is part of the tool

```
cf-deploy list all_hubs
cf-deploy list hubs # non-test only
```

Is there more?

```
$ cf-deploy list projects
opera
myownpc
example
$ cf-deploy show myownpc
Description for project myownpc
Project type:   local
Git project ID: myownpc
Target dir:     /var/cfengine/inputs
$ cf-deploy show example
Description for project example
Project type:   remote
Git project ID: example
Hubs:
  cfengine.example.com
  cfengine-test.example.com
```

Summing up...

- One size **doesn't** fit all
- one repository for all projects
- tools, common libraries and project-specific parts together in the same repository
- libraries and project-specific parts merged at deploy time

Advantages of this solution

- all projects benefit from the improvements made to the libraries
- branches are used mainly for development of new features or the implementation of non-trivial changes
- possibility to use branches as "masters" for projects that need a non rolling approach for the deployment of shared libraries

Shortcomings

- projects can get part of the shared libraries they are not interested in
- an unnoticed bug in one of the shared libraries can propagate easily to all projects
- not suitable wherever a strong separation for different projects is needed

Questions?

Where's the code?

- On github: https://github.com/brontolinux/cf-deploy
- Pull requests and code contributions more than welcome

Thank you!

twitter: @brontolinux

email: mmarongiu@tiscali.it

LinkedIn: http://no.linkedin.com/in/marcomarongiu/

Blog: http://syslog.me/

Can I borrow one more minute?

KREFTFORENINGEN

KWF
KANKER
BESTRIJDING

Give a chance to the ones you love

Donate to cancer research

Donate ***today***

Fondation
contre le Cancer
Ensemble pour la vie

CONTRE
LE CANCER
LA LIGUE
pour la vie

AIRC
ASSOCIAZIONE ITALIANA
PER LA RICERCA SUL CANCRO