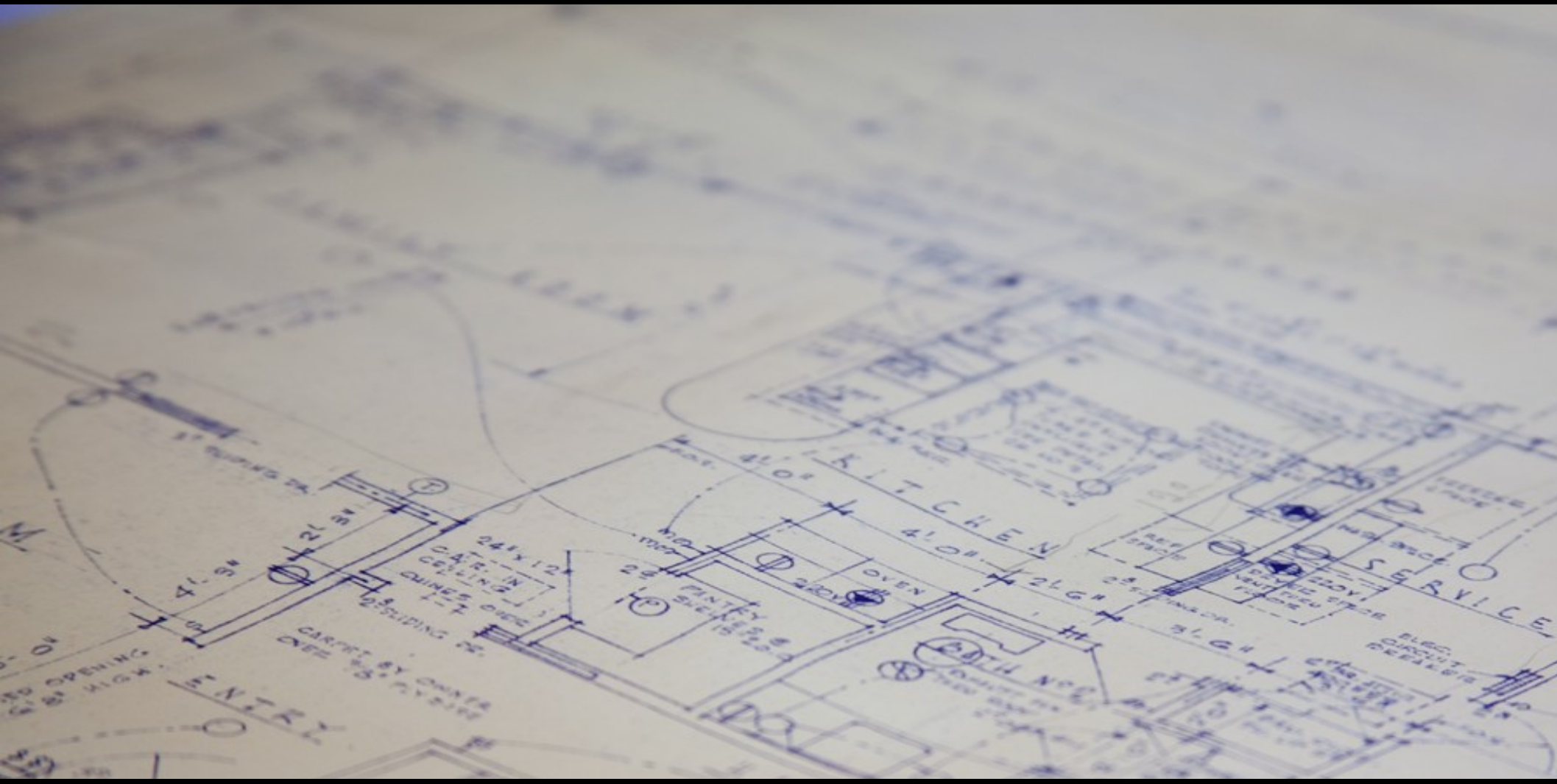
Infrastructure as an Open Source Project



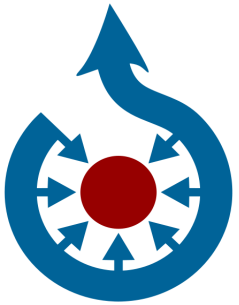
Ryan Lane

WIKINEWS

Mission

WIKIBOOKS

WIKIPEDIA
The Free Encyclopedia



WIKIVERSITY

Early Tech Community:
No staff

Current Ops Community:
No new volunteers

Editor Retention

Ops

Current Dev Community: Lack of staff/community collaboration

Open Infrastructure to Volunteers

Approach

Release puppet configuration

Main page
Recent changes
Server Admin Log

OpenStack
Manage Instances
Manage Addresses
Manage Security Groups
Manage Your SSH Keys
Manage DNS Domains
Manage Projects
Manage Global Roles
Manage Sudo Policies
Manage Puppet Groups

Help
Access
Git and Gerrit

Toolbox
What links here
Related changes
Upload file
Special pages
Printable version
Permanent link
Browse properties

Main Page

This is Wikimedia Labs. It is meant to make it easier for developers and system administrators to try out improvements to Wikimedia infrastructure, including MediaWiki, and to do analytics and bot work.

Number of projects:	47
Number of instances:	80
Amount of RAM in use (in MBs):	162,304
Amount of allocated storage (in GBs):	2,890
Number of virtual CPUs in use:	91

Contents [hide]

1 Documentation
1.1 General
1.2 Managing instances
1.3 Logging

Documentation [edit]

For a complete list of help pages see Content of help

General [edit]

- For information about terminology used in Labs, see terminology.
- For information about access and privileges, see access.
- For information about git and gerrit, see Git.

Managing instances [edit]

- For information about instances, see instances help.
- For information about security groups, see security groups page

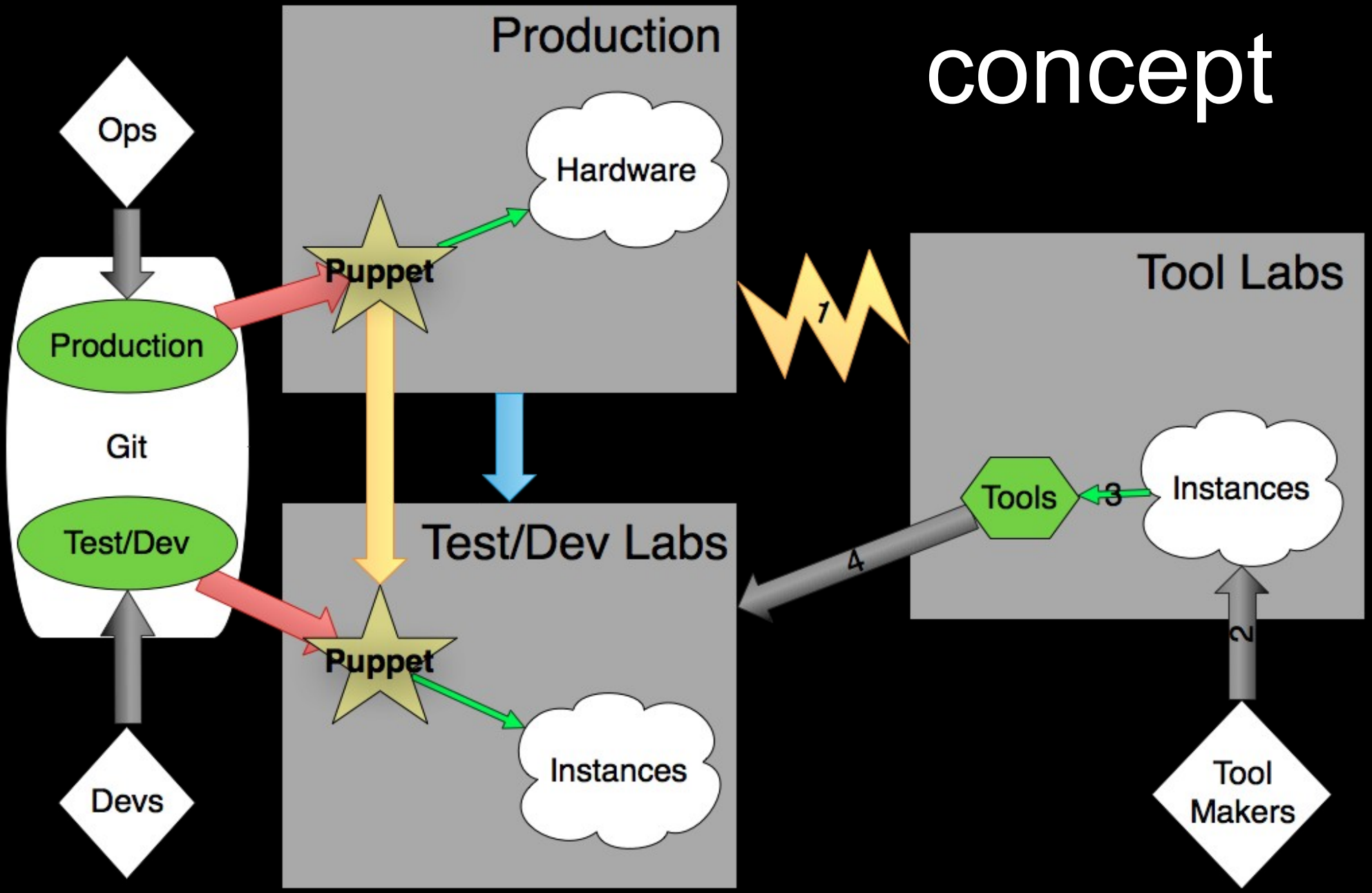
Logging [edit]

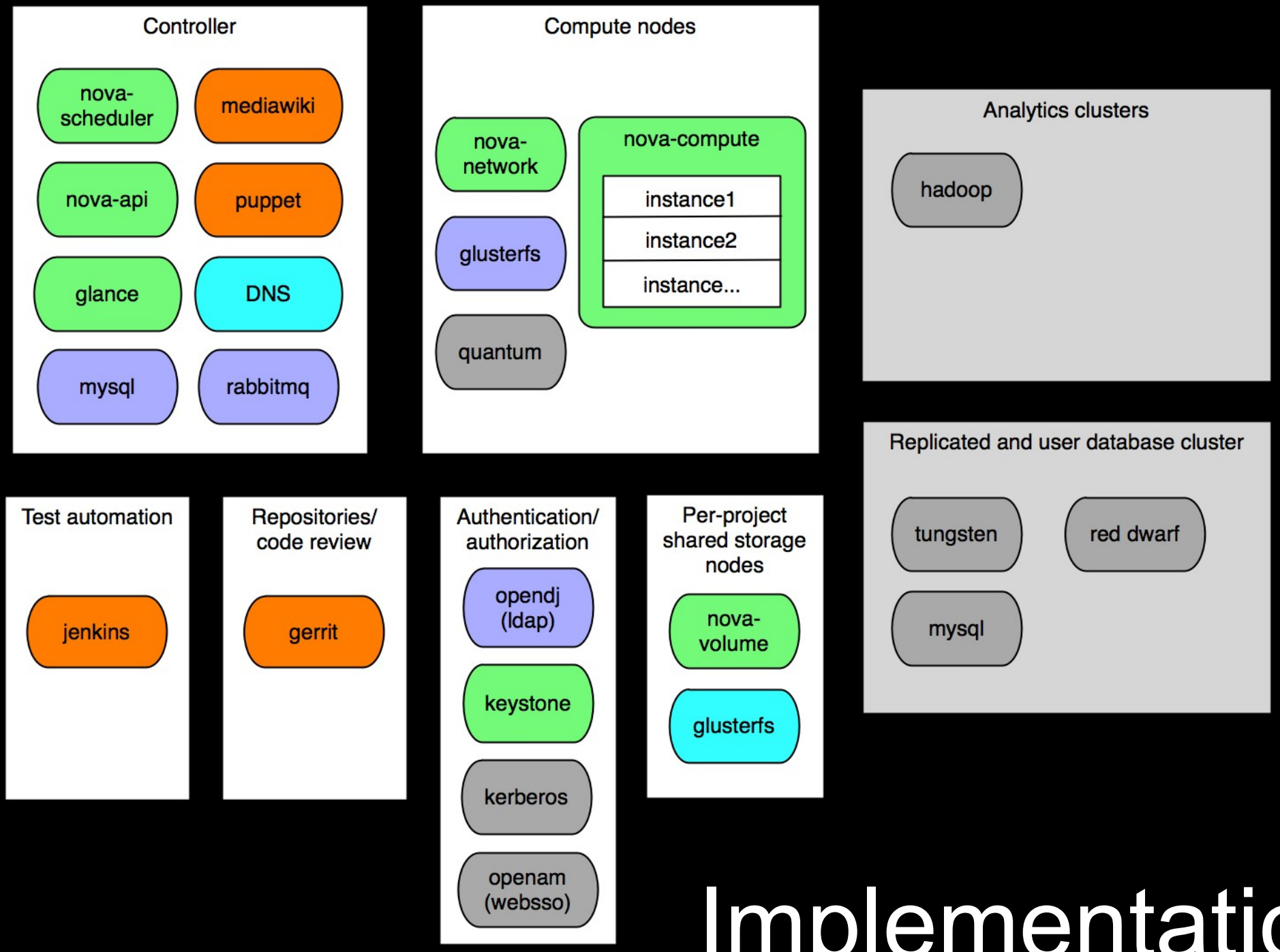
Every project has a Server Admin Log. The logs are combined together in a single Server Admin Log that displays logs for all projects, ordered by most recent activity. You can add a log entry in the #wikimedia-labs channel on Freenode by using the following command:

```
!log <projectname> <message>
```

It's best to log all changes to the instances in your project. Doing so makes it easier for others to follow your progress, and to more easily join your project and help out.

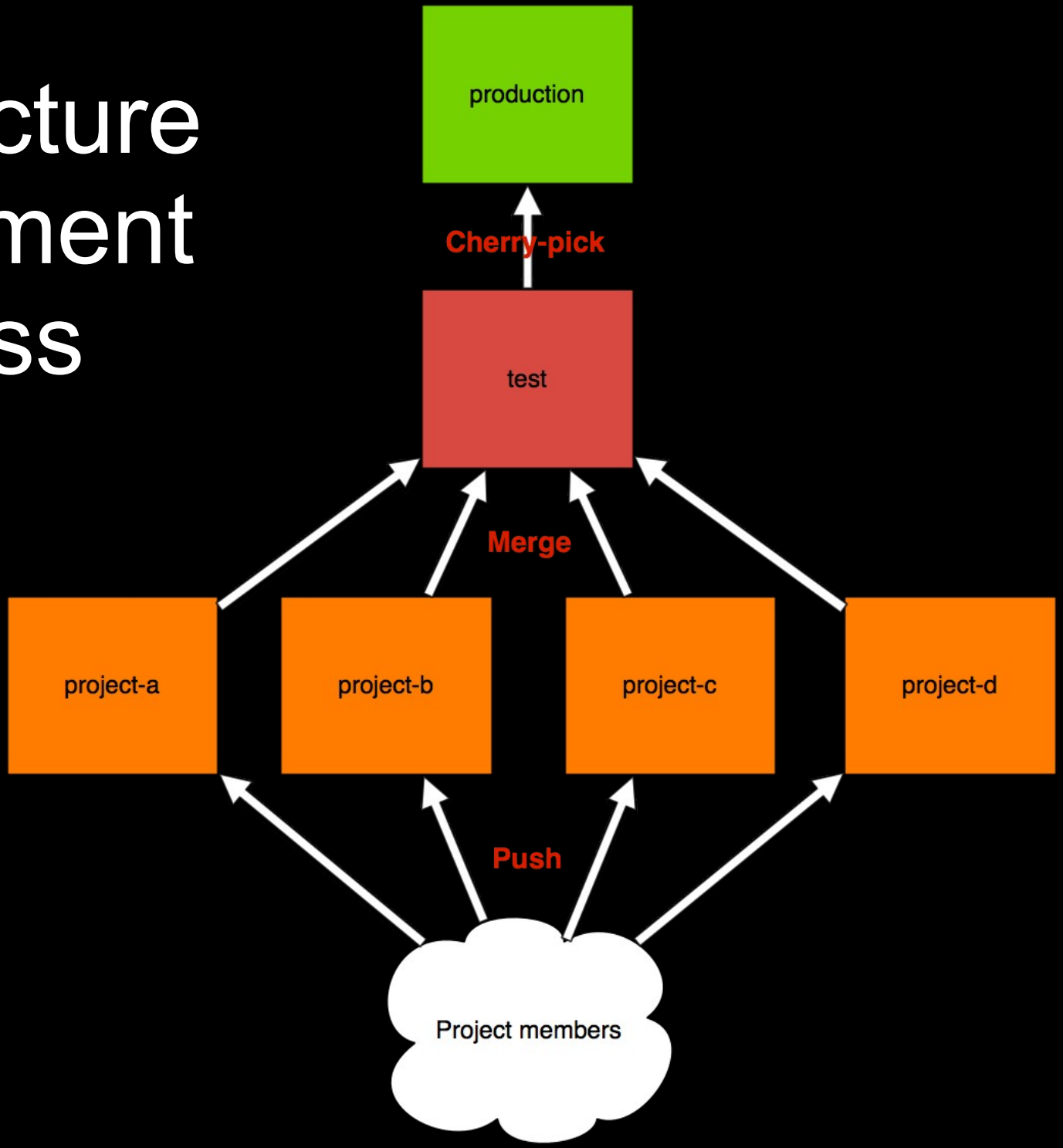
Initial concept

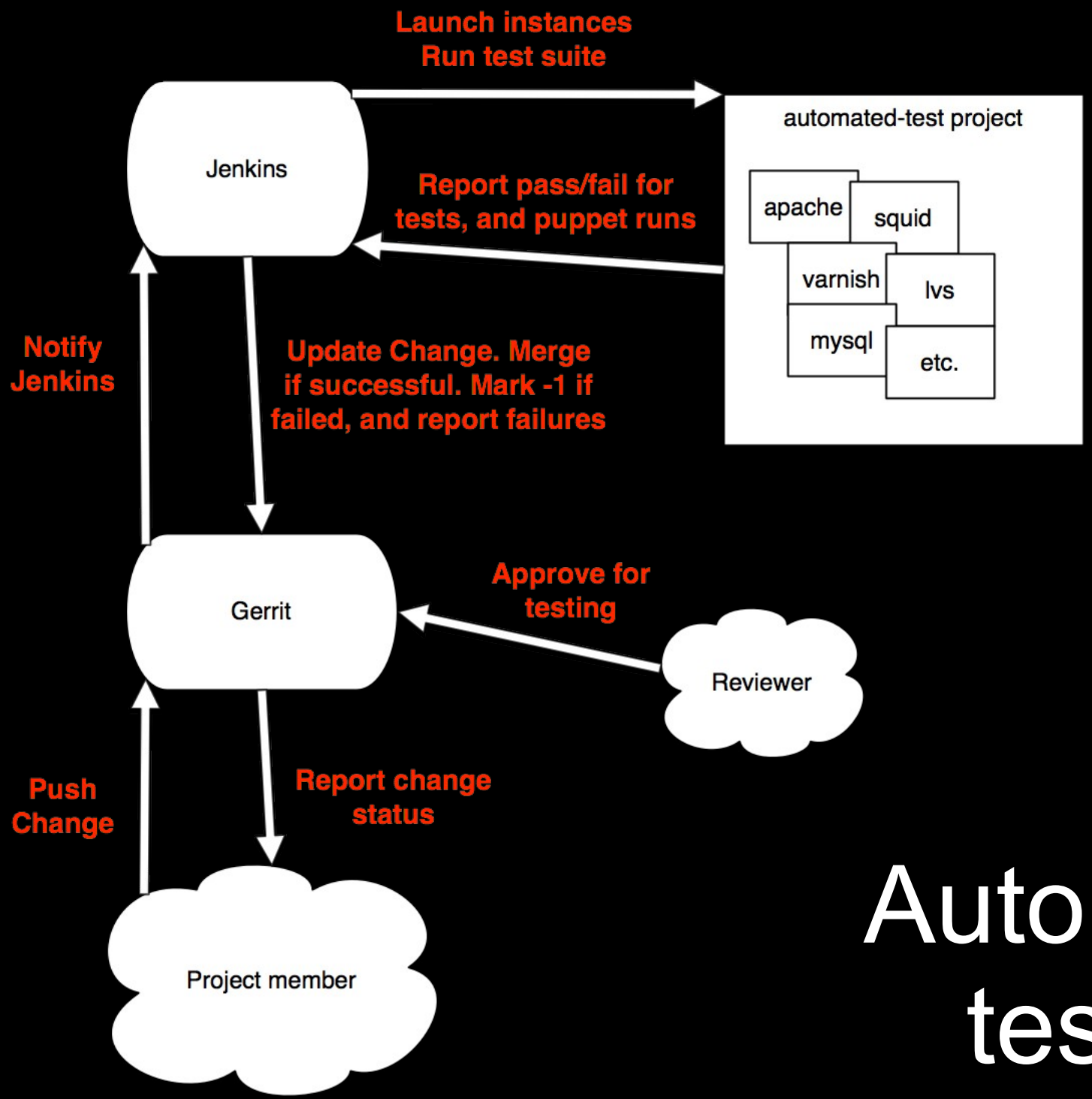



Controller
nova-scheduler
mediawiki
nova-api
puppet
glance
DNS
mysql
rabbitmq
Compute nodes
nova-network
nova-compute
instance1
instance2
instance...
glusterfs
quantum
Analytics clusters
hadoop
Replicated and user database cluster
tungsten
red dwarf
mysql
Test automation
jenkins
Repositories/ code review
gerrit
Authentication/ authorization
opendj (ldap)
keystone
kerberos
openam (websso)
Per-project shared storage nodes
nova-volume
glusterfs
Implementation

Projects.
Not customer accounts.

Infrastructure development process





Automated testing

Joining projects

Creating projects

Creating gerrit repositories

Following operations team procedures

Nova Resource:Deployment-prep

Contents [show]

Resource Type	project
Project Name	deployment-prep
Members	[Collapse] • User:MarkAHershberger • User:Petrb • User:Ryan Lane • User:Jeremyb • User:Oren • User:MaxSem • User:Krinkle • User:Platonides • User:RobLa • User:Tim Starling • User:TheDJ • User:Zaran • User:J • User:Mdale • User:Reedy

Documentation

Edit Documentation

Description

This project aims to replace the previously buggy en.labs.wikimedia.org and the different prototype wikis with a dedicated area just for testing of new code before it goes live on Wikimedia websites. It is intended to follow the different configuration that Wikimedia uses for its sites, along with the on-wiki MediaWiki configuration so as to allow developers to test code without breaking Wikimedia.

Server Admin Log

January 30

- 18:46 petan: configuring some boxes for cluster to handle high load

January 29

- 01:13 hexmode: oom reboot -web

January 27

- 13:30 j^: install upstart script /etc/init/timedmediahandler.conf on deployment-transcoding and start service
- 13:05 j^: touch /etc/wikimedia-image-scaler on deployment-transcoding; transcoding needs more wgMaxShellMemory too
- 12:47 Platonides: updated /usr/local/apache/common/live/extensions/TimedMediaHandler to r110117 per j^request
- 10:11 j^: add-apt-repository ppa:j/timedmediahandler and update ffmpeg on deployment-web to support frame extraction from WebM videos
- 06:35 j^: update ffmpeg on deployment-transcoding (new security release from ppa)

Nova Resource:I-00000130

Resource type	instance
Instance Name	driver-dev3
Reservation Id	r-acru4kqf
Instance Id	i-00000130
Instance State	running
Instance Host	virt2
Instance Type	m1.small
RAM Size	2048
Number of CPUs	1
Amount of Storage	20
Image Id	ami-00000023
Project	gluster
Availability Zone	nova
Region	
Security Group	default, web
Launch Time	2012-02-04T00:38:43Z

Actions

delete

reboot

configure

get console output

Main page
Recent changes
Server Admin Log

OpenStack
Manage Instances
Manage Addresses
Manage Security Groups
Manage Your SSH Keys
Manage DNS Domains
Manage Projects
Manage Global Roles
Manage Sudo Policies
Manage Puppet Groups

Help

Toolbox

Visit the main page

Recent changes

Track the most recent changes to the wiki on this page.

Main page
Recent changes
Server Admin Log

OpenStack
Manage Instances
Manage Addresses
Manage Security Groups
Manage Your SSH Keys
Manage DNS Domains
Manage Projects
Manage Global Roles
Manage Sudo Policies
Manage Puppet Groups

Help

Toolbox

Recent changes options

Show last **50** | 100 | 250 | 500 changes in last **1** | 3 | **7** | 14 | 30 days
Hide minor edits | Show bots | Hide anonymous users | Hide logged-in users | Hide my edits
Show new changes starting from 12:03, 4 February 2012

Namespace: all | Invert selection | Associated namespace | Go

4 February 2012

- (diff | hist) . . N Nova Resource:I-00000130; 00:40 . . **(+682)** . . 10.4.0.34 (Talk)
- (Deletion log); 00:38 . . Andrew Bogott (Talk | contribs) deleted "Nova Resource:I-0000012f"
- (Deletion log); 00:26 . . Andrew Bogott (Talk | contribs) deleted "Nova Resource:I-00000124"

3 February 2012

- (diff | hist) . . N Nova Resource:I-0000012e; 23:10 . . **(+683)** . . 10.4.0.34 (Talk)
- (diff | hist) . . N Nova Resource:I-0000012d; 21:14 . . **(+661)** . . Declerambaul (Talk | contribs)
- (diff | hist) . . Nova Resource:Bastion; 21:06 . . (+18) . . Pyoungmeister (Talk | contribs)
- (diff | hist) . . User:Jeremyb; 20:57 . . (+117) . . Jeremyb (Talk | contribs)
- (diff | hist) . . N User:Jeremyb; 20:49 . . (+41) . . Jeremyb (Talk | contribs) *(Created page with "Wikimedia SUL: wikipedia:user:jeremyb")*
- (diff | hist) . . Requests; 20:47 . . (+159) . . Jeremyb (Talk | contribs) *(→Request for project creation:)*
- (diff | hist) . . Git; 20:16 . . (-44) . . Diederik (Talk | contribs) *(→Windows:)*
- (diff | hist) . . Git; 20:15 . . **(+1,422)** . . Diederik (Talk | contribs) *(Added windows instructions)*
- (diff | hist) . . Nova Resource:Mobile-stats; 19:11 . . (+18) . . Diederik (Talk | contribs)

2 February 2012

- (Deletion log); 20:37 . . Schmir (Talk | contribs) deleted "Nova Resource:I-0000012c"
- (diff | hist) . . m Git; 19:50 . . (+38) . . Hashar (Talk | contribs) *(→Git/Gerrit and the puppet repositories:)*

Nova Resource:I-00000078

Feb 2, 2012 (2 days ago)

by Schmir

← Older revision	Revision as of 10:46, 2 February 2012
Line 6:	Line 6:
|Private IP=10.4.0.28	|Private IP=10.4.0.28
|Public IP=10.4.0.28	|Public IP=10.4.0.28
− |Instance State=~~networking~~	+ |Instance State=running
|Instance Host=virt2	|Instance Host=virt2
|Instance Type=m1.small	|Instance Type=m1.small
Line 19:	Line 19:
|Launch Time=2011-11-09T23:50:08Z	|Launch Time=2011-11-09T23:50:08Z
|FQDN=i-00000078.pmtpa.wmflabs	|FQDN=i-00000078.pmtpa.wmflabs
− |Puppet Class=base,ldap::client::wmf-test-cluster,exim::simple-mail-sender	+ |Puppet Class=base,ldap::client::wmf-test-cluster,exim::simple-mail-sender,sudo::labs_project,generic::packages::git-core
− |Puppet Var=realm=labs,instancecreator_username=Ryan Lane,instancecreator_lang=en,instanceproject=pediapress,}}	+ |Puppet Var=realm=labs,instancecreator_username=Ryan Lane,instancecreator_lang=en,instanceproject=pediapress,instancename=labs-ocg1,}}

+1 0 Share Email Keep unread Add tags

Nova Resource:I-0000012a

Feb 2, 2012 (2 days ago)

by 10.4.0.34

New page

!log <project> <message>

Server Admin Log

Projects are listed in order of most recently updated.

Also see the recent changes for nova resources (atom).

Contents [hide]

1 Nova_Resource:Incubator/SAL
- 1.1 February 2
- 1.2 February 1
- 1.3 January 29
- 1.4 January 28
- 1.5 January 27
- 1.6 January 25
- 1.7 January 24

2 Nova_Resource:Bastion/SAL
- 2.1 February 1
- 2.2 January 22
- 2.3 January 18
- 2.4 January 13
- 2.5 January 10

3 Nova_Resource:Deployment-prep/SAL
- 3.1 January 30
- 3.2 January 29
- 3.3 January 27
- 3.4 January 26
- 3.5 January 25
- 3.6 January 24
- 3.7 January 23
- 3.8 January 22
- 3.9 January 16
- 3.10 January 15
- 3.11 January 14
- 3.12 January 13

Main page
Recent changes
Server Admin Log

OpenStack
- Manage Instances
- Manage Addresses
- Manage Security Groups
- Manage Your SSH Keys
- Manage DNS Domains
- Manage Projects
- Manage Global Roles
- Manage Sudo Policies
- Manage Puppet Groups

Help

Toolbox

#wikimedia-labs

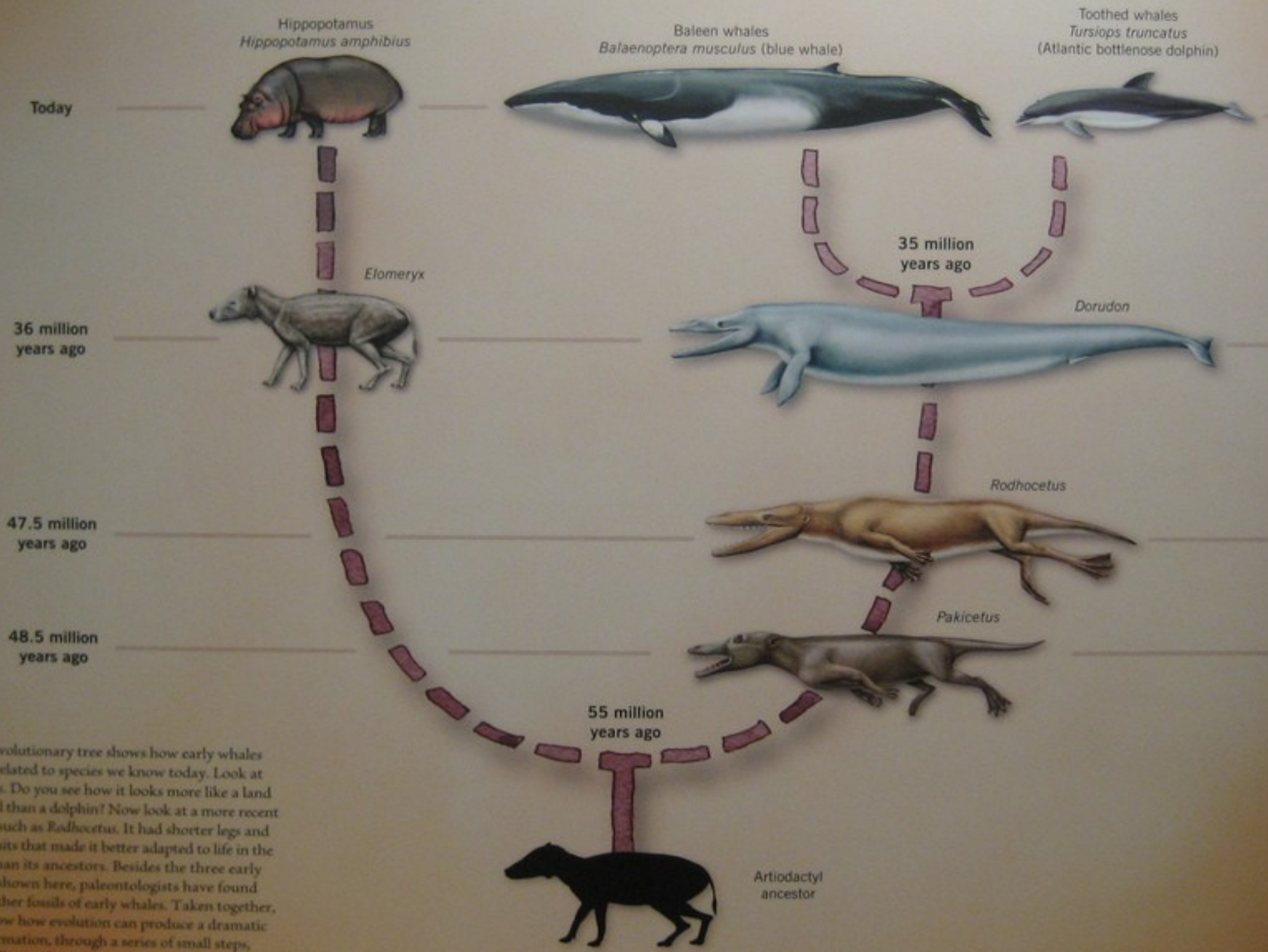
labs-l

Community reception

47 projects
80 instances
105 users

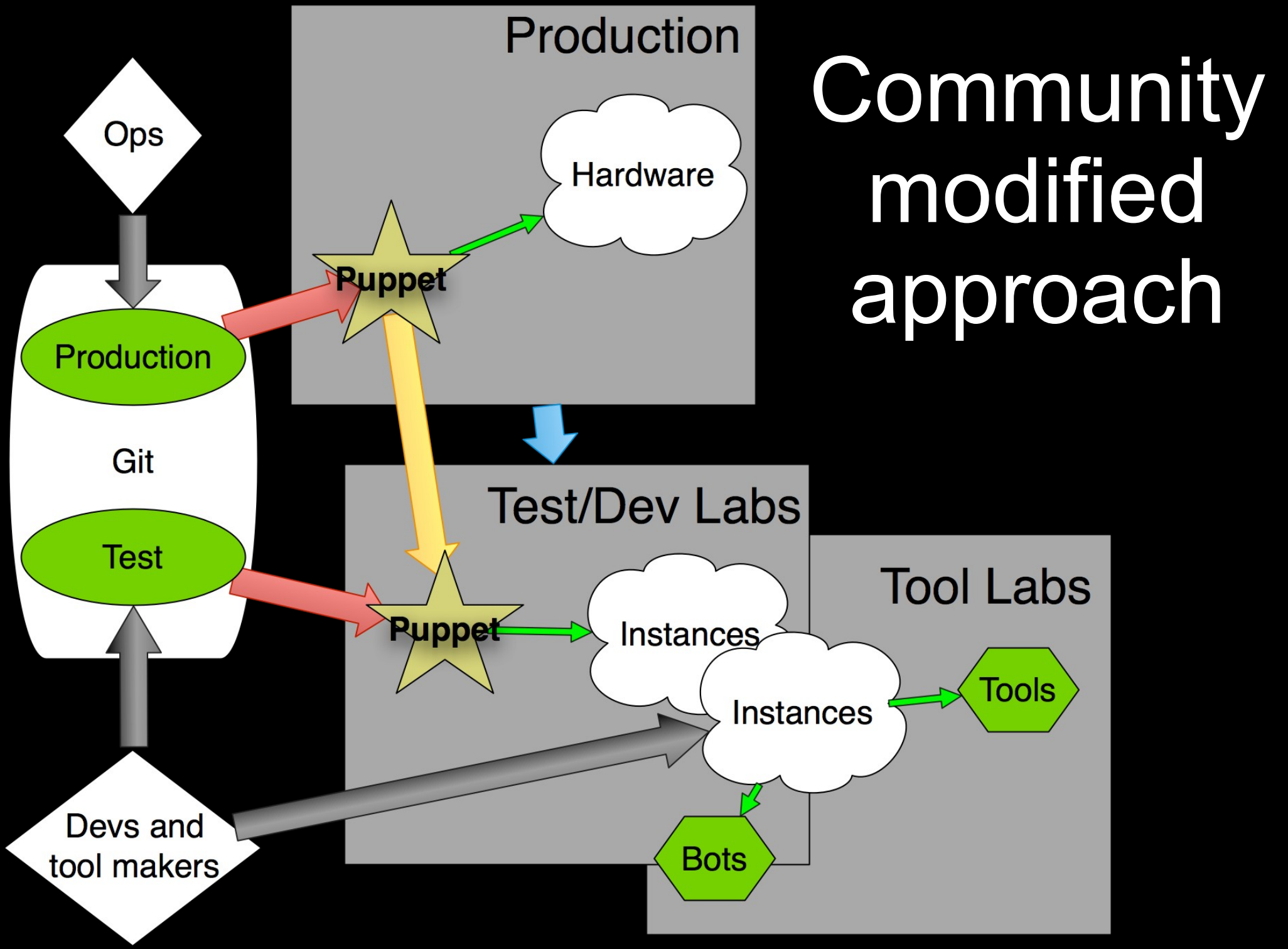
Nagios monitoring
Bot infrastructure
Clone of production environment
Jenkins and Testswarm deployments

Hackathon use



This evolutionary tree shows how early whales are related to species we know today. Look at *Pakicetus*. Do you see how it looks more like a land mammal than a dolphin? Now look at a more recent species, such as *Rodhocetus*. It had shorter legs and other traits that made it better adapted to life in the water than its ancestors. Besides the three early species shown here, paleontologists have found many other fossils of early whales. Taken together, they show how evolution can produce a dramatic transformation, through a series of small steps, over millions of years.

Community modified approach



Problems

Shift of operations bottleneck

Giving out root is dangerous

Must audit infrastructure

Must prevent escalation

Application of this model elsewhere

Get Involved!

- Help with development
- Help with operations
- Help with community

Ryan Lane
Wikimedia Foundation
ryan@wikimedia.org

IRC: Ryan_Lane on Freenode; channels:

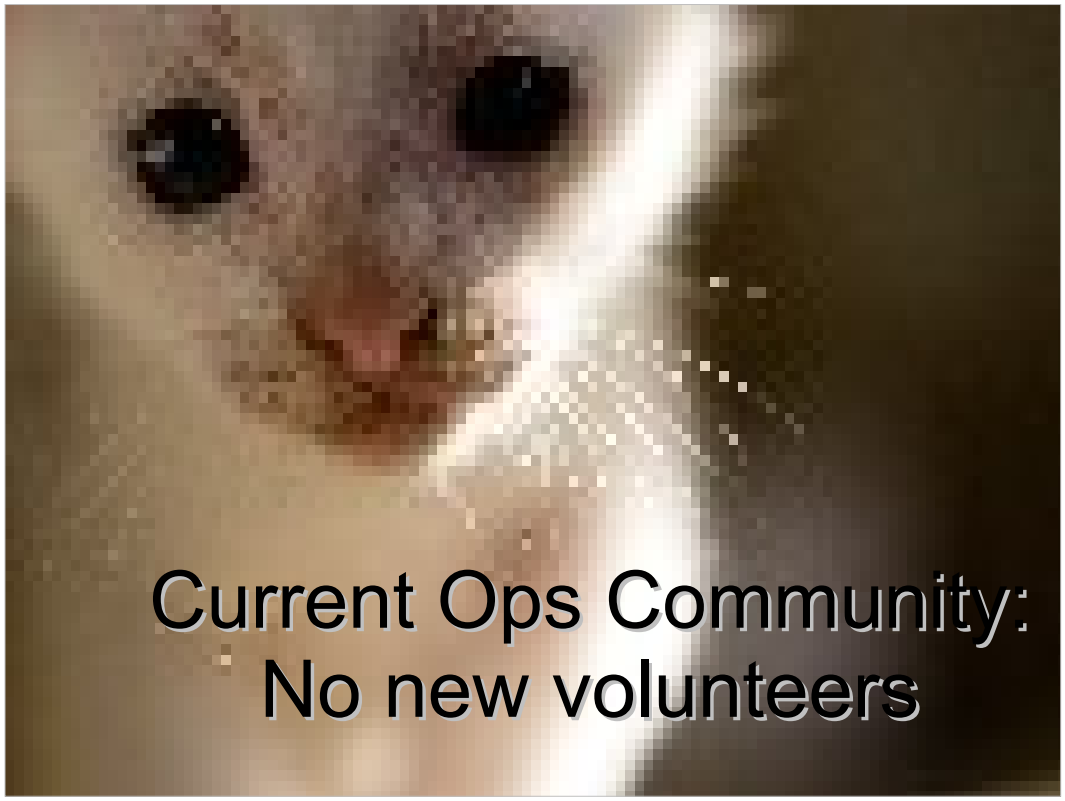
- #mediawiki
- #wikimedia-labs
- #wikimedia-tech
- #openstack
- #openstack-dev

For the past year I've been working on a project named Wikimedia Labs. Labs is a cloud computing environment for development, testing, and deployment of Wikimedia's infrastructure as a whole. Our infrastructure is being treated as an open source software project. This enables us to invite our community to contribute to how our sites are run.

Wikimedia is kind of a weird and interesting movement. It's built and maintained by volunteers. Everything is Open Source and Open Content. The software and infrastructure, however, is mostly maintained by paid staff of the Wikimedia Foundation. This hasn't always been the case.

When we started, volunteers had root. They were the only roots. Most of the infrastructure they built is still in use today.

Our lenient access policy made us flexible. Changes could happen quick. Also, the site was smaller, and had far fewer users. Changes could be made in large fundamental ways in production.



Growth has made us less willing to give out access to volunteers. Because of the size of our sites, downtime is less acceptable. Having less volunteers means we have less ideas, and due to that our ability to make changes quickly is lessened.

We haven't had a new volunteer root in years. We haven't even had a new volunteer with shell access.

Engaging volunteers and enabling them to easily contribute is a wider problem as well.

There's a trend that's been occurring for the past few years. Our active editors on the projects have been slowly declining. The movement wishes to reverse this trend, and has made it a priority.

To reverse this trend we need to deploy more code, more frequently to test usability improvements, and we need infrastructure to measure the results of these changes, and we need to engage our community very well.

Our software development community scales with volunteers. Unfortunately, operations doesn't scale right now. We're stuck with the staff operations engineers we currently have. The staff is good, but...

The fact that operations can't scale to meet the needs of a large growth of developers means that operations is a bottleneck.

Furthermore, our access policy prevents volunteer developers from learning how our infrastructure works.

http://www.flickr.com/photos/tswestendorp/4631910301/sizes/o/in/photostream/

This leads to a situation where our staff developers and volunteer developers can't easily collaborate. Our volunteers also have no way of appropriately testing their changes, since our infrastructure is complex and diff cult to replicate. This means it's harder to take contributions, which further slows the pace of changes on our sites.

The solution we are taking for this is to open our infrastructure up as widely as possible. We are goingt to make it possible for volunteers to use and modify our infrastructure like they do our software.

Opening up infrastructure development to the software development community allows developers to bypass operations for work when operations is too busy to get to a task quickly enough. It also makes it possible to take contributions from volunteer operations engineers.

Operations work can be done via conf guration management tools, orchestration, and cloud computing.

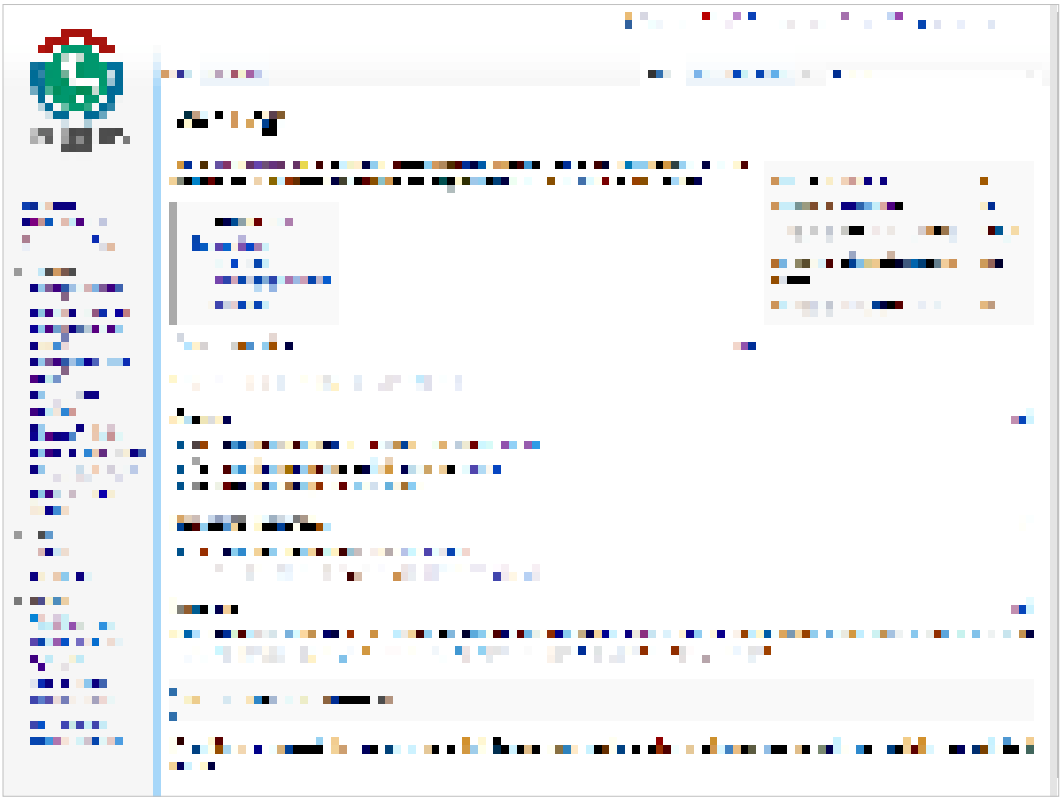
Thanks to this, operations can be treated like a software development project. If it can be a software project, then we can make it an open source project.

However, going staff-only to volunteer-based is hard. We still have the same concerns about site availability, so we must take a new approach.

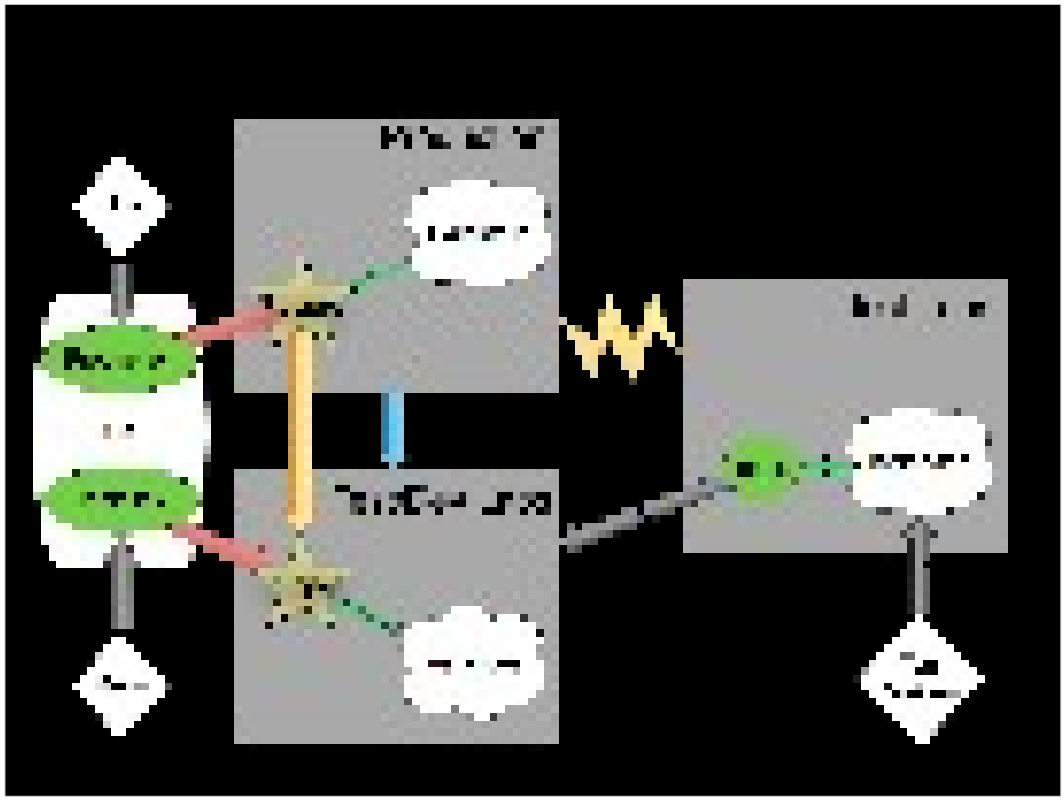
http://www.flickr.com/photos/nanoprobe67/5761031999/sizes/o/in/photostream/

Pause. Water.

Our first step was to release puppet configuration. We spent nearly a month going through the configuration, pulling out sensitive info, ensuring we don't release anything that would open us to major vulnerabliltes. This was difficult. We never expected to do this, so we didn't bother to separate sensitive configuration from non-sensitive information. We nearly released a star certificate for wikimedia.org! Thankfully a final sanity check through the configuration caught this.



Next we create an infrastructure where volunteers can test their work, document it, and eventually have it deployed live to production. We want to build a self-sustaining operations infrastructure community using this infrastructure.

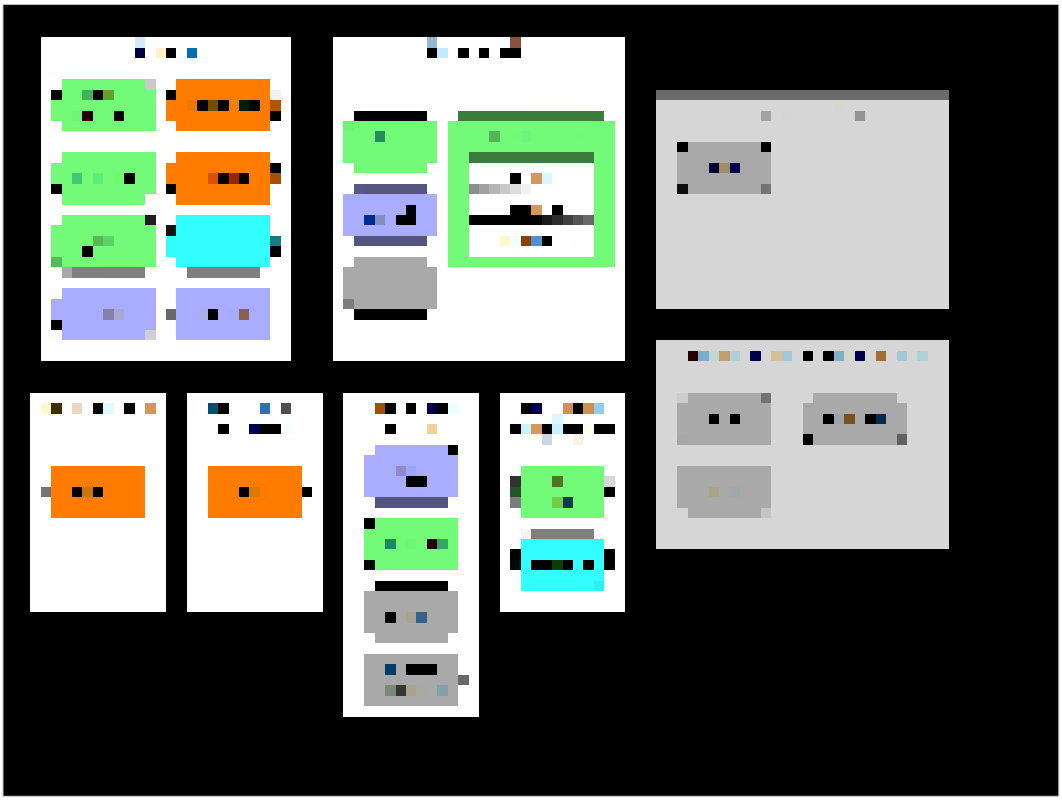


Our initial concept had three environments. Production, Tool Labs, and Test/Dev Labs. The environment was architected this way to provide a ramp to developers.

We currently have a an environment, called toolserver, for members of our community to write tools that don't perform well enough to run in our production cluster, and for bots to run to edit our sites. Tool Labs is loosely based on that concept, but is conceived to be an easier to use environment, where most operations tasks were already handled the development environment isn't as complex as production.

For our software development community and our operations volunteers we'd have another environment that is conf gured as closely to the production cluster as possible. In this environment developers and operations engineers should be able to build infrastructure, test it, puppetize it, then take those changes and push them into the test branch for review, and integration testing

Successful changes would be cherry-picked across to production, then deployed.



This is our current and planned infrastructure. This is a tightly integrated cloud computing cluster, using OpenStack components, MediaWiki, GlusterFS, OpenDJ (LDAP), MySQL, RabbitMQ, Puppet, Gerrit, and Jenkins.

Planned services include Kerberos, OpenAM (WebSSO), Quantum, tungsten, red dwarf, and hadoop.

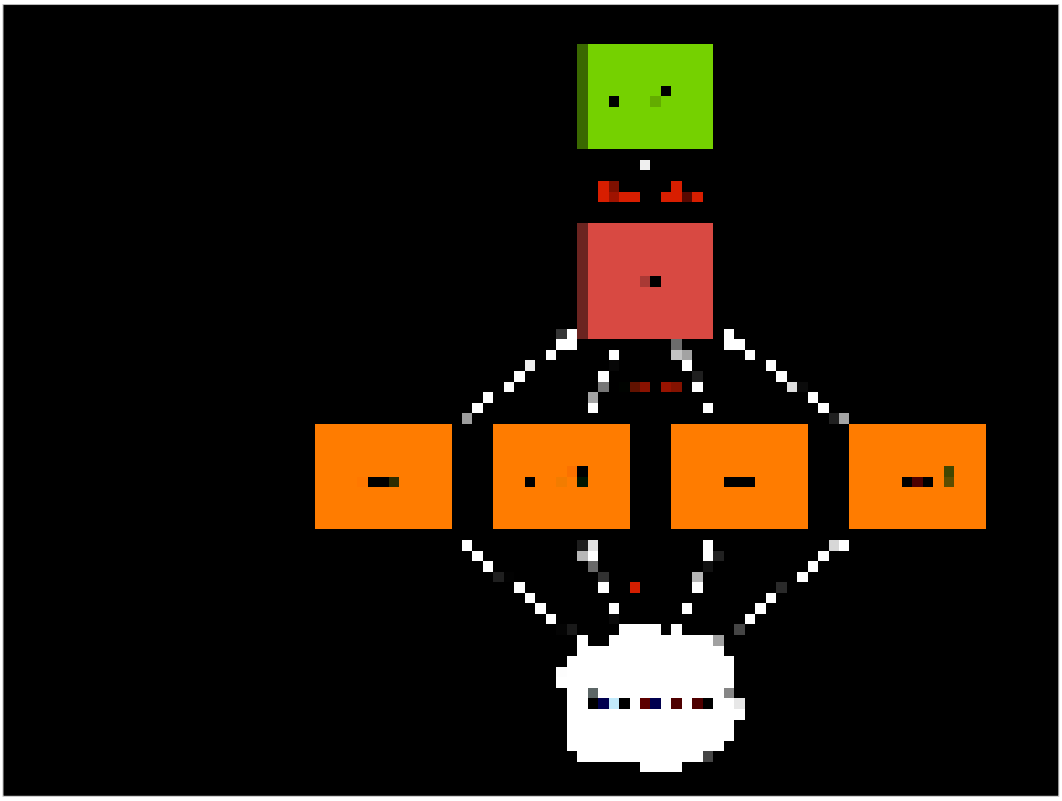
Rather than using Nova projects as means to separate customers, we use it to separate real world projects. These separations are meant to be very loose. Projects are meant to work together as needed.

To facilitate the looseness of the separation, the project membership is community managed, and it's easy to be added to projects.

Projects can be treated as specific or global. Specific projects are treated like real world projects (like gerrit, openstack, deployment-prep, etc). Global projects are projects whose resources are shared with all projects. In specific projects, all members have root. In global projects, by default only the operations team has root. Root can be explicitly given to non-operations team members in global projects.

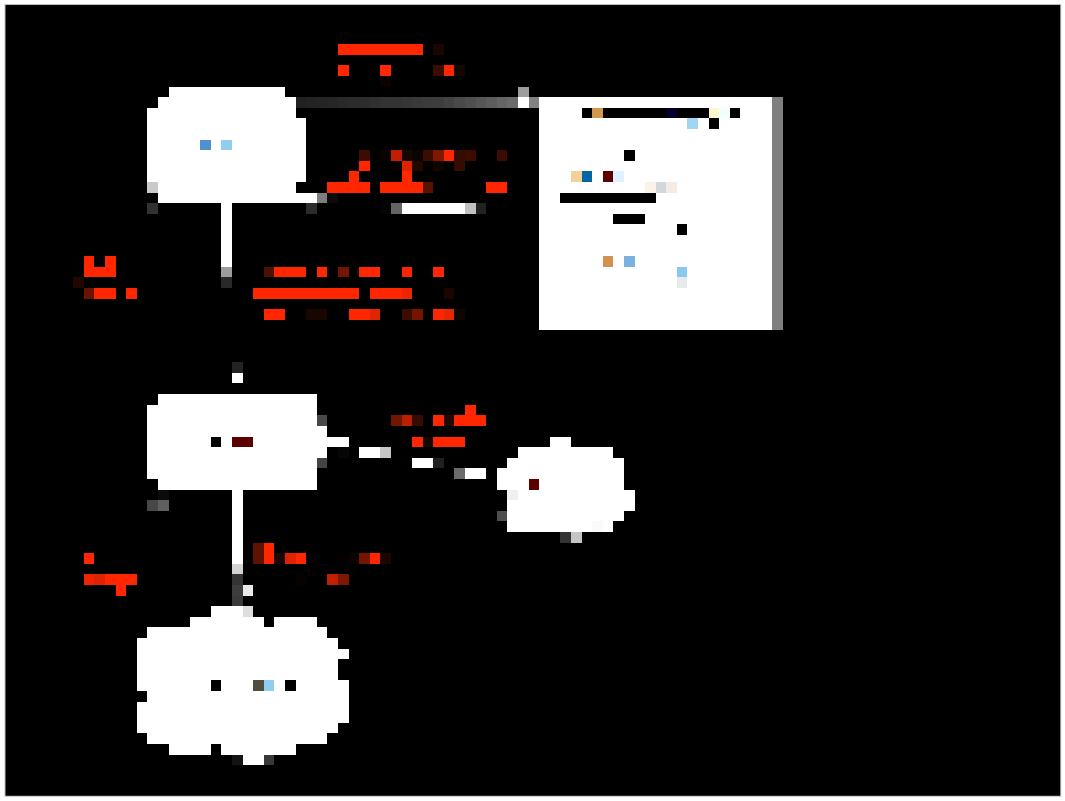
We treat resources in a per-project way, rather than a per-instance way. We do this since we assume every instance in a project should be able to access the same set of resources.

We also extend the concept of a project to the instances. Projects are in kept in LDAP, and are posix groups as well as nova projects. This allows us to use the projects in the instances for security separation. For instance, in a global project, you could use the project as a way to manage filesystem permissions.



Te initial concept of using a test and production branch to manage everything was f awed. It makes it diff cult to make changes in one project that don't affect another. This means we need to code review every change before it can be tested. It doesn't scale. So, we are tweaking the concept so that every project has a branch.

For this, we'll decentralize puppet - no puppet master. Every instance will run puppet locally, with a checkout of the test branch. Project members can switch branches on their instances. Changes to project branches can bypass review. When a project has puppetized a service fully, and is ready to test it in a production-like environment, the changes can be pushed from the project branch into the test branch, code reviewed, merged, and tested. If testing in a production-like environment goes well, the changes can be cherry-picked into the production branch, and deployed live.



A side effect of handling changes this way is that we can run automated infrastructure tests. Using Jenkins and Gerrit, we can have the following procedure:

1. After a change has been tested locally in the project, the requestor can push the change for review into the test branch.
2. The change will then be code reviewed by operations.
3. If the change passes code review, operations can trigger a jenkins run by adding a specific comment to the review.
4. Jenkins can launch an infrastructure clone, then run smoke tests against it.
4a. If the test passes, it can add a +1 verified review to the change.
4b. If the test failes, it can add a -1 verified review to the change, along with any puppet failures, and the results of the tests.
5. On successful test runs, operations can then cherry-pick the change to production.

In this environment, we wanted to make it relatively easy to join projects. We want it to be totally community maintained. To do so, we allow project members to add any Labs user to a project they are a member of. Role members can add any project member to a role they are in.

We also want to make it relatively easy to create nova projects. Currently only the operations team can create projects. This won't scale. One possible solution for this would be to allow anyone that is a project owner to create projects for others.

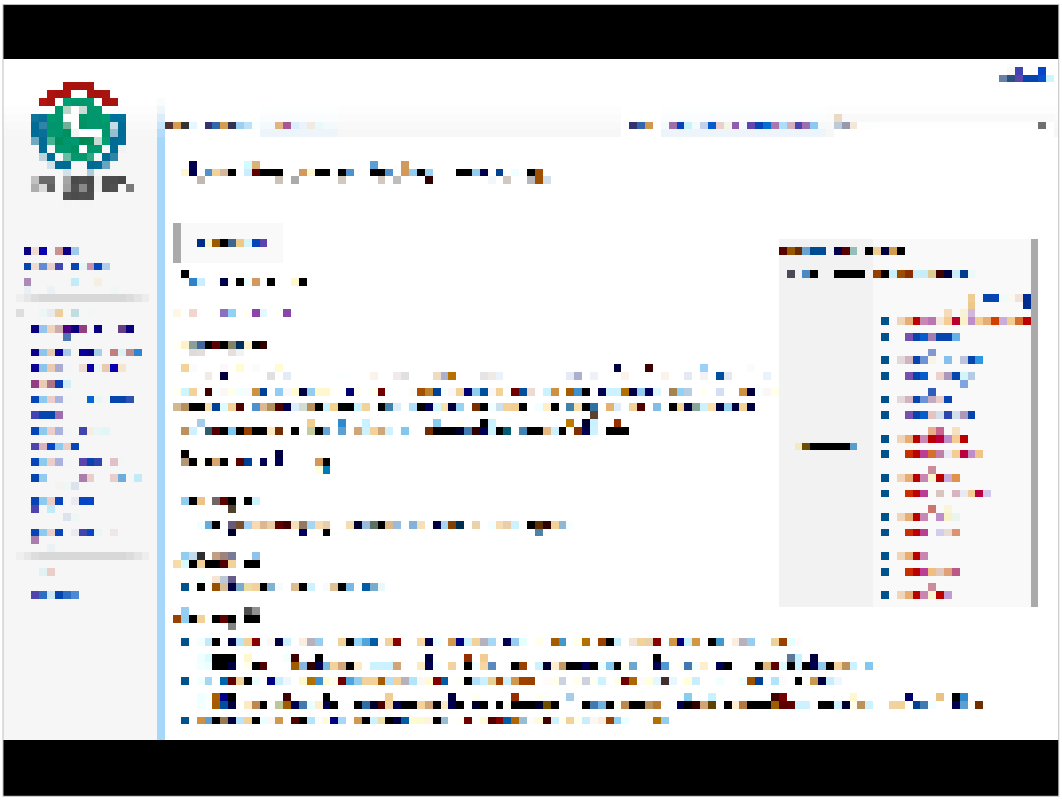
Since one of the primary goals of Labs is to facilitate software development, we'd like to make it relatively easy to create gerrit repositories. Currently, like projects, only the operations team can create gerrit repos. We don't really have any worked out solutions for this. If you have ideas on how this could work, please talk to us!

The operations team works in a very specif c way. We've created procedures that make it eff cient for us to work with a globally distributed team. For operations to scale when treated like an open source project, we must make it easy for developers and operations volunteers to follow the same procedures.

For this, everyone must document hosts, services, and changes, and all changes must be coordinated and communicated.

But people hate documenting things, and we can neither force, nor expect volunteers to do all of the necessary documentation.

To make this easier, we should have the system automatically document everything that can be documented in an automated way.

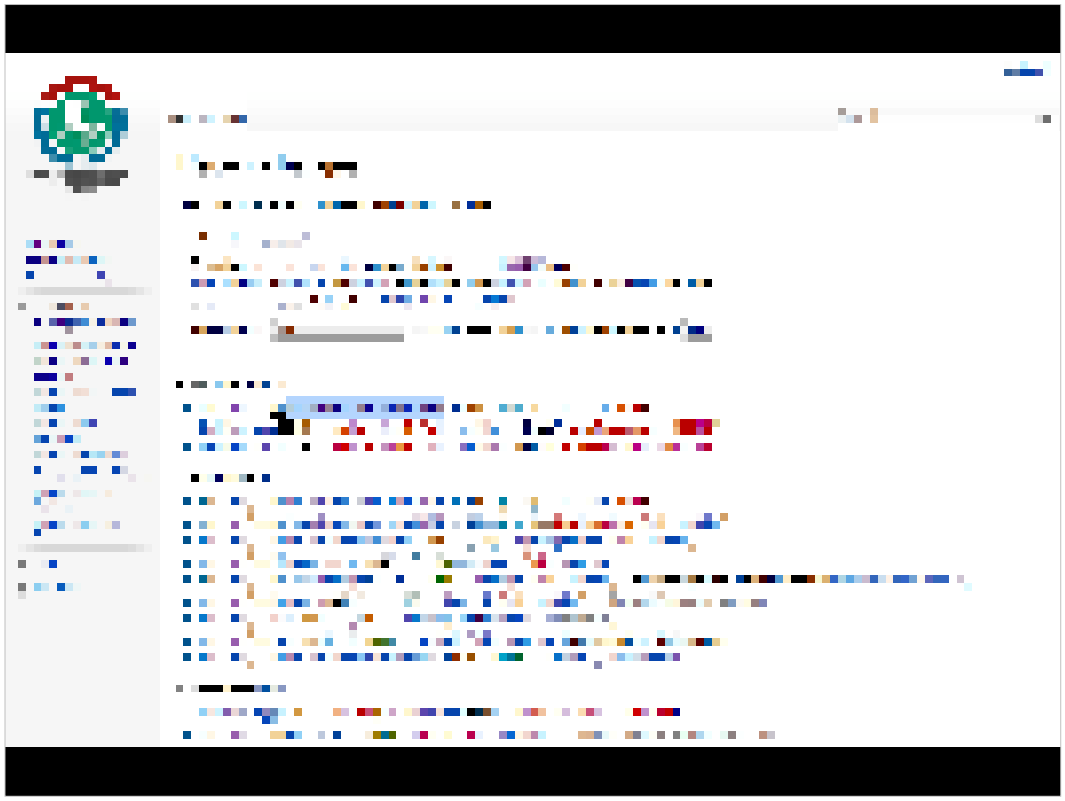


So, to facilitate this, every project has a documentation page. The project page is created when a project is created, and automatically lists the project members, has a manual documentation section, a Server Admin Log, and a list of instances inside of that project.
The documentation page is free-form documentation, and a description of the project. The description is shown in project lists.
The Server Admin Log is automatically generated when admins log actions via IRC, and will be discussed later.
Project pages are automatically added to the Projects category.
Some project properties are semantically annotated, I'll describe what this means soon.

Instance documentation is also automatically created upon creation of instances, and it is updated when instances are modified.

Additionally, all information documented is also semantically marked-up. Each piece of data is a semantic triple: (page name, key, value). For instance: (Nova Resource:I-00000130, Instance Type, m1.small). This data can be used in queries, which can be output in a number of ways, like table format, lists, and JSON.

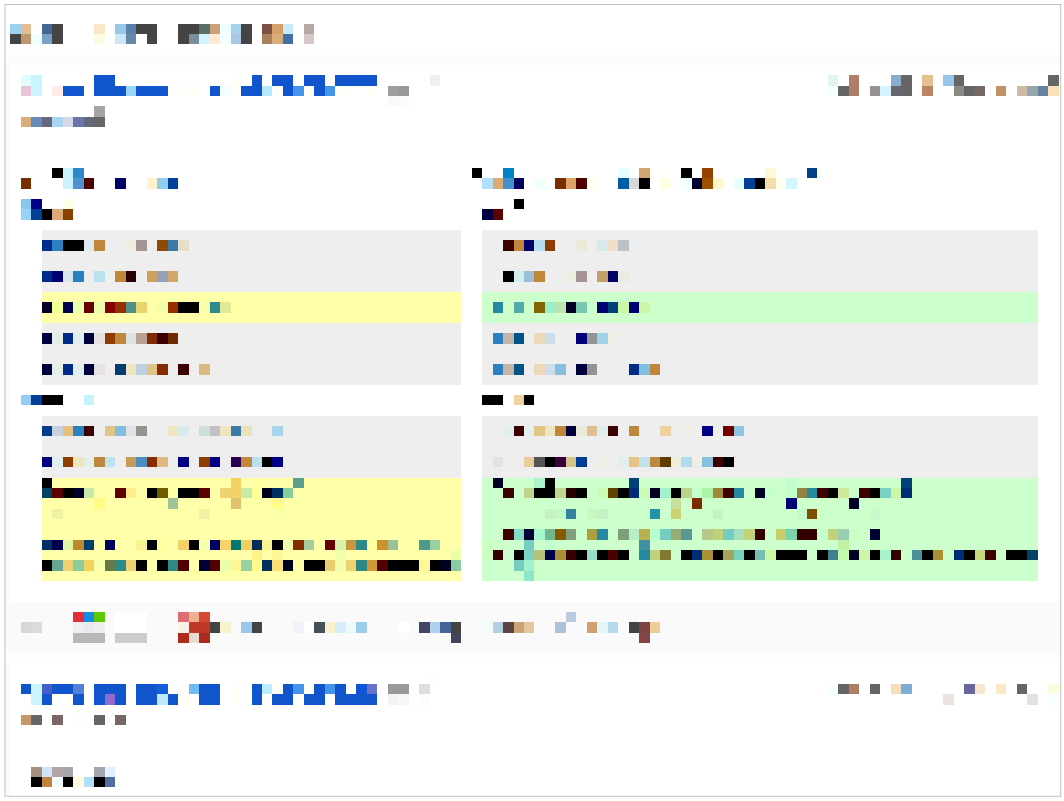
Instance resources are automatically added to the Instances Category.



Since this system is using MediaWiki, we've decided to extend some MediaWiki concepts. If you've ever used MediaWiki, or have ever edited Wikipedia, you may be familiar with the Recent Changes special page.

When you edit a page, change something, then hit save, that edit shows up in Recent Changes. It shows who did an action, when it was done, what the action was, and lets you see a diff of the edit.

The OpenStackManager extension is also creating, editing and deleting pages. When it does so, it is also updating the Recent Changes. So, it's possible to see actions done by users in real time.



The Recent Changes also has a feed, so it's possible to view changes in an reader. I check what's been happening in Labs by adding this feed to my Google Reader.

As mentioned before on the project pages there is a Server Admin Log. This log is created automatically when users log actions via IRC. In our #wikimedia-labs channel, if someone uses:

!log bots Added cluebot to bots1

It'll automatically show up in the project's Server Admin Log.

We take the project server admin logs, and transclude all of them into a combined Server Admin Log, sorted by last modified project. This let's people see what's going on in all projects.

The operations team is globally distributed. We use IRC as our primary means of communication. Our volunteers are also globally distributed. We expect everyone to use IRC as their primary means of communication, so that everyone can participate.

Of course, IRC is more real-time communication, and people can miss things. For major changes, especially changes that affect multiple projects, of all projects, they should be discussed via the mailing list, labs-l@lists.wikimedia.org.

So, we have this neat model, but is it working?

We launched in late October. We're currently in closed beta. As of today, we have 47 projects, with 81 instances, and 105 users. Our current limiting factor is infrastructure available.

Volunteer operations community is vibrant. This is a small list of volunteer accomplishments thus far.

I had originally planned to clone our production infrastructure myself, since it's fairly involved, and not everything we do is puppetized. Unfortunately, I was overwhelmed trying to keep up with the growth of Labs, and was having a hard time getting to it. Our community came to the rescue. They've set up a clone with nearly all of our production services.

Also, there was a volunteer who wanted to make a bot infrastructure in Labs, since they were having difficulties with Toolserver. This wasn't on the roadmap till much later in the year when tool labs was set to launch. Like the clone of the production cluster, a volunteer took this over. User petan created a bot infrastructure with shared storage, shared mysql databases, multiple bot running instances, and an apache frontend to view bot actions.

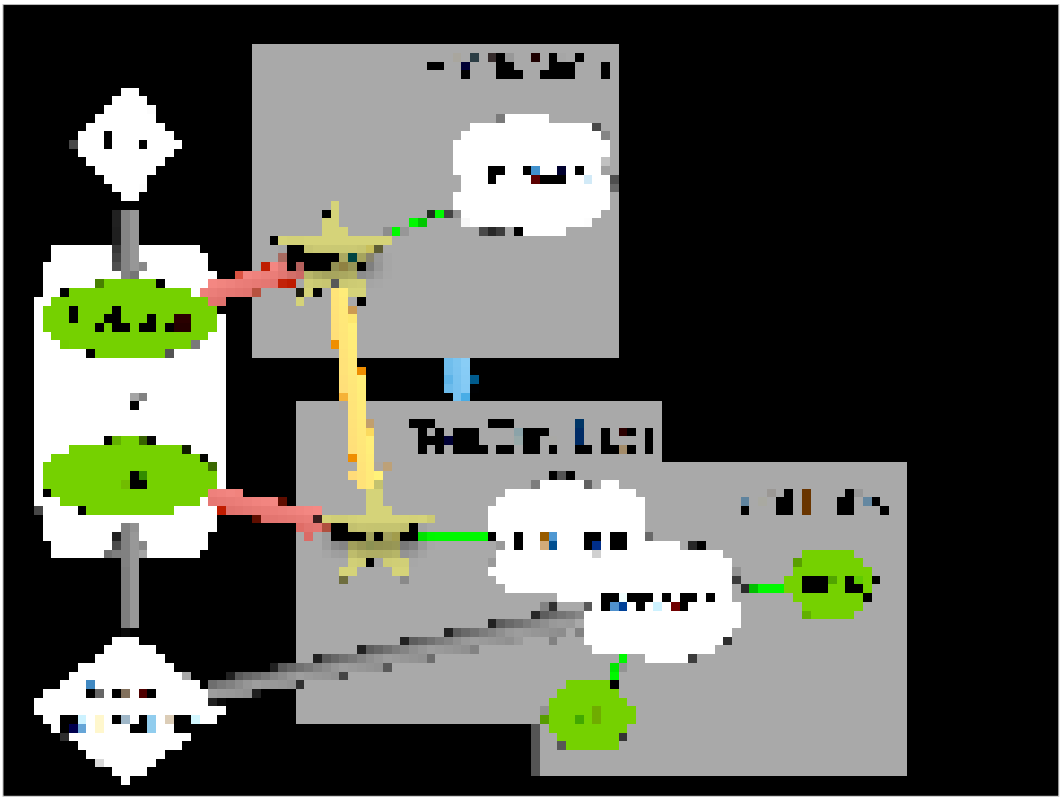
Petan also wanted to be able to monitor the bots, but nagios was one of the services we had in production that wasn't puppetized, so he worked around this with a really interesting approach. As mentioned earlier, instance data is semantically annotated, and can be used in queries. One of the query outputs is JSON. He wrote a script that pulls the instance information from MediaWiki via a Semantic MediaWiki query. He then gets the puppet variables and classes defined for the instances, and created monitors based on the classes and variables. His implementation is actually faster than the implementation we have in production.

There's a number of other volunteer contributions that have occurred in such a small period of time. See the combined Server Admin Log for more examples!

- At our most recent hackathon in SF most developers set up or joined a labs project, developed a feature, and demoed it. Most projects at the hackathon had multiple people working on a feature, and were able to work on them collaboratively. Only one ops person was needed to facilitate this, and it was only for account creation, troubleshooting bugs in Labs and education on how Labs works.
- We even had a feature from a volunteer developer go through development, testing, code review and deployment to production in those two days. We have an nginx logging module that is custom written to send UDP log messages in our custom format to some collecting daemons. It was broke in a couple ways. Abe Music decided to take on one of the module's issues. He created an instance, installed our nginx source package, made his changes, tested them on the instance in labs, then pushed his changes. We reviewed them, did some more testing, then pushed the change out to production. It was a pretty major accomplishment to have a production change go live in such a short period of time from a volunteer!

As mentioned earlier, some pieces of the infrastructure we implemented by our community. One piece was something we didn't think we'd be able to implement this early on. What this showed us that that our initial concept was again wrong. So, we've evolved our concept to match with a community focused approach.

http://www.flickr.com/photos/wagnerr/3059248309/



Since we are implementing tool labs features right now, we've basically just merged them together into one project. There is still the need to treat tool labs as an intro developer ramp, but those pieces of the infrastructure can be handled over time, and can also be handled by the community.

Of course, nothing is without problems, and our approach has a ton of them.

First off this approach hasn't totally eliminated our operations bottleneck, but has shifted it to other things. People are still bottlenecked by waiting on resources to become available in Labs. Operations now has to do code review. Also, a lot of operations' time is now eaten up by educating people on how to use Labs.

Thankfully, most of the bottlenecks will lessen over time. After the initial closed beta, resources will be dramatically increased. Code review is a known problem in the software development community, and there are approaches to handling it that we can leverage. Education is less of a problem over time as well, as documentation improves and the community takes over some of these responsibilities.

Though there is a bottleneck shift, the bottleneck is much wider than it was previously!

I know most people will be shocked to hear this, but giving root to random people on the internet can be dangerous. If you give root to everyone who offers to help, you'll likely run into a person that offered to help simply to abuse the resources.

This is of course a problem in public clouds, but it is even more of a problem in private clouds, and especially in ours.

Since the security separations in our projects is purposely loose, it makes our model very susceptible to social engineering attacks. An attacker can work his way into a single project, then work his way through other projects by requesting access.

So, it's important to audit systems much more thoroughly than normal. A lot of the normal red flags still apply, thankfully:

- Check network traffic
  - Especially check traffic initiating from the Labs cluster to the outside world (most traffic should either be going to other instances, other projects (to a lesser extent), and to our production cluster.
  - High bandwidth use is also something worth investigating.
  - Some protocols should simply be blocked. There is no legitimate reason for anyone to be using bittorrent, for instance. If someone has reason to use it, they can ask for an exemption.
- Check storage usage
  - Large storage usage is something worth investigating, as well. Some projects will require large amounts of storage for video, images, dumps of Wikipedia, etc. But, if someone is using a lot of storage, we should know why.
  - We should also check for file and mime-types that are odd. We have a very specific set of file and mime-types acceptable to upload to our projects. If we run into some outside of that, we should investigate why there are there.
- Do manual inspections occasionally
  - If a user gained access to do a specific project, make sure the user is actually doing that.

Thankfully, our community is generally getting access to do legitimate work, and they have a strong mission drive. We can trust our community to self-police and to report problems that aren't caught by audits. Also, we can trust some members of our community to handle a lot of the auditing tasks as well.

Another worry is that the Labs infrastructure could be used to escalate privileges into our production environment if someone with production access does something stupid, or is lazy.

There's some ways around this. One is to separate the bastion intances. One bastion instance only allows people with production access, and the other only allows people without production access. Escalations from the bastions wouldn't then be possible.

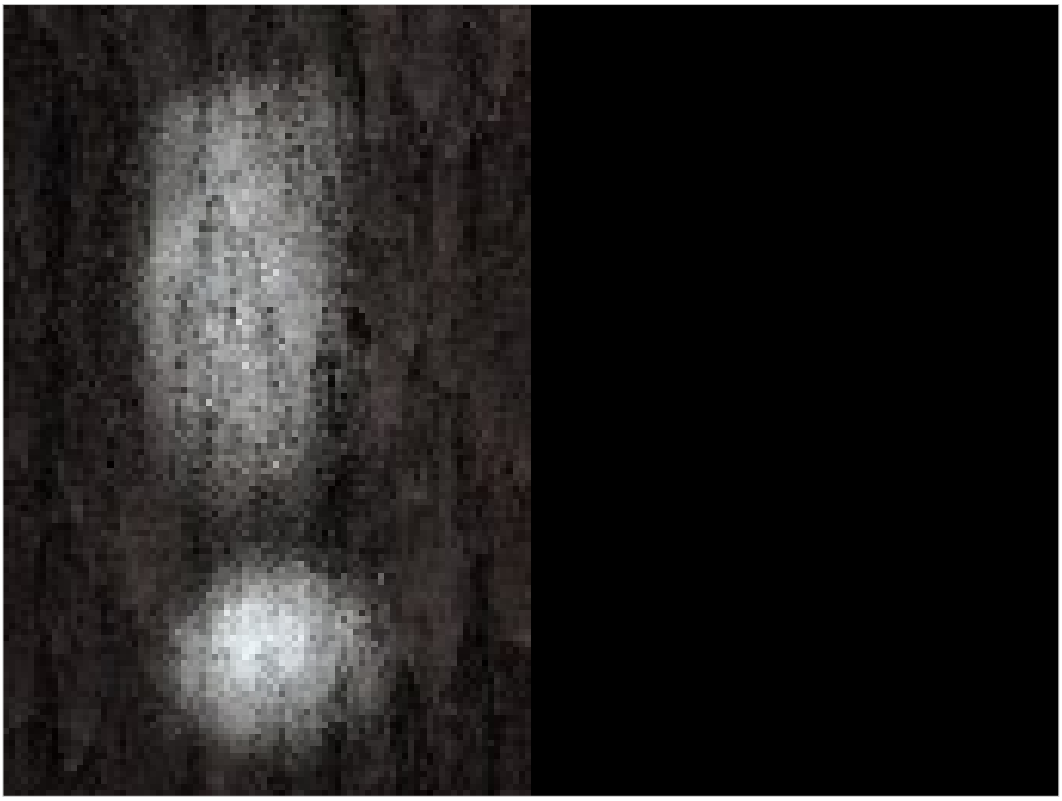
Of course, there's still the worry of people forwarding their agents past bastion instances. So, using another authentication method inside of the Labs environment is likely a good idea. Rather than using ssh keys, the use of kerberos would allow people to access instances past the bastion instance. With kerberos enabled, all instances could be conf gured to not allow agent forwarding.

There isn't much purpose in having an open source project if it can't be reused. Thankfully, this model, and the work produced by it can be used elsewhere directly.

There is obvious applications of this model for other non-profits with a technically oriented volunteer community.

Open Source software projects can also benefit from this.

This model is also applicable to private organizations, though. Like other open source software, it's possible to reuse portions of this, and upstream their changes. It isn't necessary for them to completely open source their configuration (and it's likely they never will, so there's little worry of license pollution). Specifically, this model would allow private organizations to let developers bypass operations for normal tasks, but still have them operate in a way that the operations team deems necessary (code reviewers can deny changes to production that don't take the right approach).



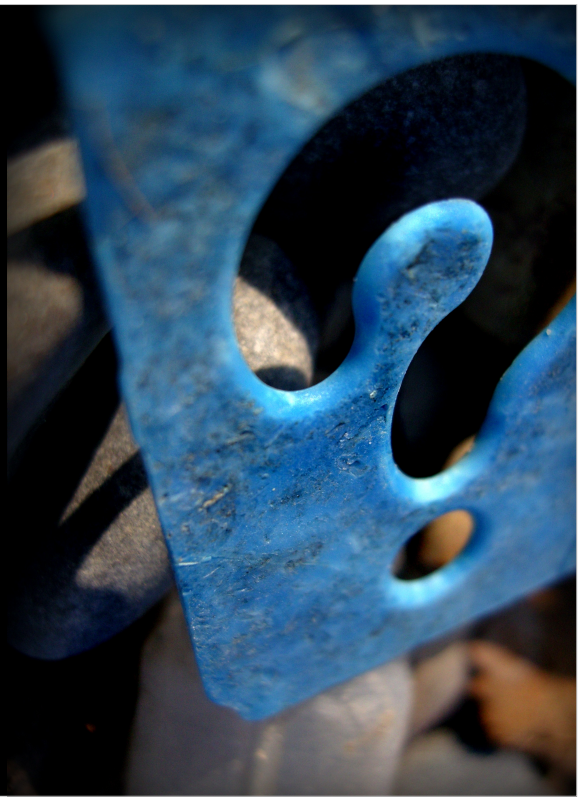
Since this is an open source project, you can get involved!

* Help us improve our infrastructure. Join a current project, or suggest and implement another.
* We are building and testing the cloud environment inside of the cloud environment. Help us improve OpenStack, the OpenStackManager MediaWiki extension, our storage architecture, our MySQL infrastructure, our queuing infrastructure, our authn/z infrastructure.
* We *really* need help building OpenStack modeled services for a number of things
* Help us improve our documentation
* Help us maintain our community
* Help us move our puppet code into modules
* Come work with the foundation, we have a lot of open positions right now, but I'd really love to have one of you work with me as an Operations Engineer for Labs.

Ryan Lane
Wikimedia Foundation
ryan@wikimedia.org

IRC: Ryan_Lane on Freenode; channels:

- #mediawiki
- #wikimedia-labs
- #wikimedia-tech
- #openstack
- #openstack-dev



Please don't hesitate to contact me. I'm very active on IRC, and would love to talk to you.