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/clouds.yaml`

```
clouds:
  openstack:
    auth:
      auth_url: https://overcloud.cs.uchicago.edu:5000
      username: "CNetID"
      password: "sekret"
      project_id: YOUR PROJECT UUID
      project_name: "CNetID"
      user_domain_name: "CS_LDAP"
    region_name: "RegionOne"
    interface: "public"
    identity_api_version: 3
```

The Python Examples use the same configuration, please read that document for detailed information about configuration and recommendations.

You can also 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.

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 are prebuilt disks that are used to launch instances. 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.

```
openstack image list
```

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)

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

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 it is far better to size them correctly when they are built.

```
openstack flavor list
```

Look at the Networks that are available (an Openstack "Network" captures L2 semantics, and houses L3 subnets). FYI: As of 6DEC19, we have not tested IPv6 for instances.

You are free to use the Network called cloud, if you don't need your hosts to be L2 isolated from other people, 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
```

[EDITOR NOTE: This section should be isolated from the main body]

OPTIONALLY CREATING YOUR OWN NETWORK GEAR

Should you want to create a network of your own that your hosts will be on, not all of these options are necessary

```
openstack network create mynet \
  --provider-network-type geneve \
  --enable-port-security \
  --internal
```

Now create a subnet for your network. This is mandatory for launching instances in the network that you just created.

After this, we now consider you to be a Network Administrator, and that may be more than you bargained for. The meaning of this is that we hold you responsible for connectivity into and out of the subnet, and any conflicts that might arise from your usage of it.

The cloud will **not** restrict your choices without cause. This means you can create impossible and insane situations that have no valid solution. There's no unique danger to the cloud infrastructure, however.

You are now advised that there is no "correct" choice for subnet range and IP. Overlaps are **not** a concern unless you intend to perform route/tunneling among the overlapping regions.

```
openstack subnet create \
  --network mynet \
  --ip-version 4 \
  --subnet-range 192.168.222.0/24 \
  --allocation-pool start=192.168.222.10,end=192.168.222.240 \
  --dns-nameserver 128.135.164.141 mysubnet \
  --gateway 192.168.222.1 \
  --dhcp
```

After creating your own network and subnet(s), a router is also needed. However, a router is **not** needed if your instances only talk to each other. The router will take the gateway of your subnet automatically, and allow clients to access the internet via outbound NAT. Much more is possible, and a router is a prerequisite for the next step, which is inbound NAT (DNAT).

```
openstack router create --enable myrouter
```

```
openstack router add subnet myrouter mysubnet
```

With the router created and attached to your own subnet, develop it further. You need to obtain a free

IP address on the UC Campus. We call this network campus37. The Internet-connected subnet on that network is called public37.

After this command, the router will have one leg in your subnet and one leg in the public campus network (and internet).

Only you will be able to use this address until you destroy it. **DONT ever take more than you need and free this resource as soon as you project ends.**

```
openstack router set myrouter \
  --external-gateway campus37 \
  --enable-snat
```

This is all that will be needed to launch instances. If you had used the network known as cloud, you can skip the steps for this custom network and subnet and router.

Finally Creating an Instance

If all of this worked, 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 care about the L2 boundary nor 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 bionic-server-cloudimg-amd64.raw \
  --boot-from-volume=32 \
  --flavor m1.small \
  --config-drive=true \
  --user-data=/home/chudler/openstack/cluster_test/cloud-init.yml \
  --network mynet \
  myserver
```

The command executed asynchronously, check the status:

```
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 bionic-server-cloudimg-amd64.raw \
  --boot-from-volume=32 \
  --flavor m1.small \
  --config-drive=true \
  --user-data=/home/chudler/openstack/cluster_test/cloud-init.yml \
  --network mynet \
  --min 10 \
  --max 10 \
  myserver
```

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. By default, **no communication is possible**.

```
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. The above was attempting to suggest effective use of the tools, in combination.

If everything so far has succeeded. If the server's status shows "Active", choose one and get remote access to it. You could also use the web interface to access the console, but that's not quite the same. As before, in the Network Gear section, 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 <network>
```

You now have an IP you can use:

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

Note that the command is showing you a deeper and more rare UX pattern than before:

```
openstack server $action $subresource $more_options
```

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. We had created the router earlier for the purpose of SNAT, and had we not done that, this command would have failed. This means that if you are not doing SNAT, you should create a router anyway, but do not give it a campus address of its own.

A WORD ABOUT CLOUD INIT

Your author uses cloud init extensively and does not imagine a life without it. It is optional. The file used in these examples is available on request, but you should develop your own if you use it at all.