Inception Cloud User's Guide

1 Overview

Creating an inception cloud consists of preparing your workstation, preparing the VM environment by adding a temporary boot-up machine, and then executing the `orchestrator` programme to build the cloud. Once the inception cloud has been created, there is a small amount of housekeeping that is necessary to be able to reach the new cloud. This document describes the software necessary to support and manage an inception cloud, lists the steps needed to start an inception cloud, provides information on accessing the inception cloud and includes the command needed to stop the cloud. The final section contains a small bit of guidance which addresses how to access VMs running on the inception cloud.

Some knowledge of OpenStack, virtual machines and the Nova client command line interface is assumed. This document provides some OpenStack commands to assist with VM setup, but makes no attempt to explain their syntax or to further illustrate any command line parameters that are not directly used in the examples.

1.1 Workstation Requirements

Typically, an inception cloud is started and managed from a regular user account on a remote workstation. There are several software requirements that are necessary for the workstation, and the user must have slightly advanced privledges (sudo). The following list the workstation and user requirements:

- User must have `sudo` privileges that allow /etc/hosts and iptables to be modified
- Python version 2.7
- IPython version 0.13.2 or later
- Sshuttle
- Nova Client version 2.13.0 or later
- Nova oslo.config version 1.1.1 or later
- Git

- Inception software from github

It is possible to use the boot-up VM, described later, as the workstation instead of using it just as a passthrough machine. With the exception of when sshuttle is needed, the setup and startup of an inception cloud is nearly the same regardless of whether the "workstation" is real or virtual. This document will address just the use of a physical workstation, outside of the OpenStack virtual world, to create and manage an inception cloud; the reader is left to make the small extrapolations that allow the boot-up VM to be used in place of a real workstation if that is desired.

2 Preparation

To prepare the workstation and the virtual environment to start an inception cloud, the tasks listed below, and explained in more detail in subsequent sections, must first be performed. All of these should be done on the workstation or from the workstation browser via the OpenStack dashboard or the Nova client CLI.

1. Install software
2. Set environment variables
3. Create keys
4. Start small boot-up VM
5. Add floating IP to the VM
6. Start `sshuttle`

2.0.1 Install Software

Some or all of the required software might already be installed. Verify that the correct versions of each of the software packages are available on the workstation and take steps to upgrade or load the missing packages as is needed. The flavour of Linux installed on your workstation will dictate the exact commands (e.g. apt-get or zypper) that are needed to load andor upgrade Python, sshuttle, and pip. Pip can then be used to install Nova and Oslo. Examples of each of the commands that might be needed to manage the required software packages are presented in Appendix A.

2.0.2 Inception source

The source for inception is available from github. The command below will fetch the inception source and place it in a directory under the current working directory.

```
git clone https://github.com/stackforge/inception.git
```

Following the execution of the `git` command, switch to the inception directory and verify that the directory was populated. Inception may be installed, or used from this directory. If the decision is made to install inception, the following command should be used:

```
python setup.py install
```

2.0.3 Set Environment Variables

Ensure that the environment variables which define the authorisation URL and credentials for OpenStack are set and exported. OpenStack credentials can be obtained using the OpenStack dashboard interface and following these steps:

1. Log into the dashboard
2. Click the `Settings` link (top right)
3. Click the `OpenStack API` link on the left
4. Click the `Download RC File` button and save the file to disk.

Once the file has been saved to disk you can source the file (the assumption is made that the shell being used is bash compatible). Sourcing the file should prompt for a password, and then export the following variables to the environment:

OS_AUTH_URL	The reference to a process which provides user authentication.
OS_PASSWORD	The password for your account (you'll be prompted to enter this when the file is sourced).
OS_TENANT_ID	The ID string for the tenant (currently refered to as the project on the dashboard).
OS_TENANT_NAME	The human form of the tenant ID.
OS_USERNAME	Your user name.

2.0.4 Create Keys

Create (if needed) a public/private key pair and register it with OpenStack. If you do not have a key pair, generate one using nova. (I prefer to name these with the OpenStack cluster/environment name, and then the key name with an extension that indicates private key: agave.scooter.pk.)

```
touch agave.scooter.pk
chmod 600 agave.scooter.pk
nova keypair-add scooter >>agave.scooter.pk
```

The commands above will create the key, write it to disk and register the public key with OpenStack. Executing the `touch` and `chmod` commands prior to generating the key adds a bit of security which prevents the exposure of the key which results if the permissions on the key file are changed after it is generated. Regardless of when the permissions are changed, they will need to be changed in order for the file to recognised and used by ssh.

If you already have a private key (one that several users might share) then a public key can be generated from the private key and registered with OpenStack:

```
ssh-keygen -f agave.shared.pk -y >agave.shared.puk
nova keypair-add --pub-key agave.shared.puk shared
```

If you have both a public and private key file, then the `ssh-keygen` command can be skipped; it is only necessary to supply the existing public key to OpenStack using the nova command line.

2.0.5 Start Tiny Boot-up VM

Create and initialise a tiny VM that will act as the initial gateway to the virtual environment for processes running on the workstation. For the examples used in the remainder of this document, the boot-up VM is given the name `scooter_bv.` The VM should be started with the key that was registered with OpenStack.

```
nova boot --image centos --flavor m1.tiny --key_name shared \
    --security_groups default scooter_bv
}
```

The VM should boot quickly and once it is active you may continue.

2.0.6 Add A Floating IP Address

Add a floating (public) IP address to the new VM so that it can be reached from the "real world." (xxx.xxx.xxx.xxx is one of the public floating IP addresses that are available; use `nova floating-ip-list` to get a list of available addresses.

```
nova  add-floating-ip scooter_bv xxx.xxx.xxx.xxx
```

2.0.7 Start sshuttle

The sshuttle programme creates a tunnel through ssh allowing programmes on the workstation to access VMs created on the same internal network as the boot-up VM without having to assign each VM a public address. The sshuttle command is given the private key portion of the key pair that was used to start the boot-up VM; this is necessary to allow sshuttle to start an ssh session through which the tunnel is created. The user name (ubuntu in the example below) is any user on the boot-up VM that is available and allows access via the key (the assumption is that OpenStack created this user, or it was a part of the saved imagee, and the public key was inserted into the `authorized_keys` file in the user's `.ssh` directory.)

```
sshuttle -e ssh -A -i ~/.vmkeys/agave.shared.pk -v \
   -r ubuntu@xxx.xxx.xxx.xxx 192.168.254.0/24
```

The `-v` option causes sshuttle to be more verbose with messages to the standard error device. Ultimately, redirecting the output of sshuttle to `/dev/null` and running the process asynchronously, is probably wise, but initially seeing the verbose messages scroll by in the window is a nice confirmation that the tunnel is active and data is being transferred. The `xxx` IP address is the public floating IP address assigned earlier. The second address (192.168.254.0 in the example) bb the virtual network that is created by OpenStack. If it is not known, the following command (with suitable path for the public key) might provide the address:

```
ssh -i ~/.vmkeys/agave.shared.pk ubuntu@xxx.xxx.xxx.xxx ifconfig eth0
```

The netmask can be used to determine the number of bits of the address that are treated as the network ID (probably 24) and thus added after the slant on the sshuttle command.

3 Running Orchestrator

The `orchestrator` command, located in the bin directory under the source that was fetched from github, is used to start and stop an inception cloud. The inception cloud environment consists of at least 4 Inception Control VMs (ICVMs) in the environment:

- A gateway machine that will be given a public IP address and function much in the same way as the boot-up VM being used to start the cloud.
- A controller machine used to run the OpenStack software and to provide the OpenStack Dashboard interface.
- A chef machine used to manage Chef installation and configuration scripts.
- One or more worker machines used to host the inception virtual machines (iVMs).

The following sections describe how orchestrator is used.

3.1 Starting The Inception Cloud

The `orchestrator` command is located in the `bin` directory within the source cloned from github. The bin directory can be added to the path, or the command can be executed with a fully qualified path. It will probably be necessary to add the top level inception directory to the `PYTHONPATH` environment variable if inception was not installed. Using the `--help` option will cause all of the

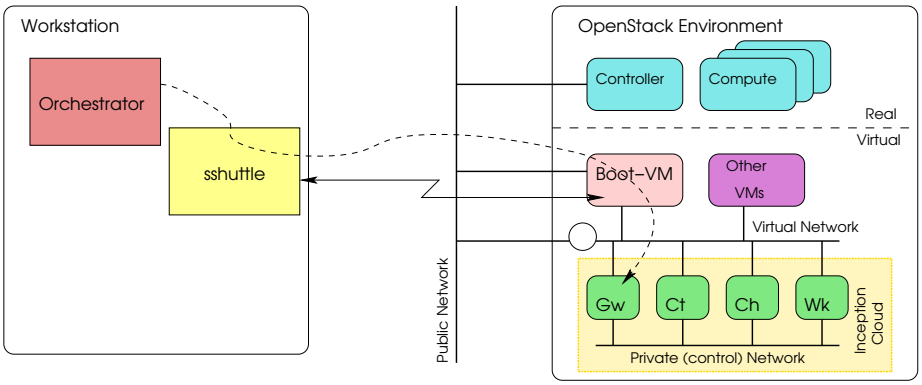

Fig. 1: The environment, after orchestrator has created ICVMs, showing the communication path between orchestrator and the ICVMs.

possible command line options to be written to the tty device. For the most part, at least for the first time user, only a few are needed and are described below.

-p prefix	This command line flag is required and supplies the prefix string that is used when defining the ICVM names.
-n n	Specifies the number of work ICVMs that are created. The iVMs which are created in the inception cloud are hosted on the work VMs thus the number needed is directly related to the number of iVMs that will be created in the inception cloud.
- -image=	Supplied the image name to be used for all ICVMs. If not supplied a base image of Ubuntu 12.04 (64 bit) is created and used for each ICVM.
- -ssh_keyfile=	Provides the name of the private key that is to be injected as the user key for each of the control VMs that are created.
- -user=	The user name created on each node with sudo capabilities. If not given, ubuntu is used.

The following illustrates the command to start an inception cloud with one worker and a prefix of `scooter0`.

```
orchestrator  -n 1 -p scooter0 \
    --ssh_keyfile=$HOME/.vmkeys/agave.shared.pk
```

The creation and initialisation of the ICVMs takes about 20 minutes during which time a fair few messages are written to standard error. When orchestrator

has finished, a set of messages should be written to stdout indicating success and which give the IP addresses and URLs for various things in the newly created environment. The following is a sample of these messages (date, time, and system indentification information has bee excluded for brevity):

```
Your inception cloud 'scooter0' is ready!!!
Gateway IP is 135.207.223.158
Chef server WebUI is http://192.168.254.28:4040
OpenStack dashboard is https://192.168.254.29
```

3.2 Stopping The Inception Cloud

The inception cloud can be stopped manually by halting all of the ICVMs, or orchestrator can be used giving it the `--cleanup` command line flag which causes it to terminate all of the ICVMs.

```
orchestrator  -p scooter0  --cleanup
```

3.3 Finalisation

It will take approximately 20 minutes for orchestrator to start the inception cloud. Once orchestrator reports that the inception cloud is ready, a small amount of housekeeping should be done. These tasks include:

- Determining the network addresses of the gateway ICVM.
- Repointing sshuttle to use the gateway ICVM and to reference the ICVM control network
- Stopping the boot-up VM
- Adding the controller to /etc/hosts
- Creating credentials for your inception cloud

3.3.1 Repointing Sshuttle

Once the ICVMs are running, sshuttle should be "pointed" at the gateway ICVM so that the boot-up VM can be stopped. Sshuttle must also be set to tunnel requests for the private control network that is used by the ICVMs as this is the network on which the nova authorisation and dashboard processes listen on. The following commands illustrate how this can be done:

```
nova list | grep scooter0-gateway
ssh ubuntu@yyy.yyy.yyy.yyy ifconfig eth1
sshuttle -e ssh -A -i ~/.vmkeys/agave.shared.pk -v \
         -r ubuntu@yyy.yyy.yyy.yyy  \
         192.168.254.0/24 zzz.zzz.zzz.0/24
```

Where:

scooter0	Is the prefix that was given to orchestrator when the inception cloud was started.
yyy.yyy.yyy.yyy	Is the public IP address for the gateway.
zzz.zzz.zzz.0	Is the network address of the control network. The netmask also must be checked to determine if /24 is the appropriate number of bits being used to represent a host id; if not it must be changed to match the netmask.
ubuntu	Is the user name that was injected onto each of the ICVMs.

After these commands are executed sshuttle will again be running on the workstation and managing a tunnel between the workstation and both the virtual network and the inception cloud's private control network in the OpenStack environment.

3.3.2 Modifying /etc/hosts

In order to use the inception cloud OpenStack dashboard (URL given in the last set of messages generated by orchestrator), the workstation must be able to resolve the controller host name (e.g. `scooter0-controller` using the earlier example prefix). The easiest way to do this is to have sshuttle forward all DNS requests to the inception cloud environment for resolution. This is done with the addition of a command line flag on the sshuttle command, however shuffling all of the workstation's DNS traffic into the VM environment is probably not a very wise choice. Instead, the host name of the controller, and it's control network IP address (zzz.zzz.zzz.hhh) should be added to the `/etc/hosts` file on the workstation.

3.3.3 Setting Credentials

Credentials must be set in the environment to allow nova to be used on the workstation to control the iVMs in the inception cloud. The following is a list of variables that must be exported and their approximate values (the IP address supplied for the authorisation URL will be different as might the username). The password was given to the user *demo* via the dashboard using the *admin* user ID.

```
export OS_AUTH_URL=http://10.251.0.3:5000/v2.0/
export OS_TENANT_NAME="demo"
export OS_USERNAME=demo
export OS_PASSWORD=demo
```

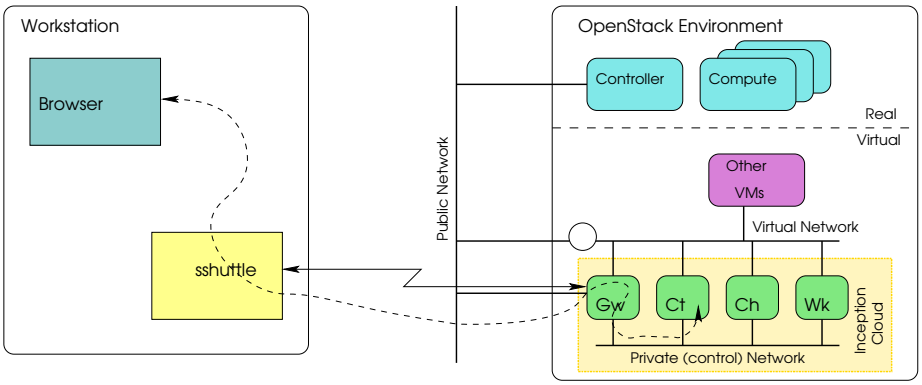

Fig. 2: The virtual environment after cleanup showing path of browser traffic with the dashboard.

The network address given is that of the private control network. The dashboard can be used to setup any users andor projects (tenants) that are needed in the inception cloud. The admin user ID and password are admin/admin by default.

Following housekeeping, the environment should be as shown in figure **??**.

4 Starting and Using iVMs

The dashboard andor nova command line commands can be used to start and manage iVMs within the inception environment. The iVMs are allocated across the worker ICVM(s) that were created by orchestrator. The nova command line interface can be used on either the controller ICVM (Ct in the illustration), or directly from the workstation (provided that the proper environment variables have been defined and that sshuttle is tunneling traffic to the private control network). Figure **??** illustrates the logical relationship of the inception *plane* with the hosting OpenStack (virtual plane) environment.

There are two methods which can be used to ssh into an iVM host. First is to `ssh` into the gateway, then to `ssh` to a worker ICVM, and finally to `ssh` to the desired iVM. The second is to *string* all of the ssh commands together into a single command. Regardless of which method is used, the `-A` and `-i` command line options will need to be given on the initial ssh command in order to forward authorisation with each step, and to use the private key that is associated with the ICVMs. While it is necessary to use the `-i` option only on the first command, the `-A` option must be given on all of the ssh commands.

If a single command line is used, it will also be necessary to use the `-t` option to force ssh to treat each *hop* as a terminal session. The next command example illustrates a single ssh command that creates a log-in session on an iVM.

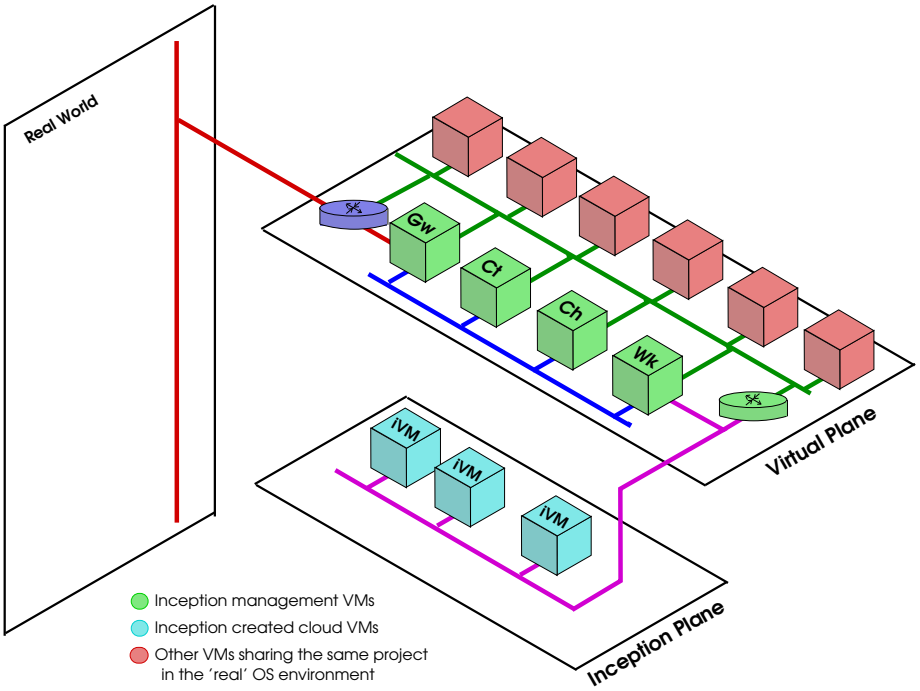

Fig. 3: The relationship between the inception plane and the virtual plane.

```
ssh -t -A -i agave.shared.pk ubuntu@135.207.223.158  \
    ssh -t -A ubuntu@scooter0-worker1  \
    ssh -t ubuntu@10.252.0.2
```

If this command is going to be executed from a script, or just to make life easier, it might be good to add the following options to each of the ssh commands.

```
-o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no
```

These prevent information about the hosts being accessed from being saved in the `known_hosts` file and prevent ssh from complaining if the host information was previously saved and is different from the current information. Be aware that because the host information is being pushed to `/dev/null`, ssh will indicate that it has been added each time the commands are executed (this is less bothersome than having to clean out the known hosts file as the VM host information changes).

A Installation examples

The following commands illustrate how each of the necessary software packages can be installed. Your milage may vary depending on wither your system uses `apt-get, zypper,` or some other package management software.

```
apt-get -y install python
apt-get -y install ipython
apt-get -y install pip
apt-get -y install sshuttle
pip install python-novaclient
pip install oslo.config

cd $HOME/repo               # any writable directory should do
git clone https://github.com/stackforge/inception.git
( cd inception && python setup.py install )
```