Google ganeti

Ganeti

Ganeti Core Team - Google
LISA '13 - 5 Nov 2013

Ganeti Design

A cluster virtualization manager, internals

- Guido Trotter <ultrotter@google.com>
- Helga Velroyen <helgav@google.com>

Latest version of these slides

Please find the latest version of these slides at:

https://code.google.com/p/ganeti/wiki/LISA2013

Node roles

Management Level

- Master Node
    - runs ganeti-masterd, rapi, noded, confd and mond
- Master candidates
    - have a full copy of the config, can become master
    - run ganeti-confd, noded and mond
- Master capable
    - can be upgraded to master candidates, if needed
- Regular nodes
    - cannot become master
    - get only part of the config
    - run noded and mond
- Offline nodes, are in repair

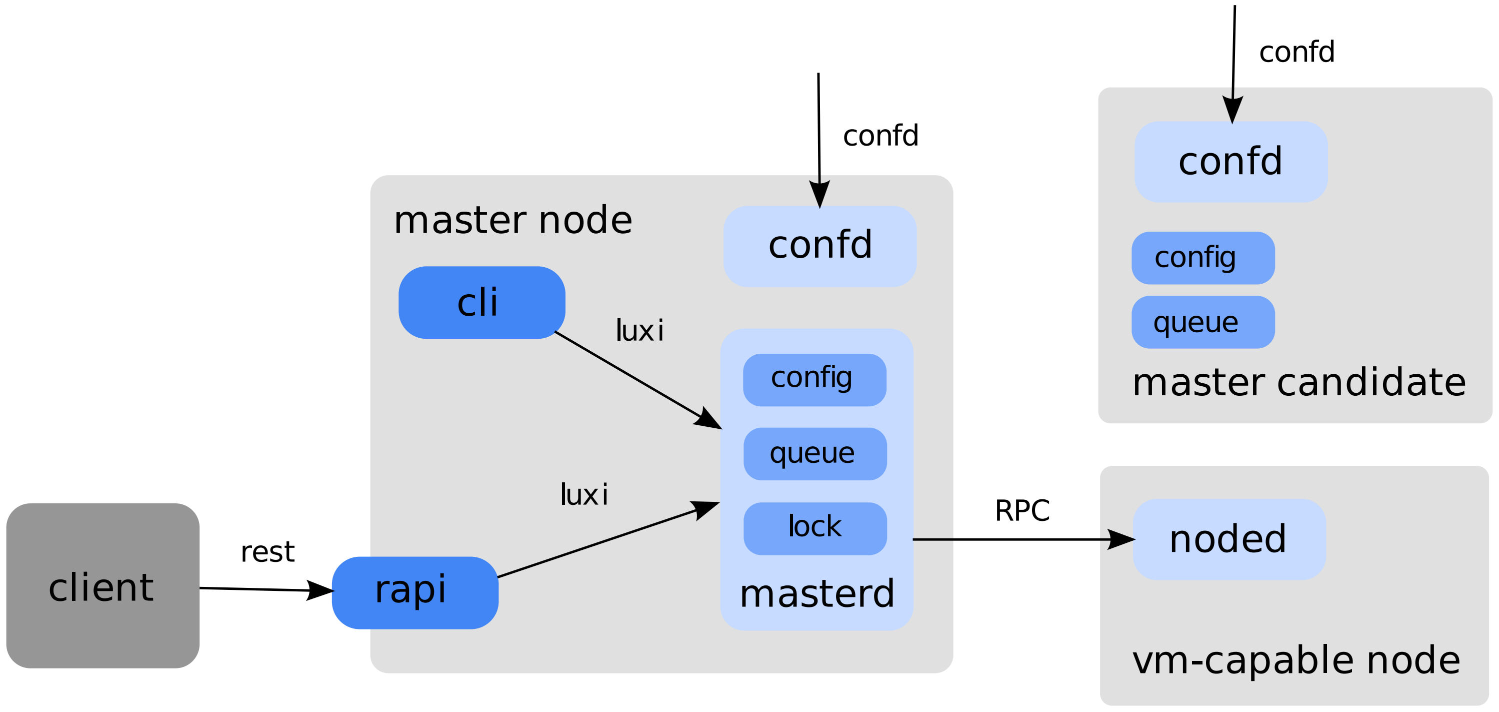
Node roles

Instance hosting level

- VM capable nodes
    - can run virtual machines
- Drained nodes
    - are being evacuated
- Offlined nodes, are in repair

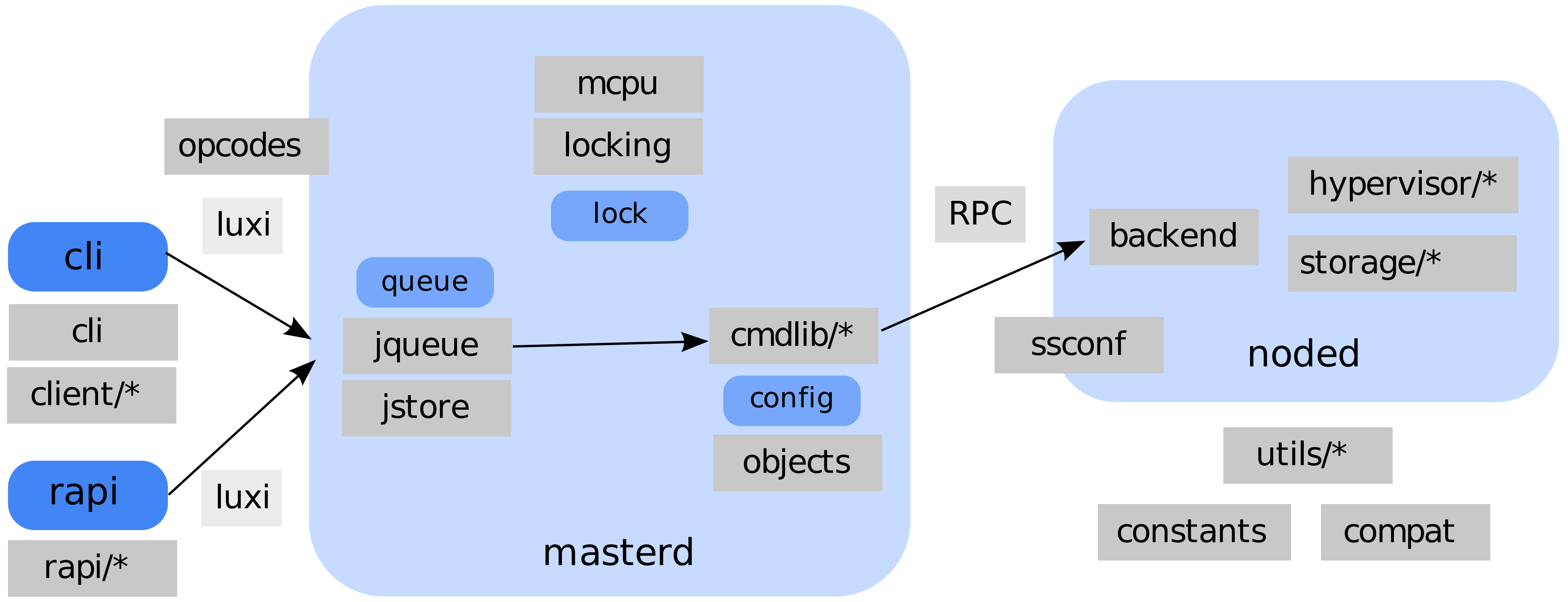
Ganeti Components

Main Ganeti components, and how they communicate



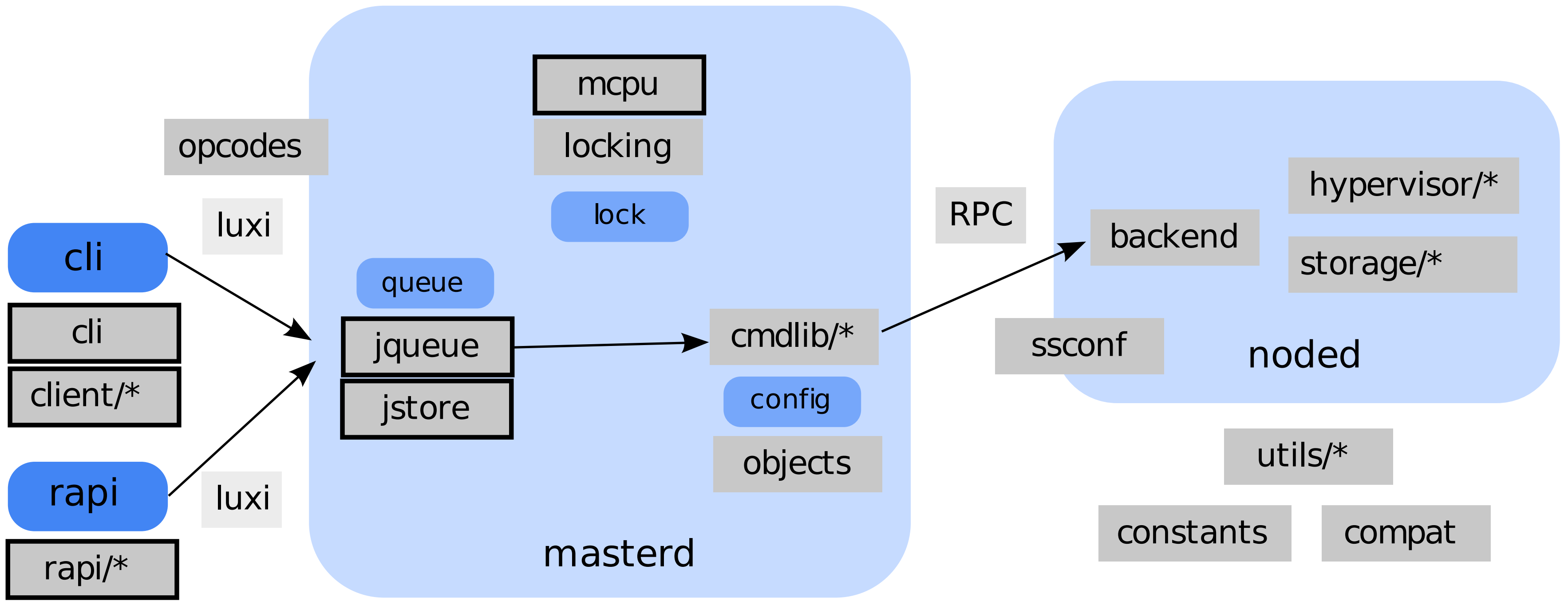
Ganeti Core Structure

Core source structure:



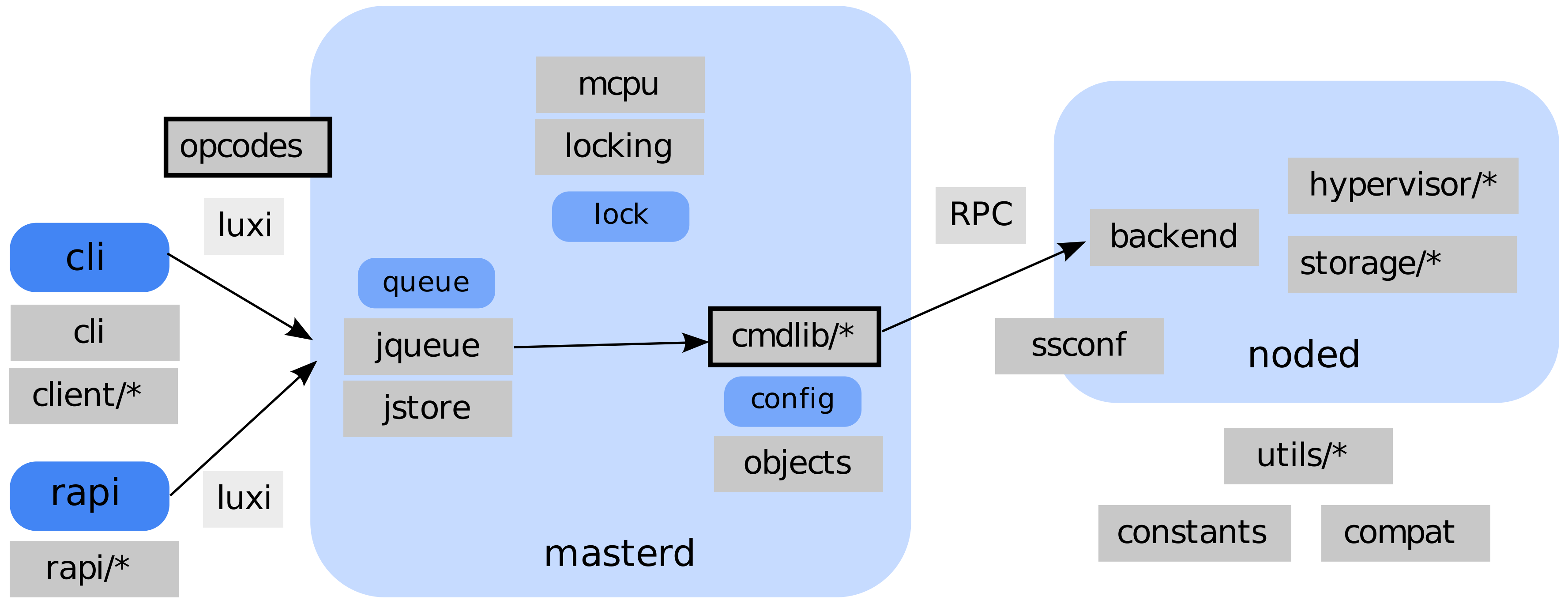
Jobs

- List of opcodes, executed in sequence
- Submitted by the cli, or via rapi, or by other jobs



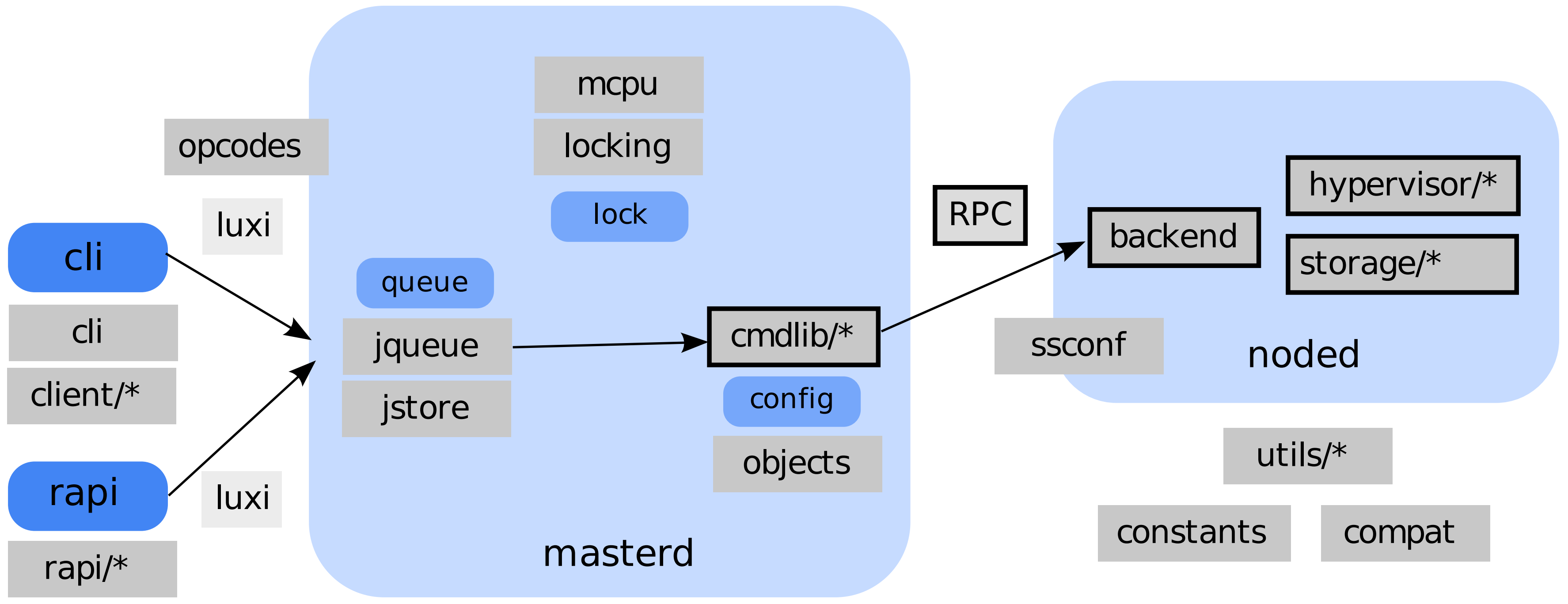
Opcodes

- Cluster business logic
- Implemented in cmdlib



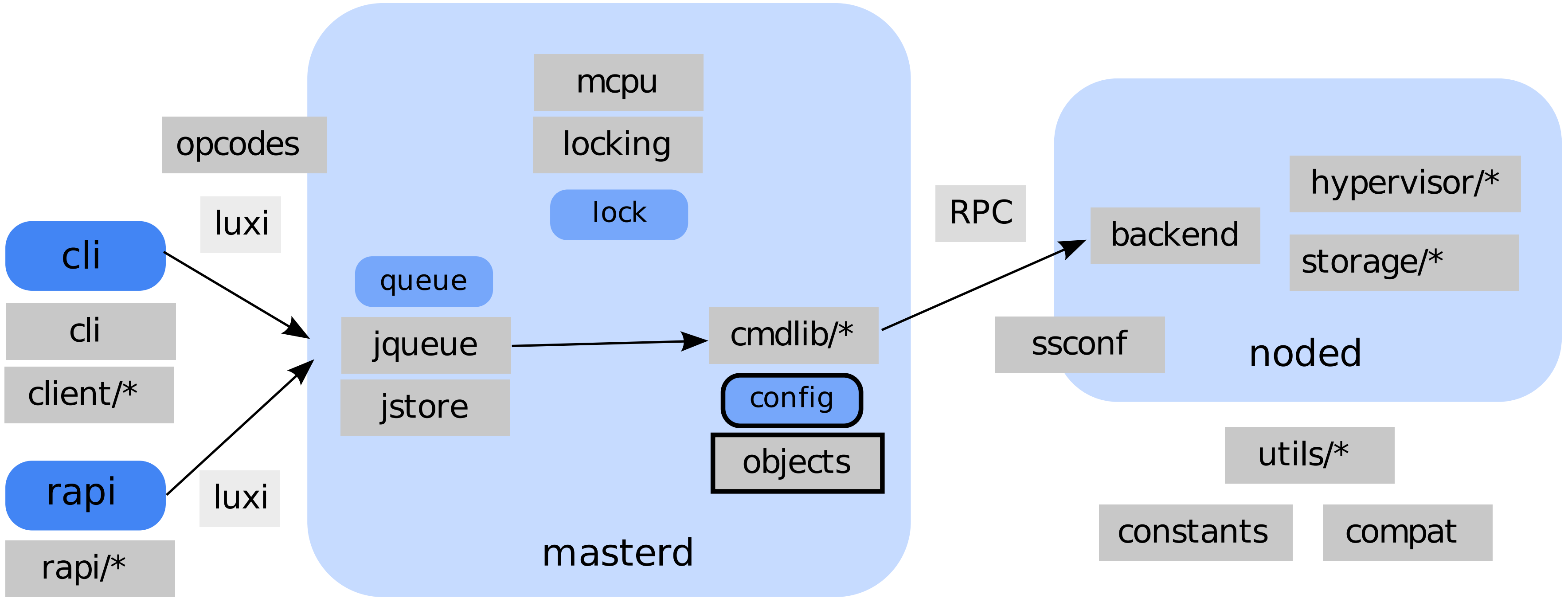
RPCs

- Per-node business logic
- Implemented in backend (using bdev, hypervisor, runcmd)



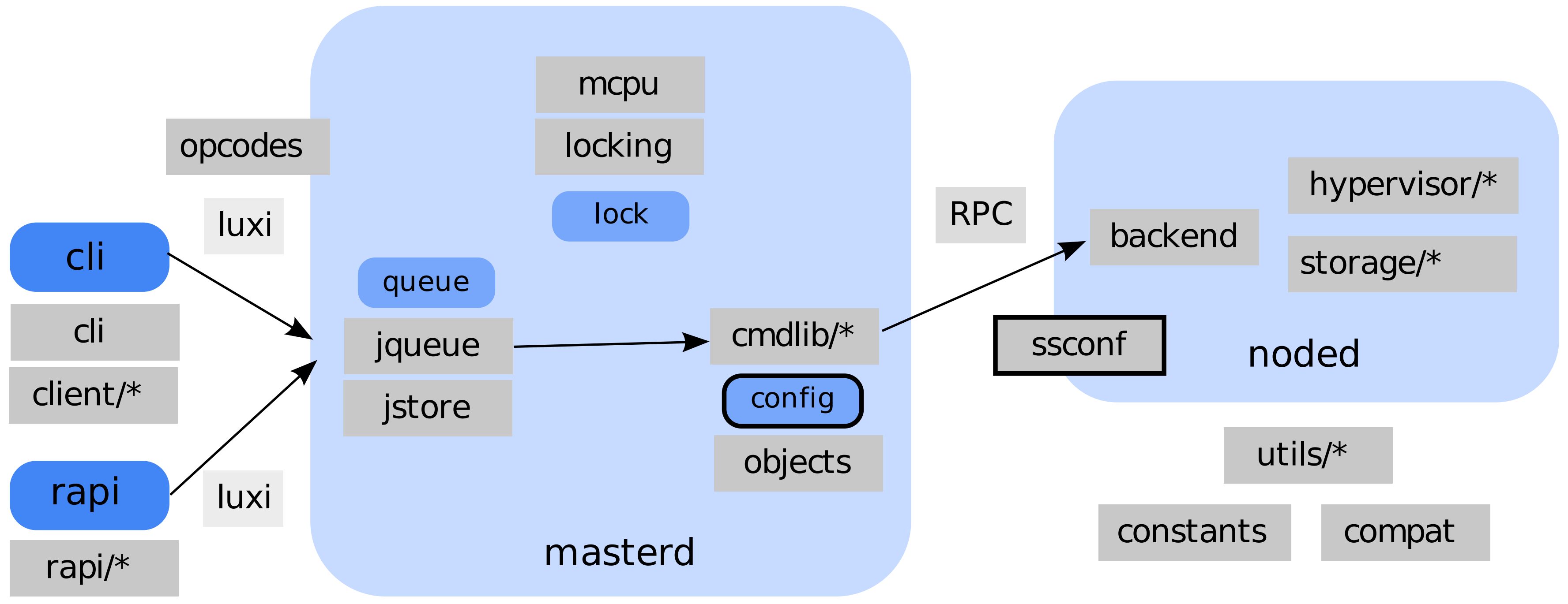
Config

- Tree of "objects" with all the cluster entities
- Replicated to all master candidates



ssconf

- Flat-file export of parts of the config
- Available on all nodes



Customizing Ganeti

Most common customizations:

- Altering hypervisor behavior
- Adding an hypervisor parameter
- Altering cluster business logic
- Adding an option to the cluster business logic
- Adding a backend storage
- Adding a new hypervisor
- Adding a new data collector

Altering hypervisor behavior

Difficulty: simple

- Edit the logic in your hypervisor's file
- For example add a command line flag to kvm, or a config value for xen

Adding an hypervisor parameter

Difficulty: simple

- Add the parameter in `constants.py` (*)
    - eg: `HV_KVM_SPICE_USE_VDAGENT`
- Edit the logic in your hypervisor's file
    - eg: migration bandwidth and downtime control: commit e43d4f9f

(*) soon `constants.hs`

Altering cluster business logic

Difficulty: medium

- Change the logic in `cmdlib.py`
- Be careful w.r.t. locking (do you need more? less?)
- Add any rpc to `backend.py`, `rpc_defs.py`, `server/noded.py`
- If the hypervisor interface changes, update all hypervisors

Adding opcode level options

Difficulty: medium

- Add the option field to `opcodes.py`, use the right type (see `ht.py`)
- Use the option in `cmdlib` (see "altering cluster business logic")
- Add the command line flag to `cli.py` and the right utility in `client/*`

Adding a backend storage

Difficulty: hard

- Implement the `BlockDev` interface in `storage/bdev.py`
- Add the logic in `cmdlib` (eg. migration, verify)
- Add the new storage type name to constants
- Add any parameter the new storage needs to constants
- Modify `objects.Disk` to suppport your storage type
    - eg: adding support for RBD: commit 7181fba

Adding a new hypervisor

Difficulty: medium

- "just" implement the hypervisor API (easy)
- Add the hypervisor name and parameters to `contants.py`
- Alter `cmdlib` as needed for supporting it
- Alter the hypervisor API as needed for supporting it

Adding a new data collector

Difficulty: medium/hard

- Implement the haskell function to read the data you need
  - Be careful about the availability of new dependencies
  - Please submit a design if you'd like the collector to be included
  - Mind latency: data collectors should be fast
- Add the collector exporting the data (eg. 2da679f)

Thank You!

Questions?

Survey at https://www.usenix.org/lisa13/training/survey