Ganeti: Cluster Virtualization Manager

GUIDO TROTTER AND TOM LIMONCELLI

Guido is a Senior Engineer at Google Munich, where he leads the development of Ganeti, an open source cluster virtualization manager. His interests and speaking topics are in virtualization, distributed systems, and advanced networking for Linux. He has presented Ganeti and taught classes about it at LISA, Fosdem, KVM Forum, LinuxCon, and many more gatherings. He is also a Debian developer and active member of the community. His activities can mostly be seen through the Ganeti lists (ganeti@googlegroups.com and ganeti-devel@googlegroups.com). gt@google.com

Tom is an internationally recognized author, speaker, and system administrator. His best known books include Time Management for System Administrators (O'Reilly) and The Practice of System and Network Administration (Addison-Wesley). In 2005 he received the USENIX SAGE Outstanding Achievement Award. He blogs at http://EverythingSysadmin.com. tal@everythingsysadmin.com

Ganeti is a software stack that allows easily managing a collection of physical machines (nodes) to host virtualized machines (instances). This article explains its background, principles, usage, and direction.

Virtualization and the Cloud

Virtualization is an important building block in today's computer infrastructures. Whether you're running a small lab or a huge datacenter, the decoupling of the physical hardware from the services that run on it is key for increasing uptime, resource utilization, and maintainability.

Many solutions exist both in the open source world and in the proprietary one for virtualizing workloads. Some of these solutions are stand-alone, whereas others require many different components in order to work. All of them are based on the same basic necessary components: (1) a hypervisor, which provides the basic layer on which the virtualized system runs; (2) a storage backend that hosts VMs' data; and (3) a network infrastructure that allows VMs to send and receive traffic. Optionally, extra software can be used to run and manage these systems, and provide abstractions on top of them. Ganeti is software that fulfills this role, and it can be used to manage the previously listed components.

Why Ganeti?

People choose Ganeti to run their virtualized infrastructure for many reasons. Most of them are captured by the low barrier to entry at which they can get started on Ganeti; they don't need to configure a huge system if all they need is to build a small infrastructure while, at the same time, retaining the option to scale easily to thousands of nodes.

Other reasons include its architecture (see below), which they can easily plug in to and extend, and the friendly community. Ganeti also has a good test and QA suite, which hopefully reflect in high-quality releases.

Of course, like anything, Ganeti doesn't suit all needs, but we'd like you to try it and see if it serves your organization. We're interested in getting feedback about your experience, what does or doesn't work for you. Ganeti is an enterprise production-ready system, used at Google and in many other organizations, which might help you build part of your infrastructure.

Building Blocks

As mentioned above, virtualization hosting needs many different building blocks in order to work. We'll now go through the most important ones, focusing on technologies that are supported by Ganeti. In each section we'll show how we can mix and match the underlying technology to get a virtualized environment that is customized to our needs, while being managed in a common way.

Note that the ability to "mix and match" is a key feature of Ganeti often not found in other environments, which tend to pick all the technologies that one must use; this makes Ganeti a powerful "building block" rather than a prebuilt "one-size-fits-all" system.

Hypervisor

The following hypervisors are supported by Ganeti:

- Xen is a microkernel style stand-alone hypervisor that runs "under" its guest systems and provides them resources with the help of one or more trusted VMs (e.g., dom0). Xen can run different types of virtualized environments depending on the guests' support.
- KVM is a system to run a hardware-aided virtualization environment on top of a standard Linux system.
- LXC (experimental support) is a way of running independent containers on top of the same kernel. The virtualization level is provided by giving the processes running in each container a different "view" of the system by insulating its namespaces (user, network, process, etc.).

Xen and KVM are two of the most adopted open source hypervisors available today, especially in the server world. Using them, you can easily configure one machine to run virtualized workloads. The two have a different architecture and approach to virtualizing machines.

Naturally, they are not the only options, and research is needed to see which virtualization system is a better fit for your infrastructure and workload.

Storage

Possible storage solutions for Ganeti virtualized environments include:

- Simple files residing on the host machine
- Local volumes on the host machines (e.g., physical block devices or LVM ones)
- Local data with over-the-network replication (e.g., using DRBD)
- Volumes (or files) hosted on an external dedicated hardware (SAN)
- Volumes hosted on a distributed object store (e.g., RADOS)

Ganeti will automatically manage disks and create/delete them as needed for its instances. Stand-alone management of disks is not supported yet but is considered in the roadmap.

Networking

Ganeti allows running VMs with any number of virtual network interfaces. Each of them can be connected to a Linux kernel bridge, an Open vSwitch, or have its traffic routed by the host system.

Currently, all "connection" systems (bridges, Open vSwitches, routing) must be configured on the nodes, but dynamic connection/disconnection of them is in the product roadmap.

"The Cloud"

Although the cloud is a vague term, in the virtualization context it is usually used to mean "Infrastructure as a Service" (IaaS). This is usually implemented as a collection of software providing an API that can be used to provide the resources described above. Ganeti can be used as a basic building block to provide such a service and effectively run a "private cloud."

Other Choices

Of course it might happen that your favorite technology is not yet working well with Ganeti. Although we couldn't implement all possible choices, we tried to keep our APIs standard; as such, perhaps it will be possible with some development work to design and support your architecture and make it part of the Ganeti ecosystem. After all, most of these choices were added during the product lifetime, rather than being born into it, and as such we hope to be able to accommodate more in the future, as the product and the technology space develops.

Ganeti's Principles

Ganeti is built on the following ideas:

- Extremely simple to install and use: We strive to make Ganeti as simple as possible, not requiring hand configuration of many different components, at least for a basic usage.
- Usable as a basic infrastructure building block: A Ganeti cluster must be able to host your DNS, DHCP, and many other basic services. As such, Ganeti itself will not depend on them at startup time, allowing you to cold-power-on a cluster without those services. Some of them may still be needed during normal run, for example, to add new nodes or instances.
- Multi platform: We want Ganeti to run on and with your favorite distribution of Linux, hypervisor, backend storage, and networking system. We try to make sure it exports a customizable interface that you can use to plug in your customizations.
- Simple software dependencies: We try not to depend on too new libraries and languages, to make the system easy to build and deploy. To be extremely conservative, we chose Debian stable (plus, occasionally, backports.org) as our reference for choosing library dependencies.
- Minimal hardware/platform/system dependencies: We avoid depending on any component or infrastructure choice that we feel is not strictly necessary. We are happy to support extra features when those are present, but we don't want to impose them on everybody's installation.
- Good open source citizen: We discuss designs and code on the public development list before committing to it. We make sure our entire development process is transparent, and we don't do "code dumps" at release time. This allows for ease of cooperation between members of the community.

We encourage external contributions and designs, and are happy to cooperate on the direction of this software platform. Issues as well as designs that we're working on are publicly visible, as is each code change and the related discussion before it gets merged. Patches from team members, other Google teams, or external contributors go through exactly the same review process in order to become part of the tree.

Quick Ganeti Example

Ganeti administration is done through command-line tools or the Ganeti Remote API (RAPI). Web-based admin tools have also been created, such as the Ganeti Web Manager project at Oregon State University Open Source Lab (http://ganeti-web-mgr.readthedocs.org/en/latest/); however, this being *;login:,* we will present some command-line examples.

Ganeti commands all start with `gnt-` and require a subcommand. For example `gnt-instance info foo` outputs information about an instance named `foo`.

Initializing the Cluster

This is the first step for setting up Ganeti and requires an unused host name for the cluster, associated with an unused IP address. Ganeti will set up the IP address automatically on the master node:

```
# gnt-cluster init [-s secondary_ip] cluster.example.com
```

Note that the basic initialization has many default assumptions. You may want to configure your enabled hypervisors, their parameters, and much more. See `gnt-cluster modify` for more information. The `-s` parameter configures the cluster with a separate replication network. If it is set, all nodes will also need to be added specifying the `-s` option, and their secondary IP.

Creating an Instance

Creating an instance can be simple if cluster-wide defaults have been set; it can be as simple as specifying an operating system image name, and amount of RAM to allocate to the VM, size of the virtual disk to be created, the storage type, and the instance's name:

```
# gnt-instance add -o ubuntu_std -B memory=1024M -s 100G -t drbd inst1.example.com
Thu Mar 21 14:16:04 2013 * creating instance disks...
Thu Mar 21 14:16:08 2013 adding instance inst1.example.com to cluster config
Thu Mar 21 14:16:08 2013 * wiping instance disks...
Thu Mar 21 14:16:09 2013  - INFO: * Wiping disk 0
Thu Mar 21 14:16:20 2013  - INFO:  - done: 1.0% ETA: 18m 44s
Thu Mar 21 14:17:26 2013  - INFO:  - done: 7.0% ETA: 17m 4s
[...]
Thu Mar 21 14:34:18 2013  - INFO:  - done: 91.0% ETA: 1m 48s
Thu Mar 21 14:35:21 2013  - INFO:  - done: 96.0% ETA: 48s
Thu Mar 21 14:36:12 2013  - INFO: Waiting for instance inst1.example.com to sync disks.
Thu Mar 21 14:36:12 2013  - INFO: Instance inst1.example.com's disks are in sync.
```

Ganeti will use its allocation algorithm to find nodes that have room and create the instance. The default allocation algorithm can be overridden by specifying the exact nodes the instance should live on (primary:secondary) with `-n node1:node2`. Other instance parameters can be specified, such as configuring the virtual NIC to have a specific MAC address: e.g., `--net 0:mac=aa:00:00:fa:3a:3f`

Listing Instances

Here is an example that shows a list of instances:

```
# gnt-instance list
Instance           Hypervisor OS          Primary_node      Status     Memory
george.example.com xen-pvm    ubuntu_std node3.example.com running    4.0G
inst1.example.com  xen-pvm    ubuntu_std node2.example.com running    512M
john.example.com   xen-pvm    ubuntu_std node2.example.com ADMIN_down 4.0G
paul.example.com   xen-pvm    ubuntu_std node4.example.com running    2.0G
ringo.example.com  xen-pvm    ubuntu_std node1.example.com running    2.0G
```

Listing Nodes

Here is an example of listing all the nodes. Note that node5 is offline for repairs and therefore not all information about it can be displayed. Before sending it to repairs, the instances were migrated to other nodes; thus, the Pinst/Sinst values (number of primary and secondary instances) are 0.

```
# gnt-node list
Node                         Dtotal  Dfree Mtotal  Mnode Mfree  Pinst Sinst
node1.example.com            671.9G  83.1G  16.0G  1023M  5.8G      4     3
node2.example.com            671.9G  99.1G  16.0G  1023M  8.3G      4     3
node3.example.com            671.9G 212.9G  16.0G  1023M  6.8G      3     5
node4.example.com            671.9G 268.9G  16.0G  1023M  6.3G      4     4
node5.example.com                 *      *      *      *     *      0     0
```

Adding a New Node

Adding a new node to the cluster is surprisingly easy. Once the software is installed and storage/network configurations are complete, the command to add the node only requires specifying the two things Ganeti cannot determine on its own: the nodes name and the IP address of its replication NIC.

```
# gnt-node add -s 192.168.20.2 node6.example.com
-- WARNING --
Performing this operation is going to replace the ssh daemon keypair on the target machine
    (node6.example.com) with the ones of the current one and grant full intra-cluster ssh root
    access to/from it

Sat Mar 16 14:53:04 2013  - INFO: Node will be a master candidate executed successfully
```

Verifying Consistency

In our final example, we run the Ganeti cluster command to check the system health and warn of any issues. In this case, we see a warning about an expired certificate used to authenticate RAPI requests. The command also checks for connectivity problems between the master and each node, storage problems, and much more:

```
# gnt-cluster verify
Submitted jobs 74499, 74500
Waiting for job 74499 ...
Thu Mar 21 14:33:23 2013 * Verifying cluster config
Thu Mar 21 14:33:23 2013 * Verifying cluster certificate files
Thu Mar 21 14:33:23 2013  - ERROR: cluster: While verifying
    /var/lib/ganeti/rapi.pem: Certificate is expired
    (valid from 2011-12-09 07:01:06 to 2012-12-09 07:11:06)
```

```
Thu Mar 21 14:33:23 2013 * Verifying hypervisor parameters
Thu Mar 21 14:33:23 2013 * Verifying all nodes belong to an existing group
Waiting for job 74500 ...
Thu Mar 21 14:33:23 2013 * Verifying group 'GROUP1'
Thu Mar 21 14:33:23 2013 * Gathering data (2 nodes)
Thu Mar 21 14:33:26 2013 * Gathering disk information (2 nodes)
Thu Mar 21 14:33:26 2013 * Verifying configuration file consistency
Thu Mar 21 14:33:26 2013 * Verifying node status
Thu Mar 21 14:33:26 2013 * Verifying instance status
Thu Mar 21 14:33:26 2013 * Verifying orphan volumes
Thu Mar 21 14:33:26 2013 * Verifying N+1 Memory redundancy
Thu Mar 21 14:33:26 2013 * Other Notes
Thu Mar 21 14:33:26 2013 * Hooks Results
```

The Ganeti commands are extensively documented with detailed man pages plus help summaries when the `--help` flag is given.

Running Ganeti in Production

Besides Ganeti itself, we recommend the use of other tools in order to have a scalable enterprise level environment:

- Self-installing nodes: These can be achieved from any automated installer, coupled with a good configuration management system.
- Monitoring: Various products can be configured to do black-box and white-box monitoring of Ganeti nodes, its storage devices, and its instances.
- Self-healing products: Ganeti can be coupled with Linux-HA or your monitoring system can be instrumented to perform cluster self-healing operations, and not require manual intervention on node downtime or other hardware errors. If this is coupled with white-box monitoring, nodes can be evacuated when they start to show problems but before they fail, thus avoiding any downtime.
- Administration interfaces: These allow users to self-service create/delete and modify their instances, and to access the instance console.

Ganeti Internals

The Ganeti platform is a collection of daemons and command line utilities written in Python and Haskell. The platform's main components are:

- The CLI scripts, which take user commands and transmit them to the master daemon via the LUXI protocol.
- The RAPI daemon, which accepts Ganeti operations over https and transmits them to the master daemon via the LUXI protocol.
- The Master daemon, which performs most Ganeti operations (opcodes) by executing logical units of code. Opcodes are units of Ganeti work, and do things such as starting instances, creating new ones, and so on. They can be serialized together in jobs, or submitted separately to allow the master daemon to run them concurrently in safety, without conflicting with each other. Jobs (consisting of one or more opcodes) are accepted via a JSON-on-UNIX-sockets protocol, and are sent mostly by the CLI utilities, the RAPI daemon, or our extra tools.
- The Node daemon runs on all nodes and performs the subunits of actual work needed to implement the logical units. It performs minimal self-standing operations on each target node (e.g., creating a block device, writing a file to disk, executing a command), and it is controlled by the master daemon by an RPC system.
- htools are Ganeti's "cluster optimizations" suite. They include tools to rebalance the cluster's load (hbal), to allocate instances automatically in the best possible place (hail), to calculate how many more instances a cluster can accommodate (hspace), or to calculate how to best run maintenances on a cluster (hroller).

Other less crucial, and sometimes optional, Ganeti components are:

- The confd daemon, which has read-only access to the cluster config, and uses it to answer config queries.
- The monitoring agent daemon (mond, which provides real-time per-node information via http+json, and can be queried by a monitoring system to perform white-box monitoring on the cluster.

How to Reach the Community

Ganeti is discussed at the ganeti@googlegroups.com list. There you can ask questions, get help debugging issues, or just discuss your setup.

Development happens at ganeti-devel@googlegroups.com in the format of patches (which can be sent via git format-patch plus git send-email) and design docs. Contributing requires signing the Google Code Contributor License Agreement (CLA) by the copyright owner (either the individual or the company, depending), who retains copyright to his or her contributions.

The Ganeti project can be found at https://code.google.com/p/ganeti, the source code is at http://git.ganeti.org/, and documentation is built from the git tree and exported in real-time at http://docs.ganeti.org/.

Ganeti Roadmap

We have many ideas about Ganeti, which will be tackled as time and priority allow. In the near-to-medium future, we want to focus on:

1. better storage alternatives, and promoting disks from second-class citizens to first-class ones, which can be managed without being just part of a virtual machine;
2. dynamic networking, to make the datacenter networking architecture more efficient, scalable, and not dependent on preconfiguring; and
3. better integration with other clouds, harnessing the private/public cloud interconnection.

But the first idea we work on could be your idea. Just install Ganeti, and if you find something missing, let's discuss a design. We'll be happy to help you get up to speed and upstream your features, for the benefit of your installation and the entire community.