ANVIL

Release 2012.5.11-dev

September 10, 2015

Contents

Everything about ANVIL, a set of python scripts and utilities to quickly deploy an OpenStack cloud.

CHAPTER 1

Why the rename?

- Helps avoid confusion with devstack.org
- Allows our team to focus on any different features
- Creates a clearer goal as to where we diverge
- More awesomeness?

1.1 Goals

- To aid developers getting involved with OpenStack!
- To quickly build developer OpenStack environments in a clean environment (as well as start, stop, and uninstall those environments) with as little baggage as possible.
- To describe working configurations of OpenStack.
  - Which code branches work together?
  - What do config files look like for those branches?
  - What packages are needed for installation for a given distribution?
- To make it easier for developers to dive into OpenStack so that they can productively contribute without having to understand every part of the system at once.
- To make it easy to prototype cross-project features.
- To have an installer that works!

1.2 Features

- Supports more than one distribution.
  - Currently RHEL 6.2 (with epel), Ubuntu 11.10, Fedora 16, Ubuntu 12.10 (seems to work)
- Supports dry-run mode (to see what *would* happen)
- Supports varying installation *personas*
- A single `anvil.ini` file that shows common configuration
- Supports install/uninstall/starting/stopping of OpenStack components.

  - In various styles (daemonizing via forking, screen, upstart)
- Written in python so it matches the style of other OpenStack components.
- Extensively documented distribution specifics
  - Packages and pip (with versions known to work!) dependencies
  - Any needed distribution specific actions (ie service names...)
- Follows standard software development practices (for everyones sanity).
  - Functions, classes, objects and more (oh my!)
  - Still *readable* by someone with limited python knowledge.
- The ability to be unit-tested!
- Extensive logging

1.3 Important!

Warning: Be sure to carefully read `smithy` and any other scripts you execute before you run them, as they install software and may alter your networking configuration. We strongly recommend that you run `smithy` in a clean and disposable virtual machine when you are first getting started.

1.4 Basics

1.4.1 Getting Started

Simple setup!

Made to be as simple as possible, but not to simple.

Prerequisites

Linux

One of the tested Linux distributions (RHEL 6.2, Ubuntu 11.10, Fedora 16)

You can get Ubuntu 11.10 (**64-bit** is preferred) from http://releases.ubuntu.com/11.10/

You can get RHEL 6.2 (**64-bit** is preferred) from http://rhn.redhat.com/.

You can get Fedora 16 (**64-bit** is preferred) from https://fedoraproject.org/get-fedora, so don't worry if you do not have a RHN subscription.

Networking

Important!

Since networking can affect how your cloud runs please check out this link:

http://docs.openstack.org/diablo/openstack-compute/admin/content/configuring-networking-on-the-compute-node.html

Check out the root article and the sub-chapters there to understand more of what these settings mean.

This is typically one of the hardest aspects of *OpenStack* to configure and get right!

ANVIL will configure the network in a identical manner to version *1.0*. This means that the default network manager will be the *FlatDHCPManager*. The following settings are relevant in configuring your network.

```
flat_network_bridge = ${FLAT_NETWORK_BRIDGE:-br100}
flat_interface = ${FLAT_INTERFACE:-eth0}
public_interface = ${PUBLIC_INTERFACE:-eth0}
```

The above settings will affect exactly which network interface is used as the *source* interface which will be used as a network *bridge*.

```
fixed_range = ${NOVA_FIXED_RANGE:-10.0.0.0/24}
fixed_network_size = ${NOVA_FIXED_NETWORK_SIZE:-256}
floating_range = ${FLOATING_RANGE:-172.24.4.224/28}
test_floating_pool = ${TEST_FLOATING_POOL:-test}
test_floating_range = ${TEST_FLOATING_RANGE:-192.168.253.0/29}
```

The above settings will determine exactly how nova when running assigns IP addresses. By default a single network is created using *fixed_range* with a network size specified by *fixed_network_size*. Note the size here is *256* which is the number of addresses in the *10.0.0.0/24* subnet (*32 - 24* bits is 8 bits or 256 addresses). The floating pool is similar to fixed addresses (**TODO** describe this more).

Installation

Pre-setup

Since RHEL/Fedora requires a tty to perform `sudo` commands we need to disable this so `sudo` can run without a tty. This seems needed since nova and other components attempt to do `sudo` commands. This isn't possible in RHEL/Fedora unless you disable this (since those instances won't have a tty).

For RHEL and Fedora 16:

```
$ sudo visudo
```

Then comment out line:

```
Default requiretty
```

Also disable selinux:

```
$ sudo vi /etc/sysconfig/selinux
```

Change *SELINUX=enforcing* to *SELINUX=disabled* then it seems you need to reboot.

```
$ sudo reboot
```

For Ubuntu:

You are off the hook.

Users

We need to add a admin user so that horizon can run under apache.

For Ubuntu:

```
$ apt-get install sudo -y
$ sudo adduser horizon
$ sudo adduser horizon admin
```

For RHEL/Fedora 16:

You are off the hook as long as your user has `sudo` access.

Get git!

For Ubuntu:

```
$ sudo apt-get install git -y
```

For RHEL/Fedora 16:

```
$ sudo yum install git -y
```

Download

We'll grab the latest version of ANVIL via git:

```
$ git clone git://github.com/yahoo/Openstack-Anvil.git anvil
```

Now setup the prerequisites needed to run:

```
$ cd anvil && sudo ./warmup.sh
```

Configuration

Apache configuration We need to adjust the configuration of ANVIL to reflect the above user (`iff you created a user`).

Open `conf/anvil.ini`

Change section:

```
[horizon]

# What user will apache be serving from.
#
# Root will typically not work (for apache on most distros)
# sudo adduser <username> then sudo adduser <username> admin will be what you want to set this up (ir
# I typically use user "horizon" for ubuntu and the runtime user (who will have sudo access) for RHEL
#
# NOTE: If blank the currently executing user will be used.
apache_user = ${APACHE_USER:-}
```

To:

```
[horizon]

# What user will apache be serving from.
#
# Root will typically not work (for apache on most distros)
# sudo adduser <username> then sudo adduser <username> admin will be what you want to set this up (ir
# I typically use user "horizon" for ubuntu and the runtime user (who will have sudo access) for RHEL
#
# NOTE: If blank the currently executing user will be used.
apache_user = ${APACHE_USER:-horizon}
```

Network configuration We need to adjust the configuration of ANVIL to reflect our above network configuration.

Please reference:

http://docs.openstack.org/diablo/openstack-compute/admin/content/configuring-networking-on-the-compute-node.html

If you need to adjust those variables the matching config variables in `anvil.ini` are:

```
# Network settings
# Very useful to read over:
# http://docs.openstack.org/cactus/openstack-compute/admin/content/configuring-networking-on-the-comp
fixed_range = ${NOVA_FIXED_RANGE:-10.0.0.0/24}
fixed_network_size = ${NOVA_FIXED_NETWORK_SIZE:-256}
network_manager = ${NET_MAN:-FlatDHCPManager}
public_interface = ${PUBLIC_INTERFACE:-eth0}

# DHCP Warning: If your flat interface device uses DHCP, there will be a hiccup while the network
# is moved from the flat interface to the flat network bridge. This will happen when you launch
# your first instance. Upon launch you will lose all connectivity to the node, and the vm launch will
#
# If you are running on a single node and don't need to access the VMs from devices other than
# that node, you can set the flat interface to the same value as FLAT_NETWORK_BRIDGE. This will stop
flat_interface = ${FLAT_INTERFACE:-eth0}
vlan_interface = ${VLAN_INTERFACE:-$(nova:public_interface)}
flat_network_bridge = ${FLAT_NETWORK_BRIDGE:-br100}

# Test floating pool and range are used for testing.
# They are defined here until the admin APIs can replace nova-manage
floating_range = ${FLOATING_RANGE:-172.24.4.224/28}
test_floating_pool = ${TEST_FLOATING_POOL:-test}
test_floating_range = ${TEST_FLOATING_RANGE:-192.168.253.0/29}
```

Installing

Now install *OpenStacks* components by running the following:

```
sudo ./smithy -a install -d ~/openstack
```

You should see a set of distribution packages and/or pips being installed, python setups occurring and configuration files being written as ANVIL figures out how to install your desired components (if you desire more informational output add a `-v` or a `-vv` to that command).

Starting

Now that you have installed *OpenStack* you can now start your *OpenStack* components by running the following.

```
sudo ./smithy -a start -d ~/openstack
```

If you desire more informational output add a `-v` or a `-vv` to that command.

Check horizon Once that occurs you should be able to go to your hosts ip with a web browser and view horizon which can be logged in with the user `admin` and the password you entered when prompted for `Enter a password to use for horizon and keystone`. If you let the system auto-generate one for you you will need to check the final output of the above install and pick up the password that was generated which should be displayed at key `passwords/horizon_keystone_admin`. You can also later find this authentication information in the generated `core.rc` file.

If you see a login page and can access horizon then:

```
Congratulations.  You did it!
```

Command line tools In your ANVIL directory:

```
source core.rc
```

This should set up the environment variables you need to run OpenStack CLI tools:

```
nova <command> [options] [args]
nova-manage <command> [options] [args]
keystone <command> [options] [args]
glance <command> [options] [args]
....
```

If you desire to use eucalyptus tools (ie euca2ools) which use the EC2 apis run the following to get your EC2 certs:

```
euca.sh $OS_USERNAME $OS_TENANT_NAME
```

It broke? *Otherwise* you may have to look at the output of what was started. To accomplish this you may have to log at the `stderr` and `stdout` that is being generated from the running *OpenStack* process (by default they are forked as daemons). For this information check the output of the start command for a line like `Check * for traces of what happened`. This is usually a good starting point, to check out those files contents and then look up the files that contain the applications PID and `stderr` and `stdout`.

If the install section had warning messages or exceptions were thrown there, that may also be the problem. Sometimes running the uninstall section below will clean this up, your mileage may vary though.

Another tip is to edit run with more verbose logging by running with the following `-v` option or the `-vv` option. This may give you more insights by showing you what was executed/installed/configured (uninstall & start by installing again to get the additional logging output).

Stopping

Once you have started *OpenStack* services you can stop them by running the following:

```
sudo ./smithy -a stop -d ~/openstack
```

You should see a set of stop actions happening and `stderr` and `stdout` and `pid` files being removed (if you desire more informational output add a `-v` or a `-vv` to that command). This ensures the above a daemon that was started is now killed. A good way to check if it killed everything correctly is to run the following.

```
sudo ps -elf | grep python
sudo ps -elf | grep apache
```

There should be no entries like `nova`, `glance`, `apache`, `httpd`. If there are then the stop may have not occurred correctly. If this is the case run again with a `-v` or a `-vv` or check the `stderr`, `stdout`, `pid` files for any useful information on what is happening.

Uninstalling

Once you have stopped (if you have started it) *OpenStack* services you can uninstall them by running the following:

```
sudo ./smithy -a uninstall -d ~/openstack
```

You should see a set of packages, configuration and directories, being removed (if you desire more informational output add a `-v` or a `-vv` to that command). On completion the directory specified at ~/openstack be empty.

Issues

Please report issues/bugs to https://launchpad.net/anvil. Much appreciated!

1.4.2 Usage Examples

Commands

To get the help display try:

```
$ ./smithy --help
```

To examine what installing the basics (nova, horizon, glance, keystone...) will do (with installation in `~/openstack`) try:

```
$ sudo ./smithy -d ~/openstack -a install --dryrun
```

With more information/debugging/auditing output try:

```
$ sudo ./smithy -d ~/openstack  -a install -vv
```

1.4.3 Solved Problems

Solutions

Mysql user denied

This seems common and can be fixed by running one of the steps at http://dev.mysql.com/doc/refman/5.0/en/resetting-permissions.html

Mysql unknown instance

This seems to happen sometimes with the following exception:

```
ProcessExecutionError: None
Command: service mysql restart
Exit code: 1
Stdout: ''
Stderr: 'restart: Unknown instance: \n'
```

To resolve this the following seems to work:

```
MYSQL_PKGS=`sudo dpkg --get-selections | grep mysql | cut -f 1`
echo $MYSQL_PKGS
sudo apt-get remove --purge $MYSQL_PKGS
```

Horizon dead on start

If you get the following (when starting *horizon*) in ubuntu 11.10:

```
.: 51: Can't open /etc/apache2/envvars
```

Run:

```
APACHE_PKGS=`sudo dpkg --get-selections | grep apache | cut -f 1`
echo $APACHE_PKGS
sudo apt-get remove --purge $APACHE_PKGS
```

Then stop and uninstall and install to resolve this.

1.5 Questions and Answers

1.5.1 Can I use ANVIL for production?

Up to u! Beware of the sea and the black waters!

1.5.2 How do I get program usage?

```
$ ./smithy --help
```

1.5.3 How do I run a specific OpenStack milestone?

OpenStack milestones have tags set in the git repo. Set the appropriate setting in the **branch** variables in *conf/anvil.ini*.

Note: Swift is on its own release schedule so pick a tag in the Swift repo that is just before the milestone release.

For example:

```
# Quantum client git repo
quantum_client_repo = git://github.com/openstack/python-quantumclient.git
quantum_client_branch = essex-3

# Melange service
```

```
melange_repo = git://github.com/openstack/melange.git
melange_branch = essex-3

# Python melange client library
melangeclient_repo = git://github.com/openstack/python-melangeclient.git
melangeclient_branch = essex-3
```

1.5.4 OMG the images take forever to download!

Sometimes the images that will be uploaded to glance take a long time to download and extract and upload.

To adjust this edit *conf/anvil.ini* and change the following:

```
[img]

...

# uec style cirros 0.3.0 (x86_64) and ubuntu oneiric (x86_64)
image_urls = http://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-uec.tar.gz, http:/
```

To something like the following:

```
[img]

...

# uec style cirros 0.3.0 (x86_64)
image_urls = http://launchpad.net/cirros/trunk/0.3.0/+download/cirros-0.3.0-x86_64-uec.tar.gz
```

This will remove the larger ubuntu image and just use the smaller cirros image (which should not take to long to upload).

1.6 Known Issues

1.6.1 Ubuntu 11.10 (Oneiric Ocelot)

- Resetting/cleaning up the network on uninstall doesn't seem to be *100%* correct

There is a script in `tools/clear-net-ubuntu.sh` that might help with this.

1.6.2 RHEL 6.2

- *numpy* (for novnc) pulls in *python-nose* which we are installing from *EPEL* and *symlinking* so that we don't have to modify github code directly but the *numpy* dependency can't be installed with the previous symlink. We are currently just using *pip* to fix this.
- Fixing up the network on uninstall doesn't seem to be 100% correct

We might need a script like the `tools/clear-net-ubuntu.sh` to help in this situation.

1.6.3 Others

Any other piece of code with a *TODO* probably should be looked at...

1.7 Bugs & Hugs & Code

1.7.1 Community

ANVIL is an open-source tool released under the apache version 2.0 license. It *depends* on its **community** to keep it alive.

1.7.2 Source code

The source code is on github located at:

https://github.com/yahoo/Openstack-Anvil

Feel free to fork it and contribute to it. You can also get a tarball or zip archive of the code.

Tags

Stable *tags* can also be downloaded:

https://github.com/yahoo/Openstack-Anvil/tags

> **Note:** that for these tags you may have to edit `conf/anvil.ini` to point to tags other than `master`

1.7.3 Bugs/Features

Please use github's issue tracking system or launchpad's issue tracking system to report or follow bugs or to discuss features and get support.

1.7.4 Hacking

Feel free to hack but please try to follow the hacking guidelines

1.7.5 Discussions

Please either use launchpad's email system or find us on `irc.freenode.net` in channel `#openstack-anvil` or in the main openstack dev channel `#openstack-dev`. Feel free to bug us!

1.8 Advanced

1.8.1 Adding your own distribution

Your mission...

So you have decided you want to venture into the bowels of ANVIL and want to get support for your latest and greatest distribution. This wiki will hopefully make that adventure simpler by listing out the key places, files and configs that may have to be adjusted to get that to work. This tape will self-destruct in 5 seconds. 1..2..3..4..5, boom!

Steps

Snapshot

One of the most useful things to do will be to get a virtual machine with your distribution and setup a stable state. Create a snapshot of that stable state just in-case you *bork* your machine later on.

Logging

Now turn ensure you run `DEBUG/VERBOSE` logging using `-vv`. This will be useful to see exactly what actions and commands are being ran instead of the default `INFO` level logging which is just meant for simple informational messages about the underlying actions which are occurring.

Configs

By looking at the config folder `distros` you should exactly which packages and pips and commands are needed for each component by looking at a similar distribution. So you first task is to determine exactly what versions are available for your distribution. If a version doesn't exist the you may need to resort to either using the pypi index or having to package it yourself. If a version is to new, this is usually ok (your mileage may vary) and if its to old then that might also not be ok (your mileage may vary).

Try it

Now that you have provided a new YAML distro file you should be able to run through the simple setup wiki and see if the install will pass. If that does try starting and then seeing if everything has started up correctly.

1.8.2 Adding your own persona

Your mission...

So you have decided you want to venture into the bowels of ANVIL and want to alter what is installed/started/stopped, the order of what is installed/started/stopped, what subsystems are activated (or the component options). This wiki will hopefully make that adventure simpler by listing out the key places, files and configs that may have to be adjusted to get that to work. This tape will self-destruct in 5 seconds. 1..2..3..4..5, boom!

Steps

Snapshot

One of the most useful things to do will be to get a virtual machine with your distribution and setup a stable state. Create a snapshot of that stable state just in-case you *bork* your machine later on.

Logging

Now turn ensure you run `DEBUG/VERBOSE` logging using `-vv`. This will be useful to see exactly what actions and commands are being ran instead of the default `INFO` level logging which is just meant for simple informational messages about the underlying actions which are occurring.

Configs

By looking at the config folder `personas` you should a file called `devstack.sh.yaml`. This file contains the component order of installation (ie the `db` before `keystone`), a nice useful description of the persona and subsystems for the previously specified components and any options these components may have. So you first task is to determine exactly what of these you wish to change (if any). Note that changing the component order may not always work (ie typically starting components are dependent, ie the message queue needs to be started before nova). To add in new components check the `distros` folder to determine exactly what that component is named (typically this is common) and alter the persona file as desired. To alter the `subsystems` or `options` section you will have to jump in the code and check for what these values could be (TODO make that better).

Try it

Now that you have provided a new YAML persona file you should be able to run the `stack` program with that persona through the `-p` option.

1.8.3 Design

How it works

ANVIL is based along the following system design

- Having shared components/actions be shared (using object oriented practices)
- Having specific actions be isolated to its component (and easily readable)
- Being simple enough to read yet following standard python software development practices and patterns

Directory structure is the following

- Parent classes located in *anvil/component.py*, it contains the root install/uninstall/start/stop classes
- Subclasses in *anvil/components/*, it contains the individual install classes for each openstack component
- Running modes implementations in *anvil/runners/* (ie fork, screen, upstart)
- Packaging implementations in *anvil/packaging/*
- Image uploading/registry/management (for glance) in *anvil/image/*
- Shared classes and utils in *anvil/*
- Main entry point/s in *anvil/progs/*
- Other various tools in *tools/*
- Configuration in *conf/* (see below)

Example

Install object model Here is the **install** components (default set only) class hierarchy:

Classes From this example the root classes job are:

- Python install component
- Ensures *pips* are installed (child classes specify which pips)
- Ensures python directories (ie *setup.py*) are setup (child classes specify which directories)
- Package install component
- Installs packages that are required for install (child classes specify which packages)
- Sets up and performs parameter replacement on config files (child classes specify which config files)
- Sets up symlinks to config or other files (child classes specify these symlinks)
- Tracks what was setup so that it can be removed/uninstalled
- Component base (used by **all** install/uninstall/start/stop component classes)
- Holds configuration object, component name, packaging and other shared members...
- Allows for overriding of dependency function and pre-run verification

Functions For a install class the following functions are activated (in the following order by *anvil/actions.py*):

```
  download()

Performs the main git download (or other download type) to the
application target directory.
```

```
  configure()

Configures the components files (symlinks, configuration and logging
files...)
```

```
  pre_install()

Child class specific function that can be used to do anything before
an install (ie set a ubuntu mysql pre-install root password)
```

```
  install()

Installs distribution packages, python packages (*pip*), sets up
python directories (ie *python setup.py develop*) and any other
child class specific actions.
```

```
  post_install()

Child class specific function that can be used to do anything after
an install (ie run *nova-manage db sync*)
```

Other object models

- Start object model (for default set only)
- Stop object model (for default set only)
- Uninstall object model (for default set only)

Configuring

For those of you that are brave enough to change *stack* here are some starting points.

conf/stack.ini

Check out *conf/stack.ini* for various configuration settings applied (branches, git repositories...). Check out the header of that file for how the customized configuration values are parsed and what they may result in.

conf/distros

Check out *conf/distros* for the YAML files that describe pkgs/cmds/pips for various distributions which are required by the different OpenStack components to run correctly. The versions and pip names listed for each distribution should be the correct version that is known to work with a given OpenStack release.

conf/templates/

Check out *conf/templates/* for various component specific settings and files. All of these files are *templates* with sections or text that needs to be filled in by the `stack` script to become a *complete* file.

These files may have strings of the format `%NAME%` where `NAME` will most often be adjusted to a real value by the *stack* script.

An example where this is useful is say for the following line:

```
admin_token = %SERVICE_TOKEN%
```

Since the script will either prompt for this value (or generate it for you) we can not have this statically set in a configuration file.