Keystone ~~OpenStack~~ in the context of Fog/Edge Massively Distributed Clouds

Fog/Edge/Massively Distributed Clouds (FEMDC) SIG

Beyond the clouds - The Discovery initiative

Who are We?

Adrien Lebre

Fog/Edge/Massively Distributed SiG co-chair
https://wiki.openstack.org/wiki/Fog_Edge_Massively_Distributed_Clouds

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

Marie Delavergne

Master Candidate at University of Nantes

Intern at Inria Discovery Initiative

Juice main developer
https://github.com/BeyondTheClouds/juice

Ronan-Alexandre Cherrueau

Fog/Edge/Massively Distributed SiG and Performance team Contributor

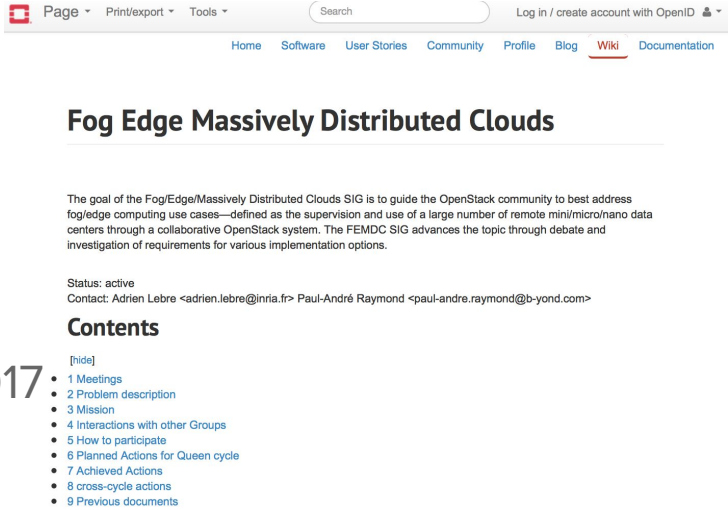
Discovery Initiative Researcher Engineer
EnOS main developer
http://enos.readthedocs.io

FEMDC SIG

Fog Edge Massively Distributed Clouds SIG

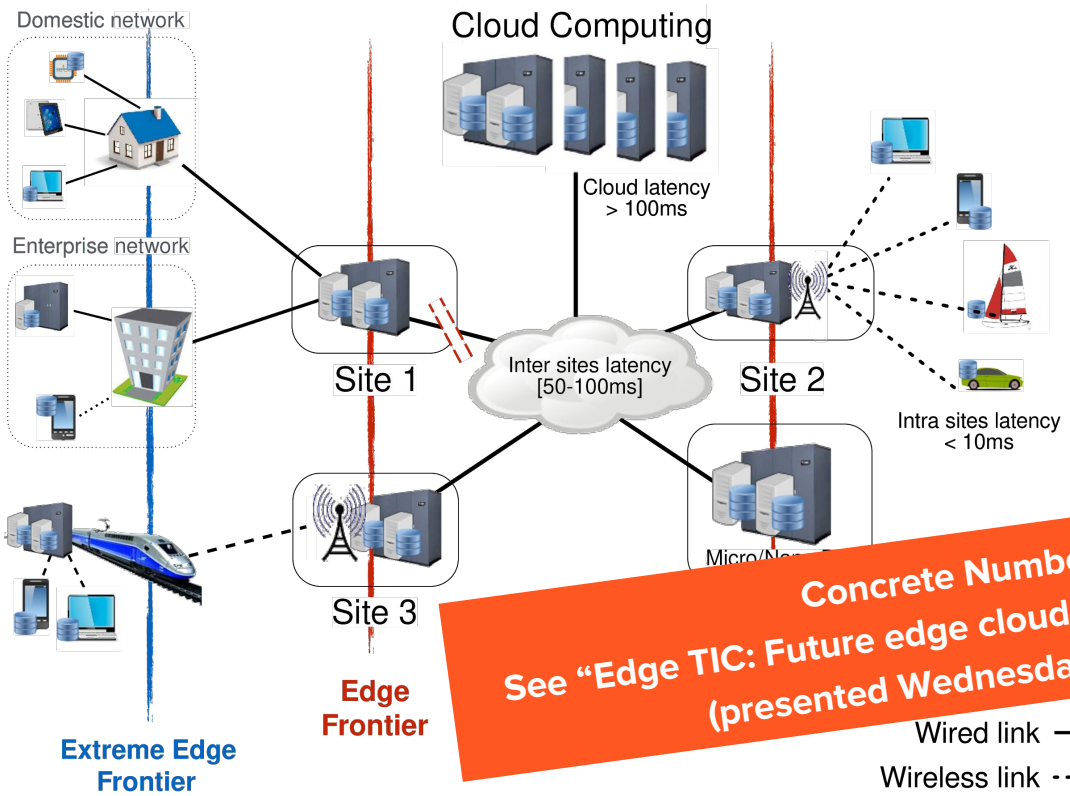
"Guide the OpenStack community to best address fog/edge computing use cases — defined as the supervision and use of a large number of remote mini/micro/nano data centers — through a collaborative OpenStack system."

- The FEMDC SIG advances the topic through debate and investigation of requirements for various implementation options.
- Proposed as a WG in 2016, evolved to a SIG in 2017
- IRC meeting every two weeks



https://wiki.openstack.org/wiki/Fog_Edge_Massively_Distributed_Clouds

Fog Edge Massively Distributed Clouds SIG (cont.)



Concrete Numbers?
See "Edge TIC: Future edge cloud for China Mobile" talk (presented Wednesday morning)

Fog Edge Massively Distributed Clouds SIG (cont.)

- Major achievements since 2016
  - EnOS/EnOS Lib - Understanding OpenStack Performance
    - Scalability (Barcelona 2016)
    - WANWide (Boston 2017)
    - OpenStack Deployments (Sydney 2017)
    - **AMQP Alternatives (Vancouver 2018)**
    - **Keystone/DB Alternatives (Vancouver 2018)**

OpenStack Performance studies (internal mechanisms and alternatives)

  - Identification of use-cases (Sydney 2017)
  - Participation to the writing of the Edge White Paper (Oct 2017-Jan 2018)
  - **Classification of requirements/impacts on the codebase** (Dublin PTG 2018, **Vancouver 2018**, HotEdge 2018)
  - **Workloads control/automation needs (Vancouver 2018)**

Use-cases/requirements specifications

LET'S START

Motivations

- “Can We Operate and Use an Edge Computing Infrastructure with OpenStack?”
  - Inter/Intra-services collaborations are mandatory between key services (Keystone, Nova, Glance, Neutron)
    Start a VM on Edge site A with VMI available on site B, Start a VM either on Site A or B, ...
  - Extensions vs new mechanisms
  - Top/down and Bottom/Up
- How to deliver such collaboration features: the keystone use-case?
  - Top/Down approach: extensions/revisions of the default keystone workflow)
    - Federated keystone or keystone to keystone
    - Several presentations/discussions this week (see the schedule)
  - Bottom/up: revise low level mechanisms to mitigate changes at the upper level

Agenda

1. Storage Backend Options
2. Juice, a performance framework
3. Evaluations
4. Wrap up

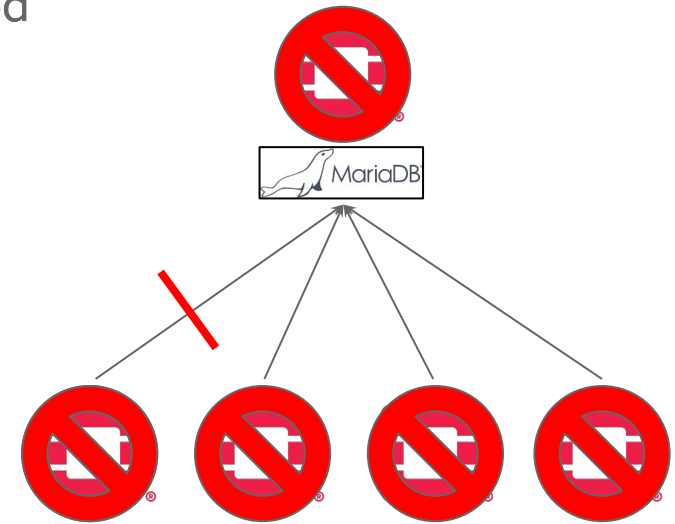
Option 1: Centralized MariaDB

Each instance has its own Keystone but a centralized MariaDB for all:

- Every Keystone refers to MariaDB in the OpenStack instance that stores it
- Easy to setup/maintain
- Scalable enough for the expected load

Possible limitations

- Centralized MariaDB is a SPoF
- Network disconnection leads to instance unusability



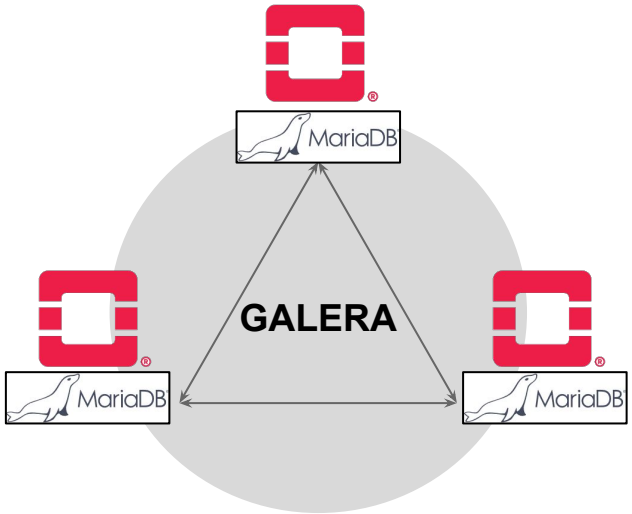
Option 2: Synchronization using Galera

Each instance has its own Keystone/DB. DBs are synchronized thanks to Galera:

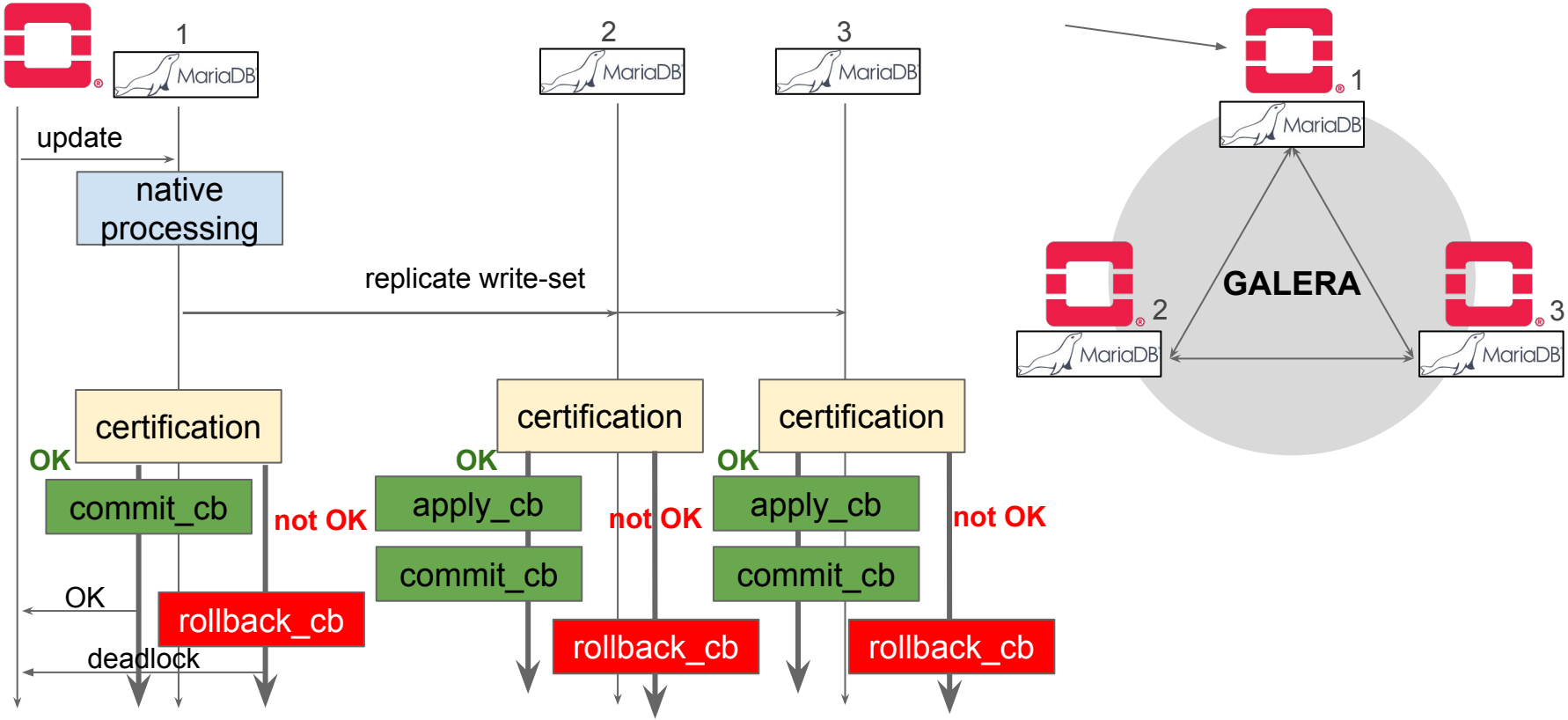
- Multi-master topology
  - Synchronously replicated
  - Allows reads and writes on any instances
  - High availability

Possible limitations

- Synchronous replication on high latency networks
- Galera clustering scalability
- Cluster partition/resynchronization



Option 2: Synchronization using Galera



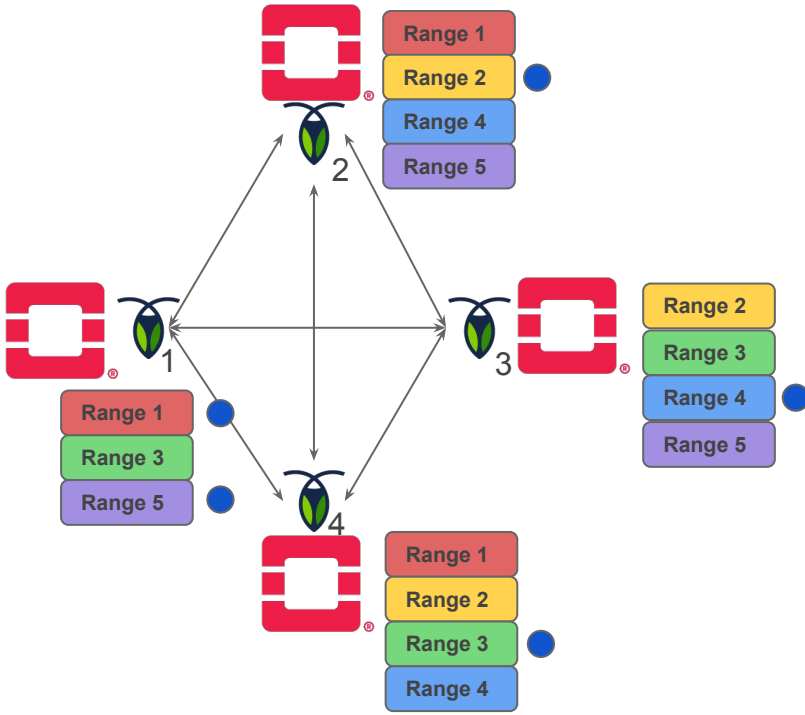
Option 3: Geo-Distributed CockroachDB

Each instance has its own Keystone using the **global geo-distributed CockroachDB**:

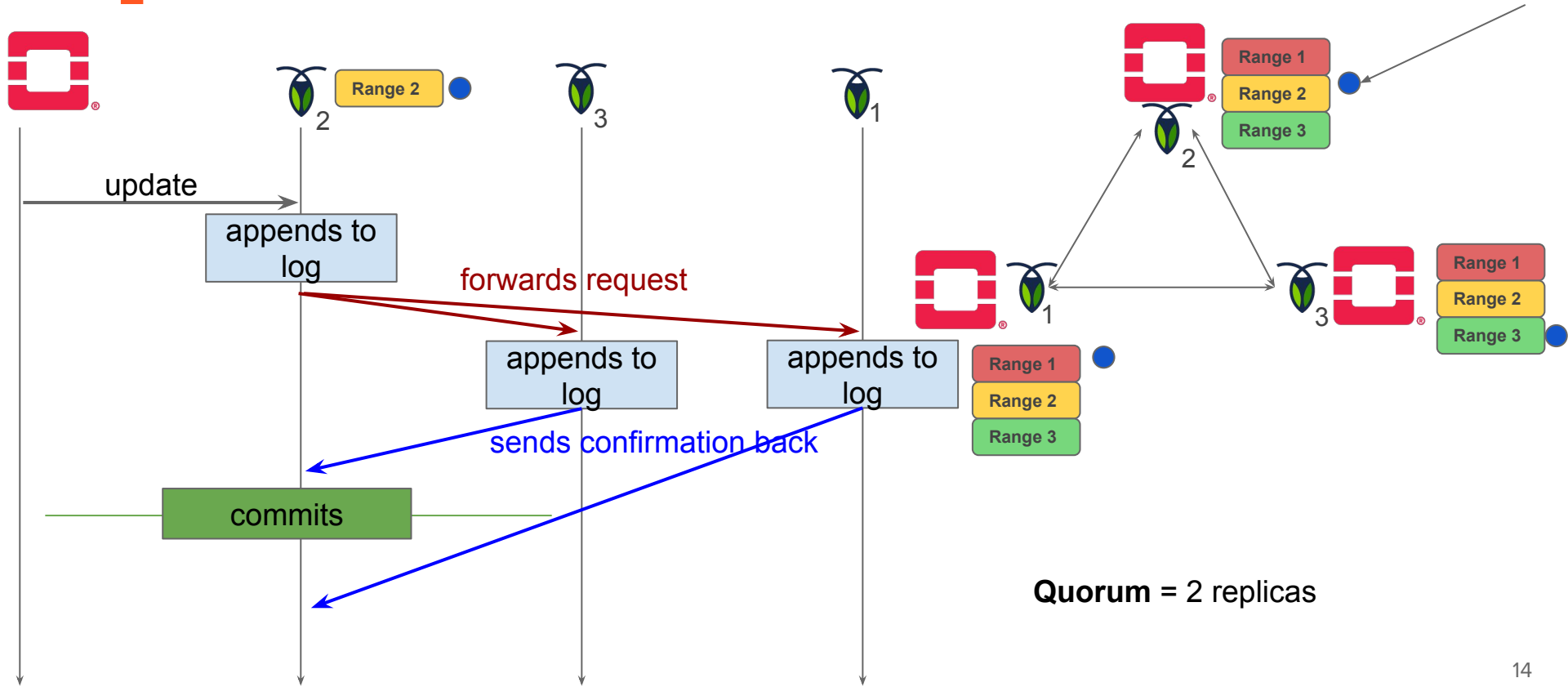
- A key-value DB with SQL interface (enabling "straightforward" OpenStack integration)
  - Tables are split into ranges
  - Ranges are distributed/replicated across **selected** peers

Possible limitations

- Distribution/replication on high latency network
- Network split/resynchronization
- Transaction contentions



Option 3: Geo-Distributed CockroachDB

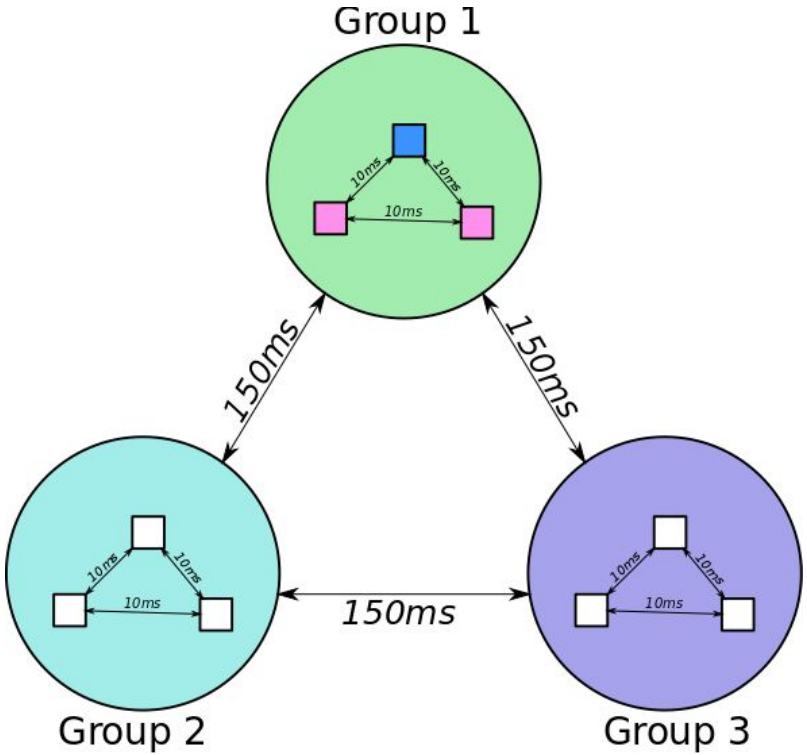


Quorum = 2 replicas

Option 3: Geo-Distributed CockroachDB

Locality matters!

- Replicas/Quorum location impact
  - 1/2/3 replicas in the same site
- The nodes are placed on different sites
  - Each sites are separated by 150 ms
  - Latency between nodes in a site is 10ms
- Allows to understand the behaviour of differents datacenters across continents



Juice: Conduct Edge evaluations with DevStack

Juice 🚀

https://github.com/BeyondTheClouds/juice

- Motivation: Conducting DevStack based performance analyses with defined storage backends
  - **In a scientific and reproducible manner (automated)**
  - At small and large-scale
  - Under different network topologies (traffic shaping)
  - With the ability to add a new database easily

- Built on **Enoslib** **https://github.com/BeyondTheClouds/enoslib**
- Workflow
  - `$ juice deploy`
  - `$ juice rally`
  - `$ juice backup`

Juice deploy/openstack

- Deploy your storage backend environment, OpenStack with DevStack, and the required control services
- Emulate your Edge infrastructure by applying traffic shaping rules

To add a database, you simply have to add in the database folder:

- a *deploy.yml* that will deploy your database on each (or one) region
- a *backup.yml* to backup the database if you want
- a *destroy.yml* to ensure reproducibility throughout the experiments

Then add the name of your database in *juice .py deploy*

Finally, add the appropriate library to connect your services to the DB

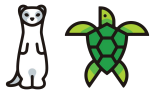
Juice rally/sysbench

- Run the wanted tests for Rally and sysbench
- To run the sysbench test:

```
$ juice stress
```

- Allows to run any rally scenario:

```
$ juice rally --files
keystone/authenticate-user-and
-validate-token.yaml
```



Juice backup/destroy

- `juice backup` produces a tarball with:
  - Rally reports
  - InfluxDB database with cAdvisor/Collectd measures
  - the database if configured to do so

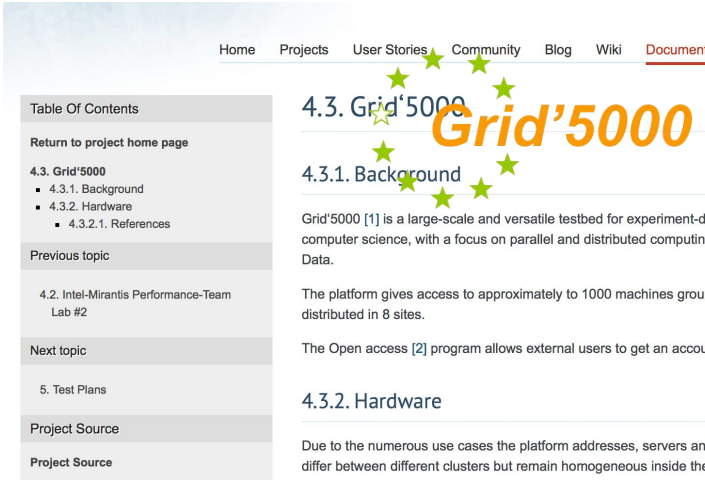
https://github.com/collectd/collectd
https://github.com/influxdata/influxdb
https://github.com/google/cadvisor

- `juice destroy` removes everything needed to begin a new experiment from a clean environment

Evaluations

Experimentation

- Evaluate a distributed Keystone using the three previous bottom/up options
  - Juice deploys OpenStack instances on several nodes
  - OS services are disabled except Keystone
  - Keystone relies on MariaDB, or Galera, or CockroachDB to store its state
- One of the world-leading **testbeds for Distributed Computing**
  - 8 sites, 30 clusters, 840 nodes, 8490 cores
  - Dedicated 10Gbps backbone network
  - Design goal: Support **high-quality, reproducible experiments**
    (i.e. in a fully controllable and observable environment)



Experimental Protocol (Parameters)

- OpenStack instances number
  - **[3, 9, 45]**
  - LAN link between each OpenStack instance
  - Does the number of OpenStack instances impact completion time?
- Homogeneous network latency
  - 9 OpenStack instances
  - **[LAN, 100, 300] ms RTT**
  - Does the network latency between OpenStack instances impact completion time?
- Heterogeneous network latency
  - **3 groups of 3 OpenStack instances**
  - **20 ms of network delay between OpenStack instances of one group**
  - **300 ms of network delay between groups**

Experimental Protocol (Rally Load )

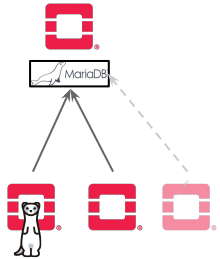
- Rally scenarios (%reads, %writes)
  - **Authenticate and validate a keystone token (96.46, 3.54)**
  - Create user role, add it and list user roles for given user (96.22, 3.78)
  - Create a keystone tenant with random name and list all tenants (92.12, 7.88)
  - Get instance of a tenant, user, role and service by id's (91.9, 8.1)
  - **Create user and update password for that user (89.79, 10.21)**
  - Create a keystone user and delete it (91.07, 8.93)
  - Create a keystone user with random name and list all users (92.05, 7.95)
- Load mode
  - **Light**: starts a Rally in **one** OpenStack instance
    - With 45 OpenStack instances:
      - 10 constant concurrent requests
      - 100 iterations
  - **High**: starts a Rally in **each** OpenStack instances
    - With 45 OpenStack instances:
      - 450 constant concurrent requests
      - 4,500 iterations

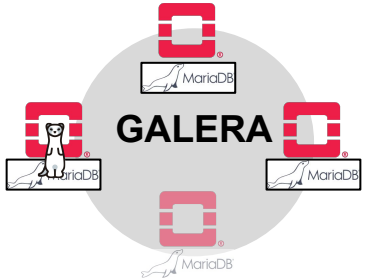
Generates a lot of contention on the distributed RDBMS

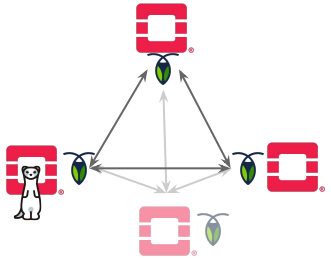
Auth. & Validate Keystone Token (1)

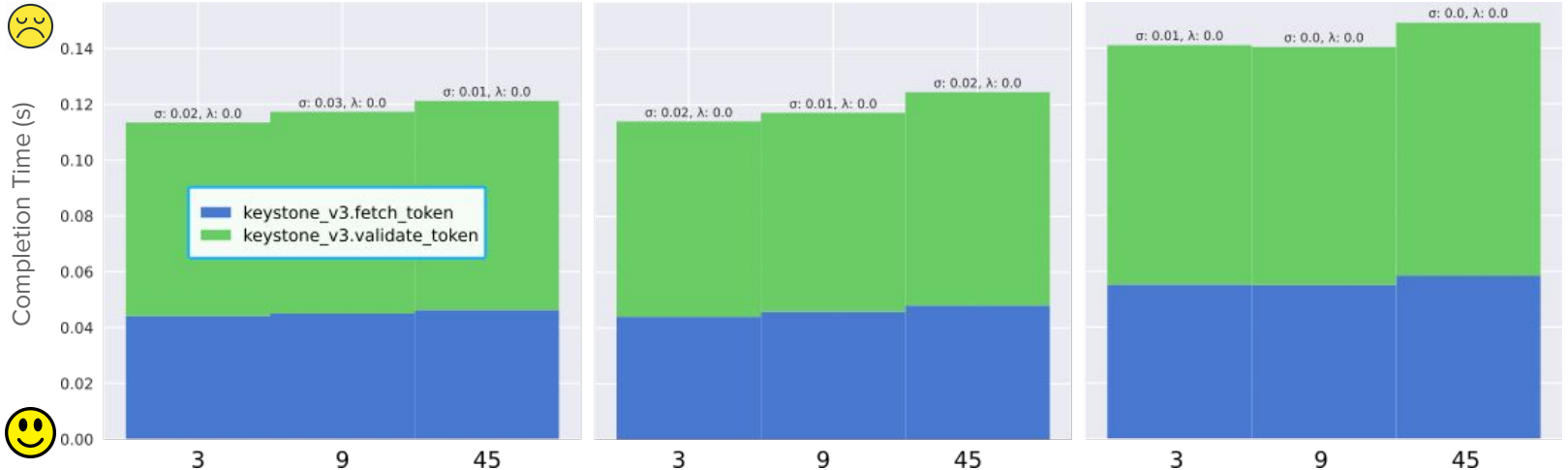
Impact of the **number of OpenStack** instances, %r: 96.46, %w: 3.54, **Light Load**

OpenStack Instances
[3, 9, 45]





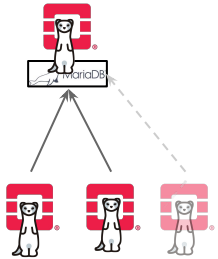


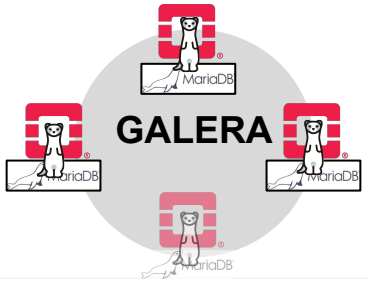


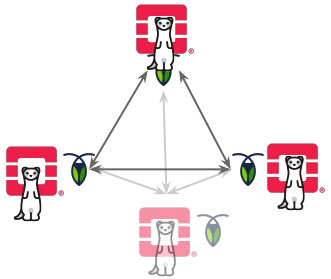
Auth. & Validate Keystone Token (2)

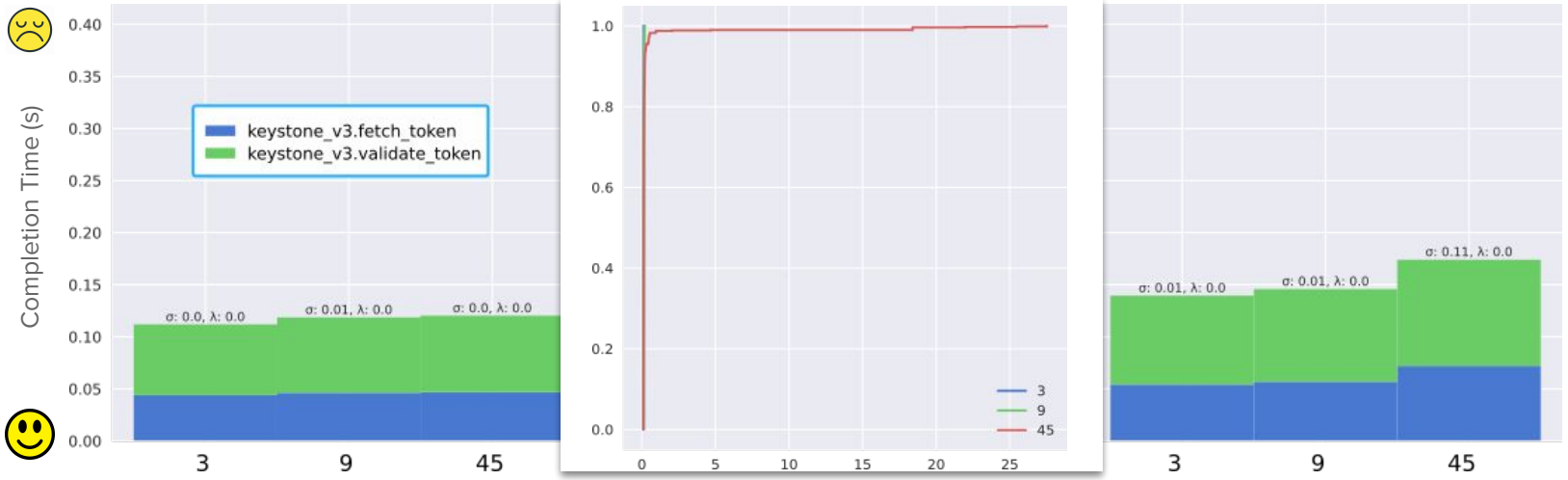
Impact of the **number of OpenStack** instances, %r: 96.46, %w: 3.54, **High Load**

OpenStack Instances
[3, 9, 45]



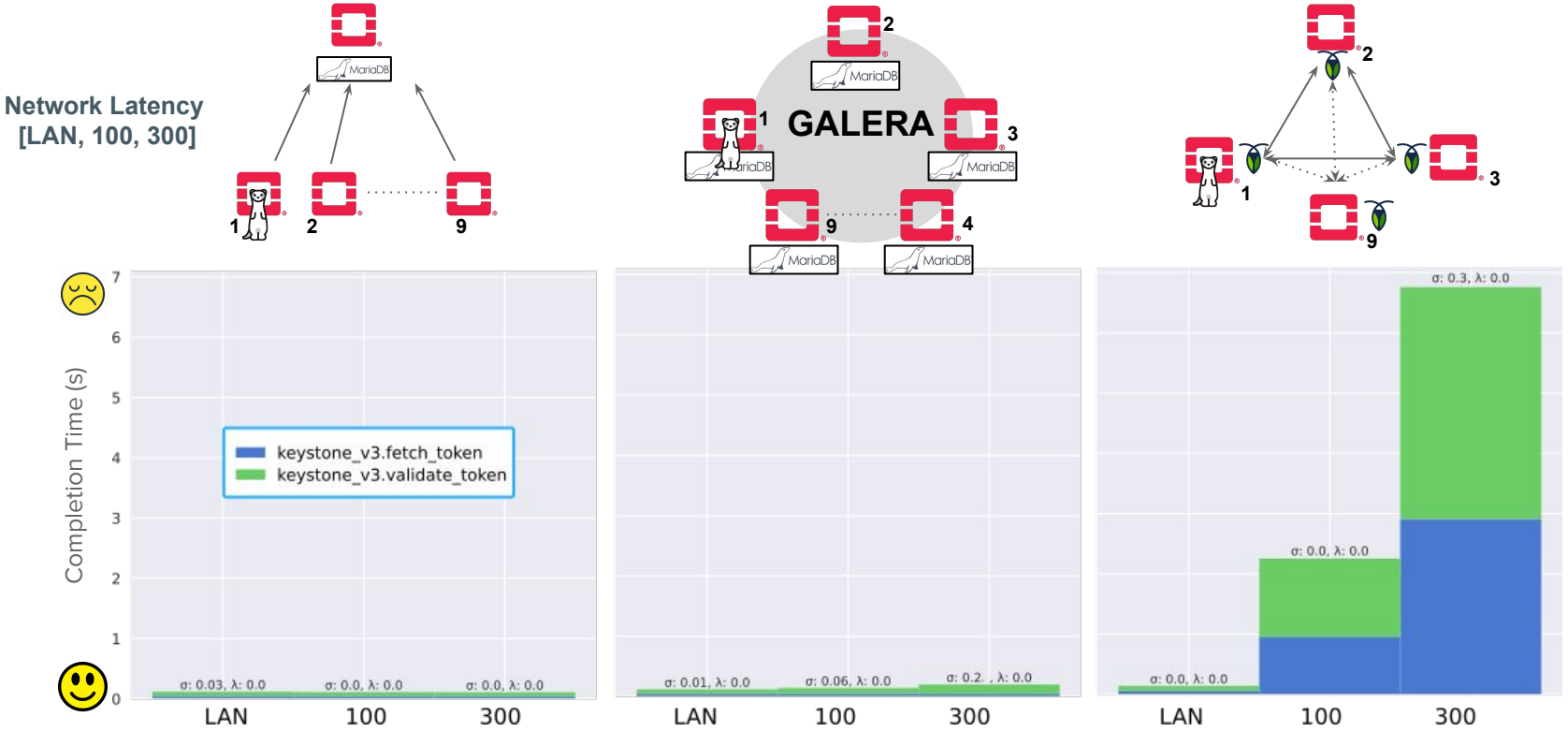






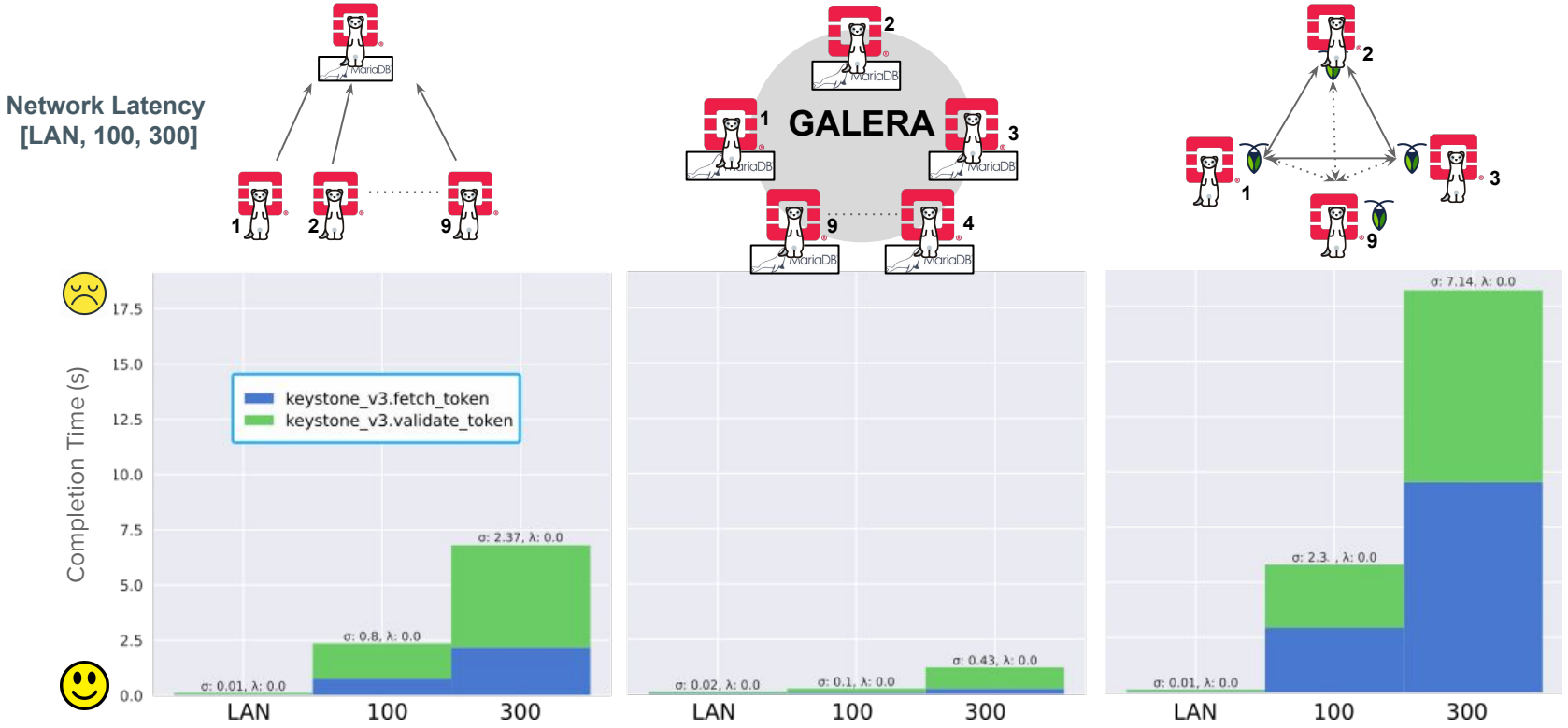
Auth. & Validate Keystone Token (3)

Impact of the **network delay** between OS instances, %r: 96.46, %w: 3.54, **Light Load**



Auth. & Validate Keystone Token (4)

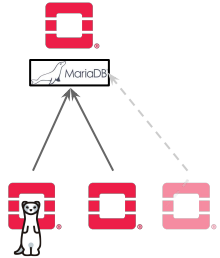
Impact of the **network delay** between OS instances, %r: 96.46, %w: 3.54, **High Load**

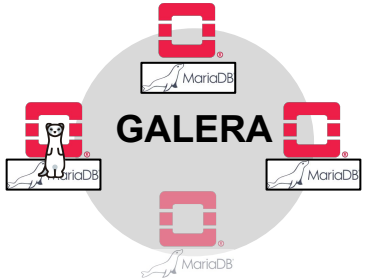


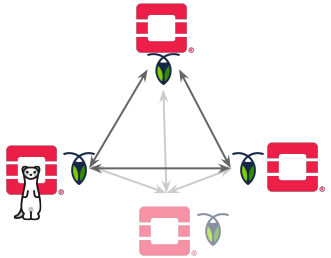
Create User & Update its Pwd (1)

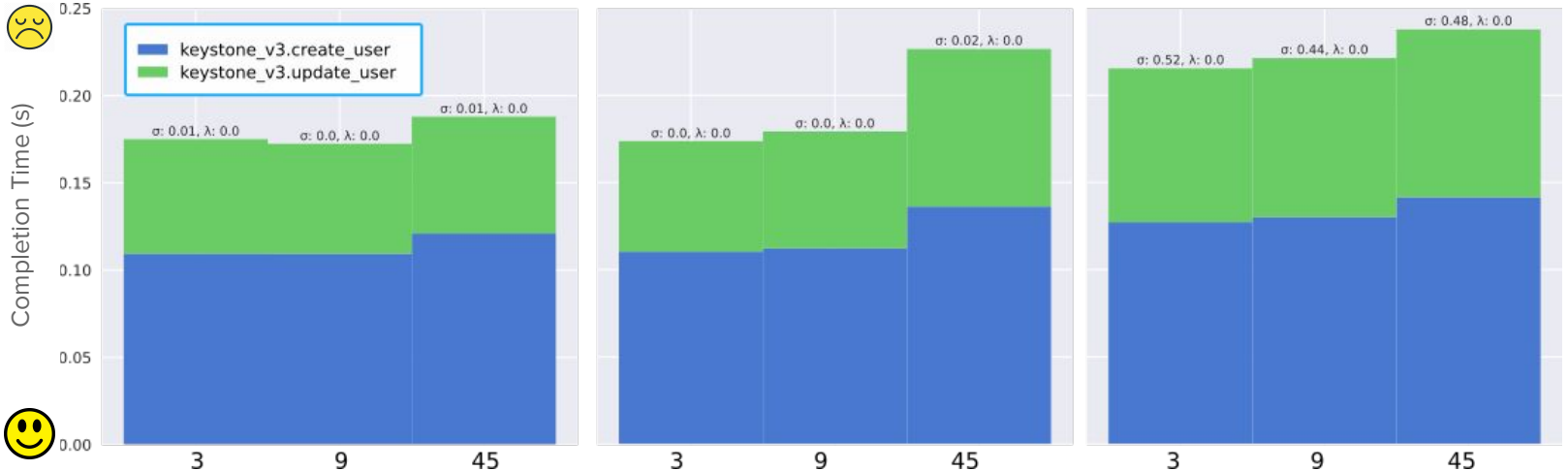
Impact of the **number of OpenStack** instances, %r: 89.79, %w: 10.21, **Light Load**

OpenStack Instances
[3, 9, 45]





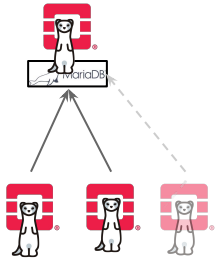


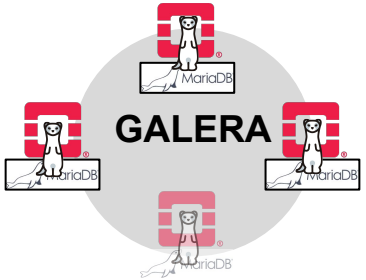


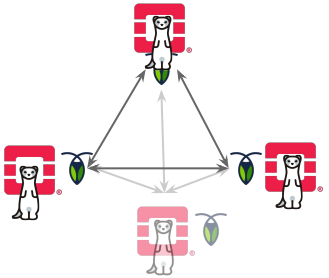
Create User & Update its Pwd (2)

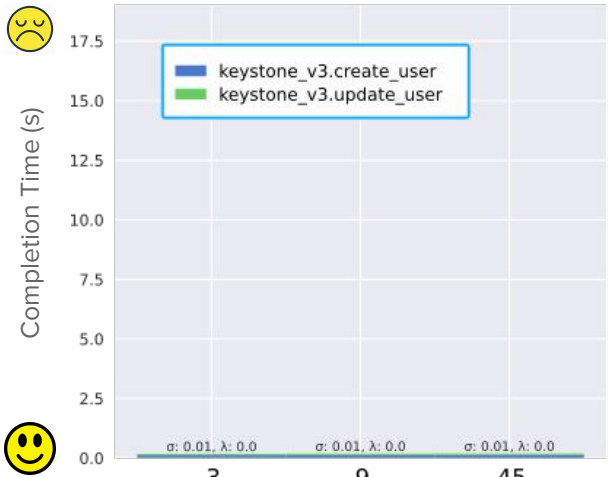
Impact of the **number of OpenStack** instances, %r: 89.79, %w: 10.21, **High Load**

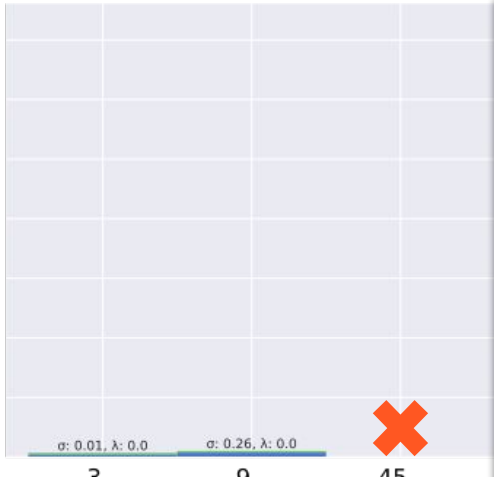
OpenStack Instances
[3, 9, 45]

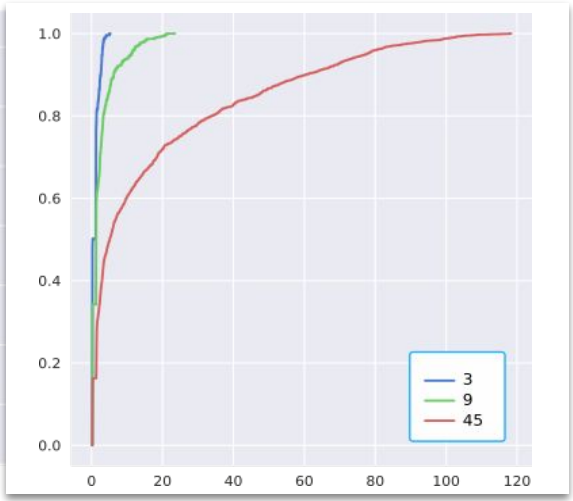






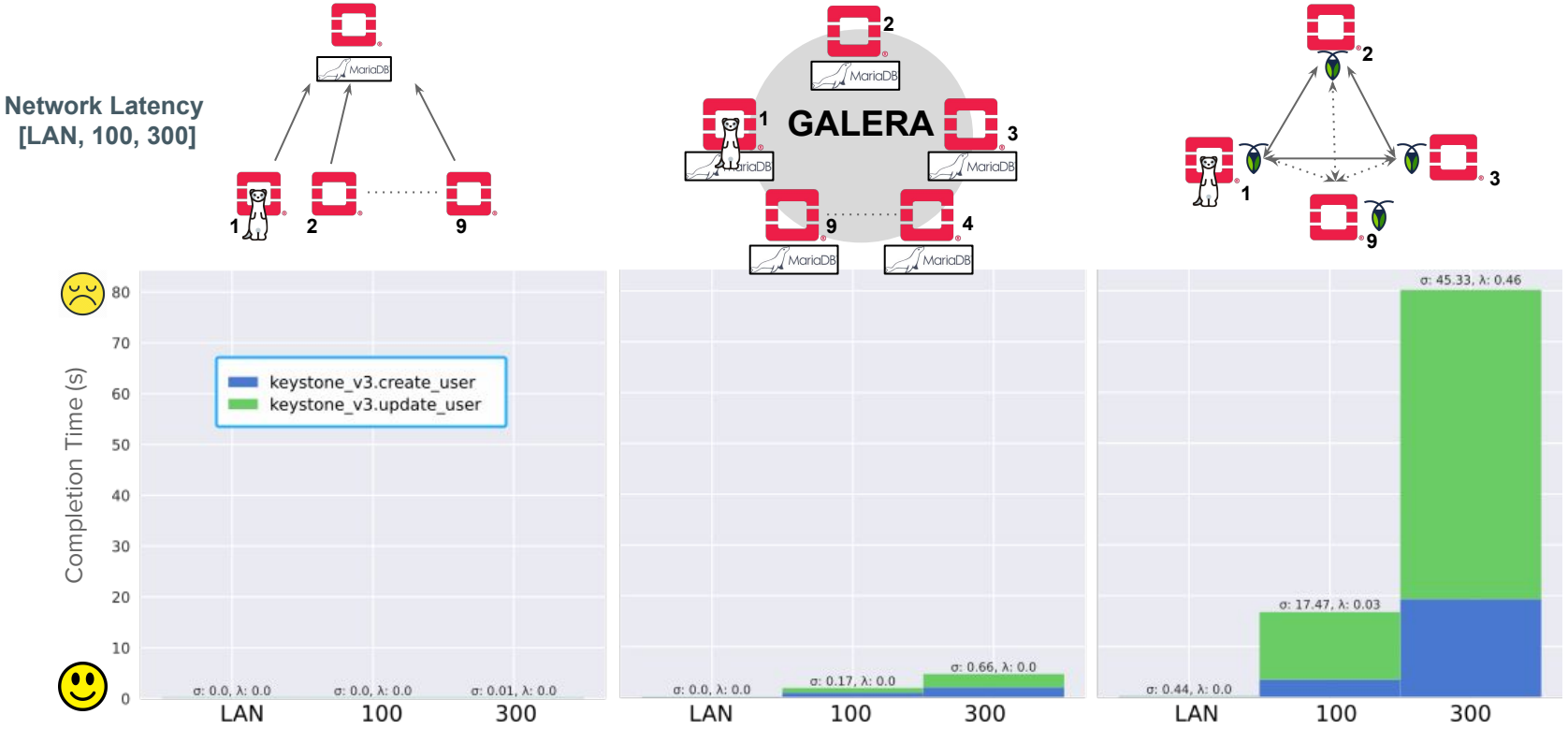






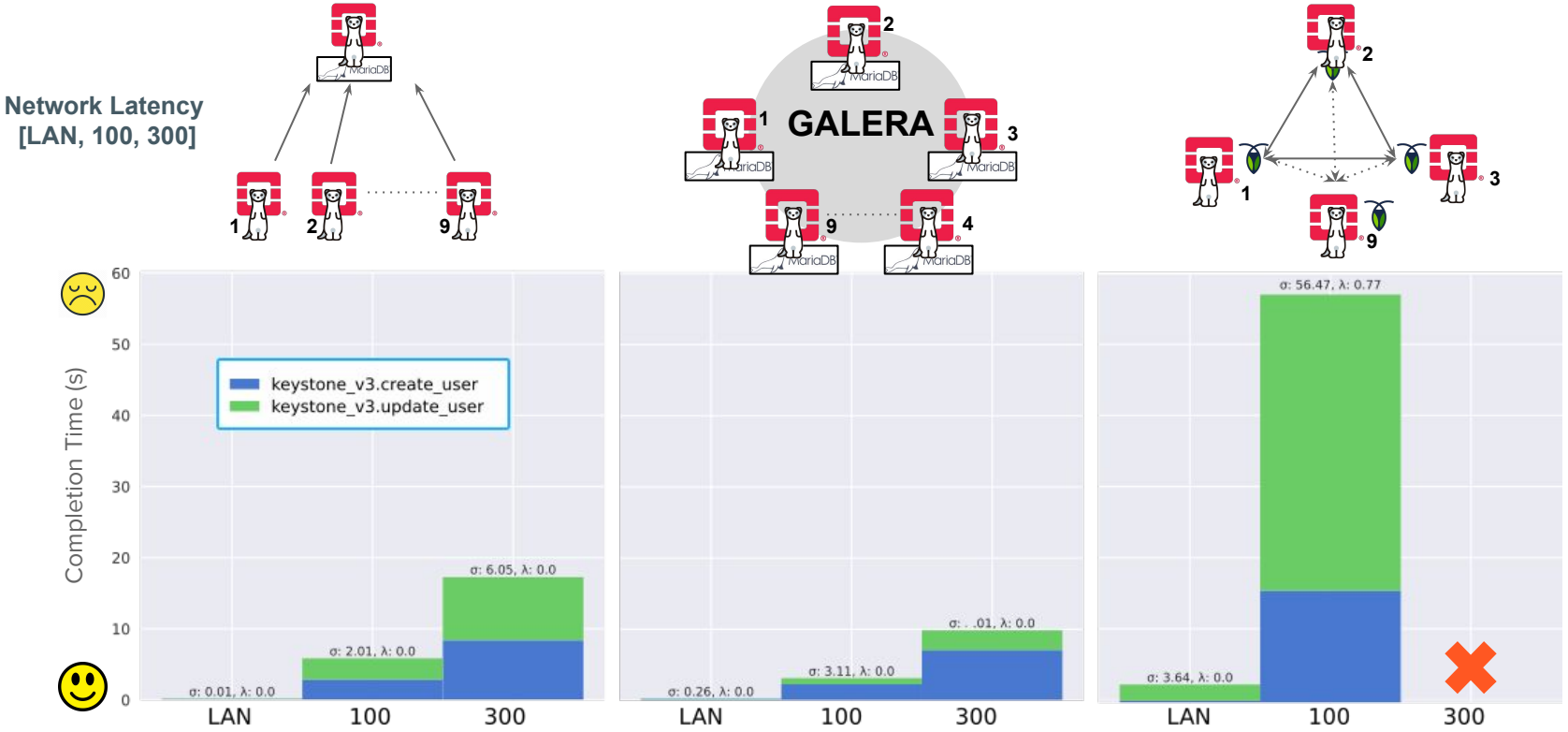
Create User & Update its Pwd (3)

Impact of the **network delay** between OS instances, %r: 89.79, %w: 10.21, **Light Load**



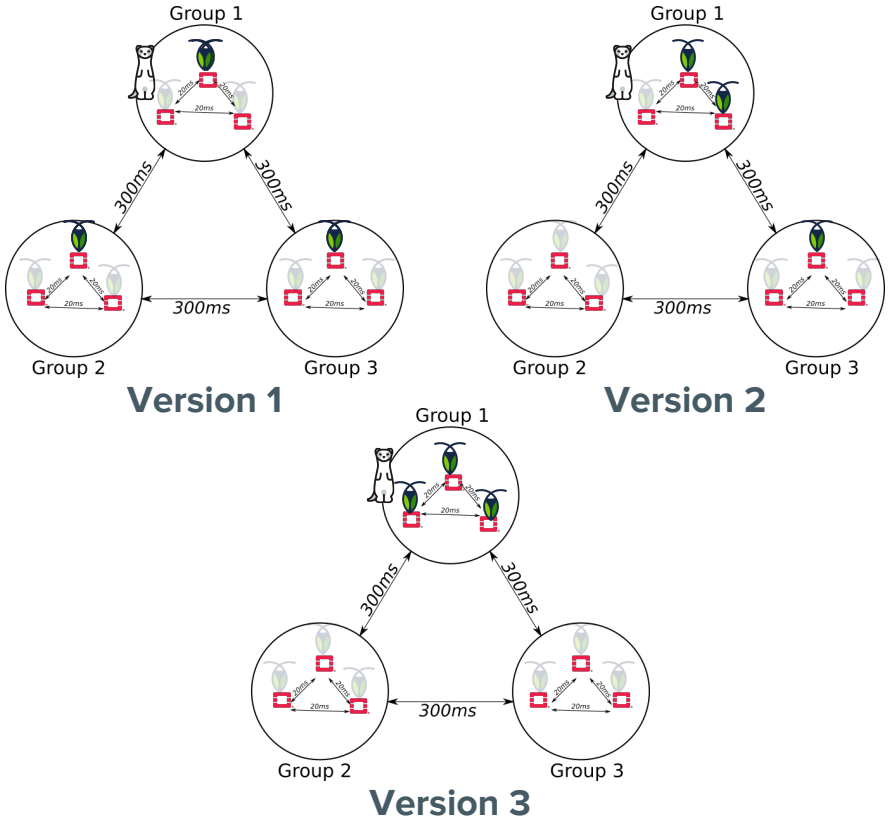
Create User & Update its Pwd (4)

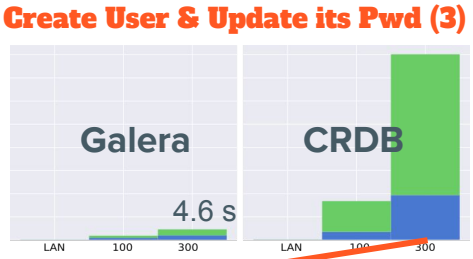
Impact of the **network delay** between OS instances, %r: 89.79, %w: 10.21, **High Load**

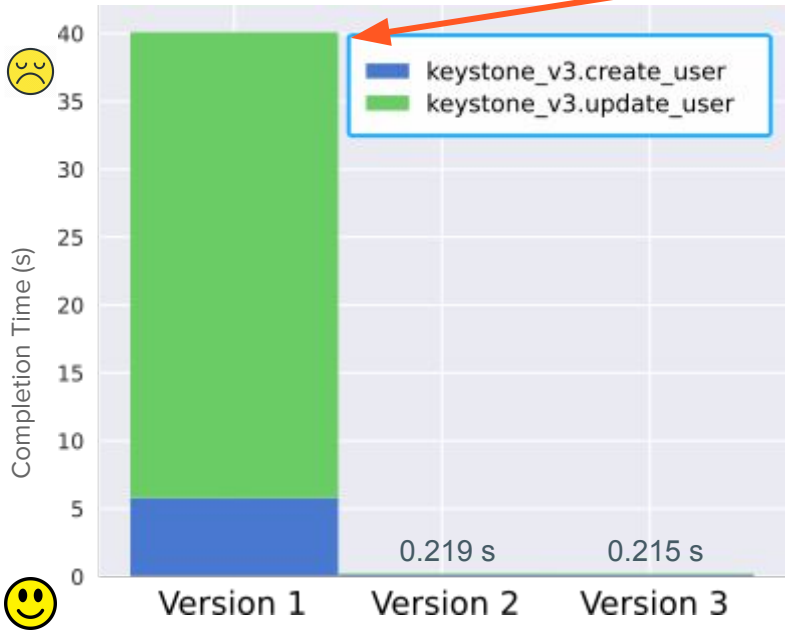


Create User & Update its Pwd (5)

Importance of data **locality**, %r: 89.79, %w: 10.21, **Light Load**







Wrap up

Summary

Evaluate the relevance of a geo-distributed DB in comparison to the usual Galera proposal to deliver a global view.

- Galera performance for keystone is rather good in comparison to our initial expectations and the MariaDB reference
  - Multi-master on high network latency is OK
  - Clustering scalability on high load is NOK
- In case of high network latency CockroachDB internal mechanisms face important overheads, but read performance is in the same order of magnitude of other solutions.
  - Write access can benefit from locality awareness.
- Additional investigations should be performed to understand corner-cases

Summary (cont.)

CockroachDB for OpenStack

- Discovery blog: A POC of OpenStack Keystone over CockroachDB (Dec 2017)
- oslo.db: https://github.com/BeyondTheClouds/oslo.db
  Add 3 lines to handle CockcorachDB exceptions.
- keystone: https://github.com/BeyondTheClouds/keystone
  Add 11 `retry_on_deadlock` , usefull for CockroachDB and Galera
- sqlalchemy-migrate: https://github.com/BeyondTheClouds/sqlalchemy-migrate
  partial support of CockroachDB schema migration (require some efforts ;))

More results/graphs available soon on our blog (see the FEMDC/Edge mailing list)

Take away message

- Collaborations between edge sites require data-information sharing
- Sharing can be done either through remote calls or storage backends
- To mitigate data exchanges only when they are required, data locality capabilities are mandatory:
  - A communication bus adapted to edge computing infrastructures
    “OpenStack internal messaging at the edge : in depth evaluation” (see the online video)
  - A data storage backend dedicated to edge computing infrastructures
    This talk, a starting point
- Understanding the impact of infrastructure resiliency is needed
  - Intermittent networks
  - Edge site apparitions/removals

Keystone as an initial use-case!
Can we apply similar approaches to other (OpenStack) services?

Keystone OpenStack in the context of Fog/Edge Massively Distributed Clouds

Thanks

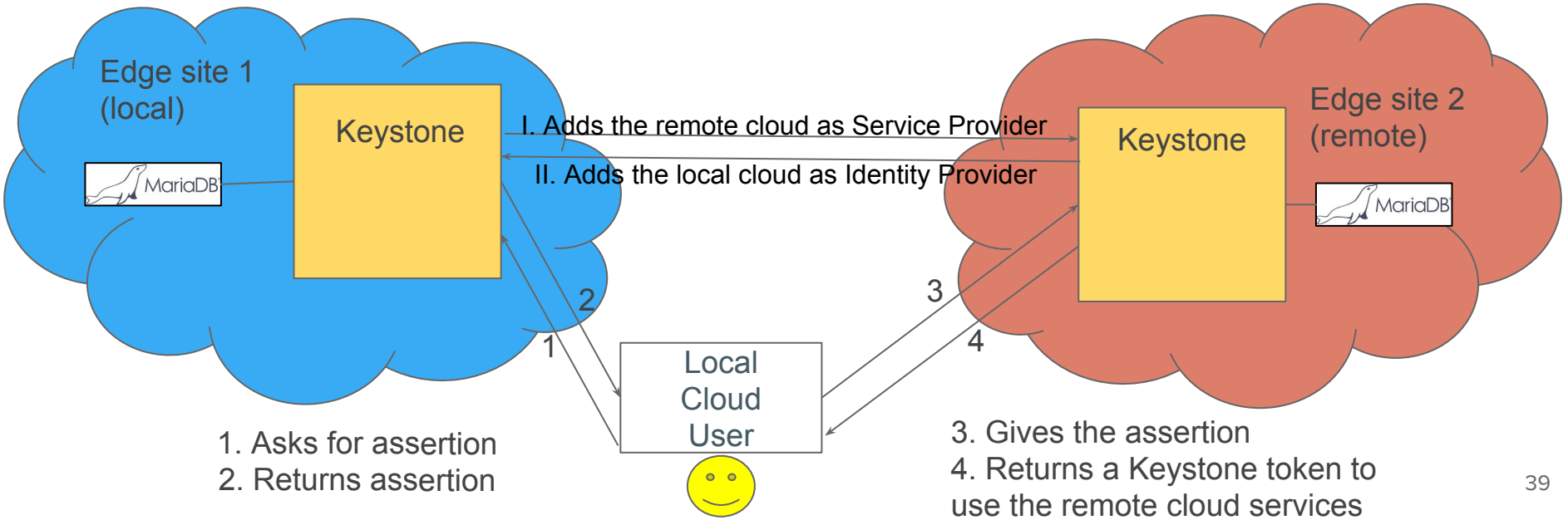
@BeyondClouds_io
http://beyondtheclouds.github.io

Juice deploy/openstack

- Deploy your storage backends environment, OpenStack with DevStack, and the required control services
- Install your services, OpenStack (Keystone) and the required control services
- To add a database, you simply have to add in the database folder:
  - a *deploy.yml* that will deploy your database on each (or one) region
  - a *backup.yml* to backup the database if you want
  - a *destroy.yml* to ensure reproducibility without having to deploy a new Debian
- Then add the name of your database in *juice.py deploy*
- Finally, add the appropriate lib for Keystone so it can connect

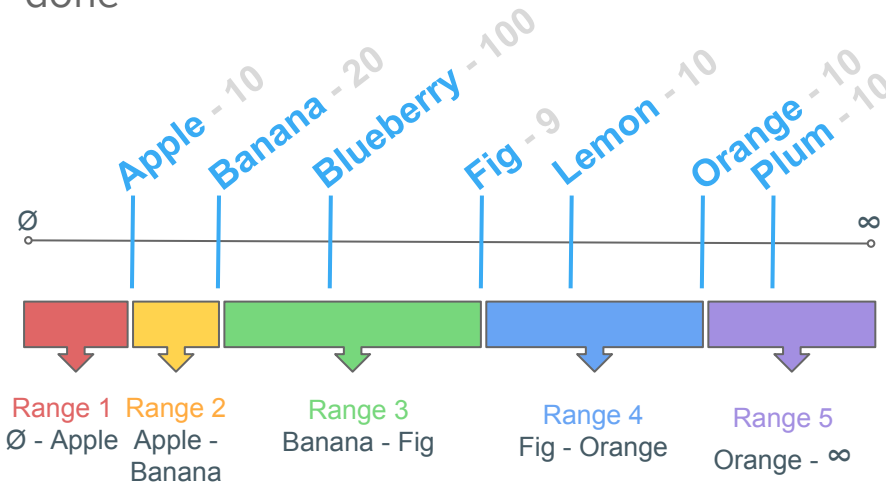
Federated Keystone (not evaluated)

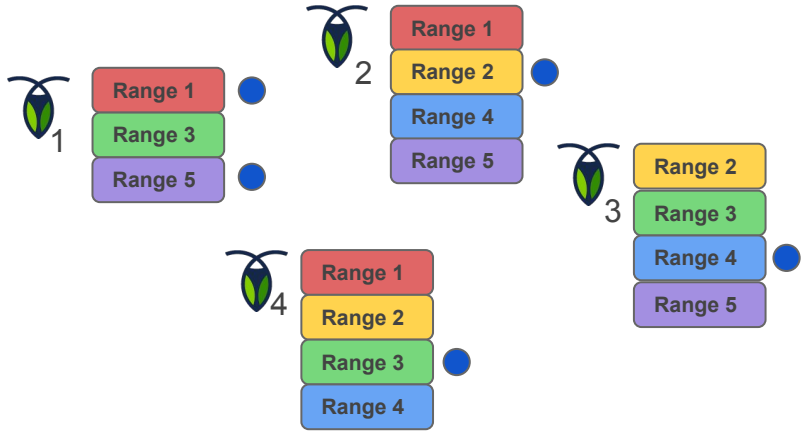
Allows to use different databases on each region, using multiple endpoints from different authorized clouds.



Focus on CockroachDB

Range: A set of sorted, contiguous data from your cluster.
Replicas: Copies of your ranges, which are stored on at least 3 nodes to ensure survivability.
Range Lease: For each range, one of the replicas holds the "range lease". This replica, referred to as the "leaseholder" ● , is the one that receives and coordinates all read and write requests for the range.
Quorum: Minimum number of nodes/replicas required to ensure that a transaction can be done





From https://www.cockroachlabs.com/blog/automated-rebalance-and-repair/

CockroachDB library

```
_XTRACE_PG=$(set +o | grep xtrace)
set +o xtrace

register_database cockroachdb

# Functions
# ---------

# Returns the name of the protocol for SQLAlchemy url connexion
function get_database_type_cockroachdb {
    echo cockroachdb
}

# Get rid of everything enough to cleanly change database backends
function cleanup_database_cockroachdb {
    # Handled by cockroachdb/tasks/destroy.yml
    return
}

function recreate_database_cockroachdb {
    local db=$1
    {% if inventory_hostname == dbmaster_node %}
    sudo docker exec -i cockroachdb-{{ inventory_hostname_short }} \
           ./cockroach sql --execute "DROP DATABASE IF EXISTS $db CASCADE" --insecure

    sudo docker exec -i cockroachdb-{{ inventory_hostname_short }} \
           ./cockroach sql --execute "CREATE DATABASE $db ENCODING = 'UTF8'" --insecure
    {% endif %}
}

function configure_database_cockroachdb {
    # Handled by cockroachdb/tasks/deploy.yml
    return
}

function install_database_cockroachdb {
    # Handled by cockroachdb/tasks/deploy.yml
    return
}

function install_database_python_cockroachdb {
    # Mostly handled by deploy.yml
    return
}

function database_connection_url_cockroachdb {
    local db=$1
    echo "$BASE_SQL_CONN:26257/$db?client_encoding=utf8"
}

# Restore xtrace
```

Timeline

- title + who are we: 1 -1 (Adrien+Marie + Ronan+)

- femdc: 3:30' - 4:30 (Adrien)

- start: 1’30 - 6:00 (Adrien)

- Storage backends: 7:30' - 13:30 (Marie)

- Juice: 4:30' - 18:00 (Marie)

- Evaluations: 12' - 30:00 (Ronan)

- conclusion: 2 : 32:00 (Adrien)