SCOPE OF THIS DOCUMENT

This guide covers the common subset of tasks that users would need to perform to have a set of clustered computer instances and associated resources, isolated from others, and accessible to a project for any general purpose, both long-term and short. We are catering heavily to short-term and periodic usage, perhaps lasting no more than a few quarters.

Some things that are not written about here but perhaps should be covered elsewhere

- Theory of operations (everything here is by example)
- Accomplishing tasks from the Web Interface
- Background and History
- Alternative Services within CS and without
- Organizational Policy, such as who can do what
- Deployment Architecture
- Systemic Limitations
- Good Practices (because they are nascent, at best)
- Cloud init, Fog, Terraform, Heat, and other operational tools
- Network and Information Security
- Backup and Restore

The list is not comprehensive.

A WORD ABOUT SECURITY

The security of the virtual computers that you launch is your responsibility. With this software you are able to create wildly insecure configurations that will be hacked within seconds, with no hope of recovering. This can put all of us at risk. The highest risk that you will encounter is to expose your computers to the Internet. Beware of the dangers associated with doing so!

INTRODUCTION AND NOTES

This cluster can spring into being computer resources, easily, and without the involvement of other personnel. The software has some exotic capabilities, but almost everyone will use a common subset:

- L2 and L3 Network
- Router (SNAT and DNAT devices, etc)
- Compute, and all that it entails such as RAM, CPU, Disk, …
- Additional Block Storage (volume mounts)
- Security groups (firewall service)

Some want

- Distributed/Replicated hash tables (K/V store, Object Storage)

The cloud can also manage resources on your behalf that are traditionally handled by a human operator. This is less common, and amounts to cognitive debt that you may eventually have to pay:

- Load balancer

- NFS
- Rancher Kubernetes (among others)
- Lots more

Web Access and Certificates

The cloud is named **Overcloud**. The web interface uses a non-public certificate authority and can be reached at https://overcloud.cs.uchicago.edu. You will have to accept the certificate for all purposes: API, HTTPS, and CLI clients.

NOTE: Our cloud DNS service might not meet your needs. Please test it anyway if you know how (TODO: document)

PROJECTS

Openstack requires Users and their cloud resources to belong to a Project. Users have pre-defined Roles within that Project, such as Member or Admin. The Role, Project, and User together constitute in-context access control. So, when a user is in a certain Project, that User can read, modify, destroy the cloud resources in that Project, or even create new resources. All actions depend on the precise Role of the User in the project. Non-members of a project are not able to do anything with cloud resources of the project to which they are a non-member, including View Access. Users can belong to any number of Projects, and with potentially different Roles. Project Admins can modify the memberships of their own Projects. However, there is not a Role known as Owner.

Your Default Project

All users are initially given a project with the same name as their username and the Admin Role. NB: Not all of Openstack cloud requires a project boundary, and some aspects of the design of these features are imperfect and subject to change in the future.

ACCOUNTS

Any user of Openstack should use their CNetID to login. Because University policy forbids you from storing your password in a persistent manor, if you are automating and in production, create Application Credentials using the Web interface or API. Give the Application Credential permissions that you deem appropriate for the cloud-enabled client or application that you are running.

WEB INTERFACE

Use a modern browser for https://overcloud.cs.uchicago.edu

NOTE: You MUST type the word "csldap" in the Domain field in order to login. Sorry about that.

At the least, the web interface is a convenient way to access the Physical Terminal of a virtual machine, although the command line can do that also. The interface also provides a unique status

and usage dashboard.

The site includes interactive help, and is designed for first-time users. Use it as soon as you can and come back with questions if you have them. This guide may not cover any more.

If you have learned any part of the Command Line Interface, you have also learned the corresponding web interface, because they contain the same concepts and requrements.

CLI INTERFACE

[Editors note: create a new wiki page]

INSTALL

It is installable in many ways.

Use your favorite package manager on your own computer. Pip is preferred because the upstream packages it for themselves and it is in pure python. The general CS infrastructure will become a managed client for you to use in the near future (e.g., linux.cs.uchicago.edu). However, our experience has been that the software installs cleanly and is free from dependency problems.

Try:

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

PRELIMINARY SETUP

Use environment variables to direct your client. Below is a canonical example, but you will have to modify the variables according to your account.

```
# Clear any old environment that may conflict.
for key in $( set | awk '{FS="="}  /^OS_/ {print $1}' ); do unset $key ;
done
export OS_NO_CACHE=True
export COMPUTE_API_VERSION=1.1
export OS_USERNAME=chudler
export no_proxy=overcloud.cs.uchicago.edu
export OS_REGION_NAME=regionOne
export OS_USER_DOMAIN_NAME=Default
export OS_VOLUME_API_VERSION=3
export OS_CLOUDNAME=overcloud
export OS_AUTH_URL=https://overcloud.cs.uchicago.edu:13000
export NOVA_VERSION=1.1
export OS_IMAGE_API_VERSION=2
export OS_PASSWORD=sekret
export OS_PROJECT_DOMAIN_NAME=Default
export OS_IDENTITY_API_VERSION=3
```

```
export OS_PROJECT_NAME=chudler
export OS_AUTH_TYPE=password
export PYTHONWARNINGS="ignore:Certificate has no, ignore:A true SSLContext
object is not available"
export OS_COMPUTE_API_VERSION=2.latest
```

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.

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 shell's environment populated 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.

Look around. The list could be empty, but 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.

```
openstack floating ip create
```

```
openstack server add floating ip myserver 128.135.37.XX
```

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.

NAQ (Never Asked Questions)

Q: Why does it use a self-signed certificate? A: This is a loose end that might be addressed in the future. Let us know if it overburdens you. Note: we are unlikely to acknowledge security concerns associated with this.

Q: What are all of the services enabled? A:

- cinder-backup
- heat
- barbican
- mistral
- ironic (we *do* support baremetal instances!)
- octavia
- sahara
- manila
- ganesha
- ceph (that is, ceph as a service. Your instances can be ceph clients in a software-defined manor if you wish, directly accessing RBD)
- MDS/RADOS/"Kitchen Sink" (you can have a distributed POSIX via ceph, or an object store, etc).
- swift
- designate

- DVR/HA
- Full Blown OVN, as you desire
- nova
- placement (standalone)
- glance
- gnocchi (as-a-service. Hundreds of metrics are collected for everything in your project)

Q: What about containers (docker)? A: We do not provide any support for them directly. We expect that you will want to manage this from inside the VMs that you create. We have no plans to deploy Magnum right now.

Q: My servers are in ERROR state!!! A: If the servers had been running previously, this is bad and may not be recoverable. Talk to us ASAP about anything that you know. We'll troubleshoot. The upside is that the servers might be gone but the volumes and anything else associated with them (ports, etc) can be attached to entirely new servers, as is often done in clouds.

Q: How fast is it? A: We haven't done enough measurements, but believe it to be good enough for instructional use (compare with research use). There is some evidence that it can sustain 100MB/s random writes through to the backend

Q: Is my data safe? A: PROBABLY NOT AT THIS STAGE! The backend is fully redundant and replicated 4x across a cluster of several dozen servers (etc, etc). The problem is that software failures are still happening on the backend and operators want to know why.

Q: How does the SSH key injection work? A: Cloud-init. You won't be able to do this unless you are using a "cloud-ready" operating system

Q: What operating systems are supported? A: We are prepared to run any workload if you are willing to put in requisite work also. We know it to be compatible with all major Unixes and Windows. Building an image is required before an instance can be launched and this is usually done with image building tools (chroot, et al).

Q: What about limitations? A: The following quotas are set on your account and projects, only for the safety of the cloud. We will lift these easily if you need it (values are unspecified here, as yet, sorry):

- gigabytes
- volumes
- secgroups
- secgroup-rules
- server-groups
- ram
- instances
- fixed-ips
- server-group-members
- cores
- per-volume-gigabytes
- backup-gigabytes
- snapshots
- volumes
- backups
- subnetpools
- ports

- subnets
- networks
- floating-ips
- routers

URLS

Most of this is dangerously TMI and the Openstack project can't document *our* cloud. Not everything that they say will be correct.

(compute instances) User's Guide: <https://docs.openstack.org/nova/train/user/> Images Guide: <https://docs.openstack.org/glance/train/user/index.html>

If you want to know how I install and manage the cloud, I adhere to the Install Guide very closely: <https://docs.openstack.org/train/install/> (tripleo)

Administrator Guides aren't for End Users, but they illuminate many things for you <https://docs.openstack.org/train/admin/>

This document is not public but is publicly reachable: <https://howto.cs.uchicago.edu/cloud:recipe:swift>

I am not aware of good quality documentation for much else.

GOOGLE DOES NOT TEND TO PRODUCE GOOD ANSWERS TO OPENSTACK QUERIES. BEWARE! We are running the "Train" realease. FYI. ___