LARGE SCALE OPEN SOURCE DEVELOPMENT MODELS

A COMPARATIVE ANALYSIS

By Joe Gordon

WHY

- Saw OpenStack grow from around 60 developers to over 2,100 developers
- Unusual development model
- But how do other projects solve the same problems?

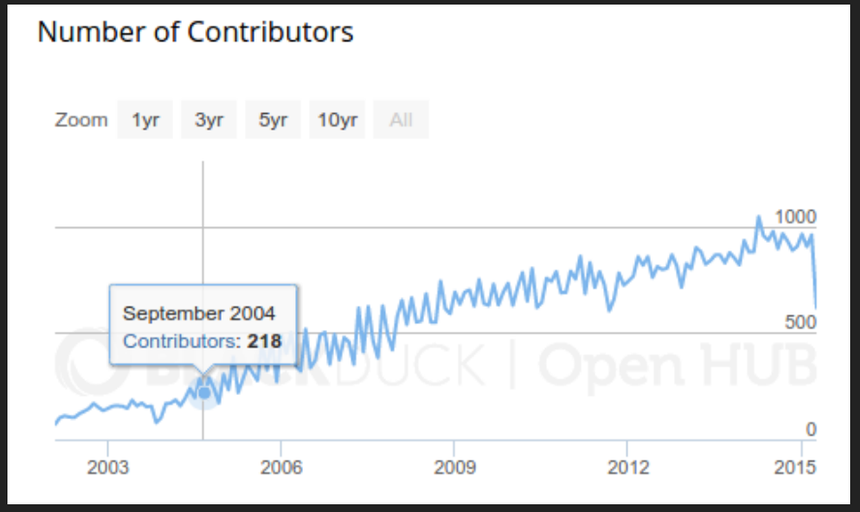
WHY IS PICKING THE RIGHT DEVELOPMENT MODEL IMPORTANT?

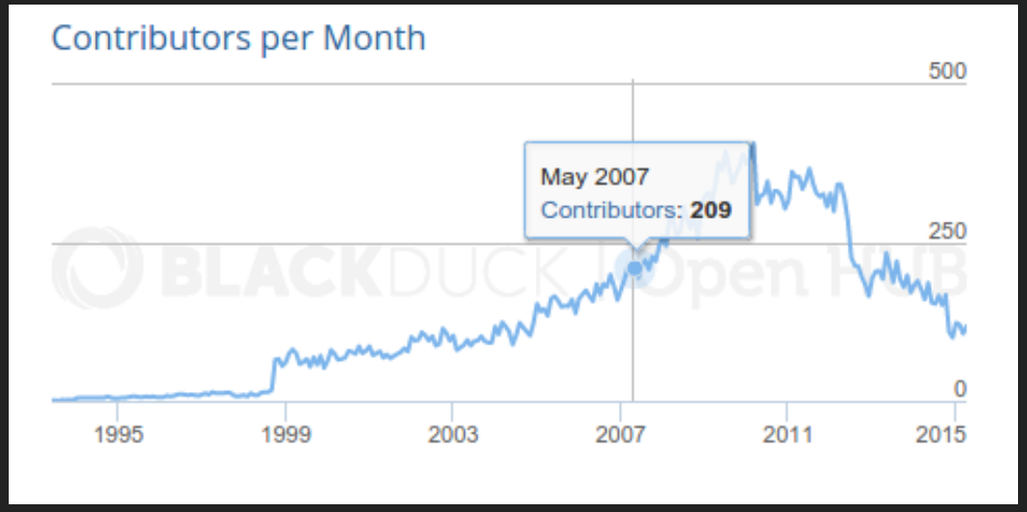
ACCELERATING GROWTH

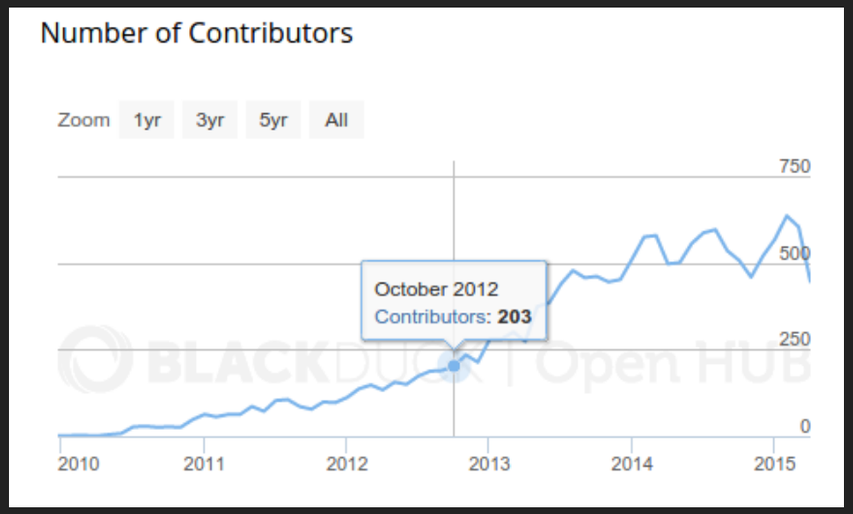
- Time till 100 contributors
  - Linux: 1991 - 2 years
    - Linux 2.0 had 190 contributors in 1996 in credits
  - OpenStack: 2010 - 1 year
  - Docker: 2013 - several months
    - over 300 contributors in its first year
- 200 contributors per month
  - Linux: 1991 - June 2004 (13 years)
  - Debian: 1993 - March 2007 (14 years)
  - OpenStack: 2010 - October 2012 (2 years)

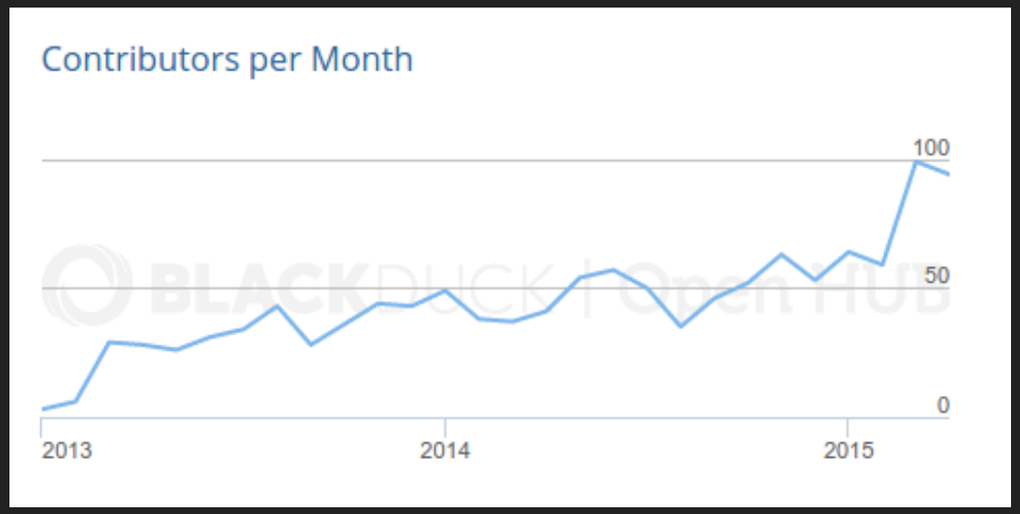
ACCELERATING GROWTH

Linux, Debian, Docker, OpenStack (clockwise from top left)
source: openhub









OPEN SOURCE IS BIG BUSINESS

- Open source is the new standard bodies
- Balancing corporate interests

Linux foundation Gold and Platinum Members

PICKING THE RIGHT DEVELOPMENT MODEL

CONWAY'S LAW

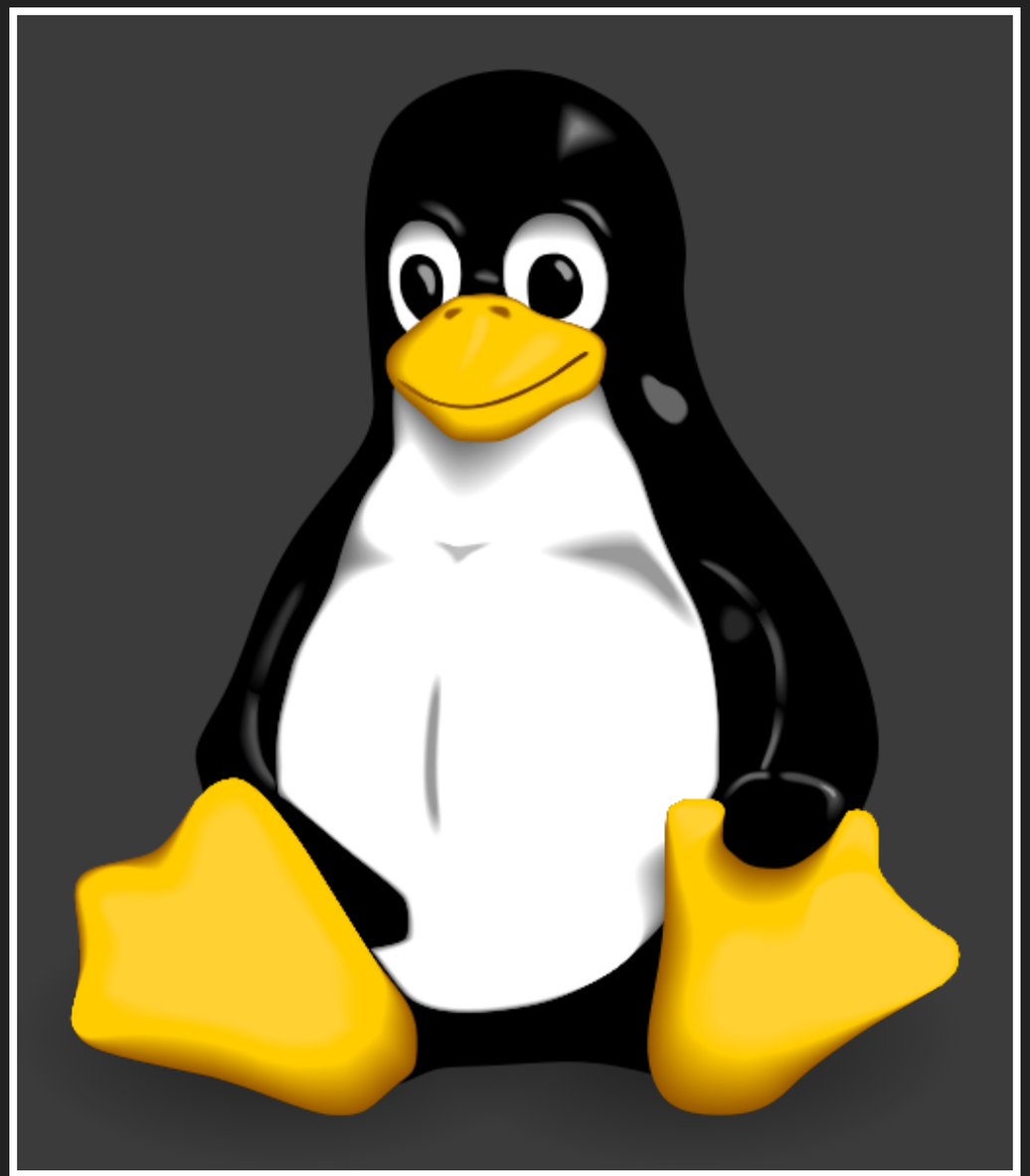
organizations which design systems ... are constrained to produce designs which are copies of the communication structures of these organizations

DEVELOPMENT MODELS

PROJECTS COVERED

- Linux Kernel
- Apache Software Foundation
- Debian
- OpenStack
- Docker

LINUX KERNEL



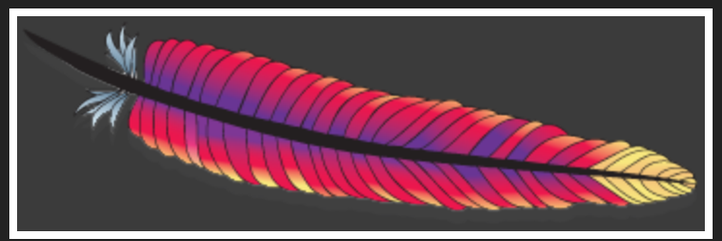
LINUX KERNEL

- Time based release
- 100-150 subsystem maintainers
  - Chain of trust
  - Decentralized review process
- Rolling development model

CULTURE

- Chain of trust
- About the individual
- Value frankness over politeness
- Corporate friendly
- No single company controls
- Not much automated pre commit testing
  - Failing testing is very bad for author

APACHE SOFTWARE FOUNDATION



APACHE SOFTWARE FOUNDATION

- ASF is more of a governance umbrella and culture
- Separate projects
  - 4,400+ committers
  - 150+ top level projects
- flat (ish) trust model
- 'Review then commit' vs. 'commit then review'

```
In order to reduce friction and allow for diversity to emerge, rather than forcing
a monoculture from the top ... each project is delegated authority over development
of its software, and is given a great deal of latitude in designing its own technic
charter and its own governing rules.
```

CULTURE

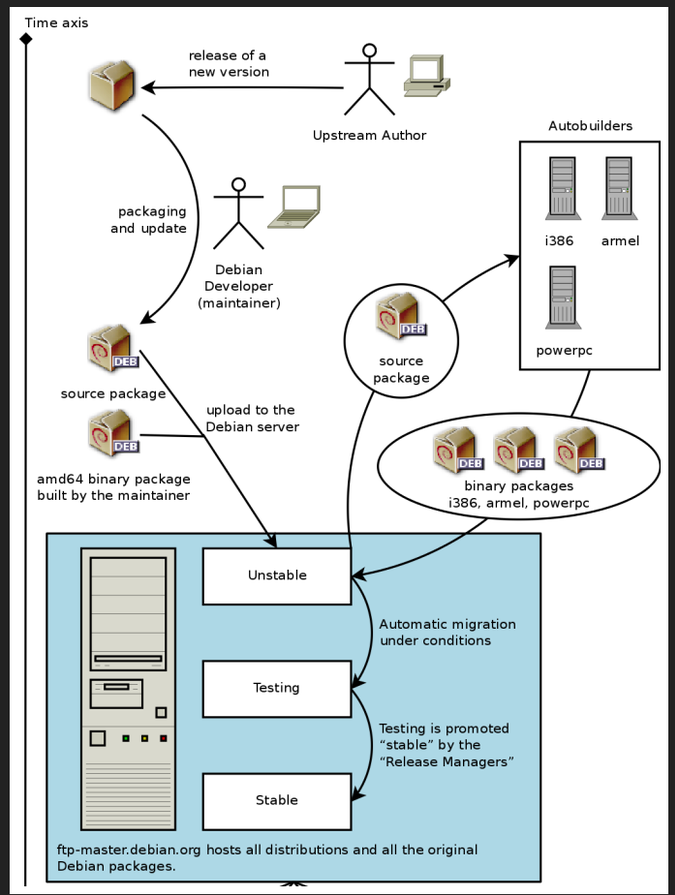
- Lazy consensus
- Focus is on the team
  - All decisions are team based
- Focus is on contributors not companies
- No monoculture

> *Within the ASF we worry about any community which centers around a few individuals who are working virtually uncontested.*

DEBIAN

DEBIAN

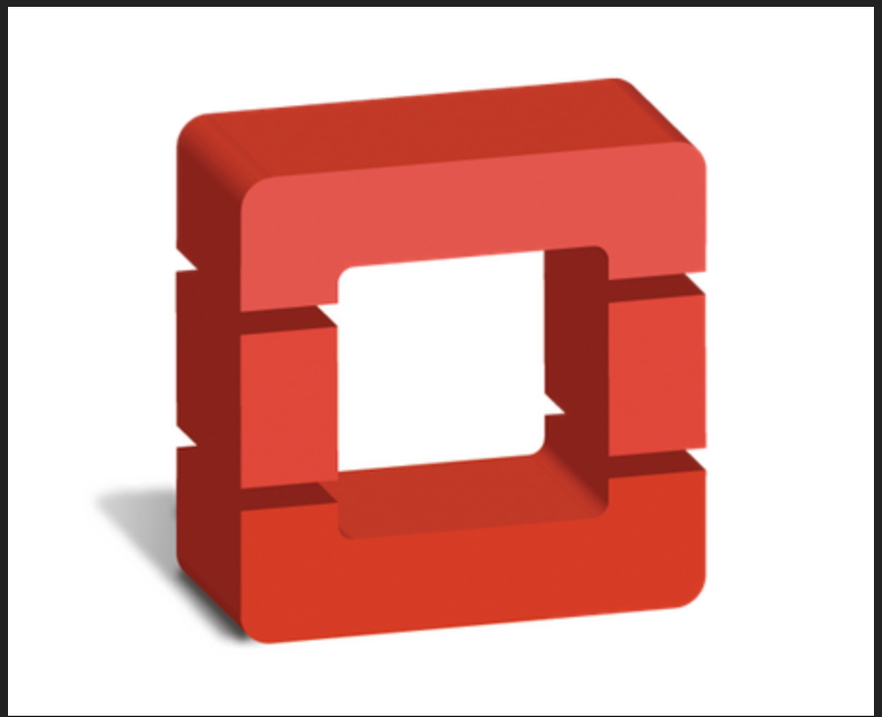
- When its ready, not time based. Notoriously slow
- Package Maintainers: No review, trust maintainers more



CULTURE

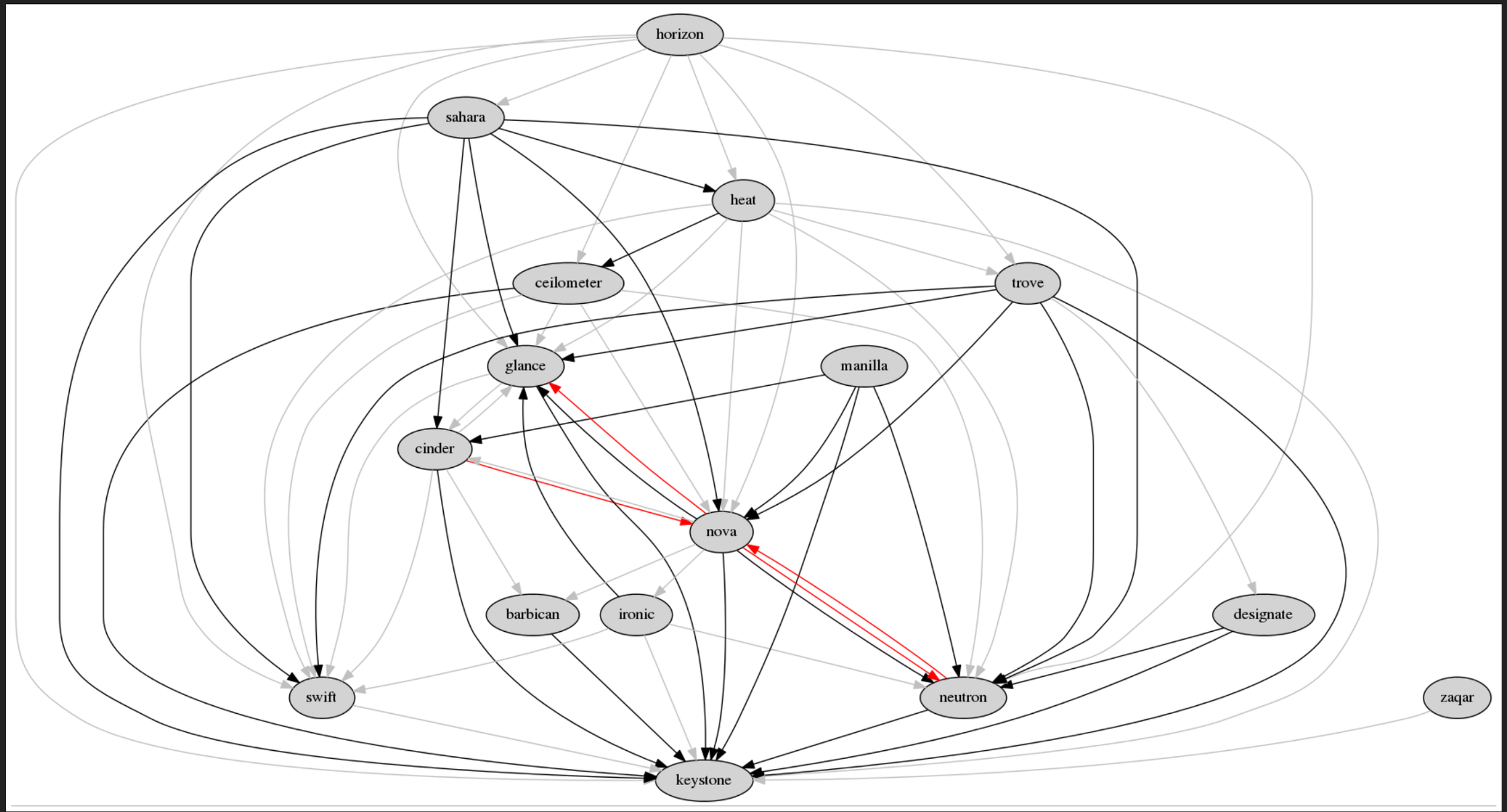
- Rotating leadership (elections)
- Do-ocracy: *An individual Developer may make any technical or nontechnical decision with regard to their own work*
- Open development
- Independent not 'profit-driven': *no imposed decisions by who has money, infrastructure, people*
- *no benevolent dictator, no oligarchy*
- It is all about the individual (although individual's can form groups)
- Territorial

OPENSTACK



OPENSTACK

- continuous delivery + stable releases
  - No rolling development
- Break down into services and build teams around each service
  - 5,000 commits per month from 500 contributors
- Strong centralized review process (two core reviews per patch)
- Automated testing to reduce reviewer burden
- Having trouble with scaling the team responsible for a single repository
  - Can't get past 15 or so members on a core team


horizon
sahara
heat
ceilometer
trove
glance
manilla
cinder
nova
barbican
ironic
designate
swift
neutron
zaqar
keystone

CULTURE

- Group over individual
- Egalitarian
- Elections
- Welcoming to new contributors
- Corporate friendly
- Not controlled by single company
- Lazy consensus
- Decentralized design
- Uniform tooling/process across projects

CULTURE

OPENSTACK'S 4 OPENS

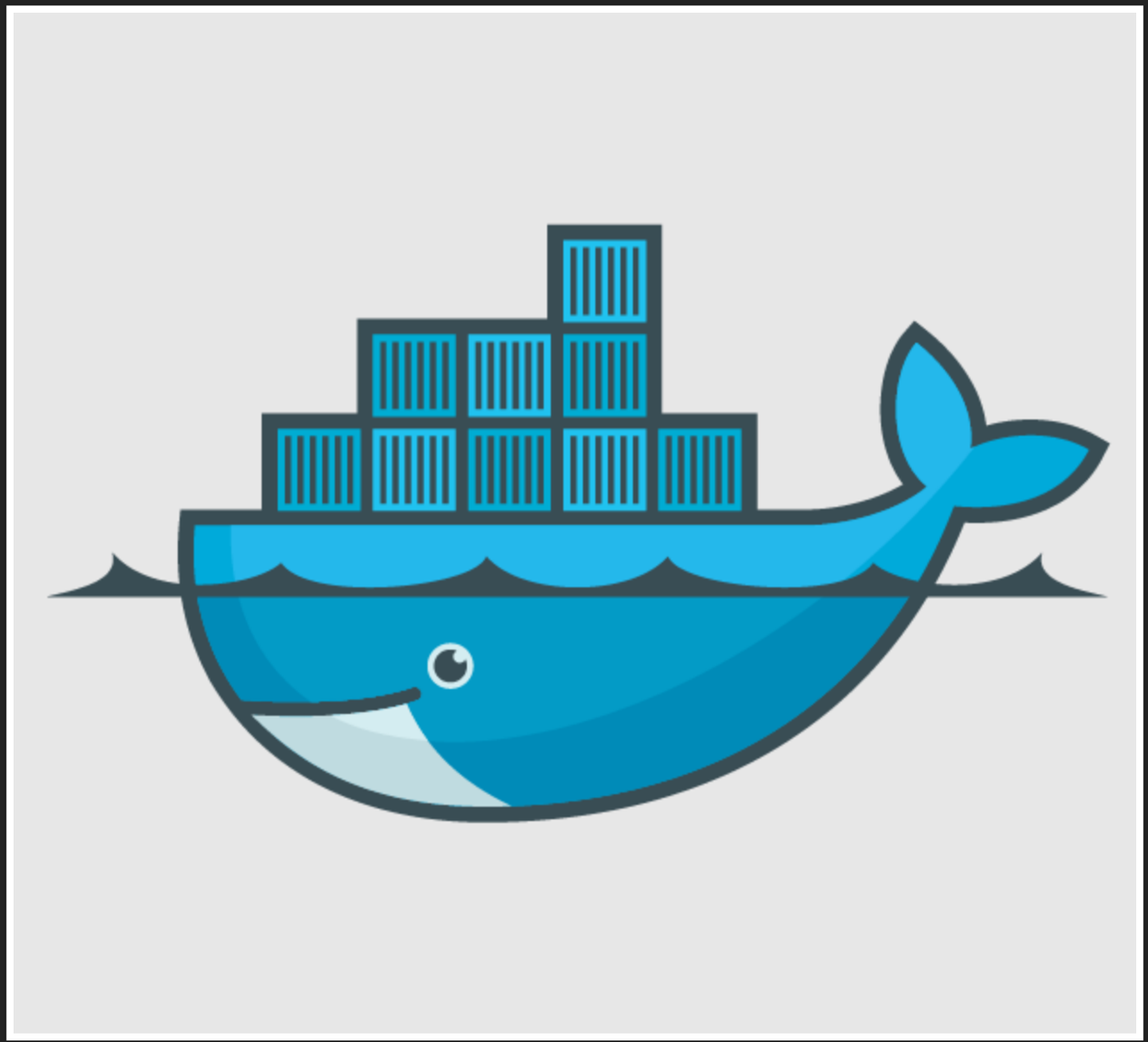
- Open *Source*, not open core
- Open *Design*
- Open *Development*
- Open *Community*
  - Lazy consensus
  - technical governance is a meritocracy
  - put everything in the public

FACTORS LIMITING GROWTH

- Cross project issues
- Team size
- Single vision

DOCKER

'Github' development model



DOCKER

- Release every 2 months from stable branch (master isn't frozen)
- Scaling
  - split into multiple repos
  - Maintainers / subsystem maintainers
  - 35+ maintainers in Docker

```
1) They share responsibility in the project's success.
2) They have made a long-term, recurring time investment to improve the project.
3) They spend that time doing whatever needs to be done, not necessarily what
is the most interesting or fun."

This "cellular division" is the primary mechanism for scaling maintenance of the
project as it grows.
```

BDFL

```
Ideally, the BDFL role is like the Queen of England: awesome crown, but not
an actual operational role day-to-day. The real job of a BDFL is to NEVER GO AWAY.
... the BDFL will always be there, preserving the philosophy and principles of the
project, and keeping ultimate authority over its fate. This gives us great
flexibility in experimenting with various governance models, knowing that we can
always press the "reset" button without fear of fragmentation or deadlock. See the
US congress for a counter-example.

BDFL daily routine:

* Is the project governance stuck in a deadlock or irreversibly fragmented?
    * If yes: refactor the project governance
* Are there issues or conflicts escalated by core?
    * If yes: resolve them
* Go back to polishing that crown.
```

CULTURE

- Embraces the BDFL
- Open source
- Open design
- Docker the company dominates
  - Most of the maintainers are docker employees
- Automated testing
- *Be Nice* and *Encourage diversity and participation*

THINGS TO CONSIDER FOR YOUR OWN PROJECT

There is no one size fits all solution

WHAT DOES *OPEN* EVEN MEAN?

- Open *Source*
- Open *Core*
- Open *Design*
- Open *Development*
- Open *Community*

TOOLING CONSIDERATIONS

- Bug tracking
- Review process
- Testing
- Barrier to entry for new contributors
  - DCO vs. CLA
    - Developer Certificate of Origin
    - Contributor Licensing Agreement
  - Overall workflow

DEVELOPMENT MODEL COMPONENTS

- Communication
- Team scaling model
  - chain of trust model
  - flat trust model
  - maintainers
- Release cadence
  - Stabilization periods
  - Rolling development
  - CI/CD
- Number of artifacts
- Decision making process -- Consensus model
- **Project culture**

PROBLEMS SPECIFIC TO OPEN SOURCE

- BDFL
- Vision
- Managing competing interests
- Corporate
- Consensus model
- Trust and Ownership
  - Team vs individual
  - First come first serve?
  - How do you fire someone?

THANK YOU

QUESTIONS?

Slides can be found at jogo.github.io/development-models

Powered by reveal.js