CLOUD CLI ACCESS

INSTALL

Pip is preferred. The general CS infrastructure is a good starting point. However, our experience has been that the software installs cleanly and is free from dependency problems.

Try:

```
python3 -m pip install --user python-openstackclient
```

CONFIGURATION

Use a clouds.yaml file to direct your client. Below is a canonical example, but you will have to modify the variables according to your account. `$HOME/.config/openstack/clouds.yaml`

```
clouds:
  openstack:
    auth:
      auth_url: https://overcloud.cs.uchicago.edu:5000
      username: "CNetID"
      password: "sekret" # don't store your credential on our servers
      project_name: "CNetID"
      user_domain_name: "CS_LDAP"
    region_name: "RegionOne"
    interface: "public"
    identity_api_version: 3
```

NOTE The problem here is that you are taking risks by storing your University credentials in a file. Instead, you can use the API or web interface to create an Application Credential. For example,

```
openstack application credential create --secret sekret --role member --
expiration 2020-10-10:10:10:10 --restricted myapp
```

See the Python Examples Document for detailed configuration recommendations.

Download Configuration File

You can download a customized version of this data after you authenticate to the Web Interface . Click `API Access` from the menu and then the button `Download Openstackrc File`). Make sure you read this file carefully in case you want to customize it. Your password is not included in the file by default.

For convenience, export the cloud name to your environment. Otherwise, all commands must include the flag `–os-cloud=openstack`:

```
export OS_CLOUD=openstack
```

USAGE

First, Take note of a loose UX pattern that the client has:

```
openstack $resource $action $more_options_or_flags
```

take note and always use help for guidance

```
--help
```

For example

```
openstack server create --help
```

Once you have the software installed and the configuration file created, as above, you can start to use the client. What follows is an annotated example that will create an infrastructure of 10 Ubuntu hosts accessing the internet via a NAT device (SNAT), and with one of the hosts reachable at a separate pubic IP address (DNAT).

Annotated Example

Read what has been written above before you read this.

We use this command a lot

```
openstack server list
```

Images

Images are prebuilt operating systems that are used to launch instances. It is equivalent to a live CD. They are usually a few GB in size. A copy of the disk image is written into the instance's boot volume just before it starts running.

There are images that Techstaff provides, some of which are restricted-use. We can build images for you or you can build and upload your own. Our images are generic, bare bones, cloud enabled, popular operating systems, that are a firm foundation for you to customize from. They are often in RAW format, not qcow2, for performance reasons.

Beware of images that are used internally to provide cloud services. You should not usually launch these directly. You do have access to them for the use of a service, and are welcome to customize for advanced usage.

```
openstack image list
```

SSH Keys

Openstack can hold a public key in its db, and insert it into instances when told. This is optional (your author does not use this capability)

```
openstack keypair create --public-key ~/.ssh/id_ed25519.pub mykey
```

Flavors

A flavor is a pre-chosen size for resources that make up an instance. It is a mandatory parameter when creating instances. Look at the available flavors, which your admins have created.

Servers can grow after creation. For example, the `disk-size` attribute merely expresses the **minimum** size of the boot volume, and most cloud-enabled operating systems expand the root volume on first-boot. In spite of this, relying on dynamically resizing instances increases risk, and you should choose a size that is close to what you expect to use.

```
openstack flavor list
```

Networks

Look at the Networks that are available. The meaning of an Openstack "Network" captures L2 semantics, and houses L3 subnets. IPv6 is in preview mode at this time, and is not fully supported on the UC campus.

You are free to use the Network called cloud, if you don't need your hosts to be L2 isolated from other users, and you would like to proceed directly to creating servers.

Using the cloud network cuts down your complexity significantly, and can be changed later, or mixed with other modes at your leisure. Please talk with us if you want to attach a Router to the cloud network.

```
openstack network list
```

See also Advanced Networking hints.

Creating an Instance

You now have all of the prerequisites for launching a virtual computer. These are the prerequisites:

- Properly prepared Network – or use the one called `cloud` if you don't mind sharing a broadcast domain, nor wish to control the source address of your NAT clients
- Flavor Name

- Image Name

NOTE: you won't be able to SSH into the instance, because the NAT is SNAT. Down below you can read how to add a dedicated public ("floating") IP address to any server.

Like other openstack activities, creating a server has many complex options and scenarios. This is a simple and ordinary depiction, creating one server

```
openstack server create \
  --image 20.04 \
  --boot-from-volume=32 \
  --flavor m1.medium \
  --config-drive=true \
  --user-data=/home/chudler/openstack/cluster_test/cloud-init.yml \
  --network cloud \
  myserver
```

The command executed asynchronously, check the status, or supply the –`wait` option next time:

```
openstack server list --name myserver
```

```
openstack server show myserver
```

Here's an example for creating 10 of them, as promised (only the change at the end of the command)

```
openstack server create \
  --image 20.04 \
  --boot-from-volume=32 \
  --flavor m1.medium \
  --config-drive=true \
  --user-data=/home/chudler/openstack/cluster_test/cloud-init.yml \
  --network cloud \
  --min 10 \
  --max 10 \
  myserver
```

Mandatory Firewall Rules

If you are using the default security groups, all ingress network communication is dropped.

Here's a nasty thing I use to determine what the security group is for a server (it can be determined also by looking at security groups directly) [ITS BRITTLE, BEWARE]

```
SEC_GROUP=$(openstack port list \
  --server `openstack server show --format value --column id myserver` \
  --long \
  --column "Security Groups" \
  --format json \
  | jq '.[]."Security Groups"[]' \
```

```
  | sed 's/"//g')
```

If I learned the security group successfully, I can let in SSH.

```
openstack security group rule create \
  --ingress \
  --dst-port 22 \
  --protocol tcp $SEC_GROUP
```

In actual fact, all of the servers you create will be in the same security group, so you will not need to "discover" it more than once.

Internet Addresses

If the server's status shows `Active`, you can assign it an additional IP address. When doing networking work, you might wish to connect to web interface to access the console of the virtual machine.

As in Advanced Networking get a campus IP address from our pool.

Where do you want to create your floating IP?

```
openstack network list
```

Use the network from the previous command:

```
openstack floating ip create campus37
```

You now have an IP you can use:

```
openstack server add floating ip myserver <floating_ip_address>
```

At last, you can ssh into 128.135.37.XX. It is important for you to realize that your local server IP does not change (no new interface is given to the instance). Instead, the router on the subnet simply performs DNAT on behalf of the clients.

Here's another possibility:

```
$ openstack server add network myserver campus37
```

Now your server does have a **new** network interface attached to it, and will be served a DHCP address on it. You will almost certainly have to inform the OS about this manually; the cloud may not help you do that.

This section added a floating ip address directly to the server. You must realize that a router was needed on the subnet for that to happen. On default subnets, your cloud admin has pre-created suitable routers. The command will fail if you are creating your own subnets and networks without taking similar steps.

Mixing and matching these techniques will create hilarious disasters.

A WORD ABOUT CLOUD INIT

Your author uses cloud init extensively and does not contemplate alternative. It is optional. A minimal cloud-init for a modern Ubuntu cloud OS might look like this

```
#cloud-config
network:
  version: 2
  ethernets:
    net0:
      match:
        name: en*
      dhcp4: true
preserve_hostname: false
users:
  - name: ubuntu
    ssh-authorized-keys:
      - CONTENTS OF YOUR ssh key .pub file
timezone: America/Chicago
datasource:
 OpenStack:
  metadata_urls: ["http://169.254.169.254"]
  max_wait: -1
  timeout: 10
  retries: 5
  apply_network_config: true
manage_etc_hosts: false
manual_cache_clean: false
```