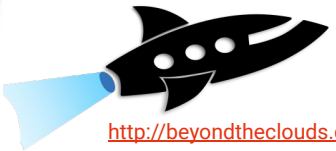
OpenStackoïd: Collaborative OpenStack Clouds On-Demand

— Beyond the Clouds, The Discovery Initiative



http://beyondtheclouds.github.io

Who Are We?

Adrien Lebre
Prof. IMT Atlantique

STACK Team leader
http://stack.inria.fr

Inria Discovery Chair
http://beyondtheclouds.github.io

Javier Rojas Balderrama
Inria research engineer

OpenStackoïd developer
https://github.com/BeyondTheClouds/openstackoid/

EnosLib developer
http://enoslib.readthedocs.io

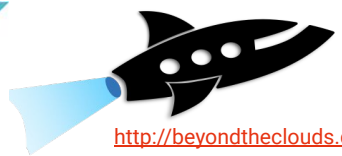
Ronan-Alexandre Cherrueau
Inria research engineer

OpenStackoïd developer
https://github.com/BeyondTheClouds/openstackoid/

EnOS developer
http://enos.readthedocs.io

Inria

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom



http://beyondtheclouds.github.io

Managing Resources of an Edge Infrastructure?

— What does it mean?

Edge Infra?

*A kind of **Distributed Cloud Infrastructure***

Properties

- 100s/1000s of locations (*i.e.*, data centers)
- Dozen of servers per data centers
- WAN links (10 to 300 ms RTT)

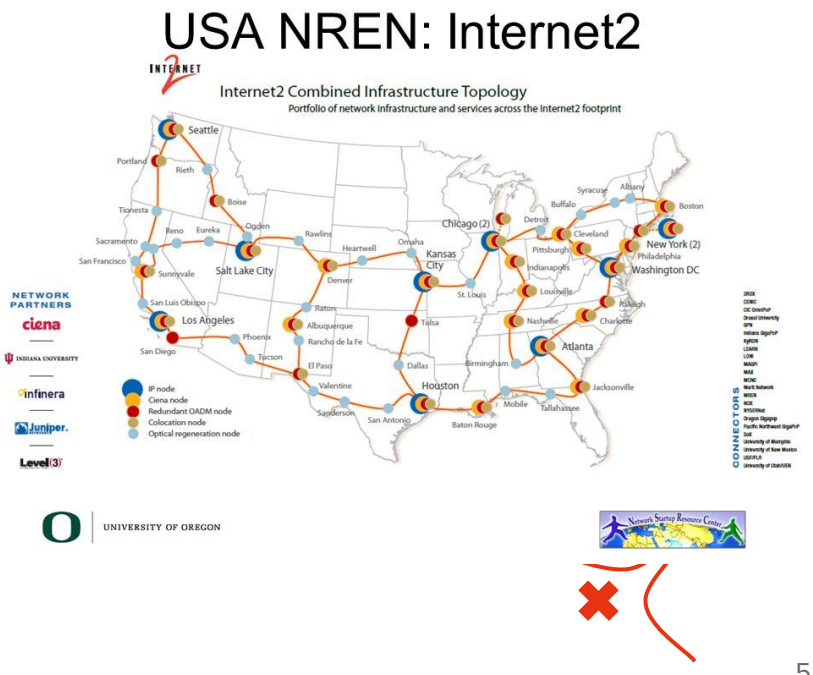
- Intermittent connectivity
- Network partitioning issues

Example of an Edge Infrastructure

- A National Research and Education Network
  - Internet2/Renater/...

- Red point is a Point of Presence (**PoP**)
- A **PoP** contains a micro **Data Center**
  - Dozen of servers
- **WAN links** interconnect **PoPs**
  - `10ms`, Paris ↔ Marseille
  - `150ms`, Berlin ↔ Denver

- Losing connection may lead to **network partitions** (e.g. Marseille Corte in France)

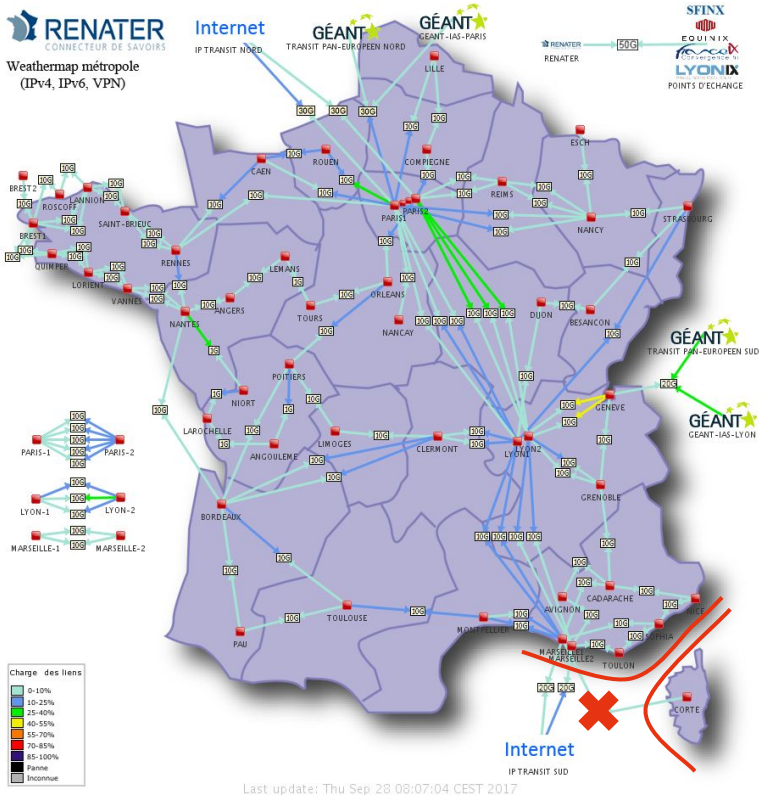


Example of an Edge Infrastructure

- A National Research and Education Network
  - Internet2/Renater/...

- Red point is a Point of Presence (**PoP**)
- A **PoP** contains a micro **Data Center**
  - Dozen of servers
- **WAN links** interconnect **PoPs**
  - `10ms`, Paris ↔ Marseille
  - `150ms`, Berlin ↔ Denver

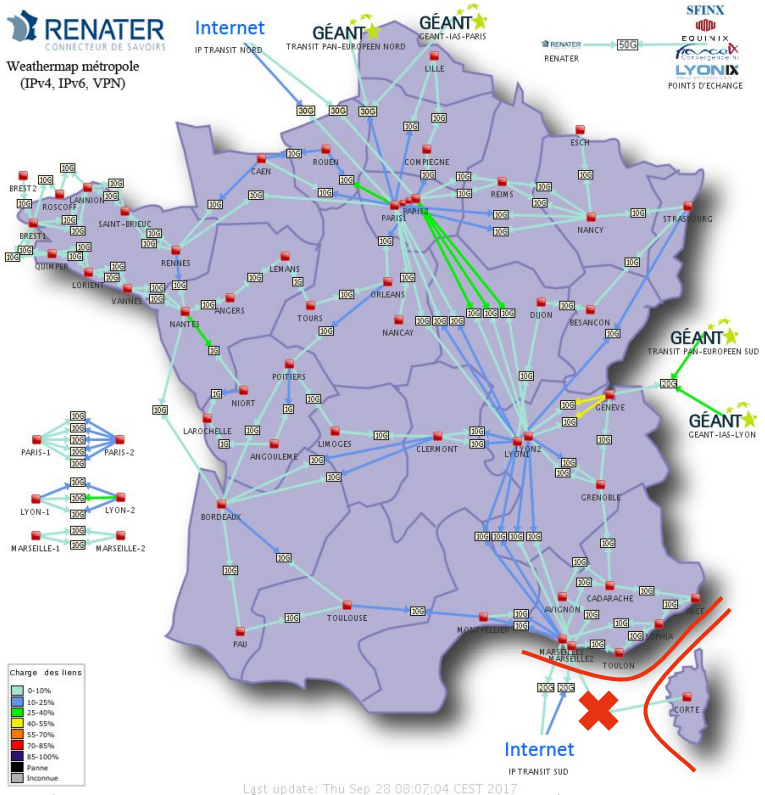
- Losing connection may lead to **network partitions** (e.g. Marseille Corte in France)



Managing Resources of an Edge Infra?

***Same** as in Cloud Computing. **Tuned** for the Edge‡.*

1. Operate/use a **single DC**
   - Manage users, flavors, quotas
   - Provision compute, storage, net
2. Operate/use **several DCs**
   - **Cross-DC** collaborative provisioning (intra/inter services)
   - Manage **multiple** DC **simultaneously**
3. Robustness w.r.t. network delay & **disconnections**
   - Access/Manage reachable resources (full isolation)



‡. https://www.openstack.org/videos/summits/vancouver-2018/can-we-operate-and-use-an-edge-computing-infrastructure-with-openstack

Managing Resources of an Edge Infra with OpenStack

— Review past and ongoing actions.

Red Thread: Boot of a VM

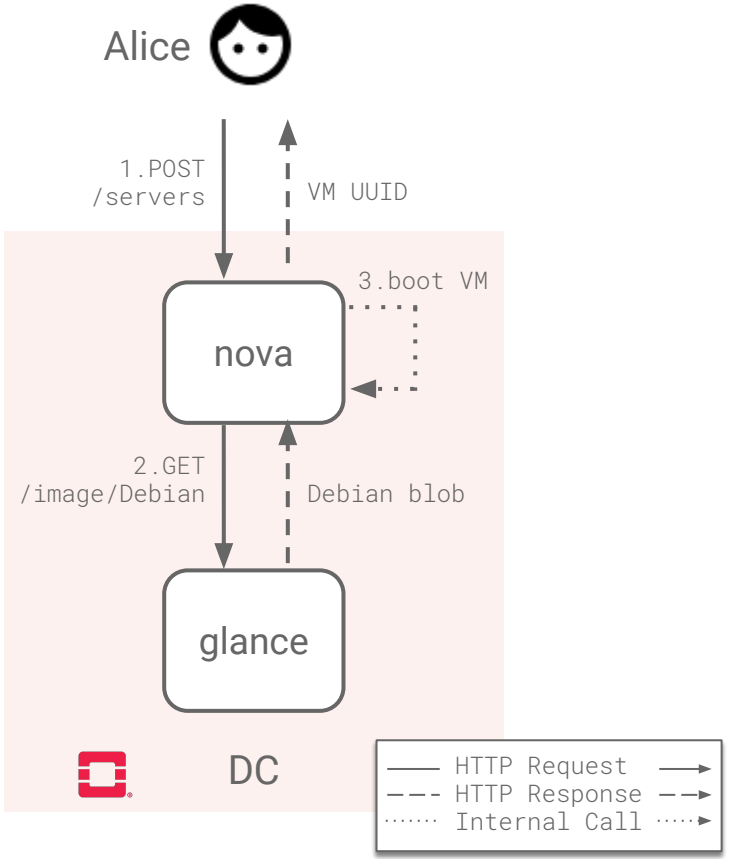
Boot of a **Debian VM** (simplified)

1. Operator requests a `boot` to nova
2. Nova contacts glance to get Debian
3. Nova boots VM internally

Boot VM **scenarios**

- in a single DC (**1-DC**)
- in one DC with an image from another DC (**x-DC**)
- in multiple DC (***-DC**)
- **Globally** vs. **partially** connected infa.

Boot VM	1-DC	x-DC	*-DC
global	??	??	??
partial	??	??	??

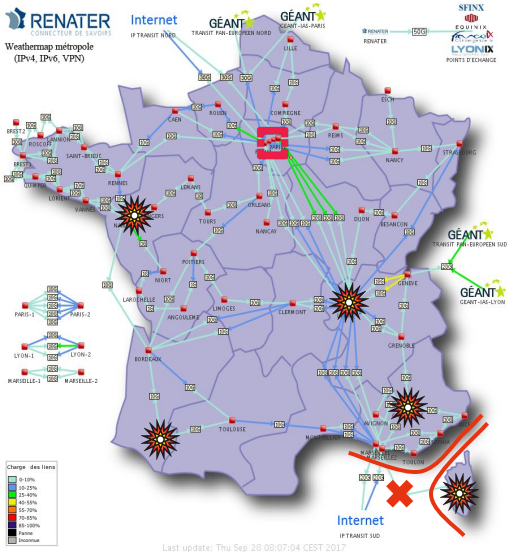


Approach 1: Centralized Management

One DC** hosts the **control plane**; **Other DCs** host **compute nodes

- **Theoretically**, manipulating remote compute resources does not change the OpenStack behavior
- **Practically**, a lot of issues/challenges‡ †
  - Impact of latency, throughput, intermittent connectivity, etc.
  - What are the deployment rules for each service (*e.g.*, Cinder)?
  - Deployment/Upgrade of the system

- Several studies (scalability, communication buses, etc.) show that it is a viable approach but they are still challenges to tackle
  - StarlingX TC as well as other actors (RedHat, Orange, etc.) are investigating those issues



Boot VM	1-DC	x-DC	*-DC
global	✓	-	-
partial	✗	✗	✗

→ Operational, but focuses on specific use-cases

‡. https://www.openstack.org/videos/summits/boston-2017/toward-fog-edge-and-nfv-deployments-evaluating-openstack-wanwide
†. https://www.openstack.org/videos/summits/berlin-2018/rabbitmq-or-qpid-dispatch-router-pushing-openstack-to-the-edge

Approach 2: Distributed Management

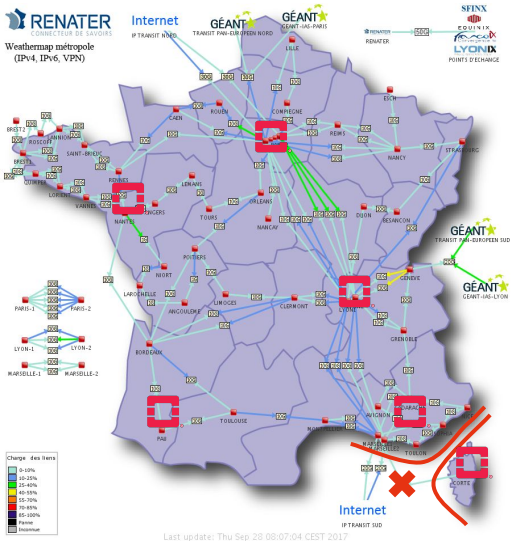
***Every DC** is one OpenStack that **collaborates** with others (à la peer-to-peer)*

Theoretically, should fulfill needs for managing Edge resources infra.

- One control plane per DC
  - A DC is **independent** of others (partition resiliency)
    - Boot VM in Paris, Marseille, or Corte, till we can connect to
- DCs are **collaborative** with each other
  - Share resources with others (benefits from natural sharding)
    - Boot VM in Paris with Debian image from Marseille
  - Replicate resources at some locations (preserve from delay/partition)
    - Replicate Debian image in Paris, Marseille and Corte

Practically, a sophisticated solution

- Implementing collaboration is a **conundrum**
- OpenStack doesn't provide a general solution

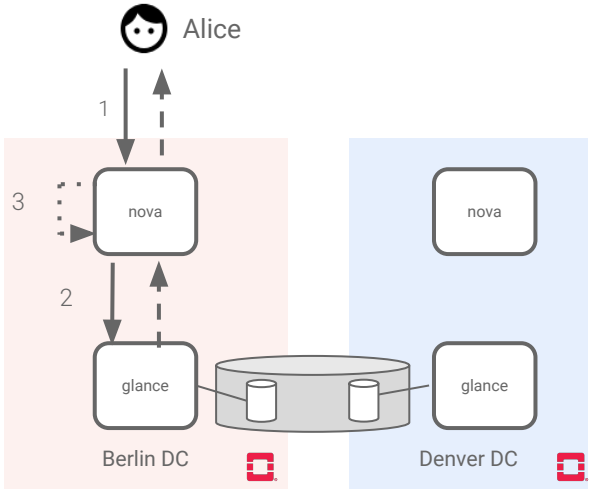


Boot VM	1-DC	x-DC	*-DC
global	✓	✓	✓
partial	✓	✓	✓

Theoretical

DataBase Collaboration

*Every DC is an OpenStack; Implement **collaboration** by making **resources global** via the **DB** (active-active Galera, CockroachDB, …)‡ †*



Pro

- Do not need to modify OpenStack code

Issues

- Maintain **consistency** of **all data across all DCs**
  - Forbids any writes in case of network partition (partial: ✗)
- DataBase **only considers data**
  - A **resource** is made of **data and effects**
  - Collaboration via DB **misses effects** (x-DC/*-DC: Keystone ✓, Neutron ✗, ...)

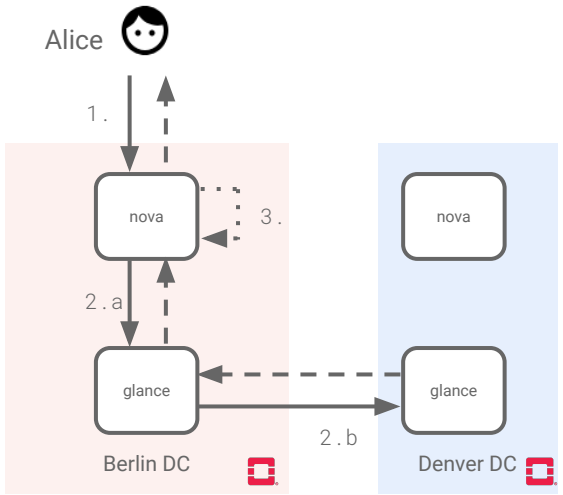
Boot VM	1-DC	x-DC	*-DC
global	✓	✓, ✗	✓, ✗
partial	✗	✗	✗

→ Resources could not be global (CAP theorem)
→ Resource has to come with its side effects

‡. https://www.openstack.org/videos/summits/vancouver-2018/keystone-in-the-context-of-fogedge-massively-distributed-clouds
†. https://wiki.openstack.org/wiki/Fog_Edge_Massively_Distributed_Clouds

Service-to-Service Collaboration

*Make the **service natively collaborative** (K2K‡, Glance to Glance†)*



Pro

- Know the features of the service (deal with side effects)
- Efficient/Optimal implementation (optimistically scale at edge)

Issues

- **Tangle sophisticated collaboration** code with **vanilla** code
  - Force core developers to maintain collaboration code, make new features collaborative
  - Intrusive collaboration is not an option for some services (not everyone want to do edge/need collaboration)

→ Collaboration code should be decoupled from vanilla code

Boot VM	1-DC	x-DC	*-DC
global	✓	✓?	✓?
partial	✓?	✓?	✓?

‡. https://docs.openstack.org/security-guide/identity/federated-keystone.html
†. https://wiki.openstack.org/wiki/Image_handling_in_edge_environment

Broker Collaboration

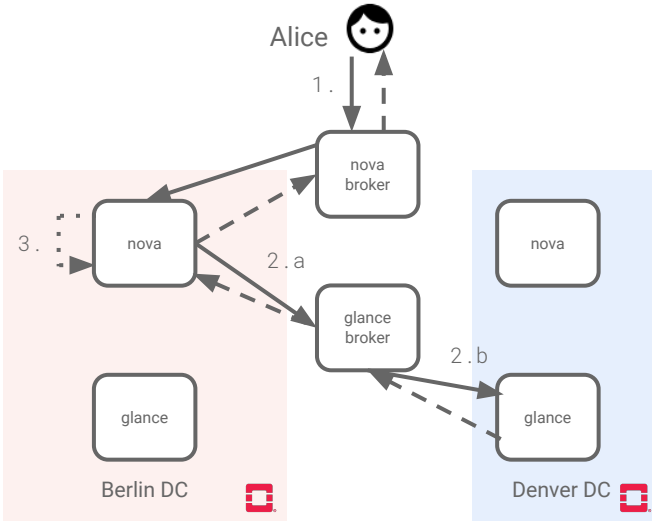
***Broker** on top **orchestrates** the **collaboration** (Tricircle‡, Mixmatch†, ...)*

Pro

- Put **collaboration code outside** of vanilla code (in the broker)
- Enable **enhancement of APIs** for sharing/replication
- Deal with side effects (inter-service for free, intra in the broker)

Issues

- Current implementations
  - Rely on a central broken (partial: ✗)
  - Miss mechanism for replication (*-DC: ✗)
- **Broker** has to be **exhaustive** with the underlying **APIs**
  - **Lot of code** to simply **expose APIs** at **broker level**



→ Broker should not reimplement API to the risk of developing a new OpenStack on top of OpenStack

Boot VM	1-DC	x-DC	*-DC
global	✓	✓	✗
partial	✗	✗	✗

‡. https://wiki.openstack.org/wiki/Tricircle
†. https://mixmatch.readthedocs.io/en/latest/

Distributed Management with OpenStackoïd

— The OpenStack-to-OpenStack vision

OpenStackoïd

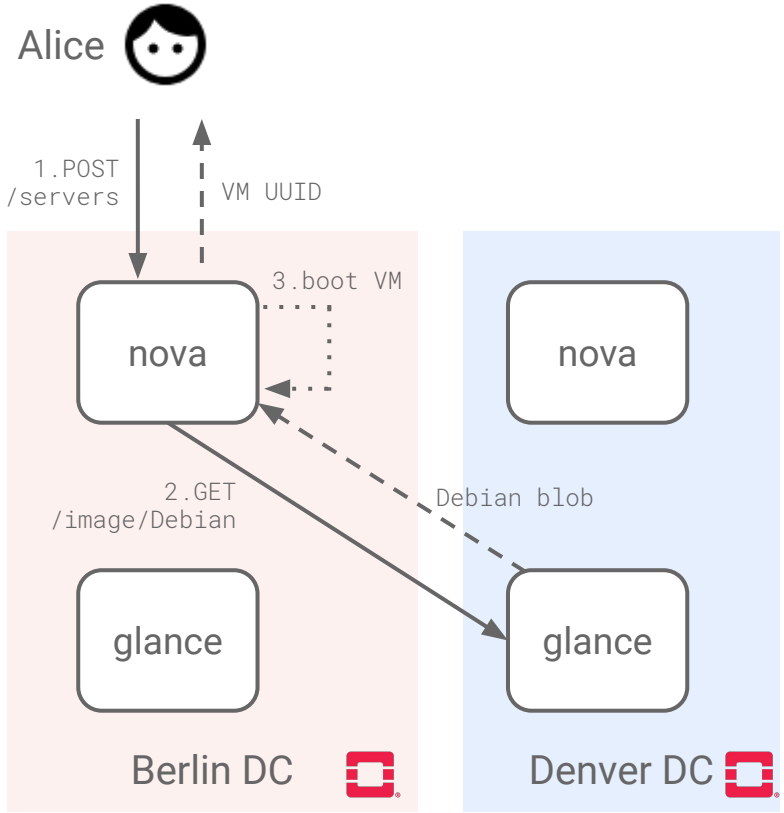
*A broker based solution **without** a broker*

Alice defines the scope of the request into the CLI. The scope specifies **where the request applies**

```
openstack server create my-vm
     --image Debian
     --os-scope { nova:   Berlin
                , glance: Denver }
```

Generalization to all APIs

- Don't have to be exhaustive with the underlying API
- Don't require a specific code for an API



Boot VM in Berlin with Debian from Denver

OpenStackoïd

Scope for **1-DC** operations

```
OS@Berlin$ openstack server create my-vm --image Debian
    --os-scope {nova: Berlin, glance: Berlin}
```

Scope for **x-DC** operations

```
OS@Berlin$ openstack server create my-vm --image Debian
    --os-scope {nova: Berlin, glance: Denver}
```

Scope for ***-DC** operations

```
OS@Berlin$ openstack server create my-vm --image Debian
    --os-scope {nova: Berlin&Denver, glance: Denver}
```

*-DC: and ‘&’

Do the operation **here and there**

- Create a user in Berlin and Denver

```
openstack user create Alice
  --password-prompt
  --os-scope {keystone: Berlin&Denver}
```

- List VMs in Berlin and Denver

```
openstack server list
  --os-scope {nova: Berlin&Denver}
```

Properties

- **On-demand** partial replication
  - Replication at scope locations
  - Keep consistency 🏆

- Query **multiple** DCs at once
- Don’t **change** computation **type** 🏆
  - List VM **&** List VM **=** List VM
  - List a **&** List a **=** List a
  - a **&** a **=** ???

*-DC: or ‘|’

Do the operation **here or there**

- Boot a VM in Berlin with image from Denver or Paris

```
openstack server create my-vm
   --image Debian
   --os-scope { nova: Berlin
              , glance: Denver|Paris }
```

Properties

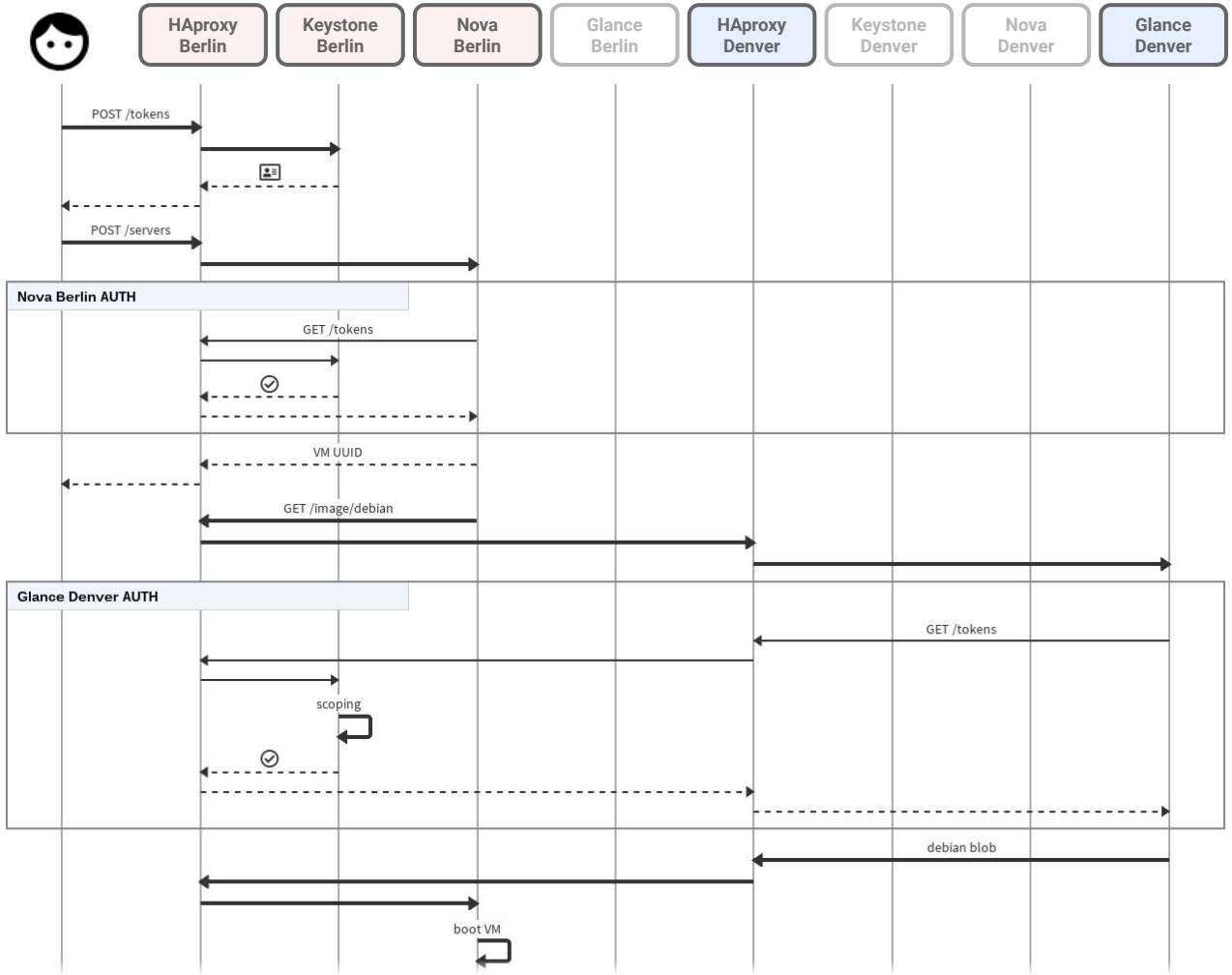
- Let the operator **implements retries workflow**

  No matter if one is down or don’t have the image, till the other is up and has the image.

OpenStackoïd Proof-of-Concept

— http://github.com/BeyondTheClouds/openstackoid

```
openstack server create …
  scope:{keystone: Berlin,
         nova: Berlin, glance: Denver}
```



Features

Scope **stick** to operation

- **Session** from start to end of the operation execution
- Enable **concurrent operations** with different scopes

Scope at **service level** (nova, glance, keystone, …)

- High level understanding of OpenStack is enough to define collaborations
- **Inter-service** collaboration only

Risk of **bad collaborations** (x-DC)

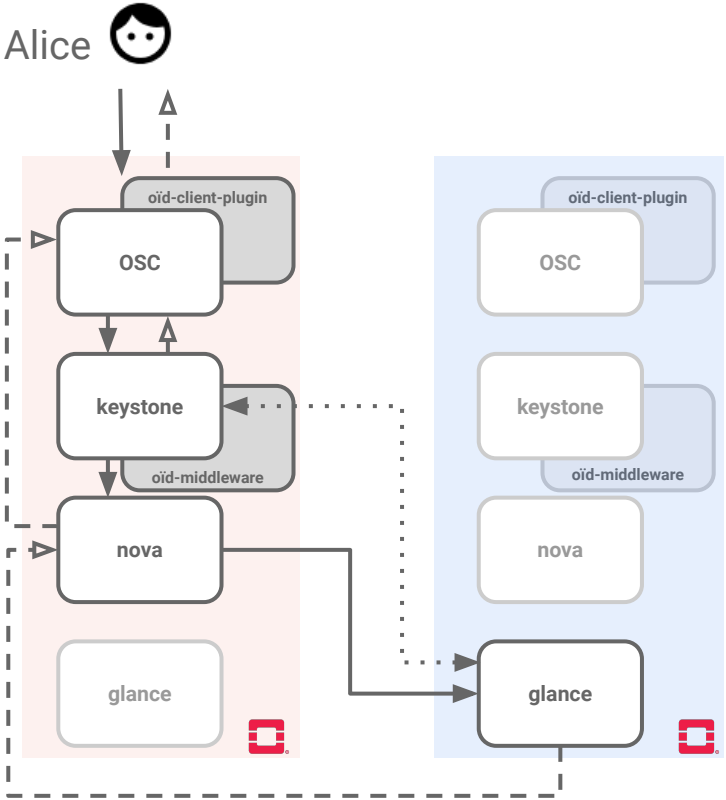
- **Resources unreachability**: boot VM in Berlin with **local** network in Denver; it is not yet possible to extend network resources across DCs (API limitations, technical issues)
- **Local state**: verify in keystone of Berlin the glance service token from Denver

OpenStackoïd MVP

An approach that:

- enables the implementation of operators &, | (not possible with HAproxy)
- does not modify the code source of current OpenStack services
- should deal with potential issues:
  - High granularity of some services
  - Distant side effects
  - Exclusion of bad collaborations

Boot VM	1-DC	x-DC	*-DC
global	✓	✓	✓
partial	✓	✓	✓



Wrap Up

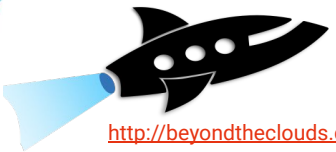
Takeaway

- Collaboration between Edge should be done on demand (and only if needed)
  - Thousands of independent sites
  - Collaboration between network ASes
- Implement on demand collaboration ideas with other systems
  - OpenStack**oïd** for OpenStack,
  - ***oïd** for K8S,
  - new edge application services, etc.
- One problem among many others (zero touch provisioning, etc.)

http://github.com/BeyondTheClouds/openstackoid

Manage	Needs	scope
Single DC	Manage resources **locally** • boot VM in **Berlin** • List VMs in **Denver**	1-DC‡ • {nova:**Berlin**, glance:**Berlin**} • {nova:**Denver**}
Multiple DCs	**Cross-DC** collaboration • boot VM in **Berlin** with Debian from **Denver**	x-DC • {nova:**Berlin**,glance:**Denver**}
	Manage resources **simultaneously** • create image in **Berlin** and **Denver** • boot VM in **Berlin** with Debian from **Denver** or **Paris**	*-DC • {glance:**Berlin&Denver**} • {nova:**Berlin**, glance:**Denver**|**Paris**}

‡ Unnecessary; scope is implicitly local



http://beyondtheclouds.github.io