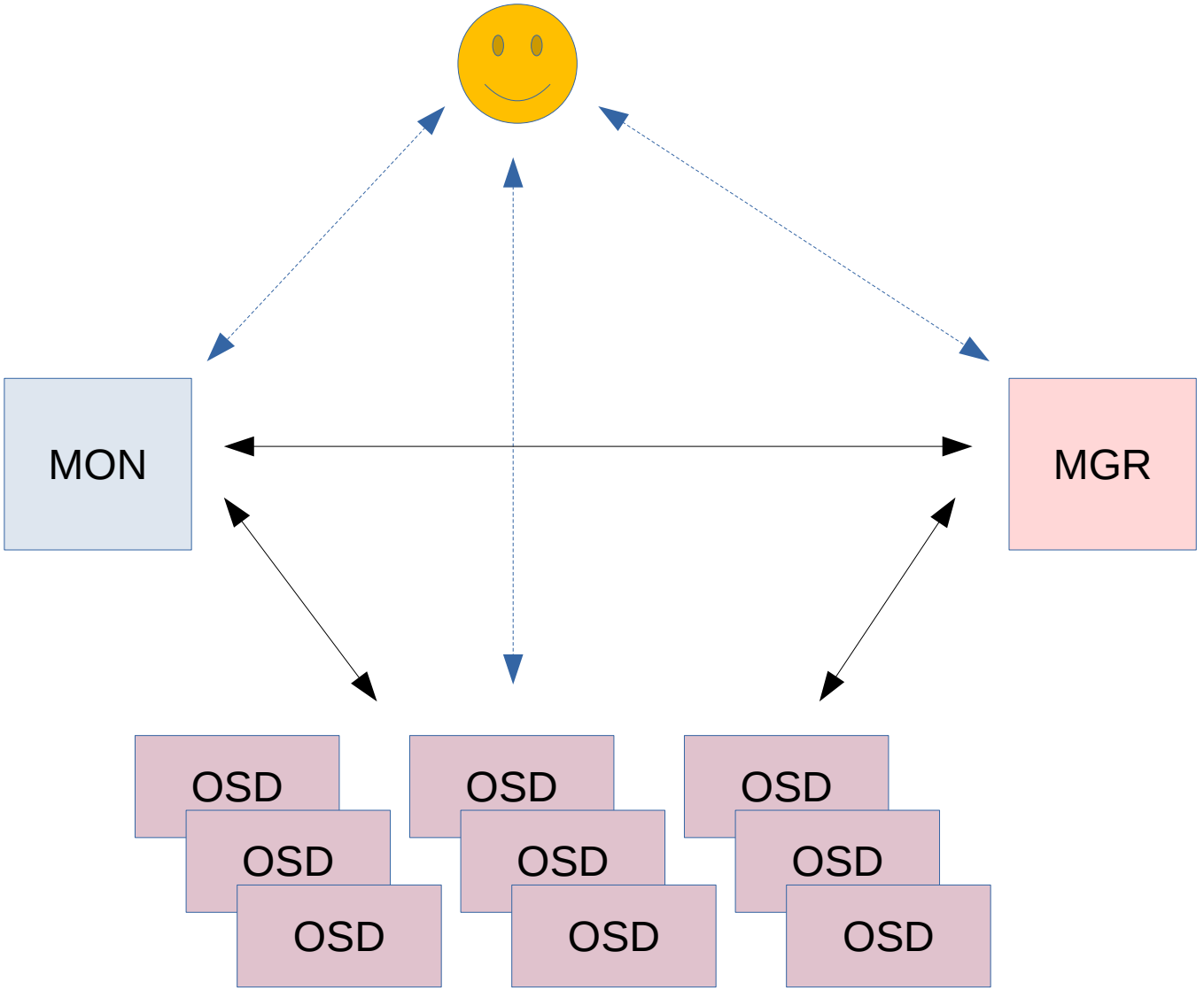
Ceph Manager Modules for Fun and Profit

Joao Eduardo Luis <joao@suse.com>

Ceph Manager (ceph-mgr)

- Monitoring and Management
  - Dashboard, orchestrator, balancer
- Interface with external tools
  - Prometheus exporter, disk prediction failures
- Aggregate a bunch of data everyone's used to
  - PG statistics, state, health information, services

Ceph Manager (ceph-mgr)



Architecture

- Leverages CPython to allow Python code to be run
- Each module is its own entity (sub interpreter)
- Modules get real-time access to
  - Maps
  - Health Info
  - Juicy cluster statistics

How-To, kind of quickly...

- MgrModule
  - Where all the hard work is done
  - Calls out certain functions on the module for reasons

How-To, kind of quickly...

- OurModule extends MgrModule, and implements
  - handle_command()
  - shutdown()
  - notify()
  - serve()

Handling commands

- *ceph ourmodule do-this arg1 arg2 -i file.txt*

```
1  class OurModule(MgrModule):
2
3      COMMANDS = [
4          {
5              "cmd": "ourmodule do-this "
6                     "name=arg1,type=CephString "
7                     "name=arg2,type=CephString",
8              "desc": "Our Module does this",
9              "perm": "w"
10         }
11     ]
12
13
14     def handle_command(self, inbuf, cmd):
15         if cmd['prefix'] == "ourmodule do-this":
16             self.handle_this(cmd['arg1'], cmd['arg2'], inbuf)
17         return (0, "yay", "")
18
```

Handling commands

- *ceph ourmodule do-this arg1 arg2 -i file.txt*

```
1  class OurModule(MgrModule):
2
3      @CLIWriteCommand("ourmodule do-this",
4                       "name=arg1,type=CephString"
5                       "name=arg2,type=CephString",
6                       "Our Module does this")
7      def handle_this(self, arg1, arg2, inbuf):
8          # do this
9          return (0, "yay", "")
10
```

What did we do with this?

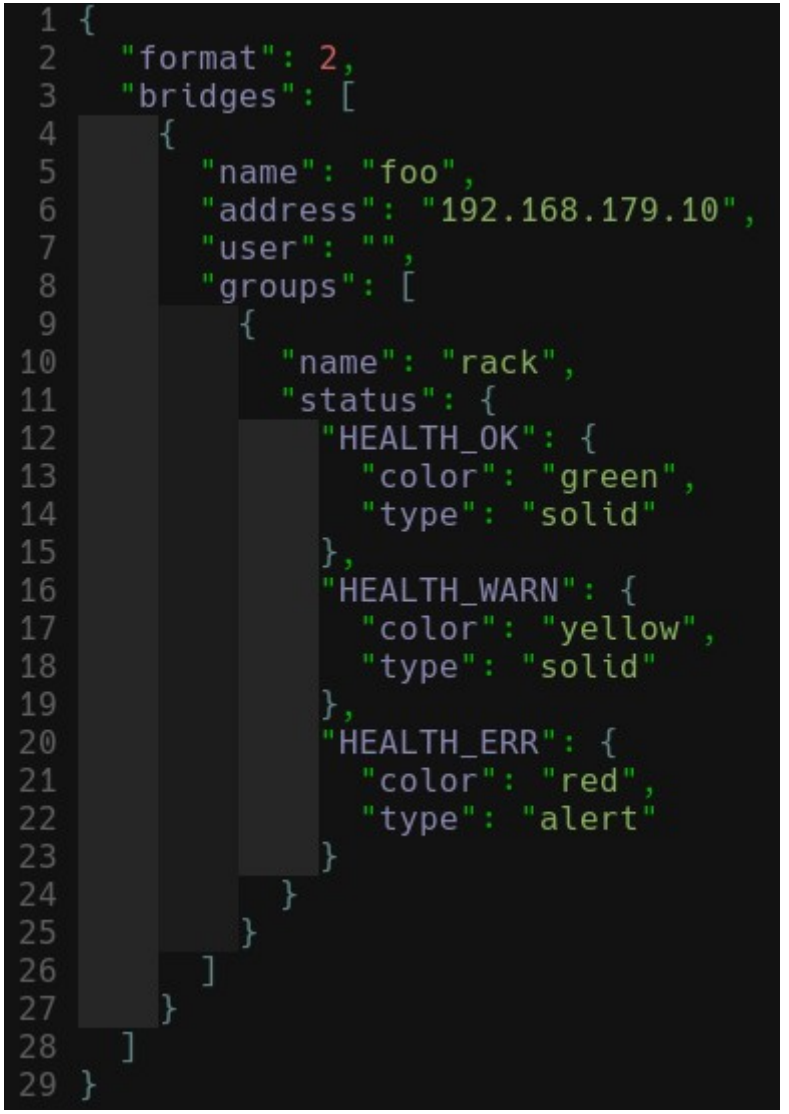
BLINKING LIGHTS

(Philips) Hue module

- Philips Hue bridge
- Some Hue bulbs, LEDs
- A (vstart) Ceph cluster
- A **very** patient flatmate

Basic config

- Basic config to specify bridges
  - Address, user, groups and states
- Each group can react to different status updates



```
{
  "format": 2,
  "bridges": [
    {
      "name": "foo",
      "address": "192.168.179.10",
      "user": "",
      "groups": [
        {
          "name": "rack",
          "status": {
            "HEALTH_OK": {
              "color": "green",
              "type": "solid"
            },
            "HEALTH_WARN": {
              "color": "yellow",
              "type": "solid"
            },
            "HEALTH_ERR": {
              "color": "red",
              "type": "alert"
            }
          }
        }
      ]
    }
  ]
}
```

And then...

- We setup the bridge on **serve()**
- We listen for health updates using **notify()**
- We turn off the lights on **shutdown()**

Insert Demo Here (hopefully)

Where to go from here

- Pulsing lights while PGs are recovering
- Making it crushmap aware, or
- Grab the inventory from the orchestrator
  - And then blink lights for specific servers or racks

Q&A