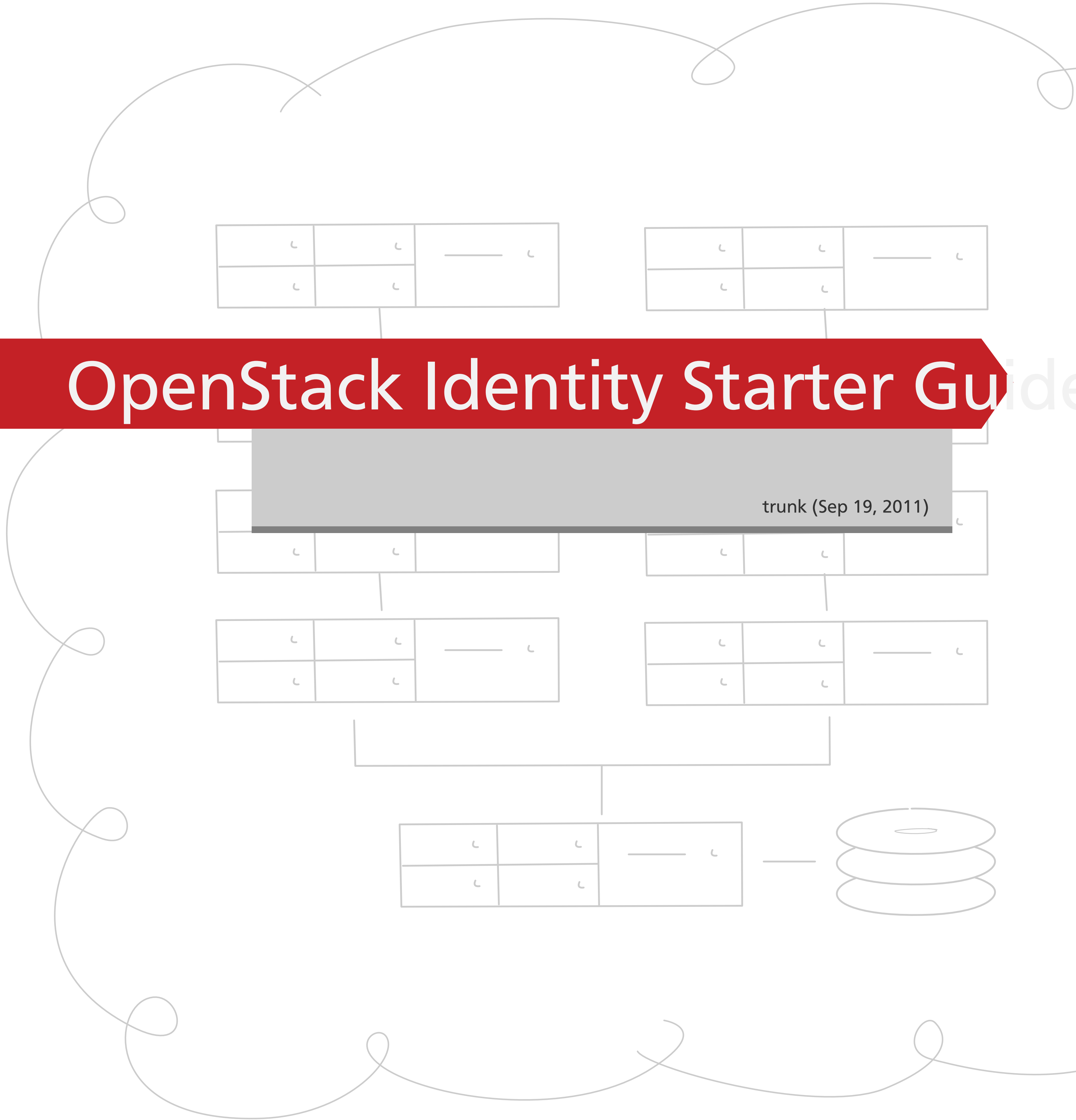
OpenStack Identity Starter Guide

trunk (Sep 19, 2011)



openstack™

docs.openstack.org

OpenStack Identity Starter Guide

trunk (2011-09-19)

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

First, you will need to install keystone, if you haven't done so already. Refer to Installing for more information.

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 your first global admin and tenant admin

Change directory to your Keystone install path.

1. Run the following to create the first tenant:

   ```
   $> bin/keystone-manage tenant add "MyTenant"
   ```

2. Run the following to create the first tenant admin:

   ```
   $> bin/keystone-manage user add MyAdmin P@ssw0rd MyTenant
   ```

   ### Note

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

3. Associate your tenant admin with the Admin role:

   ```
   $> bin/keystone-manage role grant Admin MyAdmin
   ```

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
```

2. Installing Keystone

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

Installing from packages

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

Debian/Ubuntu

1. Add the Keystone PPA to your sources.lst:

   ::

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

2. Install Keystone:

   ::

   ```
   $> sudo apt-get install keystone
   ```

Installing from source tarballs

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
   ```

Installing from a Github Branch

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

Debian/Ubuntu

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
```

..note:

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

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

   ::

   ```
   $> git checkout master $> git pull origin master
   ```

2. Install Keystone:

   ::

   ```
   $> sudo python setup.py install
   ```

3. 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.