StratusLab Administrator's Guide

StratusLab Collaboration

Version 13.12.0

Contents

Chapter 1

Preface

1.1 Target Audience

This guide provides information for system administrators wanting to install and maintain a StratusLab cloud.

Those interested in using the resources in a StratusLab cloud should instead consult the StratusLab User's Guide.

Those wishing to contribute to the StratusLab software or documentation should consult the StratusLab Contributor's Guide.

1.2 Typographic Conventions

This guide uses several typographic conventions to improve the readability.

links	some link
filenames	`$HOME/.stratuslab/stratuslab-user.cfg`
commands	`stratus-run-instance`
options	`--version`

Table 1.1: Typographic Conventions

Extended examples of commands and their outputs are displayed in the monospace Courier font. Within these sections, command lines are prefixed with a '$' prompt. Lines without this prompt are output from the previous command. For example,

```
$ stratus-run-instance -q BN1EEkPiBx87_uLj2-sdybSI-Xb
5507, 134.158.75.75
```

the `stratus-run-instance` is the command line which returns the virtual machine identifier and IP address.

Chapter 2

Introduction

Describe all of the services and how they fit together in detail. Provide a detailed view of a cloud deployment.

How StratusLab fits into the various cloud service models. It is a IaaS!

Different types of deployments: public, private, and community.

2.1 Organization

The guide starts with this introduction and then is followed by an installation tutorial showing all of the steps to install a minimal, two-machine StratusLab cloud infrastructure. The subsequent chapter provide more detail on various aspects of the software.

2.2 Terminology

The older terminology, still used in most parts of this (and other) documents is the following:

- Frontend: A physical machine that runs the services for virtual machine management and storage services. This is the machine which is contacted by the cloud users for managing their cloud resources.
- Node: A physical machine that hosts virtual machines.

The newer terminology that has been adopted is:

- Cloud Entry Point: One or more physical machines that run the CIMI service. Users use this management interface to acquire, control, and release all of their cloud resources.
- VM Host: One or more physical machines that host virtual machines.
- Storage Controller: One or more physical machines that act as a gateway to a storage services.
- Couchbase Node: A physical machine that hosts a Couchbase instance running as part of a Couchbase cluster. This may be combined with either the Cloud Entry Points or the VM Hosts to minimize the number of physical machines used for the supporting services of the cloud.

The software and documentation is being gradually updated to consistently use this newer terminology.

Chapter 3

Installation

This tutorial demonstrates how to install manually a StratusLab cloud infrastructure using the StratusLab system administrator command line utilities. A minimal StratusLab cloud consists of two physical machines, although additional machines may be necessary if the internet cannot be accessed.

3.1 Installation Overview

The StratusLab distribution provides a simple command line client to install, configure and start the StratusLab Cloud services and components.

The default deployment has two types of machines:

1. **Front-End** - machine for VM management and storage services
2. **Node** - machine that hosts virtual machines

A quick overview of the procedure is:

1. Ensure all prerequisites are satisfied.
2. Define all of the StratusLab service parameters.
3. Install and configure the Front End, containing the VM management service (OpenNebula) and the storage management (Persistent Disk) service.

4. Install and configure the Node(s) via SSH from the Front End. By default KVM is used for the hypervisor on the Node(s).

5. Validate the installation by starting a virtual machine.

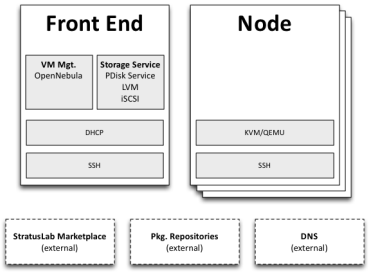

Figure 3.1: Minimal StratusLab Cloud

3.2 Prerequisites

3.2.1 Physical Machines

This tutorial demonstrates a minimal installation of a StratusLab cloud on two physical machines. The physical machines should be relatively modern machines with the following **minimum** characteristics:

- 1 **64-bit multicore** CPU (>= 4 cores) with **VT-x extensions**
- 4 GB of RAM
- 200 GB local disk space

The hardware virtualization extensions must be enabled in the BIOS on the "Node" machine. Many vendors ship machines with these extensions disabled.

In general cloud infrastructures prefer "fat" machines, that is machines that have a maximum number of CPUs, RAM, and disk space as possible. This is because the maximum size of a single virtual machine is limited by the size of the largest physical machine.

3.2.2 Operating System

Install a minimal version of CentOS 6 on the two physical machines that will be used for the cloud infrastructure.

3.2.3 Disable SELinux

The SELinux system must be disabled on **all of the machines**. Normally this is enabled by default. To disable SELinux, ensure that the file `/etc/selinux/config` has the following line:

```
SELINUX=disabled
```

You must reboot the machine for this to take effect.

3.2.4 Python Version

The default version of Python installed with CentOS should be correct. StratusLab requires a version of Python 2 with a version **2.6 or later**. The StratusLab command line tools **do not work with Python 3**.

Verify that the correct version of Python is installed:

```
$ python --version
Python 2.6.6
```

3.2.5 Disk Configuration

StratusLab allows for a variety of storage options behind the persistent disk service. The tutorial uses the defaults using LVM and iSCSI.

The machines must be configured to use LVM for the disk storage.

The Front End must be configured with two LVM groups: one for the base operating system (~20 GB) and one for the StratusLab storage service (remaining space).

The "Node" machine can be configured with a single LVM group.

Below, we assume that the volume group names are "vg.01" for the operating system and "vg.02" for the StratusLab storage service. You can use other names, but then change the commands below as necessary.

3.2.6 Package Repositories

The StratusLab installation takes packages from four yum repositories:

1. The standard CentOS repository,
2. The EPEL 6 repository,
3. The StratusLab repository, and
4. The IGTF Root Certificates.

The configuration for the CentOS repository is done when the system is installed. The others require additional configuration.

To configure **both** the Front End and Node for the EPEL repository, do the following:

```
$ wget -nd http://mirrors.ircam.fr/pub/fedora/epel/6/i386/epel-release-6-8.noarch.
$ yum install -y epel-release-6-8.noarch.rpm
```

This will add the necessary files to the `/etc/yum.repos.d/` directory.

To configure **both** the Front End and Node for the StratusLab repository, put the following into the file `/etc/yum.repos.d/stratuslab.repo`:

```
[StratusLab-Releases]
name=StratusLab-Releases
baseurl=http://yum.stratuslab.eu/releases/centos-6.2-v13.02/
gpgcheck=0
```

replacing the URL with the version you want to install.

Although not strictly necessary, it is advisable to clear all of the yum caches and upgrade the packages to the latest versions:

```
$ yum clean all
$ yum upgrade -y
```

This may take some time if you installed the base operating system from old media.

3.2.7 DNS and Hostname

Ensure that the **hostname** is properly setup on the Front End and the Node. The DNS must provide both the forward and reverse naming of the nodes. This is required for critical services to start.

You can verify this on both the Front End and the Node with the command:

```
$ hostname -f
```

Set the hostname if it is not correct.

Throughout this tutorial we use the variables $FRONTEND_HOST ($FRONTEND_IP) and $NODE_HOST ($NODE_IP) for the Front End and Node hostnames (IP addresses), respectively. Change these to the proper names for your physical machines when running the commands.

3.2.8 SSH Configuration

The installation scripts will automate most of the work, but the scripts require **password-less root access**:

- From the Front End to each Node and
- From the Front End to the Front End itself

Check to see if there is already an SSH key pair in `/root/.ssh/id_rsa*`. If not, then you need to create a new key pair **without a password**:

```
$ ssh-keygen -q
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

Now ensure that you can log into the Front End from the Front End without needing a password. Do the following:

```
$ ssh-copy-id $FRONTEND_HOST
The authenticity of host 'onehost-5.lal.in2p3.fr␣(134.158.75.5)' can't␣be␣established.
RSA␣key␣fingerprint␣is␣e9:04:03:02:e5:2e:f9:a1:0e:ae:9f:9f:e4:3f:70:dd.
Are␣you␣sure␣you␣want␣to␣continue␣connecting␣(yes/no)?␣yes
Warning:␣Permanently␣added␣'onehost-5.lal.in2p3.fr,134.158.75.5'␣(RSA)␣to␣the␣list␣of␣kr
```

```
root@onehost-5.lal.in2p3.fr's password:
Now try logging into the machine, with "ssh 'onehost-5.lal.in2p3.fr'", and check i

  .ssh/authorized_keys

to make sure we haven't added extra keys that you weren't expecting.
```

Do the same thing for the node:

```
$ ssh-copy-id $NODE_HOST
...
```

And verify that the password-less access works as expected.

```
$ ssh $FRONTEND_HOST

Last login: Mon May 27 14:26:29 2013 from mac-91100.lal.in2p3.fr
#
# exit
logout
Connection to onehost-5.lal.in2p3.fr closed.

$ ssh $NODE_HOST

Last login: Mon May 27 14:26:43 2013 from mac-91100.lal.in2p3.fr
#
# exit
logout
Connection to onehost-6.lal.in2p3.fr closed.
```

Now that SSH is properly configured, the StratusLab scripts will be able to install software on both the Front End and the Node.

3.2.9 DHCP Server

A DHCP server must be configured to assign static IP addresses corresponding to known MAC addresses for the virtual machines. These IP addresses must be publicly visible if the cloud instances are to be accessible from the internet.

If an external DHCP server is not available, the StratusLab installation command can be used to properly configure a DHCP server on the Front End for the virtual machines.

This uses a DHCP server on the Front End.

3.2.10 Network Bridge

A network bridge must be configured on the Node to allow virtual machines access to the internet. You can do this manually if you want, but the StratusLab installation scripts are capable of configuring this automatically.

This tutorial allows the installation scripts to configure the network bridge.

3.3 Front End Deployment

3.3.1 Deployment tool installation

The first step is to install the StratusLab system administrator command line client from the StratusLab repository **on the Front End**:

```
$ yum install -y stratuslab-cli-sysadmin
```

This will install the system administrator client and all of the necessary dependencies. You can verify that it is correctly installed by doing the following:

```
$ stratus-config --help

Usage: stratus-config [options] [key [value]]
If the [value] is not provided, the command returns the current value
of the key.
...
```

3.3.2 Configuration file customization

The entire StratusLab Cloud is configured from a single configuration file `/etc/stratuslab/stratuslab.cfg`. This file contains many options, but only a few are required.

StratusLab ships with a default configuration file in the standard location and a reference configuration file located in `/etc/stratuslab/stratuslab.cfg.ref`.

To simplify viewing the configuration parameters and changing them, the `stratus-config` command can be used.

To list the content of the configuration, and show the differences between the `stratuslab.cfg` file and the reference configuration, you can use the `-k` or `--keys` option:

```
$ stratus-config -k

... lots of parameter values! ...
```

To change a value, specify the key and the new value. To view a single value, simply specify the key.

We will use this command to set the various configuration parameters below.

3.3.3 VM Management Service

The parameters for the frontend and VM management:

```
$ stratus-config frontend_system centos
$ stratus-config frontend_ip $FRONTEND_IP
```

3.3.4 Storage Service

Similar parameters must also be set for the Persistent Disk service.

For this tutorial, this service is installed on the Front End, so the same IP address should be used.

```
$ stratus-config persistent_disk_system centos
$ stratus-config persistent_disk_ip $FRONTEND_IP
$ stratus-config persistent_disk_merge_auth_with_proxy True
```

The Persistent Disk service and the Nodes communicate using a strategy defined by the `persistent_disk_storage` and `persistent_disk_share` parameters. The default values ("lvm" and "iscsi", respectively) will be used for this tutorial.

One needs to specify what device will be used for the physical storage for the Persistent Disk service:

```
$ stratus-config persistent_disk_lvm_device /dev/vg.02

# Provide detailed parameters for storage backend plugins.
# (NOTE: The opening and closing single quotes!)
$ stratus-config persistent_disk_backend_sections '
[%(persistent_disk_ip)s]
```

```
␣␣␣␣␣␣␣␣type=LVM
␣␣␣␣␣␣␣␣volume_name␣=␣/dev/vg.02
␣␣␣␣␣␣␣␣lun_namespace␣=␣stratuslab
␣␣␣␣␣␣␣␣volume_snapshot_prefix␣=␣pdisk_clone
␣␣␣␣␣␣␣␣initiator_group␣=
'
```

If you've used another name for the LVM volume group, then change the above command.

3.3.5 Network configuration

Use the frontend as the general gateway for the cloud:

```
$ stratus-config default_gateway $FRONTEND_IP
```

Set the IP and mac addresses for virtual machines:

```
$ stratus-config one_public_network_addr \
    134.158.xx.yy 134.158.xx.yy 134.158.xx.yy

$ stratus-config one_public_network_mac \
    0a:0a:86:9e:49:2a 0a:0a:86:9e:49:2b 0a:0a:86:9e:49:2c
```

In this example, the Front-End is configured on IP address $FRONTEND_IP and three IP/MAC address pairs are defined for virtual machines.

You must use the real values for the Front End IP addresses and for the range of addresses you will use for the virtual machines.

More network parameters are described in the "one-network" section in the reference configuration file.

3.3.6 DHCP Configuration

Allow the script to automatically configure and start the DHCP server on the Front End. Do the following:

```
$ stratus-config dhcp True
$ stratus-config dhcp_subnet 134.158.75.0
$ stratus-config dhcp_netmask 255.255.255.0
$ stratus-config dhcp_lease_time 3600

$ stratus-config dhcp_one_public_network True
$ stratus-config dhcp_one_local_network_routers $FRONTEND_IP
```

```
$ stratus-config dhcp_one_local_network_domain_name lal.in2p3.fr
$ stratus-config dhcp_one_local_network_domain_name_servers \
    134.158.91.80, 134.158.88.149
```

Use **your** values for these parameters!

3.3.7 Finalize Front End Installation

Now that we have defined all of the configuration parameters, you can now do the full Front End installation by issuing the following command:

```
$ stratus-install -vv
```

To get more details on what the command is (because of curiosity or errors), use the option `-v`, `-vv`, or `-vvv`.

If you run into errors, the `stratus-install` command can simply be rerun after adjusting the configuration parameters.

3.4 Node Deployment

The deployment of the StratusLab Nodes is done from the Front End, thus, **all the commands below should be run from the Front End.**

To add a Node to the cloud, specify the Linux distribution of the machine and indicate that the bridge should be configured:

```
$ stratus-config node_system centos
```

Request the automatic configuration of the network bridge:

```
$ stratus-config node_bridge_configure True
$ stratus-config node_bridge_name br0
$ stratus-config node_network_interface eth0
```

Check carefully the name of the interface on the node!

Invoke installation by

```
stratus-install -vv -n $NODE_IP
```

As before, you can increase the verbosity level by adding the option `-v` or `-vv`.

3.5 User Configuration

At this point, you have both the Front End and one Node installed. This is a functional installation, but you have not yet authorized any users for the cloud. Here we will create a new StratusLab user account. Note that StratusLab accounts are independent of the Unix accounts on the machine itself.

Add the following line to the end of the file `/etc/stratuslab/authn/login-pswd.properties`.

```
$ cat >> /etc/stratuslab/authn/login-pswd.properties
sluser=slpass,cloud-access
```

This creates a new StratusLab user 'sluser' with a password 'slpass'. The group 'cloud-access' is mandatory for the user to have access to the cloud services. (Crypted or hashed password values are also allowed in the configuration.)

The StratusLab distribution supports other authentication methods (LDAP, X509 certificates, X509 proxies, etc.), but a username/password pair is the simplest for this tutorial.

3.6 StratusLab Client

Now we will test that the cloud functions correctly by starting a new virtual machine instance and logging into it. We'll test the cloud service from a normal Unix user account on the Front End.

First, ensure that the StratusLab user client is installed on the machine. Do the following as root:

```
$ yum install -y stratuslab-cli-user
```

It is very likely that the user client commands are already installed.

(Note: For normal client installations, it is strongly recommended to use pip or easy_install with virtualenv. See the usual client installation instructions.)

Now create a normal Unix user for testing:

```
$ adduser sluser
```

Now log in as the user and setup the account for using StratusLab. An SSH key pair is required to log into your virtual machines and the client requires that a complete client configuration file.

Log in as the user and create an SSH key pair. This is similar to the process used for the root account on the machine.

```
$ su - sluser
$ ssh-keygen -q
...
```

Now copy the reference configuration file into place and edit the parameters.

```
$ mkdir ~/.stratuslab
$ cd ~/.stratuslab
$ cp /etc/stratuslab/stratuslab-user.cfg.ref ~/.stratuslab/stratuslab-user.cfg
$ vi ~/.stratuslab/stratuslab-user.cfg # endpoint, username, password
```

You will need to set the “endpoint”, “username”, and “password” parameters in this file. For the “endpoint” use the hostname or IP address of your Front End. For the “username” and “password” use “sluser” and “slpass”, respectively.

Everything should be setup now. So try deploying a virtual machine. You can look in the Marketplace to find an interesting machine to deploy. We’ll use a ttylinux image here. This is a micro distribution that boots very quickly and is ideal for tests.

```
# Deploy a ttylinux virtual machine.
$ stratus-run-instance BN1EEkPiBx87_uLj2-sdybSI-Xb

 ::::::::::::::::::::::::
 :: Starting machine(s) ::
 ::::::::::::::::::::::::
 :: Starting 1 machine
 :: Machine 1 (vm ID: 1)
 Public ip: 134.158.75.42
 :: Done!
```

Check the status of the machine as it starts:

```
# Check its status.  Pending -> not yet assigned to a Node
$ stratus-describe-instance
id  state     vcpu memory    cpu% host/ip                 name
1   Pending   1    0         0    vm-42.lal.stratuslab.eu one-1

# Check again.  Prolog -> resources for VM are being initialized
$ stratus-describe-instance
```

```
id  state     vcpu memory    cpu% host/ip                  name
1   Prolog    1    0         0    vm-42.lal.stratuslab.eu one-1

# Check again. Running -> hypervisor has started machine
$ stratus-describe-instance
id  state     vcpu memory    cpu% host/ip                  name
1   Running   1    0         0    vm-42.lal.stratuslab.eu one-1
```

When the machine reaches the 'running' status, the virtual machine is running in the hypervisor on the Node. It will probably take some additional time for the operating system to boot.

Verify that the machine has fully booted and is accessible from the network:

```
# Ping the virtual machine to see if it is accessible.
$ ping vm-42.lal.stratuslab.eu
PING vm-42.lal.stratuslab.eu (134.158.75.42) 56(84) bytes of data.
From onehost-5.lal.in2p3.fr (134.158.75.5) icmp_seq=2 Destination Host
 Unreachable
...
From onehost-5.lal.in2p3.fr (134.158.75.5) icmp_seq=8 Destination Host
 Unreachable
64 bytes from vm-42.lal.stratuslab.eu (134.158.75.42): icmp_seq=9
 ttl=64 time=1.44 ms
...

# Now login to the machine as root.
$ ssh root@vm-42.lal.stratuslab.eu

The authenticity of host 'vm-42.lal.stratuslab.eu␣(134.158.75.42)'
 can't␣be␣established.
RSA␣key␣fingerprint␣is
␣6a:bd:f7:2d:b6:82:39:61:e6:ca:3f:c7:61:9d:72:31.
Are␣you␣sure␣you␣want␣to␣continue␣connecting␣(yes/no)?␣yes
Warning:␣Permanently␣added␣'vm-42.lal.stratuslab.eu,134.158.75.42'
␣(RSA)␣to␣the␣list␣of␣known␣hosts.


#␣␣␣␣␣␣␣#␣<-␣we're logged into the ttylinux virtual machine
# exit  # just logout of the session
logout
Connection to vm-42.lal.stratuslab.eu closed.
```

Now the machine can be terminated:

```
$ stratus-kill-instance 1
```

Going through the full lifecycle of a machine shows that all of the services are working.

3.7 Conclusions

You've successfully installed a minimal StratusLab cloud. You can checkout the documentation to see what other configuration parameters are available or try the user tutorials to discover more of the StratusLab services.

You can get help on the installation or use of StratusLab through the support mailing list. You can also report bugs and provide feedback on the same list.

Chapter 4

Network Configuration

4.1 Network Services

4.1.1 DHCP

4.1.2 DNS

4.1.3 IP Addresses

4.2 VLANs

Describe the different VLANs for the cloud infrastructure.

4.3 Services and Ports

List all of the services and the ports that they use. Describe to what extent those ports need to be open.

Chapter 5

Authentication and Authorization

5.1 Authentication Methods

StratusLab supports a wide range of different authentication methods and account storage options, allowing it to fit into a wide range of sites with pre-existing authentication systems.

All of the configuration files for authentication are located in the directory `/etc/stratuslab/authn`. The main configuration file controlling the active authentication methods and their parameters is `login.conf`. Properties files with username/password or certificate information are also located in this directory.

5.1.1 Username and Password

Users can be identified through username/password pairs. Accounts associated with these credentials can be stored in:

- A java properties file or
- An LDAP server

If you use an LDAP server, the StratusLab Registration service can be used to allow users to register for an account.

5.1.2 X509 Credentials

Users can also be identified via client X509 credentials. Raw X509 certificates, RFC3820 certificate proxies, or VOMS proxies can all be used to connect to and to authenticate with the StratusLab services.

As above, these credentials can be authorized in:

- A java properties file or
- An LDAP server

The registration service can also manage the required X509 Distinguished Names (DNs) if using an LDAP server.

5.2 Authorization

Apart from the super user, users can only see and control resources that they create themselves.

5.3 Registration Service

StratusLab provides an optional service, the “Registration Service”, that allows users to register for an account on your cloud infrastructure. Through a web-accessible interface, users can register for an account and update the information associated their accounts.

The registration workflow proceeds through the following stages:

1. User submits form with required information,
2. Service sends email validation message to user,
3. User confirms email address by visiting confirmation link,
4. Service sends account validation request to administrator,
5. Administrator accepts or rejects account request, and
6. An email is sent to the user with the decision.

When users update their email addresses, the new email address is also validated through an email and confirmation link.

The user information is stored in a separate LDAP server. The registration service must have full access to that server, so that it can manage the user entries.

To use the accounts managed by the Registration Service on your cloud infrastructure, you must configure the authentication service to use the LDAP server as a source of information. Users who supply a certificate DN, can also authenticate with the cloud using their certificate.

Chapter 6

Running a Cloud

Information on running the cloud as a service.

6.1 Service Logging Information

Where are the service logs?

6.2 Monitoring Activity

Where to find monitoring information.

6.3 Security Concerns

Various topics.

Chapter 7

Troubleshooting

Information about common problems.

Chapter 8

Architecture

The architecture of the StratusLab software has evolved to make the overall system more scalable and more robust. At the center of the system is a distributed database that removes single points-of-failure in the system and permits redundant deployments of StratusLab service components for better reliability.

StratusLab also now exposes the CIMI interface–a standard from DMTF that provides a coherent, unified API for all of the StratusLab cloud resources and that follows the usual REST patterns. It simplifies programmatic access to the cloud and provides a good foundation for browser-based access.

8.1 StratusLab Components

The diagram shows a high-level view of the various components in the StratusLab architecture. The distributed database is at the core of the system. On the user-facing side are machines that provide the CIMI interface to the system. On the service side are a set of controllers for different types of resources (storage, VMs, etc.). Behind the controllers are the physical resources used by the cloud.

The distributed database contains the complete state of the cloud. Consequently, the CIMI interfaces and the various controllers can be completely stateless, allowing redundant instances to be deployed as necessary.

As all of the communication between controllers takes place through the database, specialized controllers can be added to the cloud infrastructure

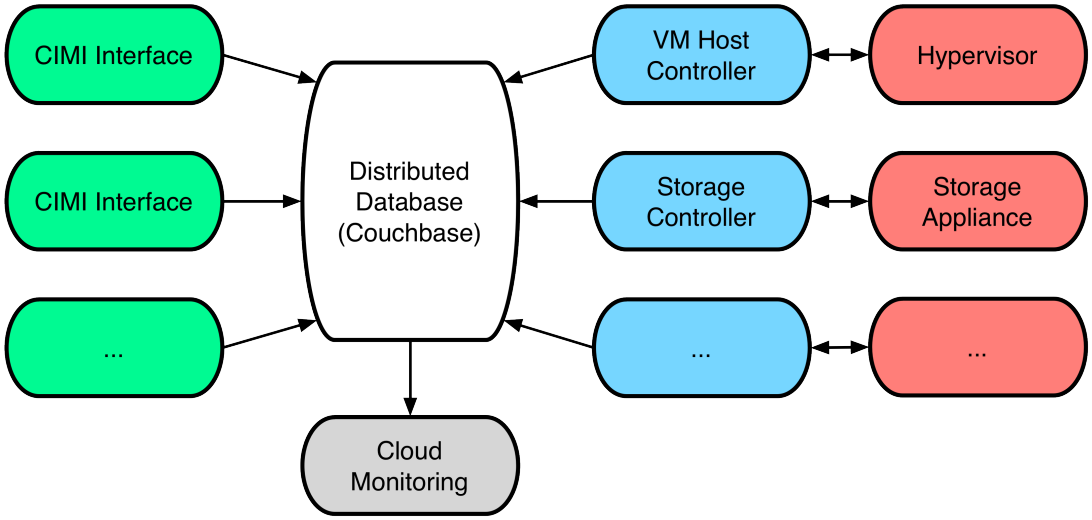

Figure 8.1: StratusLab Architecture

easily, such as:

- Different type of storage (backed up, shared, fast, etc.)
- Support for Linux containers as well as virtual machines
- Dynamic network configurations

The controllers react to jobs and resources in the database, and update those entries when completing tasks or making adjustments to resources.

8.2 Service Configuration

Generally, the configuration of the StratusLab cloud services also resides within the database. This allows deployment of redundant services while maintaining a consistent configuration of those services across machines.

The configuration information within the database is split into a series of JSON-formatted documents. Each document has an document identifier of the form `ServiceConfiguration/service-instance`, where the "service" corresponds to the name of the service and the "-instance" is optional

and used only if the configuration of an instance differs from the general service configuration.

Most configuration exists within the database, but there are two notable exceptions: third-party services and the Couchbase access parameters.

To minimize the StratusLab development effort, third-party services used by StratusLab are not modified to use the database. Consequently, these continue to use configuration files on the local file system. One example are the certificate authorities used for PKI authentication in the European Grid Infrastructure. When deploying multiple service instances, these files must be coordinated between physical machines.

The StratusLab services must discover the contact parameters for the Couchbase database. This information resides in a file on each node (`/etc/stratuslab/couchbase.cfg`). If this file doesn't exist, then services will use the default bucket and assume that the local machine is a member of the Couchbase cluster.

The standard "ini" format is used for the file:

```
[DEFAULT]
host=localhost
bucket=default
username=default
password=

[service]
config=ServiceConfiguration/service-instanceX
```

This example shows the default values. You can define a different bucket and location for the database. However, the same database must be shared by all of the StratusLab cloud services. **If changes are made to this file, then the StratusLab service must be restarted.**

The example also shows the possibility of providing instance-specific configuration documents for services. Generally, these should not be needed but are available for allow for specialized service configurations.

Despite the exceptions, having the majority of configuration information in the database will make configuring and maintaining the services simpler for system administrators. Generally, it is just a matter of updating a document in the database to change the behavior of the cloud infrastructure.

Chapter 9

Planning the Deployment

The components of the StratusLab distribution are quite modular and allow for a variety of different deployment layouts. This chapter provides some advice when considering the deployment of StratusLab in your data center.

The chapter concludes with a description of the minimal deployment of StratusLab on two machines. This deployment serves as a good testbed for understanding the installation process and how the cloud services work.

9.1 Your Cloud Users

When considering your deployment, you must consider who are your intended users. The level of trust you have in those users will impact further decisions on the network configuration, service layout, and authentication methods.

For clouds at the "Infrastructure as a Service" level, like StratusLab, there are generally three types of cloud deployments: private, community, and public.

Private cloud deployments target a limited set of known and trusted users. An example is using a cloud deployment to provide a flexible infrastructure for an institute's services. These types of clouds are often run by and used by the same system administrators. Because of the high-level of trust in the users, less needs to be done to isolate cloud services from the cloud's virtual machines.

On the other end of the spectrum are public cloud infrastructures. These target external and possibly unknown users outside of the institute's administrative domain. These generally require a higher level of isolation of service and more fine-grained network segmentation.

Between the two are "community" clouds. These function like public clouds but have a more limited set of users. A cloud infrastructure destined for scientists at a number of collaborating institutes is a good example of a community cloud infrastructure.

9.2 Network Configuration

StratusLab puts very few constraints on the network configuration supporting the cloud infrastructure. At its simplest, it can work on a single network tying together all cloud services and resources. It can also be made to work on complicated network configurations with multiple VLANs, protocols, etc.

The only network constraints for StratusLab are:

- Range of addresses to allocate to virtual machines,
- DCHP server configured to serve those addresses, and
- DNS configured with reverse-DNS lookups for those addresses.

The range of addresses must be IPv4 addresses, but each address can also have an IPv6 address associated with it.

There are three different types of network traffic on a StratusLab cloud infrastructure:

- Control messages between the cloud services (and through the Couchbase database),
- Network access to virtual machines in the cloud, and
- Traffic between nodes hosting virtual machines and physical resources (e.g. storage).

The level of isolation between these different types of traffic depends on your type of cloud deployment and your desire for a secure infrastructure.

For a private cloud infrastructure, mixing these three types of traffic on the same network doesn't pose any particular problem and is the easiest to configure.

However, for a public cloud infrastructure it is a good idea to separate the three types of traffic on separate VLANs. Even better, physical segregation of the virtual machine traffic from the other traffic prevents any possible compromise of the cloud services from the running virtual machines. This is obviously more complicated to configure. You must weight the additional complexity against the potential threats from your users.

9.3 Mapping Services to Machines

There are three classes of services that need to be mapped to physical machines: Couchbase instances, CIMI interfaces, and resource controllers.

The usual mapping for Couchbase instances and CIMI interfaces is to deploy these on the same physical machine. For scalability and redundancy, two or more such physical machines are deployed.

Resource controllers are generally deployed on the same physical nodes that provide the underlying resources. For example, the VM controller is deployed directly on the machine(s) running the hypervisors. This arrangement minimizes the number of physical machines that need to be deployed.

You can alter the mapping as necessary to adapt the needs of your data center. For example, you may have resources (like a storage appliance) that require having the controller hosted on a different machine than the resource it is controlling.

9.4 Minimal Deployment

The examples in this guide use a minimal deployment of two physical machines. One machine (the "cloud entry point") contains the Couchbase database and the CIMI interface. The other contains all of the resources

controllers and resources used for the cloud. A single network for all traffic types is assumed for simplicity in this minimal deployment. The following diagram summarizes this deployment.

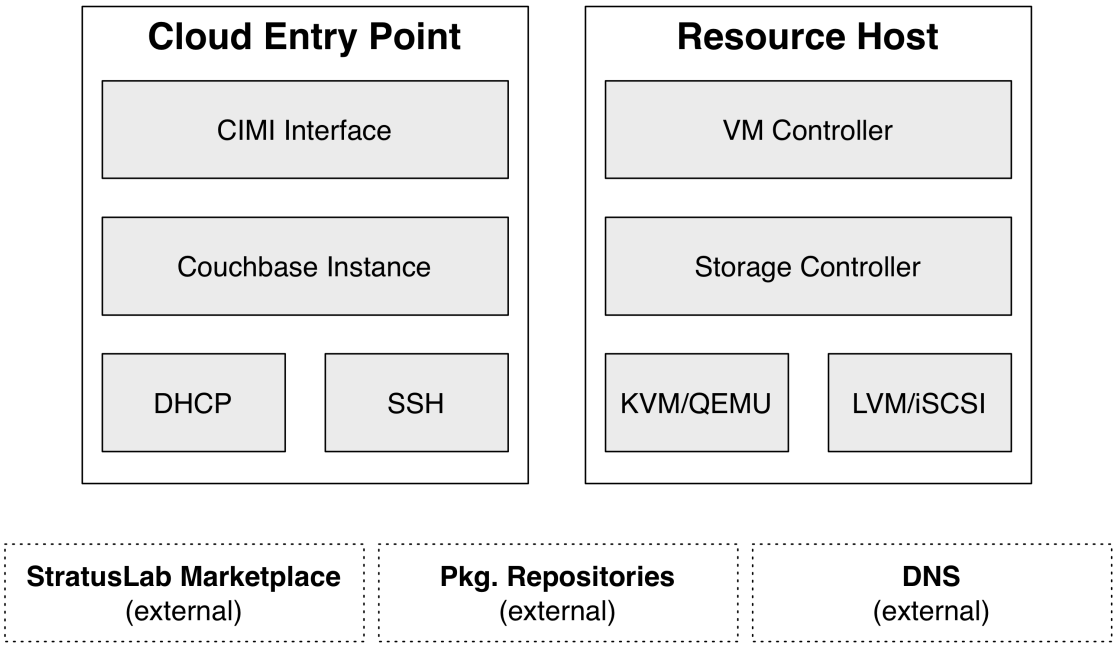

Figure 9.1: Minimal StratusLab Deployment

Chapter 10

Prerequisites

10.1 Physical Machines

This tutorial demonstrates a minimal installation of a StratusLab cloud on two physical machines. The physical machines should be relatively modern machines with the following **minimum** characteristics:

- 1 **64-bit multicore** CPU (>= 4 cores) with **VT-x extensions**
- 4 GB of RAM
- 200 GB local disk space

The hardware virtualization extensions must be enabled in the BIOS on the "Node" machine. Many vendors ship machines with these extensions disabled.

In general cloud infrastructures prefer "fat" machines, that is machines that have a maximum number of CPUs, RAM, and disk space as possible. This is because the maximum size of a single virtual machine is limited by the size of the largest physical machine.

10.2 Operating System

Install a minimal version of CentOS 6 on the two physical machines that will be used for the cloud infrastructure.

10.3 Disable SELinux

The SELinux system must be disabled on **all of the machines**. Normally this is enabled by default. To disable SELinux, ensure that the file `/etc/selinux/config` has the following line:

```
SELINUX=disabled
```

You must reboot the machine for this to take effect.

10.4 Python Version

The default version of Python installed with CentOS should be correct. StratusLab requires a version of Python 2 with a version **2.6 or later**. The StratusLab command line tools **do not work with Python 3**.

Verify that the correct version of Python is installed:

```
$ python --version
Python 2.6.6
```

10.5 Disk Configuration

StratusLab allows for a variety of storage options behind the persistent disk service. The tutorial uses the defaults using LVM and iSCSI.

The machines must be configured to use LVM for the disk storage.

The Front End must be configured with two LVM groups: one for the base operating system (~20 GB) and one for the StratusLab storage service (remaining space).

The "Node" machine can be configured with a single LVM group.

Below, we assume that the volume group names are "vg.01" for the operating system and "vg.02" for the StratusLab storage service. You can use other names, but then change the commands below as necessary.

10.6 Package Repositories

The StratusLab installation takes packages from four yum repositories:

1. The standard CentOS repository,
2. The EPEL 6 repository,
3. The StratusLab repository, and
4. The IGTF Root Certificates.

The configuration for the CentOS repository is done when the system is installed. The others require additional configuration.

To configure **both** the Front End and Node for the EPEL repository, do the following:

```
$ wget -nd http://mirrors.ircam.fr/pub/fedora/epel/6/i386/epel-release-6-8.noarch.rpm
$ yum install -y epel-release-6-8.noarch.rpm
```

This will add the necessary files to the `/etc/yum.repos.d/` directory.

To configure **both** the Front End and Node for the StratusLab repository, put the following into the file `/etc/yum.repos.d/stratuslab.repo`:

```
[StratusLab-Releases]
name=StratusLab-Releases
baseurl=http://yum.stratuslab.eu/releases/centos-6.2-v13.02/
gpgcheck=0
```

replacing the URL with the version you want to install.

Although not strictly necessary, it is advisable to clear all of the yum caches and upgrade the packages to the latest versions:

```
$ yum clean all
$ yum upgrade -y
```

This may take some time if you installed the base operating system from old media.

10.7 DNS and Hostname

Ensure that the **hostname** is properly setup on the Front End and the Node. The DNS must provide both the forward and reverse naming of the nodes. This is required for critical services to start.

You can verify this on both the Front End and the Node with the command:

```
$ hostname -f
```

Set the hostname if it is not correct.

Throughout this tutorial we use the variables $FRONTEND_HOST ($FRONTEND_IP) and $NODE_HOST ($NODE_IP) for the Front End and Node hostnames (IP addresses), respectively. Change these to the proper names for your physical machines when running the commands.

10.8 SSH Configuration

The installation scripts will automate most of the work, but the scripts require **password-less root access**:

- From the Front End to each Node and
- From the Front End to the Front End itself

Check to see if there is already an SSH key pair in `/root/.ssh/id_rsa*`. If not, then you need to create a new key pair **without a password**:

```
$ ssh-keygen -q
Enter file in which to save the key (/root/.ssh/id_rsa):
/root/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

Now ensure that you can log into the Front End from the Front End without needing a password. Do the following:

```
$ ssh-copy-id $FRONTEND_HOST
The authenticity of host 'onehost-5.lal.in2p3.fr␣(134.158.75.5)' can't␣be␣establis
RSA␣key␣fingerprint␣is␣e9:04:03:02:e5:2e:f9:a1:0e:ae:9f:9f:e4:3f:70:dd.
Are␣you␣sure␣you␣want␣to␣continue␣connecting␣(yes/no)?␣yes
Warning:␣Permanently␣added␣'onehost-5.lal.in2p3.fr,134.158.75.5'␣(RSA)␣to␣the␣list
root@onehost-5.lal.in2p3.fr's password:
Now try logging into the machine, with "ssh␣'onehost-5.lal.in2p3.fr'", and check i

  .ssh/authorized_keys

to make sure we haven't␣added␣extra␣keys␣that␣you␣weren't expecting.
```

Do the same thing for the node:

```
$ ssh-copy-id $NODE_HOST
...
```

And verify that the password-less access works as expected.

```
$ ssh $FRONTEND_HOST

Last login: Mon May 27 14:26:29 2013 from mac-91100.lal.in2p3.fr
#
# exit
logout
Connection to onehost-5.lal.in2p3.fr closed.

$ ssh $NODE_HOST

Last login: Mon May 27 14:26:43 2013 from mac-91100.lal.in2p3.fr
#
# exit
logout
Connection to onehost-6.lal.in2p3.fr closed.
```

Now that SSH is properly configured, the StratusLab scripts will be able to install software on both the Front End and the Node.

10.9 DHCP Server

A DHCP server must be configured to assign static IP addresses corresponding to known MAC addresses for the virtual machines. These IP addresses must be publicly visible if the cloud instances are to be accessible from the internet.

If an external DHCP server is not available, the StratusLab installation command can be used to properly configure a DHCP server on the Front End for the virtual machines.

This uses a DHCP server on the Front End.

10.10 Network Bridge

A network bridge must be configured on the Node to allow virtual machines access to the internet. You can do this manually if you want, but the StratusLab installation scripts are capable of configuring this automatically.

This tutorial allows the installation scripts to configure the network bridge.

Chapter 11

Couchbase

The StratusLab architecture uses a distributed database at the core to hold the full state of the cloud. This simplifies all of the other components of the system by allowing them to stateless and by acting as a means for coordinating the components of the system.

StratusLab uses Couchbase for this distributed database. It was chosen because of its ability to store and to index JSON-formatted documents efficiency, matching well the needs of managing cloud resources through the CIMI data model. It is easy to deploy for small database, while retaining the ability to scale to very large systems.

Because all other StratusLab components rely on having access to the Couchbase database, it is the first component that must be installed.

11.1 Service Overview

Couchbase is available as a standard RPM package from the Couchbase website. The company distributes two versions of the database from their website: a "community" version and an "enterprise" version.

The community version is open source and released under the Apache 2 license (the same as for StratusLab). StratusLab uses this version for all of its own deployments and tests. For convenience, the RPM package for Couchbase is included in the StratusLab yum repository.

The enterprise version requires the purchase of a commercial license for anything other than small test deployments. However the commercial

support for deployment problems and software bugs may be interesting for those deploying mission-critical cloud infrastructures. StratusLab should work with the enterprise version, but does not require it.

init.d script	couchbase-server
language	erlang
APIs	Java, Python, and C
log file(s)	/opt/couchbase/var/lib/couchbase/logs/*
initial password	/opt/couchbase/cluster-password.txt
8091	web administration port
8092	Couchbase API port
11209*, 11210	internal cluster ports
4369*	Erlang Port Mapper
21100-21199*	Node data exchange

Table 11.1: Couchbase Server Characteristics

All of the listed ports must be open to communication between the nodes participating in the Couchbase cluster. All ports except those with an asterisk must be open to cloud services accessing the database.

Couchbase is written in Erlang, but StratusLab uses the Java and Python APIs to access the database. The Python API depends on the C API, so it must also be installed.

11.2 Installation and Configuration

The usual service mapping puts the Couchbase instances on the “cloud entry point” nodes, along with the CIMI interface. The instances on these nodes form the Couchbase cluster for the full StratusLab cloud. The minimal deployment has only one “cloud entry point” node and hence only one Couchbase instance.

Production StratusLab deployments should have more than one Couchbase instance in the Couchbase cluster for reliability and redundancy.

Although the default location for these instances is on the "cloud entry point" nodes, you can deploy them on dedicated nodes or other cloud services nodes, if needed.

The installation of Couchbase consists of installing the RPM package and then initializing a Couchbase cluster. The StratusLab installation commands automate the process.

Log into the node that will function as the "cloud entry point" node for your cloud infrastructure as "root". Verify that all of the prerequisites detailed in the previous chapter are satisfied.

Install the StratusLab system administrator command line interface (CLI). This installs the commands that simplify and automate the installation of all of the StratusLab components. Assuming that you have already configured the machine for using the StratusLab yum repository (see "Prerequisites"), this should be as simple as:

```
$ yum install -y stratuslab-cli-sysadmin
```

Once this completes, you should have a set of StratusLab commands in your path. All of the StratusLab commands start with the prefix `stratus-`. You may want to look at the help for the `stratus-install` command:

```
$ stratus-install --help

Usage: stratus-install [options]

Install selected services of the StratusLab cloud distribution.
...
```

This is the command that we will use to automate the installation of the cloud services.

We will now use this command to install and initialize the Couchbase database on the machine. Do the following:

```
$ stratus-install --couchbase

...
Starting couchbase-server       [OK]
```

You can get more detailed output if you add the `-vvv` option. This command installs the necessary packages and then setup the database. The last line should indicate that Couchbase has been started; if successful you will see an "OK" indication.

The installation creates an administrator account for the database called 'admin'. The randomly generated password for this account is available in `/opt/couchbase/cluster-password.txt`.

If you wish, you can use the Couchbase web interface to view the current status of the cluster. This interface is available on port 8091, but by default, is only accessible locally on the machine. To view it remotely you will need to tunnel to the machine:

```
$ ssh -L2000:cep.example.org:8091 -N root@cep.example.org
```

in a separate terminal window. You can then connect to the interface on the "http://localhost:2000/" URL. You will have to use the "admin" account with the generated password to log in.

If everything has been installed correctly, you should see a display similar to the screenshot below.

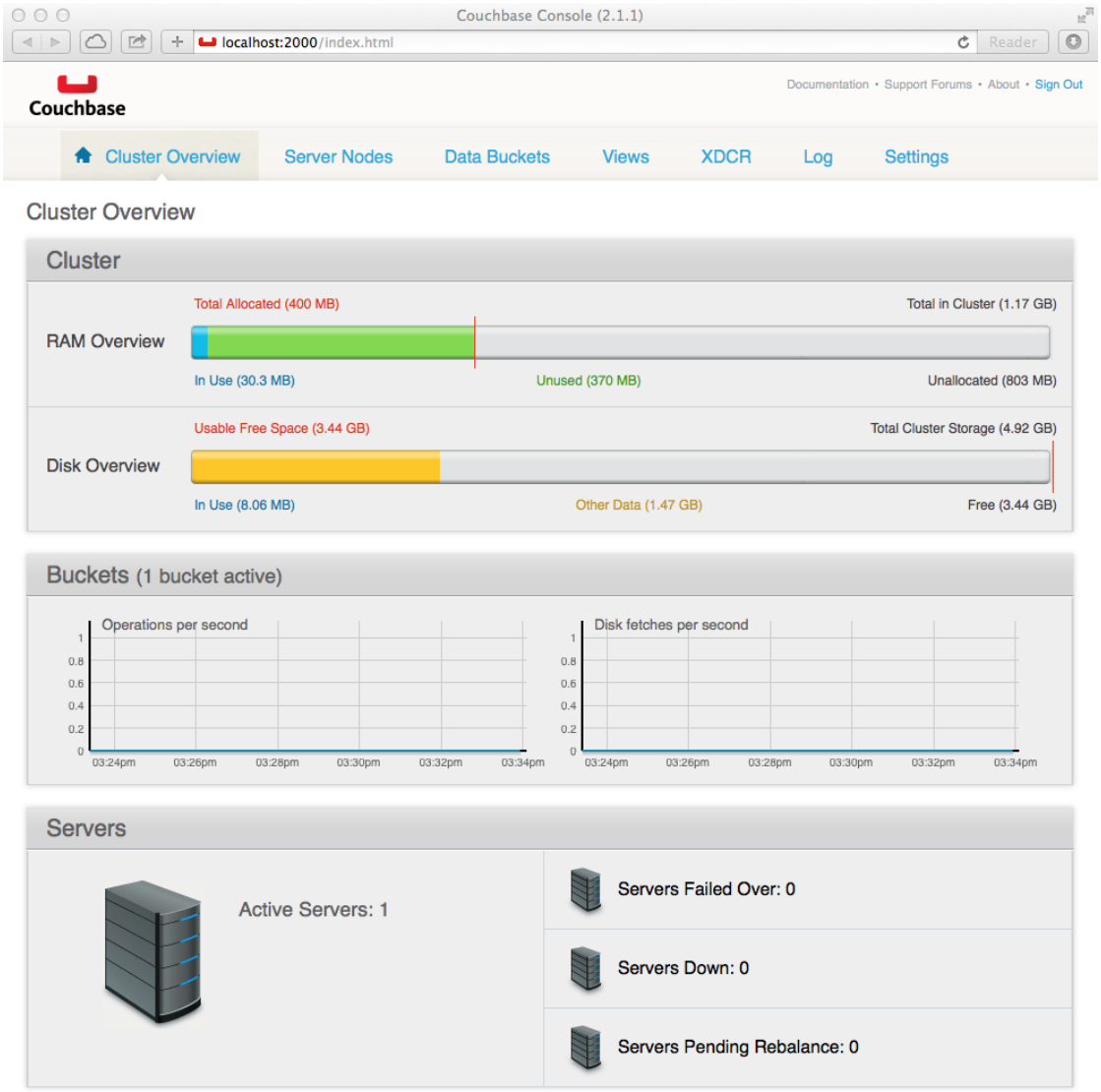

Figure 11.1: Couchbase Management Console

Chapter 12

CIMI

StratusLab uses CIMI as its native user interface. CIMI is a standard from DMTF that provides a coherent, unified, RESTful API. The API covers all of the StratusLab cloud resources, including:

- Virtual machines
- Volumes for data storage
- Machine images or appliances

In addition, StratusLab has extended the interface to provide information to cloud users in "ServiceMessage" resources; it has also been extended to handle resources used to configure the cloud services and authentication mechanisms.

As the CIMI specification follows the usual REST patterns, the standard CRUD (create, read, update, and delete) are represented by:

- Create: HTTP POST to a resource collection URL,
- Read: HTTP GET to a resource URL,
- Update: HTTP PUT to a resource URL, and
- Delete: HTTP DELETE to a resource URL.

All of the resources are represented as JSON documents and contain metadata to allow users to understand what operations they can perform on the resource.

12.1 Service Overview

The CIMI service is the only StratusLab service accessible by users on a cloud infrastructure. All interactions between the user and the underlying resources occur through this service. This service also handles all of the user authentication for the cloud, passing verified authentication information to the underlying services.

The characteristics of the service are summarized in the following table. The service is the primary service on the "cloud entry point" node(s). It runs within a dedicated Jetty web service container.

The service is written in clojure and uses the Ring and Compojure frameworks for implementing the web service and the Friend framework for authentication.

init.d script	cimi
language	clojure (lisp on the Java VM)
APIs	REST, python, and libcloud
log file(s)	/opt/stratuslab/cimi/logs/*
initial password	see `cimi.log` log file
443	port for API and web interface (HTTPS)
80	redirects to secured port (443)

Table 12.1: CIMI Server Characteristics

12.2 Installation

As for all of the StratusLab services, the installation consists of the installation of an RPM package followed by configuration of the service. This process is automated via the `stratus-install` command.

Logged in as "root" on the "cloud entry point" machine, execute the following command to install the CIMI service:

```
$ stratus-install --cimi
```

This should complete without error and start the cimi service. You can check that it is running with the command:

```
$ service cimi status

START_INI      =  /opt/stratuslab/cimi/start.ini
JETTY_HOME     =  /opt/stratuslab/cimi
...

Jetty running pid=2099
```

This gives a lot of information about the service configuration and ends with giving the PID of the running service. If this is not running, then no PID will be given.

12.3 Configuration

12.3.1 Service Certificate

The first time the service starts, it will generate a self-signed certificate for the service. This certificate will not normally be accepted by clients (web browsers or command line interfaces) without a security warning. While this is fine for testing, a production service should use a certificate signed by an accepted certificate authority.

To install a certificate, create a java keystore that contains your certificate and key using the java `keytool` utility. Use "jettycred" as the password for the keystore. Put this keystore onto the machine in the file `/etc/stratuslab/cimi/etc/jetty.jks`. Restart the service to make the service use the new certificate.

You can actually put the keystore anywhere on the machine and use any password that you would like. However, if you use non-standard values, then you will need to update the file `/opt/stratuslab/cimi/start.d/50-ssl.ini`. This file will be overwritten on updates, so you will need to modify this file after each upgrade.

12.3.2 Trusted Certificate Authorities

By default, the installation procedure will install the set of Certificate Authorities trusted by the European Grid Infrastructure (EGI) as well as

the Virtual Organizations (VOs) that use that infrastructure. The files for these CAs and VOs are installed in the `/etc/grid-security/` directory. If these CAs are acceptable, then all of the necessary configuration has been done for you.

> **NOTE**: Before the service will accept connections with grid certificates, the **certificate revocation lists (CRLs) must be generated**. Do this by hand, by executing the command in the `/etc/cron.d/fetch-crl-cron` file. This will generally take a few minutes to complete. Until the CRLs are generated, connections using a client certificate will fail.

If you do not want to trust these CAs and want to use the default set of commercial CAs trusted by the java distribution, then you must modify one of the start up files for the CIMI server. Edit the file `/opt/stratuslab/cimi/start.d/50-ssl.ini`:

```
#=========================================================
# SSL Context
# Create the keystore and trust store for use by
# HTTPS and SPDY
#---------------------------------------------------------
jetty.keystore=etc/jetty.jks
jetty.keystore.password=jettycred
jetty.keymanager.password=jettycred
jetty.truststore=etc/jetty.jks
jetty.truststore.password=jettycred
jetty.secure.port=443
etc/jetty-grid-ssl.xml
```

Change the last line to “etc/jetty-ssl.xml” and restart the server. Note that this configuration change will need to be reapplied for each service upgrade.

12.3.3 Administrator Account

The details for configuring the authentication for the service are explained in the next chapter. For now, it is enough to know that an administrator account is created the first time the service starts. The username is “admin”; the randomly-generated password is available in the service log `/etc/stratuslab/cimi/logs/cimi.log`.

12.4 Testing the CIMI Service

The "CloudEntryPoint" resource as well as a few others are visible to anyone, even those without an account on the cloud. We can verify that the service is working correctly by retrieving the CloudEntryPoint.

To do this via the command line, just use `curl` on the base URL of the service.

```
$ curl -s --insecure https://cimi.example.org/ \
    python -mjson.tool

{
    ...

    "baseURI": "https://onevm-142.lal.in2p3.fr:443/",
    "created": "2013-11-12T16:10:47.990Z",
    "id": "CloudEntryPoint",
    "jobs": {
        "href": "Job"
    },
    "machineConfigs": {
        "href": "MachineConfiguration"
    },
    "resourceURI": "http://schemas.dmtf.org/cimi/1/CloudEntryPoint",

    ...
}
```

This resource (in JSON format) contains the list of all of the cloud resource collecitons supported by this cloud infrastructure, along with relative URLs (in the "href" field) for those resource collections. It also contains metadata concerning the cloud infrastructure itself.

> **NOTE**: The first access to the server takes some time to respond because the server is dynamically compiling the source clojure files and initializing the database. Subsequent accesses to the service should be much faster.

There is also a rudimentary web browser interface provided by the service. Point a browser at the URL http://cimi.example.org/webui, replacing the hostname with your own. You should see an HTML representation of the CloudEntryPoint as in the following screenshot.

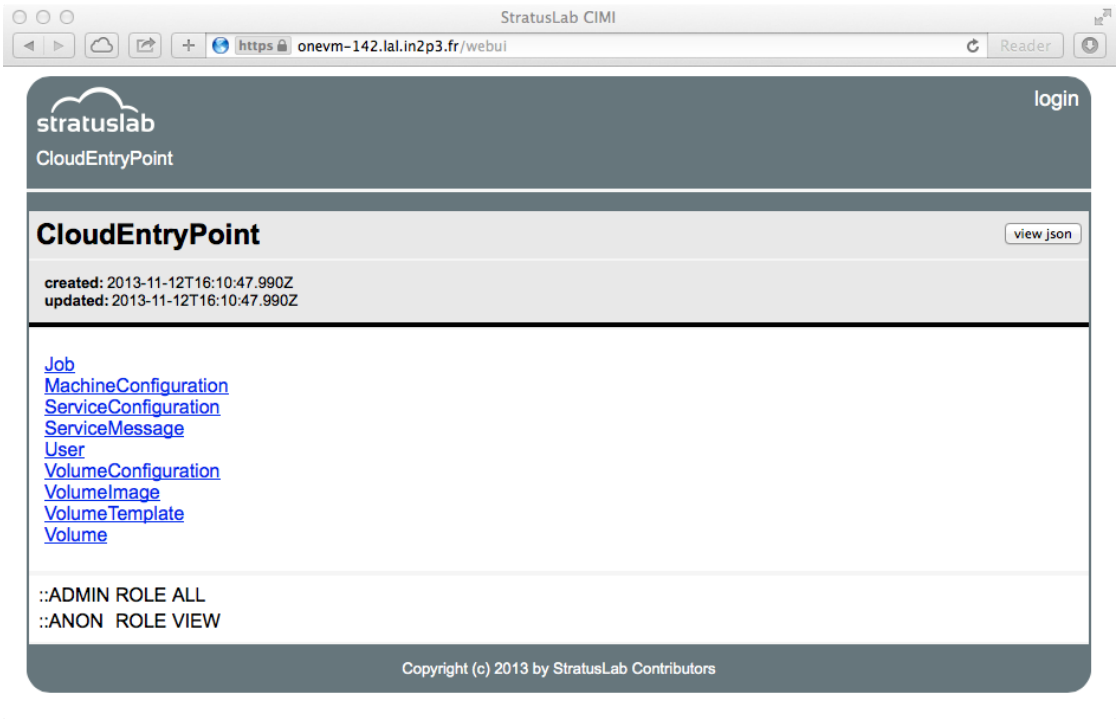

Figure 12.1: CloudEntryPoint Viewed in CIMI Web Browser Interface

12.5 Verify Administrator Account

You will be using the administrator account to update the service configuration. To verify that it works, first recover the administrator's account password from the service log. You should find a message in the log like the following:

```
... User/admin entry created; initial password is 6GfRtIeWVygK
```

The username of this initial account is always "admin"; the "6G. . . " value is the generated password. Use the value given in your log file.

To login as the administrator from the web interface, click on the "login" link in the upper right corner, fill in the username and password on the form, and then click the "login" button. If the login was successful, then you should be redirected back to the CloudEntryPoint, but you will see your login information on the right side of the header.

> **NOTE**: You can always see your full authentication information by visiting the URL https://cimi.example.org/authn. The most

Figure 12.2: Logged in User Information

important fields are the “identity” field (giving your username) and the “roles” field (giving your authorizations).

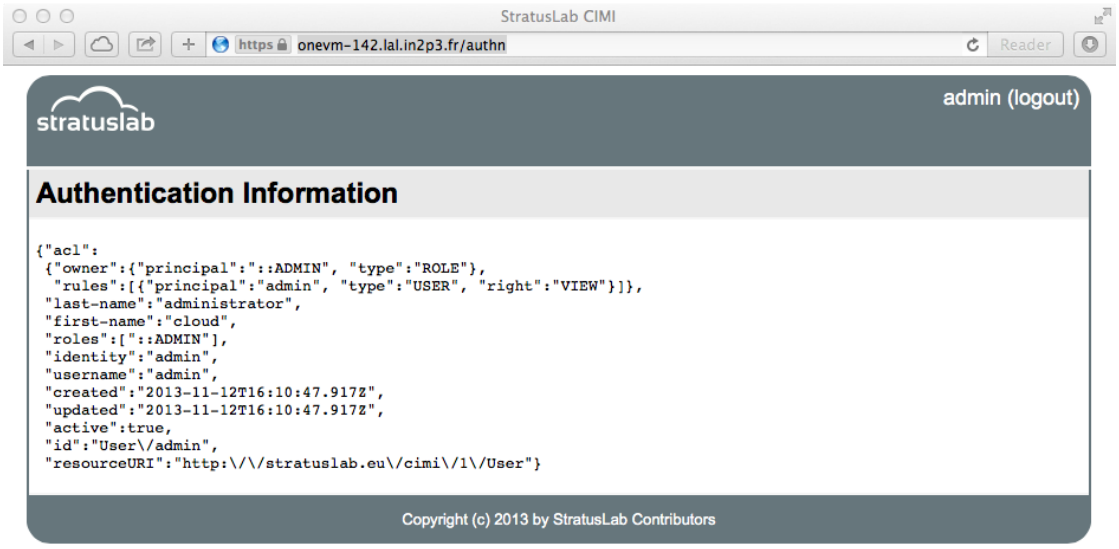

Figure 12.3: Full Authentication Information

If you can see pages similar to the screenshots, the administrator of the CIMI is correctly configured. However, you will likely want to **change the password of the administrator account**. Now that you are logged into the server, you can do this.

Return to the CloudEntryPoint using the web browser interface (i.e. the URL ending with “webui”). From there, click on “User”. This brings up the list of user records; only the “admin” account should be listed. Then click on “admin” to view the user record. You should see a page listing characteristics of the “admin” account, notably there will be a field “password” containing the bcrypt hash of the current administrator password.

You should see three buttons on the right of the page: “view json”, “edit”, and “delete”. You will want to click on the “edit” button which will bring

up a JSON editor with the current contents of the “admin” user.

However, before doing this, you want to generate the bcrypt hash for a new (memorable) password. This can be done with python using the following command:

```
$ python -c "
> import bcrypt
> h=bcrypt.hashpw('hello', bcrypt.gensalt())
> print h
> "
$2a$12$zvS7axGrws6/YH2AuIyXpufc174KV5bjBTp.vo400sGZsehP7CpFS
```

You may have to install the package “py-bcrypt” on CentOS for this to work. It returns the hash of your password. Change the ‘hello’ in the example to the password you want to use.

Now that you have a new password, click on the “edit” button, change the value of the password field to the hash value you’ve generated and click on the “save” button. You should be redirected back to the same page, but the password field will have been updated.

You can now logout via the “logout” link and log back into the service (with your new password!) using the same procedure as before.

12.6 Service Messages

As a further example of how to use the web interface (which you will use to handle service configuration), you can create “ServiceMessage” resources on the server.

The ServiceMessage resources are visible to anyone but can only be created by the administrator. These messages are intended to provide general service information to users, like the MOTD (message of the day) text on many operating systems.

From the CloudEntryPoint, click on the “ServiceMessage” link. This should bring up an empty list of ServiceMessage resources. Click on the “add” button, which will bring up the same JSON editor you saw previously.

Add something like the following to the editor panel:

```
{
  "name": "StratusLab is Alive!",
 "description": "Deploying StratusLab clouds is fun."
```

```
}
```

and then click on the “save” button. You should then see a summary panel of the message along with metadata that was added to the entry. You can view JSON for the entry with the “view json” button, update it with the “edit” button, or delete it with the “delete” button.

If you go back to the ServiceMessageCollection, you will see the entry in the list.

For ServiceMessage resources the “name” field is treated like a title and the “description” gives the full message.

Chapter 13

Authentication

As mentioned in the previous chapter, the CIMI server also handles all of the user authentication for the cloud. The server supports the use of internal or external databases of users.

The internal database, with user records stored in Couchbase, is convenient because all information is contained within the cloud infrastructure and user records can be managed in the same way as other configuration files.

Using an external database, such as LDAP, allows the user information to be exported to other system or allows better integration of the cloud with other services in the data center. In the case of VOMS proxies, this also delegates some authority to third parties to manage the users associated with a particular Virtual Organization.

13.1 Overview

There are two aspects to the authentication configuration for a StratusLab cloud: defining the methods to be used and managing the users.

To define the authentication mechanisms for the cloud infrastructure, you must create the document "ServiceConfiguration/authn" in the Couchbase database. You can use the same web interface shown earlier to browse to the "ServiceConfiguration" collection and then to add the new document.

The document must follow the defined schema:

```
{
  "service": "authn",
  "localdb":
    {
      "password-enabled": true,
      "cert-enabled": false
    },
  "ldap":
    {
      "password-enabled": false,
      "cert-enabled": false,
      ...
    },
  "voms":
    {
      "enabled": false
    }
}
```

where the “service” and “localdb” fields are required (and the value for the “service” field must be “authn”!). The fields “ldap” and “voms” are optional. The details of the “ldap” map are given below.

If the configuration document does not exist, then the default is to only use the local user database with password credentials. It is recommended to always keep this method active along with at least one administrative account defined in the database.

> **NOTE**: Enabling or disabling authentication methods requires that the CIMI service be restarted.

Adding or removing users from the system, requires changes to the local or external user databases. The details on this are given in the following sections.

> **NOTE**: Adding, removing, or modifying users does **not** require restarting the CIMI service. These changes will be taken into account immediately for all future authentication actions.

You can also define “roles” for the users to either indicate group membership or rights to perform particular actions. There are three special roles “::ADMIN”, “::USER”, and “::ANON”. The first confers administrative rights to the user; users with this role will have complete control over the cloud configuration. The “::USER” role is given to anyone who has been authenticated. The “::ANON” role is used for users that have not been authenticated.

13.2 Couchbase User Entries

You can manage users directly in the Couchbase database via the CIMI interface. There is a "User" collection and you can add, modify, and remove entries from the system with the same web browser interface you've used for the other resources.

The schema for user records is:

```
{
  "last-name": "required",
  "first-name": "required",
  "username": "required, must be unique",
  "password": "optional, value is bcrypt hash",
  "enabled": true,
  "roles": [ "list", "of", "roles" ],
  "altnames": {
    "x500dn": "DN of user in RFC2253 format"
  }
}
```

If the "enabled" field is not supplied, the default value is "false". The "password" is used only for password authentication; similarly, the "x500dn" field is only used for certificate authentication.

The schema is a "loose" one, meaning that other fields may be added if you want to track additional information. One example would be an "email" field.

13.2.1 Using Passwords

To use password authentication, the "localdb/password-enabled" flag must be true in the "ServiceConfiguration/authn" document. The user record for the given username must exist, must be enabled, and must have a password set.

The value of the password field is the bcrypt hash of the clear text password. See the preceeding text for generating the bcrypt hash.

13.2.2 Using Certificates

To use certificate authentication, the "localdb/cert-enabled" flag must be true. The field "x500dn" must exist and have the RFC2253-formatted

DN of the user's certificate. The user can use the raw certificate or a proxy generated from that certificate. The user will be identified with the value of the "username" field in the user record.

For certificate authentication to work, the SSL configuration for the Jetty server must be done. By default, the configuration for trusting the EGI Certificate Authorities is done by the `stratus-install` command. If you want to trust different Certificate Authorities, you must adjust the SSL configuration. See the previous chapter for what needs to be done.

13.3 LDAP User Database

To authenticate users against an LDAP database, you must have deployed and populated an LDAP server. The authentication configuration for using LDAP is quite flexible, so nearly any standard layout for the information should work.

The "ldap" field in the "ServiceConfiguration/authn" document must follow the schema:

```
"ldap": {
  "password-enabled": true,
  "cert-enabled": false,

  "connection": {
    "host": {
      "address": "localhost",
      "port": 389
    },
    "ssl?": false
  },

  "user-object-class": "inetOrgPerson",
  "user-base-dn": "ou=users,o=cloud"
  "user-id-attr": "uid",

  "role-object-class": "groupOfUniqueNames",
  "role-base-dn": "ou=groups,o=cloud",
  "role-member-attr": "uniqueMember",
  "role-name-attr": "cn",

  "skip-bind?": false,
}
```

All of the fields are required. The "connection" values provide the location of the server. The "user-" fields indicate what objects in the LDAP

database are user records and the "role-" fields indicate the groups that are mapped to roles.

13.3.1 Using Passwords

The "skip-bind?" field indicates whether to try to bind to the database with the given user password to authenticate the user. Normally, for password authentication this should be false and the LDAP server should be setup to allow only the user to view her user record.

13.3.2 Using Certificates

This is not supported in the current version, but is planned for a future version.

13.4 VOMS Proxy Authentication

VOMS proxies are a mechanism by which users can delegate some rights to a third party such as a service. These proxies also convey certain rights associated with a user who belongs to a Virtual Organization. The Virtual Organization itself manages its membership, defines the rights, and allocates those rights to members.

By default, the SSL configuration of the server will be setup to allow validation of VOMS proxies according to the policies of the European Grid Infrastructure. You must use this configuration (or a similar one if you are an expert) to use VOMS proxy authentication.

To enable this, set the "voms/enabled" flag to true in the authentication configuration. You must then add a pseudo-user entry in the local database for each Virtual Organization you want to support.

An example pseudo-user entry for the "vo.lal.in2p3.fr" Virtual Organization is:

```
{
  "last-name": "Virtual␣Organization",
  "first-name": "LAL",
  "username": "vo:vo.lal.in2p3.fr",
  "enabled": true
}
```

It is important that the username contain the "vo:" prefix to the name of the Virtual Organization that is being authorized. The entry should *not* have a password associated with it.

The "username" associated with users identified via VOMS proxies is the DN of their certificate. The FQANs (fully-qualified attribute names) of the VOMS certificate are mapped to roles in the StratusLab authentication.

13.5 Future Authentication Methods

The authentication framework used by StratusLab is extensible allowing different mechanisms to be incorporated fairly easily. Candidates for additional mechansims to support in the future include OAUTH2 and Shibboleth. Feedback on the desire for these (or other) mechanisms is welcome.

Chapter 14

Using Ceph

Ceph is a set of storage technologies that allow object, block, and file storage that is reliable and scalable while maintaining high-performance.

The block storage component of Ceph can be used as a storage backend to the persistent disk service.

14.1 Requirements

- A working Ceph cluster
- A Ceph pool for pdisk
- A a manager identity for this pool

To create the pool: ceph osd pool create ${poolname} 128 128

To create the manager identity for the pool:

```
ceph auth get-or-create client.${identity} \
  mon 'allow r' osd \
  'allow class-read object_prefix rbd_children, allow rwx pool=${poolname}'
```

14.2 Modifications

Some modifications are needed for the persistent disk server and the VM hosts.

- Linux kernel 3.x
- Ceph package
- Ceph configuration in `/etc/ceph/ceph.conf`
- Manager keyring in `/etc/ceph/ceph.client.${identity}.keyring`
- Parameters in `pdisk-backend.cfg` and `pdisk-host.conf`

For testing, you can use a package provided by ELrepo repository which provides a 3.x kernel for CentOS v6.4.

Currently, using a proxy server is not very useful for this backend because the persistent disk server needs a 3.x kernel and the Ceph package to attach pdisks to itself when importing a remote appliance (cf. DiskUtils.copyUrlToVolume).

There is not automatic configuration via the StratusLab installer yet. This is being worked on.

The `oneadmin` user must also be able to execute Ceph commands as root. Update the `/etc/sudoers` file appropriately:

```
# /etc/sudoers
oneadmin ALL= NOPASSWD: /usr/bin/rbd
```