CIRRAX

how and why we deploy OpenStack virtual environments

Benedikt Trefzer
benedikt.trefzer@cirrax.com

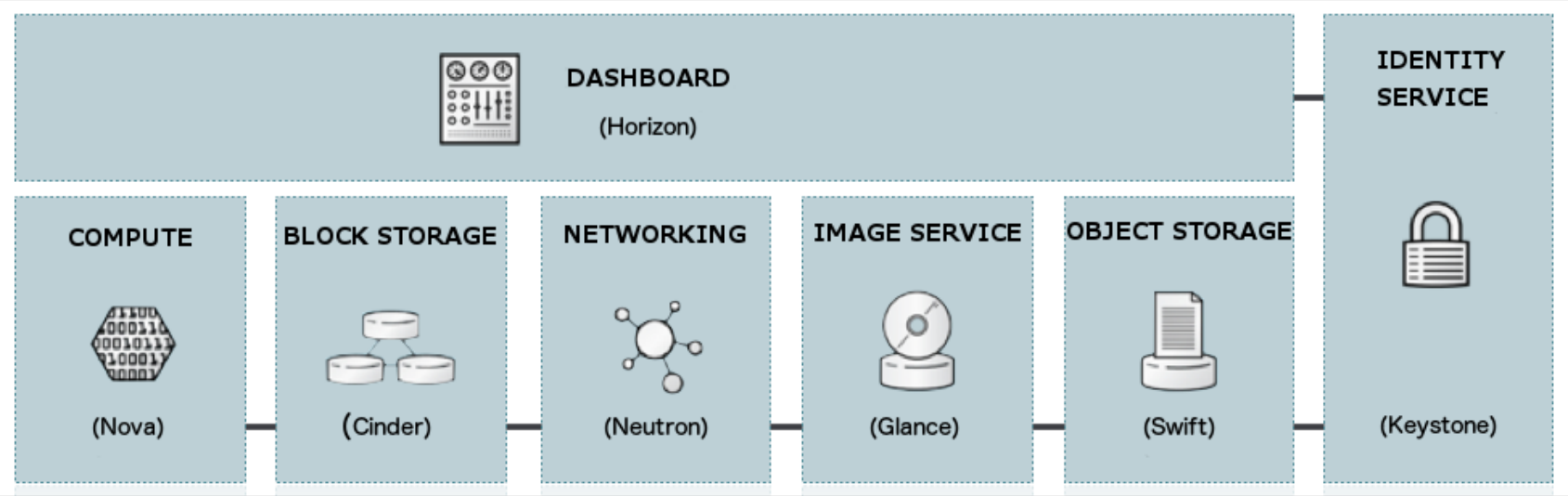
Cirrax GmbH

Circumstances

- Cirrax GmbH running an OpenStack cloud since grizzly release (2013)
- using Debian and the packages provided by Debian
- configuration with puppet manifests
- unhappy with release cycle (Debian: years, OpenStack: months)
- upgrades where a pain
  - dependency constraints
  - vendor repositories versus main/backport repository
  - debian specific reconfiguration mechanism

!time to evaluate a better approach!

OpenStack



[1]

- consist of many projects (nova, neutron, cinder, oslo ...)
- separate projects for services, (python-) libraries and clients
- projects use API calls to talk to each other
- written in python

[1] OpenStack service overview by OpenStack is licensed under CC-BY 3.0

OpenStack releases

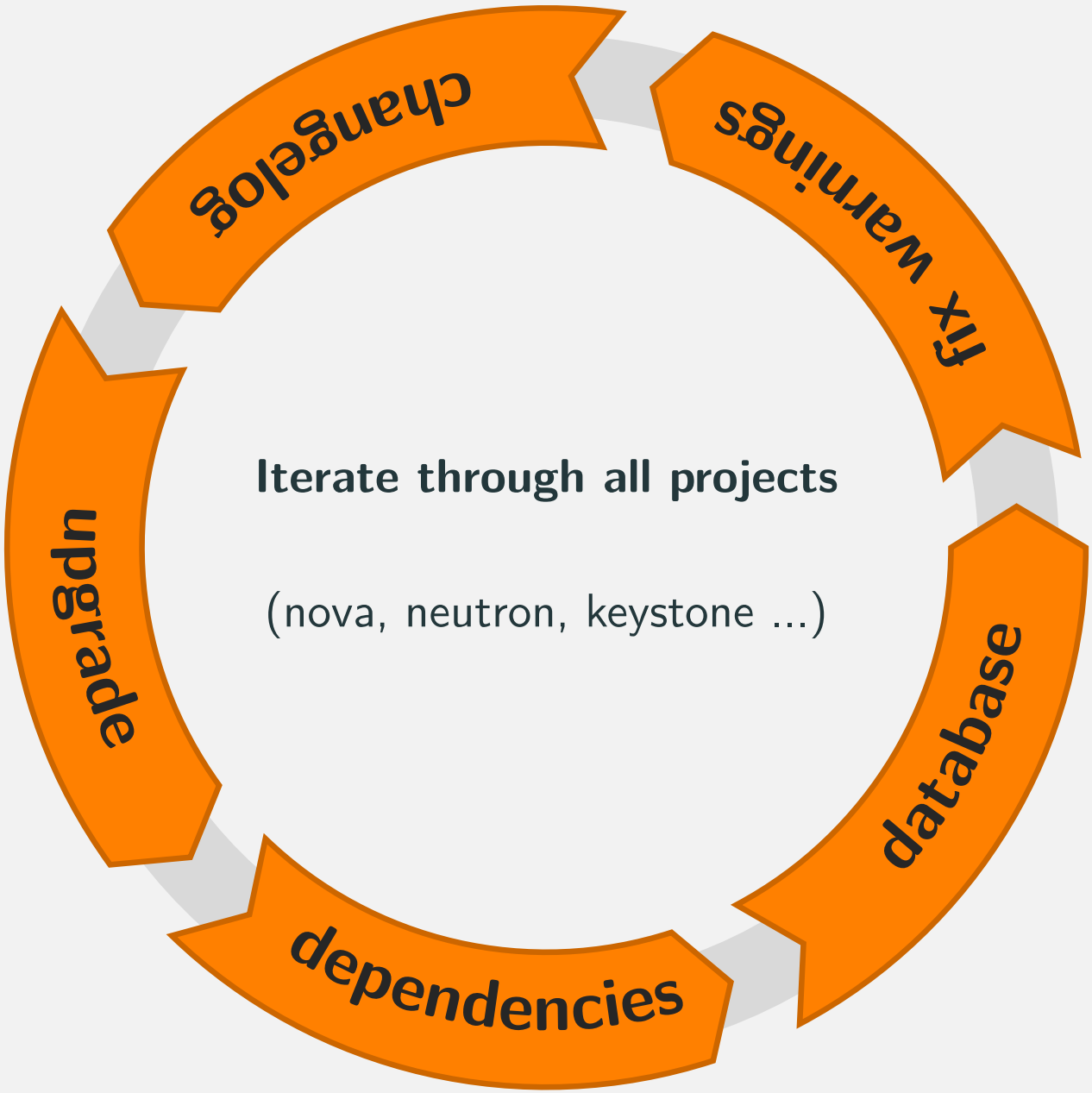
Series	Status	Initial Release Date	Next Phase	EOL Date
Rocky	Development	2018-08-30 *estimated* (schedule)	Maintained *estimated 2018-08-30*	
Queens	Maintained	2018-02-28	Extended Maintenance *estimated 2019-08-25*	
Pike	Maintained	2017-08-30	Extended Maintenance *estimated 2019-03-03*	
Ocata	Maintained	2017-02-22	Extended Maintenance *estimated 2018-08-27*	
Newton	End Of Life	2016-10-06		2017-10-25

[2]

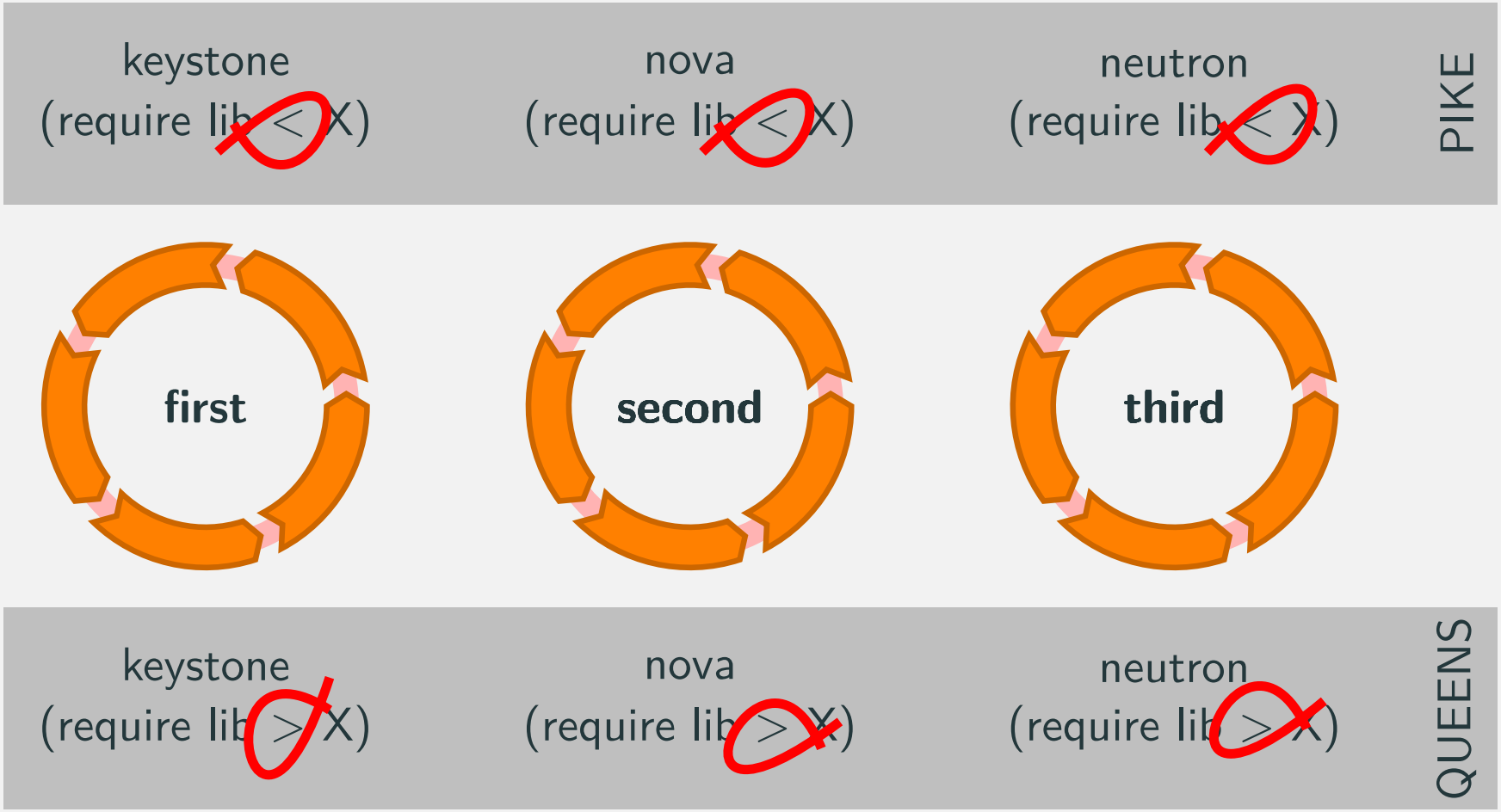
- 6-month release cycle
- only n+1 upgrades for database
- currently: discussions about LTS release and n+x upgrades

[2] `https://releases.openstack.org`

Openstack typical upgrade for one project



dependencies



!! troubles if a node uses more than one project !!

OpenStack dependency management

- git repo with OpenStack global requirements for all projects[3]
- ensures library versions across projects per release
- used for testing by OpenStack gates
- dependencies are included in operating system packages
- updates in stable branch for security issues

[3] `https://github.com/openstack/requirements`

solution 1: use deb/rpm packages from vendor

use deb/rpm packages from vendor, needs BigBang for upgrade

pros:

- usage of operating system packages (security upgrade etc.)
- established vendor upgrade procedure
- established devops tools can be used (puppet manifests)

cons:

- high risk since all projects are in transition at the same time
- all projects need to be on the same version (no per project upgrade)
- dependency of OpenStack and distribution releases

solution 2: Separate container for each project

separate OpenStack projects into container and or docker images

pros:

- no dependency constraints that block per project upgrades
- possibility to use operating system packages

cons:

- difficult for compute/network nodes (interaction with network, libvirt and ceph)
- adaptation for puppet needed
- additional upgrade chain for docker/container

solution 3: python VirtualEnvs

use pip to install OpenStack into python virtual environments

pros:

- no dependency constraints that block per project upgrades

cons:

- needs build dependencies on all nodes (compiler etc)
- additional upgrade mechanism (pip)
- difficulty to reinstall same virtual environment (eg. for test, production)
- need adaptation of puppet manifests (no package installation)

solution 4: python VirtualEnvs in operating system packages

use pip to install an OpenStack project into a python virtual environments and pack the virtual environment into a deb/rpm package

pros:

- no dependency constraints that block per project upgrades
- use of etablished system package management
- no build dependencies on nodes
- easy reinstall of same environment
- established puppet manifests can be used

cons:

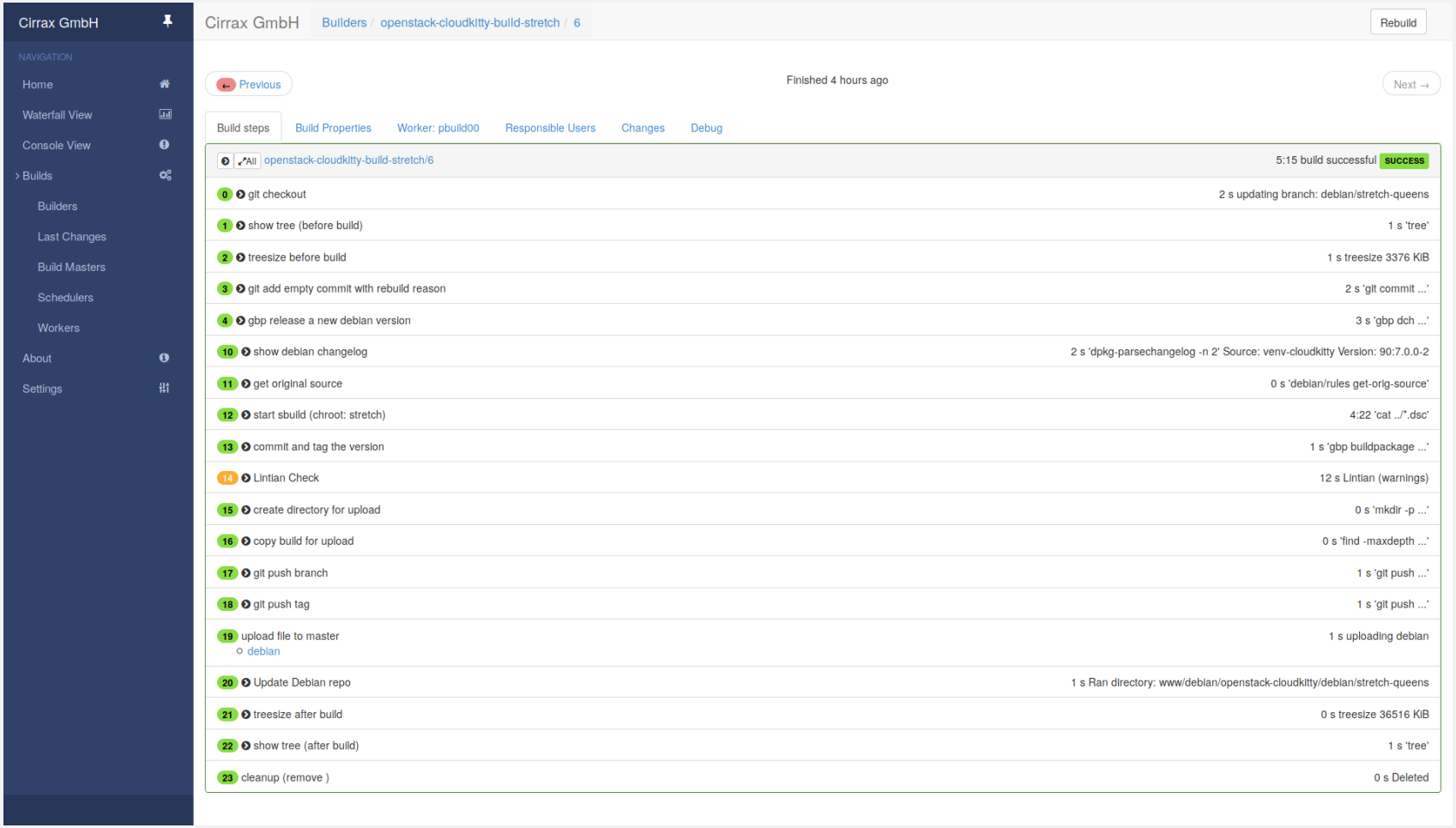
- need to create lots of packages

details for VirtualEnvs in deb packages

1. create requirements package (one per release)
   - compiles all requirements of release
   - creates a wheel (binary) repository suitable for pip to install from
   - squeeze repository into deb package
2. install a virtualenv for project
   - debian helper for virtualenvs (dh-virtualenv[4])
   - no compilation since binaries are in requirements package
   - no need to download from `https://pypi.org/`
   - gives a reproducible build
3. create a package containing the virtual environment with links
4. automated tests
5. add package to staging repository (reprepro)

[4] `https://dh-virtualenv.readthedocs.io`

automated using buildbot[5]



[5] https://buildbot.net/

lessons learned

- upgrades are less painfull, dependencies problem is solved
- puppet manifests needed minimal adaptation (usage of venv for wsgi[6])
- OpenStack projects are quite equal (if one project is done, the others are similar)
- automation with buildbot
- minor upgrades are auto triggerd
- new major version are implemented fast
- independent upgrade of OpenStack and operating system

[6] https://review.openstack.org/#/q/topic:add_wsgi_process_override

Questions

Contact:

Cirrax GmbH
https://cirrax.com

benedikt.trefzer@cirrax.com