The release cycle of DPM

Fabrizio Furano
DPM workshop 2019

DPM release cycle

- The DPM release cycle was shaped in 2012
- Remained pretty much the same, and still will
- Since then the number of SRPMs has gone from tens down to 3, through various rounds of rationalisation

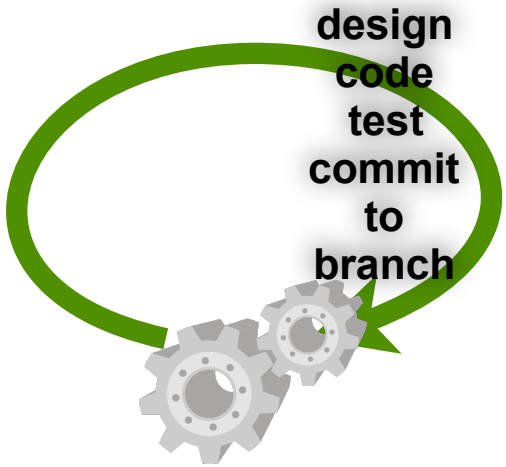
- Some details/terminology have changed, due to the migration to GitLab

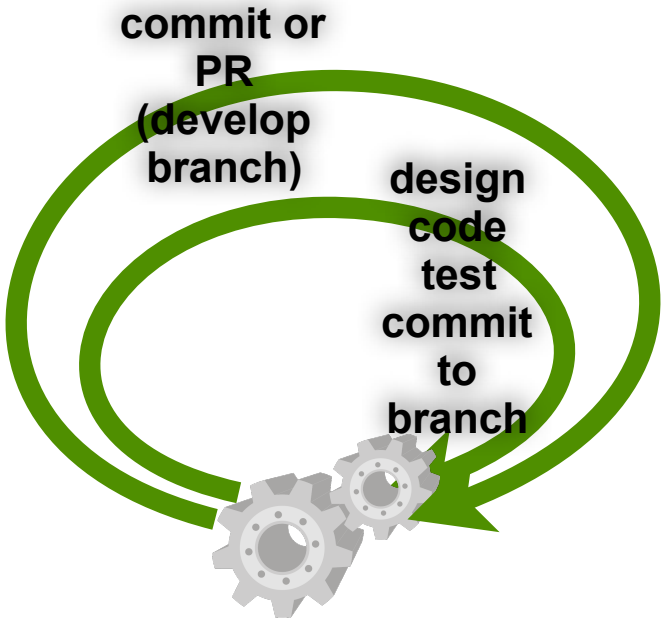
- Proved to be very reliable, and able to catch issues before they are exposed to sites
  - Also issues with external components or incompatibilities with 3rd party libraries (e.g. the recent story with libvomsxrd)

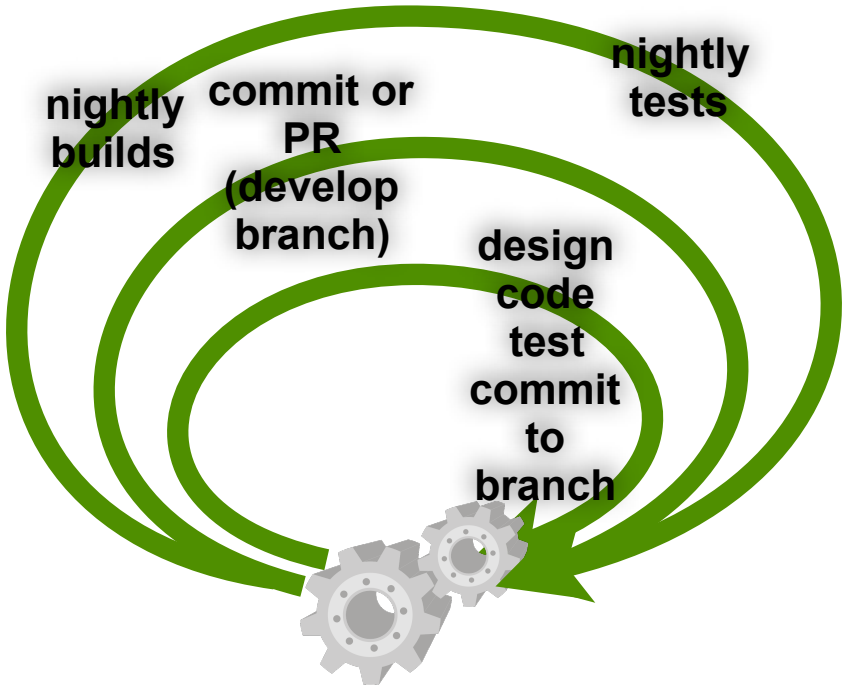
DPM SRPMs

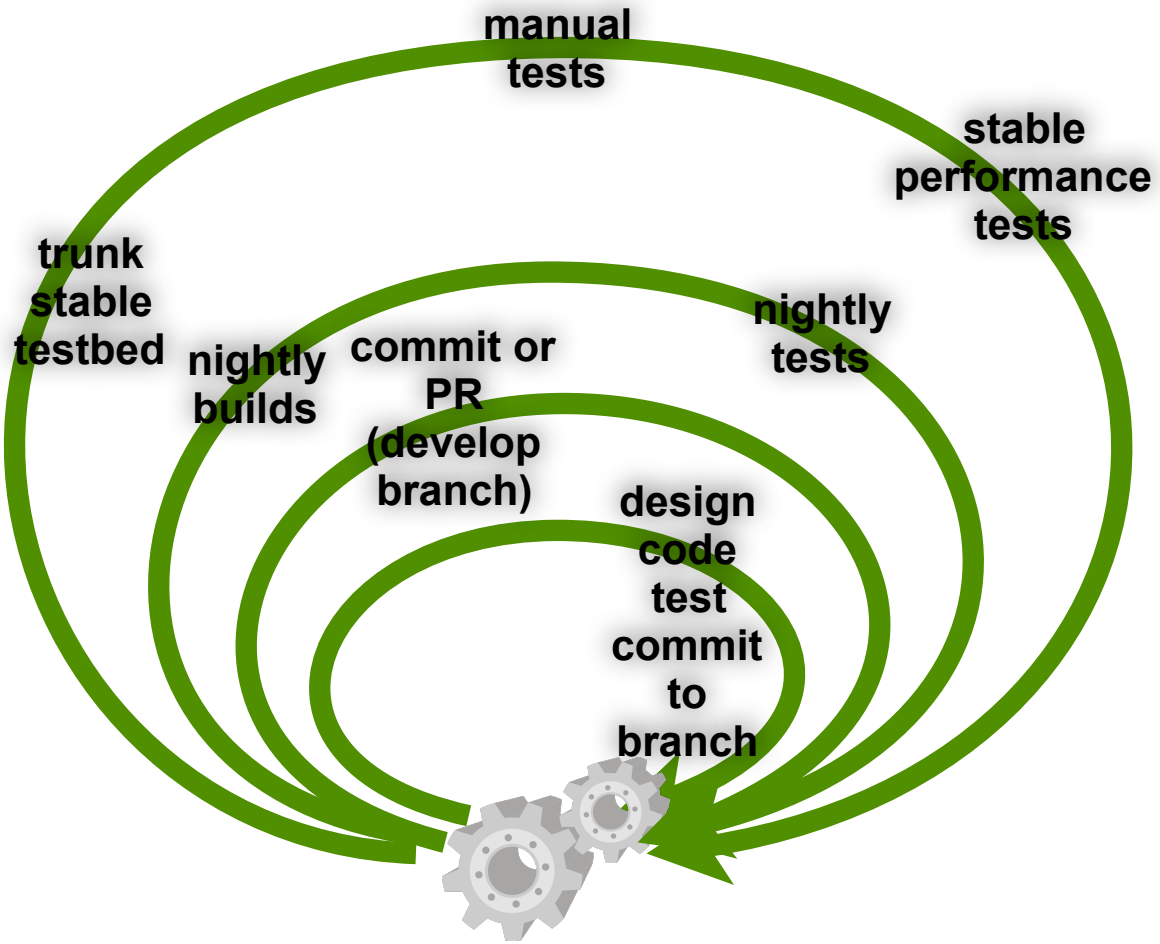
- SRPMs are what is submitted to EPEL for making a release
- They are just 3 nowadays
  - dmlite (core, dev headers for C and C++, plugins)
  - lcg-dm (the legacy codebase)
  - lcgdm-dav

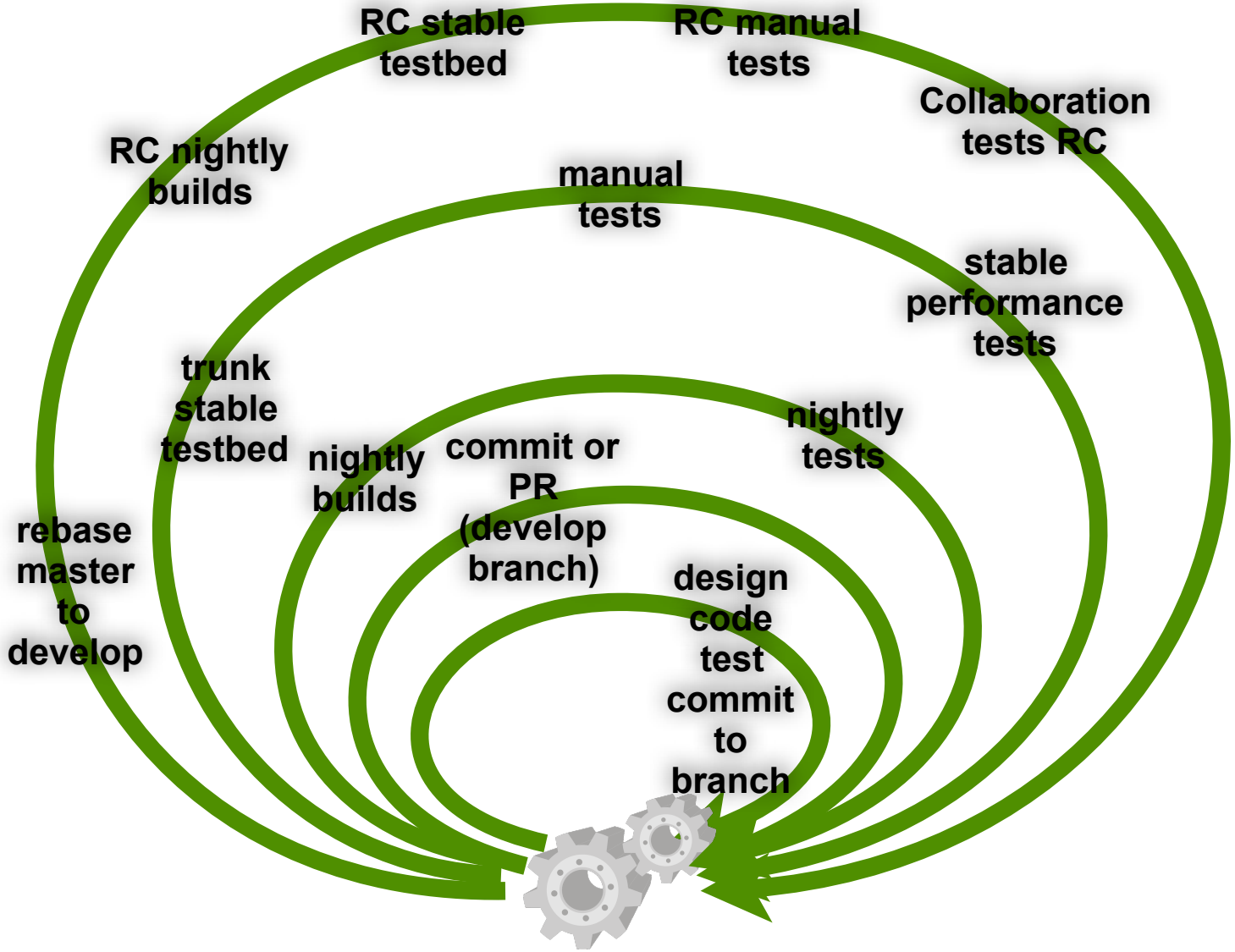
- We will have a look at moving lcgdm-dav into the dmlite SRPM, as a final simplification step
- Of course, it makes sense only if it simplifies things :-) will see

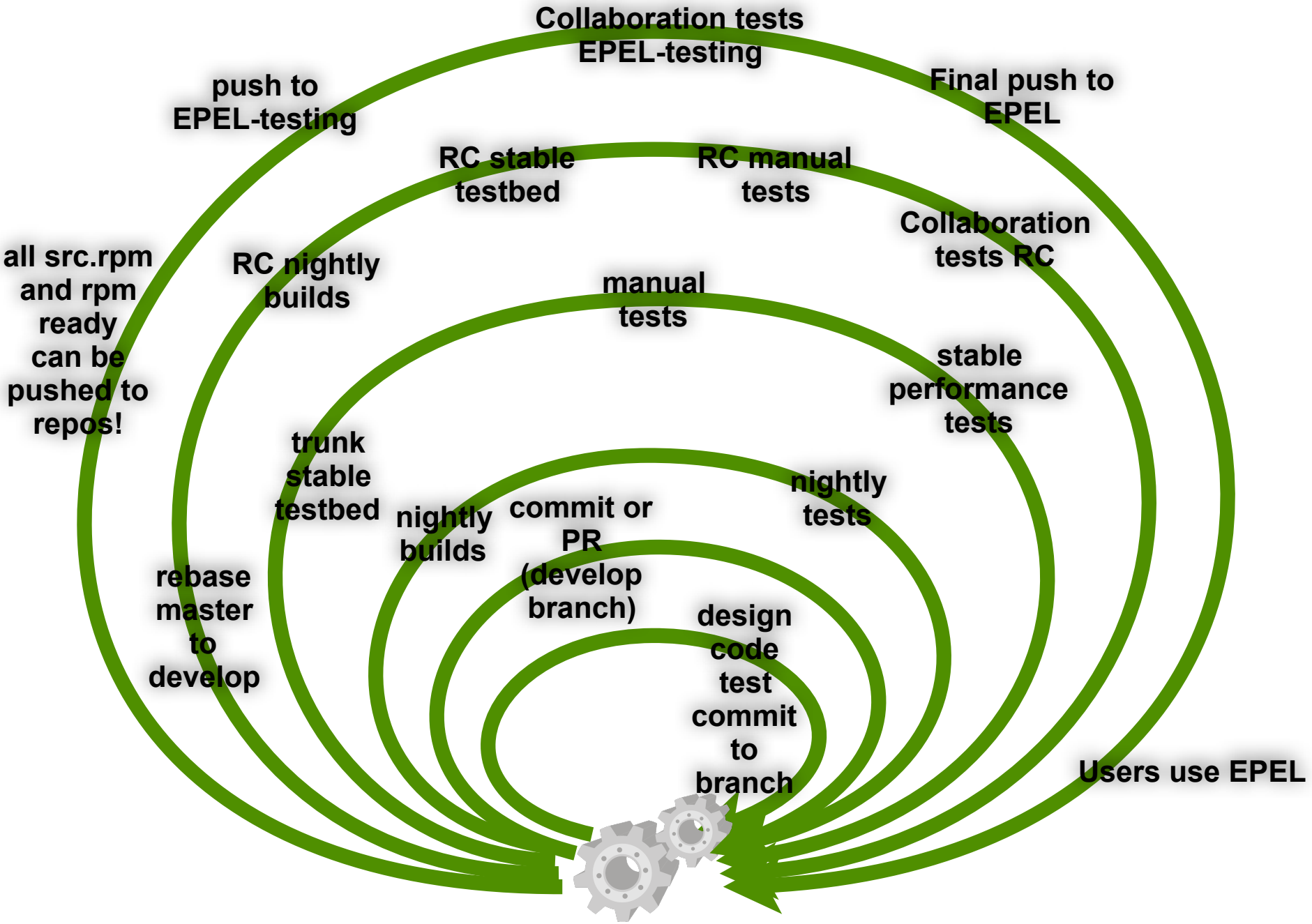

design
code
test
commit
to
branch

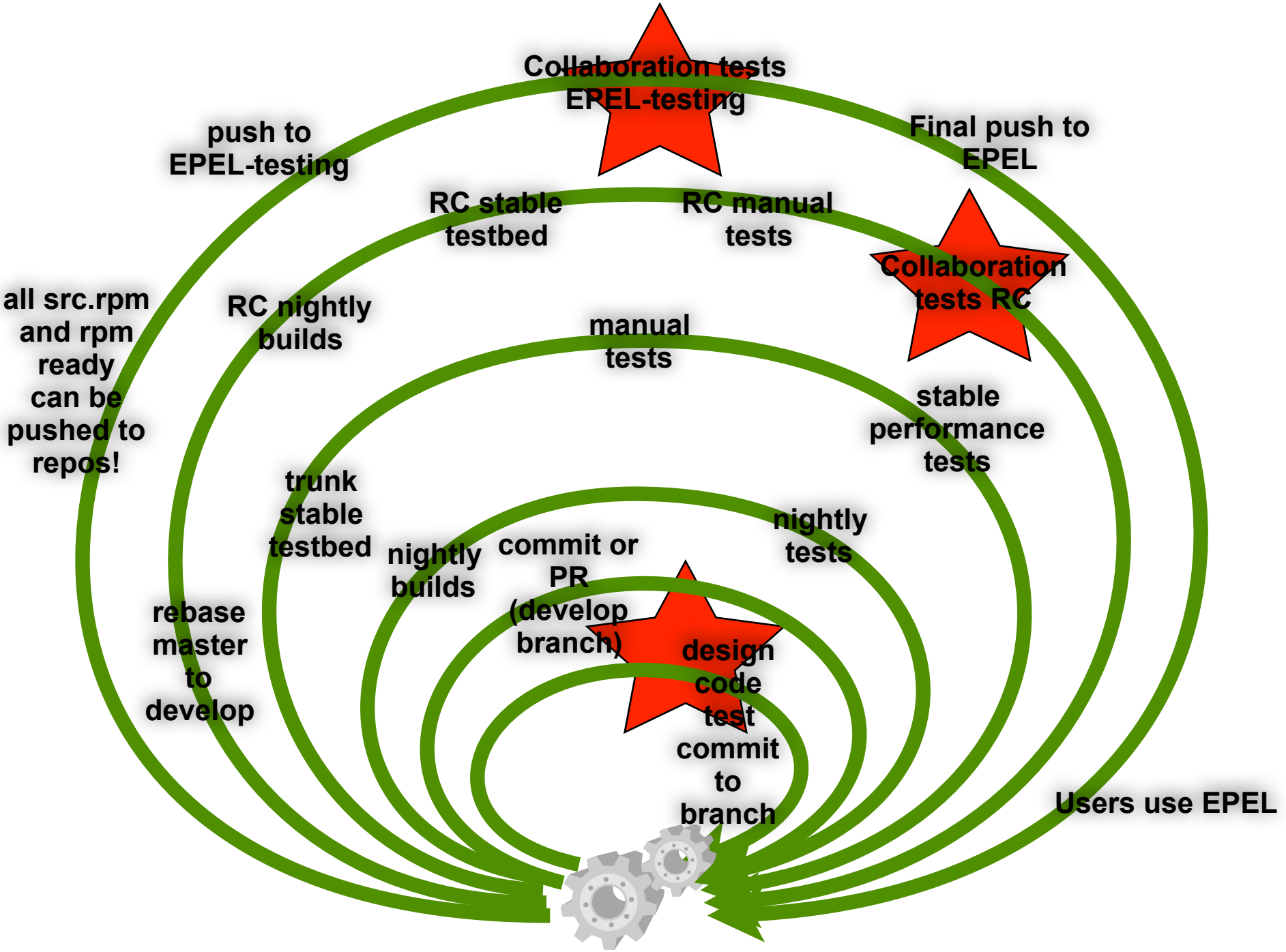

commit or
PR
(develop
branch)
design
code
test
commit
to
branch


nightly builds
commit or PR (develop branch)
nightly tests
design
code
test
commit
to
branch

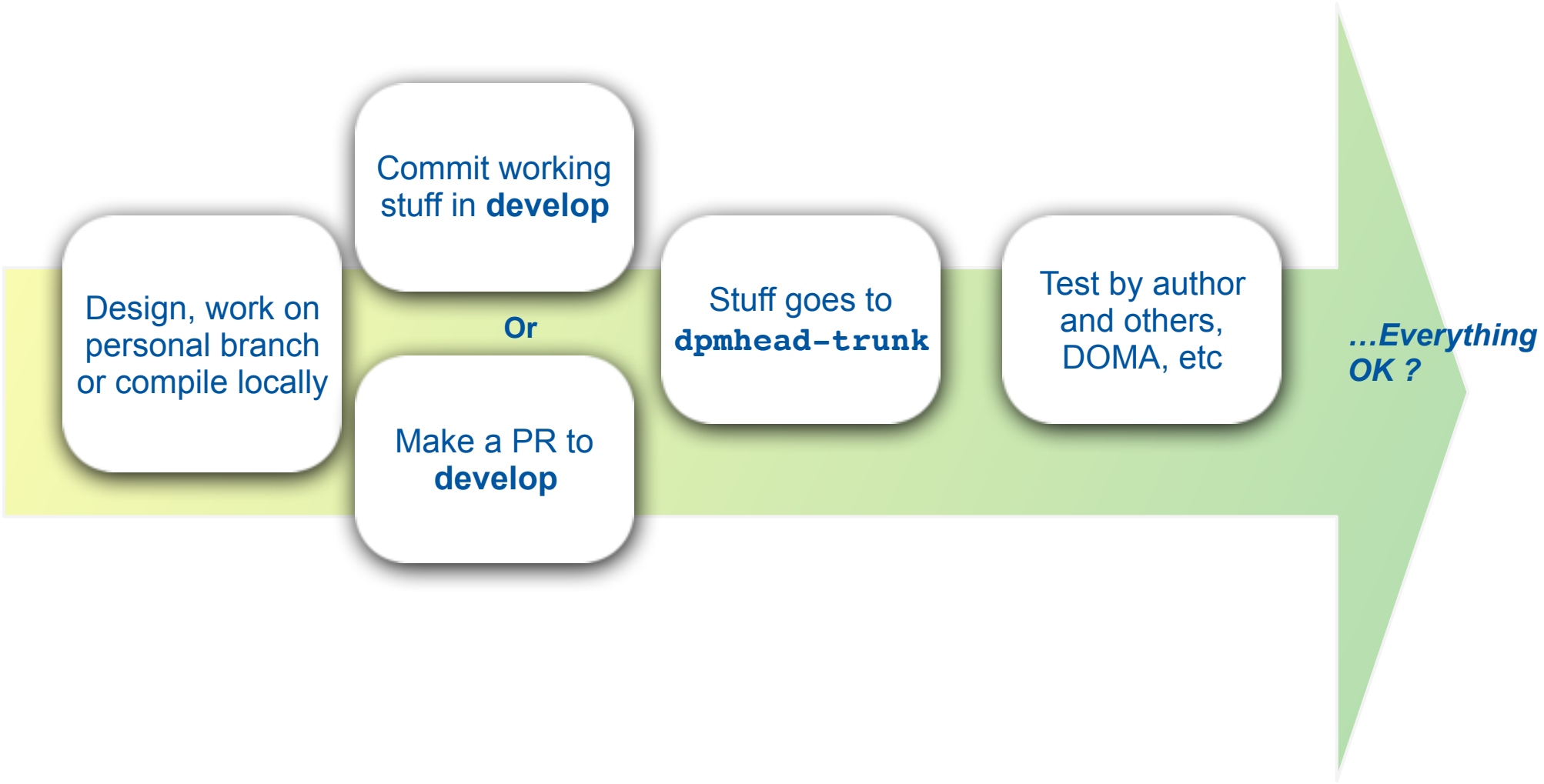

manual
tests
stable
performance
tests
trunk
stable
testbed
nightly
builds
commit or
PR
(develop
branch)
nightly
tests
design
code
test
commit
to
branch


RC stable testbed
RC manual tests
Collaboration tests RC
RC nightly builds
manual tests
stable performance tests
trunk stable testbed
nightly builds
commit or PR (develop branch)
nightly tests
rebase master to develop
design code test commit to branch


Collaboration tests
EPEL-testing
push to
EPEL-testing
Final push to
EPEL
RC stable
testbed
RC manual
tests
Collaboration
tests RC
all src.rpm
and rpm
ready
can be
pushed to
repos!
RC nightly
builds
manual
tests
stable
performance
tests
trunk
stable
testbed
nightly
builds
commit or
PR
(develop
branch)
nightly
tests
rebase
master
to
develop
design
code
test
commit
to
branch
Users use EPEL

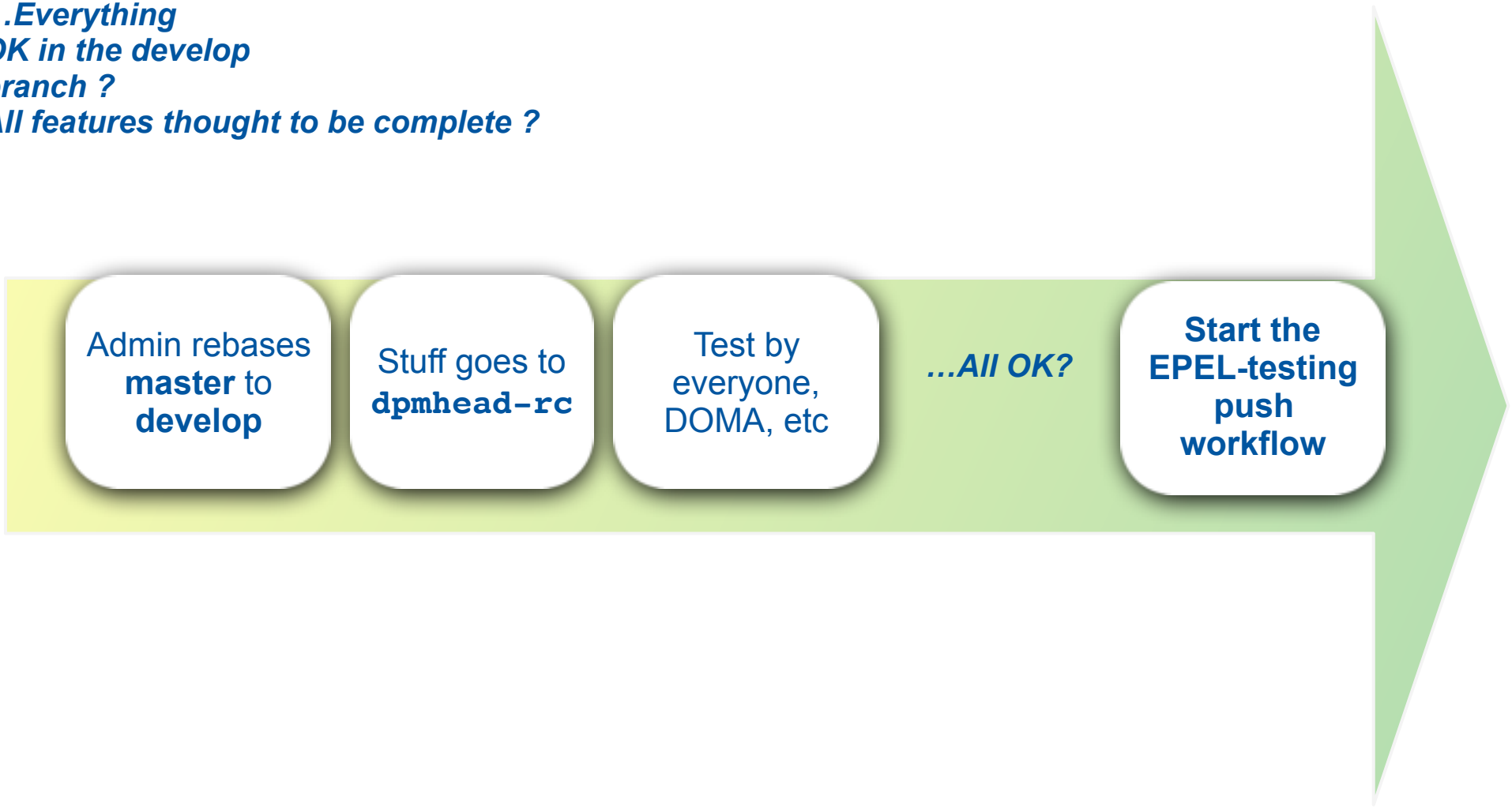

Collaboration tests
EPEL-testing
push to
EPEL-testing
Final push to
EPEL
RC stable
testbed
RC manual
tests
Collaboration
tests RC
all src.rpm
and rpm
ready
can be
pushed to
repos!
RC nightly
builds
manual
tests
stable
performance
tests
trunk
stable
testbed
nightly
builds
commit or
PR
(develop
branch)
nightly
tests
rebase
master
to
develop
design
code
test
commit
to
branch
Users use EPEL

Gitlab workflow (1/2)



Gitlab workflow (2/2)

…Everything OK in the develop branch ? All features thought to be complete ?



Develop branch - conventions

- The develop branch is the official branch where the features and the fixes are assembled

- ***The stuff must compile***
- Features can be incomplete for a short time (days), with a commitment to fix them

- Every push triggers a build that goes into the “continuous builds repo”
- `dpmhead-trunk.cern.ch` updates overnight (or manually)
- Nightly tests run, the results can be seen in Jenkins

Master branch - conventions

- The **master** branch cannot be committed to
- No **pull requests** can be made to the **master** branch

- The master branch gets the develop branch when it works, usually through rebasing

- ***In the master branch everything is at least supposed to work***

- Every push triggers a build that goes into the “release-candidate repo”
- `dpmhead-rc.cern.ch` updates overnight (or manually)
- Nightly tests run, the results can be seen in Jenkins

Binary compatibility

- Strong requirement: when a release is made, all the resulting RPMs must work together and together with the rest of EPEL

- Accomplished by
  - Construction: we build together all the DPM packages
  - Dependencies: all the RPMs built together have strict dependency rules. Accidental mixing is not possible, we release what we test
  - Testing!

- External components/projects rely on the usual ABI rules, i.e. the public API must not change across releases
  - This is the case of lcgdm-dav, which uses the public C API of dmlite

- The private API (i.e. the dmlite C++ API) instead can change, and the price of a change is to rebuild the external components that use it (e.g. dynafed)

Same version for all

```
dmlite-shell-1.13.0-1.20190603.1517.el7.x86_64
dmlite-plugins-adapter-1.13.0-1.20190603.1517.el7.x86_64
dmlite-dpm-tester-1.13.0-1.20190603.1517.el7.x86_64
dmlite-dpm-dsi-1.13.0-1.20190603.1517.el7.x86_64
dmlite-plugins-mysql-1.13.0-1.20190603.1517.el7.x86_64
dmlite-puppet-dpm-1.13.0-1.20190603.1517.el7.noarch
dmlite-libs-1.13.0-1.20190603.1517.el7.x86_64
python-dmlite-1.13.0-1.20190603.1517.el7.x86_64
dmlite-dpmhead-1.13.0-1.20190603.1517.el7.x86_64
dmlite-debuginfo-1.13.0-1.20190603.1517.el7.x86_64
dmlite-dome-1.13.0-1.20190603.1517.el7.x86_64
dmlite-dpm-xrootd-1.13.0-1.20190603.1517.el7.x86_64
dmlite-plugins-domeadapter-1.13.0-1.20190603.1517.el7.x86_64
```

Puppet templates on RPMs

- Recently we decided to provide our best puppet templates as a versioned RPM package belonging to a release

- This is to avoid fuzzy questions like “does template X in puppetforge work with dpm version x.y.z ?”
  - **In general I have no idea**

- We trust the templates inside the rpm because these are the ones used in our testbeds with our blessed versions

Puppetforge

- So far we have provided the templates in puppetforge for conveniency
- These are more suited for deriving personal templates, and we will try to make it clearer

- Templates have been published in puppetforge “as is”, for reference by puppetforge aficionados
  - **AFAIU Puppetforge can be used for quick exchanges, not for releases**

- We can’t use puppetforge for doing certified “atomic” releases, so we consider it more like a development branch
  - Difficult to have quality control

External packages - admintools

- The admintools package is not part of a DPM release
  - Useful scripts and queries to perform various tasks
  - Released asynchronously from time to time
  - Not versioned with the rest of DPM

- Some of these tools can only work if the legacy daemons are running
- We would not change the nature of this package, i.e. “3rd party” contributed tools
- What do we do ? More details in Andrea’s talk