Ganeti automatic instance allocation

Documents Ganeti version 1.2

1. Introduction

Currently in Ganeti the admin has to specify the exact locations for an instance's node(s). This prevents a completely automatic node evacuation, and is in general a nuisance.

The iallocator framework will enable automatic placement via external scripts, which allows customization of the cluster layout per the site's requirements.

2. User-visible changes

There are two parts of the ganeti operation that are impacted by the auto-allocation: how the cluster knows what the allocator algorithms are and how the admin uses these in creating instances.

An allocation algorithm is just the filename of a program installed in a defined list of directories.

2.1. Cluster configuration

At configure time, the list of the directories can be selected via the `--with-iallocator-search-path=LIST` option, where **`LIST`** is a comma-separated list of directories. If not given, this defaults to `$libdir/ganeti/iallocators`, i.e. for an installation under `/usr`, this will be `/usr/lib/ganeti/iallocators`.

Ganeti will then search for allocator script in the configured list, using the first one whose filename matches the one given by the user.

2.2. Command line interface changes

The node selection options in instanece add and instance replace disks can be replace by the new `--iallocator` *`NAME`* option, which will cause the autoassignation. The selected node(s) will be show as part of the command output.

3. IAllocator API

The protocol for communication between Ganeti and an allocator script will be the following:

1. ganeti launches the program with a single argument, a filename that contains a JSON-encoded structure (the input message)
2. if the script finishes with exit code different from zero, it is considered a general failure and the full output will be reported to the users; this can be the case when the allocator can't parse the input message;
3. if the allocator finishes with exit code zero, it is expected to output (on its stdout) a JSON-encoded structure (the response)

3.1. Input message

The input message will be the JSON encoding of a dictionary containing the following:

version

: the version of the protocol; this document specifies version 1

cluster_name

: the cluster name

cluster_tags

: the list of cluster tags

request

: a dictionary containing the request data:

  type

  : the request type; this can be either `allocate` or `relocate`; the `allocate` request is used when a new instance needs to be placed on the cluster, while the `relocate` request is used when an existing instance needs to be moved within the cluster

  name

  : the name of the instance; if the request is a realocation, then this name will be found in the list of instances (see below), otherwise is the FQDN of the new instance

  required_nodes

  : how many nodes should the algorithm return; while this information can be deduced from the instace's disk template, it's better if this computation is left to Ganeti as then allocator scripts are less sensitive to changes to the disk templates

disk_space_total

: the total disk space that will be used by this instance on the (new) nodes; again, this information can be computed from the list of instance disks and its template type, but Ganeti is better suited to compute it

If the request is an allocation, then there are extra fields in the request dictionary:

disks

: list of dictionaries holding the disk definitions for this instance (in the order they are exported to the hypervisor):

  mode

  : either `w` or `w` denoting if the disk is read-only or writable; for Ganeti 1.2, this will always be `w`

  size

  : the size of this disk in mebibyte

nics

: a list of dictionaries holding the network interfaces for this instance, containing:

  ip

  : the IP address that Ganeti know for this instance, or null

  mac

  : the MAC address for this interface

  bridge

  : the bridge to which this interface will be connected

vcpus

: the number of VCPUs for the instance

disk_template

: the disk template for the instance

memory

: the memory size for the instance

os

: the OS type for the instance

tags

the list of the instance's tags

If the request is of type relocate, then there is one more entry in the request dictionary, named `relocate_from`, and it contains a list of nodes to move the instance away from; note that with Ganeti 1.2, this list will always contain a single node, the current secondary of the instance.

instances

a dictionary with the data for the current existing instance on the cluster, indexed by instance name; the contents are similar to the instance definitions for the allocate mode, with the addition of:

should_run

if this instance is set to run (but not the actual status of the instance)

nodes

list of nodes on which this instance is placed; the primary node of the instance is always the first one

nodes

dictionary with the data for the nodes in the cluster, indexed by the node name; the dict contains:

total_disk

the total disk size of this node (mebibytes)

free_disk

the free disk space on the node

total_memory

the total memory size

free_memory

free memory on the node; note that currently this does not take into account the instances which are down on the node

total_cpus

the physical number of CPUs present on the machine; depending on the hypervisor, this might or might not be equal to how many CPUs the node operating system sees;

primary_ip

the primary IP address of the node

secondary_ip

the secondary IP address of the node (the one used for the DRBD replication); note that this can be the same as the primary one

tags

list with the tags of the node

3.2. Respone message

The response message is much more simple than the input one. It is also a dict having three keys:

success

a boolean value denoting if the allocation was successfull or not

info

a string with information from the scripts; if the allocation fails, this will be shown to the user

nodes

the list of nodes computed by the algorithm; even if the algorithm failed (i.e. success is false), this must be returned as an empty list; also note that the length of this list must equal the `requested_nodes` entry in the input message, otherwise Ganeti will consider the result as failed

4. Examples

4.1. Input messages to scripts

Input message, new instance allocation

```
{
  "cluster_tags": [],
  "request": {
    "required_nodes": 2,
    "name": "instance3.example.com",
    "tags": [
      "type:test",
      "owner:foo"
    ],
    "type": "allocate",
    "disks": [
      {
```

```
      "mode": "w",
      "size": 1024
    },
    {
      "mode": "w",
      "size": 2048
    }
  ],
  "nics": [
    {
      "ip": null,
      "mac": "00:11:22:33:44:55",
      "bridge": null
    }
  ],
  "vcpus": 1,
  "disk_template": "drbd",
  "memory": 2048,
  "disk_space_total": 3328,
  "os": "etch-image"
},
"cluster_name": "cluster1.example.com",
"instances": {
  "instance1.example.com": {
    "tags": [],
    "should_run": false,
    "disks": [
      {
        "mode": "w",
        "size": 64
      },
      {
        "mode": "w",
        "size": 512
      }
    ],
    "nics": [
      {
        "ip": null,
        "mac": "aa:00:00:00:60:bf",
        "bridge": "xen-br0"
      }
    ],
    "vcpus": 1,
    "disk_template": "plain",
    "memory": 128,
    "nodes": [
      "nodee1.com"
    ],
    "os": "etch-image"
  },
  "instance2.example.com": {
    "tags": [],
```

```
    "should_run": false,
    "disks": [
      {
        "mode": "w",
        "size": 512
      },
      {
        "mode": "w",
        "size": 256
      }
    ],
    "nics": [
      {
        "ip": null,
        "mac": "aa:00:00:55:f8:38",
        "bridge": "xen-br0"
      }
    ],
    "vcpus": 1,
    "disk_template": "drbd",
    "memory": 512,
    "nodes": [
      "node2.example.com",
      "node3.example.com"
    ],
    "os": "etch-image"
  }
},
"version": 1,
"nodes": {
  "node1.example.com": {
    "total_disk": 858276,
    "primary_ip": "192.168.1.1",
    "secondary_ip": "192.168.2.1",
    "tags": [],
    "free_memory": 3505,
    "free_disk": 856740,
    "total_memory": 4095
  },
  "node2.example.com": {
    "total_disk": 858240,
    "primary_ip": "192.168.1.3",
    "secondary_ip": "192.168.2.3",
    "tags": ["test"],
    "free_memory": 3505,
    "free_disk": 848320,
    "total_memory": 4095
  },
  "node3.example.com.com": {
    "total_disk": 572184,
    "primary_ip": "192.168.1.3",
    "secondary_ip": "192.168.2.3",
    "tags": [],
```

```
      "free_memory": 3505,
      "free_disk": 570648,
      "total_memory": 4095
    }
  }
}
```

Input message, reallocation. Since only the request entry in the input message is changed, the following shows only this entry:

```
  "request": {
    "relocate_from": [
      "node3.example.com"
    ],
    "required_nodes": 1,
    "type": "relocate",
    "name": "instance2.example.com",
    "disk_space_total": 832
  },
```

4.2. Response messages

Successful response message:

```
{
  "info": "Allocation successful",
  "nodes": [
    "node2.example.com",
    "node1.example.com"
  ],
  "success": true
}
```

Failed response message:

```
{
  "info": "Can't find a suitable node for position 2 (already selected: node2.example.com)"
  "nodes": [],
  "success": false
}
```

4.3. Command line messages

```
# gnt-instance add -t plain -m 2g --os-size 1g --swap-size 512m --iallocator dumb-allocator
Selected nodes for the instance: node1.example.com
* creating instance disks...
[...]
```

```
# gnt-instance add -t plain -m 3400m --os-size 1g --swap-size 512m --iallocator dumb-alloca
Failure: prerequisites not met for this operation:
Can't compute nodes using iallocator 'dumb-allocator': Can't find a suitable node for posit

# gnt-instance add -t drbd -m 1400m --os-size 1g --swap-size 512m --iallocator dumb-allocat
Failure: prerequisites not met for this operation:
Can't compute nodes using iallocator 'dumb-allocator': Can't find a suitable node for posit
```