CECP Docs Documentation

Release 1.0

Aleksander

Jul 31, 2018

OPENSTACK

1 Podstawy **3**
- 1.1 Nazewnictwo . . . 3
- 1.2 Przegląd GUI - Horizon . . . 3
- 1.3 Repozytorium obrazów - Glance . . . 3
- 1.4 Compute - Nova . . . 5
- 1.5 Sieci - Neutron . . . 6
- 1.6 Block Storage - Cinder . . . 7

2 Tematy zaawansowane **9**
- 2.1 [Use cases](./use-cases/index.md) . . . 9
- 2.2 [Troubleshooting & Support](./trbl-support/troubleshoot.md) . . . 9
- 2.3 [DNS, Proxy etc.](./util_systems.md) . . . 9

3 Tutorial **11**
- 3.1 Naglowki . . . 11
- 3.2 Styl tekstu . . . 11
- 3.3 Listy . . . 12
- 3.4 Tabelki . . . 13
- 3.5 Linki . . . 14
- 3.6 Obrazy . . . 14
- 3.7 Pobieranie . . . 15
- 3.8 Podstawienia . . . 15
- 3.9 Table of Contents . . . 15
- 3.10 Oznaczenia zwracajace uwage . . . 15
- 3.11 Kod w tekscie . . . 16
- 3.12 Zewnetrzne instrukcje . . . 17

Hi, this is a documentation for CECP. This is new version for dev branch!!!

CHAPTER

ONE

PODSTAWY

1.1 Nazewnictwo

- Instancja = VM
- Block storage/Persistent storage = volumen
- Ephemeral storage = "/dev/sda"
- Flavor = rozmiar VM
- "Snapshot" = Image
- Floating IP = NAT
- Sieć zewnętrzna = sieć routowalna w CDC = DMZ

More:

- Ephemeral storage is allocated for an instance and is deleted when the instance is deleted. The Compute service manages ephemeral storage and by default.
- Persistent storage exists outside all instances. Two types of persistent storage are provided:
  - The Block Storage service (cinder) that can use LVM or Ceph RBD as the storage back end.
  - The Image service (glance) that can use the Object Storage service (swift) or Ceph RBD as the storage back end.

1.2 Przegląd GUI - Horizon

Web-based GUI.

https://cecp.cadc.pl

1.2.1 Resources

Docs

1.3 Repozytorium obrazów - Glance

Glance image services include discovering, registering, and retrieving virtual machine (VM) images. Glance has a RESTful API that allows querying of VM image metadata as well as retrieval of the actual image.

1.3.1 How to use

- Horizon: The official web ui for the OpenStack Project.
- OpenStack Client: The official CLI for OpenStack Projects. You should use this as your CLI for most things, it includes not just nova commands but also commands for most of the projects in OpenStack.

```
$ openstack help image create
usage: openstack image create [-h] [-f {json,shell,table,value,yaml}]
                              [-c COLUMN] [--max-width <integer>]
                              [--fit-width] [--print-empty] [--noindent]
                              [--prefix PREFIX] [--id <id>]
                              [--container-format <container-format>]
                              [--disk-format <disk-format>]
...
                        Image disk format. The supported options are: ami,
                        ari, aki, vhd, vmdk, raw, qcow2, vhdx, vdi, iso,
                        ploop. The default format is: raw
  --min-disk <disk-gb>  Minimum disk size needed to boot image, in gigabytes
  --min-ram <ram-mb>    Minimum RAM size needed to boot image, in megabytes
  --file <file>         Upload image from local file
  --volume <volume>     Create image from a volume
```

1.3.2 Custom images requirements

- Disk partitions and resize root partition on boot (cloud-init)
- No hard-coded MAC address information
- Ensure ssh server runs
- Disable firewall
- Access instance by using ssh public key (cloud-init)
- Process user data and other metadata (cloud-init)
- Ensure image writes boot log to console

Detailed requiremetns

1.3.3 Ready images

- CentOS 6 - http://cloud.centos.org/centos/6/images/
- CentOS 7 - http://cloud.centos.org/centos/7/images/
- Ubuntu - http://cloud-images.ubuntu.com/trusty/current/
- Red Hat 6&7 – requires access to RHN.

1.3.4 Resources

- GlanceClient: Glance CLI

1.4 Compute - Nova

Nova is the OpenStack project that provides a way to provision compute instances (aka virtual servers).

It requires the following additional OpenStack services for basic function:

- Keystone: This provides identity and authentication for all OpenStack services.
- Repozytorium obrazów - Glance: This provides the compute image repository. All compute instances launch from glance images.
- Neutron: This is responsible for provisioning the virtual or physical networks that compute instances connect to on boot.

1.4.1 How to use

- Horizon: The official web ui for the OpenStack Project.
- OpenStack Client: The official CLI for OpenStack Projects. You should use this as your CLI for most things, it includes not just nova commands but also commands for most of the projects in OpenStack.

```
$ openstack help server create
usage: openstack server create [-h] [-f {json,shell,table,value,yaml}]
                               [-c COLUMN] [--max-width <integer>]
                               [--fit-width] [--print-empty] [--noindent]
                               [--prefix PREFIX]
                               (--image <image> | --volume <volume>) --flavor
                               <flavor> [--security-group <security-group>]
...

Create a new server
positional arguments:
  <server-name>         New server name
optional arguments:
  -h, --help            show this help message and exit
  --image <image>       Create server boot disk from this image (name or ID)
  --volume <volume>     Create server using this volume as the boot disk (name
                        or ID).
                        This option automatically creates a block device
                        mapping with a boot index of 0. On many hypervisors
                        (libvirt/kvm for example) this will be device vda. Do
                        not create a duplicate mapping using --block-device-
                        mapping for this volume.
  --flavor <flavor>     Create server with this flavor (name or ID)
  --security-group <security-group>
```

- Nova Client: For some very advanced features (or administrative commands) of nova you may need to use nova client. It is still supported, but the openstack cli is recommended.

1.4.2 API resources

- Compute API Guide: The concept guide for the API. This helps lay out the concepts behind the API to make consuming the API reference easier.
- Compute API Reference: The complete reference for the compute API, including all methods and request / response parameters and their meaning.

1.4.3 More resources

- Reference Material
- OpenStackClient Docs
- NovaClient Docs

1.5 Sieci - Neutron

Neutron is an OpenStack project to provide “network connectivity as a service” between interface devices (e.g., vNICs) managed by other OpenStack services (e.g., nova).

1.5.1 How to use

- Horizon: The official web ui for the OpenStack Project.
- OpenStack Client: The official CLI for OpenStack Projects. You should use this as your CLI for most things, it includes not just nova commands but also commands for most of the projects in OpenStack.

```
$ openstack help network create
usage: openstack network create [-h] [-f {json,shell,table,value,yaml}]
                                [-c COLUMN] [--max-width <integer>]
                                [--fit-width] [--print-empty] [--noindent]
                                [--prefix PREFIX] [--share | --no-share]
                                [--enable | --disable] [--project <project>]
                                [--description <description>]
                                [--project-domain <project-domain>]
                                [--availability-zone-hint <availability-zone>]
                                [--enable-port-security | --disable-port-security]
...

Create new network

positional arguments:
  <name>                New network name

optional arguments:
  -h, --help            show this help message and exit
  --share               Share the network between projects
  --no-share            Do not share the network between projects
  --enable              Enable network (default)
  --disable             Disable network
  --project <project>   Owner's project (name or ID)
  --description <description>
```

1.5.2 Resources

CLI Reference Neutron API Reference

1.6 Block Storage - Cinder

1.6.1 Cinder - block storage service

Cinder is an OpenStack project to provide "block storage as a service".

1.6.2 How to use

- Horizon: The official web ui for the OpenStack Project.
- OpenStack Client: The official CLI for OpenStack Projects. You should use this as your CLI for most things, it includes not just nova commands but also commands for most of the projects in OpenStack.

```
$ openstack help volume create
usage: openstack volume create [-h] [-f {json,shell,table,value,yaml}]
                               [-c COLUMN] [--max-width <integer>]
                               [--fit-width] [--print-empty] [--noindent]
                               [--prefix PREFIX] [--size <size>]
                               [--type <volume-type>]
                               [--image <image> | --snapshot <snapshot> | --source
↪<volume> | --source-replicated <replicated-volume>]
                               [--description <description>] [--user <user>]
                               [--project <project>]
                               [--availability-zone <availability-zone>]
                               [--consistency-group consistency-group>]
                               [--property <key=value>] [--hint <key=value>]
                               [--multi-attach] [--bootable | --non-bootable]
                               [--read-only | --read-write]
                               <name>


Create new volume

positional arguments:
  <name>                Volume name

optional arguments:
  -h, --help            show this help message and exit
  --size <size>         Volume size in GB (Required unless --snapshot or
                        --source or --source-replicated is specified)
  --type <volume-type>  Set the type of volume
  --image <image>       Use <image> as source of volume (name or ID)
  --snapshot <snapshot>
                        Use <snapshot> as source of volume (name or ID)
  --source <volume>     Volume to clone (name or ID)
```

1.6.3 Resources

API Reference
CLI Docs

```
>>> print "This is a doctest block."
This is a doctest block.
```

CHAPTER

TWO

TEMATY ZAAWANSOWANE

1. [CLI](./advanced/cli.md)
2. [Tworzenie obrazów](./advanced/images.md)
3. [Firewall & LB](./advanced/firewall.md)
4. [Scheduling](./advanced/scheduling.md)
5. [Moduł orkiestracji](./advanced/orchestration.md)
6. [API](./advanced/api.md)

2.1 [Use cases](./use-cases/index.md)

2.2 [Troubleshooting & Support](./trbl-support/troubleshoot.md)

2.3 [DNS, Proxy etc.](./util_systems.md)

CHAPTER

THREE

TUTORIAL

Prosty tutorial zawierajacy podstawowe formatowania w restructuredText

New in version 1.0.3.

zmiana na masterze

3.1 Naglowki

Kolejne sekcje deklarujemy (zagniezdzamy) w nastepujacy sposob:

```
Tutorial (glowny na strone)
***************************
Powinien byc tylko jeden taki naglowek na dana strone,
-- dodatkowo naglowek ten bedzie uzyty jako rozdzial przy
   konwersji do .pdf

Naglowki (sekcje strony)
========================

H3 - Podsekcje
--------------

H4 - Podpodsekcje
+++++++++++++++++
```

Powoduje to zagniezdzanie kolejnych paragrafow w drzewe (w tym przypadku tutoriala) jak po lewej, a naglowki paragrafow dodaja sie jak poniżej:

3.1.1 H3 - Podsekcje

H4 - Podpodsekcje

3.2 Styl tekstu

```
**bold**
*italics*
```

bold

italics

```
| Tekst podzielony
| na linie
| zaczyna sie od '|'
| i spacji
| tak jak ten tekst
ktory edytuje sie zgodnie
ze znakami nowej linii
```

Tekst podzielony
na linie
zaczyna sie od ‘|’
i spacji
tak jak ten tekst

ktory edytuje sie zgodnie ze znakami nowej linii

3.3 Listy

```
* Cos.
* Cos jeszcze.

albo

1. Punkt 1
2. Punkt 2
3. Punkt 3

albo

- Cos.
- Jeszcze cos.
- I znowu cos.
```

- Cos.
- Cos jeszcze.

albo

1. Punkt 1
2. Punkt 2
3. Punkt 3

albo

- Cos.
- Jeszcze cos.
- I znowu cos.

3.4 Tabelki

```
Skomplikowana tabela:

+--------------+--------------+-------------+
| Naglowek 1   | Naglowek 2   | Naglowek 3  |
+==============+==============+=============+
|   wiersz 1   |  kolumna 2   |  kolumna 3  |
+--------------+--------------+-------------+
|   wiersz 2   |     Polaczone komorki.     |
+--------------+--------------+-------------+
|   wiersz 3   |  Polaczone   | - komorki   |
+--------------+  komorki.    | - zawieraja |
|   wiersz 4   |              | - punkty    |
+--------------+--------------+-------------+

Prosta tabelka:

=====  =====  =======
  Wejscia     Wyjscia
------------  -------
  A      B    A or B
=====  =====  =======
False  False  False
True   False  True
False  True   True
True   True   True
=====  =====  ======
```

Wynik:

Skomplikowana tabela:

Naglowek 1	Naglowek 2	Naglowek 3
wiersz 1	kolumna 2	kolumna 3
wiersz 2	Polaczone komorki.	
wiersz 3	Polaczone komorki.	• komorki • zawieraja • punkty
wiersz 4		

Prosta tabelka:

Wejscia		Wyjscia
A	B	A or B
False	False	False
True	False	True
False	True	True
True	True	True

3.5 Linki

```
Zwykly link zewnetrzny:
`Glowne repozytorium CECP <http://git.cadc.pl/>`_
```

Zwykly link zewnetrzny: Glowne repozytorium CECP

```
Odnosnik do jakiejs sekcji na tej stronie:
`Tabelki`_
```

Odnosnik do jakiejs sekcji na tej stronie: *Tabelki*

3.6 Obrazy

```
.. _to_lena: <---- to jest 'link' do zdjecia, dziala jak zwykly odnosnik (patrz Linki)
.. figure::  images/lena.png
   :align:   center
   :scale:   50 %

   To jest Lena w rozmiarze 50% oryginalu.
```

Fig. 3.1: To jest Lena w rozmiarze 50% oryginalu.

```
Obrazek, ktory szukasz to_lena_.
```

Obrazek, ktory szukasz *to_lena*.

3.7 Pobieranie

```
:download:`Zdjecie Leny<images/lena.png>`
```

`Zdjecie Leny`

```
:download:`Jakis .pdf<basics/iaas-training-basic.pdf>`
```

`Jakis .pdf`

3.8 Podstawienia

```
.. |biohazard| image:: images/biohazard.png
Symbol |biohazard| musi byc umieszczany na opakowaniach z odpadami radioaktywnymi.
```

Symbol ☣ musi byc umieszczany na opakowaniach z odpadami radioaktywnymi.

Note: Podstawienia moga okazac sie bardzo przydatne gdy zdefiniujemy je w osobnym pliku .rst i dolaczymy go za pomoca '.. include' na gorze kazdego pliku. Wtedy zawsze mozemy z nich korzystac

```
.. include myfile.rst
taka dyrektywa powoduje 'wpisanie' zawartosci myfile.rst w tym pliku
```

3.9 Table of Contents

Nowe sekcje dokumentacji tworzymy jako nowy plik .rst i podpinamy go do source/index.rst jak ponizej:

```
.. toctree::
  :maxdepth: 2
  :hidden:
  :caption: Basic stuff

  basics/README   // tu dopisujemy nazwe pliku
  tutorial        // bez rozszerzenia
```

Istniejace sekcje po prostu edytujemy w danym pliku .rst

3.10 Oznaczenia zwracajace uwage

```
.. note:: Wiadomosc

.. warning:: Ostrzezenie

.. seealso:: Zobacz

.. tip:: Porada

.. versionadded:: 1.0.5
.. versionchanged:: 2.4.7
```

Note: Wiadomosc

Warning: Ostrzezenie

See also:

Zobacz

Tip: Porada

New in version 1.0.5.

Changed in version 2.4.7.

3.11 Kod w tekscie

```
Surowy kod mozna wklejac po podwojnym dwukropku::

   def my_fn(foo, bar=True):
       """A really useful function.

       Returns None
       """
```

Surowy kod mozna wklejac po podwojnym dwukropku:

```
def my_fn(foo, bar=True):
    """A really useful function.

    Returns None
    """
```

Kod powinien byc wklejony z odstepem pustej linii i wciety za pomoca spacji.
Dzieki temu symbol '::' umozliwia rowniez dowolne tworzenie bloku np:

```
::
(pusta linia)
(spacja)kod kod kod
(spacja)kod kod kod
 ...
(brak spacji) koniec bloku
```

Kod wklejony 'w linie' tekstu wklejamy za pomoca ‘‘ (podwojny backtick):

```
To jest kod wklejony w linie: ``def my_fn(foo, bar=True)``
```

To jest kod wklejony w linie: `def my_fn(foo, bar=True)`

3.12 Zewnetrzne instrukcje

Jesli cos nie dziala, najpewniej brakuje odstepow pustych linii lub spacji. Bardziej szczegolowy opis tych i innych mozliwosci restructuredText mozna znalezc np pod adresami:

Cheatsheet
Pelna dokumentacja