What's new in Ganeti?

Technical details of changes since GanetiCon 2014

Klaus Aehlig <aehlig@google.com>
Lisa Velden <velden@google.com>

September 15, 2015

Forthcoming instances

reserve now, create later

Forthcoming instances

- New type of instances: forthcoming (`forthcoming` field in the config, default `false`)
- Those instances only exist in the configuration
  - however, resources are fully accounted for
  - can be moved and renamed just as real ones
  - are also balanced by `htools`

Instance reservations—use case

- Only want to create instances once DNS is set up

Instance reservations—use case

- Only want to create instances once DNS is set up

⇝ Choose cluster, then IP accordingly, propagate DNS
. . . and only then create the instance

Instance reservations—use case

- Only want to create instances once DNS is set up

$\rightsquigarrow$ Choose cluster, then IP accordingly, propagate DNS
. . . and only then create the instance

$\Rightarrow$ During DNS propagation,
the new resources are not accounted for

Instance reservations—use case

- Only want to create instances once DNS is set up

$\rightsquigarrow$ Choose cluster, then IP accordingly, propagate DNS
. . . and only then create the instance

$\Rightarrow$ During DNS propagation,
the new resources are not accounted for

- Now if DNS propagation is slow
and lots of instances are requested. . .

Instance reservations—non use case

- speed up instance creation by first reserving
  locking-wise no difference
  - reservation takes the same locks as adding a real instance
  - creation will hold the same locks as adding a real instance afer node choice

  Remember: NAL is gone anyway

Using instance reservations

```
gnt-instance add --forthcoming --no-name-check
    ... tmp123.example.com

gnt-instance rename tmp123.example.com
    finalname.example.com

gnt-instance add --commit ... finalname.example.com
```

OS Installations

public, private, and secret parameters

OS Parameters

	Ganeti Config	Job File		Log Files
		queued	running	
public	✓	✓	✓	✓
private	✓	✓	×	×
secret	×	×	×	×

Secret Parameters - Previous State

- do not appear in log files
- do not appear in job files for running jobs

Secret Parameters - Previous State

- do not appear in log files
- do not appear in job files for running jobs
- written into job files for queued jobs

Secret Parameters - Current State

- keep secret parameters only in memory
- transmit them in the last step when a job process is forked off
- re-inject them into the job description of the forked process

Secret Parameters - Current State

How to prevent secret parameters from appearing in job files?

Secret Parameters - Current State

How to prevent secret parameters from appearing in job files?

- value is shown as `<redacted>`
- new type **Secret** (similar to `Private`):
  - wrap secret value
  - different `showJSON` method:
    prints `<redacted>` instead of value
  - changed to `Private` before transmission to forked job process

Secret Parameters - Current State

What happens if we re-try a job with secret parameters?

Secret Parameters - Current State

What happens if we re-try a job with secret parameters?

- we do not want the value <`redacted`> to appear in the instance

Secret Parameters - Current State

What happens if we re-try a job with secret parameters?

- we do not want the value <`redacted`> to appear in the instance
- jobs fail if they read <`redacted`> as secret parameter value

News from the htools

Redundancy, Metrics, `hail`

Additional redundancy checks

traditional Ganeti approach towards N+1 redundancy

Additional redundancy checks

traditional Ganeti approach towards N+1 redundancy

- N+1 redundancy for DRBD
  by reserving memory on the secondary

Additional redundancy checks

traditional Ganeti approach towards N+1 redundancy

- N+1 redundancy for DRBD
  by reserving memory on the secondary
- instances on shared storage can move anywhere

Additional redundancy checks

traditional Ganeti approach towards N+1 redundancy

- N+1 redundancy for DRBD
  by reserving memory on the secondary
- instances on shared storage can move anywhere
  . . . so it's probably fine

Additional redundancy checks

traditional Ganeti approach towards N+1 redundancy

- N+1 redundancy for DRBD
  by reserving memory on the secondary
- instances on shared storage can move anywhere
  . . . so it's probably fine
- instances on plain/file are lost on failure

Additional redundancy checks

traditional Ganeti approach towards N+1 redundancy

- N+1 redundancy for DRBD
  by reserving memory on the secondary
- instances on shared storage can move anywhere
  . . . so it's probably fine
- instances on plain/file are lost on failure
  . . . so nothing we can do anyway

Additional redundancy checks

traditional Ganeti approach towards N+1 redundancy

- N+1 redundancy for DRBD
  by reserving memory on the secondary
- instances on shared storage can move anywhere
  . . . so it's probably fine $\leftarrow$ not necessarily!
- instances on plain/file are lost on failure
  . . . so nothing we can do anyway

Additional redundancy checks

traditional Ganeti approach towards N+1 redundancy

- N+1 redundancy for DRBD
  by reserving memory on the secondary
- instances on shared storage can move anywhere
  . . . so it's probably fine ← not necessarily!
- instances on plain/file are lost on failure
  . . . so nothing we can do anyway ← reinstall?

Additional redundancy checks

Ganeti 2.15+ approach

- N+1 redundancy for DRBD
  by reserving memory on the secondary
- instances on shared storage can move anywhere
- instances on plain/file are lost on failure

Additional redundancy checks

Ganeti 2.15+ approach

- N+1 redundancy for DRBD
  by reserving memory on the secondary
- instances on shared storage can move anywhere
  $\rightsquigarrow$ capacity check!
- instances on plain/file are lost on failure
  $\rightsquigarrow$ capacity check!

Capacity check: for each node, verify that we can

- failover DRBD instances, and then
- evacuate/reinstall other instances in the same group

Memory reservation for DRBD instances

Components of the cluster metrics

Memory reservation for DRBD instances

Components of the cluster metrics

- counting violations
  (instances on offline nodes, . . .)

Memory reservation for DRBD instances

Components of the cluster metrics

- counting violations
  (instances on offline nodes, . . .)
- standard deviations (of relative usage)
  to keep resource usage balanced

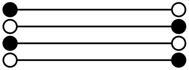
Memory reservation for DRBD instances

Components of the cluster metrics

- counting violations
  (instances on offline nodes, . . .)
- standard deviations (of relative usage)
  to keep resource usage balanced

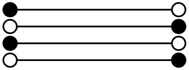
However, the reserved memory is not a constant amount to be distributed.

Memory reservation for DRBD instances



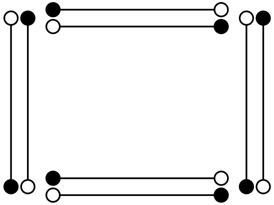
Components of the cluster metrics

- counting violations
  (instances on offline nodes, . . .)
- standard deviations (of relative usage)
  to keep resource usage balanced



However, the reserved memory is not a constant amount to be distributed.

Memory reservation for DRBD instances

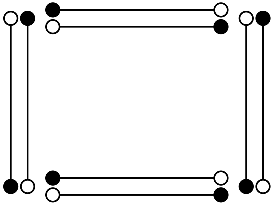


Components of the cluster metrics

- counting violations
  (instances on offline nodes, . . .)
- standard deviations (of relative usage)
  to keep resource usage balanced

However, the reserved memory is not a constant amount to be distributed.

Memory reservation for DRBD instances

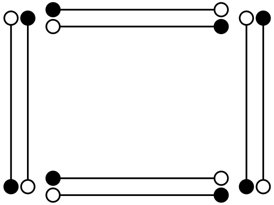


Components of the cluster metrics

- counting violations
  (instances on offline nodes, . . .)
- standard deviations (of relative usage)
  to keep resource usage balanced

However, the reserved memory is not a constant amount to be distributed. ⇒ Try to save to increase capacity.

Memory reservation for DRBD instances



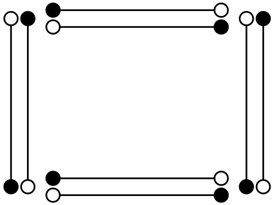
Components of the cluster metrics

- counting violations
  (instances on offline nodes, . . .)
- standard deviations (of relative usage)
  to keep resource usage balanced

However, the reserved memory is not a constant amount to be distributed. $\Rightarrow$ Try to save to increase capacity.

$\rightsquigarrow$ add sum of (relative) reserved memory as component *(Ganeti 2.15+)*

Memory reservation for DRBD instances



Components of the cluster metrics

- counting violations
  (instances on offline nodes, . . .)
- standard deviations (of relative usage)
  to keep resource usage balanced

However, the reserved memory is not a constant amount to be distributed. $\Rightarrow$ Try to save to increase capacity.

$\rightsquigarrow$ add sum of (relative) reserved memory as component
(Ganeti 2.15+)

!! Best metric value no longer 0.
(all `htools` interpret limits relative to the theoretical minimum)

Location awareness

. . . was discussed ever since the very first GanetiCon. . .

Location awareness

. . . was discussed ever since the very first GanetiCon. . .

and finally implemented (Ganeti 2.16+)!

Location awareness

. . . was discussed ever since the very first GanetiCon. . .
and finally implemented (Ganeti 2.16+)!

- cluster tags `htools:nlocation:x` make `x:foo` location tags
  (typically: common cause of failure; not hierarchical)

Location awareness

. . . was discussed ever since the very first GanetiCon. . .
and finally implemented (Ganeti 2.16+)!

- cluster tags `htools:nlocation:x` make `x:foo` location tags *(typically: common cause of failure; not hierarchical)*

  avoid *(cluster-metrics)*
  - primary and secondary in the same location
  - same service (exclusion tags!) in the same location

Location awareness

. . . was discussed ever since the very first GanetiCon. . .
and finally implemented (Ganeti 2.16+)!

- cluster tags `htools:nlocation:x` make `x:foo` location tags *(typically: common cause of failure; not hierarchical)*

  avoid *(cluster-metrics)*
  - primary and secondary in the same location
  - same service (exclusion tags!) in the same location

  Bonus: desired location of an instance
  $\rightsquigarrow$ Instance tag `htools:desiredlocation:x`

  (again, cluster metrics)

Location awareness

. . . was discussed ever since the very first GanetiCon. . .
and finally implemented (Ganeti 2.16+)!

- cluster tags `htools:nlocation:x` make `x:foo` location tags *(typically: common cause of failure; not hierarchical)*

Location awareness

...was discussed ever since the very first GanetiCon...
and finally implemented (Ganeti 2.16+)!

- cluster tags `htools:nlocation:x` make `x:foo` location tags *(typically: common cause of failure; not hierarchical)*

- Migration restrictions *(hypervisor upgrades)*
  cluster tags `htools:migration:x` ...

Location awareness

. . . was discussed ever since the very first GanetiCon. . .
and finally implemented (Ganeti 2.16+)!

- cluster tags `htools:nlocation:x` make `x:foo` location tags
  (typically: common cause of failure; not hierarchical)

- Migration restrictions *(hypervisor upgrades)*
  cluster tags `htools:migration:x` . . .

  migration only if
  - all migration tags of the source node also on the target, or
  - cluster tag `htools:allowmigration:y::z`
    for source tagged `y` and target node tagged `z`

Location awareness

. . . was discussed ever since the very first GanetiCon. . .
and finally implemented (Ganeti 2.16+)!

- cluster tags `htools:nlocation:x` make `x:foo` location tags
  (typically: common cause of failure; not hierarchical)

- Migration restrictions *(hypervisor upgrades)*
  cluster tags `htools:migration:x` . . .

  migration only if
  - all migration tags of the source node also on the target, or
  - cluster tag `htools:allowmigration:y::z`
    for source tagged `y` and target node tagged `z`

  Example: simple hypervisor update
  - tag updated nodes `hv:new`
  - cluster tags `htools:migration:hv`

Location awareness

. . . was discussed ever since the very first GanetiCon. . .
and finally implemented (Ganeti 2.16+)!

- cluster tags `htools:nlocation:x` make `x:foo` location tags
  (typically: common cause of failure; not hierarchical)

- Migration restrictions *(hypervisor upgrades)*
  cluster tags `htools:migration:x` . . .

  migration only if
  - all migration tags of the source node also on the target, or
  - cluster tag `htools:allowmigration:y::z`
    for source tagged `y` and target node tagged `z`

  Example: complex hypervisor situation
  - tag nodes `hv:foo`, `hv:bar`, `hv:baz`. . .
  - cluster tags `htools:migration:hv`
    `htools:allowmigration:hv:foo::hv:baz`, . . .

Location awareness

. . . was discussed ever since the very first GanetiCon. . .
and finally implemented (Ganeti 2.16+)!

- cluster tags `htools:nlocation:x` make `x:foo` location tags
  (typically: common cause of failure; not hierarchical)

- Migration restrictions *(hypervisor upgrades)*
  cluster tags `htools:migration:x` . . .

Allocation for partitioned Ganeti

Partiotioned Ganeti

Allocation for partitioned Ganeti

Partiotioned Ganeti

- recall idea: separate instance resources as far as possible to get reliable performace

Allocation for partitioned Ganeti

Partiotioned Ganeti

- recall idea: separate instance resources as far as possible to get reliable performace

$\rightsquigarrow$ Instances not moved

Allocation for partitioned Ganeti

Partiotioned Ganeti

- recall idea: separate instance resources as far as possible to get reliable performace

$\rightsquigarrow$ Instances not moved

$\Rightarrow$ Once a small instance (e.g. 1/4 node) is on a node, no full instance (1/1 node) can be put on there

Allocation for partitioned Ganeti

Partiotioned Ganeti

- recall idea: separate instance resources as far as possible to get reliable performace

$\rightsquigarrow$ Instances not moved

$\Rightarrow$ Once a small instance (e.g. 1/4 node) is on a node, no full instance (1/1 node) can be put on there

$\therefore$ Spreading instances equally is not the best choice *(want to fill up nodes to use capacity)*

Allocation for partitioned Ganeti

Partiotioned Ganeti

- recall idea: separate instance resources as far as possible to get reliable performace

$\rightsquigarrow$ Instances not moved

$\Rightarrow$ Once a small instance (e.g. 1/4 node) is on a node, no full instance (1/1 node) can be put on there

$\therefore$ Spreading instances equally is not the best choice *(want to fill up nodes to use capacity)*

Allocation metric for partitionend (2.15+): "Lost allocations"

- recall: instances come in discrete size (as per policy)

$\rightsquigarrow$ for each size, can count number that fits on a node
. . . and number lost by placement of new instance

- compare lexicographically, biggest size most important (disk space left as tie breaker)

Allocation for partitioned Ganeti

Allocation metric for partitionend (2.15+): “Lost allocations”

- recall: instances come in discrete size (as per policy)

⇝ for each size, can count number that fits on a node
  . . . and number lost by placement of new instance

- compare lexicographically, biggest size most important
  (disk space left as tie breaker)

Allocation for partitioned Ganeti

Allocation metric for partitionend (2.15+): "Lost allocations"

- recall: instances come in discrete size (as per policy)

$\rightsquigarrow$ for each size, can count number that fits on a node
 . . . and number lost by placement of new instance

- compare lexicographically, biggest size most important
 (disk space left as tie breaker)

Example: instances of size 1/1, 1/2, 1/4

Allocation for partitioned Ganeti

Allocation metric for partitionend (2.15+): "Lost allocations"

- recall: instances come in discrete size (as per policy)

$\rightsquigarrow$ for each size, can count number that fits on a node
  . . . and number lost by placement of new instance

- compare lexicographically, biggest size most important
  (disk space left as tie breaker)

Example: instances of size 1/1, 1/2, 1/4
preferences for 1/4 instance

- 3/4; lost (0, 0, 1), no left-over
- 1/4; lost (0, 0, 1), left-over 1/2
- 1/2; lost (0, 1, 1)
- 0/1; lost (1, 1, 1)

Allocation for partitioned Ganeti

Allocation metric for partitionend (2.15+): “Lost allocations”

- recall: instances come in discrete size (as per policy)

$\rightsquigarrow$ for each size, can count number that fits on a node
...and number lost by placement of new instance

- compare lexicographically, biggest size most important
(disk space left as tie breaker)

Example: instances of size 1/1, 1/2, 1/4
preferences for 1/2 instance

- 1/2; lost (0, 1, 2), no left-over
- 1/4; lost (0, 1, 2) left-over 1/4
- 0/1; lost (1, 1, 2)

Allocation of secondary node

- Ganeti supports disk-template conversions
  `gnt-instance modtify -t ...`

Allocation of secondary node

- Ganeti supports disk-template conversions
  `gnt-instance modtify -t ...`
- For conversion `plain` to `drbd` we need to chose a secondary

Allocation of secondary node

- Ganeti supports disk-template conversions
  `gnt-instance modtify -t ...`
- For conversion `plain` to `drbd` we need to chose a secondary
- Why not let `hail` choose it?

Allocation of secondary node

- Ganeti supports disk-template conversions
  `gnt-instance modtify -t ...`
- For conversion `plain` to `drbd` we need to chose a secondary
- Why not let `hail` choose it? **Now (2.16+) you can!**

Allocation of secondary node

- Ganeti supports disk-templace conversions
  `gnt-instance modtify -t ...`
- For conversion `plain` to `drbd` we need to chose a secondary
- Why not let `hail` choose it? **Now (2.16+) you can!**
- Extension of the IAllocator interface! *(official interface)*

Allocation of secondary node

- Ganeti supports disk-template conversions
  `gnt-instance modtify -t ...`
- For conversion `plain` to `drbd` we need to chose a secondary
- Why not let `hail` choose it? **Now (2.16+) you can!**
- Extension of the IAllocator interface! *(official interface)*
  Btw, who uses an allocator other than `hail`?

Allocation of secondary node

- Ganeti supports disk-templace conversions
  `gnt-instance modtify -t ...`
- For conversion `plain` to `drbd` we need to chose a secondary
- Why not let `hail` choose it? **Now (2.16+) you can!**
- Extension of the IAllocator interface! *(official interface)*

Allocation of secondary node

- Ganeti supports disk-template conversions
  `gnt-instance modtify -t ...`
- For conversion `plain` to `drbd` we need to chose a secondary
- Why not let `hail` choose it? **Now (2.16+) you can!**
- Extension of the IAllocator interface! *(official interface)*

New request type

```
"request": {
  "name": "notyetdrbd.example.com",
  "type": "allocate-secondary"
}
```

Job Filtering

reject, defer, and throttle jobs

Job Filters

New (2.13+) entity: **job filters.**

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail
- priority *(non-negative integer; smaller is more important)*

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail
- priority *(non-negative integer; smaller is more important)*
- watermark *(highest job id at submission time)*

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail
- priority *(non-negative integer; smaller is more important)*
- watermark *(highest job id at submission time)*
- list of predicates *(implicit "and")*

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail
- priority *(non-negative integer; smaller is more important)*
- watermark *(highest job id at submission time)*
- list of predicates *(implicit "and")*

A predicate is list: predicate name + suitable parameters

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail
- priority *(non-negative integer; smaller is more important)*
- watermark *(highest job id at submission time)*
- list of predicates *(implicit "and")*

A predicate is list: predicate name + suitable parameters
so far, always a predicate in the query language

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail
- priority *(non-negative integer; smaller is more important)*
- watermark *(highest job id at submission time)*
- list of predicates *(implicit "and")*

A predicate is list: predicate name + suitable parameters
so far, always a predicate in the query language

- `jobid`. Field `id`, values numbers or `"watermark"`

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail
- priority *(non-negative integer; smaller is more important)*
- watermark *(highest job id at submission time)*
- list of predicates *(implicit "and")*

A predicate is list: predicate name + suitable parameters
so far, always a predicate in the query language

- `jobid`. Field `id`, values numbers or `"watermark"`
- `opcode`. Fields `OP_ID`, plus whatever fields the opcode has *("or" over the op-codes of a job)*

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail
- priority *(non-negative integer; smaller is more important)*
- watermark *(highest job id at submission time)*
- list of predicates *(implicit "and")*

A predicate is list: predicate name + suitable parameters
so far, always a predicate in the query language

- `jobid`. Field `id`, values numbers or `"watermark"`
- `opcode`. Fields `OP_ID`, plus whatever fields the opcode has *("or" over the op-codes of a job)*
- `reason`. Fields `source`, `reason`, `timestamp` *("or" over all entries of all opcodes)*

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail
- priority *(non-negative integer; smaller is more important)*
- watermark *(highest job id at submission time)*
- list of predicates *(implicit "and")*

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail
- priority *(non-negative integer; smaller is more important)*
- watermark *(highest job id at submission time)*
- list of predicates *(implicit "and")*
- action

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail
- priority *(non-negative integer; smaller is more important)*
- watermark *(highest job id at submission time)*
- list of predicates *(implicit "and")*
- action
  - `ACCEPT`

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail
- priority *(non-negative integer; smaller is more important)*
- watermark *(highest job id at submission time)*
- list of predicates *(implicit "and")*
- action
  - `ACCEPT`
  - `PAUSE`

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail
- priority *(non-negative integer; smaller is more important)*
- watermark *(highest job id at submission time)*
- list of predicates *(implicit "and")*
- action
  - `ACCEPT`
  - `PAUSE`
  - `REJECT`

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail
- priority *(non-negative integer; smaller is more important)*
- watermark *(highest job id at submission time)*
- list of predicates *(implicit "and")*
- action
  - `ACCEPT`
  - `PAUSE`
  - `REJECT`
  - `CONTINUE`

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail
- priority *(non-negative integer; smaller is more important)*
- watermark *(highest job id at submission time)*
- list of predicates *(implicit "and")*
- action
  - `ACCEPT`
  - `PAUSE`
  - `REJECT`
  - `CONTINUE`
  - `RATE_LIMIT` *n*

Job Filters

New (2.13+) entity: **job filters.** Given by the following data

- UUID *(Ganeti will assign, if not provided)*
- reason trail
- priority *(non-negative integer; smaller is more important)*
- watermark *(highest job id at submission time)*
- list of predicates *(implicit "and")*
- action

Examples of Job Filters

Examples of Job Filters

- Soft drain a queue

```
{"priority": 0, "action": "PAUSE",
 "predicates": [["jobid", [">", "id", "watermark"]]] }
```

Examples of Job Filters

- Soft drain a queue

```
{"priority": 0, "action": "PAUSE",
 "predicates": [["jobid", [">", "id", "watermark"]]] }
```

- reject jobs not belonging to a specific maintenance

```
{"priority": 0, "action": "ACCEPT",
 "predicates": [["reason", ["=~", "reason",
                                  "maint pink bunny"]]] }
{"priority": 1, "action": "REJECT",
 "predicates": [["jobid", [">", "id", "watermark"]]] }
```

Examples of Job Filters

- Soft drain a queue

```
{"priority": 0, "action": "PAUSE",
 "predicates": [["jobid", [">", "id", "watermark"]]] }
```

- reject jobs not belonging to a specific maintenance

```
{"priority": 0, "action": "ACCEPT",
 "predicates": [["reason", ["=~", "reason",
                          "maint pink bunny"]]] }
{"priority": 1, "action": "REJECT",
 "predicates": [["jobid", [">", "id", "watermark"]]] }
```

- limit disk-replacements to throttle replication traffic

```
{"priority": 99, "action": ["RATE_LIMIT", 10],
 "predicates": [["opcode", ["=", "OP_ID",
                     "OP_INSTANCE_REPLACE_DISKS"]]] }
```

Upcoming (2.17)

`maintd`

Upcoming (2.17): Maintenance Daemon

- new data collector for node-status

Upcoming (2.17): Maintenance Daemon

- new data collector for node-status
    - Command in white-listed directory

Upcoming (2.17): Maintenance Daemon

- new data collector for node-status
  - Command in white-listed directory
  - should output a JSON object (`status` plus opaque `details`)

Upcoming (2.17): Maintenance Daemon

- new data collector for node-status
  - Command in white-listed directory
  - should output a JSON object (`status` plus opaque `details`)
    `Ok`, `live-repair`, `evacuate`, `evacuate-failover`

Upcoming (2.17): Maintenance Daemon

- new data collector for node-status
  - Command in white-listed directory
  - should output a JSON object (`status` plus opaque `details`)
    `Ok`, `live-repair`, `evacuate`, `evacuate-failover`
  - default `""` for “everything OK”

Upcoming (2.17): Maintenance Daemon

- new data collector for node-status
  - Command in white-listed directory
  - should output a JSON object (`status` plus opaque `details`)
    `Ok`, `live-repair`, `evacuate`, `evacuate-failover`
  - default `""` for “everything OK”
- New daemon `maintd`

Upcoming (2.17): Maintenance Daemon

- new data collector for node-status
    - Command in white-listed directory
    - should output a JSON object (`status` plus opaque `details`)
      `Ok`, `live-repair`, `evacuate`, `evacuate-failover`
    - default `""` for "everything OK"
- New daemon `maintd`
    - handles repairs requested by node-status data collector
      (opt-in by setting the collector)

Upcoming (2.17): Maintenance Daemon

- new data collector for node-status
  - Command in white-listed directory
  - should output a JSON object (`status` plus opaque `details`) `Ok`, `live-repair`, `evacuate`, `evacuate-failover`
  - default `""` for “everything OK”
- New daemon `maintd`
  - handles repairs requested by node-status data collector *(opt-in by setting the collector)*
  - does `harep`-style repairs *(opt-in by setting tags)*

Upcoming (2.17): Maintenance Daemon

- new data collector for node-status
  - Command in white-listed directory
  - should output a JSON object (`status` plus opaque `details`)
    `Ok`, `live-repair`, `evacuate`, `evacuate-failover`
  - default `""` for "everything OK"
- New daemon `maintd`
  - handles repairs requested by node-status data collector
    (opt-in by setting the collector)
  - does `harep`-style repairs
    (opt-in by setting tags)
  - does load-based balancing
    (opt-in by setting flag in the configuration)

The End

Thank you for your attention

Ganeti releases are availbale from `http://downloads.ganeti.org/` and signed by the following key.

```
pub   4096R/6AA8276A 2013-12-10 [expires: 2017-12-29]
      Key fingerprint = 7A8D 09A0 12E9 1D94 56E2  996B A876 A343 6AA8 276A
uid                  Ganeti (Release signing key) <ganeti@googlegroups.com>
sub   4096R/3F3F9806 2013-12-10 [expires: 2017-12-29]
```