PURESTORAGE

Configuring Pure Storage as a Cinder Storage Backend for Red Hat OpenStack 8 and 9

Simon Dodsley, OpenStack Solutions Architect

Introduction

This document covers the configuration process required to enable a single Pure Storage array to be used as an iSCSI Cinder Block Storage backend in Red Hat OpenStack 8 and 9 distributions.

The following items are assumed by this document:

- Ensure your Red Hat OpenStack Platform Overcloud has been correctly deployed through the Director, with a correctly functioning Block Storage service.
- Your Pure Storage Array should be available in the cloud management network or routed to the cloud management network with the Pure Storage iSCSI ports correctly configured.
- The Pure Storage management IP and iSCSI port IPs must have connectivity from the controller and compute nodes.
- You have obtained a privileged API token from the Pure Storage FlashArray that will be used by OpenStack Block Storage service.

When RHEL OpenSstack Platform is deployed through the Director, all major Overcloud settings must be defined and orchestrated through the Director as well. This will ensure that the settings persist through any Overcloud updates.

This document will not discuss the different deployment configurations possible with the backend. To learn more about these see the OpenStack Best Practises documents provided by Pure Storage.

At present, the Director only has the integrated components to deploy a single instance of a Pure Storage backend. Therefore this document only describes the deployment of a single backend.

Configure Pure Storage as an iSCSI Cinder backend

RHEL OpenStack Platform includes all the drivers and puppet manifests required for the Pure Storage FlashArray, however, there are two environment files required to be added to your Undercloud for full integration of the FlashArray into your Overcloud.

The two YAML files required can be found on the Pure Storage OpenConnect GitHub repository https://github.com/PureStorage-OpenConnect/tripleo-deployment-configs.

Obtain both YAML files from this repository and copy into the following locations in your Undercloud:

`cinder-pure-config.yaml` into `~stack/templates/`

`cinder-pure.yaml` into `/usr/share/openstack-tripleo-heat-templates/puppet/extraconfig/pre_deploy/controller/`

After you have copied these files you need to edit the `~/templates/cinder-pure-config.yaml` to populate it with your specific FlashArray data.

In the parameter_defaults section of this file add the management virtual IP address of your FlashArray into the `CinderPureSanIp` parameter and the API Token you had from your FlashArray into the `CinderPureAPIToken` parameter.

Optionally, you can configure your iSCSI FlashArray to use the CHAP security protocol by changing the default parameter setting of `false` to be `true` in the parameter `CinderPureUseChap`.

Deploying the Configured Backend

To deploy the single backend configured above, first, log in as the stack user to the Undercloud. Then deploy the backend (defined in the edited `~/templates/cinder-pure-config.yaml`) by running the following:

```
$ openstack overcloud deploy –templates -e ~/templates/cinder-pure-config.yaml
```

If you passed any extra environment files when you created the Overcloud you must pass them again here using the `-e` option to avoid making undesired changes to the Overcloud.

Test the Configured Backend

After deploying the backend, test whether you can successfully create volumes on it. Doing so will require loading the necessary environment variables first. These variables are defined in `/home/stack/overcloudrc` by default.

To load these variables, run the following command as the `stack` user:

```
$ source /home/stack/overcloudrc
```

You should now be logged into the Controller node. From there you can create a *volume type*, which can be used to specify the back nd you want to use (in this case the newly-defined backend). This is required in an OpenStack deployment where you have other backends enabled.

To create a volume type named `pure`, run:

```
$ cinder type-create pure
```

Next, map this volume type to the backend defined above and given the backend name `tripleo_pure` (as defined in through the **CinderPureBackendName** parameter) by running:

```
$ cinder type-key pure set volume_backend_name=tripleo_pure
```

You should now be able to create a 2GB volume on your newly defined backend by invoking its volume type. To do this run:

```
$ cinder create –volume-type pure 2
```

About the Author

As Global Solutions Architect, Simon is helping implement OpenStack technologies on Pure Storage. Core items include best practices, reference architectures and configuration guides.

With over 25 years of storage experience across all aspects of the discipline, from administration to architectural design, Simon has worked with all major storage vendors' technologies and organisations, large and small, across Europe and the USA as both customer and service provider. He also specialises in Data Migration methodologies assisting customers in their Pure Storage transition.

Blog: http://www.purestorage.com/blog/author/simon

PURESTORAGE

Pure Storage, Inc.
Twitter: @purestorage
www.purestorage.com

650 Castro Street, Suite #260
Mountain View, CA 94041

T: 650-290-6088
F: 650-625-9667

Sales: sales@purestorage.com
Support: support@purestorage.com
Media: pr@purestorage.com
General: info@purestorage.com