downpour Documentation

Release

OpenStack Foundation

May 01, 2017

Contents

downpour exports tenant data from an OpenStack cloud to create a set of Ansible playbooks for importing the data into another cloud.

Note: The project is in a very very early prototyping stage.

CHAPTER 1

Contents

Background

Downpour is being created to solve the problem of moving workloads between clouds. It is only one of several possible approaches to the problem, and fits into a very specific niche at the hard end of the range of use cases.

	Easy	Moderate	Hard
Ownership	Operator	Admin	Tenant
Backend	Shared storage	Fast interconnect	Shared nothing
Applications	One per tenant	Multi-app with naming conventions	Rats nest

Downpour does not assume the user has an special access to the cloud, either as an operator with access to backend systems or via admin APIs.

Downpour does not assume that the source and destination clouds are connected in any way. Not only is it possible to move data between clouds that do not share backend services, it is possible to move data between clouds that cannot be accessed from the same system at the same time.

Downpour does not make any assumptions about the mapping between applications and tenants. It is possible to extract only part of the resources owned by a tenant. The grouping is completely up to the user, and can represent an application or a single node in a multi-VM configuration.

Downpour does not assume the source and destination clouds are build using the same architecture or configured in the same way. As long as the two clouds pass the standard OpenStack interoperability tests, it should be possible to use Downpour to move your workload.

These requirements do come with trade-offs, the impact of which will depend on how "cloud native" an application really is. For example, the benefits of copy-on-write images may be lost during the migration if the entire image from each VM needs to be uploaded into the new cloud. The UUIDs associated with resources will also change, since there is no way to guarantee the assignment of a specific UUID for resources created in a separate cloud.

Installation

Prerequisites

Downpour is written to take advantage of features of Python 3.5, so you will need a Python 3.5+ interpreter installed.

Installing with pip

At the command line:

```
$ pip install os-downpour
```

Note: The dist name for downpour is `os-downpour`.

Cloud Access Credentials

downpour uses os-client-config for settings related to accessing the cloud. Fill in your `clouds.yaml` or use the environment variables or command line arguments provided.

Usage

Downpour uses a four step process. Between each step it is possible to stop and modify the data that has been prepared to pass to the next step.

1. Identify Resources to Export

The first phase of using Downpour is to identify exactly what resources will be exported from the cloud to build the *resource file*. This step can be performed by hand by creating the required input file in a text editor, or the file can be build using the `query` command.

The resource file is a YAML file with sections for the principle resource types, `keypairs`, `images`, `volumes`, and `servers`. Resources are identified by name, and may also include extra parameters to control how the export and re-import operations are performed. For example, this resource file causes the `downpour-demo-tiny` server to be exported but when it is recreated a different ssh key is used to provide access to log in.

```
# Resource file for downpour using the instance created in tiny.yml.
servers:
  - name: downpour-demo-tiny
    # Create the server using a separate key than
    # it was created with in tiny.yml.
    key_name: downpour-demo2
keypairs:
  - name: downpour-demo
  - name: downpour-demo2
images:
  - name: cirros-0.3.5-x86_64-disk
```

The `downpour query` command also can be used to find resources visible in the cloud, and add them to the resource file. It supports wildcard patterns in names and other criteria for filtering resources. For example, this command finds all servers with "`tiny`" in their name.

```
$ downpour query --server-name '*tiny*' export.yml
```

See also:

Resource File Format includes more details about resource files.

2. Exporting Resources

The second phase of operation is to actually export the resources from the cloud using `downpour export`, passing the resource file as input. Downpour starts by processing the resources listed in the file explicitly, and identifies any extra dependencies needed to recreate the configuration of those resources. For example, the networks, subnets, and security groups used by a server are exported automatically, as are the volumes attached to the server.

```
$ downpour export export.yml ./export/
```

The output for the export process is an Ansible playbook to recreate the resources, with all relationships intact. For images, volumes, and servers with the `save-state` flag set to true, the content of the resource will be downloaded and saved to the output directory where it can be used to recreate the resource.

3. Importing Resources

The import phase uses `ansible-playbook` to run the playbook created by the exporter.

Note: Although Downpour currently requires Python 3.5 or greater, Ansible is a Python 2.x application. If you are using `pip` and `virtualenv` to install the tools, you will need to install them in separate virtual environments.

Ansible uses uses os-client-config for settings related to accessing the cloud. The simplest way to configure the cloud is via a `clouds.yaml` file, but any mechanism supported by Ansible will work. The credentials used for the import phase do not need to be the same as the export phase. In fact, they're likely to be completely different because they will refer to a separate cloud's API endpoints.

Downpour supports some customizations during export, such as changing the ssh key to be used for accessing a server. Other changes can be made by editing the playbook before running it.

The playbook produced by Downpour creates each resource, then adds a line to a file `uuids.csv` to map the UUID in the source cloud to the UUID in the target cloud. This file may be useful for updating scripts or other configuration that rely on the UUID instead of a unique name for the resource.

```
"Resource Type","Resource Name","Old","New"
"security group","downpour-demo","6deea469-54bd-4846-b12a-79fa6b482280","a4b80ffc-
↪bc51-485c-915a-9ba9a7b4dcf0"
"volume","downpour-demo-tiny","256868c6-441f-4cd3-96fd-bda92c33822c","62e5616c-9a8c-
↪44e2-bd14-4685b905ea94"
"security group","downpour-demo","3c7dcb77-d9ac-4af1-ba95-3f5d89a85227","a4b80ffc-
↪bc51-485c-915a-9ba9a7b4dcf0"
"volume","downpour-demo-tiny","a6192546-c36e-4bee-ad00-8229e0b0efc5","62e5616c-9a8c-
↪44e2-bd14-4685b905ea94"
"network","private","56a86bdb-13b2-4c9f-b8f5-a942d52602b5","f3027502-e4a2-4610-81fb-
↪c6df99ead5c3"
"subnet","ipv6-private-subnet","8d736fe4-6b8f-4bf5-a38e-b511dce21f7f","01025e33-703b-
↪4aa4-b6ec-80036bb3679b"
```

```
"subnet","private-subnet","e6baf9f4-09b5-4292-8236-3cca609ec2a3","2f9a1686-8125-4316-
↪acd3-dbee51c44c1d"
"keypair","downpour-demo","downpour-demo","downpour-demo"
"image","cirros-0.3.5-x86_64-disk","570ec7bd-011b-4fbe-9968-626225654a7f","570ec7bd-
↪011b-4fbe-9968-626225654a7f"
```

4. Decomissioning Resources

Downpour is not a live-migration tool, and it does not delete any resources from the source cloud. This allows you to perform application-specific migration (such as a final database sync) before updating any load balancers or DNS records and deleting old information.

Resource File Format

A Downpour resource file is a YAML file containing explicitly identified resources to be exported, along with instructions for how to handle the export.

keypairs

The keypairs section lists the names of the keypairs to be exported. Keys associated with servers are exported automatically, but if it is important to move keys not in use by any of the servers those keys can be listed separately.

Each item in the keypairs list should be a mapping with a value for `name`.

```
keypairs:
  - name: downpour-demo
```

images

The images section lists the names of the images to be exported.

Each item in the images list should be a mapping with a value for `name`.

```
images:
  - name: cirros-0.3.5-x86_64-disk
```

volumes

The volumes section lists the names and settings for the unattached volumes to be exported. This section should **not** include volumes attached to servers, because those are exported as part of exporting the server definition.

Each item in the images list should be a mapping with a value for `name` and an optional boolean value for `save_state`, indicating whether the contents of the volume should be exported. If `save_state` is false, a new volume with the same name and size will be created but it will be empty. The default is to save the contents of the volume.

```
volumes:
  - name: downpour-demo-unattached
    save_state: false
```

servers

The servers section lists the names and settings for the virtual machines to be exported.

Each item in the images list should be a mapping with a value for `name`. It can also contain an optional boolean value for `save_state`, indicating whether the contents of the VM should be exported. If `save_state` is false, a new VM with the same name and flavor will be created, but it will not contain any of the files from the current VM. The default is to save the contents of the volume.

If an optional `key_name` setting is given, the new VM will be initialized using that ssh keypair instead of the one already associated with the server. The keypair does not need to exist on the source system.

```
servers:
  - name: downpour-demo-tiny
    # Create the server using a separate key than
    # it was created with in tiny.yml.
    key_name: downpour-demo2
```

Contributing

There are two ways to contribute to downpour, using OpenStack's gerrit system and GitHub. The repository managed via gerrit and visible at https://git.openstack.org/cgit/openstack/downpour is the canonical repository.

Gerrit Process

If you would like to contribute using the standard OpenStack tools and processes, you should follow the steps in this page:

http://docs.openstack.org/infra/manual/developers.html

If you already have a good understanding of how the system works and your OpenStack accounts are set up, you can skip to the development workflow section of this documentation to learn how changes to OpenStack should be submitted for review via the Gerrit tool:

http://docs.openstack.org/infra/manual/developers.html#development-workflow

GitHub Process

If you would prefer to use GitHub, you may also submit pull requests to https://github.com/dhellmann/downpour. I will do the work to push the patch through the OpenStack review process on your behalf. That may involve changing some of the git metadata, such as committer. I will try to keep the author field intact so you retain credit. Please add the DCO signature to your commit messages, just to be safe.

The repository https://github.com/openstack/downpour is synced from gerrit. Pull requests to that repository will be closed automatically.

Bug Reports

Bugs should be filed on Launchpad, not GitHub:

https://bugs.launchpad.net/os-downpour

Glossary

resource file A YAML file containing explicitly identified resources to be exported. See *Resource File Format* for more details.

Index

R