

Ganeti design



A cluster virtualization manager, internals.

© 2010-2011 Google

Use under GPLv2+ or CC-by-SA

Some images borrowed/modified from Lance Albertson and Justin Pop

Node roles (management level)

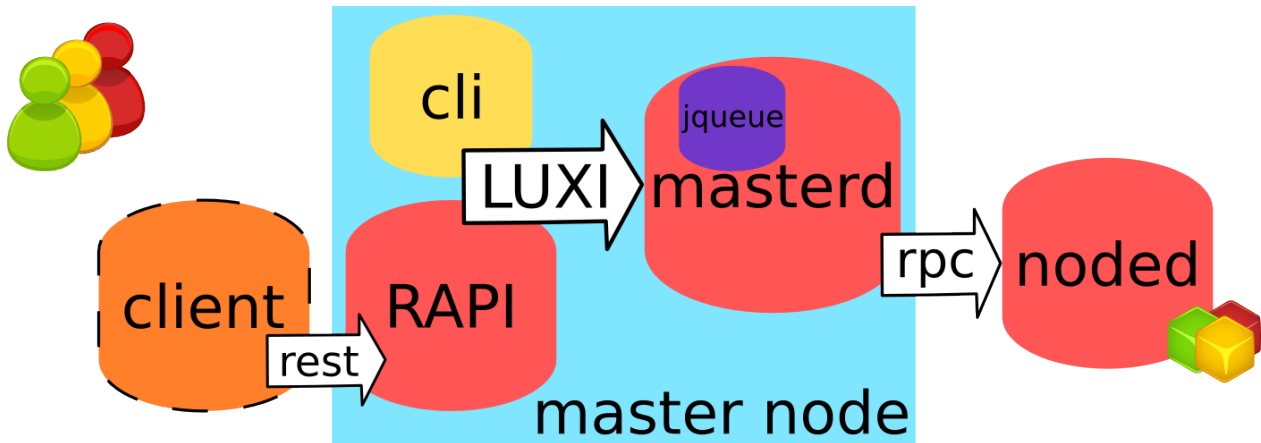
- Master Node
 - runs ganeti-masterd, rapi, noded and confd
- Master candidates
 - have a full copy of the config, can become master
 - run ganeti-confd and noded
- Master capable
 - can be upgraded to master candidates, if needed
- Regular nodes
 - cannot become master
 - get only part of the config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

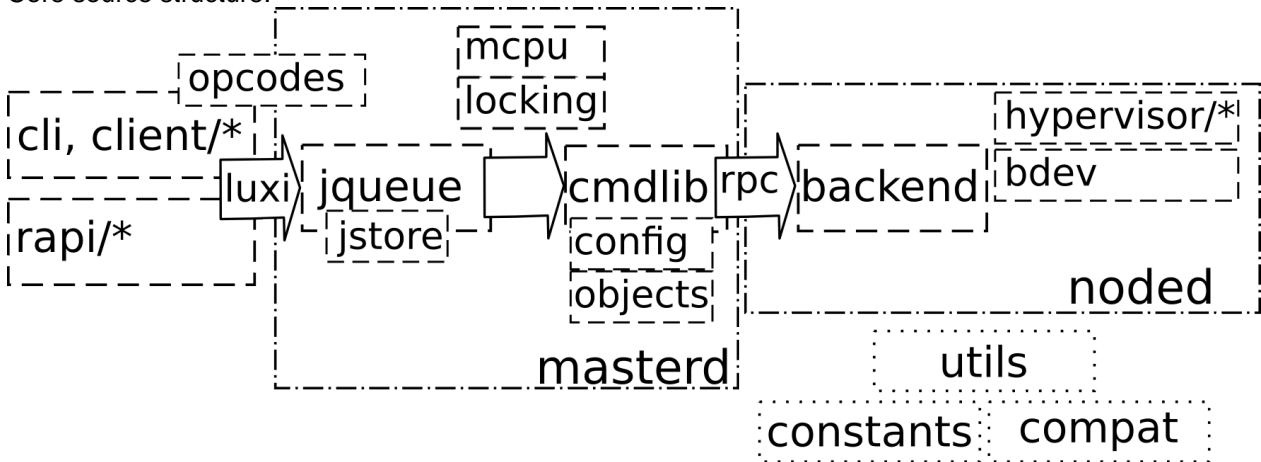
Ganeti Components

Main Ganeti components, and how they communicate:



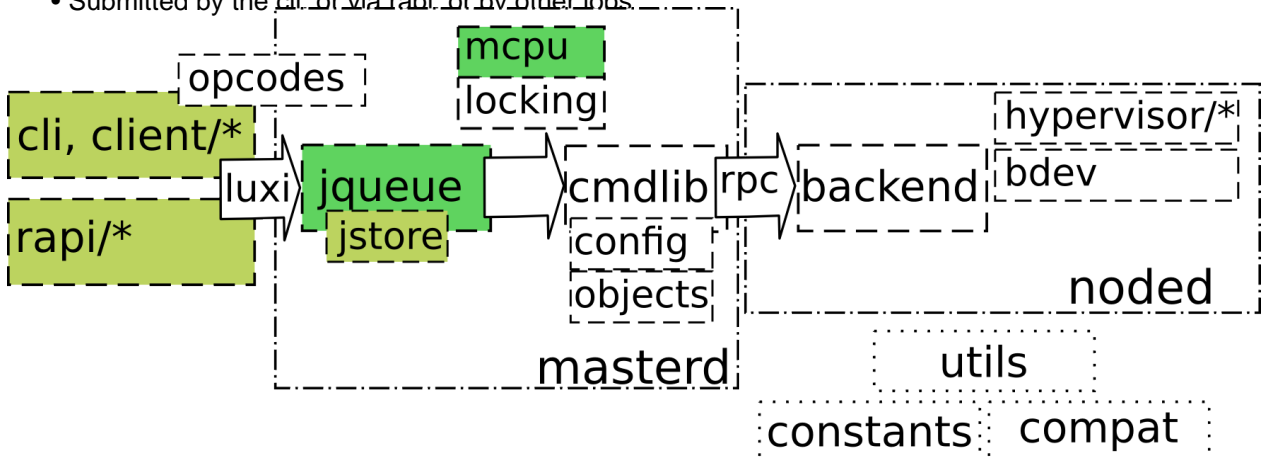
Ganeti Core Structure

Core source structure:



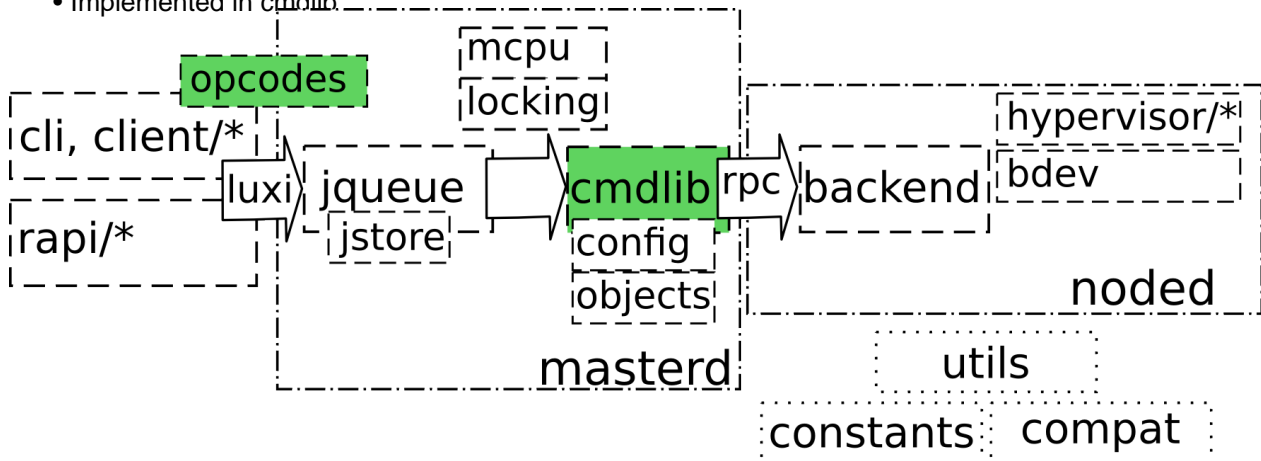
Jobs

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



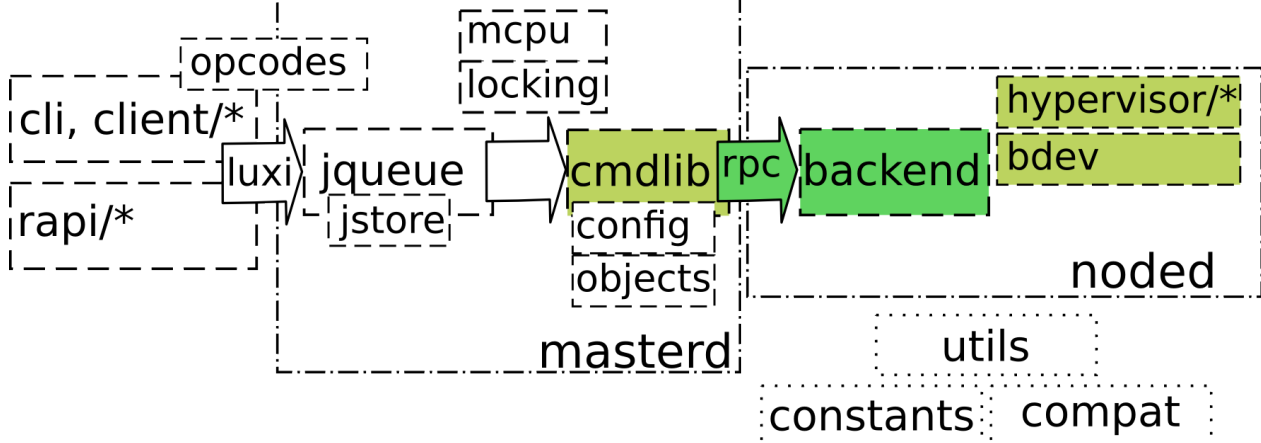
Opcodes

- Cluster business logic
- Implemented in `cmdlib`



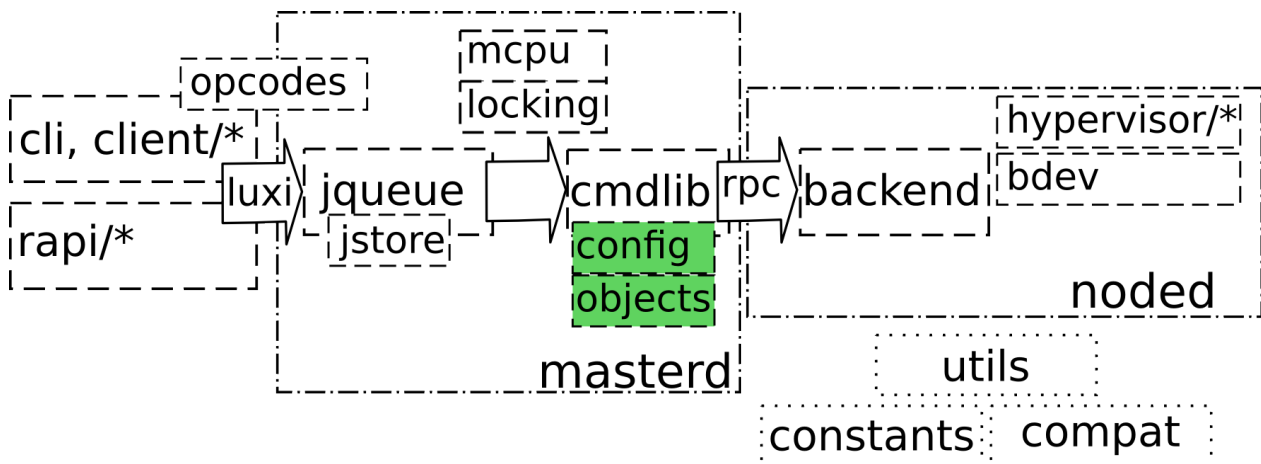
RPCs

- Per-node business logic
- Implemented in `backend` (`using bdev_hypervisor_runcmd`)



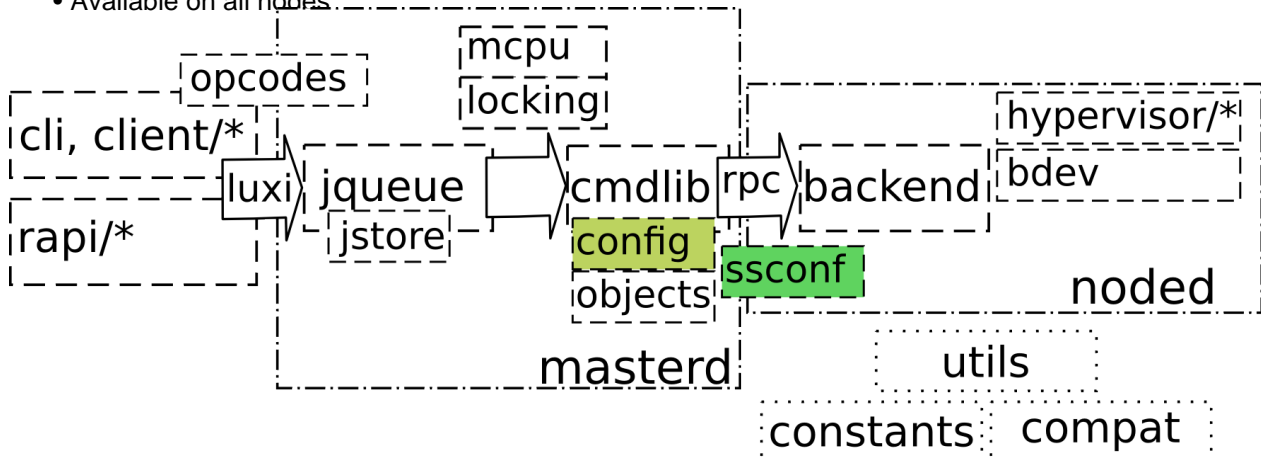
Config

- 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

Altering hypervisor behavior (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 (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

Altering cluster business logic (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 (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 (hard)

- Implement the BlockDev interface in 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 support your storage type
- eg: adding support for RBD: commit 7181fba

Adding a new hypervisor (medium)

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

Conclusion

Questions?

© 2010-2011 Google

Use under GPLv2+ or CC-by-SA

Some images borrowed/modified (with permission) from Lance Albertson

