StratusLab User's Guide

StratusLab Collaboration

Version 13.05.0.RC1

Contents

Chapter 1

Preface

1.1 Organization

1.2 Typographic Conventions

Chapter 2

Introduction

Describe all of the services and how they fit together. Provide an overall view of a cloud deployment and the terminology.

The tools for end-users include:

- Pyton CLI client
- Web interface
- SlipStream

Programmatic access to the StratusLab service for application and platform developers includes:

- StratusLab python API
- Libcloud API
- Direct use of REST interfaces (CIMI)

Chapter 3

Quick Start

How to get up and running as quickly as possible with the command line client.

3.1 Prerequisities

- Existing installation of virtualenv and pip
- Availability of SSH client
- SSH key pair
- Active StratusLab account

3.2 Procedure Overview

- Installation of python CLI client
- Searching the Marketplace
- Running a virtual machine
- Creating and using a disk

Chapter 4

Command Line Client Installation

StratusLab provides a simple command line client that is easy to install on all platforms. This client provides access to all of the StratusLab services; it provides a scriptable alternative to access via a web browser.

After installation and configuration of the client follow the instructions to launch a Virtual Machine and then, read the in-depth documentation on the use of the StratusLab cloud.

4.1 Prerequisites

- Python >= 2.6 and < 3.x
- Java 1.6+
- SSH client
- SSH user key-pair

4.1.1 Python

Ensure that you have a recent version of Python **>= 2.6**, but **< 3.0** installed on your system. From the command line type

```
$ python -V
  Python 2.6.1
```

See the python site for Python downloads and instructions.

4.1.2 SSH Setup

In most of the cases SSH will be used to log into virtual machines that have been started in the cloud. Ensure that you have SSH public and private key pair.

Linux/UNIX

The private and public keys are usually located under `~/.ssh/` and have names like `id_rsa` and `id_rsa.pub`.

You can generate them with the following command

```
$ ssh-keygen
```

The default values are appropriate in most cases, but you should provide a passphrase and not leave it empty.

Verify the generated key pair permissions. The `id_rsa` should have permissions 0600 (read/write access for owner only) and the `id_rsa.pub` - 0644 (read access for all; write access for owner).

Be sure to remember the passphrase that you have used!

Be careful if an ssh agent is configured by default for your operating system. Ensure that it is setup to use the correct key and that it provides the correct password for that key.

Windows

- You need to generate an SSH key pair on a linux or Unix system.
- Import the private key into Putty

To generate an SSH key pair on a linux or Unix system, follow the above instructions described in the Linux/UNIX part above. In your Windows machine, install Putty and PuttyGen from page.

To import your id_rsa file to Putty: 1. Start PuttyGen, 2. Click “Load”, and browse to your id_rsa file, 3. Click “Save private key”. Your private key will be saved in the format required by Putty.

To log in your virtual machine using Putty: 1. Start Putty, 2. In “session” category provide the Host Name or IP address 3. In Connection/SSh/Auth category, in “Private key for authentication” field, browse to your private key. 4. Open

More information on how to “Connecting to Linux/UNIX Instances from Windows Using PuTTY” can be found on this page

4.2 Individual Client Installation

Look for the latest version of the command-line client tarball `stratuslab-cli-user-*.tar.gz` in the yum repositories. (Even though this is a CentOS repository, **all of the tarballs and zip files work on all platforms**.) Unpack the tarball in a convenient location.

Update your `PATH` and `PYTHONPATH` variables:

```
export PATH=$PATH:<install location>/bin
export PYTHONPATH=$PYTHONPATH:<install location>/lib/
   stratuslab/python
```

The above are appropriate for Bourne-type shells. Modify the commands as necessary if you are using a different shell.

4.3 System Wide Client Installation

These packages only work on RedHat Enterprise Linux systems (and its derivatives like CentOS and ScientificLinux) and OpenSuSE. You must have root access to your machine to install them.

For RHEL and RHEL-like systems, it is recommended to do the installation with yum. The configuration for yum can be found here, choosing the "centos-6.2" repository. Execute:

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

to install the latest version of the client tools.

For OpenSuSE, configure zypper for the StratusLab OpenSuSE repository ("opensuse-12.1") and use it to install the package.

4.4 Client Configuration

One has to know at least one endpoint of the StratusLab cloud deployment and possess valid credentials to access it. Refer to StratusLab Reference infrastructure.

4.4.1 Credentials

For working with user command-line tools one needs to possess the following credentials (requirement depends on use-case and utility used)

```
username/password of the account in StratusLab Cloud
   frontend(s)
X509 key/pair
Globus/VOMS proxy
PKCS12 certificate
```

For more see documentation on user credentials.

4.4.2 Configuration file

Configuration file should contain definitions of StratusLab services endpoints and credentials required for the user client. For example

```
endpoint = cloud.lal.stratuslab.eu
pdisk_endpoint = pdisk.lal.stratuslab.eu
username = clouduser
password = cloudpass
pem_certificate = /Users/localuser/.globus/usercert.pem
pem_key = /Users/localuser/.globus/userkey.pem
```

Linux/UNIX

By default the user client expects the configuration file at `$HOME/.stratuslab/stratuslab-user.cfg`.

The client ships with a reference configuration file

- in tarball <install location>/conf/stratuslab-user.cfg.ref
- in RPM /etc/stratuslab/stratuslab-user.cfg.ref

User has to copy the file to the default location `$HOME/.stratuslab/stratuslab-user.cfg` and modify it following explanations to the variables in the file. The variables that are commented out (e.g. p12_certificate) take their default values from the code.

Windows

By default the user client expects the configuration file at `%HOMEDRIVE%%HOMEPATH%\.stratuslab\stratuslab-user.cfg`.

```
mkdir %HOMEDRIVE%%HOMEPATH%\.stratuslab
```

The reference configuration file

- in zip package `<install location>\conf\stratuslab-user.cfg.ref`

User has to copy the file to the default location `%HOMEDRIVE%%HOMEPATH%\.stratuslab\stratuslab-user.cfg` and modify it following explanations to the variables in the file. The variables that are commented out (e.g. p12_certificate) take their default values from the code.

4.5 Providing user parameters

The following are they means of providing user parameters and presented in the order of precedence

- via command line parameters
- via environment variables (STRATUSLAB_*)
- in configuration file (by default: HOME/.stratuslab/stratuslab-user.cfg)
- default in the code

4.6 Cloud Endpoint

Used for VMs instantiation. There is no default value for the endpoint in the code.

Command-line parameters

```
--endpoint
--username
--password
```

Environment variables

```
STRATUSLAB_ENDPOINT
STRATUSLAB_USERNAME
STRATUSLAB_PASSWORD
```

Configuration file

```
endpoint
username
password
```

4.7 Persistent Disk service

Used for manipulation machine and disk images (create, delete, hot-attach/detach) in the Persisten Disk storage service. There is no default for PDisk endpoint in the code. If not provided, the client will use the value of 'endpoint'.

Command-line parameters

```
--pdisk-endpoint
--pdisk-username
--pdisk-password
```

Environment variables

```
STRATUSLAB_PDISK_ENDPOINT
STRATUSLAB_PDISK_PROTOCOL  # default 'https'
STRATUSLAB_PDISK_PORT      # default '8445'
STRATUSLAB_PDISK_USERNAME
STRATUSLAB_PDISK_PASSWORD
```

Configuration file

```
pdisk_endpoint
pdisk_port
```

NB! If 'pdisk-username' and/or 'pdisk-password' are not provided by any of the means the ones defined for 'endpoint' will be used.

4.8 Marketplace

Marketplace endpoint to be used when uploading metadata, and/or when creating instance of VM.

Default value for Marketplace endpoint in the code

```
https://marketplace.stratuslab.eu
```

Command-line parameter

```
--marketplace-endpoint
```

Environment variable

```
STRATUSLAB_MARKETPLACE_ENDPOINT
```

Configuration file

```
marketplace_endpoint
```

4.9 Private/Public keys

4.9.1 SSH

SSH public key file. Used at VM instantiation. The public key is pushed to the VM to enable password-less SSH access. Default: **HOME/.ssh/id_rsa.pub**.

Command-line parameter

```
--key
```

Environment variable

```
STRATUSLAB_KEY
```

Configuration file

```
user_public_key_file
```

4.9.2 PEM

Used to authenticate to the cloud endpoint. If you want to use authentication with grid certificate and if username/password and PEM key/cert are both enabled, PEM key/cert takes precedence.

Default: **HOME/.globus/userkey.pem**, **HOME/.globus/usercert.pem**

Command-line parameters

```
--pem-cert
--pem-key
```

Environment variables

```
STRATUSLAB_PEM_CERTIFICATE
STRATUSLAB_PEM_KEY
```

Configuration file

```
pem_key
pem_certificate
```

4.9.3 PKCS12

PKSC12-formatted certificate, for signing and validating image metadata.

Default: **HOME/.globus/usercert.p12**

Command-line parameters

```
--p12-cert
--p12-password
```

Environment variables

```
STRATUSLAB_P12_CERTIFICATE
STRATUSLAB_P12_PASSWORD
```

Configuration file

```
p12_certificate
p12_password
```

4.10 Custom sections in configuration file

In the configuration file it is possible to create uniquely named user specific sections to define any of the variables described above. Example of 'endpoint'

```
[my-section]
endpoint = <another.cloud.frontend.hostname>
username = <another.username>
password = <another.password>
```

which can be activated using the **selected_section** parameter in the **default** section of the configuration file

```
selected_section = my-section
```

or using command-line option

```
--user-config-section or -S
```

or using environment variable

```
STRATUSLAB_USER_CONFIG_SECTION
```

Example. Configuration file defines custom 'fav-cloud' section with endpoint/credentials and custom SSH key

```
[fav-cloud]
endpoint = favorit-cloud.tld
username = clouduser
password = cloudpass
user_public_key_file = /home/clouduser/.ssh/id_rsa-
    favcloud.pub
```

Launch a ttylinux image on the cloud endpoint defined by 'fav-cloud' section

```
stratus-run-instance -S fav-cloud $TTYLINUX_IMAGE
```

Chapter 5

Web Interfaces

- Marketplace
- Storage

Chapter 6

Virtual Machine Lifecycle

6.1 Using the CLI

6.1.1 Launch

Configure Cloud endpoint and credentials in the configuration file (default: `$HOME/.stratuslab/stratuslab-user.cfg`).

Search for an image in StratusLab Marketplace and copy the image **identifier**. Say we searched for **ttylinux** and chose **BN1EEkPiBx87_uLj2-sdybSI-Xb**.

Launch an instance of the image

```
$ stratus-run-instance BN1EEkPiBx87_uLj2-sdybSI-Xb

  ::::::::::::::::::::::::
  :: Starting machine(s) ::
  ::::::::::::::::::::::::
  :: Starting 1 machine
  :: Machine 1 (vm ID: 5507)
         Public ip: 134.158.75.75
  :: Done!
```

6.1.2 Connect

Check the VM is up and running

```
$ stratus-describe-instance 5507
```

```
id   state     vcpu memory    cpu% host/ip
                   name
5507 Running   1    131072    4    vm-75.lal.stratuslab.
   eu one-5507
```

Connect to the VM

```
$ ssh root@vm-75.lal.stratuslab.eu
# uname -a
Linux ttylinux_host 2.6.38.1 #1 SMP PREEMPT Tue Apr 10
   21:55:48 MST 2012 x86_64 GNU/Linux
# logout
Connection to vm-75.lal.stratuslab.eu closed.
$
```

6.1.3 Terminate

```
$ stratus-kill-instance 5507
```

Chapter 7

Storage Management

Persistent Disk Storage is the StratusLab service that physically stores machine and disk images as volumes on the cloud site. It facilitates quick startup of VMs and hot-plugging of disk volumes as block devices to the VMs.

7.1 Create persistent disk

Before creating persistent disks (or volumes)[1], you should define the persistent disk storage endpoint by either of

- in your HOME/.stratuslab/stratuslab-user.cfg
- set the environment variable STRATUSLAB_PDISK_ENDPOINT
- pass it as **–pdisk-endpoint** command when creating one.

Lets create a persistent disk of 5GB and named “myprivate-disk”

```
$ stratus-create-volume --size=5 --tag=myprivate-disk
DISK 9c5a2c03-8243-4a1b-a248-0f0d22d948c2
```

The command returned the UUID of the created persistent disk. Newly created disk is by default

- private - can be read, written and deleted only by their owner
- of type **read-write data image**

[1] “disk” and “volume” are used interchangeably.

To create public persistent disk, pass --public argument to stratus-create-volume

```
$ stratus-create-volume --public --size=10 --tag=mypublic
   -disk
DISK d955fda6-bf9c-4aa8-abc4-5bbcdb83021b
```

or update the disk with

```
stratus-update-volume --public <UUID>
```

To get the list of available disk types and the properties that can be updated on the disk run

```
stratus-update-volume -h
```

7.2 List persistent disks

stratus-describe-volumes allows you to query the list of all your public and private persistent disks, and also all the public persistent disks created by other users.

```
$ stratus-describe-volumes
:: DISK a4324f26-39e0-4965-8c8f-3287cd0936e5
        created: 2011/07/20 16:37:10
        visibility: public
        tag: mypublic-disk
        owner: testor2
        size: 5
        users: 0
:: DISK 9c5a2c03-8243-4a1b-a248-0f0d22d948c2
        created: 2011/07/20 16:10:37
        visibility: private
        tag: myprivate-disk
        owner: testor1
        size: 5
        users: 0
:: DISK d955fda6-bf9c-4aa8-abc4-5bbcdb83021b
        created: 2011/07/20 16:26:31
        visibility: public
        tag: mypublic-disk
        owner: testor1
        size: 5
        users: 0
```

The above command lists ‘testor1’ public and private persistent disks, and also ‘testor2’ public ones.

7.3 Using Persistent Disks

Workflow:

- Launch a virtual machine instance referencing a persistent disk
- Format the disk in the running VM
- Write data to the disk
- Dismount the disk or halt the machine instance
- Disk with persistent data is available for use by another VM
- Launch another VM referencing the modified persistent disk

7.3.1 Launch VM with a persistent disk attached

To Launch a VM we will be using the ttylinux image identifier GOaxJFdoEXvqAm9ArJgnZ0_ky6F from StratusLab Marketplace (default one).

--persistent-disk=UUID option when used with stratus-run-instance, tells StratusLab to attach the referenced persistent disk(UUID) to the VM.

Instantiate ttylinux image with reference to your private persistent disk 9c5a2c03-8243-4a1b-a248-0f0d22d948c2.

```
$ stratus-run-instance \
    --persistent-disk=9c5a2c03-8243-4a1b-a248-0
       f0d22d948c2 \
    GOaxJFdoEXvqAm9ArJgnZ0_ky6F

 ::::::::::::::::::::::::
 :: Starting machine(s) ::
 ::::::::::::::::::::::::
 :: Starting 1 machine
 :: Machine 1 (vm ID: 3)
        Public ip: 134.158.75.35
```

Log into your VM using ssh, depending in the linux kernel and distribution version of your VM, your persistent disk will be referenced as /dev/hdc or /dev/sdc.

In ttylinux, it will be /dev/hdc.

Make sure that your disk was attached to your VM

```
$ ssh root@134.158.75.35
# fdisk -l
.........
Disk /dev/hdc: 5368 MB, 5368709120 bytes
255 heads, 63 sectors/track, 652 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
.........
```

7.3.2 Format attached disk

```
# mkfs.ext3 /dev/hdc
```

7.3.3 Mount disk, store data, unmount disk

```
# mount /dev/hdc /mnt
# echo "Testing␣Persistent␣Disk" > /mnt/test_pdisk
# umount /mnt
```

Your persistent disk is ready to be used by another VM.

7.3.4 Launch VM with the modified persistent disk attached

Instantiate new VM ttylinux with the same reference to your private persistent disk 9c5a2c03-8243-4a1b-a248-0f0d22d948c2.

```
$ stratus-run-instance \
    --persistent-disk=9c5a2c03-8243-4a1b-a248-0
       f0d22d948c2 \
    GOaxJFdoEXvqAm9ArJgnZ0_ky6F

 ::::::::::::::::::::::::
 :: Starting machine(s) ::
 ::::::::::::::::::::::::
 :: Starting 1 machine
 :: Machine 1 (vm ID: 4)
        Public ip: 134.158.75.36
```

Log into your VM using ssh, verify existence of your persistent disk

```
$ ssh root@134.158.75.35
# fdisk -l
...........
Disk /dev/hdc: 5368 MB, 5368709120 bytes
255 heads, 63 sectors/track, 652 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
...........
```

Mount your persistent disk

```
# mount /dev/hdc /mnt
# ls /mnt
lost+found  test_pdisk
# cat /mnt/test_pdisk
Testing Persistent Disk
```

7.4 Hot-plug Persistent Disks

StratusLab storage also provides hot-plug feature for persistent disk. With stratus-attach-instance you can attach a volume to a running machine and with stratus-detach-instance you can release it.

To use the hot-plug feature, the running instance needs to have acpiphp kernel module loaded. Image like ttylinux doesn't have this feature, you have to use base image like Ubuntu, CentOS or Fedora.

Before hot-plugin a disk, make sure acpiphp is loaded. In your VM execute

```
modprobe acpiphp
```

To attach two volumes to the VM ID 24 with the UUIDs 1e8e9104-681c-4269-8aae-e513c6723ac6 and 5822c376-9ce1-434e-95d1-cdaa240cd47c

```
$ stratus-attach-volume -i 24 1e8e9104-681c-4269-8aae-
   e513c6723ac6 5822c376-9ce1-434e-95d1-cdaa240cd47c
ATTACHED 1e8e9104-681c-4269-8aae-e513c6723ac6 in VM 24 on
    /dev/vda
ATTACHED 5822c376-9ce1-434e-95d1-cdaa240cd47c in VM 24 on
    /dev/vdb
```

Use the fdisk -l command as above to see the newly attached disks.

Make sure to unmount any file systems on the device within your operating system before detaching the volume. Failure to unmount file systems, or otherwise properly release the device from use, can result in lost data and will corrupt the file system.

```
umount /dev/vda
umount /dev/vdb
```

When you finish using your disks, you can detach them from the running VM

```
$ stratus-detach-volume -i 24 1e8e9104-681c-4269-8aae-
   e513c6723ac6 5822c376-9ce1-434e-95d1-cdaa240cd47c
DETACHED 1e8e9104-681c-4269-8aae-e513c6723ac6 from VM 24
   on /dev/vda
DETACHED 5822c376-9ce1-434e-95d1-cdaa240cd47c from VM 24
   on /dev/vdb
```

On running instance detaching not hot-plugged disks or disks that were not or are no longer attached to the instance will result in an error

```
$ stratus-detach-volume -i 41 2a17226f-b006-45d8-930e-13
   fbef3c6cdc
DISK 2a17226f-b006-45d8-930e-13fbef3c6cdc: Disk have not
   been hot-plugged
```

If you have attached the volume at instance start-up, it cannot be detached while the instance is in the 'Running' state. To detach the volume, stop the instance first.

7.5 Delete Persistent Disks

To delete a persistent disk use the stratus-delete-volume command, note that you can delete only your disks.

To delete 'myprivate-disk' with UUID 9c5a2c03-8243-4a1b-a248-0f0d22d948c2

```
$ stratus-delete-volume 9c5a2c03-8243-4a1b-a248-0
   f0d22d948c2
DELETED 9c5a2c03-8243-4a1b-a248-0f0d22d948c2
```

Check that the disk is no longer there

```
$ stratus-describe-volumes
:: DISK a4324f26-39e0-4965-8c8f-3287cd0936e5
        created: 2011/07/20 16:37:10
        visibility: public
        tag: mypublic-disk
        owner: testor2
        users: 0
        size: 5
:: DISK d955fda6-bf9c-4aa8-abc4-5bbcdb83021b
        created: 2011/07/20 16:26:31
        visibility: public
        tag: mypublic-disk
        owner: testor1
```

```
        users: 1
        size: 5
```

Now try to delete ‘mypublic-disk’ of ‘testor2’ user persistent disk

```
$ stratus-delete-volume a4324f26-39e0-4965-8c8f-3287
   cd0936e5
  [ERROR] Service error: Not enough rights to delete disk
```

Chapter 8

Image Management

Overall view of the Marketplace and how it works.

Describe also the image creation methods and process.

Chapter 9

Programmatic Access

How to use the various StratusLab APIs.

9.1 Libcloud API

The Libcloud API provides a generic python interface for controlling cloud resources from multiple providers. StratusLab now provides a Libcloud compute driver allowing users to access StratusLab cloud infrastructures via this API.

9.1.1 Installing the Driver

The driver is intended to be installed with pip. You should be able to simply do the following:

```
pip install stratuslab-libcloud-drivers
```

which will install the StratusLab Libcloud driver, Libcloud itself (0.12.1), and the StratusLab client. If you want to use the `deploy_node()` function, you'll also need to install paramiko, a python SSH library, as well.

```
pip install paramiko
```

If pip is configured to do system-wide installations, then the PYTHONPATH and PATH should already be set correctly. If it is setup for user area installations, you will likely need to set these variables by hand.

You can download the package directly from PyPi. The name of the package is "stratuslab-libcloud-drivers". You will also need to download and install all of the dependencies. **Using pip is very strongly recommended.**

9.1.2 Configuring the StratusLab Client

The Libcloud drivers for StratusLab use the same configuration file as for the command line client.

To copy a reference configuration file into place, use the command `stratus-copy-config` to copy an example configuration file into place. The command will print the location of your configuration file. The example configuration file contains extensive documentation on the parameters. Edit the file and put in your credentials and cloud endpoints.

More detailed documentation can be found in the StratusLab documentation area on the website.

9.1.3 Using the Driver

Once you've downloaded, installed, and configured the necessary dependencies, you are ready to start using the driver.

From the Python interactive shell do the following:

```
from libcloud.compute.providers import set_driver

set_driver('STRATUSLAB',
           'stratuslab.libcloud.compute_driver',
           'StratusLabNodeDriver')
```

This registers the driver with the Libcloud library. This import must be done **before** asking Libcloud to use the driver! Once this is done, then the driver can be used like any other Libcloud driver.

```
# Obtain an instance of the StratusLab driver.
from libcloud.compute.types import Provider
from libcloud.compute.providers import get_driver
StratusLabDriver = get_driver('stratuslab')
driver = StratusLabDriver('default')

# Use the Libcloud methods to find, create and control
   nodes.
nodes = driver.list_nodes()
```

There are a couple examples in the test area of the GitHub repository for this driver lc-sl-examples. You can also find general information on the Apache Libcloud website.

9.1.4 Driver Status

This driver is currently a prototype and of beta quality. This driver is probably *not* suitable for production.

The driver is functionally complete and should work with all of the standard Libcloud workflows. Problems encountered when using the driver should be reported via the StratusLab support mailing list.

In detail, the following functions have working implementations:

- `list_images`: list all valid images in Marketplace
- `list_locations`: list of sections in configuration file
- `list_sizes`: list of standard machine instance types
- `create_node`: start a virtual machine
- `deploy_node`: start a VM and run a script (see notes below)
- `destroy_node`: terminate a virtual machine
- `list_nodes`: list of active virtual machines
- `create_volume`: create persistent disk
- `destroy_volume`: destroy a persistent disk
- `attach_volume`: attach a volume to node
- `detach_volume`: remove a volume from a node

The `list_volumes` function is specific to the StratusLab driver and is not part of the Libcloud standard abstraction.

The `reboot_node` function will not be implemented as the required functionality is not provided by a StratusLab cloud.

Notes for `deploy_node`:

1. The SSH library used by Libcloud seems to only work correctly with DSA SSH keys. You can have both RSA and DSA keys available in parallel.
2. This function uses sftp to transfer the script between the client and the virtual machine. Consequently, SSH implementations that do not support sftp will not work. This include, notably, ttylinux.

9.2 StratusLab API

9.3 CIMI API