Building large Tarantool cluster with 100+ nodes

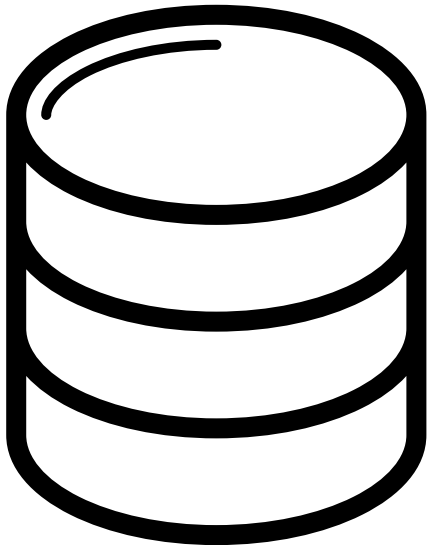
Yaroslav Dynnikov

Tarantool, Mail.Ru Group

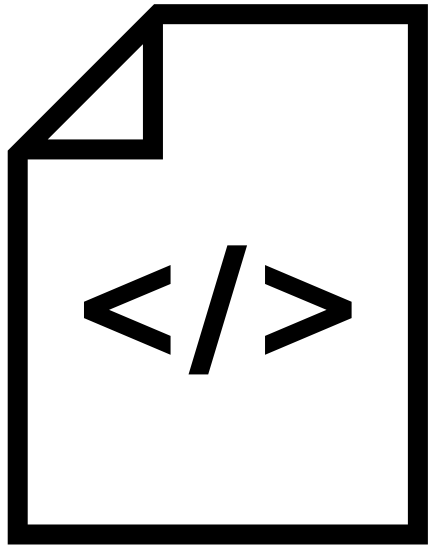
10 October 2019

Slides: rosik.github.io/2019-bigdatadays

Tarantool =



+



Database

(Transactions, WAL)

Application server (Lua)

(Business logics, HTTP)

Core team

- 20 C developers
- Product development

Solution team

- 35 Lua developers
- Commertial projects

Core team

- 20 C developers
- Product development

Solution team

- 35 Lua developers
- Commertial projects

Common goals

- Make development **fast** and **reliable**

In-memory no-SQL

- Not only in-memory: `vinyl` disk engine
- Supports SQL (since v.2)

In-memory no-SQL

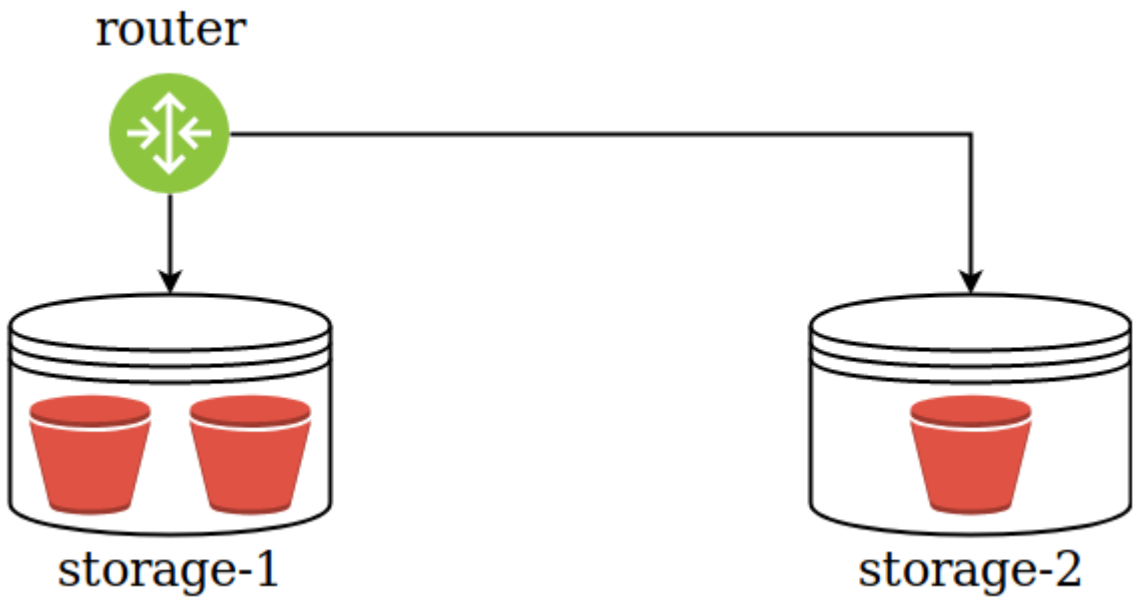
- Not only in-memory: `vinyl` disk engine
- Supports SQL (since v.2)

But

- We need **scaling** (horizontal)

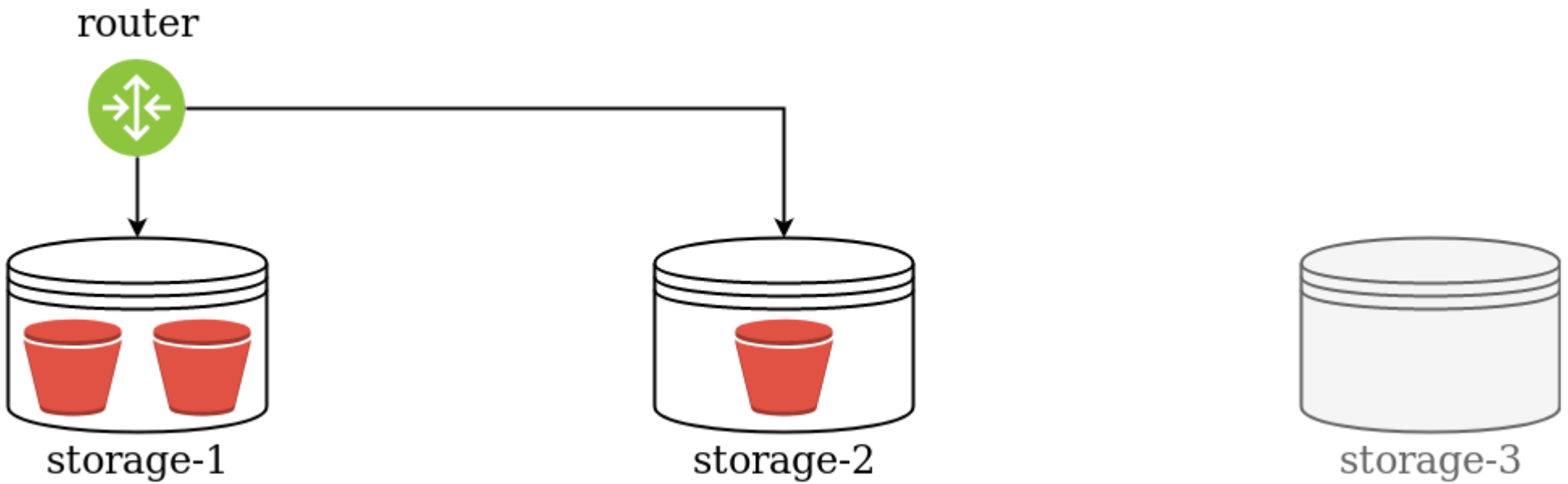
Vshard - horizontal scaling in tarantool

- Vshard assigns data to virtual buckets
- Buckets are distributed across servers



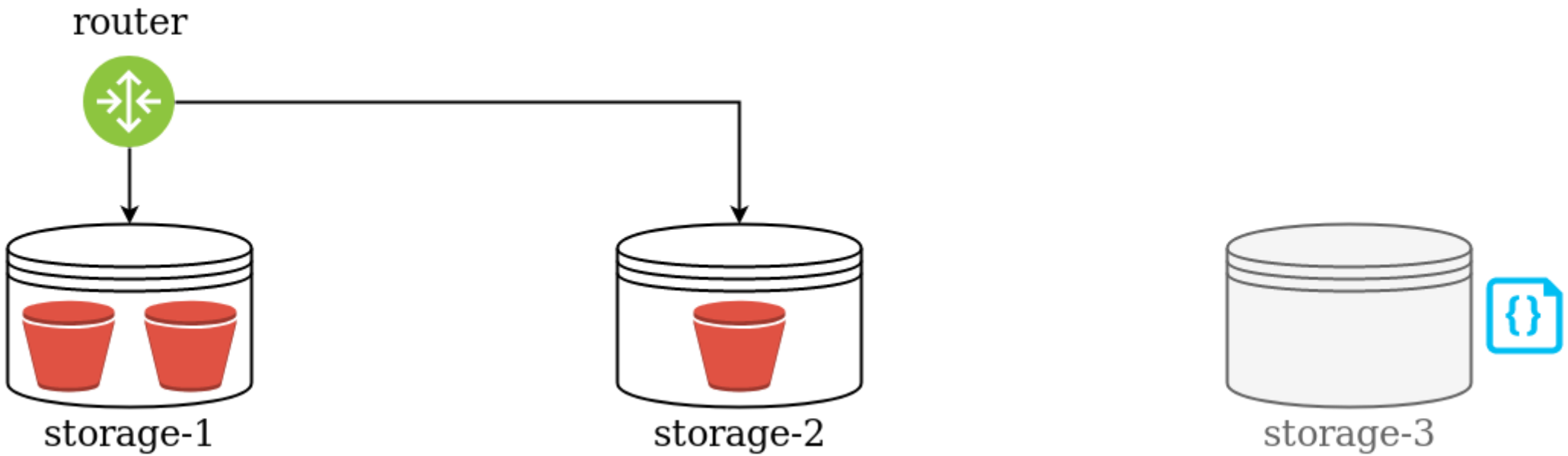
Vshard - horizontal scaling in tarantool

- Vshard assigns data to virtual buckets
- Buckets are distributed across servers



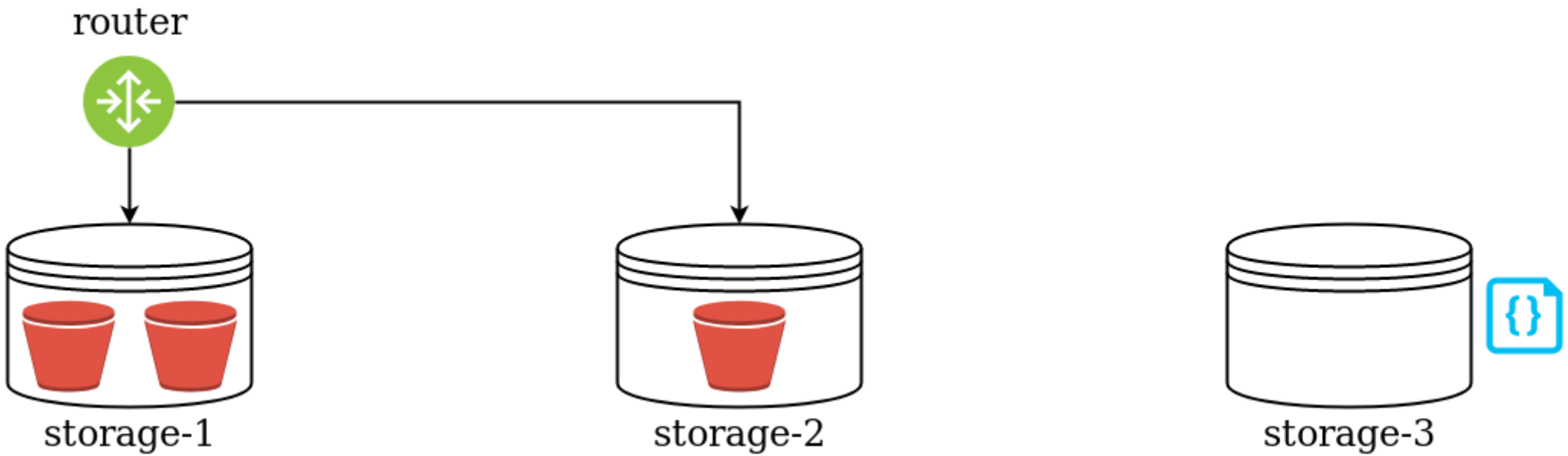
Vshard - horizontal scaling in tarantool

- Vshard assigns data to virtual buckets
- Buckets are distributed across servers



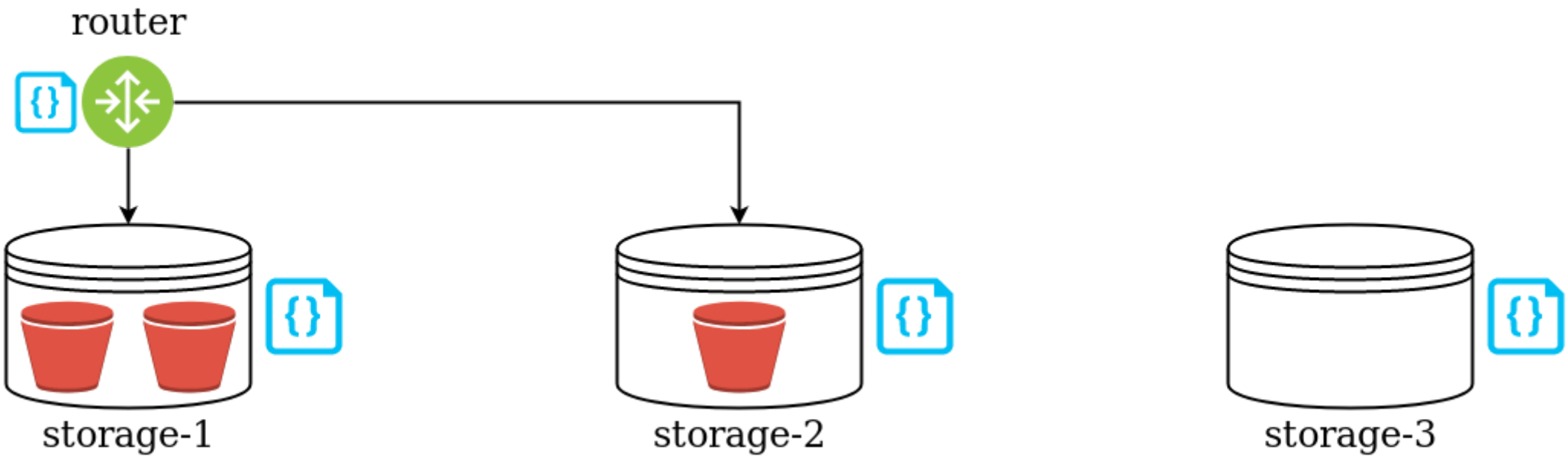
Vshard - horizontal scaling in tarantool

- Vshard assigns data to virtual buckets
- Buckets are distributed across servers



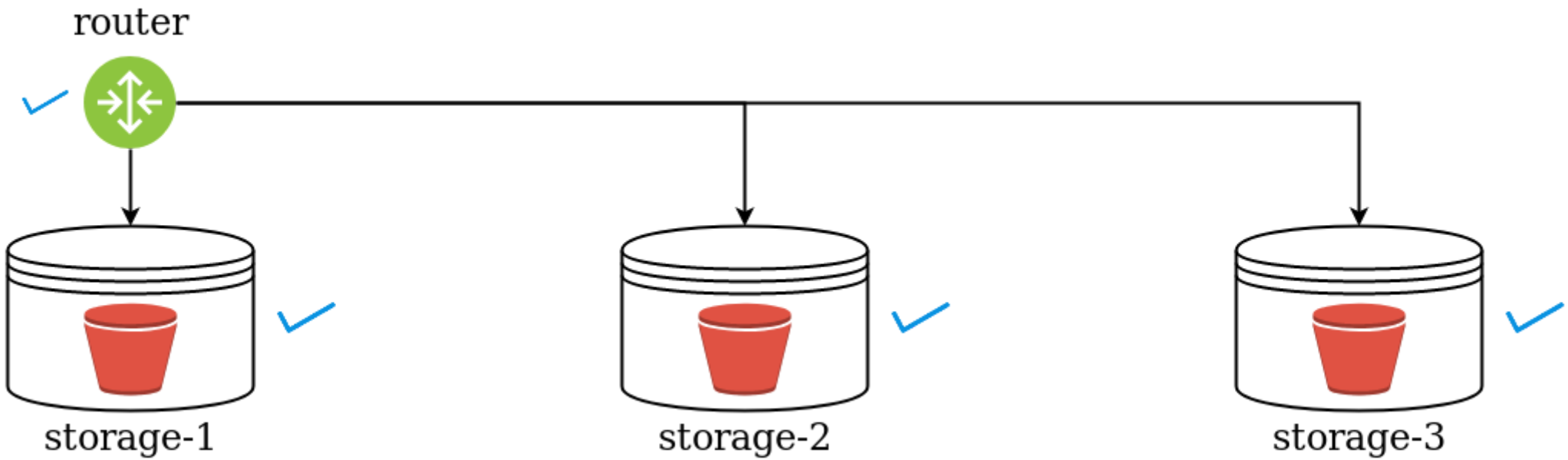
Vshard - horizontal scaling in tarantool

- Vshard assigns data to virtual buckets
- Buckets are distributed across servers



Vshard - horizontal scaling in tarantool

- Vshard assigns data to virtual buckets
- Buckets are distributed across servers



Vshard configuration

- Lua tables

```
sharding_cfg = {
    ['cbf06940-0790-498b-948d-042b62cf3d29'] = {
        replicas = { ... },
    },
    ['ac522f65-aa94-4134-9f64-51ee384f1a54'] = {
        replicas = { ... },
    },
}
```

```
vshard.router.cfg(...)
vshard.storage.cfg(...)
```

Vshard automation. Options

- Deployment scripts
- Docker compose
- Zookeeper

Vshard automation. Options

- Deployment scripts
- Docker compose
- Zookeeper

- Our own orchestrator

Orchestrator requirements

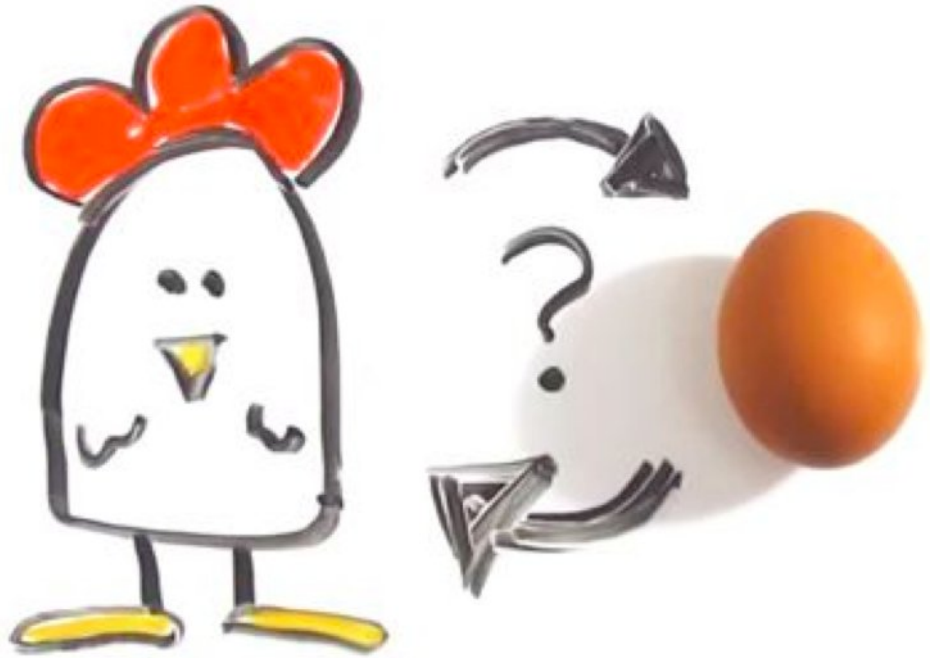
- Doesn't start/stop instances (systemd can do that)
- Applies configuration only
- Every cluster node can manage others
- Built-in monitoring

Clusterwide configuration

- **Must** be the same everywhere
- Applied with two-phase commit

```
topology:
  servers:                # two servers
    s1:
      replicaset_uuid: A
      uri: localhost:3301
    s2:
      replicaset_uuid: A
      uri: localhost:3302
  replicasets:            # one replicaset
    A:
      roles: ...
```

Which came first?



Database
`tcp_listen()`

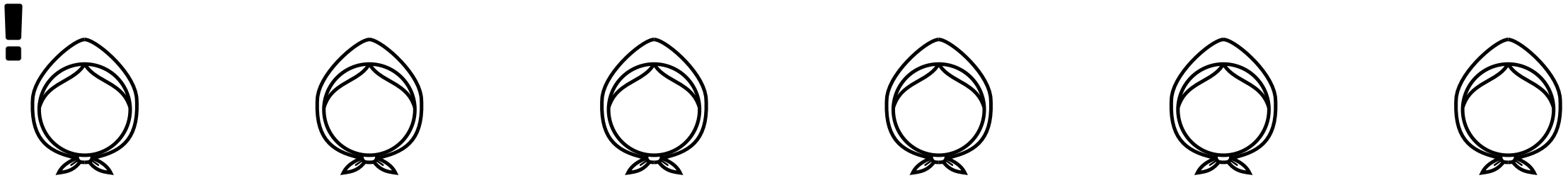
Orchestrator
`apply_2pc()`

Membership implementation

- SWIM protocol **-** one of the **gossips** protocols family

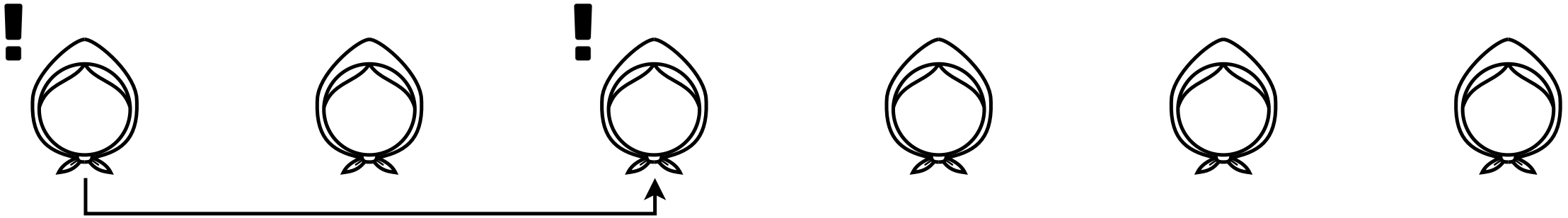
Membership implementation

- SWIM protocol **-** one of the **gossips** protocols family



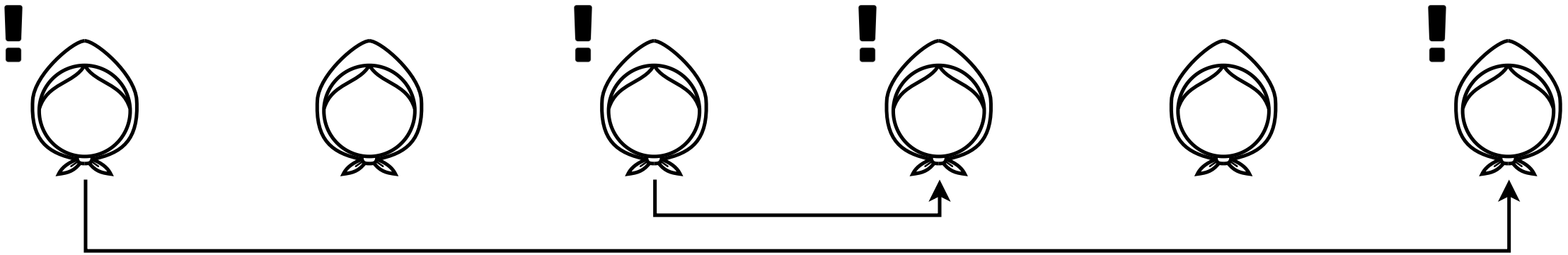
Membership implementation

- SWIM protocol **-** one of the **gossips** protocols family



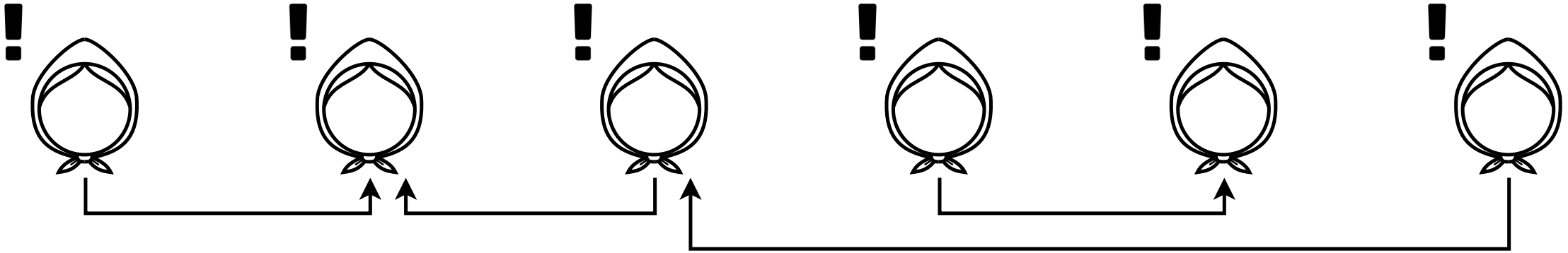
Membership implementation

- SWIM protocol **-** one of the **gossips** protocols family



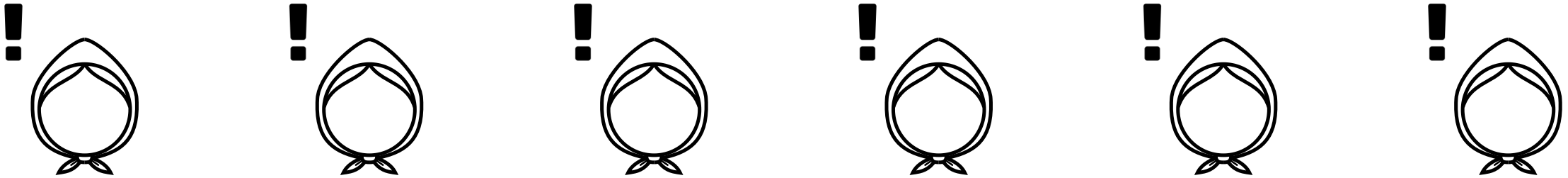
Membership implementation

- SWIM protocol **-** one of the **gossips** protocols family



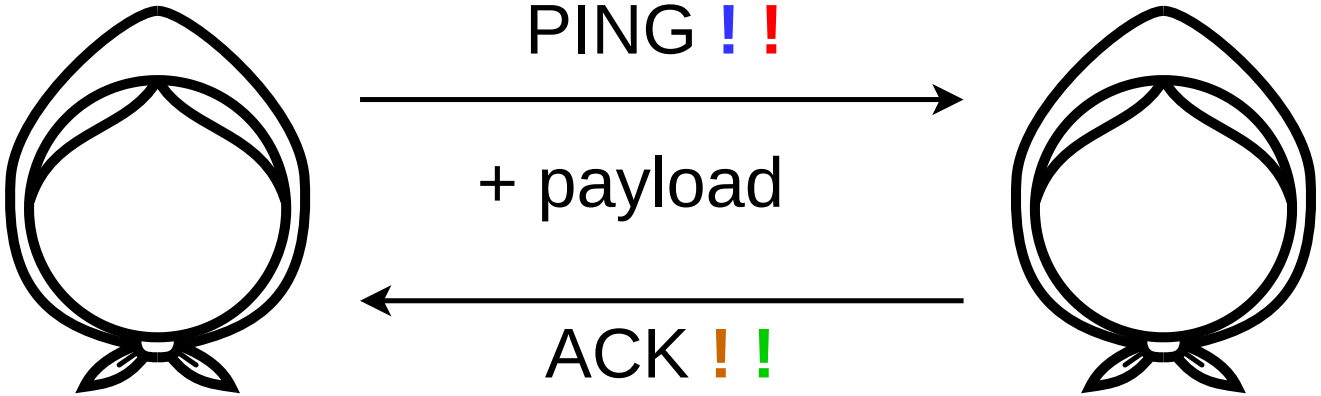
Membership implementation

- SWIM protocol - one of the **gossips** protocols family
- Dissemination speed: O(logN)
- Network load: O(N)



Membership implementation

- SWIM protocol - one of the **gossips** protocols family
- Dissemination speed: O(logN)
- Network load: O(N)



Bootsrapping new instance

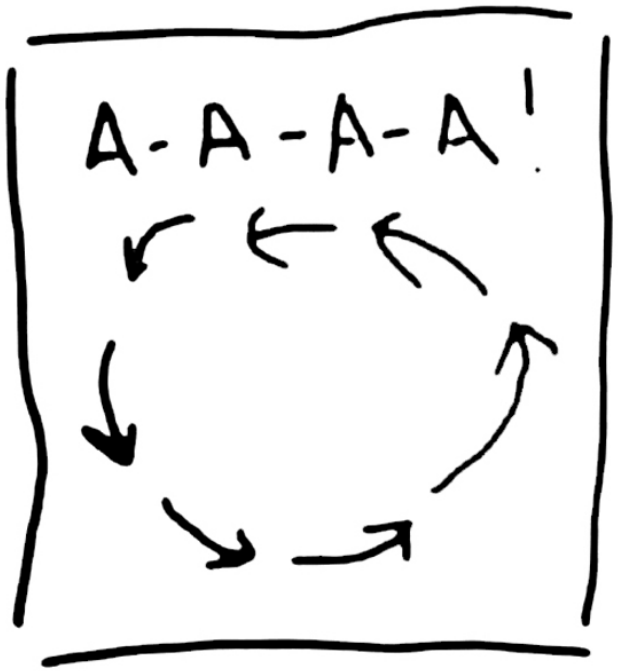
1. New process starts
2. New process joins membership
3. Cluster checks new process is alive
4. Cluster applies configuration
5. New process polls it from membership
6. New process bootstraps

7. Repeat N times

Bootsrapping new instance

1. New process starts
2. New process joins membership
3. Cluster checks new process is alive
4. Cluster applies configuration
5. New process polls it from membership
6. New process bootstraps

7. Repeat N times



N

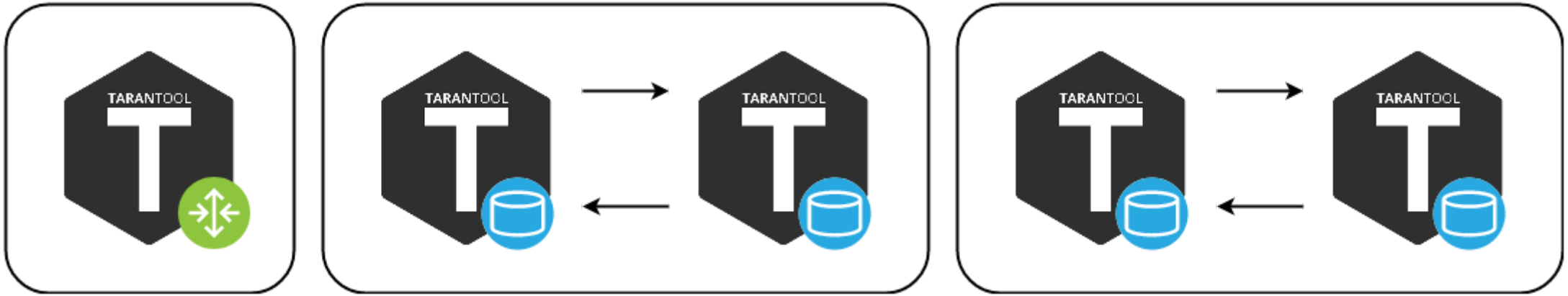
= 100

Benefits so far

1. Orchestration works

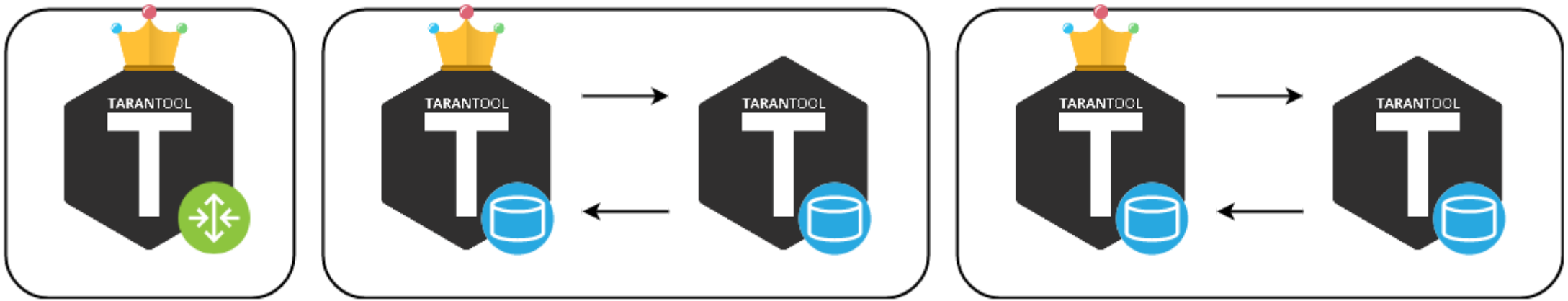
Benefits so far

1. Orchestration works



Benefits so far

1. Orchestration works



Benefits so far

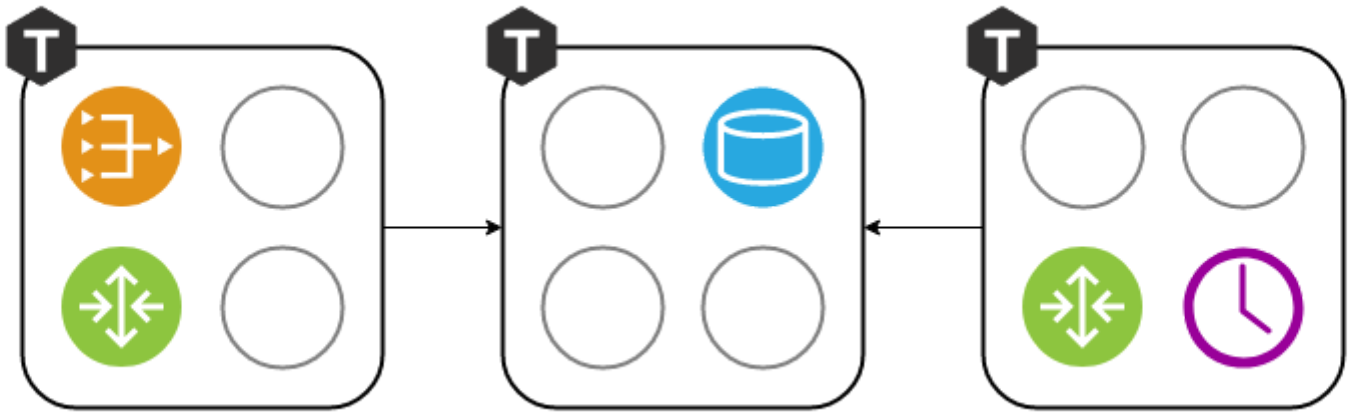
1. Orchestration works
2. Monitoring works

Benefits so far

1. Orchestration works
2. Monitoring works

Benefits so far

1. Orchestration works
2. Monitoring works
3. We can assign any role to the instance



Benefits so far

1. Orchestration works
2. Monitoring works
3. We can assign any role to the instance

Role management

- `function init()`

Role management

- `function init()`
- `function validate_config()`
- `function apply_config()`

Role management

- `function init()`
- `function validate_config()`
- `function apply_config()`
- `function stop()`

Refactoring the bootstrap process

- Assembling large clusters with 100+ instances is slow
- N two-phase commits are slow
- N config pollings is slow

Refactoring the bootstrap process

- Assembling large clusters with 100+ instances is slow
- N two-phase commits are slow
- N config pollings is slow

Solution

- Bootstrap all instances with a single 2pc
- Re-implement binary protocol and reuse port

Links

- tarantool.io
- github.com/tarantool/tarantool
- Telegram - @tarantool, @tarantool_news

- Cartridge framework - github.com/tarantool/cartridge
- Cartridge CLI - github.com/tarantool/cartridge-cli
- Posts on Habr - habr.com/users/rosik/
- This presentation - rosik.github.io/2019-bigdatadays

Questions?