
SwiftStack Plugin for Fuel Documentation

Release 0.3-0.3.0-1

SwiftStack Inc.

June 17, 2016

CONTENTS

1	SwiftStack Fuel Plugin	1
1.1	Key terms, acronyms and abbreviations	1
1.2	Requirements	1
1.3	Limitations	1
1.4	Known issues	1
2	Installation Guide	3
2.1	Prepare Fuel Environment	3
2.2	Install Plugin	3
3	User Guide	4
3.1	SwiftStack Swift Cluster	4
3.2	SwiftStack Controller	4
3.3	Fuel Slave Nodes	5
3.4	Network summary	5
3.5	Use SwiftStack On-Premises Controller	5
3.6	Use SwiftStack Public Controller (Platform)	6
3.7	Deploying Mirantis OpenStack with a SwiftStack Swift cluster	7
3.8	Verification	14
4	Troubleshooting Guide	20
4.1	Source file (~/.openrc)	20
4.2	External Load Balancer	20
4.3	Swift endpoint in Keystone DB	20
4.4	401 Unauthorized issue from clients	20
4.5	403 Forbidden issue from clients through S3 APIs	21
5	Appendix	23

SWIFTSTACK FUEL PLUGIN

Allow Mirantis OpenStack environment able to use a running Swift cluster managed by a SwiftStack Controller. In SwiftStack fuel plugin, it disables the default Swift cluster on Controller and Primary-Controller nodes, and then reconfigures Swift API endpoints, Keystone, Glance settings and point them to a running SwiftStack Swift cluster.

1.1 Key terms, acronyms and abbreviations

SwiftStack On-Premises controller Provides a management service inside user's private place to help users to deploy and manage Swift clusters.

SwiftStack Public Controller Provides a public management service in public cloud that help users to deploy and manage Swift clusters.

SwiftStack Nodes A node installed SwiftStack agents and packages, that can be managed by a SwiftStack Controller, the node could be assigned a Swift role likes `Swift node` (Proxy/Account/Container/Object services are running in a single node)

1.2 Requirements

Requirement	Version
Mirantis OpenStack compatibility	8.0
A running SwiftStack Swift cluster	All versions Please enable Keystone Auth and Keystone Auth Token Support middlewares

1.3 Limitations

The plugin only supports a running SwiftStack Swift cluster and it able to reach from the OpenStack environment. Make sure you have the correct network configuration for the Swift cluster and Mirantis OpenStack environment before you enable this plugin.

1.4 Known issues

1. Need DNS server support to map Swift APIs hostname and IP

SwiftStack provides a software load balancer, which requires an external DNS server to operate. Please use DNS server instead of static hostname records in `/etc/hosts`.

2. Self-signed SSL certificates are not supported in the SwiftStack plugin

Self-signed certificates could be an issue when used in a production environment because all clients need to trust the cert to pass the TLS/SSL verification. It is highly recommended to use certificates signed by a known, trusted Certificate Authority if you require TLS/SSL for your Swift cluster endpoint.

INSTALLATION GUIDE

2.1 Prepare Fuel Environment

1. Prepare a Fuel Master node to install [MOS 8.0](#)
2. Download plugin from [Fuel Plugins Catalog](#)

2.2 Install Plugin

1. Copy plugin to the Fuel Master node

```
$ scp swiftstack-0.3.0.3.0-1.noarch.rpm root@<THE_FUEL_MASTER_NODE_IP>:/tmp/
```

2. Install SwiftStack plugin

```
[root@fuel ~]$ fuel plugins --install swiftstack-0.3.0.3.0-1.noarch.rpm
```

3. List all Fuel plugins and make sure it's running

```
[root@fuel ~]$ fuel plugins
```

id	name	version	package_version
2	swiftstack	0.3.0	4.0.0

SwiftStack provides **On-Premises** and **Public(Platform)** Controller to manage Swift clusters. Here is an overview for network topology between SwiftStack cluster, controller and Fuel slave nodes.

3.1 SwiftStack Swift Cluster

In a SwiftStack Swift cluster, each node has three networks which can be configured on its interfaces:

1. Outward-facing network:

The clients traffic comes into this interface, so if you consider putting an external load balancer in front of the cluster, you should add these outward-facing IPs to the load balancer pool.

2. Cluster-facing network:

The interface for Swift internal traffic (i.e. proxy-server from/to object-server).

3. Data replication network:

The interface for object-server replication.

If the node only has one network interface, you can assign all network interfaces to this interface. However, this could bottleneck performance, so we suggest using dedicated interface for these three networks. Check [Configure network](#) to get more detail.

3.2 SwiftStack Controller

SwiftStack provides two options for the Controller: the **Hosted Controller** (we called [Platform controller](#)) and the **On-Premises Controller**. The Hosted Controller is a as-a-service solution for customers who don't want to set up and maintain a SwiftStack Controller in their data center. This option requires the SwiftStack nodes have internet connectivity to be managed. If you don't have an account on the *Platform controller*, [sign up on our website](#).

The On-Premises controller is a SwiftStack controller deployed in a customer datacenter behind the customer's firewall. To obtain the On-Premises controller, please [contact SwiftStack Sales](#).

Before you can use the plugin, you will need to have deployed an On-Premises Controller, or have an account on the Hosted Controller.

SwiftStack nodes communicate with the SwiftStack Controller over OpenVPN connections, so the nodes must have network connectivity to reach the controller. If you have a firewall between your Nodes and the Controller, please check [SwiftStack Controller Security](#) and [SwiftStack Node Security](#) for information on how to configure the firewall.

Note: There is no difference when you use On-Premises or Platform controller to create you own Swift cluster, and do the integration with SwiftStack Fuel plugin. All configuration of SwiftStack Fuel plugin will be the same.

3.3 Fuel Slave Nodes

Fuel slave nodes have three major networks(public, storage, management) to configure, so if SwiftStack Nodes are connected to these three networks and use same IP range of [Fuel's configuration](#), you need to skip the IPs that are used for SwiftStack Nodes so there will be no conflict between the SwiftStack nodes and other Fuel nodes.

The SwiftStack Swift cluster is a standalone cluster, and each client should come from Outward-facing network (Fuel Public Network). The Fuel Managment network will be used for running user token validation between the Swift cluster and Keystone server. The SwiftStack cluster-facing and data replication network should be over Fuel Storage network.

3.4 Network summary

Please make sure the network configuration like:

1. Fuel controller nodes (Keystone, Glance) can talk to Swift Proxy-server (i.e., Proxy-only, PAC, PACO node) over Fuel Management network
2. Clients can talk to [Swift API IP Address](#) (Swift Proxy or External/Internal Load Balancer)
3. SwiftStack nodes can optionally talk to each over Fuel Storage network
4. SwiftStack nodes can talk to SwiftStack Controller via Management (SwiftStack) network (for On-Premises) or Public network (for public Swiftstack Controller)

Note: We only use one PACO (Proxy/Account/Comtainer/Object) nodes to deploy an all-in-one Swift cluster in this document which is a considered a minimum deployment. In real environment, as the cluster scales, it might be necessary to assign nodes into separate Proxy/Account/Container/Object tiers. If the Fuel Storage network does not have adequate bandwidth to support Replication & Cluster-Facing traffic, these interfaces can be on a network external to Fuel

3.5 Use SwiftStack On-Premises Controller

Please setup an On-Premises SwiftStack controller first, and then setup a single node Swift cluster with SwiftStack controller, here is our [quick start guide](#).

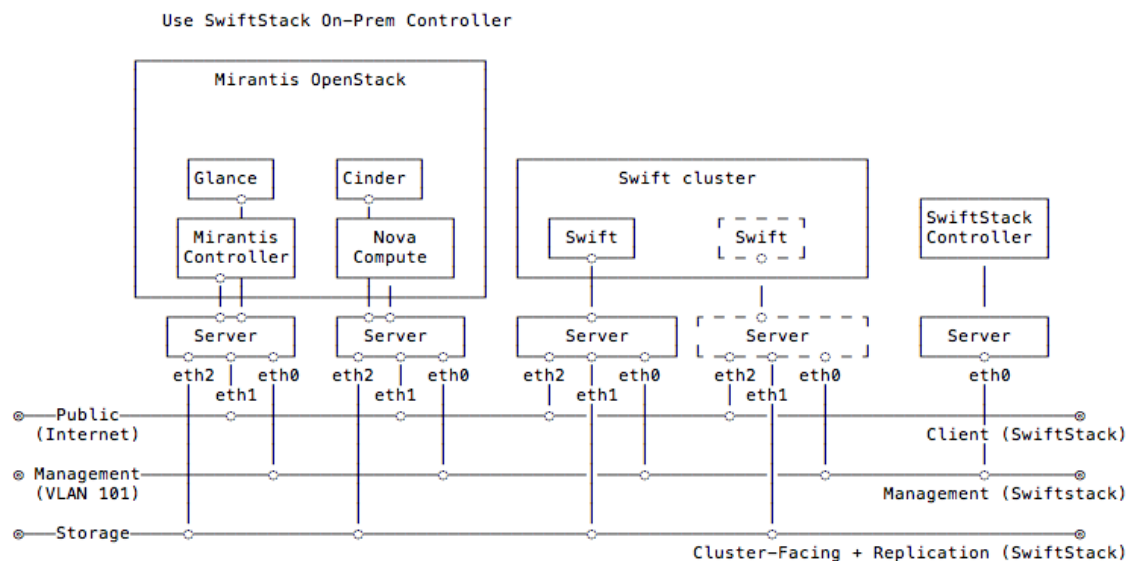
- 1 SwiftStack On-Premises controller
- 1 Swift cluster (single node)

Also prepare a Fuel environment using Slave nodes according to the [Fuel Install Guide](#).

Note: In this diagram, the Swift cluster is also connected to Fuel Storage network for SwiftStack cluster-facing and data replication network, if you have performance concern, please consider to separate Swift cluster-facing and data replication network out of Fuel networks. That prevents network starvation on

Fuel Storage network when Swift service daemons are moving data or clients upload large data into the Swift cluster.

Also, SwiftStack Nodes need to communicate with the On-Premises controller over Fuel Management network, so please make sure the On-Premises controller is also connected to Fuel Management network. You can run a CLI command `ssdiag` on SwiftStack nodes to check the connectivity between SwiftStack Nodes and Controller.



3.6 Use SwiftStack Public Controller (Platform)

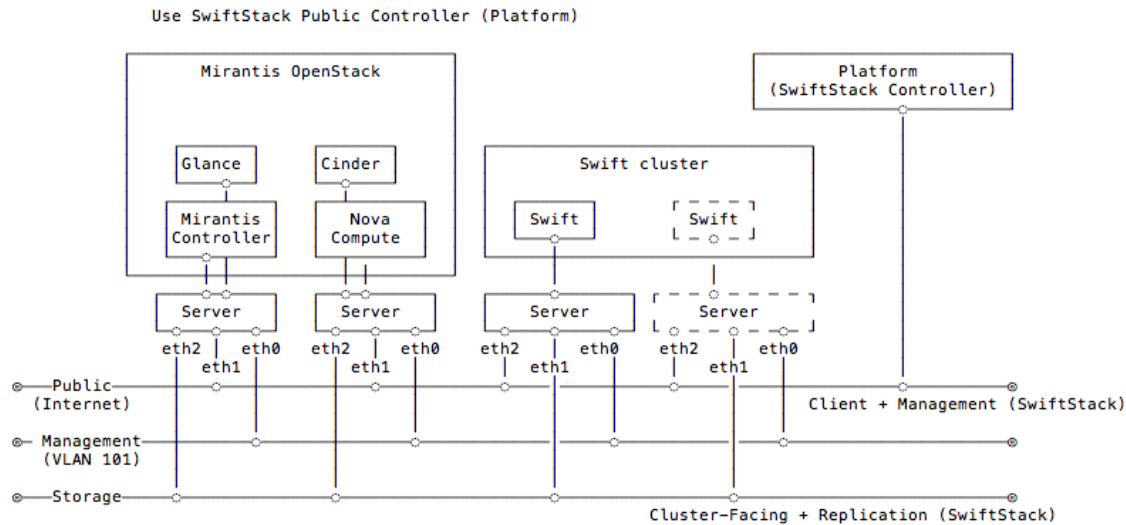
Please setup a single node Swift cluster with our public controller, here is our [quick start guide](#).

- 1 Swift cluster (single node)

Also prepare a Fuel environment using Slave nodes according to the [Fuel Install Guide](#).

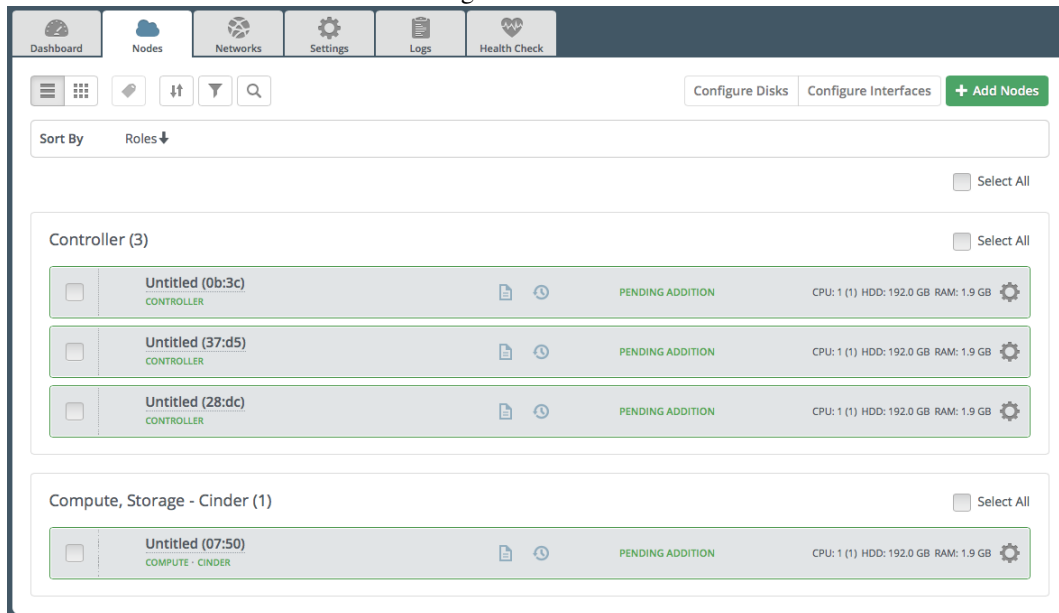
Note: In this diagram, the Swift cluster is also connected to Fuel Storage network for SwiftStack cluster-facing and data replication network, if you have performance concern, please consider to separate Swift cluster-facing and data replication network out of Fuel networks. That prevents network starvation on Fuel Storage network when Swift service daemons are moving data or clients upload large data into the Swift cluster.

Also, SwiftStack Nodes need to communicate with SwiftStack Public controller over Fuel Public network, so please make sure SwiftStack Nodes are able to reach Internet.



3.7 Deploying Mirantis OpenStack with a SwiftStack Swift cluster

1. Create a new environment with available Slave nodes:
 - Select **Liberty on Ubuntu Trusty (14.04)** as the distribution
 - Select **Neutron with VLAN segmentation** as the networking setup
 - Use all default settings
 - Select node roles according to the [Fuel Install Guide](#).



1. Go to the Settings tab of the Fuel Web UI
2. Scroll down to **Storage** section
3. Select **Enable SwiftStack Swift Cluster Integration** checkbox and fill in the following parameters:
 - (a) **Enable TLS for Swift endpoints:**

This option will use HTTPS for Swift endpoints including public, admin and internal urls.

(b) **Swift API IP Address** and **Swift API hostname**:

The IP address is the default value for Swift endpoints, if you fill up the API hostname, that overwrites Swift endpoints with hostname.

(c) **Use Swift as Glance backend** and **Enable upload test**:

These two options for Glance integration

Note: If **Use Swift as Glance backend** is disabled, please consider enabling **Ceph RBD for images (Glance)** or other storage for Glance backend.

If **Enable upload test** is disabled, Fuel won't upload testVM image(cirros-testvm) to Glance and store in Swift cluster. That means some **health checks** will fail (i.e., Create volume and boot instance from it.)

The settings in below,

(a) Swift API IP Address: 172.16.0.100.

(b) Use Swift as Glance backend: Checked

(c) Enable upload test: Checked

☒ **Enable SwiftStack Swift cluster integration**

Versions ☒ 0.3.0

Swift API IP Address

If enabled,
* Swift deployment in controller nodes will be disabled
* Swift API endpoint in Keystone DB will redirect to external Swift cluster.

Swift API Hostname

Swift API Hostname



Enable TLS for Swift endpoints

Configures all Swift endpoint urls (public/internal/admin) with TLS support.



Use Swift as Glance backend 

Config Glance backend storage to Swift cluster



Enable upload test

Upload cirros base image to Glance and store it in Swift when the deployment is done

1. Go to the **Networks** tab, scroll down to **Public** section and then modify **IP Range** to skip the IPs of SwiftStack Outward-facing and Swift API IP Address.

Here is our network configuration for a single SwiftStack node.

Swift Drives

Network

Change Zone

Change Role

Delete Node

Disable

✓ Success

Successfully communicated with node.

Edit Network Interfaces

If you have multiple interfaces, you may specify how interfaces are used by Swift services. **The same interface may be used for external an**

Outward-facing interface*

✓ eth2 - 172.16.0.100

eth0.101 - 192.168.0.100

eth1 - 192.168.1.100

is used for proxy server and load-balancing services

Cluster-facing interface*

eth1 - 192.168.1.100

The cluster-facing interface is used for intra-cluster communication among Swift daemons.

Data replication interface*

eth1 - 192.168.1.100

The data replication interface is used to propagate modified object data to other replicas.

Reassign Interfaces

Skip *172.16.0.100* (Outward-facing) on Public network.

Public

The Public network allows inbound connections to VMs (Controllers and Tenant VMs) from external networks (e.g., the Internet) as well as outbound connections from VMs to the external networks.

CIDR

172.16.0.0/24

☐ Use the whole CIDR

Start

End

IP Range

172.16.0.2

172.16.0.90

+

Gateway

172.16.0.1

Use VLAN tagging

☐

Also, skip the IPs of SwiftStack Cluster-facing and data replication in **IP Range** of **Storage** section, so skip *192.168.1.100* (Cluster-facing/data replication) on Storage network

Storage

The Storage network is used to provide storage services such as replication traffic from Ceph. The Management network is used for Ceph Public traffic.

CIDR

192.168.1.0/24

☐ Use the whole CIDR

Start

End

IP Range

192.168.1.2

192.168.1.90

+

Gateway

192.168.1.1

Use VLAN tagging

☐

If you use SwiftStack On-Premises Controller, you need to do same thing in **Management** section to skip the IPs of SwiftStack nodes and On-Premises Controller.

Management

The Management network is primarily used for OpenStack Cloud Management. It is used to access OpenStack services (nova-api, OpenStack dashboard, etc).

CIDR ☐ Use the whole CIDR

IP Range

Gateway

Use VLAN tagging ☒

Note: If you have more than one Proxy server (Proxy-only, PAC, PACO nodes), or you use external/internal load balancer (Swift API IP Address) for your Swift cluster, please consider to skip these IPs.

- Outward-facing IP from SwiftStack Controller UI

Swift Nodes

Enabled? ^	Diagnostics Check ⇅	Hostname ^	Role ⇅	Zone ⇅	Outward-Facing IP ⇅	Cluster-Facing IP ⇅	Data Replication IP ⇅
Yes	<input checked="" type="button" value="✓ Okay"/>	dev21.Trusty <input type="button" value="Manage"/> <input type="button" value="Monitor"/>	Swift Node	Region 1, Zone 1 (r1z1)	172.16.0.100	192.168.1.100	192.168.1.100

- Swift API IP address (Load balancer IP) from SwiftStack Controller UI

Network Configuration ?




Will your external clients need to connect with HTTPS? ☒ no ☐ yes

How will your external clients connect? ? ☐ External Load Balancer ☐ SwiftStack Virtual Load Balancer (cannot be used with IPv6) ☒ No Load Balancer (e.g. Single Node "Cluster" or Round-Robin DNS)

Cluster API IP Address* ?

Cluster API Hostname ?

- Go to the **Nodes** tab of the Fuel Web UI, drag **Storage** interface to **eth2** and untagged the VLAN for all nodes:

<input type="checkbox"/>	 Name: enp0s3 Speed: 1.0 Gbps	Admin (PXE)	Management VLAN ID:101	Private VLAN IDs:1000-1030
Offloading Modes: Default				MTU Default
<input type="checkbox"/>	 Name: enp0s8 Speed: 1.0 Gbps	Public		
Offloading Modes: Default				MTU Default
<input type="checkbox"/>	 Name: enp0s9 Speed: 1.0 Gbps	Storage		
Offloading Modes: Default				MTU Default

Note: The management network is tagged with VLAN ID 101 by default, so you also need to configure VLAN ID for interfaces of SwiftStack Nodes and On-Premises Controller

3. Find the settings from deployment information:

- Keystone IP Address (management_vip)
- Swift password

Please login to the Fuel Master node and create a script file called **swiftstack.sh** with contents in below,

```
#!/bin/bash
cd /root
fuel env
echo -e "\n\n"
read -p "Which environment?" environment

# Export environment
fuel deployment --env $environment --default

# put error checking here
SwiftIP=$(sed -e '/ management:\/,/ipaddr:\/!d' \
  deployment_*/primary-controller*.yaml \
  | grep ipaddr | awk '{print $2}')
SwiftPW=$(sed -e '/swift:\/,/user_password:\/!d' \
  deployment_*/primary-controller*.yaml \
  | grep user_password | awk '{print $2}')

echo "Configure Keystone Auth Token Support middleware in below :"
echo "-----"
echo " identity_url      : http://$SwiftIP:5000/"
echo " auth_url          : http://$SwiftIP:5000/"
echo " auth_url (for s3)  : http://$SwiftIP:35357/"
echo " admin_user        : swift"
echo " admin_password     : $SwiftPW"
```

Change permissions and run it.

```
[root@fuel ~]$ chmod +x swiftstack.sh
[root@fuel ~]$ ./swiftstack.sh

id | status | name      | release_id | pending_release_id
---|-----|-----|-----|-----
5  | new    | MOS 8.0   | 2          | None
```

```
Which environment?5
Default deployment info was downloaded to /root/deployment_5
Configure Keystone Auth Token Support middleware in below :
-----
identity_url      : http://192.168.0.2:5000/
auth_url          : http://192.168.0.2:5000/
auth_url (for s3) : http://192.168.0.2:35357/
admin_user        : swift
admin_password     : v4LiGbh6xPU0vtqXQSMedjxc
```

4. Once we get Keystone IP (192.168.0.2) and Swift user's password (v4LiGbh6xPU0vtqXQSMedjxc), let's login to SwiftStack Controller UI to configure Swift cluster

- Go to the **Middleware** tab, enable and configure **Keystone Auth Token Support** middleware as below:

```
identity_url:      http://192.168.0.2:5000/
auth_url:          http://192.168.0.2:5000/
admin_user:        swift
admin_password:    v4LiGbh6xPU0vtqXQSMedjxc
admin_tenant_name: services
```

Keystone Auth Token Support

Configuring Keystone Auth Token Support

This middleware is required for Keystone Authentication/Authorization (along with the "Keystone Auth" middleware).

Settings

☒ Enabled

identity_uri	<input type="text" value="http://192.168.0.2:5000/"/>	Complete admin Identity API endpoint.
auth_uri	<input type="text" value="http://192.168.0.2:5000/"/>	Complete public Identity API endpoint.
admin_user	<input type="text" value="swift"/>	Service username.
admin_password	<input type="text" value="v4LiGbh6xPU0vtqXQSMedjxc"/>	Service user password.
admin_tenant_name	<input type="text" value="services"/>	Service tenant name.

- Enable and configure **Keystone Auth** middleware as below:

```
reseller_admin_role: admin
```

Keystone Auth

Configuring Keystone Authorization

This middleware is required for Keystone Authentication/Authorization (along with the "Keystone Auth Token Sup". The "reseller_prefix" must match the value used in your Keystone endpoint's publicurl and privateurl and must not be empty. For example, if your Keystone endpoint's publicurl was `http://192.168.22.100:80/v1/KEY_${tenant_id}services` then the "reseller_prefix" should be `KEY_`.

Settings

☒ Enabled

operator_roles	admin, swiftoperator
reseller_prefix	KEY_
reseller_admin_role	admin

- If you want your Swift cluster to support S3 APIs, please also enable [Swift S3 Emulation Layer Middleware](#) and [Swift3 Keystone Integration Middleware](#).

(a) Enable Swift S3 Emulation Layer Middleware, select `Enabled` checkbox and submit it.

Swift3 -- S3 Emulation Layer

This middleware allows your Swift cluster to emulate the Amazon S3 API. For more information, see the [documentation](#).

 To support multi-part uploads, the [SLO middleware](#) must also be enabled.

 To support Keystone authentication with Swift3, the [Swift3 Keystone Integration middleware](#) must also be enabled.

Settings

☒ Enabled

location	US	Region name to return for the GET Bucket location API
dns_compliant_bucket_names	False	Require DNS-compliant bucket names. In AWS, DNS-compliant bucket names are required for all regions except US Standard.
s3_acl	False	Experimental: Use custom ACLs to achieve best S3 compatibility.
allow_no_owner	False	When s3_acl is enabled, expose containers that have no ownership information (i.e., containers created via the Swift API). Such containers will be globally accessible via the S3 API.
check_bucket_owner	False	When s3_acl is enabled, only return buckets owned by the requesting user during GET Service operation. This may cause significant performance degradation; only enable it if your use-case demands it.

- (b) Enable Swift3 Keystone Integration Middleware, select Enabled checkbox and fill **http://192.168.0.2:35357/** to auth_url and then submit it

auth_url (for s3): http://192.168.0.2:35357/

Swift3 Keystone Integration

The Swift3 Keystone Integration middleware is used in conjunction with the [Swift3 S3-emulation middleware](#) to allow Keystone authentication with a cluster emulating the Amazon S3 API.

Settings

☒ Enabled

auth_uri

Complete public Identity API endpoint.

reseller_prefix

1. Push configure settings to SwiftStack Swift cluster.
2. Network verification check Please check Fuel network configuration and SwiftStack settings before you deploy the OpenStack environment:
 - (a) SwiftStack Nodes should able to reach Keystone endpoint (internalURL) on Management network.
 - (b) Clients should able to reach SwiftStack Nodes over Public network.
 - (c) All IPs of SwiftStack Nodes (includes Load Balancer) should be skip in Fuel networks.
 - (d) If you use VLAN, please check VLAN settings on each node
3. Get back to the Fuel Web UI and deploy your OpenStack environment.
4. Once Mirantis OpenStack environment is done, you will see the SwiftStack plugin is also deployed.



3.8 Verification

Please run the verification steps below to ensure your SwiftStack plugin is configured properly:

3.8.1 Check API endpoints with Keystone CLI:

```
### Login to Controller node
~$ source ~/openrc
~$ cat ~/openrc | grep OS_AUTH_URL
export OS_AUTH_URL='http://192.168.0.2:5000/'
```



```
##
## Correct OS_AUTH_URL, append 'v2.0' in the end of line
##
~$ export OS_AUTH_URL='http://192.168.0.2:5000/v2.0'

~$ keystone endpoint-list |grep KEY
| b858f41ee3704f32a05060932492943b | RegionOne |
http://172.16.0.100:80/v1/KEY_%(tenant_id)s |
http://172.16.0.100:80/v1/KEY_%(tenant_id)s |
http://172.16.0.100:80/v1/KEY_%(tenant_id)s |
19966ec76f0d455d94caa87d9569a347 |
```

3.8.2 Verify Swift cluster, Keystone and Glance integration through Swift cli

Check admin account

```
# Login to one of nodes of Swift cluster.

# Test admin account
~$ cat rc.admin
export ST_AUTH=http://192.168.0.2:5000/v2.0
export ST_USER=admin:admin
export ST_KEY=admin
export ST_AUTH_VERSION=2

~$ source rc.admin
~$ swift stat -v
      StorageURL: http://172.16.0.100:80/v1/KEY_9f12acc2fc1c4b4cb
                    75916b2724e2903
      Auth Token: gAAAAABXV5CFn_cx-Y2pJK4de7XDDXvEmfo4SlhmCAAOWeG
                    -RHLkSCCqfc_mGHoJ-7ee4cACSzzx5bXi jCtopbRA-Mh2vr
                    _SGK9GKSBlAIt-Q1kSsUJTNgjL0T6Hws66r7gh4PmiTFwhO
                    uhV9BTswzF9GzIHdUpKusd3jhrc1cc9ipQdnF_bF1c
      Account: KEY_9f12acc2fc1c4b4cb75916b2724e2903
      Containers: 0
      Objects: 0
      Bytes: 0
      X-Put-Timestamp: 1465356423.33437
      X-Timestamp: 1465356423.33437
      X-Trans-Id: txf07064e2471544b29f84d-0057579086
      Content-Type: text/plain; charset=utf-8
```

Check glance account when Use Swift as Glance backend is enabled

```
# Find glance password from deployment yaml
[root@fuel ~]$ sed -e '/glance:/,/user_password:!/d' \
                    deployment_*/primary-controller*.yaml

glance:
  db_password: XkxjTF4LKu7FgaY2YyXlUMI
  image_cache_max_size: '13928339865'
  user_password: iqxWViMcHUjxbWD0hqkvjbon

# Test glance account
~$ cat rc.glance
export ST_AUTH=http://192.168.0.2:5000/v2.0
```

```

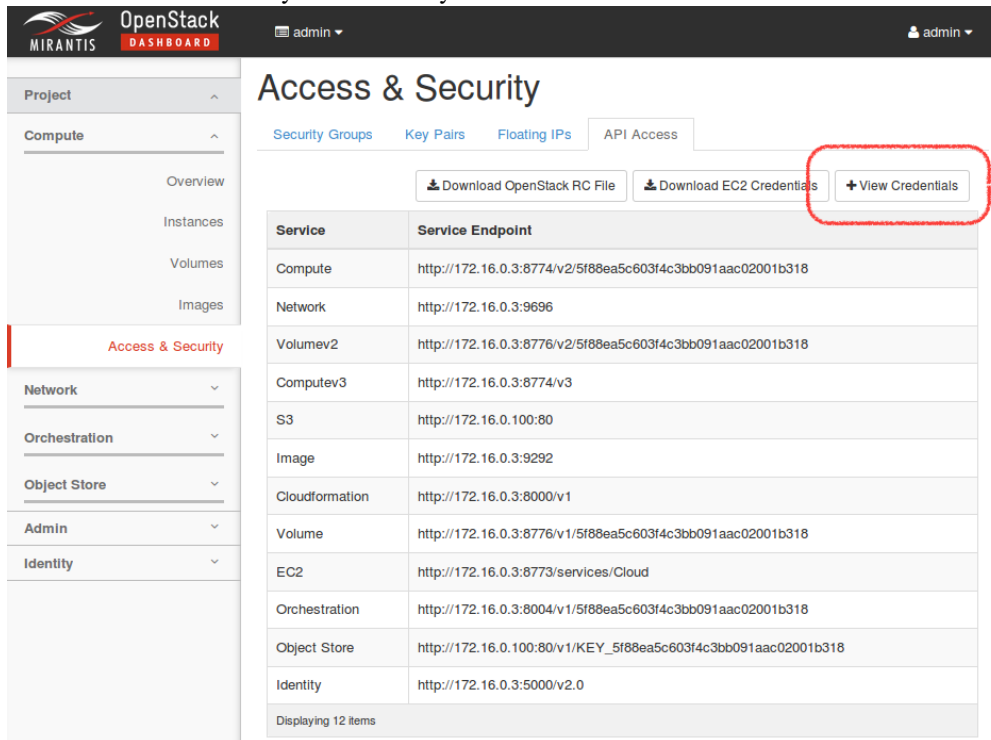
export ST_USER=services:glance
export ST_KEY=iqxWViMcHUjxbWD0hqkvjbon
export ST_AUTH_VERSION=2

~$ swift stat -v
StorageURL: http://172.16.0.100:80/v1/KEY_63bda2
0adcb24e2eb37d2dcb13d2a29b
Auth Token: gAAAAABXV4-d_FKAboXyxKooWVdmbiDCLtgX
0diSqMed9gzXTPHkt5ko7AMffp28iKBX984g
KXqUKk82pjQ9tpSIu-TA9cTLoZYz0Cabp9Y
s-zIH-BJOP1DZsEaOIOB8wTrvU2i_eGyPKgN
25iaARIahh2MYUkNU21Xfzg7Q7bQ1wvFFhMo
d7g
Account: KEY_63bda20adcb24e2eb37d2dcb13d2a29b
Containers: 1
Objects: 1
Bytes: 13287936
Containers in policy "standard-replica": 1
Objects in policy "standard-replica": 1
Bytes in policy "standard-replica": 13287936
Accept-Ranges: bytes
X-Account-Project-Domain-Id: default
X-Timestamp: 1465322384.96195
X-Trans-Id: txa59a5b16d6724fc68adb7-0057578f9e
Content-Type: text/plain; charset=utf-8

```

3.8.3 Verify S3 APIs, Swift cluster and Keystone

Find EC2 access key and secret key from Horizon



The screenshot shows the OpenStack Horizon dashboard with the 'Access & Security' section. The 'API Access' tab is selected, and the 'View Credentials' button is highlighted with a red circle. The table below lists the service endpoints for various OpenStack services.

Service	Service Endpoint
Compute	http://172.16.0.3:8774/v2/5f88ea5c603f4c3bb091aac02001b318
Network	http://172.16.0.3:9696
Volumev2	http://172.16.0.3:8776/v2/5f88ea5c603f4c3bb091aac02001b318
ComputeV3	http://172.16.0.3:8774/v3
S3	http://172.16.0.100:80
Image	http://172.16.0.3:9292
Cloudformation	http://172.16.0.3:8000/v1
Volume	http://172.16.0.3:8776/v1/5f88ea5c603f4c3bb091aac02001b318
EC2	http://172.16.0.3:8773/services/Cloud
Orchestration	http://172.16.0.3:8004/v1/5f88ea5c603f4c3bb091aac02001b318
Object Store	http://172.16.0.100:80/v1/KEY_5f88ea5c603f4c3bb091aac02001b318
Identity	http://172.16.0.3:5000/v2.0

Displaying 12 items

When you click View Credentials, it shows a dialog for EC2 keys in below,

User Credentials

User Name: admin

Project Name: admin

Project ID: 5f88ea5c603f4c3bb091aac02001b318

Authentication URL: http://172.16.0.3:5000/v2.0

EC2 URL: http://172.16.0.3:8773/services/Cloud

S3 URL: http://172.16.0.100:80

EC2 Access Key: e8f3617f41d34d02a7ba129f8581a3b6

EC2 Secret Key: 85f2ae90a9614a8b832747af3c6e6c9b

Close

Or you can use keystone CLI to get EC2 keys.

```
~$ keystone ec2-credentials-list
+-----+-----+-----+
| tenant |          access          |          secret          |
+-----+-----+-----+
| admin  | e8f3617f41d34d02a7ba129f8581a3b6 | 85f2ae90a9614a8b832747af3c6e6c9b |
+-----+-----+-----+
```

Upload single file to a container

```
~$ swift upload test rc.admin
~$ swift stat test rc.admin
  Account: KEY_5f88ea5c603f4c3bb091aac02001b318
  Container: test
  Object: rc.admin
  Content Type: application/octet-stream
  Content Length: 115
  Last Modified: Wed, 15 Jun 2016 12:48:44 GMT
  ETag: ed6eb254c7a7ba2cba19728f3fff5645
  Meta Mtime: 1465994722.799261
  Accept-Ranges: bytes
  X-Timestamp: 1465994923.49250
  X-Trans-Id: tx3dd9b89f2ebc4579857b7-005761743f
```

Please create a script file called s3get.sh and add contents in below,

```
#!/bin/bash

url=$1
s3key=$2
s3secret=$3
bucket=$4
file=$5
```

```
# Path style
resource="/${bucket}/${file}"
fullpath="${url}/${bucket}/${file}"

dateValue=`date -u +%a,%d %h %Y %T %Z`

echo ${dateValue}
echo ${resource}

stringToSign="GET\n\n\n${dateValue}\n${resource}"
signature=`echo -en ${stringToSign}|openssl sha1 -hmac ${s3secret} -binary|base64`
curl -I -v -X GET \
  -H "Date: ${dateValue}" \
  -H "Authorization: AWS ${s3key}:${signature}" \
  ${fullpath}
```

Try to retrieve the object (container: test, object: rc.admin) through S3 APIs.

```
~$ ./s3get.sh http://172.16.0.100:80 \
>                                     e8f3617f41d34d02a7ba129f8581a3b6 \
>                                     85f2ae90a9614a8b832747af3c6e6c9b \
>                                     test rc.admin
Wed, 15 Jun 2016 15:25:51 UTC
/test/rc.admin
* Hostname was NOT found in DNS cache
* Trying 172.16.0.100...
* Connected to 172.16.0.100 (172.16.0.100) port 80 (#0)
> GET /test/rc.admin HTTP/1.1
> User-Agent: curl/7.35.0
> Host: 172.16.0.100
> Accept: */*
> Date: Wed, 15 Jun 2016 15:25:51 UTC
> Authorization: AWS e8f3617f41d34d02a7ba129f8581a3b6:tHnRZjiCzPzeJhs8SAQ8msBWH3Y=
>
< HTTP/1.1 200 OK
HTTP/1.1 200 OK
< Content-Length: 115
Content-Length: 115
< x-amz-id-2: tx43598dcd71274707a7adc-0057617380
x-amz-id-2: tx43598dcd71274707a7adc-0057617380
< x-amz-meta-mtime: 1465994722.799261
x-amz-meta-mtime: 1465994722.799261
< Last-Modified: Wed, 15 Jun 2016 12:48:44 GMT
Last-Modified: Wed, 15 Jun 2016 12:48:44 GMT
< ETag: "ed6eb254c7a7ba2cba19728f3fff5645"
ETag: "ed6eb254c7a7ba2cba19728f3fff5645"
< x-amz-request-id: tx43598dcd71274707a7adc-0057617380
x-amz-request-id: tx43598dcd71274707a7adc-0057617380
< Content-Type: application/octet-stream
Content-Type: application/octet-stream
< X-Trans-Id: tx43598dcd71274707a7adc-0057617380
X-Trans-Id: tx43598dcd71274707a7adc-0057617380
< Date: Wed, 15 Jun 2016 15:25:52 GMT
Date: Wed, 15 Jun 2016 15:25:52 GMT

<
* Excess found in a non pipelined read: excess = 115 url = /test/rc.admin
                                         (zero-length body)
```

```
* Connection #0 to host 172.16.0.100 left intact
```

TROUBLESHOOTING GUIDE

4.1 Source file (~/.openrc)

If you try run swift cli in controller nodes, please check OS_AUTH_URL is correct. If the value is `http://<KEYSTONE_VIP>:5000/`, please correct it to `http://<KEYSTONE_IP>:5000/v2.0` as following.

```
root@node-17:~# cat openrc
#!/bin/sh
...
export OS_AUTH_URL='http://<KEYSTONE_VIP>:5000/v2.0'
...
```

4.2 External Load Balancer

If there is a external load balancer in front of you Swift cluster, and you configure the swift cluster with it. Please fill the external LB IP for Swift API IP Address in [plugin page](#).

4.3 Swift endpoint in Keystone DB

Before you upload any VM image to Glance, we suggest to check the Swift endpoint in Keystone DB first. Make sure the Swift endpoint is correct.

For swift endpoint, please make sure the endpoints (publicurl and internalurl) look like `http://<SWIFT_API_IP>:80/v1/KEY_%(tenant_id)s`.

```
$ openstack endpoint list | grep swift
```

4.4 401 Unauthorized issue from clients

If any client runs into 401 Unauthorized issue, please use Swift CLI verify it again and make sure the settings of middlewares in Swift cluster are correct.

For example, if you get a error with `swift stat`.

```
$ swift stat
Account HEAD failed: http://10.200.5.5:80/v1/KEY_32f0b6cd7299412e9f7966b324fb6aea
401 Unauthorized
```

Try to use `--debug` to get more details.

```
$ swift --debug stat -v
..<SKIP>..
INFO:requests.packages.urllib3.connectionpool:Starting new HTTP connection
(1): 10.200.5.5
DEBUG:requests.packages.urllib3.connectionpool:"HEAD /v1/KEY_32f0b6cd7299412e9f7966b324fb6aea HTTP/1.1" 401 0
INFO:swiftclient:REQ: curl -i http://10.200.5.5:80/v1/KEY_32f0b6cd7299412e9f7966b324fb6aea -I -H "X-Auth-Token: gAAAAABXMels87mzqZK1Ee8hyJQ86fv9NDcSCHkCLk-PTQfa353J5t3N4EL-OCbZuqt6hRFBJehUozgF4FNNd5Q_rfXBejo817U_Ff6mAy6-hP2l0KWbxON1mfZL_UCfjjWclrSD2-bK38JvTfrqWdM99cqfdMBDZS-wqHnldZz00g2r-Kzxcc"
INFO:swiftclient:RESP STATUS: 401 Unauthorized
INFO:swiftclient:RESP HEADERS: [('Content-Length', '0'), ('Connection', 'keep-alive'), ('X-Trans-Id', 'txecd82ae98e714ef0b4c0c-005731ed6c'), ('Date', 'Tue, 10 May 2016 14:17:16 GMT'), ('Content-Type', 'text/html; charset=UTF-8'), ('Www-Authenticate', 'Swift realm="KEY_32f0b6cd7299412e9f7966b324fb6aea", Keystone uri=\'http://10.200.7.2:5000/\']')
ERROR:swiftclient:Account HEAD failed: http://10.200.5.5:80/v1/KEY_32f0b6cd7299412e9f7966b324fb6aea 401 Unauthorized
Traceback (most recent call last):
  File "/usr/lib/pymodules/python2.7/swiftclient/client.py", line 1261, in _retry
    rv = func(self.url, self.token, *args, **kwargs)
  File "/usr/lib/pymodules/python2.7/swiftclient/client.py", line 541, in head_account
    http_response_content=body)
ClientException: Account HEAD failed: http://10.200.5.5:80/v1/KEY_32f0b6cd7299412e9f7966b324fb6aea 401 Unauthorized
Account HEAD failed: http://10.200.5.5:80/v1/KEY_32f0b6cd7299412e9f7966b324fb6aea 401 Unauthorized
```

If the keystone IP and Swift user and password are correct, please *find the password from deployment yaml files* and *config Swift middlewares* first. Once that's done, please *verify it with Swift CLI*.

4.5 403 Forbidden issue from clients through S3 APIs

When you saw clients get 403 response from S3 APIs, please check **Swift3 Keystone Integration Middleware** first and make sure `auth_url` is point to keystone amdinurl.

```
~$ ./s3get.sh test http://172.16.0.100:80 \
> e8f3617f41d34d02a7ba129f8581a3b6 \
> 85f2ae90a9614a8b832747af3c6e6c9b \
> test rc.admin
Wed, 15 Jun 2016 14:15:14 UTC
/test/rc.admin
* Hostname was NOT found in DNS cache
* Trying 172.16.0.100...
* Connected to 172.16.0.100 (172.16.0.100) port 80 (#0)
> GET /test/rc.admin HTTP/1.1
> User-Agent: curl/7.35.0
```

```

> Host: 172.16.0.100
> Accept: */*
> Date: Wed, 15 Jun 2016 14:15:14 UTC
> Authorization: AWS e8f3617f41d34d02a7ba129f8581a3b6:RG6hF77QUN/fmMMLSFP5SauMD7Q=
>
< HTTP/1.1 403 Forbidden
HTTP/1.1 403 Forbidden
< x-amz-id-2: tx6359093a27f642db8a398-00576162f3
x-amz-id-2: tx6359093a27f642db8a398-00576162f3
< x-amz-request-id: tx6359093a27f642db8a398-00576162f3
x-amz-request-id: tx6359093a27f642db8a398-00576162f3
< Content-Type: application/xml
Content-Type: application/xml
< X-Trans-Id: tx6359093a27f642db8a398-00576162f3
X-Trans-Id: tx6359093a27f642db8a398-00576162f3
< Date: Wed, 15 Jun 2016 14:15:15 GMT
Date: Wed, 15 Jun 2016 14:15:15 GMT
< Transfer-Encoding: chunked
Transfer-Encoding: chunked

```

And if that is still can't solve the problem or you see other error codes (500 Internal server error, etc.) from S3 APIs, please try to check the swift logs (/var/log/swift/all.log) to see is any exception on that. And you will have a X-Trans-Id for each request, so please use that to grep Swift logs likes,

```

# Please login to SwiftStack Nodes
$ grep tx6359093a27f642db8a398-00576162f3 /var/log/swift/all.log

```

And send the output to [SwiftStack Support](#).

APPENDIX

- SwiftStack Quick Start Guide: <https://swiftstack.com/docs/install/index.html>
- SwiftStack Admin Guide - Middleware: <https://swiftstack.com/docs/admin/middleware.html>
- SwiftStack Admin Guide - Keystone Auth Middleware: https://swiftstack.com/docs/admin/middleware/keystone_auth.html
- SwiftStack Admin Guide - Keystone Auth Token Middleware: https://swiftstack.com/docs/admin/middleware/keystone_auth_token.html
- SwiftStack Admin Guide - Swift S3 Emulation Layer Middleware: https://swiftstack.com/docs/admin/middleware/s3_middleware.html
- SwiftStack docs: <https://swiftstack.com/docs/>