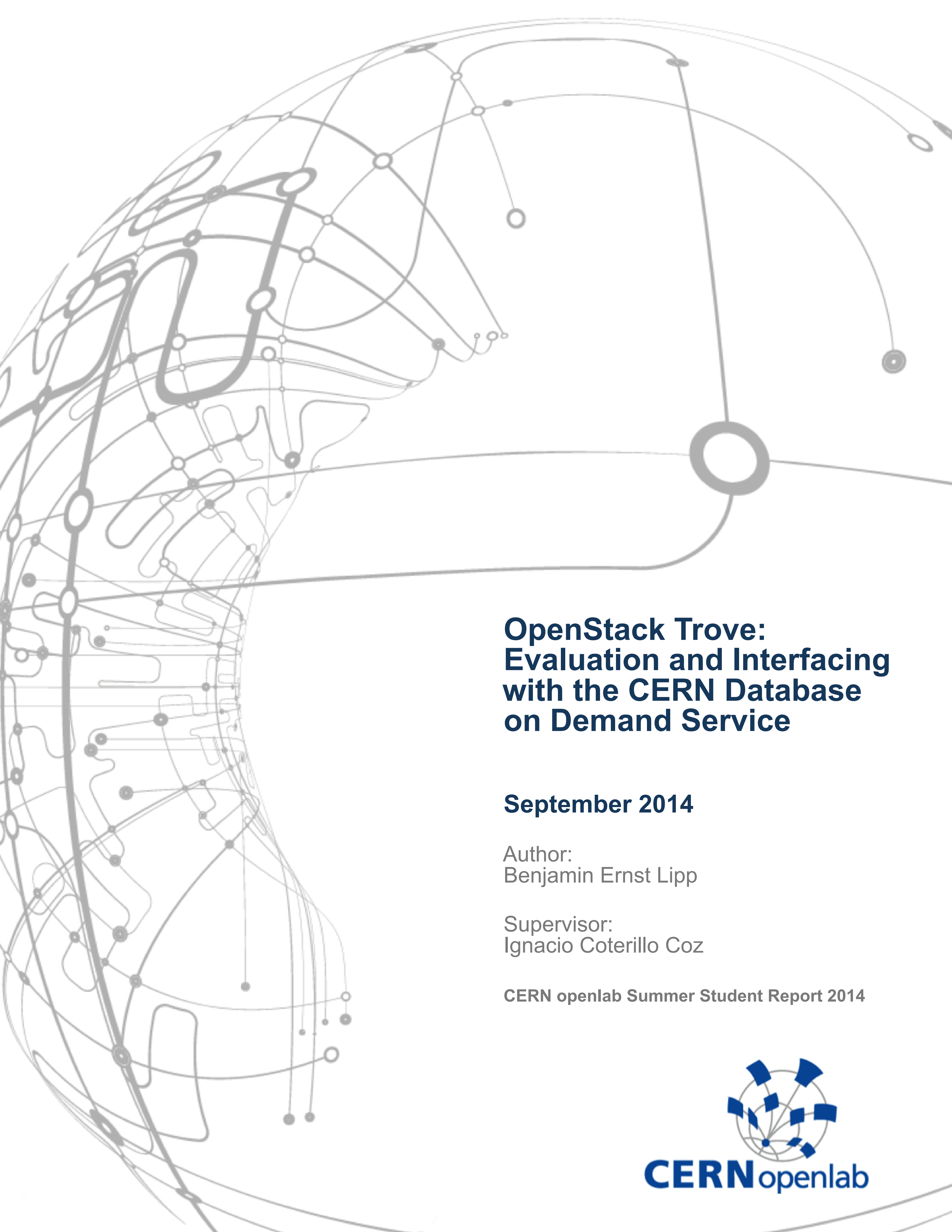


OpenStack Trove: Evaluation and Interfacing with the CERN Database on Demand Service

September 2014

Author:
Benjamin Ernst Lipp

Supervisor:
Ignacio Coterillo Coz

CERN openlab Summer Student Report 2014

Project Specification

As part of the ongoing migration of CERN's virtual infrastructure to an OpenStack based solution the CERN IT-DB group, responsible of the Database on Demand service, is migrating towards OpenStack from an Oracle VM platform, on an effort to converge with the global CERN IT infrastructure platforms.

The Database on Demand Service is a Database as a service developed in-house within the CERN IT-DB group. The service manages the underlying infrastructure while providing the end user with an easy to use interface to perform database backups, recoveries, database configuration, and monitoring.

Since its most recent release Icehouse, the OpenStack platform includes a new component, named Trove, which provides Database as a Service functionalities. Thus, the main goal of the project is to evaluate this component, OpenStack Trove, from the point of view of the feasibility of using it as a resource provider for the Database on Demand service. With this objective, some major points of interest are: the current status and maturity of the Trove project, ability to support additional database types, and compatibility with Scientific Linux as the computing platform running most CERN services.

A secondary goal of the project is to evaluate different Java interfaces to OpenStack which would enable the service's software to interact with the OpenStack service to create and manage instances for the database servers.

Table of Contents

1 Introduction and Motivation

1.1 CERN Database on Demand Service

The CERN Database on Demand Service (DBOD) started operating in 2012 with the objective of providing the users with access to new databases and different service level agreements. Traditionally, CERN users had the possibility of requesting a user or project database on the centrally managed Oracle service, operated by the IT-DB group. Over time, though, there was a growing interest from users in having a central service offering MySQL and PostgreSQL databases, as well as Oracle databases with a broader access to the underlying database system than in the already existing Oracle service. From the user's perspective, DBOD now provides hosting of databases—managing the installation and upgrade procedures—and an easy to use web interface to perform database backups, recoveries, database configuration, and monitoring, see figure 1 on the following page. Since its start, DBOD has grown rapidly within CERN, now hosting 168 database instances, whereof 141 are MySQL, see figure 2 on the next page. The Database on Demand service is in itself a framework profiting from pre-existing IT-DB management tools and infrastructures, and which, from the point of view of the user, provides databases as a service (DBaaS).

1.1.1 Database on Demand Infrastructure

The DBOD service is designed and implemented in a way such that the nature of the underlying platform running the database systems is not relevant to the service and as a consequence of this, the current infrastructure operates on both physical hosts and virtual machines—a detailed description of the infrastructure can be found in section 2 on page 9, where DBOD and OpenStack Trove will be compared.

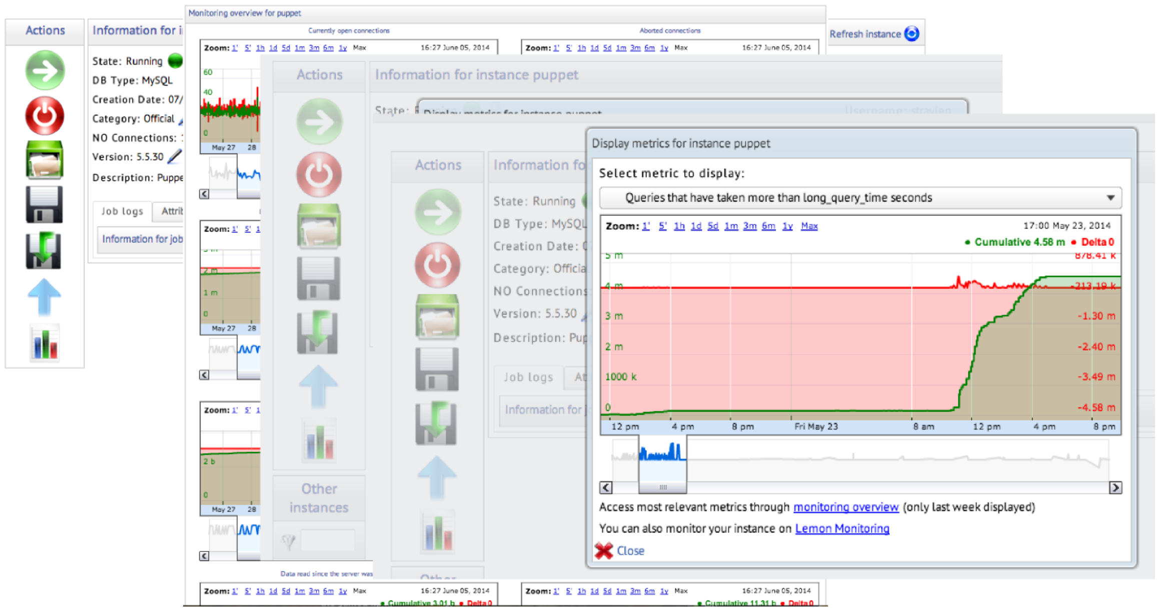
This point being made, virtualisation offers clear advantages from the point of view of consolidation and manageability of resources, and, as it will be covered in the next two subsections, plays a key role in the strategic plans for the IT department at CERN and the IT-DB group in particular.

The IT-DB group is on its final steps for completing their virtualisation platform migration towards an OpenStack based solution, and this fact affects the DBOD service, while also providing new opportunities for profiting from the OpenStack ecosystem. Hence the interest in evaluating the current status of the OpenStack DBaaS component, Trove, whose expected functionality overlaps in certain areas with the one DBOD offers.

1.2 OpenStack

OpenStack[1] is a free and open source cloud computing platform started in 2010 providing infrastructure as a service. This means offering tools to manage resources like virtual machines, servers, block and object storage and networks, distributing them automatically within a big pool of hardware resources. OpenStack has a vibrant community, including a lot of companies[2] using it for delivering their own services and contributing back to the project. On the technical side, OpenStack relies on a lot of other open source projects, making it a huge effort of integration. This is accomplished by designing OpenStack to be built of components, each of them providing its features to the other components and to the user via a REST API making it possible to use different technologies on the back end side. This modular architecture makes it possible to update an existing OpenStack cloud to the next release without downtime of the virtual machines[3]. The most common components

[1] https://www.openstack.org/
[2] https://www.openstack.org/foundation/companies/
[3] http://openstack-in-production.blogspot.fr/2014/02/our-cloud-in-havana.html


Monitoring overview for puppet
Actions
Information for instance puppet
Refresh instance
Other instances
Job logs
Currently open connections
Aborted connections

State: Running
DB Type: MySQL
Category: Official
Version: 5.5.30

Display metrics for instance puppet

Select metric to display: Queries that have taken more than long_query_time seconds

Zoom: 1' 5' 1h 1d 5d 1m 3m 6m 1y Max 17:00 May 23, 2014

● Cumulative 4.58 m ● Delta 0

Access most relevant metrics through monitoring overview (only last week displayed)

You can also monitor your instance on Lemon Monitoring

Close

Figure 1: Various screenshots of the Database on Demand web interface [1], showing the buttons to start and stop instances, execute backups and restores, perform pending upgrades and show monitoring information.

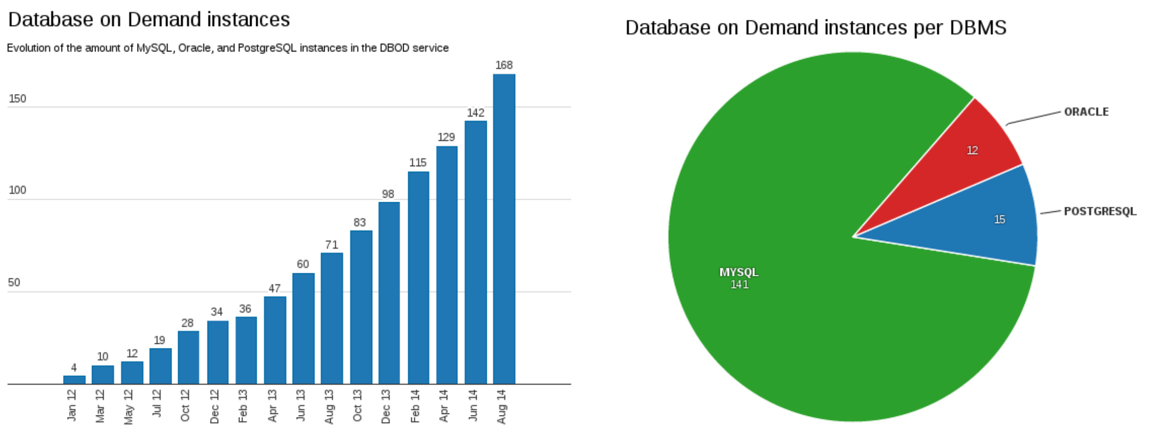


Figure 2: Statistics on the usage of the Database on Demand service. Left: Evolution of the overall amount of DBOD instances. Right: the current distribution of different database management systems

of OpenStack are shown in figure 3 on the next page, where they are represented by the task they fulfil. Besides that, they all have a name, which is usually used to refer to them. Figure 4 on the following page structures the components into layers making it easier to find a starting point. The OpenStack Dashboard, called *Horizon*, is an easy to use web interface to all components that can be used instead of the command line interfaces. It is built using the Python web framework Django. *Keystone* is OpenStack's common identity provider that is used by all components to authenticate users. *Nova*, the computing component, manages (virtual) machines and decides on which of the available nodes to boot them. Nova can use different hypervisors and virtualisation techniques to achieve this, including KVM and Docker. *Glance*, the image service, holds operating system images that are used by Nova. *Cinder* manages block storage, that is, volumes that can be attached to machines as virtual hard drives. Like Nova, Cinder can use different back ends. *Swift* can store files and objects. *Neutron* and *Nova Network* are the two components capable of managing software defined networks in OpenStack, Neutron being the more sophisticated one, based on OpenVSwitch. There are new components coming up, after a time of incubation being officially integrated into the project. This is how *Trove*, previously called *Reddwarf* became a component of OpenStack. It will be described in section 1.4.

1.3 OpenStack at CERN

OpenStack is used at CERN and its experiments at four different places at least. There is the general CERN Private Cloud, then an IT-DB internal OpenStack cloud, and furthermore ATLAS and CMS using OpenStack to deploy a computing grid on their trigger farms to use the computing power for analysis and simulation during detector shut down and maintenance periods [5].

Within the CERN Private Cloud, every group at CERN and every experiment connected to CERN can request resources and then deploy their own operating system and software packages on instances. The IT department chose OpenStack as a building block for the internal cloud because of multiple reasons: A growing number of computing resources has to be managed, with the goal to virtualise resources as much as possible and having the possibility to deploy this cloud on multiple sites, as CERN just got a new data centre in Budapest, Hungary. OpenStack as a flexible and open source cloud platform with a big community was a very suitable choice for this task. To integrate OpenStack into CERN's IT environment, some challenges had to be faced. The identity provider Keystone had to be integrated with the existing user management, making it possible to provide the OpenStack service to over 11 000 users, with 200 arrivals and departures every month [6]. In addition, to provide the instances with network connectivity, OpenStack had to be attached to CERN's own system of assigning IP addresses and hostnames, the LANDB. Insights into these experiences are provided on the *OpenStack in Production*[4] blog by four of the engineers working on the CERN Private Cloud.

The IT-DB group operates its own OpenStack cloud under the guidance and configuration of the central service because of specific configuration requirements demanding a second network interface on the virtual machines, and to profit of physical proximity to their storage servers. The ATLAS and CMS clouds use OpenStack to deploy grid middleware for connecting to the Worldwide LHC Computing Grid (WLCG) on their high level trigger systems [7].

1.4 OpenStack Trove: The Database Management Component

OpenStack Trove provides Databases as a Service to its users. It is built upon Nova and installs database management software inside the instances. Trove uses separate instances for different database management systems. The user can manage databases, backups and restores through

[4]`http://openstack-in-production.blogspot.fr/`

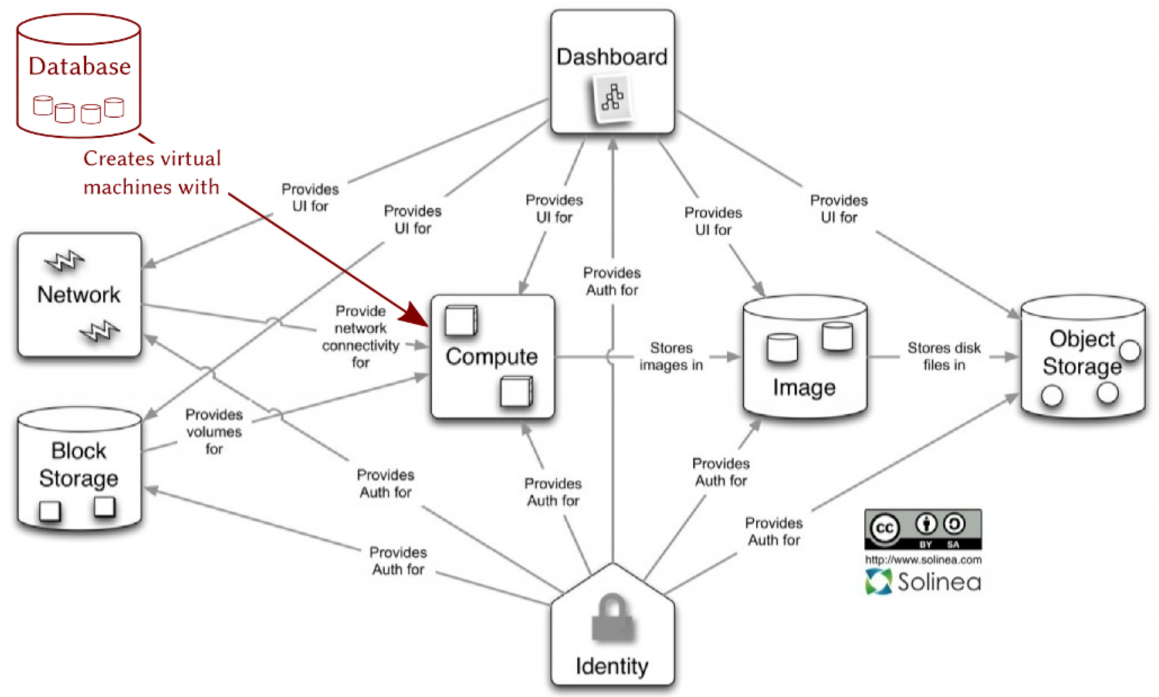


Figure 3: Logical overview of OpenStack's architecture, showing the most common components [2]

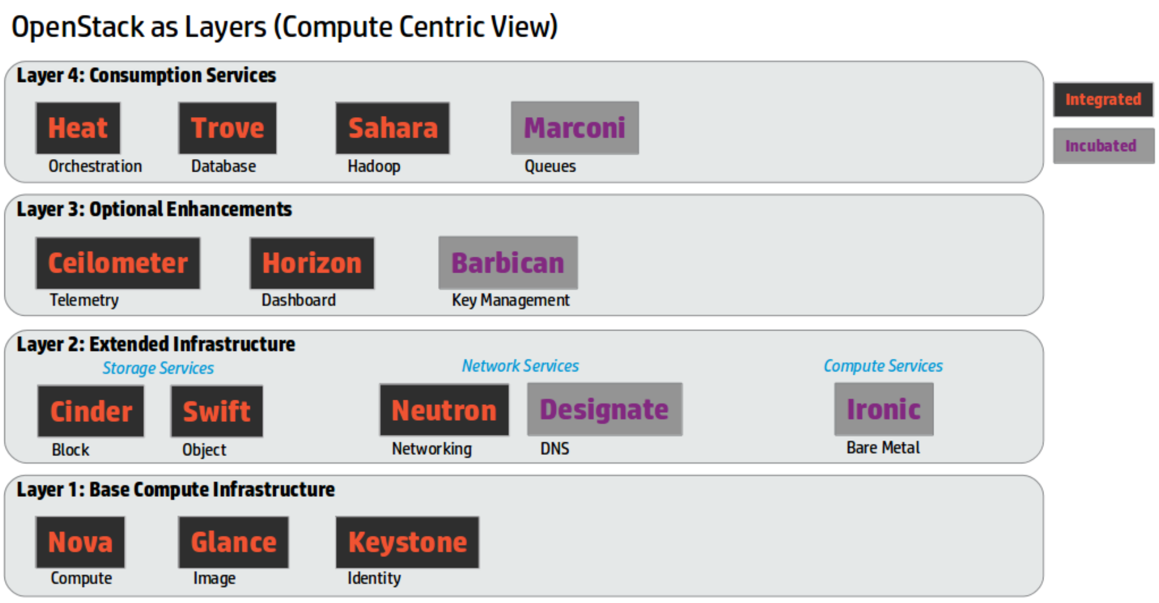
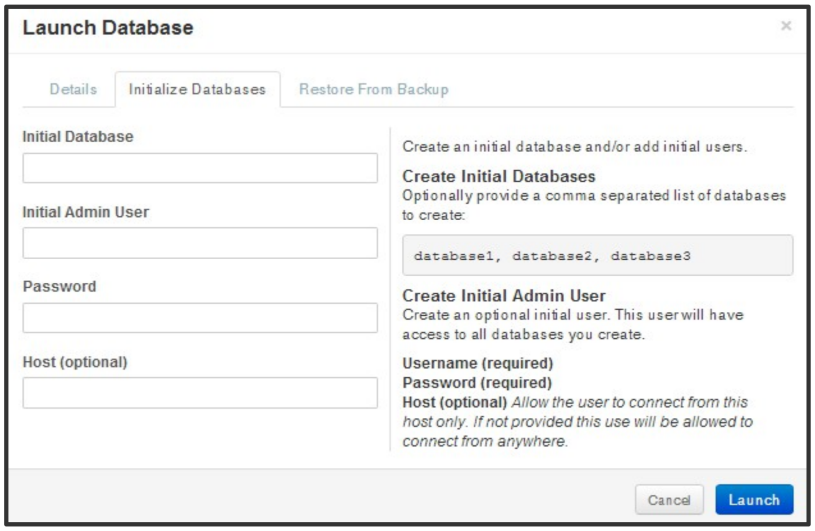


Figure 4: A more structured view to OpenStack, dividing the components into layers. This diagram includes incubated but not yet integrated components as well. Proposed by Dean Troyer [3] and visualised by Sean Dague [4]



Initial Database

Initial Admin User

Password

Host (optional)

Create an initial database and/or add initial users.

Create Initial Databases
Optionally provide a comma separated list of databases to create:

```
database1, database2, database3
```

Create Initial Admin User
Create an optional initial user. This user will have access to all databases you create.

Username (required)
Password (required)
Host (optional) *Allow the user to connect from this host only. If not provided this use will be allowed to connect from anywhere.*

Figure 5: View of OpenStack's dashboard: Create a new database [8]

OpenStack's Dashboard or the command line interface. Figure 5 shows the Dashboard during the launch of a new database. A deeper insight into Trove's architecture will be given in section 2. Currently the database management systems Cassandra, Couchbase, MongoDB, MySQL, PostgreSQL and Redis are supported.

Trove was started in 2011 at Rackspace as *Reddwarf*. It was incubated into OpenStack with release Havana and integrated with this year's release Icehouse. Like all of OpenStack, it is written in Python. Contributors to Trove are mainly the five companies Rackspace, HP, Mirantis, Tesora and eBay[5], which at the same time are the main companies using Trove in production either to offer OpenStack based products or for internal use. Current work on the Trove project is done on the integration of more database types, the completion of the architecture and on supporting replication. The plans and development can be followed on Trove's blueprints page on Launchpad[6].

2 Architectural Comparison of Trove and CERN Database on Demand

In this section, the architecture of DBOD will be illustrated. Then, the architecture of Trove will be described referencing to DBOD to point out similarities and differences. Figures 6 and 7 on page 11 support this description. Not all elements of DBOD shown in the figure will be described in detail though. The details of Trove's architecture are taken from the Trove wiki [9].

For a DBOD user, there are three different interaction points with the service. The CERN Federated Identity Management (FIM) is only needed at the beginning—it is used to keep track of resource owners within CERN. There, the user has to fill out a form to request a database. After the instance is ready, the web interface can be used to perform basic management tasks on it: start and stop it, initiate backup and restore operations, confirm software updates, change the database configuration and access monitoring information. The web interface is written in Java.

[5]`http://stackalytics.com/?user_id=&project_type=all&release=all&metric=all&company=&module=trove`

[6]`https://blueprints.launchpad.net/trove/`

To connect to the database, the user is provided with hostname, port and credentials. The database user receives full database administrator privileges. Tasks commissioned by the user via the web interface are stored in the DBOD database (DBOD DB). The daemon frequently polls the database and executes the tasks via Syscontrol—a management infrastructure developed inside CERN, which is replaced by Puppet more and more. Updates on the tasks are written back to the database by the daemon. Different monitoring solutions are in place: Oracle EM, Racmon and the CERN Agile Infrastructure monitoring. A custom component in the monitoring system is the DBOD state checker which frequently connects to the database instances to test their availability. The results are stored in the DBOD DB, so the web interface can display them.

The instances themselves run on Oracle virtual machines or physical hosts, with an ongoing migration to virtual machines within the IT-DB internal OpenStack cloud. The virtual machines host multiple database instances listening on different ports, and are not limited to one type of database management system. The data storage is done separately from these hosts on network attached storage. This enables a sophisticated backup solution based on snapshots on file system level instead of with SQL dumps.

The architecture of Trove is not yet fixed. What is shown in figure 7 is almost the current architecture, only the scheduler is not yet implemented. The entry point for Trove is the Trove API which can be accessed by the command line interface or the Dashboard. The API corresponds to DBOD's web interface. Like with DBOD, the Trove database (Trove DB) is used to store status information about instances and backups. The Trove API server directly executes synchronous tasks, like getting a list of database users from the Guestagents, and delegates asynchronous task to the Taskmanager—Trove's equivalent to DBOD's daemon. Asynchronous tasks are for example the provisioning or resizing of an instance and the initiation of a backup. Thus, the Taskmanager mostly talks to other OpenStack APIs like Nova and Cinder. In the future, the Scheduler will manage recurrent tasks like scheduled backups. In DBOD, recurrent backups are managed by the web interface as well. The Guestagents run inside the instances and are responsible for the databases, called datastore in Troves terminology. They create schemas and users and execute backups. Furthermore, the Guestagents sends a heartbeat to the Conductor.

The Guestagent and the database software need to be included in the image used for the instance. There are two possibilities to provide these images: either an image including this software and the correct configuration is prepared, or one basic image is used for all database types but a different CloudInit configuration is used to provision the instances after boot. CloudInit is a tool with a simple configuration format that can install packages, change files and basically execute arbitrary commands on the boot of a new instance. It is available for a lot of Linux distributions. CloudInit needs to be installed on the instance already, though. Images with CloudInit already included are often called "cloud enabled"—a lot of Linux distribution vendors provide cloud enabled images.

The Conductor receives the heartbeats and status updates about backups and stores them in the Trove DB. In DBOD, there is no such service as the Guestagents or the Conductor. Database users and schemas are created via Syscontrol and Puppet, thus by the daemon. The heartbeats are implemented the other way around with the state checker, providing the same result.

Trove uses Cinder to attach volumes to the instances so the data can be stored on them. This makes it possible to use different backup solutions, as well. A major difference between DBOD and Trove is that Trove uses separate virtual machines for different database types and only installs one database server per virtual machine. If resources need to be consolidated more with Trove, a solution is to use Docker as virtualisation technique, see section 5.2 on page 27.

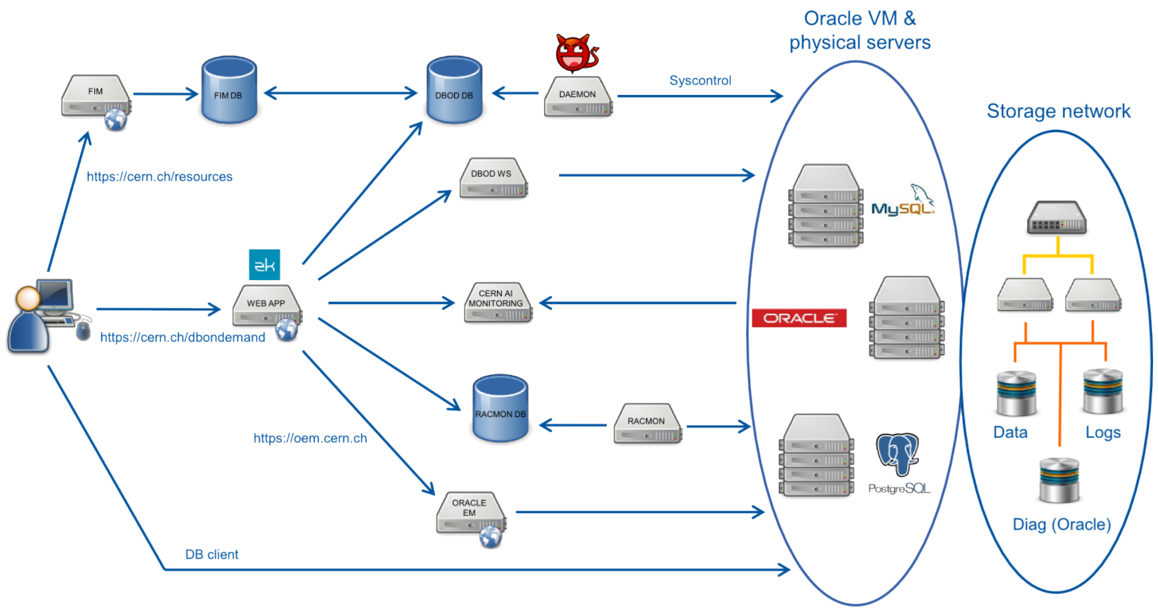


Figure 6: Architecture of the CERN Database on Demand service [1]

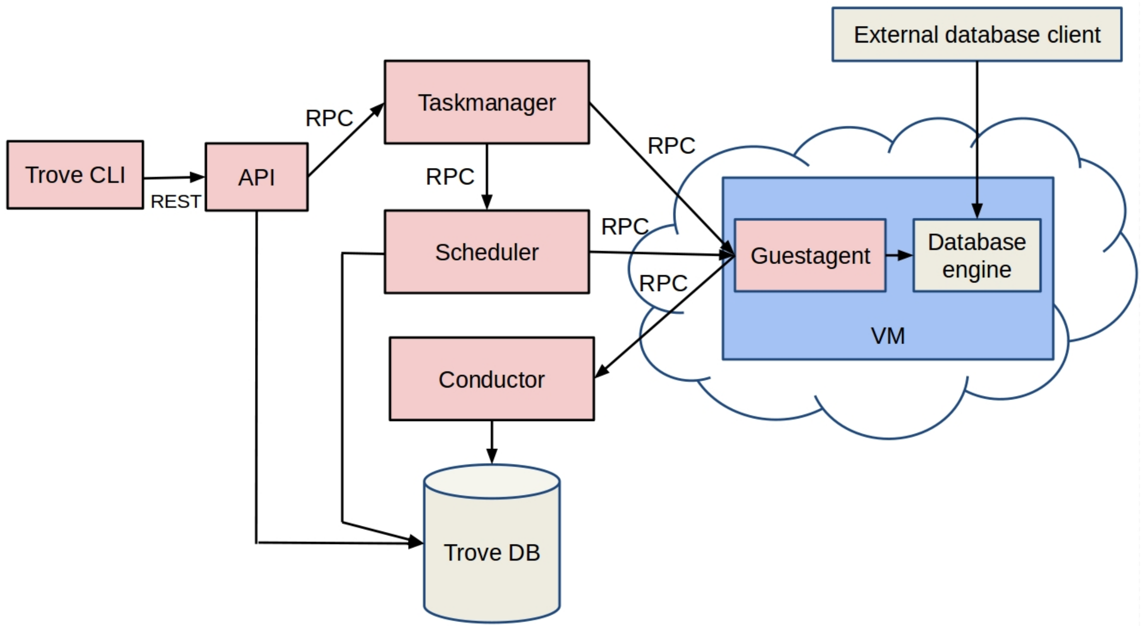


Figure 7: Architecture of OpenStack Trove. By courtesy of Denis Makogon, Mirantis [10]

3 Installation of OpenStack and OpenStack Trove

This sections covers the process of installing and configuring OpenStack and Trove on Scientific Linux, which was the main work during the project. Before starting, an overview of the ways to deploy OpenStack and to connect to OpenStack's community is given.

3.1 Ways to Deploy OpenStack

There are different possibilities to setup a new OpenStack cloud, the choice depending on the goal of the project.

Manual setup For the deployment of a new production cloud, this might be the best solution. Of course, tools like Puppet can be used to automatise this process, but for real-world production deployments, full control over all configuration is needed.

Packstack Packstack is part of Red Hat's effort of building *RDO*, a Red Hat distribution of OpenStack. This basically covers the packaging and documentation work for Red Hat based Linux distributions. The project's website is openstack.redhat.com. Since Scientific Linux is derived from Red Hat Enterprise Linux, it was advisable to use Packstack for this project.

DevStack DevStack is a tool that can be used to deploy a development environment of OpenStack on a local machine. Although it is in theory compatible with Red Hat based Linux distributions as well, it is only well tested on Ubuntu. A short glimpse on the use of DevStack will be given in section 3.5 on page 23. Section 5.1.1 on page 27 makes a recommendation on which one of Packstack and DevStack to use for a future project like this.

Crowbar Another tool that can be used to deploy OpenStack, but cannot be covered in this report, crowbar.github.io.

3.2 Diving into OpenStack or: Finding Help in the Community

OpenStack is an amazing project with a vibrant community. But for newcomers, it might be a challenge to find the right place to start, as there are a lot of guides and tutorials, for a lot of different aspects and versions of OpenStack, and a lot of mailing lists, forums and IRC channels. Thus, this section intends to present some hints on starting with OpenStack, focused on Trove and Packstack.

Documentation The main documentation of *OpenStack* is published on docs.openstack.org. There are different guides for different purposes: Installation guides, API references, an operation guide, an architecture design guide and some more for people setting up and deploying OpenStack. For users, there is an end user guide and an admin user guide. Thus, the first step is to find the most suitable documentation for the project. For this project, the most used documentation was the installation guide for Red Hat based Linux distributions[7]. Sometimes it is good to compare what the different documentations recommend, as there are many possibilities to configure OpenStack. For users of the CERN Private Cloud, there is the CERN Cloud Infrastructure Guide[8] as a really comprehensive starting point relating to the computing component Nova which mostly applies to general OpenStack

[7] `http://docs.openstack.org/icehouse/install-guide/install/yum/content/`

[8] `http://information-technology.web.cern.ch/book/cern-private-cloud-user-guide`

setups outside CERN as well. As a quick overview, the slides of the CERN OpenStack Training can be helpful[9].

The documentation of *Trove* is on a very early stage, which means that currently there is only a very basic install and usage guide available. As it will be clear later in section 3.3 on the following page, the documentation for manual installs is incomplete at the moment. The most detailed description of commands is still only the short help text included in the commend line tools.

Wiki The OpenStack Wiki is located at wiki.openstack.org. The most important pages might be How to contribute and the other general guides linked on the main page. The pages of the components, like the ones on Trove are used to track progress in development and to keep meeting notes. Single pages can be heavily outdated. *RDO with Packstack* has its own wiki at openstack.redhat.com with a *Get Involved* page.

Code, Bugs, Blueprints and Reviews All projects under OpenStack's umbrella have their source code under github.com/openstack-dev, github.com/openstack or github.com/stackforge. For bug reports and blueprints, Launchpad is used. Code changes are not applied via pull requests but via the Gerrit code review tool available at review.openstack.org.

Mailing lists There are a lot of mailing lists, all of them being covered on the corresponding page in OpenStack's Wiki and Packstack's Wiki. The mailing lists `openstack`, `openstack-announce`, `rdo-newsletter` and `rdo-list` are definitely the most important for a project on Scientific Linux. `openstack-docs` is good to see what currently happens on the part of the documentation. The list `openstack-security` is interesting but the topic was clearly out of scope of this project. As Trove is a very young project, there are not many people able to provide help on the mailing lists.

Forums The main forum for OpenStack is ask.openstack.org. It follows the modus of the sites of the Stack Exchange Network, thus it is based on questions and answers that can be up voted and edited. Questions can be tagged, so an interesting feed to subscribe to is the one for Trove. As people are asking questions on Stackoverflow as well, the feed for questions tagged with `openstack` is worth a look. But as with the mailing lists, the forums are not yet a good place to get answers concerning Trove. Better are the IRC channels.

IRC channels The list of all IRC channels operated by OpenStack is available in the wiki. The relevant channels for this project were `#openstack`, `#openstack-trove`, `#packstack-dev` and `#rdo`. On `#openstack-trove`, the developers are active and reply really fast and helpful to questions.

Blogs Announcements and explanations of new features are often published in blogs. A lot of blogs on OpenStack are collected at planet.openstack.org, so it might be good to subscribe and filter the interesting posts from time to time.

Slides The companies working on Trove typically present their most recent work on conferences, publishing the slides on slideshare.net afterwards.

[9]`http://information-technology.web.cern.ch/sites/information-technology.web.cern.ch/files/OpenStack_Training_June_2014.pdf`

3.3 Installation of OpenStack

The installation of OpenStack was done using Packstack following a step-by-step guide in the RDO wiki [11]. This and the configuration of the virtual networks, and later the installation and configuration of Trove was put into a Bash script.

3.3.1 Installation Script

This sections, *Installation of OpenStack*, and the next section *Installation of Trove* will describe how the installation was done which is at the same time an explanation of parts of the script. The script is available on Github[10]. It includes directions of how to configure and use it. The git repository includes an archive of the step-by-step guide [11] as well. The script basically consists of Bash functions that are meant to be executed in the order in which they are defined. Each of the functions performs something that can be called *one step* of the installation. The functions can be called by executing the script with one parameter being just the name of the function. If the script is executed without parameters, it will do nothing. This is intended, as the script is not well-tested enough to be just executed. Thus, the code of the script actually has to be read to see the names of the functions. The installation was put into a Bash script and not into Puppet, because with Bash, parts of it just can be copied to a shell and be executed.

3.3.2 Installation Process

Packstack makes heavy use of Puppet to install and configure all the components. The configuration of Packstack can be done via command line parameters for some simple options or via a so-called answer file. This configuration file is called answer file because if Packstack runs without any parameters, it asks the user questions to determine the configuration. On Scientific Linux CERN, some workarounds were needed to run Packstack without issues. They are available in the CERN cloud infrastructure user guide[11].

First of all, repositories had to be setup. The first needed workaround was to fix the priorities, so the right packages will be installed:

```
yum install -y http://rdo.fedorapeople.org/rdo-release.rpm
sed -i -e 's/priority.*/priority=1/g' /etc/yum.repos.d/rdo-release.repo
sed --in-place '/^exclude=libmongodb/d;s/^priority=/exclude=libmongodb,
    pymongo\*,mongodb\*,python-bson,python-webob,python-mako,python-
    webtest\npriority=/g' /etc/yum.repos.d/slc6-os.repo /etc/yum.repos.d/
    slc6-updates.repo /etc/yum.repos.d/slc6-extras.repo
```

Packstack uses SSH to connect to the hosts it should setup, as it is able to deploy OpenStack on multiple nodes. At the moment of the installation, it had problems to add the SSH key correctly to `authorized_keys`, so this had to be done manually before:

```
ssh-keygen -q -f /root/.ssh/id_rsa -t rsa -P ""
sh -c 'cat /root/.ssh/id_rsa.pub >> /root/.ssh/authorized_keys'
chmod 700 .ssh
chmod 600 .ssh/*
```

Then, Packstack itself could be installed. The next two workarounds concerned the MySQL Puppet module and SELinux. Normally, SELinux should not be muted and instead the issues with the

[10] https://github.com/blipp/openlab-openstack-trove-setup

[11] http://information-technology.web.cern.ch/book/cern-cloud-infrastructure-user-guide/advanced-topics/installing-openstack#icehouse

software be fixed[12]. But for this evaluation project, this was out of scope.

```
yum install -y openstack-packstack
sed -i'' '3 s/^/#/' /usr/share/openstack-puppet/modules/packstack/
    templates/innodb.cnf.erb
sed -i -e 's/^SELINUX=enforcing/SELINUX=permissive/g' /etc/selinux/config
```

Finally, Packstack could be called. The switch `--allinone` means that all services will be installed on the machine on which Packstack is called. `--provision-demo` and `--provision-all-in-one-ovs-bridge` stop Packstack from creating a demo user and network. Nagios had to be disabled because of an installation issue—it was not important for this test setup anyway. Furthermore, the networking component Neutron was chosen with `os-neutron-install`.

```
packstack --allinone --os-neutron-install=y --provision-demo=n
     --provision-all-in-one-ovs-bridge=n --nagios-install=n
```

If any issue occurs during the run of Packstack, log files will be provided so it can be resolved manually. Then, Packstack has to be called again, but with the answer file it produced:

```
packstack --answer-file=/root/packstack-answers-20140811-101751.txt
```

One problem that occurred during this setup was the `mongodb-server` package missing in Red Hat's repositories on that exact day—it was removed because of a security issue. An older version of the package could be installed manually.

After the installation has finished successfully, Packstack will tell where it created the RC file for the OpenStack admin user and show the URL to the dashboard. Packstack will show a notification if a reboot is necessary because of a kernel update. The RC file is used to define environment variables holding user name, password, tenant and the URL to which to connect. This file is meant to be sourced in a terminal before using the command line clients. Otherwise, these details need to be specified on each call via parameters. A tenant in OpenStack basically means *project*, a term that is used in the dashboard. A user can be associated with more than one tenant, but needs one RC file for each tenant—it would be sufficient to update the environment variables `OS_TENANT_ID` and `OS_TENANT_NAME` holding the tenant to switch them though. The RC files can be downloaded from the dashboard. Another practical feature of them is that they change the shell prompt so it is visible which user and tenant currently is in use. For security reasons, the files can be adapted to ask for the password instead of including it in plain text. In the installation script, two RC files for the admin and a regular user are sourced depending on the task.

3.3.3 Network

OpenStack users can configure private networks for their instances. For connectivity to external networks, so-called floating IP addresses need to be assigned to the instances. The according public networks and routers need to be configured by the OpenStack administrator. To provide a firewall feature and the ability to span the networks over multiple OpenStack nodes, Neutron uses rather a lot of networking layers. This architecture is complex and cannot be covered in this report. The already mentioned tutorial [11], *Networking in too much detail*[13] and *There's Real Magic behind OpenStack Neutron*[14] provide an overview backed by helpful visualisations.

A major challenge of the installation was the restricted external network. At CERN, each machine has to be registered in the network service, the LANDB. For a 9 weeks test setup, it is not possible to get

[12] http://securityblog.org/2006/05/21/software-not-working-disable-selinux/

[13] https://openstack.redhat.com/Networking_in_too_much_detail

[14] http://pinrojas.com/2014/07/29/theres-real-magic-behind-openstack-neutron/

an integration of the instances into it. Thus, another solution had to be found to achieve connectivity to the internet. The original plan to deploy OpenStack on two nodes was given up because it turned out to be too complicated within the restricted network. For a future attempt on this, the blog posts of Boris Derzhavets might be helpful as he has written a lot of networking tutorials for multi node setups[15].

The external connectivity of the instances was solved by creating a NAT inside the all in one host. This way, the instances can be fully reached from within the host, which is absolutely sufficient for this project—see the requirements of a Trove installation[16]. For connections from outside, port forwarding could be set up. The external network was chosen to be `10.0.21.0/24`. The creation of the network and the router inside Neutron had to be executed as the admin user. The allocation pool is the range of addresses that can be associated with the instances. The virtual router has to be set as gateway for this network:

```
. $RC_ADMIN
neutron net-create extnet --router:external=True | log_output extnet
NETWORK_ID=$(get_field extnet " id " 4)
# Create Subnet
neutron subnet-create extnet --allocation-pool start=10.0.21.10,end
    =10.0.21.125 --gateway 10.0.21.1 --enable_dhcp=False 10.0.21.0/24 |
    log_output extnet_subnet
# Create Router
neutron router-create rdorouter | log_output rdorouter
ROUTER_ID=$(get_field rdorouter " id " 4)
# Set the Router's Gateway
neutron router-gateway-set $ROUTER_ID $NETWORK_ID | log_output gateway-set
```

The actual NAT was created by the following script. It needs to be executed after each reboot of the host. `br-ex` is the network interface to which Neutron's public networks are attached.

```
#!/bin/bash
ip link set down br-ex
ip addr add 10.0.21.1/24 dev br-ex
ip link set up br-ex
iptables -I FORWARD -i br-ex -j ACCEPT
iptables -I FORWARD -o br-ex -j ACCEPT
iptables -t nat -I POSTROUTING -s 10.0.21.0/24 ! -d 10.0.21.0/24 -j
    MASQUERADE
```

Then, user and tenant `rdotest` without special privileges were created. This was done to separate privileges and to have a more or less standard OpenStack setup where users with standard privileges use the resources provided by the administrators.

```
keystone tenant-create --name rdotest | log_output tenant-create
TENANT_ID=$(get_field tenant-create " id " 4)
TENANT_PASS=$(pwgen 15 1)
echo "rdotest" >> $PASSWORD_FILE
echo $TENANT_PASS >> $PASSWORD_FILE
keystone user-create --name rdotest --tenant-id $TENANT_ID --pass
    $TENANT_PASS --enabled true | log_output user-create
```

The new user can now create a private network for the instances. The IP range `10.0.90.0/24` was chosen. Here, the important point is to provide the IP addresses of CERN's DNS servers, since no

[15]http://bderzhavets.blogspot.ch/2014/07/rdo-setup-two-real-node_29.html

[16]http://docs.openstack.org/developer/trove/dev/manual_install.html

other DNS servers are allowed inside CERN's network. The instances need working DNS to install software from package repositories. Furthermore, the private network needs to be attached to the previously created router.

```
. $RC_RDOTEST
neutron net-create rdonet | log_output net-create-rdonet
neutron subnet-create --dns-nameserver 137.138.17.5 --dns-nameserver
    137.138.16.5 rdonet 10.0.90.0/24 | log_output subnet-create-rdonet
SUBNET_ID=$(get_field subnet-create-rdonet " id " 4)
. $RC_ADMIN
ROUTER_ID=$(get_field rdorouter " id " 4)
neutron router-interface-add $ROUTER_ID $SUBNET_ID | log_output router-
    interface-add
```

3.3.4 Creation of Instances

As next step, an image was uploaded to Glance. For tests, CirrOS is a nice operating system, as it is very lightweight.

```
. $RC_ADMIN
glance image-create --container-format=bare --disk-format=qcow2 --name=
    cirros --is-public=True < $CIRROS_IMAGE_FILE | log_output image-create
```

To make it possible to connect to instances via SSH, OpenStack supports associating SSH keys to an user account. In OpenStack, they are called key pairs. When creating a new instance, a key pair can be chosen and then the according public key will be added to `authorized_keys` inside the machine. Security groups in OpenStack implement a firewall. Instances are associated to a certain security group. Rules can be defined within a security group to allow certain traffic. Here, ICMP and SSH were enabled:

```
. $RC_RDOTEST
nova keypair-add --pub-key /root/.ssh/id_rsa.pub rdokey | log_output
    keypair-add
neutron security-group-rule-create --protocol icmp --direction ingress
    default | log_output security-group-rule-icmp
neutron security-group-rule-create --protocol tcp --port-range-min 22 --
    port-range-max 22 --direction ingress default | log_output security-
    group-rule-ssh
```

Finally, an instance could be booted using the `nova` command, defining flavor, image and key pair. Flavors in OpenStack refer to the resources given to an instance. Administrators can define flavors and which users are allowed to use them.

```
. $RC_RDOTEST
IMAGE_ID=$(get_field image-create " id " 4)
nova boot --flavor 1 --image $IMAGE_ID --key-name rdokey "cirros$1" |
    log_output "boot-cirros$1"
```

The association of a floating IP follows the following workflow. An address has to be chosen and then associated to the port in the private network the instance is attached to. Thus, the ID of the port has to be found:

```
. $RC_RDOTEST
VM_ID=$(get_field "boot-cirros$1" " id " 4)
neutron port-list --device_id $VM_ID | log_output "port-list-cirros$1"
PORT_ID=$(get_field "port-list-cirros$1" "subnet_id" 2)
```

```
neutron floatingip-create extnet | log_output "floatingip-create-cirros$1"
FLOATINGIP_ID=$(get_field "floatingip-create-cirros$1" " id " 4)
neutron floatingip-associate $FLOATINGIP_ID $PORT_ID
```

With this setup, instances could reach other instances, the host and the internet. Unfortunately, the host could not reach the instances although the NAT in principle should work. The reason has not been found. Means to debug the network setup are `tcpdump` and the `libvirt` commands as described in [11]. To reach the instances anyway, a wrapper script around `ip netns exec` was written:

```
ip netns exec qrouter-8f9c4e18-2306-41c6-8f34-03fee14c7aeb $@
```

All network namespaces created by Neutron can be listed with `ip netns`. If there is only one private and public network, there will be only one `qrouter` and one `qdhcp` namespace—either of them can be chosen for the script. With this script it was possible to connect to the instances via SSH using `ipns ssh cirros@10.0.90.4`.

3.4 Installation of Trove

For the installation and configuration of Trove, the according guide in OpenStack's documentation was followed [12]. Some hints were taken from DevStack's installation script for Trove[17]. The installation was first done via the package manager:

```
yum install -y openstack-trove python-troveclient
```

But for an evaluation, the most recent code is a better choice since this makes it easier to find help. The upstream code, however, does not include init files for the services, as this depends on the Linux distribution. Thus, the init files were kept—actually the manual installation was just made on top of the installation from the package repositories. The only thing to pay attention to was automatic package updates that could overwrite the changes made behind the package manager's back. The installation from the Github sources is not yet integrated in the installation script but only documented here. The guide followed for this was rather outdated [13].

First of all, the Python package manager `pip` had to be updated. The second command indeed had to be run twice[18].

```
pip install -U pip
pip install -U setuptools
pip install -U setuptools
```

Then, the repositories of `trove` and the `python-troveclient` needed to be checked out.

```
git clone https://github.com/openstack/trove.git
git clone https://github.com/openstack/python-troveclient.git
```

The most recent release of Trove was `2014.2.b2`, thus the according tag was checked out in the repository.

```
cd trove
git checkout tags/2014.2.b2
```

The installation has to compile some packages, so a compiler and some libraries had to be installed:

```
yum groupinstall "Development Tools"
yum install libxml2-devel libxslt-devel
```

Then, the requirements of Trove and Trove itself were installed with `pip` and the setup script using

[17]http://devstack.org/lib/trove.html
[18]https://stackoverflow.com/a/25288078/3910830

```
pip install -r requirements.txt
python setup.py install
```

The output of the installation of the requirements had to be watched carefully: If some Python packages are updated to a version that is not compatible with the already installed OpenStack components, problems might arise. The script `setup.py` installed Trove's binaries to `/usr/bin`. The same procedure applied to `python-troveclient`.

```
pip install -r requirements.txt
python setup.py install
```

All OpenStack services were then restarted. However, Nova and Cinder did not start anymore. The error message suggested a problem with the Python package `kombu`. During the installation, it was upgraded to version 3, but Nova, Cinder and Trove only require a version greater than or equal to 2.4.8. After installing this exact version with

```
pip install kombu==2.4.8
```

the services came up again. Because Trove was previously installed via the package manager and already configured and started, the Trove DB was filled with data which was not compatible with the new version. Trove showed the error message `NoSuchTableError: `capabilities``. To resolve this, the function `remove_configuration_trove` in the script was called. This function stops all Trove services, removes all configuration files and the Trove DB. After this, the configuration was re-created with the function `configure_trove`, which is the topic of the next subsection.

3.4.1 Configuration

The configuration of Trove included creating an OpenStack user and tenant with certain privileges, configuring the Trove services, creating an image, configuring datastores within Trove and finally letting Keystone know about the new service.

The OpenStack services all run as a user named after them. These service users need to be members and administrators of the services tenant, which is usually named `services` in an installation done with Packstack, but `service` when following the official OpenStack documentation from the beginning. Thus, the user `trove` was created and assigned to the tenant `services` as `member` and `admin`.

```
keystone user-create --name=trove --pass=$TROVE_PASS --email=
    trove@localhost --tenant=services | log_output user-create-trove-
    services
keystone user-role-add --user=trove --tenant=services --role=admin |
    log_output user-role-add-trove-services
```

At the first try to run Trove, there always was the error message `ERROR: Unauthorized (HTTP 401)` coming up, regardless of which Trove command was executed. The first idea was, that the user does not have the needed privileges and thus, different possibilities were tried. In the end it turned out that not the user credentials or group membership was wrong but the wrong configuration option was used to define the user, see the question on ask.openstack [19] and the first bug report that was filed against the documentation [20].

Then, the configuration files for the different Trove services—the API service, the Taskmanager, Conductor and Guestagent—had to be written. There are the files `api-paste.ini` configuring the API itself, `trove.conf` for the API server, and `trove-taskmanager.conf`, `trove-`

[19] https://ask.openstack.org/en/question/45818/trove-trove-list-results-in-error-unauthorized-http-401/
[20] https://bugs.launchpad.net/openstack-manuals/+bug/1369116

`conductor.conf` and `trove-guestagent.conf` for the Trove services. To write the configuration options, `openstack-config` was used. This is a command line tool able to read and write `ini` files making it easy to use inside scripts. The OpenStack community seems to move to `crudini`[21] more and more though, because it supports a broader range of ini file syntax. Only some of the configuration options will be explained here. All configuration options being set can be seen in the script, and the meaning of them can be found in the OpenStack configuration reference[22].

First of all, enhanced logging was enabled for all Trove services to simplify debugging:

```
for config_file in trove.conf trove-taskmanager.conf trove-conductor.conf
    trove-guestagent.conf; do
  openstack-config --set /etc/trove/$config_file DEFAULT verbose True
  openstack-config --set /etc/trove/$config_file DEFAULT debug True
done
```

As from now, to save space, not the commands to set the configuration options but snippets from the configuration files will be shown. To keep it general, there will be still Bash variables in them, so the snippets are actually no valid configuration files.

There are different places where credentials had to be defined. In `api-paste.ini`, this was in a section `[filter:authtoken]` and in all the others within `[keystone_authtoken]`:

```
[keystone_authtoken]
auth_uri = http://$HOST_IP:35357/
identity_uri = http://$HOST_IP:35357/
admin_password = $TROVE_PASS
admin_user = trove
admin_tenant_name = services
```

The Trove Taskmanager is talking to Nova to execute tasks and thus needs credentials granting this access. The documentation suggests using the credentials of the OpenStack admin user. This might be too many privileges for the task, but of course worked:

```
[DEFAULT]
nova_proxy_admin_user = admin
nova_proxy_admin_pass = $ADMIN_PASS
nova_proxy_admin_tenant_name = services
```

Trove uses a message broker for communication between the Trove services. The host of the broker and the password had to be configured in each service configuration file:

```
[DEFAULT]
rabbit_host = $HOST_IP
rabbit_password = guest
```

`trove.conf`, `trove-taskmanager.conf` and `trove-conductor.conf` needed some basic settings like the URLs to connect to the other OpenStack services and the connection parameters for the Trove DB.

```
[DEFAULT]
trove_auth_url = http://$HOST_IP:5000/v2.0
nova_compute_url = http://$HOST_IP:8774/v2
cinder_url = http://$HOST_IP:8776/v1
swift_url = http://$HOST_IP:8080/v1/AUTH_
sql_connection = mysql://trove:$TROVE_MYSQL_PASS@$HOST_IP/trove
```

[21] `http://www.pixelbeat.org/programs/crudini/`
[22] `http://docs.openstack.org/icehouse/config-reference/content/ch_configuring-trove.html`

As already mentioned in section 2 on page 9 about the architecture of Trove, the images can be provisioned with CloudInit. The location of the configuration files for CloudInit had to be specified in the Taskmanager config. The second line was necessary to avoid the Taskmanager from crashing on startup[23].

```
[DEFAULT]
cloudinit_location = /etc/trove/cloudinit
taskmanager_manager = trove.taskmanager.manager.Manager
```

These were the most important configuration changes. After this, the Trove DB could be created. The file `config/trove.sql` was taken from the installation manual [12] and the credentials adapted using `sed`:

```
sed -e "s/TROVE_DBPASS/$TROVE_MYSQL_PASS/g" config/trove.sql | mysql -u
    root -p$MYSQL_ROOT_PASSWD
su -s /bin/sh -c "trove-manage db_sync" trove
su -s /bin/sh -c "trove-manage datastore_update mysql ''" trove
```

3.4.2 Image Creation

For this project, it was chosen to provision the instances with CloudInit because this seemed to be easier than creating an image already containing the database management software and the Guestagent. CloudInit configuration files were created for an Ubuntu and a Fedora guest. They could never be tested, though, because both images did not boot. The issue was not related to Trove but to Nova.

The syntax of CloudInit files is based on YAML and very simple—a lot of examples are available in its documentation[24]. There even is a wiki page in the CERN wiki providing some CERN specific examples[25]. The following is the configuration file created for Ubuntu. The one for Fedora is available in the Github repository of the installation script.

```
#cloud-config
packages:
- trove-guestagent
- mysql-server-5.5

# config file for trove guestagent
write_files:
- path: /etc/trove/trove-guestagent.conf
  content: |
    rabbit_host = HOST_IP
    rabbit_password = guest
    nova_proxy_admin_user = admin
    nova_proxy_admin_pass = ADMIN_PASS
    nova_proxy_admin_tenant_name = trove
    trove_auth_url = http://HOST_IP:35357/v2.0
    control_exchange = trove

ssh_authorized_keys:
- SSH_KEY

```

[23] https://bugs.launchpad.net/openstack-manuals/+bug/1369119
[24] http://cloudinit.readthedocs.org/
[25] https://twiki.cern.ch/twiki/bin/view/LCG/CloudInit

```
# restart trove-guestagent as the config has been changed
runcmd:
- stop trove-guestagent
- start trove-guestagent
```

Internally, Trove will call `nova` and append the parameter `--user-data /etc/trove/cloudinit/mysql.cloudinit`. The place holders are set using `sed` again and the file then written to the right place—for the file name this means it has to follow the rule `<datastore>.cloudinit`:

```
# create the CloudInit config file
sed -e "s/HOST_IP/$HOST_IP/g" -e "s/ADMIN_PASS/$ADMIN_PASS/g" -e "s|
    SSH_KEY|$(cat /root/.ssh/id_rsa.pub)|g" config/mysql.cloudinit > /etc/
    trove/cloudinit/mysql.cloudinit
```

An Ubuntu image in `qcow2` format was then added to Glance, and a new datastore created in Trove using this image. When creating a new datastore, the type of the datastore has to be set, here `mysql`, then a name for the version of the datastore, `mysql-5.5`, then a name for the datastore, again `mysql`, the ID of the image, packages to be installed on the new instance, and a boolean value to indicate if the datastore should be active.

```
glance image-create --name trove_mysql_ubuntu --file $IMAGE_DIR/ubuntu.
    qcow2 --property hypervisor_type=qemu --disk-format qcow2 --container-
    format bare --is-public True --owner trove | log_output image-create-
    trove-ubuntu
UBUNTU_IMAGE_ID=$(get_field "image-create-trove-ubuntu" " id " 4)
trove-manage --config-file /etc/trove/trove.conf datastore_version_update
    mysql mysql-5.5 mysql $UBUNTU_IMAGE_ID mysql-server-5.5 1
```

The possibility to provide packages to be installed overlaps with the functionality of CloudInit. The type of the datastore advises Trove which module to use for the management of the instance. This makes it possible to have more than one datastore of the same type, for example a MySQL datastore running on an Ubuntu image and one running on Fedora.

Finally, the service could be announced to Keystone. An important point was to indicate the right region. Regions in OpenStack are a possibility to use the same infrastructure for example for the identity management but having two different sites with different services available to the user. The default region used by `python-keystoneclient` is `regionOne`, but Packstack is using `RegionOne` with upper case R. This was reported as a bug in the documentation[26].

```
# create the Trove service and endpoint in Keystone
keystone service-create --name=trove --type=database --description="
    OpenStack Database Service" | log_output service-create-trove
TROVE_SERVICE_ID=$(get_field "service-create-trove" " id " 4)
keystone endpoint-create --service-id=$TROVE_SERVICE_ID --publicurl=http
    ://$HOST_IP:8779/v1.0/%\(tenant_id\)s --internalurl=http://$HOST_IP
    :8779/v1.0/%\(tenant_id\)s --adminurl=http://$HOST_IP:8779/v1.0/%\(
    tenant_id\)s --region RegionOne | log_output endpoint-create-trove
```

At a last step, the Trove services were enabled on system startup. As the configuration of Trove was tried with different configuration options a lot of times, it was convenient to use `etckeeper` to keep track of the configuration files:

```
for i in api taskmanager conductor; do
        service openstack-trove-$i start
        chkconfig openstack-trove-$i on
```

[26]https://bugs.launchpad.net/openstack-manuals/+bug/1369329

```
done

etckeeper commit "Finished setting up Trove using script"
```

The usage of Trove will shortly be covered in section 3.6 on page 25.

3.5 Installation with DevStack

At the end of the project a short attempt was made to setup OpenStack and Trove with DevStack on an Ubuntu virtual machine. This was done because the installation and configuration on Scientific Linux could not be finished completely. DevStack however promises to setup a working development environment of OpenStack including Trove [14]. A successful installation with DevStack could be used to test some more of Trove's features even though not on Scientic Linux.

3.5.1 Prerequisites

VirtualBox was chosen as hypervisor, running on the personal workstation, and Ubuntu 14.04.1 Server as guest operating system. The virtual machine was created with 4 GB of RAM and 20 GB of disk space. During the installation of Ubuntu, only the SSH server was chosen as additional service that should be installed. The virtual machine was provided with internet access via a NAT configured by VirtualBox. To connect from the host to the virtual machine via SSH, an additional network interface of type "host-only adapter" was added and configured with a static IP address [15]. Thus, the virtual machine has a private IP address only reachable from the host that can as well be used to access the OpenStack Dashboard via a web browser.

3.5.2 Configuration and Installation

DevStack is executed as a normal user. To use DevStack, the code is checked out from Github via

```
git clone https://github.com/openstack-dev/devstack.git
```

DevStack is configured via a `local.conf` configuration file. The configuration file used for this setup is shown in listing 1 on the next page. DevStack was configured to use OpenStack's stable branch Icehouse. Additionally, Trove has to be enabled as it is disabled by default. DevStack usually is configured to use the same password for all services, which is easier for development but of course not suitable for production setups. For the configuration of the network, DevStack's single machine guide [16] was followed: The `HOST_IP` is the virtual machine's IP address within VirtualBox' NAT. The network `FIXED_RANGE` is the network range OpenStack should use for internal IP addresses. The network `FLOATING_RANGE` is configured to be the same as the network from which VirtualBox' host-only network assigns IP addresses. Thus, instances created and managed by OpenStack can be reached from the outside network and can reach the outside network themselves. `FLAT_INTERFACE` is the virtual machine's interface that is connected to the host-only network. The values for `HOST_IP` and `FLAT_INTERFACE` can be found by examining the output of `ifconfig` on the virtual machine.

After the configuration file is prepared, DevStack's setup routine is called via

```
$ ./stack.sh
```

This downloads the sources of all needed OpenStack components, installs them into `/opt/stack` and starts them.

Immediately after the run of `stack.sh`, OpenStack's Dashboard and `nova` and `trove` commands were functional. However, the Ubuntu image chosen by DevStack for Trove's instances was not able to boot. An instance with CirrOS could be launched successfully though. At this point, the further investigation of DevStack was stopped due to the lack of time. Despite the issue with Trove's image, Listing 1: `local.conf` config file for DevStack

```
[[local|localrc]]

# choose branch
KEYSTONE_BRANCH=stable/icehouse
NOVA_BRANCH=stable/icehouse
NEUTRON_BRANCH=stable/icehouse
SWIFT_BRANCH=stable/icehouse
GLANCE_BRANCH=stable/icehouse
CINDER_BRANCH=stable/icehouse
HEAT_BRANCH=stable/icehouse
TROVE_BRANCH=stable/icehouse
HORIZON_BRANCH=stable/icehouse

# enable Trove
ENABLED_SERVICES+=,trove,tr-api,tr-tmgr,tr-cond

# set credentials
ADMIN_PASSWORD=*********
DATABASE_PASSWORD=$ADMIN_PASSWORD
RABBIT_PASSWORD=$ADMIN_PASSWORD
SERVICE_PASSWORD=$ADMIN_PASSWORD
SERVICE_TOKEN=a682f596-76f3-11e3-b3b2-e716f9080d50

# logging
LOGDAYS=1
LOGFILE=$DEST/logs/stack.sh.log
SCREEN_LOGDIR=$DEST/logs/screen

# networking
HOST_IP=10.0.2.15
FIXED_RANGE=172.31.1.0/24
FLOATING_RANGE=192.168.56.0/25
FLAT_INTERFACE=eth1
```

DevStack seems to be a really good starting point for trying out OpenStack and learning its concepts. Within about half an hour, an installation with a working `trove` command could be set up.

3.6 Usage of Trove

New database instances can be created using the command line or the Dashboard. Regardless of the interface used, the name of the instance has to be defined as well as the ID of the Nova flavor to use, the size of the data volume, the name of the schema, the user credentials and finally the datastore version and type to use:

```
trove create mysql_instance_1 10 --size 2 --databases myDB --users userA:
    password --datastore_version 5.5 --datastore mysql
```

The command `trove list` then immediately shows the new instance in state `BUILD`. The command `nova list` shows a new entry as well, but with a different ID—the ID of a database instance and the ID of a virtual machine are different values. Trove creates a new security group per instance only allowing access to the database port. Thus, any other ports, like for SSH, have to be allowed manually.

Unfortunately, more features of Trove could not be tested because, as already mentioned, the images did not boot, and there was no time left to resolve this issue.

4 Evaluation of Java Interfaces to OpenStack

Only a very short look was taken at different Java interfaces that expose the OpenStack API to Java. They would make it possible to interact with the OpenStack service from within the DBOD service software. Instances for the database servers could then be created and managed directly, decreasing the need of manual intervention. Different flavors could be set up in Nova, the DBOD software just choosing one of them that fits the needs of the user, and Nova carrying out the work of placing the instance on a suitable host.

Two Java interfaces implementing most of the API are *OpenStack4j* and *Apache jclouds*. The following table summarises some important characteristics of OpenStack4j and Apache jclouds.

	OpenStack4j	Apache jclouds
Website	openstack4j.com	jclouds.apache.org
Releases	since March 2014, 2 releases	since April 2009, 87 releases
Development	5 contributors	117 contributors, backed by Apache Software Foundation
API Coverage	only OpenStack	OpenStack, Docker, AWS, …
Trove API	no support	support included

While OpenStack4j focuses on OpenStack, Apache jclouds has the goal to support all cloud platforms to make it easy to port an application to another cloud provider. Both are implemented as fluent interfaces[27] and should be easy to use with the help of their documentation. The choice between them probably depends on the question if other APIs than only OpenStack are needed. If DBOD will make use of Docker, then either Apache jclouds can be used or OpenStack4j and one of the existing Java interfaces to Docker. For the case that DBOD uses Docker on OpenStack, no direct talking to

[27] `http://martinfowler.com/bliki/FluentInterface.html`

Docker should be needed, as Nova can take care of that. The following listing shows an example code booting a new instance using OpenStack4j[28]:

```
// Create a Server Model Object
Server server = Builders.server()
                         .name("Ubuntu 2")
                         .flavor("large")
                         .image("imageId")
                         .build();

// Boot the Server
Server server = os.compute().servers().boot(server);

// Create a Snapshot
os.compute().servers().createSnapshot("id", "name");
```

5 Conclusion and Future Prospects

This project has shown that the publicly available documentation and packages of Trove for Red Hat based Linux distributions are not yet ready for use. Some bug reports were filed to improve this situation, see A.2. In the end, Trove was running on Scientific Linux, but problems not related to Trove made it impossible to test its features. However, from the existing publications of companies using Trove[29], it can be believed that the basic functionality is indeed present and working. It might be a challenge to get OpenStack Trove running on Scientific Linux though—but once it is up and running, the functionality needed for DBOD might already mostly be there. Almost, because the integration of Oracle single-instance databases has not yet been started, only a blueprint is existing [17]. The integration of PostgreSQL was just finished during this summer on September, 4th, see the according blueprint [18] and has to be checked if it fulfils DBOD's needs. Also, it has to be evaluated if Trove can provide database administrator privileges to users, a feature which might already be supported with the `root-enable` capability that can be assigned to an instance [19]. Consolidation of resources on a level achieved by the current DBOD implementation could be realised by using Docker as virtualisation technique. The present backup strategy of DBOD based on snapshots probably just can be applied to Cinder volumes as well, because Cinder supports NetApp.

5.1 Further Suggestions for a Next Project on OpenStack Trove

There are some suggestions to take along for a next project on OpenStack Trove based on the experiences made during this project. Section 3.2 on page 12 shows on which ways to reach the OpenStack and Trove community. A summer project like the one presented in this report has to manage with little time. Thus, it seems best to join IRC channels, subscribe to mailing lists and forums at the very beginning so the atmosphere and the way of working of the community can be grasped. During the process of learning, it is a good way to check the own understanding of the subject and to give something back to the community by taking part in discussions or answering questions as far as the own knowledge makes it possible.

[28] the code example was taken from the project's website

[29] Some details can for example be found at slideshare.net/tesoracorp/5-hp-presentation-final and slideshare.net/tesoracorp/4-open-stack-trove-day-ebay-final. Replication and clusters seem to be the most interesting topic right now.

5.1.1 Technical Aspects

During this project, more time than expected beforehand was spent on installing the test setup and getting the basic configuration of Trove working. With this in mind, next time it might be better to start with DevStack on an Ubuntu virtual machine—or what might be the preferred development environment by then—to learn OpenStack and get familiar with its concepts. A single-machine DevStack setup would be sufficient to evaluate the status of Trove and to test its features, as section 3.5 on page 23 showed. The installation on CERN's preferred Linux distribution can be started afterwards, when there are plans to integrate Trove into DBOD.

5.2 Docker on OpenStack

As mentioned in the introduction, DBOD is moving towards using machines managed by OpenStack. Furthermore, DBOD could profit from another technology, which is working well together with OpenStack. Docker [20] is an only 1.5 year old tool providing easy-to-deploy Linux containers which is already used in some big software companies [21]. This is a virtualisation technique which doesn't rely on virtual machines but is working on kernel level using resource isolation features like cgroups and namespaces [22]. As a result, the containers use the same kernel and operating system as the host. This of course is less flexible as with fully-fledged virtual machines but much less resource intensive. Thus, if for the present use case it is not necessary to have a different kernel or operating system, Linux containers provide a good way to consolidate resources. For DBOD it would be perfectly fine to run the database instances on the same operating system than the host, as this would be the standard Scientific Linux CERN anyway.

Docker can be used as the underlying virtualisation technique for Nova in OpenStack. For users of Nova and all other OpenStack components built upon Nova, this is transparent—except the fact that the operating system cannot be chosen—making it possible that Trove can profit from this as well. There has already been a lot of work on integrating Docker into OpenStack, and it seems to be stable. Nitin Agarwal, a fellow CERN openlab summer student, has tested Docker and Docker on OpenStack during his project. The results will be available in his report [23].

5.3 From Scientific Linux to CentOS

Until now, CERN has been using a variant of Scientific Linux (SL) for all Linux based installations, called Scientific Linux CERN (SLC). As announced this year by the CERN IT Operating systems and Infrastructure Services section (IT-OIS), the next major Linux release at CERN will probably be based on CentOS [24]. This change has been caused by Red Hat and CentOS joining forces [25]. CentOS still is based on Red Hat Enterprise Linux, so any software running on SLC should be compatible with the new CERN CentOS.

The scripts written during this project were not tested on CentOS but should as well be compatible. Packstack runs on any Red Hat based Linux distribution, thus this part should not be a problem [26]. One issue that might arise is network namespaces not being supported by default, a feature of which OpenStack's network component Neutron and more precisely OpenVSwitch, makes heavy use. In this case this feature has to be installed beforehand. See [27] and [28] for guidelines on how to detect and solve this problem.

Acknowledgements

I would like to thank everyone who was involved in making my stay at CERN and my work on this project possible. The project was part of this year's CERN openlab summer student programme and belonged to CERN openlab's partnership with Oracle. Hence, I would like to thank Oracle for funding this project and CERN openlab for providing the basis for it. I thank my supervisor Ignacio Coterillo Coz that he gave me the opportunity to work with him. Thanks for the guidance throughout the whole time, the valuable discussions and the insights into DBOD. I would like to thank the whole IT-DB group for welcoming me and the other summer students into the team, it was a great time. I thank Giacomo Tenaglia from IT-DB for showing me the right way to reach the Trove community. I am grateful for the hint from `mikehn` on the IRC channel `#openstack-trove` which helped me to resolve one of the configuration issues. I thank Andrea Giardini who worked on OpenStack Trove as a summer student last year, for providing his results and notes. And finally, I would like to thank the CERN openlab management and administration team for the great work they did in preparing the whole summer programme.

This report was written in LaTeX using a template created by Kacper B Sokol.

A Appendix

A.1 Participation in the Community

ask.openstack.org The questions and answers contributed during this project can be found at `https://ask.openstack.org/en/users/7204/beni/`.

A.2 Bug Reports

Four bug reports were filed against OpenStack's documentation:

- Setting up Trove: Missing Configuration for Keystone Credentials `https://bugs.launchpad.net/openstack-manuals/+bug/1369116`
- Setting up Trove: Missing Configuration Option for Taskmanager prevents it from starting `https://bugs.launchpad.net/openstack-manuals/+bug/1369119`
- Setting up Trove: Refer to cloudinit for provisioning instances `https://bugs.launchpad.net/openstack-manuals/+bug/1369123`
- Default region for the service endpoints is different for docs and Packstack/DevStack `https://bugs.launchpad.net/openstack-manuals/+bug/1369329`

References

[1] Ruben Domingo Gaspar Aparicio, Ignacio Coterillo Coz, Daniel Gomez Blanco and David Collados Polidura. *Database on Demand: a DBaaS story*, Jun 2014. URL `http://indico.cern.ch/event/313869/`.

[2] Solinea. *OpenStack Grizzly Architecture (revisited)*, Jun 2013. URL `http://www.solinea.com/blog/openstack-grizzly-architecture-revisited`.

[3] Dean Troyer. *OpenStack - Seven Layer Dip as a Service*, Sep 2013. URL `http://hackstack.org/x/blog/2013/09/05/openstack-seven-layer-dip-as-a-service/`.

[4] Sean Dague. *OpenStack as Layers*, Aug 2014. URL `https://dague.net/2014/08/26/openstack-as-layers/`.

[5] Tim Bell. *A tale of 3 OpenStack clouds : 50,000 cores in production at CERN*, Sep 2013. URL `http://openstack-in-production.blogspot.fr/2013/09/a-tale-of-3-openstack-clouds-50000.html`.

[6] Tim Bell. *Managing identities in the cloud*, Aug 2013. URL `http://openstack-in-production.blogspot.fr/2013/08/managing-identities-in-cloud.html`.

[7] Anastasios Andronidis. *Cold start booting of 1000 VMs under 10 minutes*, 2014. URL `https://openstacksummitnovember2014paris.sched.org/event/6724030ccceb0b4ec1c694e09ce9a08b`.

[8] Andrea Giardini. *DBaaS with OpenStack Trove*. Tech. Rep. CERN-STUDENTS-Note-2013-179, CERN, Geneva, Sep 2013. URL `http://cds.cern.ch/record/1597972?ln=en`.

[9] OpenStack community. *TroveArchitecture*, Dec 2013. URL `https://wiki.openstack.org/wiki/TroveArchitecture`.

[10] Denis Makogon. *OpenStack Trove Database-as-a-Service Overview*, Nov 2013. URL `http://de.slideshare.net/mirantis/trove-d-baa-s-28013400`.

[11] RDO Community. *PackStack All-in-One DIY Configuration*, Aug 2013. URL `https://openstack.redhat.com/PackStack_All-in-One_DIY_Configuration`.

[12] OpenStack community. *Install the Database service - OpenStack Installation Guide for Red Hat Enterprise Linux, CentOS, and Fedora - icehouse*, 2014. URL `http://docs.openstack.org/icehouse/install-guide/install/yum/content/trove-install.html`.

[13] OpenStack communitiy. *Manual Trove Installation*, 2013. URL `http://docs.openstack.org/developer/trove/dev/manual_install.html`.

[14] DevStack Community. *DevStack – Deploying OpenStack for Developers*, 2014. URL `http://devstack.org/`.

[15] Stuart Colville. *Howto: SSH into VirtualBox 3 Linux Guests*, Feb 2010. URL `https://muffinresearch.co.uk/howto-ssh-into-virtualbox-3-linux-guests/`.

[16] DevStack Community, 2013. URL
`http://devstack.org/guides/single-machine.html`.

[17] Denis Makogon. *Support single instance OracleDB*, Feb 2014. URL
`https://blueprints.launchpad.net/trove/+spec/oracle-db`.

[18] Kevin Conway. *Add support for postgresql databases*, Nov 2013. URL
`https://blueprints.launchpad.net/trove/+spec/postgresql-support`.

[19] OpenStack Community. *Trove/trove-capabilities*, Jun 2014. URL
`https://wiki.openstack.org/wiki/Trove/trove-capabilities#Capabilities`.

[20] Docker. *Docker - Build, Ship, and Run Any App, Anywhere*, 2014. URL
`https://docker.com/`.

[21] Docker. *Use Cases - Examples*, 2014. URL `https://docker.com/resources/usecases/`.

[22] Docker Documentation. *Kernel Requirements*, Jan 2014. URL
`http://docker.readthedocs.org/en/v0.7.3/installation/kernel/`.

[23] Nitin Agarwal. *Docker on OpenStack*, 2014. URL
`http://openlab.web.cern.ch/education/summer-students/reports?field_published_date_value[value][year]=2014`.

[24] CERN IT-OIS Linux. *CC7: CERN CentOS 7*, Aug 2014. URL
`http://linux.web.cern.ch/linux/centos7/`.

[25] CERN IT-OIS Linux. *Scientific Linux @ CERN: Next Version*, Aug 2014. URL
`http://linux.web.cern.ch/linux/nextversion.shtml`.

[26] RDO Community. *Red Hat Deployed OpenStack*, 2014. URL
`https://openstack.redhat.com/Main_Page`.

[27] Yanis Guenane. *Enable network namespaces in CentOS 6.4*, Nov 2013. URL
`http://spredzy.wordpress.com/2013/11/22/enable-network-namespaces-in-centos-6-4/`.

[28] OpenStack Community. *Does the CentOS/RHEL 6.5 kernel support namespaces?*, 2014. URL
`https://ask.openstack.org/en/question/10738/does-the-centosrhel-65-kernel-support-namespaces/`.