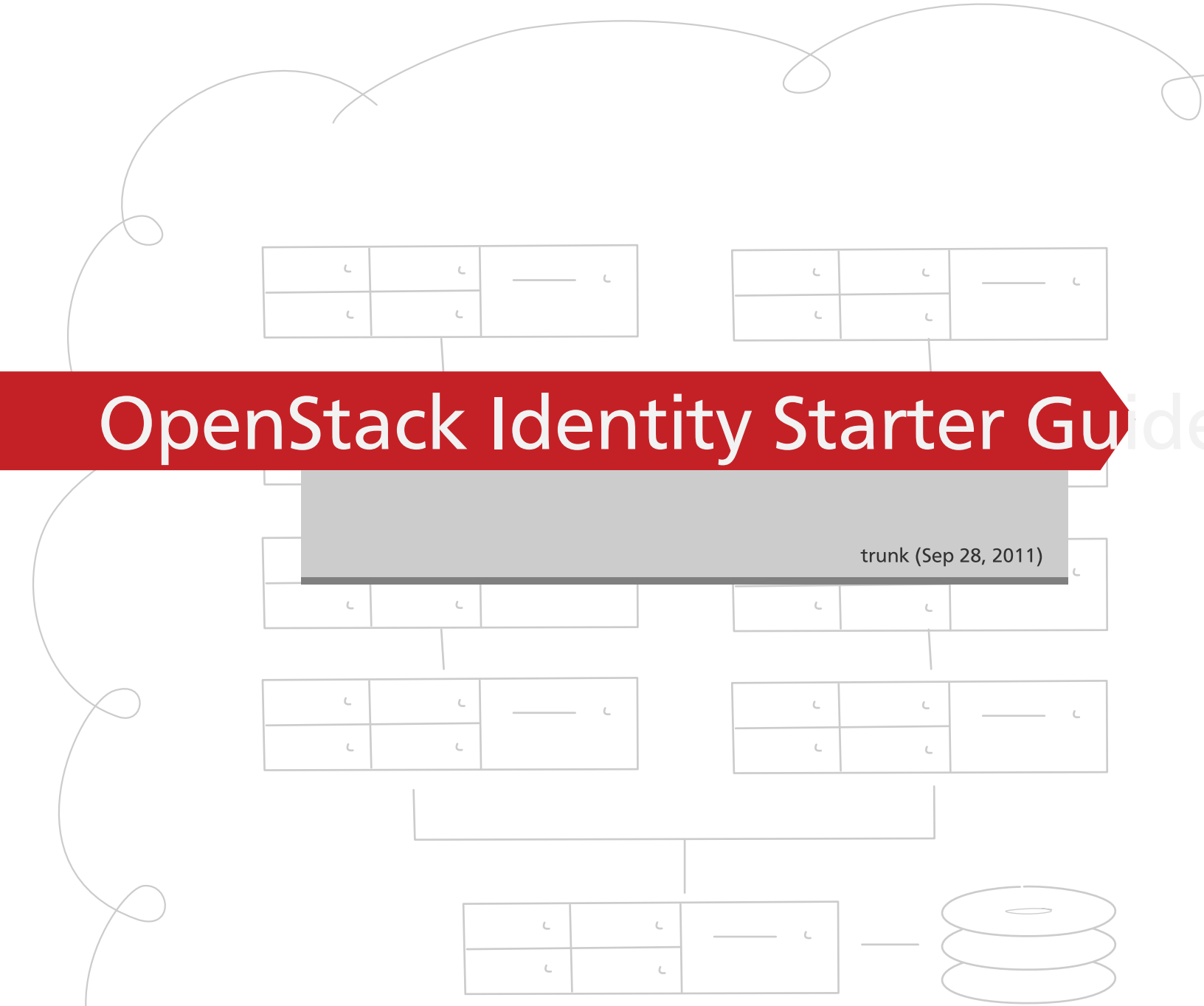

OpenStack Identity Starter Guide

trunk (Sep 28, 2011)

openstack™

docs.openstack.org

OpenStack Identity Starter Guide

trunk (2011-09-28)

OpenStack™ Identity Service offers open source software for identity management for cloud users and administrators. This manual provides guidance for installing, managing, and understanding the software that runs OpenStack Identity Service.

Table of Contents

1. Quick Guide to Getting Started with Keystone

The OpenStack Identity Service provides services for authenticating and managing user, account, and role information for OpenStack clouds running on OpenStack Compute and as an authorization service for OpenStack Object Storage.

Identity Service Concepts

The Keystone Identity Service has several key concepts which are important to understand:

User

A digital representation of a person, system, or service who uses OpenStack cloud services. Keystone authentication services will validate that incoming request are being made by the user who claims to be making the call. Users have a login and may be assigned tokens to access resources. Users may be directly assigned to a particular tenant and behave as if they are contained in that tenant.

Credentials

Data that belongs to, is owned by, and generally only known by a user that the user can present to prove they are who they are (since nobody else should know that data).

Examples are:

- a matching username and password
- a matching username and API key
- yourself and a driver's license with a picture of you
- a token that was issued to you that nobody else knows of

Authentication

In the context of Keystone, authentication is the act of confirming the identity of a user or the truth of a claim. Keystone will confirm that incoming request are being made by the user who claims to be making the call by validating a set of claims that the user is making. These claims are initially in the form of a set of credentials (username & password, or username and API key). After initial confirmation, Keystone will issue the user a token which the user can then provide to demonstrate that their identity has been authenticated when making subsequent requests.

Token

A token is an arbitrary bit of text that is used to access resources. Each token has a scope which describes which resources are accessible with it. A token may be revoked at anytime and is valid for a finite duration.

While Keystone supports token-based authentication in this release, the intention is for it to support additional protocols in the future.

	The intent is for it to be an integration service foremost, and not a aspire to be a full-fledged identity store and management solution.
Tenant	A container used to group or isolate resources and/or identity objects. Depending on the service operator, a tenant may map to a customer, account, organization, or project.
Service	An OpenStack service, such as Compute (Nova), Object Storage (Swift), or Image Service (Glance). A service provides one or more endpoints through which users can access resources and perform (presumably useful) operations.
Endpoint	An network-accessible address, usually described by URL, where a service may be accessed. If using an extension for templates, you can create an endpoint template, which represents the templates of all the consumable services that are available across the regions.
Role	A personality that a user assumes when performing a specific set of operations. A role includes a set of right and privileges. A user assuming that role inherits those rights and privileges. In Keystone, a token that is issued to a user includes the list of roles that user can assume. Services that are being called by that user determine how they interpret the set of roles a user has and which operations or resources each roles grants access to.

Installing the OpenStack Identity Service

You can install the Identity service from packages or from source.

To install the latest version of the Identity Service (Keystone) from the Github repositories, following the following instructions.

For Debian/Ubuntu, add the Keystone PPA to your sources.lst:

1. $> sudo add-apt-repository ppa:keystone-core/trunk $> sudo apt-get update

2. Install Keystone:

   $> sudo apt-get install keystone

To install the latest version of Keystone from the Launchpad Bazaar repositories, following the following instructions.

1. Grab the source tarball from Github

2. Untar the source tarball:

   $> tar -xzf <FILE>

3. Change into the package directory and build/install:

   $> cd keystone-<RELEASE> $> sudo python setup.py install

To install the latest version of Keystone from the Github repositories, see the following instructions.

These are for Debian/Ubuntu.

Note

If you want to build the Keystone documentation locally, you will also want to install the python-sphinx package.

1. Install Git and build dependencies:

   $> sudo apt-get install git python-eventlet python-routes python-greenlet swift $> sudo apt-get install python-argparse python-sqlalchemy python-wsgiref python-pastedeploy

2. Branch Keystone's trunk branch. (See http://wiki.openstack.org/GerritWorkflow to get the project initially setup):

   $> git checkout master $> git pull origin master

3. Install Keystone:

   $> sudo python setup.py install

Starting the Identity Service

By default, configuration parameters (such as the IP and port binding for each service) are parsed from etc/keystone.conf, so ensure it is up-to-date prior to starting the service.

To start up the Keystone service, enter the following:

```
$ cd ~/keystone/bin && ./keystone
```

In return you should see something like this:

```
Starting the Legacy Authentication component
        Service API listening on 0.0.0.0:5000
        Admin API listening on 0.0.0.0:5001
```

Use this command for starting the auth server only which exposes the Service API:

```
$ ./bin/keystone-auth
```

Use this command for starting the admin server only which exposes the Admin API:

```
$ ./bin/keystone-admin
```

After starting keystone or running keystone-manage a keystone.db sqlite database should be created in the keystone folder.

Configuring the Identity Service

Here are the steps to get started with authentication using Keystone, the project name for the OpenStack Identity Service.

Typically a project that uses Keystone has settings in a configuration file:

- In Compute, the settings are in etc/nova/api-paste.ini, but Keystone also provides an example file in keystone/examples/paste/nova-api-paste.ini. Restart the nova-api service for these settings to be configured.
- In Image Service, the settings are in glance-api.conf and glance-registry.conf configuration files in the examples/paste directory. Restart the glance-api service and also ensure your environment contains OS_AUTH credentials which you can set up with tools/nova_to_os_env.sh provided by the Glance project.
- In Object Storage, the settings are held in /etc/swift/proxy-server.conf in a [filter:keystone] section. Use `swift-init main start` to restart Object Storage with the new configuration. Here's an example /etc/swift/proxy-server.conf:

```
            [DEFAULT]
bind_port = 8888
user = <user>

[pipeline:main]
pipeline = catch_errors cache keystone proxy-server

[app:proxy-server]
use = egg:swift#proxy
account_autocreate = true

[filter:keystone]
use = egg:keystone#tokenauth
auth_protocol = http
auth_host = 127.0.0.1
auth_port = 5001
admin_token = 999888777666
delay_auth_decision = 0
service_protocol = http
service_host = 127.0.0.1
service_port = 8100
service_pass = dTpw

[filter:cache]
use = egg:swift#memcache
set log_name = cache

[filter:catch_errors]
use = egg:swift#catch_errors
```

Dependencies

Once Keystone is installed you need to initialize the database. You can do so with the keystone-manage command line utility. The keystone-manage utility helps with managing and configuring a Keystone installation. You configure the keystone-manage utility itself with a SQL Alchemy connection configuration via a parameter passed to the utility:

–sql_connection=CONN_STRING

Where the CONN_STRING is a proper SQLAlchemy connection string as described in http://www.sqlalchemy.org/docs/05/reference/sqlalchemy/connections.html?highlight=engine#sqlalchemy.create_engine.

One important use of keystone-manage is to setup the database. To do so, run:

```
keystone-manage db_sync
```

Creating Tenants, Users, Roles, Tokens and Endpoints

Sample data entries are available in keystone/bin/sampledata.sh. The following are just examples for a walk-through.

Note

Some reserved roles are defined (and can be modified) through the keystone.conf in the /etc folder.

Add two tenants, and administrative tenant and a tenant named demo. Tenants are equivalent to projects in the previous auth system in Compute. In Object Storage, Tenants are similar to accounts in the swauth system.

```
  bin/keystone-manage  tenant add admin
  bin/keystone-manage  tenant add demo
```

Next add two users to the Identity Service and assign their passwords. The last value in the list is an ID number.

```
  bin/keystone-manage  user add admin p4ssw0rd 1
  bin/keystone-manage  user add demo p455w0rd 2
```

Now you can assign roles, which includes a set of rights and privileges that are double-checked with the token that the user is issued.

```
  bin/keystone-manage  role add Admin
  bin/keystone-manage  role add Member
  bin/keystone-manage  role grant Admin admin
```

Now define the endpointTemplates, which are URLs plus port values that indicate where a service may be accessed. This example shows many services available to Compute including the Image Service, the Object Storage service, as well as Identity itself. Since there is just one zone in this example, it represents all the services across the single region (but could also represent all the regions).

```
  bin/keystone-manage  endpointTemplates add RegionOne swift http://%HOST_IP
%:8080/v1/AUTH_%tenant_id% http://%HOST_IP%:8080/ http://%HOST_IP%:8080/v1/
AUTH_%tenant_id% 1 1
  bin/keystone-manage  endpointTemplates add RegionOne nova_compat http://
%HOST_IP%:8774/v1.0/ http://%HOST_IP%:8774/v1.0  http://%HOST_IP%:8774/v1.0 1
 1
  bin/keystone-manage  endpointTemplates add RegionOne nova http://%HOST_IP
%:8774/v1.1/%tenant_id% http://%HOST_IP%:8774/v1.1/%tenant_id%  http://
%HOST_IP%:8774/v1.1/%tenant_id% 1 1
  bin/keystone-manage  endpointTemplates add RegionOne glance http://%HOST_IP
%:9292/v1.1/%tenant_id% http://%HOST_IP%:9292/v1.1/%tenant_id% http://%HOST_IP
%:9292/v1.1/%tenant_id% 1 1
  bin/keystone-manage  endpointTemplates add RegionOne identity http://
%HOST_IP%:5000/v2.0 http://%HOST_IP%:5001/v2.0 http://%HOST_IP%:5000/v2.0 1 1
```

Now you add a default token for the admin user to get when requesting a token.

```
bin/keystone-manage  token add 999888777666 1 1 2015-02-05T00:00
```

This section adds the tenant endpoints for each user created above (admin with ID 1 and demo with ID 2).

```
  bin/keystone-manage  endpoint add 1 1
  bin/keystone-manage  endpoint add 1 2
  bin/keystone-manage  endpoint add 1 3
  bin/keystone-manage  endpoint add 1 4
  bin/keystone-manage  endpoint add 1 5
  bin/keystone-manage  endpoint add 1 6

  bin/keystone-manage  endpoint add 2 1
  bin/keystone-manage  endpoint add 2 2
  bin/keystone-manage  endpoint add 2 3
  bin/keystone-manage  endpoint add 2 4
  bin/keystone-manage  endpoint add 2 5
  bin/keystone-manage  endpoint add 2 6
```

You can configure Identity and Compute with a single region or multiple regions using zones. You need to add a label for the endpoint for each region. Having a single region doesn't require any work other than adding label.

```
keystone-manage endpointTemplates add SWRegion identity http://%HOST_IP%:5000/
v2.0 http://%HOST_IP%:5001/v2.0 http://%HOST_IP%:5000/v2.0 1 1
```

Curl examples

All examples assume default port usage (5001) and use the example admin account created above.

Admin Initial GET

Retrieves version, full API url, pdf doc link, and wadl link:

```
$> curl http://0.0.0.0:5001
```

or:

```
$> curl http://0.0.0.0:5001/v2.0/
```

Retrieve token:

To retrieve the token and expiration date for a user:

```
$> curl -d '{"passwordCredentials":{"username": "MyAdmin", "password":
 "P@ssw0rd"}}' -H "Content-type: application/json" http://localhost:5001/v2.0/
tokens
```

This will return something like:

```
$> {"auth": {"token": {"expires": "2011-08-10T17:45:22.838440", "id":
 "0eed0ced-4667-4221-a0b2-24c91f242b0b"}}}
```

Note

Save the “id” value as you’ll be using it in the calls below.

To retrieve a list of tenants:

Run:

```
$> curl -H "X-Auth-Token:999888777666" http://localhost:5001/v2.0/tenants
```

This will return something like:

```
$> {"tenants": {"values": [{"enabled": 1, "id": "MyTenant", "description":
 null}], "links": []}}
```

Retrieve a list of users:

Run:

```
$> curl -H "X-Auth-Token:999888777666" http://localhost:5001/v2.0/users
```

This will return something like:

```
$> {"users": {"values": [{"email": null, "enabled": true, "id": "MyAdmin",
 "tenantId": "MyTenant"}], "links": []}}
```

Retrieve information about the token:

Run:

```
$> curl -H "X-Auth-Token:999888777666" http://localhost:5001/v2.0/tokens/
0eed0ced-4667-4221-a0b2-24c91f242b0b
```

This will return something like:

```
$> {"auth": {"token": {"expires": "2011-08-11T04:26:58.145171", "id":
 "0eed0ced-4667-4221-a0b2-24c91f242b0b"}, "user": {"username": "MyAdmin",
 "roleRefs": [{"roleId": "Admin", "id": 1}], "tenantId": "MyTenant"}}}
```

Revoking a token:

Run:

```
$> curl -X DELETE -H "X-Auth-Token:999888777666" http://localhost:5001/tokens/
0eed0ced-4667-4221-a0b2-24c91f242b0b
```

Creating a tenant:

Run:

```
 $> curl -H "X-Auth-Token:999888777666" -H "Content-type: application/
json" -d '{"tenant":{"id":"MyTenant2", "description":"My 2nd Tenant",
 "enabled":true}}'  http://localhost:5001/tenants
```

This will return something like:

```
$> {"tenant": {"enabled": true, "id": "MyTenant2", "description": "My 2nd
 Tenant"}}
```

Verifying the tenant:

Run:

```
$> curl -H "X-Auth-Token:999888777666" http://localhost:5001/v2.0/tenants/
MyTenant2
```

This will return something like:

```
$> {"tenant": {"enabled": 1, "id": "MyTenant2", "description": "My 2nd
 Tenant"}}
```

Updating the tenant:

Run:

```
$> curl -X PUT -H "X-Auth-Token:999888777666" -H "Content-type: application/
json" -d '{"tenant":{"description":"My NEW 2nd Tenant"}}' http://
localhost:5001/v2.0/tenants/MyTenant2
```

This will return something like:

```
$> {"tenant": {"enabled": true, "id": "MyTenant2", "description": "My NEW 2nd
 Tenant"}}
```

Deleting the tenant:

Run:

```
$> curl -X DELETE -H "X-Auth-Token:999888777666" http://localhost:5001/v2.0/
tenants/MyTenant2
```