The INFN Corporate Cloud

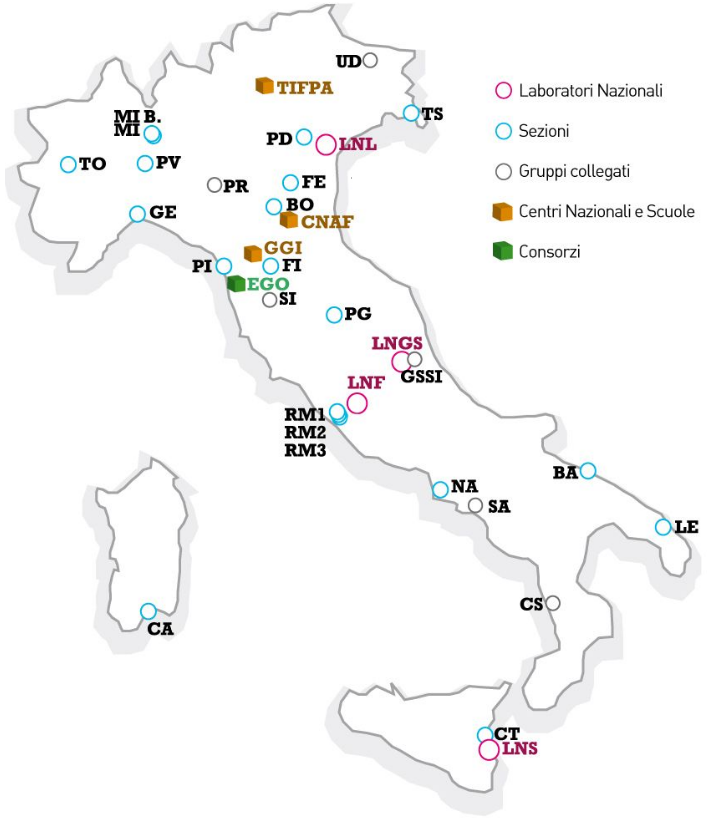
Stefano Stalio - INFN
stefano.stalio@lngs.infn.it

OpenStack Day in Rome - Sep 21, 2018

INFN - a distributed institute

“The National Institute for Nuclear Physics (INFN) is the Italian research agency dedicated to the study of the fundamental constituents of matter and the laws that govern them, under the supervision of the Ministry of Education, Universities and Research (MIUR). “

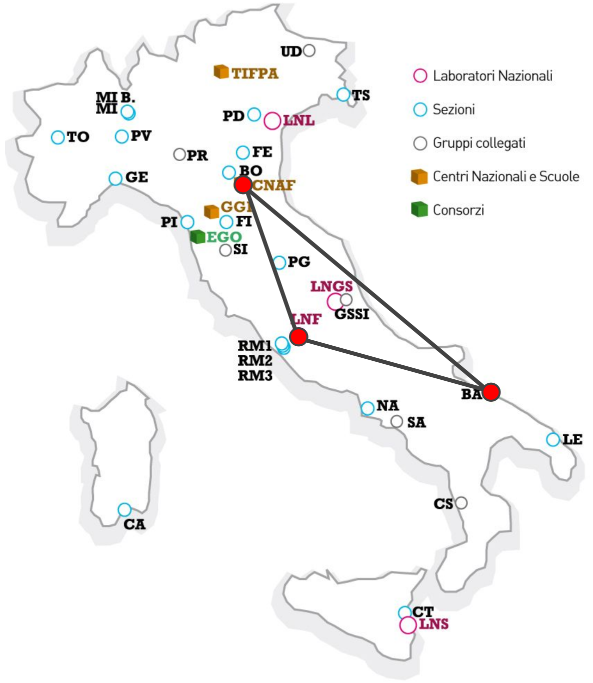
INFN is a distributed institute with more than 20 sites and many data-centers (of very different sizes), almost one per site. Among them the LHC Tier-1 at CNAF.



INFN-CC - a distributed cloud

INFN Corporate Cloud (INFN-CC) is INFN’s geographically distributed private Cloud infrastructure.

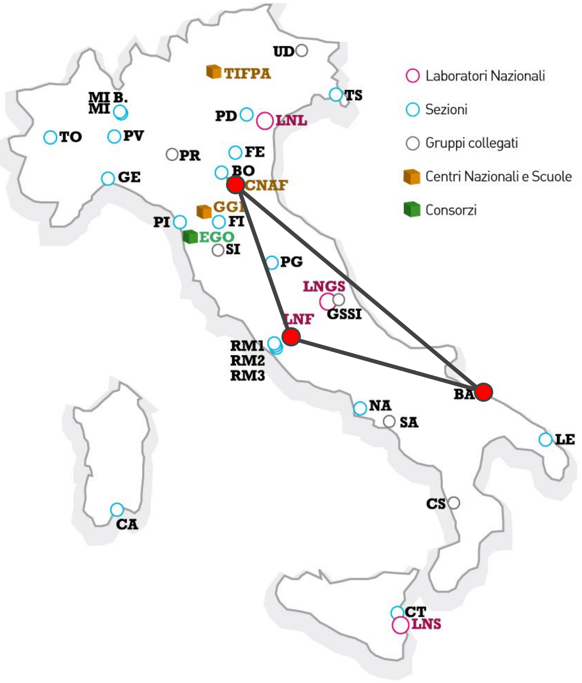
It provides services starting from the IaaS level and it is based on OpenStack.



INFN-CC - a distributed cloud

INFN-CC also deploys a **higher level PaaS layer**, developed within the **EU funded project INDIGO-DataCloud** (www.indigo-datacloud.eu)

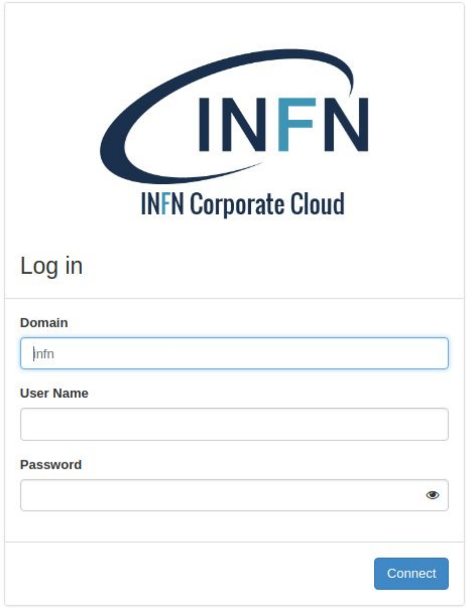
- easier access solution to computing and storage resources for the INFN scientific community
- automatic instantiation and configuration of services or applications used, like batch-system on demand or big data analytics facilities



INFN-CC - a distributed cloud

- **INFN-CC** not the only INFN Cloud
- many other cloud installations @INFN
- based on OpenStack and OpenNebula
- network services, scientific computing

INFN-CC designed and born as a geo-distributed entity



Domain: infn

architecture

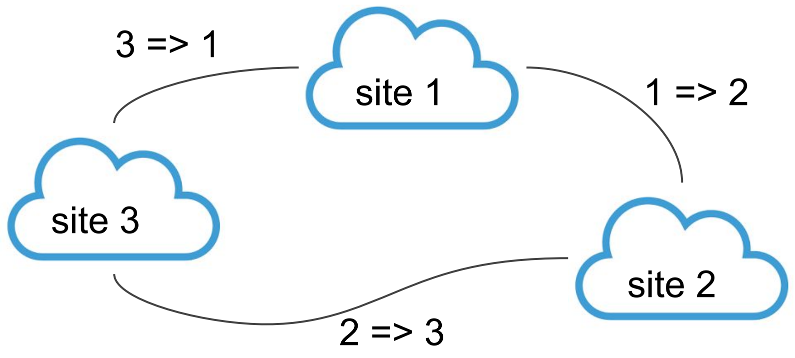
INFN-CC is deployed in three INFN sites: **Bari**, **CNAF** (Bologna), **LNF** (Roma), from the OpenStack point of view INFN-CC is a **multi-region** cloud

- **global services**
  - some services have a global scope: **Identity**, **Object Storage**, **Image**
  - implemented on all sites for high availability, backed by common DBs when needed
- **local services**
  - other have a local scope: **Compute**, **Network**, **Volume**
  - implemented independently on each site

local services

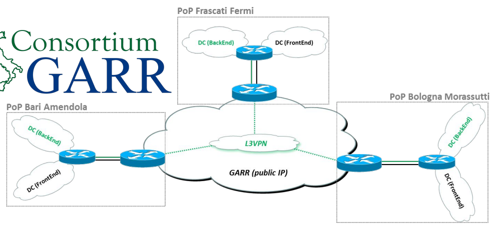
As said before, the **Compute**, **Network** and **Volume** services have a local scope, like in any other OpenStack deployment, but

- **compute** and **volume** rely on a **CEPH** back-end
- **CEPH rbd mirror** is employed to replicate data across INFN-CC sites for **disaster recovery**
- one **CEPH** instance per site



what’s behind global services

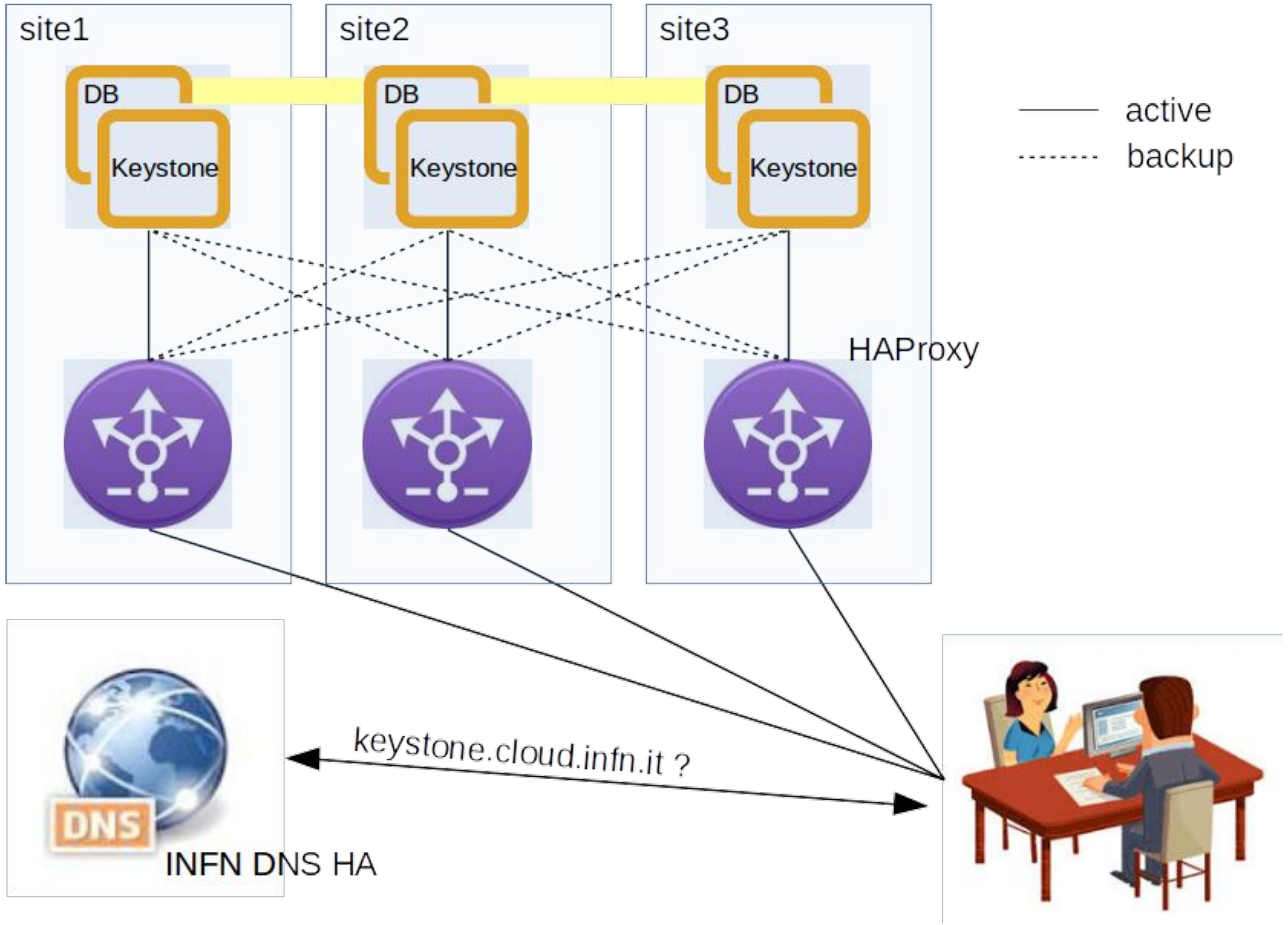
- A **level 3 distributed private network**, provided by **GARR**, allows for easy cloud management and fast data exchange
- A **distributed Percona XtraDB Cluster** relies on this network and is the back-end both for the <u>identity service</u> and the <u>image service</u> catalog
- A **distributed DNS** can be dynamically modified, by humans as well as monitoring processes, in order to make clients point only to working endpoints



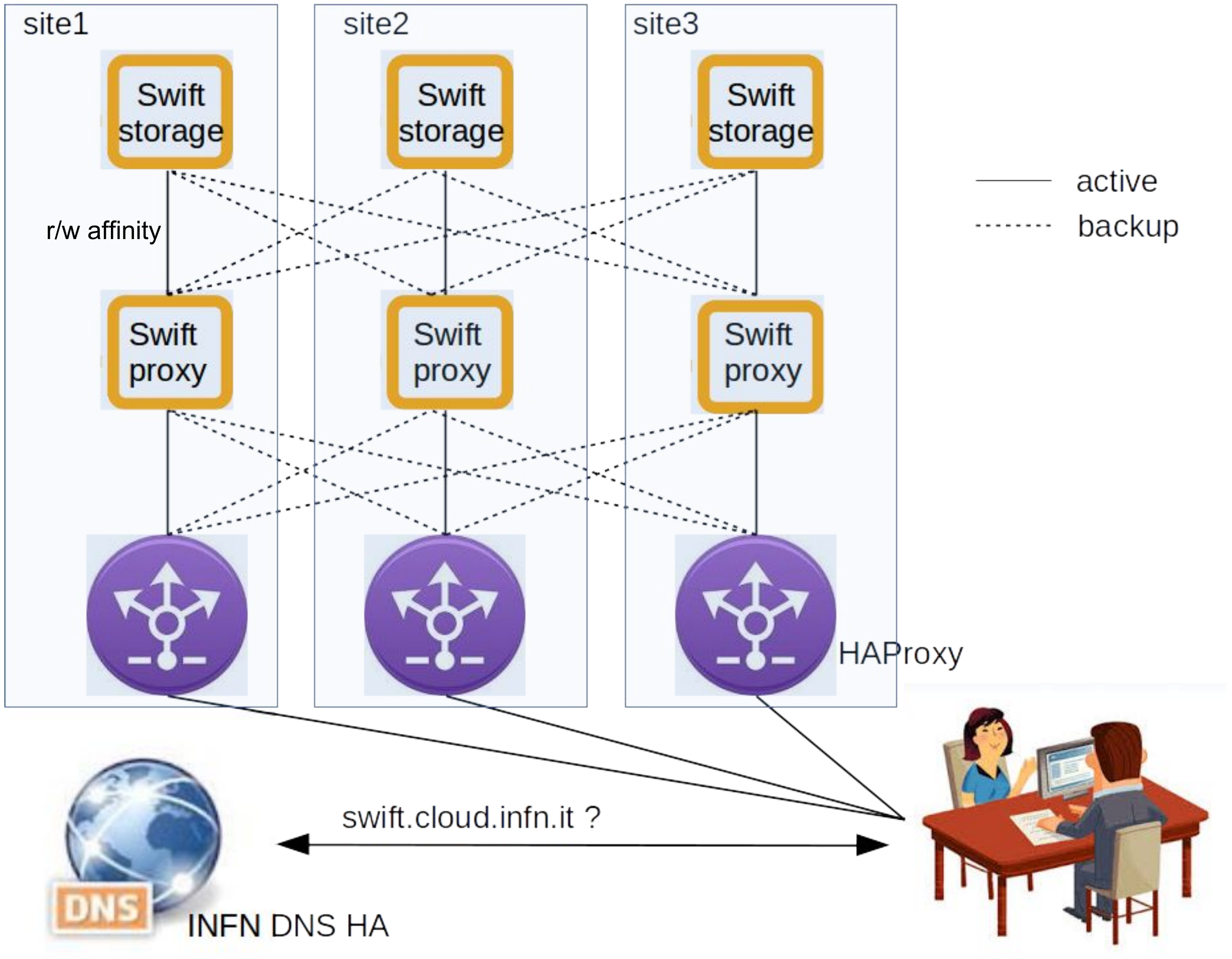
global services

- **OpenStack Keystone** access points, pointing to the above mentioned distributed DBMS, are available on all INFN-CC sites
- **OpenStack Swift** relies on the INFN-CC private network and is deployed geographically
- **Openstack Glance** relies on Swift as a storage backend and on the Percona cluster as a catalog, this way it is fully distributed as well.

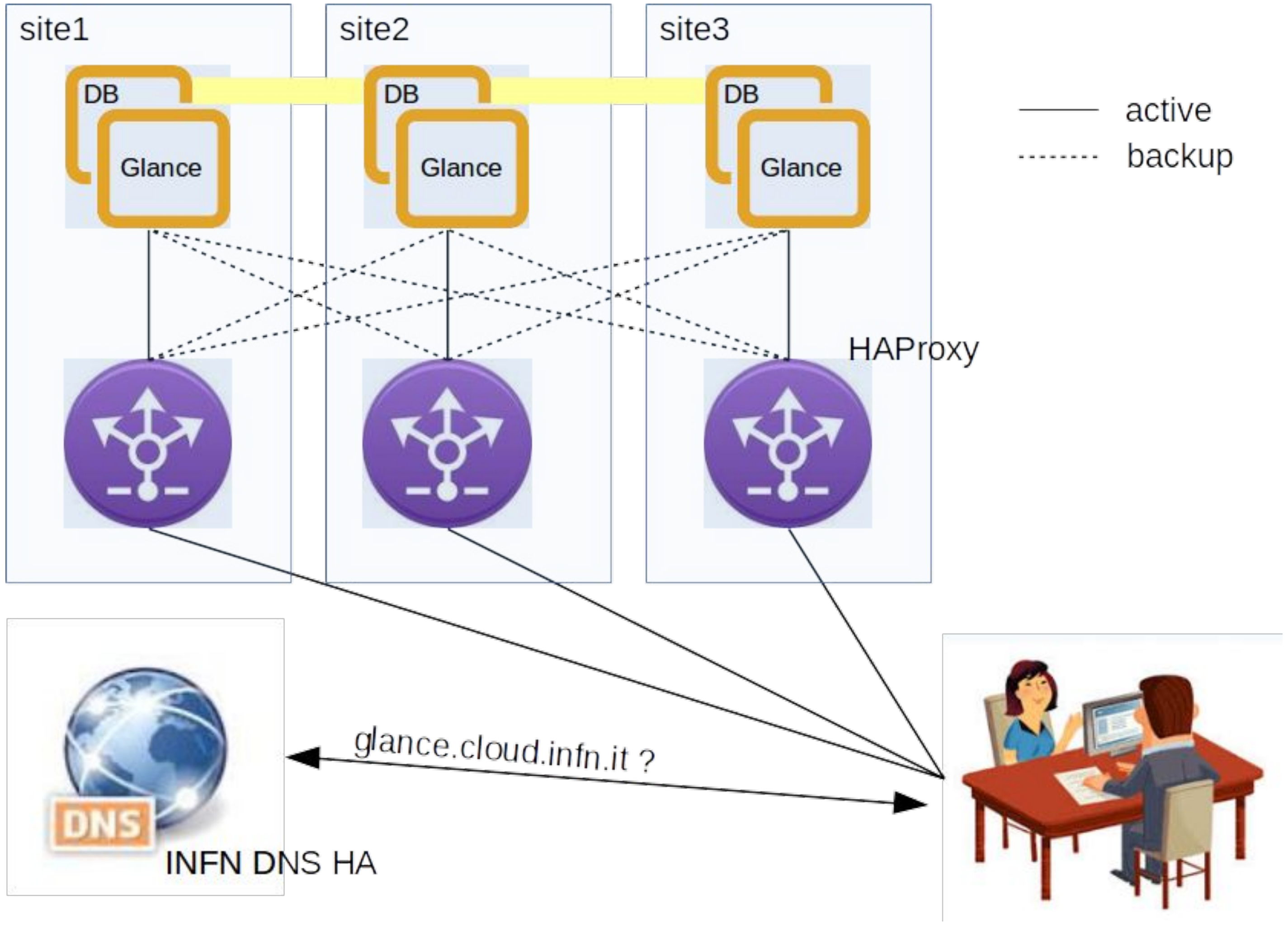
Keystone


site1
site2
site3
DB
Keystone
DB
Keystone
DB
Keystone
active
backup
HAProxy
keystone.cloud.infn.it ?
DNS
INFN DNS HA

Swift


site1
site2
site3
Swift storage
Swift storage
Swift storage
r/w affinity
Swift proxy
Swift proxy
Swift proxy
HAProxy
active
backup
swift.cloud.infn.it ?
INFN DNS HA
DNS

Glance


site1
site2
site3
DB
Glance
DB
Glance
DB
Glance
active
backup
HAProxy
glance.cloud.infn.it ?
DNS
INFN DNS HA

DNS-HA

- The INFN-CC **distributed DNS-HA** is based on PowerDNS
- DNS records are kept in a distributed SQL DB (again!)
- DNS (and DB) servers are distributed in multiple INFN sites
- PowerDNS offers multiple user interfaces: ReST APIs, nsupdate, a web UI, a command line client (local), also SQL INSERT,UPDATE,DELETE
- we built a simple python client over the ReST APIs
- working on and testing a monitoring/controlling program that updates DNS entries according to actual server availability (inspired by Nagios, linux heartbeat, HAproxy,...)

DNS-LB

A home-made program for monitoring remote services and updating a DNS

- self-health tests
- customized service tests (can use ping, curl, nagios plugins,...)
- e-mail alerts
- logging
- different operating modes:
  - load balancer
  - failover
  - failback
- dockerized

work in progress

DNS-LB

A few ideas for the future:

- any INFN-CC user could pick the DNS-LB Docker up and run it for her/his own services, but fine-grained ACLs on users, ip addresses and DNS names must be implemented somehow
- a distributed cluster of DNS-LBs, where decisions are taken by a quorum might increase the reliability of the system

features/advantages of INFN-CC

- sites survive a disaster in another site
- sites share the same users and projects
- sites share the same data (object store)
  - **same image catalog** for all sites
  - VM snapshots can be used for **site to site migration/replica**
  - user/scientific data geo-replicated
- users can deploy **distributed services**
  - critical services survive a disaster in one site (possibly two)

building IT services on top of INFN-CC

- the same technologies, strategies and tools employed for distributing OpenStack Identity, Object Storage and Image can be adopted by the cloud users to build geo-redundant services on top of INFN-CC
- plus something is needed for syncing posix volumes

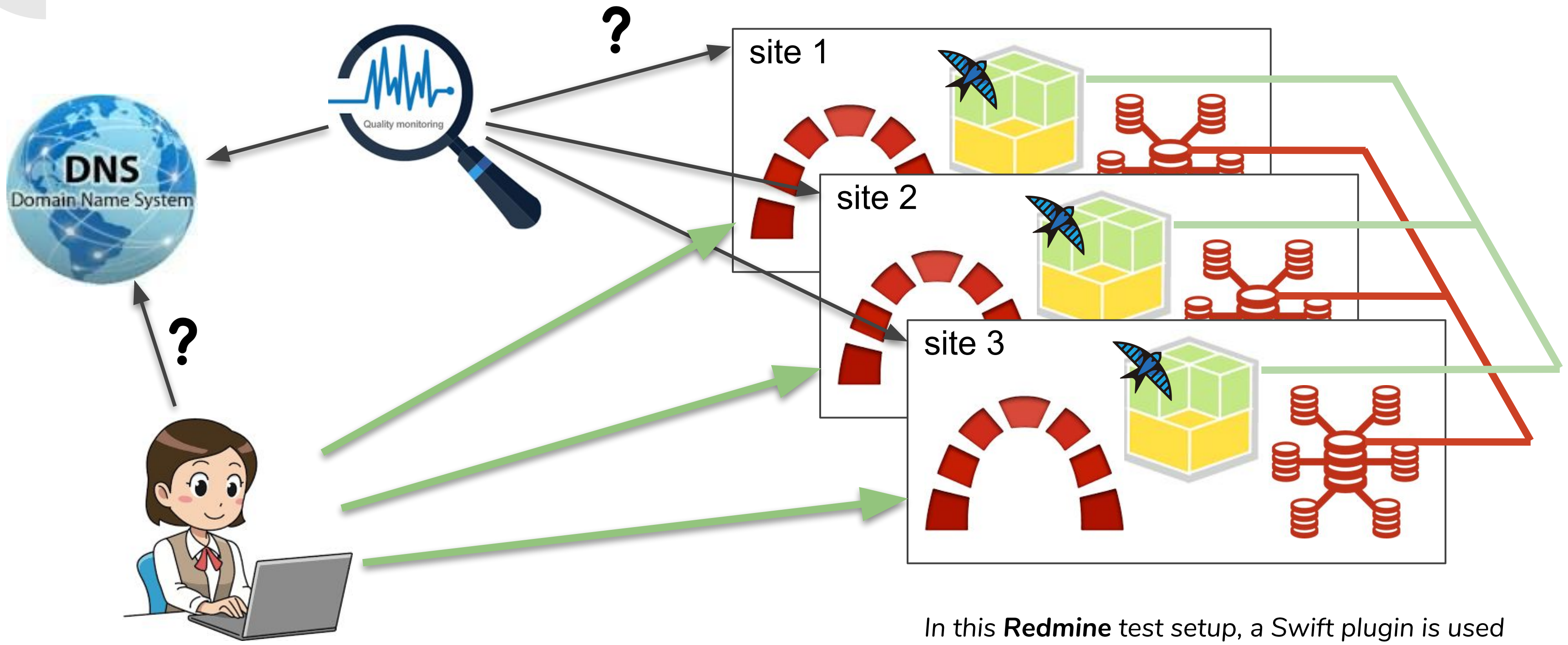
ingredients

- a distributed **SQL DBMS**
- a distributed **object storage**
- a “remotely synchronized” **posix storage**
- a **DNS load balancer** backed by a monitoring tool

syncing posix volumes???

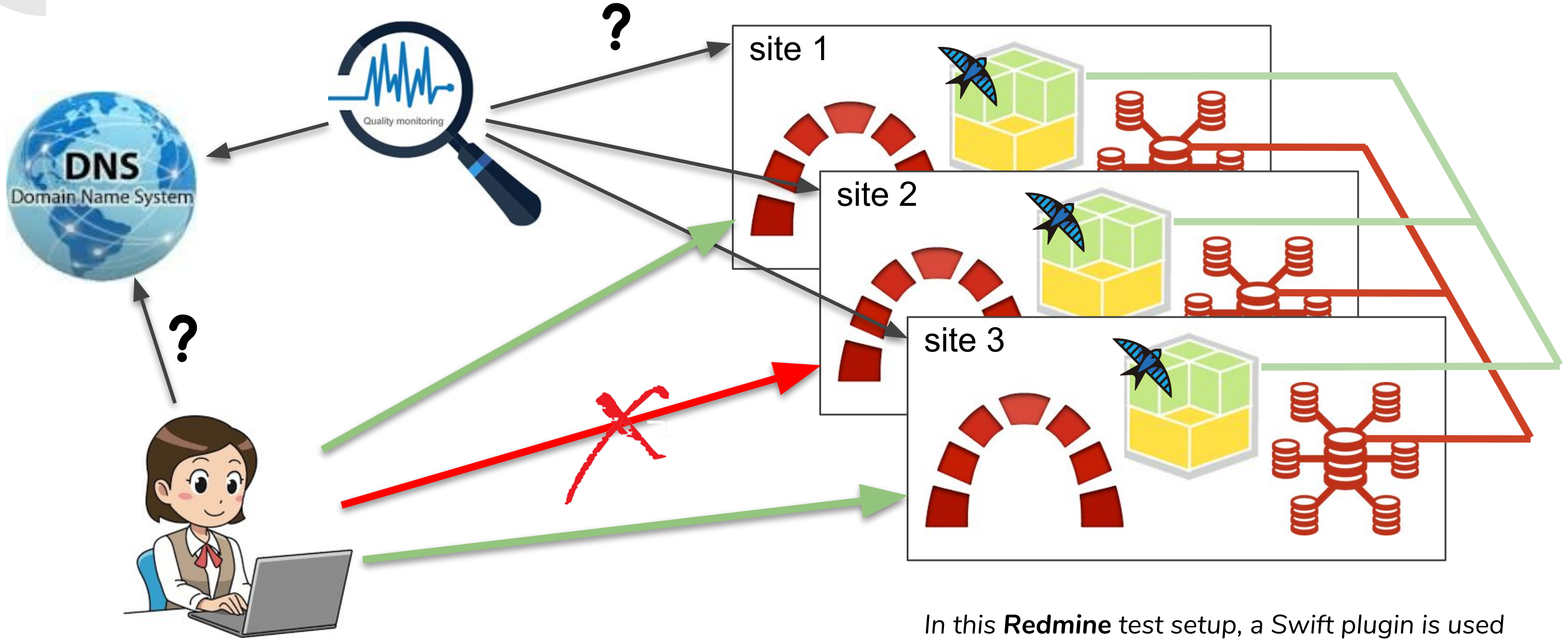
- This appears to be the **hardest challenge** over a >10ms latency network
- Not so difficult if it is one way, more if it has to be multi-master
- Some ideas:
  - lsyncd (under test)
  - DRBD
  - ......
- looking for: **resiliency**, **consistency**, **scalability**, **performance**

recipes: load balancer



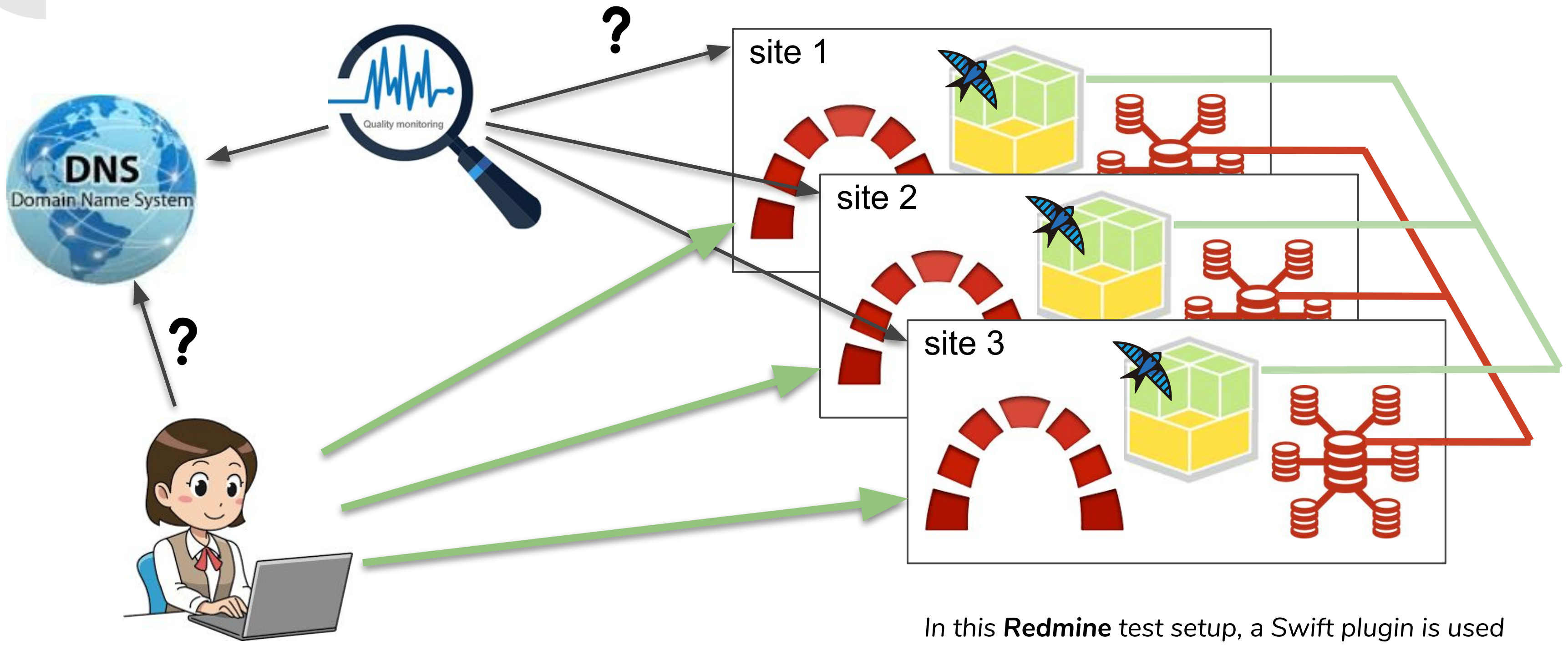
*In this **Redmine** test setup, a Swift plugin is used to provide the storage back-end while the DBMS is provided by a distributed SQL cluster .*

recipes: load balancer



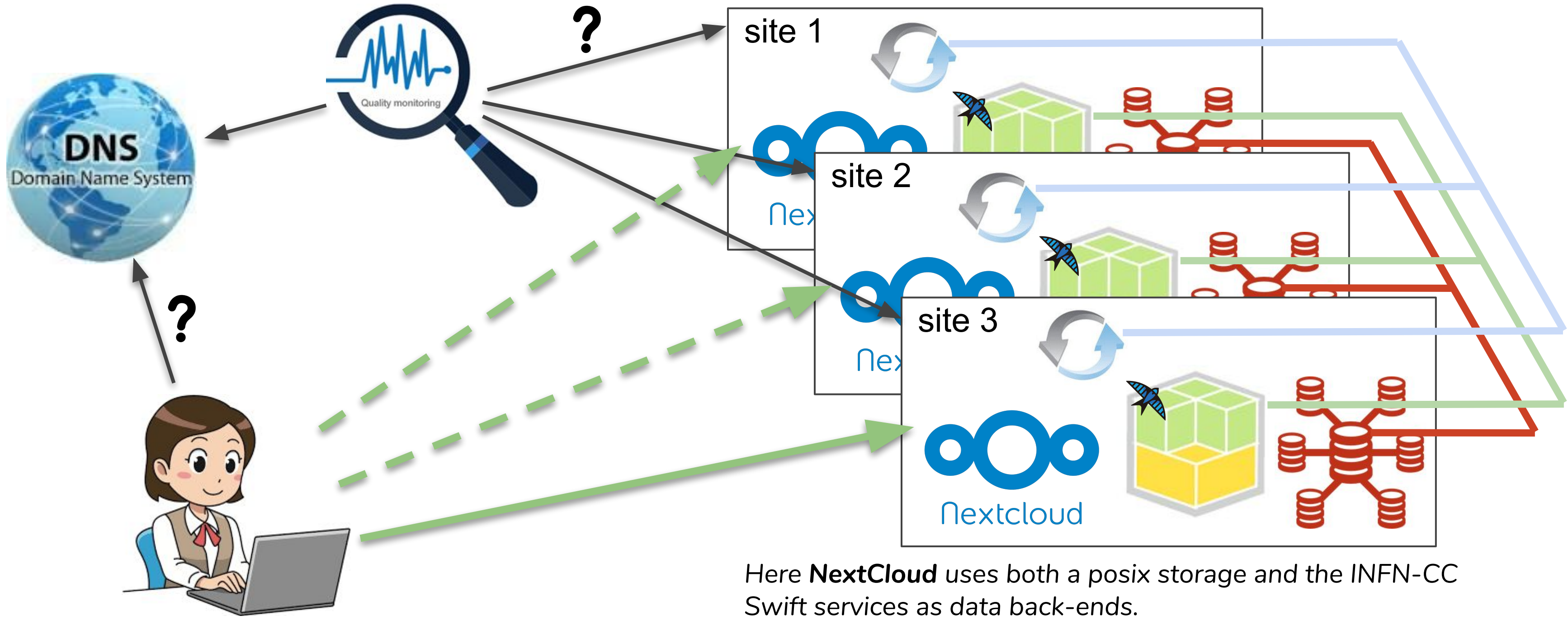
*In this **Redmine** test setup, a Swift plugin is used to provide the storage back-end while the DBMS is provided by a distributed SQL cluster .*

recipes: load balancer



In this ***Redmine*** *test setup, a Swift plugin is used to provide the storage back-end while the DBMS is provided by a distributed SQL cluster .*

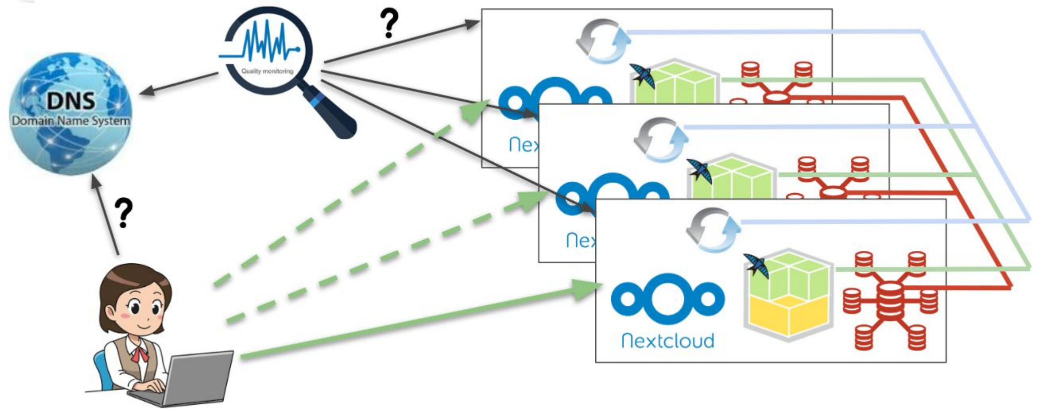
recipes: failover



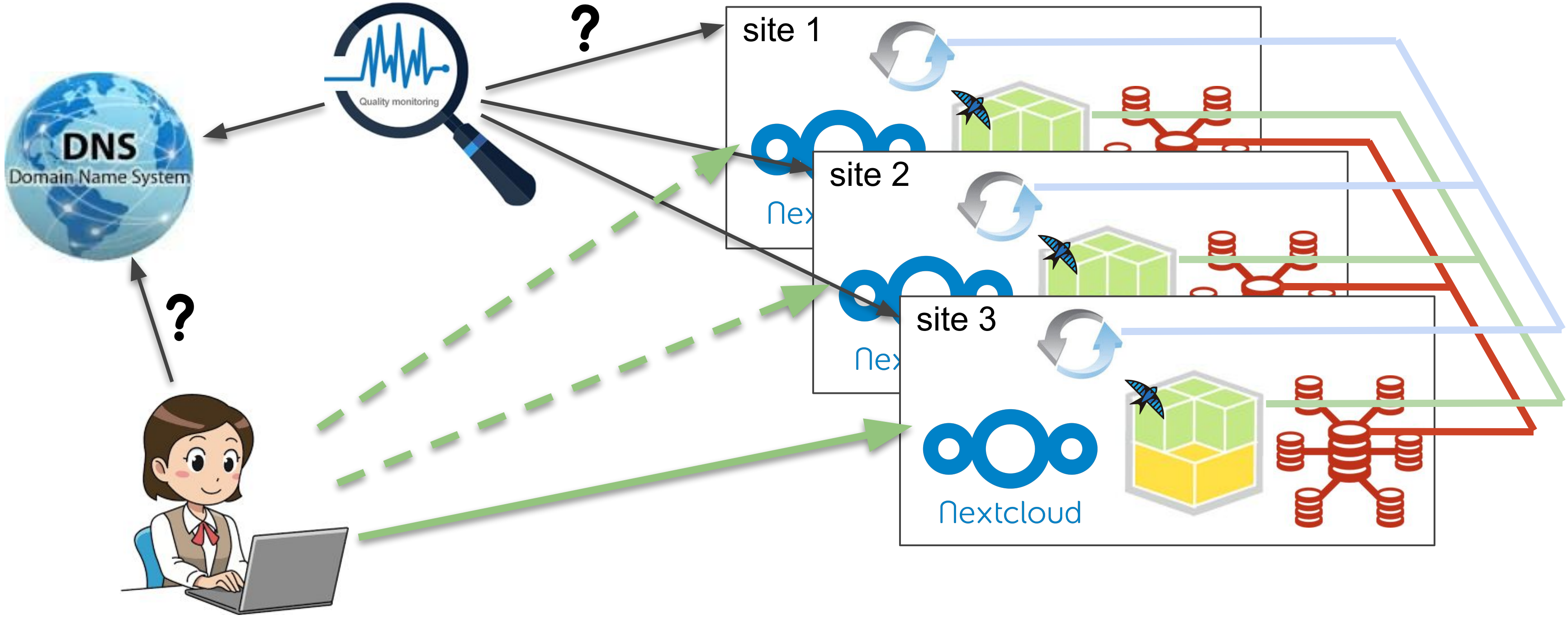
*Here **NextCloud** uses both a posix storage and the INFN-CC Swift services as data back-ends.*
Again the DBMS is provided by a distributed SQL cluster .
***Lsyncd** is in charge of keeping the posix volumes in sync.*

recipes: failback

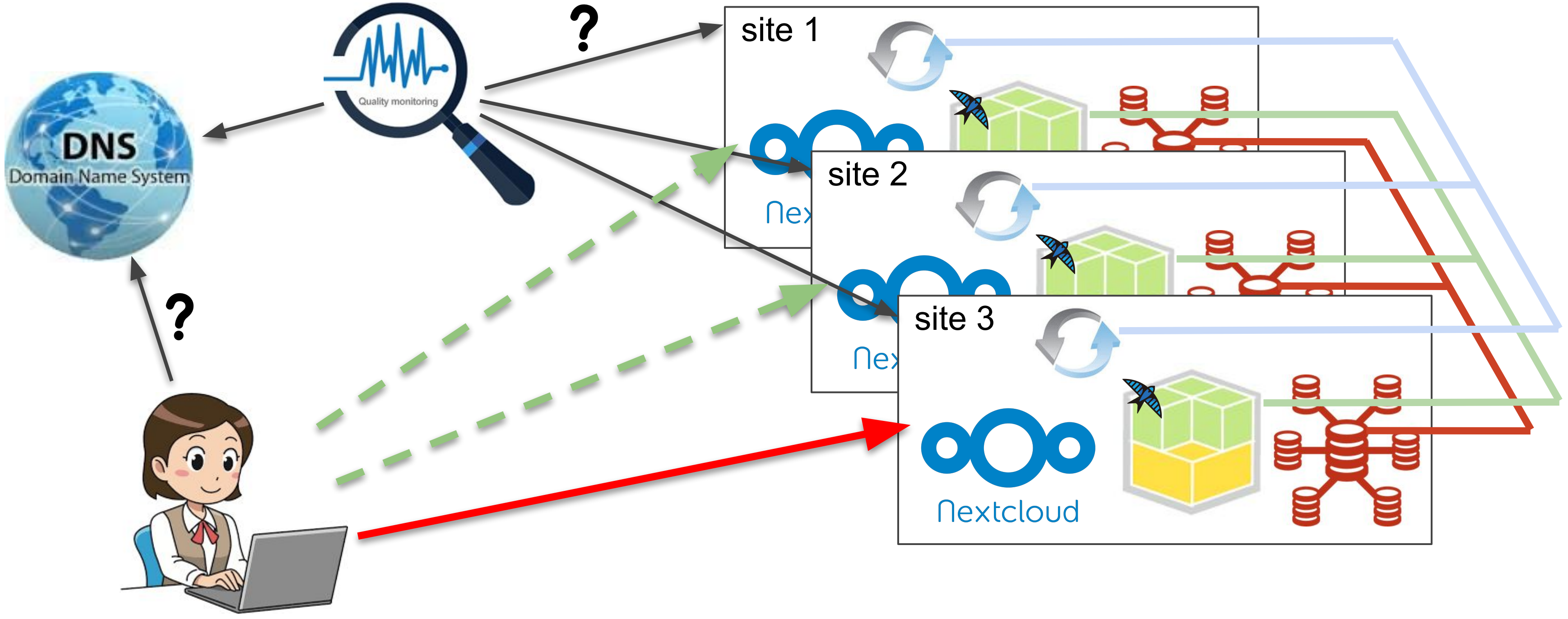
- Same as the previous case but a “master” server is defined.
- Whenever the master is healthy again after a failure the DNS will point to it again.



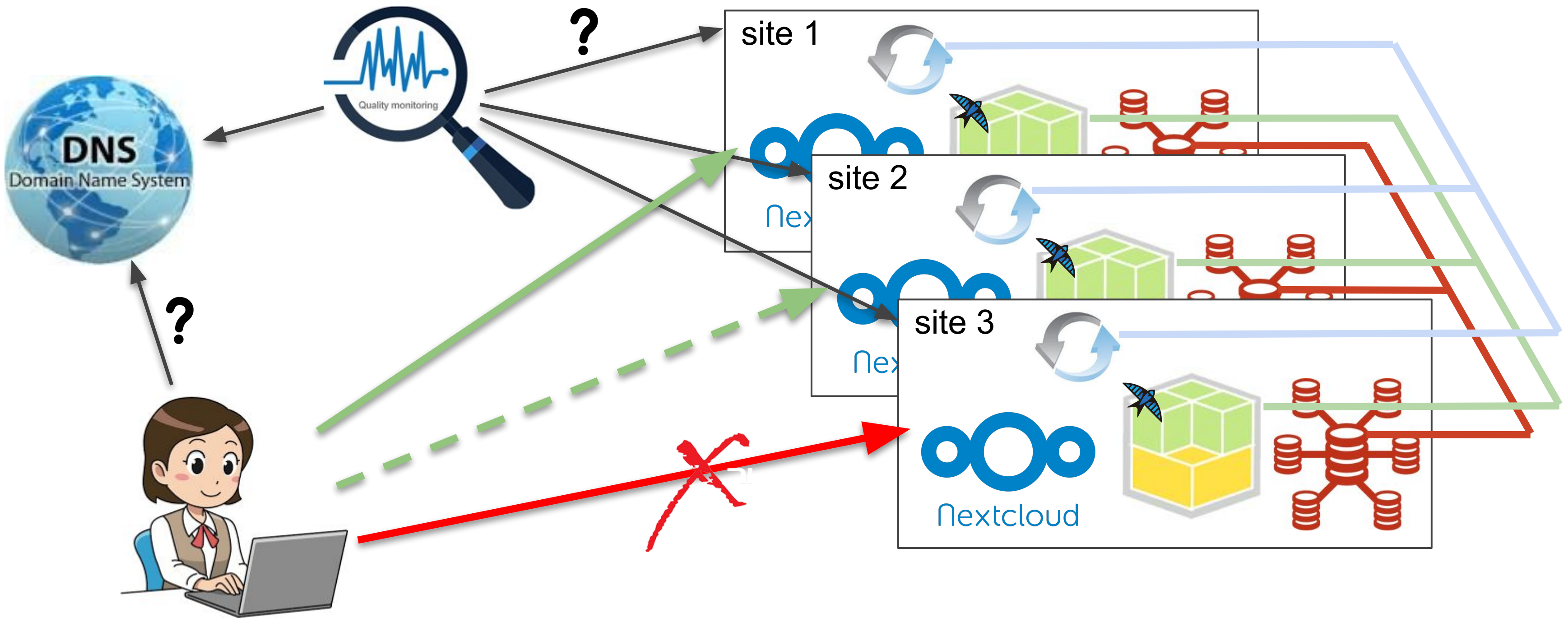
recipes: failover/failback



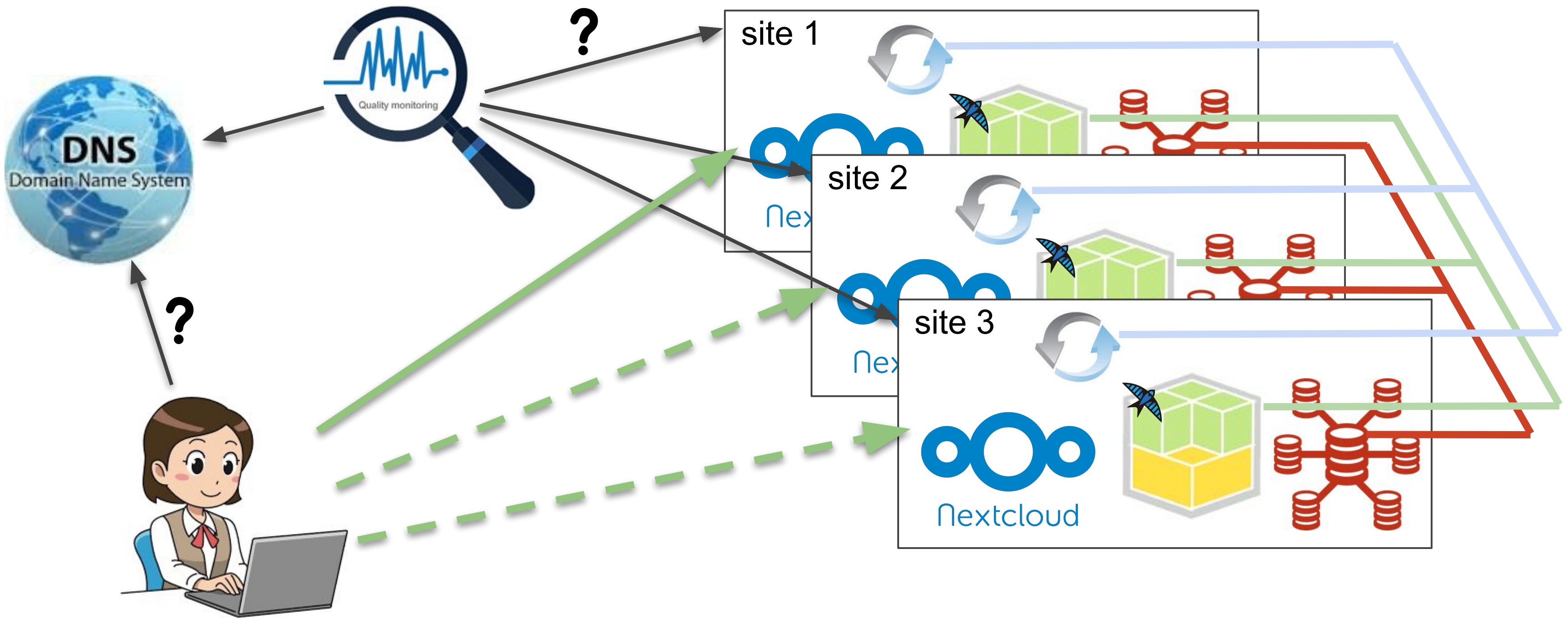
recipes: failover/failback



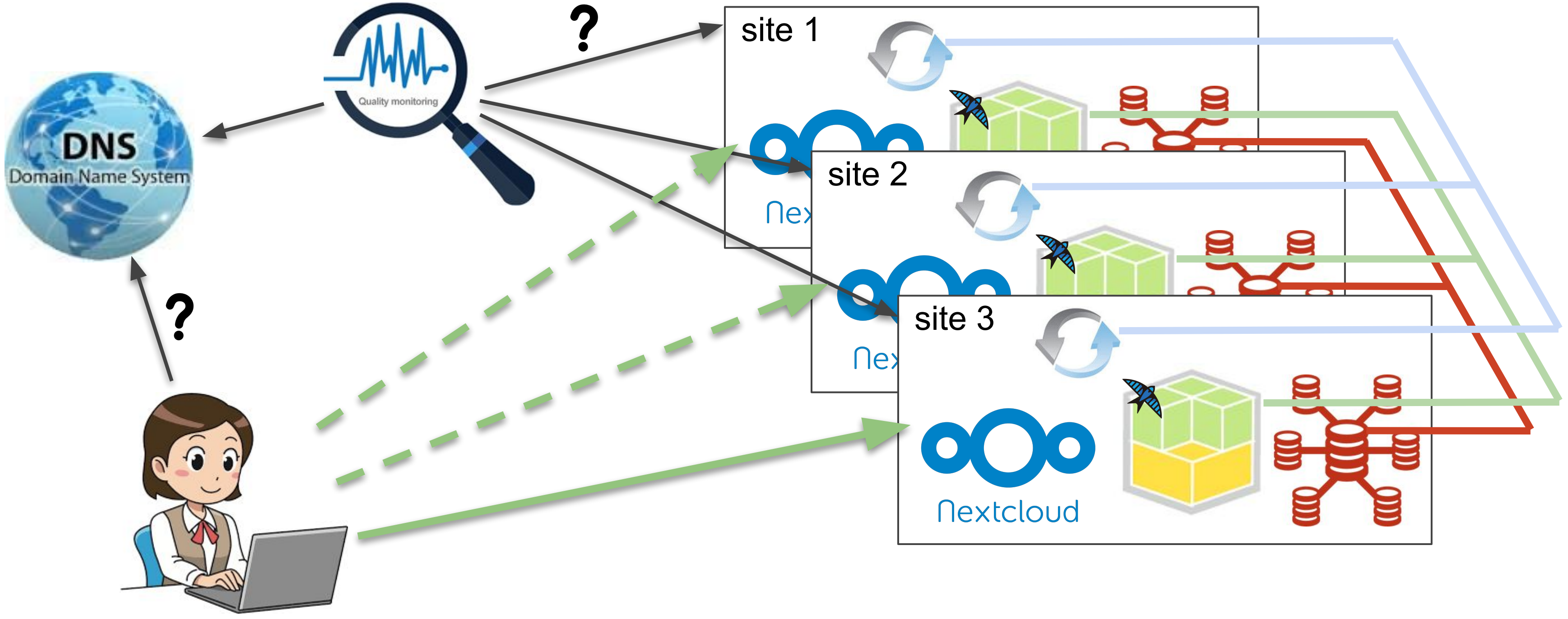
recipes: failover/failback



recipes: failover/failback

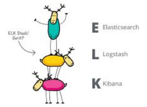


recipes: failback



Conclusions

- a geo-distributed multi-region OpenStack Cloud might be **sometimes difficult to deploy and manage** (debugging is crazy, but ELK helps!!!)



- it has **very interesting features and capabilities**
- together with some external instruments it can ease **breaking the barriers of the single data center**
- allows to build up natively geo-redundant application and services

Thanks for your attention