



Red Hat OpenStack Platform 12

Integrate with Identity Service

Use Active Directory or Red Hat Identity Management as an external authentication back end

Red Hat OpenStack Platform 12 Integrate with Identity Service

Use Active Directory or Red Hat Identity Management as an external authentication back end

OpenStack Team
rhos-docs@redhat.com

Legal Notice

Copyright © 2018 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Use Active Directory or Red Hat Identity Management as an external authentication back end.

Table of Contents

PREFACE	4
CHAPTER 1. ACTIVE DIRECTORY INTEGRATION	5
1.1. KEY TERMS	5
1.2. ASSUMPTIONS	5
1.3. IMPACT STATEMENT	5
1.3.1. High Availability options	5
1.4. OUTAGE REQUIREMENTS	6
1.5. FIREWALL CONFIGURATION	6
1.6. CONFIGURE ACTIVE DIRECTORY DOMAIN SERVICES	6
1.7. CONFIGURE THE LDAPS CERTIFICATE	8
1.8. CONFIGURE IDENTITY SERVICE	8
1.8.1. Enable command line access to keystone v3	8
1.8.2. Configure the controller	9
1.8.3. Configure Compute to use keystone v3	15
1.8.4. Configure Block Storage to use keystone v3	16
1.8.5. Allow Active Directory group members to access Projects	16
1.8.6. Allow Active Directory users to access Projects	18
1.9. GRANT ACCESS TO THE DOMAIN TAB	20
1.10. CREATING A NEW PROJECT	20
1.11. CHANGES TO THE DASHBOARD LOG IN PROCESS	20
1.12. CHANGES TO THE COMMAND LINE	21
1.13. TEST AD DS INTEGRATION	21
1.14. CONFIGURE HIGH AVAILABILITY	21
1.15. CREATE A RC FILE FOR A NON-ADMIN USER	22
1.16. TROUBLESHOOTING	22
1.16.1. Test LDAP connections	22
1.16.2. Test the Certificate Trust Configuration	23
1.16.3. Test port access	23
CHAPTER 2. IDENTITY MANAGEMENT INTEGRATION	24
2.1. KEY TERMS	24
2.2. ASSUMPTIONS	24
2.3. IMPACT STATEMENT	24
2.3.1. High Availability options	24
2.4. OUTAGE REQUIREMENTS	25
2.5. FIREWALL CONFIGURATION	25
2.6. CONFIGURE THE IDM SERVER	25
2.7. CONFIGURE THE LDAPS CERTIFICATE	26
2.8. CONFIGURE IDENTITY SERVICE	26
2.8.1. Enable command line access to keystone v3	26
2.8.2. Configure the controller	27
2.8.3. Configure Compute to use keystone v3	31
2.8.4. Configure Block Storage to use keystone v3	32
2.8.5. Allow IdM users to access Projects	32
2.9. GRANT ACCESS TO THE DOMAIN TAB	34
2.10. CREATING A NEW PROJECT	34
2.10.1. Changes to the dashboard log in process	34
2.10.2. Changes to the command line	35
2.10.3. Test IdM integration	35
2.11. CONFIGURE FOR HIGH AVAILABILITY	36
2.12. CREATE A RC FILE FOR A NON-ADMIN USER	36

2.13. TROUBLESHOOTING	37
2.13.1. Test LDAP connections	37
2.13.2. Test port access	37
CHAPTER 3. INTEGRATE WITH IDM USING NOVAJOIN	38
3.1. INSTALL AND CONFIGURE NOVAJOIN IN THE UNDERCLOUD	38
3.1.1. Add the undercloud to the CA	38
3.1.2. Add the undercloud to IdM	38
3.2. INSTALL AND CONFIGURE NOVAJOIN IN THE OVERCLOUD	39
3.2.1. Configure overcloud DNS	39
3.2.2. Configure overcloud to use novajoin	40
3.3. VALIDATE A NODE IN IDM	41
CHAPTER 4. USING DOMAIN-SPECIFIC LDAP BACKENDS WITH DIRECTOR	42
4.1. SETTING THE CONFIGURATION OPTIONS	42
4.2. CONFIGURE THE DIRECTOR DEPLOYMENT	42

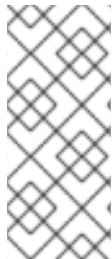
PREFACE

Identity Service (codename *keystone*) provides authentication and authorization for Red Hat OpenStack Platform 12.

This guide describes how to integrate Identity Service with Microsoft Active Directory Domain Service (AD DS), Red Hat Identity Management (IdM), and LDAP.

CHAPTER 1. ACTIVE DIRECTORY INTEGRATION

This chapter describes how to integrate Identity Service (keystone) with Active Directory Domain Services. In this use case, Identity Service authenticates certain Active Directory Domain Services (AD DS) users, while retaining authorization settings and critical service accounts in the Identity Service database. As a result, Identity Service has read-only access to AD DS for user account authentication, while retaining management over the privileges assigned to authenticated accounts.



NOTE

If you are using Red Hat OpenStack Platform director, then you will need to be aware that some of configuration files referenced below are managed by Puppet. Consequently, any custom configuration you add might be overwritten whenever you run the **openstack overcloud deploy** process. To apply these settings to director-based deployments, see [Chapter 4, Using domain-specific LDAP backends with director](#).

1.1. KEY TERMS

- *Authentication* - The process of using a password to verify that the user is who they claim to be.
- *Authorization* - Validating that authenticated users have proper permissions to the resources they are attempting to access.
- *Domain* - This term is not the same as an AD DS domain, and instead refers to the additional namespaces that are configured in Identity Service for partitioning users, groups, and projects. These separate domains can be configured to authenticate users in different LDAP or AD DS environments.

1.2. ASSUMPTIONS

This example deployment makes the following assumptions:

- Active Directory Domain Services is configured and operational.
- Red Hat OpenStack Platform is configured and operational.
- DNS name resolution is fully functional and all hosts are registered appropriately.
- AD DS authentication traffic is encrypted with LDAPS, using port 636.

1.3. IMPACT STATEMENT

These steps allow AD DS users to authenticate to OpenStack and access resources. OpenStack service accounts (such as keystone and glance), and authorization management (permissions, roles, projects) will remain in the Identity Service database. Permissions and roles are assigned to the AD DS accounts using Identity Service management tools.

1.3.1. High Availability options

This configuration creates a dependency on the availability of a single Active Directory Domain Controller; Project users will be affected if Identity Service is unable to authenticate to the AD Domain Controller. A number of options are available to manage this risk; for example, you might configure

Identity Service to query a DNS alias or a load balancing appliance, rather than an individual AD Domain Controller. You can also configure keystone to query a different Domain Controller, should one become unavailable. See [Section 1.14, “Configure high availability”](#) for more information.

1.4. OUTAGE REQUIREMENTS

- The Identity Service will need to be restarted to add the AD DS back end.
- The Compute services on all nodes will need to be restarted in order to switch over to keystone v3.
- Users will be unable to access the dashboard until their accounts have been created in AD DS. To reduce downtime, consider pre-staging the AD DS accounts well in advance of this change.

1.5. FIREWALL CONFIGURATION

If firewalls are filtering traffic between AD DS and OpenStack, you will need to allow access through the following port:

Source	Destination	Type	Port
OpenStack Controller Node	Active Directory Domain Controller	LDAPS	TCP 636

1.6. CONFIGURE ACTIVE DIRECTORY DOMAIN SERVICES

This section describes the tasks that Active Directory administrators will need to complete:

Table 1.1. Configuration steps

Task	Details
Create a service account.	This can be named according to your naming convention for service accounts, for example: svc-ldap . This can be a regular domain user account. Administrator privileges are not required.
Create a user group.	If a user needs access to OpenStack, they must be a member of this group. This can be named according to your naming convention for user groups, for example: grp-openstack . Members of this group can be granted access to <i>Projects</i> in the dashboard, if they are also members of the Project groups.
Create the Project groups.	Each OpenStack Project will require a corresponding AD group. For example, grp-openstack-demo and grp-openstack-admin .
Configure the service account.	The service account svc-ldap must be a member of the grp-openstack group.

Export the LDAPS public key.	Export the public key (not the private key) in the following format: DER-encoded x509 .cer file.
Send the key to the OpenStack administrators.	The OpenStack administrators will use this key to encrypt LDAPS communications between OpenStack and Active Directory.
Retrieve the NetBIOS name of your AD DS domain.	The OpenStack administrators will use this name for the Keystone domain, allowing consistent domain naming between the environments.

For example, the procedure below shows the PowerShell commands that would be run on the Active Directory Domain Controller:

1. Create the LDAP lookup account. This account is used by Identity Service to query the AD DS LDAP service:

```
PS C:\> New-ADUser -SamAccountName svc-ldap -Name "svc-ldap" -
GivenName LDAP -Surname Lookups -UserPrincipalName svc-
ldap@lab.local -Enabled $false -PasswordNeverExpires $true -Path
'OU=labUsers,DC=lab,DC=local'
```

2. Set a password for this account, and then enable it. You will be prompted to specify a password that complies with your AD domain's complexity requirements:

```
PS C:\> Set-ADAccountPassword svc-ldap -PassThru | Enable-ADAccount
```

3. Create a group for OpenStack users, called **grp-openstack**.

```
PS C:\> NEW-ADGroup -name "grp-openstack" -groupscope Global -path
"OU=labUsers,DC=lab,DC=local"
```

4. Create the Project groups:

```
PS C:\> NEW-ADGroup -name "grp-openstack-demo" -groupscope Global -
path "OU=labUsers,DC=lab,DC=local"
PS C:\> NEW-ADGroup -name "grp-openstack-admin" -groupscope Global -
path "OU=labUsers,DC=lab,DC=local"
```

5. Add the **svc-ldap** user to the **grp-openstack** group:

```
PS C:\> ADD-ADGroupMember "grp-openstack" -members "svc-ldap"
```

6. From an AD Domain Controller, use a **Certificates MMC** to export your LDAPS certificate's public key (not the private key) as a DER-encoded **x509 .cer** file. Send this file to the OpenStack administrators.

7. Retrieve the NetBIOS name of your AD DS domain.

```
PS C:\> Get-ADDomain | select NetBIOSName
NetBIOSName
```

LAB

Send this value to the OpenStack administrators.

1.7. CONFIGURE THE LDAPS CERTIFICATE

Keystone uses LDAPS queries to validate user accounts. To encrypt this traffic, keystone uses the certificate file defined by `keystone.conf`. This procedure converts the public key received from Active Directory into the `.crt` format, and copies to a location where keystone will be able to reference it.

1. Copy the LDAPS public key to the node running OpenStack Identity (keystone), and convert the `.cer` to `.crt`. This example uses a source certificate file named `addc.lab.local.cer`:

```
# openssl x509 -outform der -in addc.lab.local.pem -out
addc.lab.local.crt
# cp addc.lab.local.crt /etc/ssl/certs/
```

NOTE

Optionally, if you need to run diagnostic commands, such as `ldapsearch`, you will also need to add the certificate to the RHEL certificate store:

1. Convert the `.cer` to `.pem`. This example uses a source certificate file named `addc.lab.local.cer`:

```
# openssl x509 -inform der -in addc.lab.local.cer -out
addc.lab.local.pem
```

2. Install the `.pem` on your OpenStack controller. For example, in Red Hat Enterprise Linux:

```
# cp addc.lab.local.pem /etc/pki/ca-trust/source/anchors/
# update-ca-trust
```

1.8. CONFIGURE IDENTITY SERVICE

These steps prepare Identity Service (keystone) for integration with AD DS.

NOTE

If you are using Red Hat OpenStack Platform director, then you will need to be aware that some of configuration files referenced below are managed by Puppet. Consequently, any custom configuration you add might be overwritten whenever you run the `openstack overcloud deploy` process. To apply these settings to director-based deployments, see https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/12/html/integrate_with_identity_service/sec-director-ldap

1.8.1. Enable command line access to keystone v3

To manage Identity Service domains from the command line, you need to enable access to keystone v3.

Perform this procedure from the controller running the keystone service.

1. Create a copy of the existing environment variable file. In a director-based deployment, it will be called **overcloudrc**:

```
$ cp overcloudrc overcloudrc-v3
```

2. Edit the new **overcloudrc-v3** file:

- Change **OS_AUTH_URL** from *v2.0* to *v3*. For example:

```
export OS_AUTH_URL=https://controllerIP:5000/v3/
```

- Add the following entries to the bottom of **overcloudrc-v3**:

```
export OS_IDENTITY_API_VERSION=3
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
```

3. Enable these options for your current command line session by sourcing the file:

```
$ source overcloudrc-v3
```

1.8.2. Configure the controller

NOTE

If you intend to update any configuration files, you need to be aware that certain OpenStack services now run within containers; this applies to keystone, nova, and cinder, among others. As a result, there are certain administration practices to consider:

- Do not update any configuration file you might find on the physical node's host operating system, for example, **/etc/cinder/cinder.conf**. This is because the containerized service does not reference this file.
- Do not update the configuration file running within the container. This is because any changes are lost once you restart the container. Instead, if you need to add any changes to containerized services, you will need to update the configuration file that is used to generate the container. These are stored within **/var/lib/config-data/puppet-generated/**

For example:

- keystone: **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf**
- cinder: **/var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf**
- nova: **/var/lib/config-data/puppet-generated/nova/etc/nova/nova.conf**

Any changes will then be applied once you restart the container. For example:
sudo docker exec -it keystone pkill -HUP -f keystone

Perform this procedure from the controller running the keystone service. If running a HA environment with multiple controllers, then these steps must be performed on each controller:

1. Configure SELinux:

```
# setsebool -P authlogin_nsswitch_use_ldap=on
```

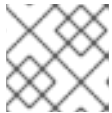
The output might include messages similar to this. They can be ignored:

```
Full path required for exclude: net:[4026532245].
```

2. Create the **domains** directory:

```
# mkdir /var/lib/config-data/puppet-generated/keystone/etc/keystone/domains/
# chown 42425:42425 /var/lib/config-data/puppet-generated/keystone/etc/keystone/domains/
```

3. Configure keystone to use multiple back ends:



NOTE

You might need to install **crudini** using **yum install crudini**.

```
# crudini --set /var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf identity
domain_specific_drivers_enabled true
# crudini --set /var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf identity
domain_config_dir /etc/keystone/domains
# crudini --set /var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf assignment driver sql
```



NOTE

If you are using Red Hat OpenStack Platform director, then you will need to be aware that **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf** is managed by Puppet. Consequently, any custom configuration you add might be overwritten whenever you run the **openstack overcloud deploy** process. As a result, you might need to re-add this configuration manually each time. For director-based deployments, see https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/12/html/integrate_with_identity_service/sec-director-ldap

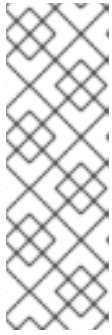
4. Enable multiple domains in dashboard. Add these lines to **/etc/openstack-dashboard/local_settings**:

```
OPENSTACK_API_VERSIONS = {
    "identity": 3
}
```

```

OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = 'Default'

```



NOTE

If you are using Red Hat OpenStack Platform director, then you will need to be aware that `/etc/openstack-dashboard/local_settings` is managed by Puppet. Consequently, any custom configuration you add might be overwritten whenever you run the **openstack overcloud deploy** process. As a result, you might need to re-add this configuration manually each time. It is expected that a future release of director will include the Puppet parameters that will allow you to re-apply these settings automatically using a post-deployment script.

Restart `httpd` to apply the settings:

```
# systemctl restart httpd
```

5. Configure an additional back end:

In this example, **LAB** is the NetBIOS name to use as the Identity Service domain.

a. Create the keystone domain for AD DS integration.

Use the NetBIOS name value retrieved previously as the domain name. This approach allows you to present a consistent domain name to users during the login process. For example, if the NetBIOS name is **LAB**:

```
# openstack domain create LAB
```



NOTE

If this command is not available, check that you have enabled keystone v3 for your command line session by running `# source overcloudrc-v3`.

b. Create the configuration file:

To add the AD DS back end, enter the LDAP settings in a new file called `/var/lib/config-data/puppet-`

generated/keystone/etc/keystone/domains/keystone.LAB.conf (where **LAB** is the NetBIOS name retrieved previously). You will need to edit the sample settings below to suit your AD DS deployment:

```

[ldap]
url = ldaps://addc.lab.local:636
user = CN=svc-ldap,OU=labUsers,DC=lab,DC=local
password = RedactedComplexPassword
suffix = DC=lab,DC=local
user_tree_dn = OU=labUsers,DC=lab,DC=local
user_objectclass = person
user_filter = (|(memberOf=cn=grp-
openstack,OU=labUsers,DC=lab,DC=local)(memberOf=cn=grp-openstack-
admin,OU=labUsers,DC=lab,DC=local)(memberOf=memberOf=cn=grp-
openstack-demo,OU=labUsers,DC=lab,DC=local))
user_id_attribute = sAMAccountName
user_name_attribute = sAMAccountName

```

```

user_mail_attribute      = mail
user_pass_attribute     =
user_enabled_attribute  = userAccountControl
user_enabled_mask       = 2
user_enabled_default    = 512
user_attribute_ignore   = password,tenant_id,tenants
user_allow_create       = False
user_allow_update       = False
user_allow_delete       = False
group_objectclass       = group
group_tree_dn           = OU=labUsers,DC=lab,DC=local
group_filter             = (CN=grp-openstack*)
group_id_attribute      = cn
group_name_attribute    = name
group_allow_create      = False
group_allow_update      = False
group_allow_delete      = False
use_tls                 = False
tls_cacertfile          =
/etc/ssl/certs/addc.lab.local.crt
query_scope              = sub
chase_referrals          = false

[identity]
driver = ldap

```

Explanation of each setting:

Setting	Description
url	The AD Domain Controller to use for authentication. Uses LDAPS port 636 .
user	The <i>Distinguished Name</i> of an AD account to use for LDAP queries. For example, you can locate the <i>Distinguished Name</i> value of the <i>svc-ldap</i> account in AD using Get-ADuser svc-ldap select DistinguishedName
password	The plaintext password of the AD account used above.
suffix	The <i>Distinguished Name</i> of your AD domain. You can locate this value using Get-ADDomain select DistinguishedName
user_tree_dn	The <i>Organizational Unit</i> (OU) that contains the OpenStack accounts.
user_objectclass	Defines the type of LDAP user. For AD, use the person type.

Setting	Description
<code>user_filter</code>	Filters the users presented to Identity Service. As a result, only members of the grp-openstack group can have permissions defined in Identity Service. This value requires the full <i>Distinguished Name</i> of the group: Get-ADGroup grp-openstack select DistinguishedName
<code>user_id_attribute</code>	Maps the AD value to use for user IDs.
<code>user_name_attribute</code>	Maps the AD value to use for <i>names</i> .
<code>user_mail_attribute</code>	Maps the AD value to use for user email addresses.
<code>user_pass_attribute</code>	Leave this value blank.
<code>user_enabled_attribute</code>	The AD setting that validates whether the account is enabled.
<code>user_enabled_mask</code>	Defines the value to check to determine whether an account is enabled. Used when booleans are not returned.
<code>user_enabled_default</code>	The AD value that indicates that an account is enabled.
<code>user_attribute_ignore</code>	Defines user attributes that Identity Service should disregard.
<code>user_allow_create</code>	Set this value to False , as keystone only requires read-only access.
<code>user_allow_update</code>	Set this value to False , as keystone only requires read-only access.
<code>user_allow_delete</code>	Set this value to False , as keystone only requires read-only access.
<code>group_objectclass</code>	Maps the AD value to use for <i>groups</i> .
<code>group_tree_dn</code>	The <i>Organizational Unit</i> (OU) that contains the user groups.
<code>group_filter</code>	Filters the groups presented to Identity Service.

Setting	Description
<code>group_id_attribute</code>	Maps the AD value to use for group IDs.
<code>group_name_attribute</code>	Maps the AD value to use for group names.
<code>group_allow_create</code>	Set this value to False , as keystone only requires read-only access.
<code>group_allow_update</code>	Set this value to False , as keystone only requires read-only access.
<code>group_allow_delete</code>	Set this value to False , as keystone only requires read-only access.
<code>use_tls</code>	Defines whether TLS is to be used. This needs to be disabled if you are encrypting with LDAPS rather than STARTTLS.
<code>tls_cacertfile</code>	Specifies the path to the <code>.crt</code> certificate file.
<code>query_scope</code>	Configures Identity Service to also search within nested child OUs, when locating users that are members of the grp-openstack group.
<code>chase_referrals</code>	Set to false , this setting prevents python-ldap from chasing all referrals with anonymous access.

6. Change ownership of the configuration file to the keystone user:

```
# chown 42425:42425 /var/lib/config-data/puppet-generated/keystone/etc/keystone/domains/keystone.LAB.conf
```

7. Restart the keystone service to apply the changes:

```
# sudo docker exec -it keystone pkill -HUP -f keystone
```

8. Grant the **admin** user access to the domain:



NOTE

This does not grant the OpenStack admin account any permissions on the actual AD DS domain. In this case, the term *domain* refers to OpenStack's usage of the keystone domain.

- a. Get the **ID** of the **LAB** domain:

```
# openstack domain show LAB
```

```
+-----+-----+
| Field   | Value                                     |
+-----+-----+
| enabled | True                                       |
| id      | 6800b0496429431ab1c4efbb3fe810d4      |
| name    | LAB                                       |
+-----+-----+
```

- b. Get the **ID** value of the *admin* user:

```
# openstack user list --domain default | grep admin
| 3d75388d351846c6a880e53b2508172a | admin      |
```

- c. Get the **ID** value of the *admin* role:

```
# openstack role list
+-----+-----+-----+
| ID                                     | Name           |
+-----+-----+-----+
| 544d48aaffde48f1b3c31a52c35f01f9    | SwiftOperator |
| 6d005d783bf0436e882c55c62457d33d    | ResellerAdmin |
| 785c70b150ee4c778fe4de088070b4cf    | admin         |
| 9fe2ff9ee4384b1894a90878d3e92bab    | _member_     |
+-----+-----+-----+
```

- d. Use the returned domain and admin IDs to construct the command that adds the **admin** user to the **admin** role of the keystone **LAB** domain:

```
# openstack role add --domain 6800b0496429431ab1c4efbb3fe810d4 --
user 3d75388d351846c6a880e53b2508172a
785c70b150ee4c778fe4de088070b4cf
```

- e. View the list of users in the AD DS domain by adding the NetBIOS name to the command:



NOTE

It might take some time for the LDAP to become queryable after a reboot or service restart.

```
# openstack user list --domain LAB
```

- f. View the service accounts in the local Identity Service database:

```
# openstack user list --domain default
```

1.8.3. Configure Compute to use keystone v3

Compute uses keystone v2.0 by default, and so needs to be configured to use keystone v3 in order to use multi-domain capabilities.

1. On each Compute node, and the controller, adjust the **keystone_authtoken** value:

- a. On the Compute nodes:

```
# crudini --set /var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf keystone_authtoken
auth_version v3
```

- b. On the controller:

```
# crudini --set /var/lib/config-data/puppet-generated/nova/etc/nova/nova.conf keystone_authtoken auth_version
v3
```

2. Restart these services on the controller to apply the changes:

```
# systemctl restart openstack-nova-api.service openstack-nova-cert.service openstack-nova-conductor.service openstack-nova-consoleauth.service openstack-nova-novncproxy.service openstack-nova-scheduler.service

# sudo docker exec -it keystone pkill -HUP -f keystone
```

3. Restart these services on each Compute node to apply the changes:

```
# systemctl restart openstack-nova-compute.service
```

1.8.4. Configure Block Storage to use keystone v3

You must also configure Block Storage (cinder) to authenticate to keystone v3.

1. In `/etc/cinder/cinder.conf`:

```
[keystone_authtoken]
auth_uri = https://controllerIP:5000/v3
auth_version = v3
```

- **auth_uri** - replace **controllerIP** with the IP address of the controller. If your deployment has more than one controller, you should use the keystone endpoint VIP instead of the controller IP.

2. Restart **cinder-api** on all controllers:

```
# systemctl restart openstack-cinder-api
```

3. Restart **cinder-scheduler** on all controllers:

```
# systemctl restart openstack-cinder-scheduler
```

4. Restart **cinder-volume** (on one controller only):

```
# pcs resource restart openstack-cinder-volume
```

1.8.5. Allow Active Directory group members to access Projects

To allow authenticated users access to OpenStack resources, the recommended method is to authorize certain Active Directory groups to grant access to Projects. This saves the OpenStack administrators from having to allocate each user to a role in a Project. Instead, the Active Directory groups are granted roles in Projects. As a result, Active Directory users that are members of these Active Directory groups will be able to access pre-determined Projects.



NOTE

If you would prefer to manually manage the authorization of individual Active Directory users, see the following section: *Allow individual Active Directory users to access Projects*

This section presumes that the Active Directory administrator has already completed these steps:

- Create a group named **grp-openstack-admin** in Active Directory.
- Create a group named **grp-openstack-demo** in Active Directory.
- Add your Active Directory users to one of the above groups, as needed.
- Add your Active Directory users to the **grp-openstack** group.

These steps assign a role to an AD group. Group members will then have permission to access OpenStack resources.

1. Retrieve a list of AD groups:

```
# openstack group list --domain LAB
+-----+
---+-----+
| ID
| Name
+-----+
---+-----+
| 185277be62ae17e498a69f98a59b66934fb1d6b7f745f14f5f68953a665b8851 |
grp-openstack
| a8d17f19f464c4548c18b97e4aa331820f9d3be52654aa8094e698a9182cbb88 |
grp-openstack-admin
| d971bb3bd5e64a454cbd0cc7af4c0773e78d61b5f81321809f8323216938cae8 |
grp-openstack-demo
+-----+
---+-----+
```

2. Retrieve a list of roles:

```
# openstack role list
+-----+-----+
| ID
| Name
+-----+-----+
| 0969957bce5e4f678ca6cef00e1abf8a | ResellerAdmin
| 1fcb3c9b50aa46ee8196aaaec2b76b7 | admin
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_
| d3570730eb4b4780a7fed97eba197e1b | SwiftOperator
+-----+-----+
```

- Grant the Active Directory groups access to Projects by adding them to one or more of these roles. For example, if you want users in the **grp-openstack-demo** group to be general users of the **demo** project, you must add the group to the **_member_** role:

```
# openstack role add --project demo --group
d971bb3bd5e64a454cbd0cc7af4c0773e78d61b5f81321809f8323216938cae8
_member_
```

As a result, members of **grp-openstack-demo** are able to log in to the dashboard by entering their AD DS username and password, when also entering **LAB** in the Domain field:



NOTE

If users receive the error **Error: Unable to retrieve container list.**, and expect to be able to manage containers, then they must be added to the **SwiftOperator** role.

1.8.6. Allow Active Directory users to access Projects

AD DS users that are members of the **grp-openstack** AD group can be granted permission to log in to a *Project* in the dashboard:

- Retrieve a list of AD users:

```
# openstack user list --domain LAB
+-----+
+-----+
| ID
| Name          |
+-----+
+-----+
| 1f24ec1f11aeb90520079c29f70afa060d22e2ce92b2eba7784c841ac418091e |
user1          |
| 12c062faddc5f8b065434d9ff6fce03eb9259537c93b411224588686e9a38bf1 |
user2          |
| afaf48031eb54c3e44e4cb0353f5b612084033ff70f63c22873d181fdae2e73c |
```

```

user3          |
| e47fc21dcf0d9716d2663766023e2d8dc15a6d9b01453854a898cabb2396826e |
user4          |
+-----+
---+-----+

```

- Retrieve a list of roles:

```

# openstack role list
+-----+-----+
| ID                               | Name           |
+-----+-----+
| 544d48aaffde48f1b3c31a52c35f01f9 | SwiftOperator |
| 6d005d783bf0436e882c55c62457d33d | ResellerAdmin |
| 785c70b150ee4c778fe4de088070b4cf | admin          |
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_      |
+-----+-----+

```

- Grant users access to Projects by adding them to one or more of these roles. For example, if you want **user1** to be a general user of the **demo** project, you add them to the **member** role:

```

# openstack role add --project demo --user
1f24ec1f11aeb90520079c29f70afa060d22e2ce92b2eba7784c841ac418091e
_member_

```

Or, if you want **user1** to be an administrative user of the **demo** project, you add them to the **admin** role:

```

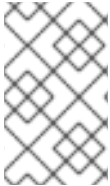
# openstack role add --project demo --user
1f24ec1f11aeb90520079c29f70afa060d22e2ce92b2eba7784c841ac418091e
admin

```

As a result, **user1** is able to log in to the dashboard by entering their AD DS username and password, when also entering **LAB** in the **Domain** field:

The screenshot shows a login interface with a dark background. It contains three input fields and a button:

- Domain:** A text input field containing the text "LAB".
- User Name:** A text input field containing the text "user1".
- Password:** A password input field with a white background and a series of dots representing the masked password.
- Connect:** A blue button with the text "Connect" in white.

**NOTE**

If users receive the error **Error: Unable to retrieve container list.**, and expect to be able to manage containers, then they must be added to the **SwiftOperator** role.

1.9. GRANT ACCESS TO THE DOMAIN TAB

To allow the **admin** user to see the **Domain** tab, you will need to assign it the **admin** role in the **default** domain:

1. Find the **admin** user's UUID:

```
$ openstack user list | grep admin
| a6a8adb6356f4a879f079485dad1321b | admin      |
```

2. Add the **admin** role in the **default** domain to the **admin** user:

```
$ openstack role add --domain default --user
a6a8adb6356f4a879f079485dad1321b admin
```

As a result, the **admin** user can now see the **Domain** tab.

1.10. CREATING A NEW PROJECT

After you have completed these integration steps, when you create a new project you will need to decide whether to create it in the **Default** domain, or in the keystone domain you've just created. This decision can be reached by considering your workflow, and how you administer user accounts. The *Default* domain can be thought of as an internal domain, used to manage service accounts and the *admin* project. For separation purposes, you might want to keep your AD-backed users in a separate keystone domain.

1.11. CHANGES TO THE DASHBOARD LOG IN PROCESS

Configuring multiple domains in Identity Service enables a new *Domain* field in the dashboard login page.

Users are expected to enter the domain that matches their login credentials. This field must be manually filled with one of the domains present in keystone. Use the *openstack* command to list the available entries.

In this example, AD DS accounts will need to specify the **LAB** domain. The built-in keystone accounts, such as *admin*, must specify **Default** as their domain:

```
# openstack domain list
+-----+-----+-----+-----+
| ID                | Name    | Enabled | Description |
+-----+-----+-----+-----+
| 6800b0496429431ab1c4efbb3fe810d4 | LAB     | True    |             |
| default           | Default | True    | Owns users and
```



```
tenants (i.e. projects) available on Identity API v2. |
```

```
+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+
```

1.12. CHANGES TO THE COMMAND LINE

For certain commands, you might need to specify the applicable domain. For example, appending `--domain LAB` in this command returns users in the LAB domain (that are members of the `grp-openstack` group):

```
# openstack user list --domain LAB
```

Appending `--domain Default` returns the built-in keystone accounts:

```
# openstack user list --domain Default
```

1.13. TEST AD DS INTEGRATION

This procedure validates AD DS integration by testing user access to dashboard features:

1. Create a test user in AD, and add the user to the `grp-openstack` AD DS group.
2. Add the user to the `_member_` role of the `demo` tenant.
3. Log in to the dashboard using the credentials of the AD test user.
4. Click on each of the tabs to confirm that they are presented successfully without error messages.
5. Use the dashboard to build a test instance.



NOTE

If you experience issues with these steps, perform steps 3-5 with the built-in `admin` account. If successful, this demonstrates that OpenStack is still working as expected, and that an issue exists somewhere within the AD ↔ Identity integration settings. See [Section 1.16, “Troubleshooting”](#).

1.14. CONFIGURE HIGH AVAILABILITY

With keystone v3 enabled, you can make this configuration highly available by listing multiple AD Domain Controllers in the configuration file for that domain.

1. Add a second server to the `url` entry. For example, updating the `url` setting in the `keystone.LAB.conf` file will have keystone send all query traffic to the first Domain Controller in the list, `addc.lab.local`:

```
url = ldaps://addc.lab.local,ldaps://addc2.lab.local
```

If a query to `addc.lab.local` fails due to it being unavailable, Identity Service will attempt to query the next server in the list: `addc2.lab.local`. Note that this configuration does not perform queries in a round-robin fashion, so cannot be considered a load-balancing solution.

- Set the network timeout in `/etc/openldap/ldap.conf`:

```
NETWORK_TIMEOUT 2
```

In addition, if you have firewalls configured between the controller and the domain controllers, then you should not configure the domain controllers to silently drop packets from the controller. This will allow **python-keystoneclient** to properly detect outages and redirect the request to the next domain controller in the list.



NOTE

There might be connection delays while queries are being redirected to the second LDAP server in the list. This is because the connection to the first server must first time out before the second is attempted.

1.15. CREATE A RC FILE FOR A NON-ADMIN USER

You might need to create a RC file for a non-admin user. For example:

```
$ cat overcloudrc-v3-user1
# Clear any old environment that may conflict.
for key in $( set | awk '{FS="="} /^OS_/ {print $1}' ); do unset $key ;
done
export OS_USERNAME=user1
export NOVA_VERSION=1.1
export OS_PROJECT_NAME=demo
export OS_PASSWORD=RedactedComplexPassword
export OS_NO_CACHE=True
export COMPUTE_API_VERSION=1.1
export no_proxy=,10.0.0.5,192.168.2.11
export OS_CLOUDNAME=overcloud
export OS_AUTH_URL=https://10.0.0.5:5000/v3
export OS_AUTH_TYPE=password
export PYTHONWARNINGS="ignore:Certificate has no, ignore:A true
SSLContext object is not available"
export OS_IDENTITY_API_VERSION=3
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=LAB
```

1.16. TROUBLESHOOTING

1.16.1. Test LDAP connections



NOTE

This command expects to find the necessary certificate in your host operating system. See the *Configure the LDAPS certificate* section for more information.

Use **ldapsearch** to remotely perform test queries against the Active Directory Domain Controller. A successful result here indicates that network connectivity is working, and the AD DS services are up. In this example, a test query is performed against the server **adcc.lab.local** on port **636**:

■

```
# ldapsearch -Z -x -H ldaps://addc.lab.local:636 -D "svc-ldap@lab.local" -
W -b "OU=labUsers,DC=lab,DC=local" -s sub "(cn=*)" cn
```



NOTE

ldapsearch is a part of the **openldap-clients** package. You can install this using `# yum install openldap-clients`

1.16.2. Test the Certificate Trust Configuration

If you receive the error **Peer's Certificate issuer is not recognized.** while testing with *ldapsearch*, confirm that your **TLS_CACERTDIR** path is correctly set. For example:

- `/etc/openldap/ldap.conf`

```
TLS_CACERTDIR /etc/openldap/certs
```



NOTE

As a temporary workaround, you may want to consider disabling certificate validation.

This setting must not be permanently configured:

- `/etc/openldap/ldap.conf`

```
TLS_REQCERT allow
```

If the **ldapsearch** query works after setting this value, you might need to review whether your certificate trusts are correctly configured.

1.16.3. Test port access

Use **nc** to check that LDAPS port **636** is remotely accessible. In this example, a probe is performed against the server **addc.lab.local**. Press `ctrl-c` to exit the prompt.

```
# nc -v addc.lab.local 636
Ncat: Version 6.40 ( http://nmap.org/ncat )
Ncat: Connected to 192.168.200.10:636.
^C
```

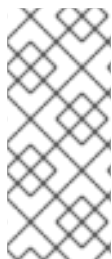
Failure to establish a connection could indicate a firewall configuration issue.

CHAPTER 2. IDENTITY MANAGEMENT INTEGRATION

This chapter describes how to integrate Identity Service (keystone) with Red Hat Identity Management.

In this use case, Identity Service authenticates certain Red Hat Identity Management (IdM) users, while retaining authorization settings and critical service accounts in the Identity Service database.

As a result, Identity Service has read-only access to IdM for user account authentication, while retaining management over the privileges assigned to authenticated accounts.



NOTE

If you are using Red Hat OpenStack Platform director, then you will need to be aware that some of configuration files referenced below are managed by Puppet. Consequently, any custom configuration you add might be overwritten whenever you run the **openstack overcloud deploy** process. To apply these settings to director-based deployments, see [Chapter 4, Using domain-specific LDAP backends with director](#).



NOTE

For additional integration options using *novajoin*, see [Chapter 3, Integrate with IdM using novajoin](#).

2.1. KEY TERMS

- **Authentication** - The process of using a password to verify that the user is who they claim to be.
- **Authorization** - Validating that authenticated users have proper permissions to the systems they're attempting to access.
- **Domain** - Refers to the additional back ends configured in Identity Service. For example, Identity Service can be configured to authenticate users from external IdM environments. The resulting collection of users can be thought of as a *domain*.

2.2. ASSUMPTIONS

This example deployment makes the following assumptions:

- Red Hat Identity Management is configured and operational.
- Red Hat OpenStack Platform is configured and operational.
- DNS name resolution is fully functional and all hosts are registered appropriately.

2.3. IMPACT STATEMENT

These steps allow IdM users to authenticate to OpenStack and access resources. OpenStack service accounts (such as keystone and glance), and authorization management (permissions and roles) will remain in the Identity Service database. Permissions and roles are assigned to the IdM accounts using Identity Service management tools.

2.3.1. High Availability options

This configuration creates a dependency on the availability of a single IdM server: Project users will be

affected if Identity Service is unable to authenticate to the IdM Server. There are a number of options available to manage this risk, for example: you might configure keystone to query a DNS alias or a load balancing appliance, rather than an individual IdM server. You can also configure keystone to query a different IdM server, should one become unavailable. See [Section 2.11, “Configure for high availability”](#) for more information.

2.4. OUTAGE REQUIREMENTS

- The Identity Service will need to be restarted in order to add the IdM back end.
- The Compute services on all nodes will need to be restarted in order to switch over to *keystone v3*.
- Users will be unable to access the dashboard until their accounts have been created in IdM. To reduce downtime, consider pre-staging the IdM accounts well in advance of this change.

2.5. FIREWALL CONFIGURATION

If firewalls are filtering traffic between IdM and OpenStack, you will need to allow access through the following port:

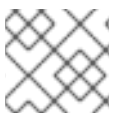
Source	Destination	Type	Port
OpenStack Controller Node	Red Hat Identity Management	LDAPS	TCP 636

2.6. CONFIGURE THE IDM SERVER

Run these commands on the IdM server:

1. Create the LDAP lookup account. This account is used by Identity Service to query the IdM LDAP service:

```
# kinit admin
# ipa user-add
First name: OpenStack
Last name: LDAP
User [administrator]: svc-ldap
```



NOTE

Review the password expiration settings of this account, once created.

2. Create a group for OpenStack users, called *grp-openstack*. Only members of this group can have permissions assigned in OpenStack Identity.

```
# ipa group-add --desc="OpenStack Users" grp-openstack
```

3. Set the *svc-ldap* account password, and add it to the *grp-openstack* group:

```
# ipa passwd svc-ldap
# ipa group-add-member --users=svc-ldap grp-openstack
```

2.7. CONFIGURE THE LDAPS CERTIFICATE

1. In your IdM environment, locate the LDAPS certificate. This file can be located using `/etc/openldap/ldap.conf`:

```
TLS_CACERT /etc/ipa/ca.crt
```

2. Copy the file to the node running OpenStack Identity (keystone). For example, this command uses `scp` to copy `ca.crt` to the controller node named `node.lab.local`:

```
scp /etc/ipa/ca.crt root@node.lab.local:/root/
```

3. On the controller node, convert the `.crt` to `.pem`:

```
# openssl x509 -in ca.crt -out ca.pem -outform PEM
```

4. Install the `.pem` on your OpenStack controller. For example, in Red Hat Enterprise Linux:

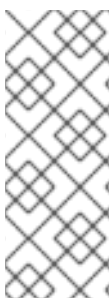
```
# cp ca.pem /etc/pki/ca-trust/source/anchors/
# update-ca-trust
```

5. Copy the `.crt` to the certificate directory:

```
# cp ca.crt /etc/ssl/certs/
```

2.8. CONFIGURE IDENTITY SERVICE

These steps prepare Identity Service for integration with IdM.



NOTE

If you are using Red Hat OpenStack Platform director, then you will need to be aware that some of configuration files referenced below are managed by Puppet. Consequently, any custom configuration you add might be overwritten whenever you run the **openstack overcloud deploy** process. To apply these settings to director-based deployments, see https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/12/html/integrate_with_identity_service/sec-director-ldap

2.8.1. Enable command line access to keystone v3

To manage Identity Service domains from the command line, you need to enable access to keystone v3.

Perform this procedure from the controller running the keystone service.

1. Create a copy of the existing environment variable file. In a director-based deployment, it will be called **overcloudrc**:

```
$ cp overcloudrc overcloudrc-v3
```

2. Edit the new `overcloudrc-v3` file:

- Change `OS_AUTH_URL` from `v2.0` to `v3`. For example:

```
export OS_AUTH_URL=https://controllerIP:5000/v3/
```

- Add the following entries to the bottom of `overcloudrc-v3`:

```
export OS_IDENTITY_API_VERSION=3
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=Default
```

3. Enable these options for your current command line session by sourcing the file:

```
$ source overcloudrc-v3
```

2.8.2. Configure the controller

NOTE

If you intend to update any configuration files, you need to be aware that certain OpenStack services now run within containers; this applies to keystone, nova, and cinder, among others. As a result, there are certain administration practices to consider:

- Do not update any configuration file you might find on the physical node's host operating system, for example, `/etc/cinder/cinder.conf`. This is because the containerized service does not reference this file.
- Do not update the configuration file running within the container. This is because any changes are lost once you restart the container. Instead, if you need to add any changes to containerized services, you will need to update the configuration file that is used to generate the container. These are stored within `/var/lib/config-data/puppet-generated/`

For example:

- keystone: `/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf`
- cinder: `/var/lib/config-data/puppet-generated/cinder/etc/cinder/cinder.conf`
- nova: `/var/lib/config-data/puppet-generated/nova/etc/nova/nova.conf`

Any changes will then be applied once you restart the container. For example:
`sudo docker exec -it keystone pkill -HUP -f keystone`

Perform this procedure from the controller running the keystone service:

1. Configure SELinux:

```
# setsebool -P authlogin_nsswitch_use_ldap=on
```

2. Create the *domains* directory:

```
# mkdir /var/lib/config-data/puppet-generated/keystone/etc/keystone/domains/
# chown 42425:42425 /var/lib/config-data/puppet-generated/keystone/etc/keystone/domains/
```

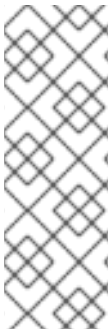
3. Configure Identity Service to use multiple back ends:



NOTE

You might need to install **crudini** using **yum install crudini**.

```
# crudini --set /var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf identity
domain_specific_drivers_enabled true
# crudini --set /var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf identity domain_config_dir
/etc/keystone/domains
# crudini --set /var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf assignment driver sql
```



NOTE

If you are using Red Hat OpenStack Platform director, then you will need to be aware that **/var/lib/config-data/puppet-generated/keystone/etc/keystone/keystone.conf** is managed by Puppet. Consequently, any custom configuration you add might be overwritten whenever you run the **openstack overcloud deploy** process. As a result, you might need to re-add this configuration manually each time. It is expected that a future release of director will include the Puppet parameters that will allow you to re-apply these settings automatically using a post-deployment script.

4. Enable multiple domains in dashboard. Add these lines to */etc/openstack-dashboard/local_settings*:

```
OPENSTACK_API_VERSIONS = {
    "identity": 3
}
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = 'Default'
```



NOTE

If you are using Red Hat OpenStack Platform director, then you will need to be aware that **/etc/openstack-dashboard/local_settings** is managed by Puppet. Consequently, any custom configuration you add might be overwritten whenever you run the **openstack overcloud deploy** process. As a result, you might need to re-add this configuration manually each time. It is expected that a future release of director will include the Puppet parameters that will allow you to re-apply these settings automatically using a post-deployment script.

Restart *httpd* to apply the settings:




```
# systemctl restart httpd
```

5. Configure an additional back end:

a. Create the keystone domain for IdM integration. You will need to decide on a name to use for your new keystone domain, and then create the domain. For example, this command creates a keystone domain named **LAB**:

```
# openstack domain create LAB
```



NOTE

If this command is not available, check that you have enabled *keystone v3* for your command line session.

b. Create the configuration file:

To add the IdM back end, enter the LDAP settings in a new file called `/var/lib/config-data/puppet-generated/keystone/etc/keystone/domains/keystone.LAB.conf` (where **LAB** is the domain name created previously). You will need to edit the sample settings below to suit your IdM deployment:

```
[ldap]
url = ldaps://idm.lab.local
user = uid=svc-ldap,cn=users,cn=accounts,dc=lab,dc=local
user_filter = (memberOf=cn=grp-
openstack,cn=groups,cn=accounts,dc=lab,dc=local)
password = RedactedComplexPassword
user_tree_dn = cn=users,cn=accounts,dc=lab,dc=local
user_objectclass = inetUser
user_id_attribute = uid
user_name_attribute = uid
user_mail_attribute = mail
user_pass_attribute =
user_allow_create = False
user_allow_update = False
user_allow_delete = False
tls_cacertfile = /etc/ssl/certs/ca.crt

[identity]
driver = ldap
```

Explanation of each setting:

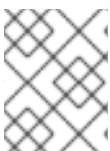
Setting	Description
url	The IdM server to use for authentication. Uses LDAPS port 636 .
user	The account in IdM to use for LDAP queries.
password	The plaintext password of the IdM account used above.

Setting	Description
<code>user_filter</code>	Filters the users presented to Identity Service. As a result, only members of the grp-openstack group can have permissions defined in Identity Service.
<code>user_tree_dn</code>	The path to the OpenStack accounts in IdM.
<code>user_objectclass</code>	Defines the type of LDAP user. For IdM, use the inetUser type.
<code>user_id_attribute</code>	Maps the IdM value to use for user IDs.
<code>user_name_attribute</code>	Maps the IdM value to use for <i>names</i> .
<code>user_mail_attribute</code>	Maps the IdM value to use for user email addresses.
<code>user_pass_attribute</code>	Leave this value blank.
<code>user_allow_create</code>	Set this value to False , as keystone only requires read-only access.
<code>user_allow_update</code>	Set this value to False , as keystone only requires read-only access.
<code>user_allow_delete</code>	Set this value to False , as keystone only requires read-only access.

6. Change ownership of the config file to the keystone user:

```
# chown 42425:42425 /var/lib/config-data/puppet-generated/keystone/etc/keystone/domains/keystone.LAB.conf
```

7. Grant the admin user access to the domain:



NOTE

This does not grant the OpenStack admin account any permissions in IdM. In this case, the term domain refers to OpenStack's usage of the keystone domain.

a. Get the **ID** of the *LAB* domain:

```
# openstack domain show LAB
+-----+-----+
| Field  | Value                                |
+-----+-----+
| enabled| True                                  |
| id     | 6800b0496429431ab1c4efbb3fe810d4    |
| name   | LAB                                  |
+-----+-----+
```

-
- b. Get the **ID** value of the *admin* user:

```
# openstack user list --domain default | grep admin
| 3d75388d351846c6a880e53b2508172a | admin      |
```

- c. Get the **ID** value of the *admin* role:

```
# openstack role list
+-----+-----+
| ID                | Name          |
+-----+-----+
| 544d48aaffde48f1b3c31a52c35f01f9 | SwiftOperator |
| 6d005d783bf0436e882c55c62457d33d | ResellerAdmin |
| 785c70b150ee4c778fe4de088070b4cf | admin         |
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_     |
+-----+-----+
```

- d. Use the returned domain and admin IDs to construct the command that adds the *admin* user to the *admin* role of the keystone LAB domain:

```
# openstack role add --domain 6800b0496429431ab1c4efbb3fe810d4 --user
3d75388d351846c6a880e53b2508172a 785c70b150ee4c778fe4de088070b4cf
```

- 8. Restart the keystone service to apply the changes:

```
# sudo docker exec -it keystone pkill -HUP -f keystone
```

- 9. View the list of users in the IdM domain by adding the keystone domain name to the command:

```
# openstack user list --domain LAB
```

- 10. View the service accounts in the local keystone database:

```
# openstack user list --domain default
```

2.8.3. Configure Compute to use keystone v3

Compute uses *keystone v2.0* by default, and so needs to be configured to use *keystone v3* in order to use multi-domain capabilities.

- 1. On each Compute node, and the controller, adjust the **keystone_authtoken** value:

On the Compute nodes:

```
# crudini --set /var/lib/config-data/puppet-generated/nova_libvirt/etc/nova/nova.conf keystone_authtoken auth_version
v3
```

On the controller:

```
# crudini --set /var/lib/config-data/puppet-generated/nova/etc/nova/nova.conf keystone_auth token auth_version v3
```

2. Restart these services on the controller to apply the changes:

```
# systemctl restart openstack-nova-api.service openstack-nova-cert.service
openstack-nova-conductor.service openstack-nova-consoleauth.service
openstack-nova-novncproxy.service openstack-nova-scheduler.service
# sudo docker exec -it keystone pkill -HUP -f keystone
```

3. Restart these services on each Compute node to apply the changes:

```
# systemctl restart openstack-nova-compute.service
```

2.8.4. Configure Block Storage to use keystone v3

You must also configure Block Storage (cinder) to authenticate to keystone v3.

1. In `/etc/cinder/cinder.conf`:

```
[keystone_auth token]
auth_uri = https://controllerIP:5000/v3
auth_version = v3
```

- **auth_uri** - replace **controllerIP** with the IP address of the controller. If your deployment has more than one controller, you should use the keystone endpoint VIP instead of the controller IP.

2. Restart **cinder-api** on all controllers:

```
# systemctl restart openstack-cinder-api
```

3. Restart **cinder-scheduler** on all controllers:

```
# systemctl restart openstack-cinder-scheduler
```

4. Restart **cinder-volume** (on one controller only):

```
# pcs resource restart openstack-cinder-volume
```

2.8.5. Allow IdM users to access Projects

IdM users that are members of the `grp-openstack` IdM group can be granted permission to log in to a project in the dashboard:

1. Retrieve a list of IdM users:

```
# openstack user list --domain LAB
+-----+
-----+
| ID                                     |
Name                                     |
```

```
+-----+
-----+
| 1f24ec1f11aeb90520079c29f70afa060d22e2ce92b2eba7784c841ac418091e | user1
|
| 12c062fidm5f8b065434d9ff6fce03eb9259537c93b411224588686e9a38bf1 | user2
|
| afaf48031eb54c3e44e4cb0353f5b612084033ff70f63c22873d181fdae2e73c | user3
|
| e47fc21dcf0d9716d2663766023e2d8dc15a6d9b01453854a898cabb2396826e | user4
|
+-----+
-----+
```

2. Retrieve a list of roles:

```
# openstack role list
+-----+-----+
| ID | Name |
+-----+-----+
| 544d48aaffde48f1b3c31a52c35f01f9 | SwiftOperator |
| 6d005d783bf0436e882c55c62457d33d | ResellerAdmin |
| 785c70b150ee4c778fe4de088070b4cf | admin |
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_ |
+-----+-----+
```

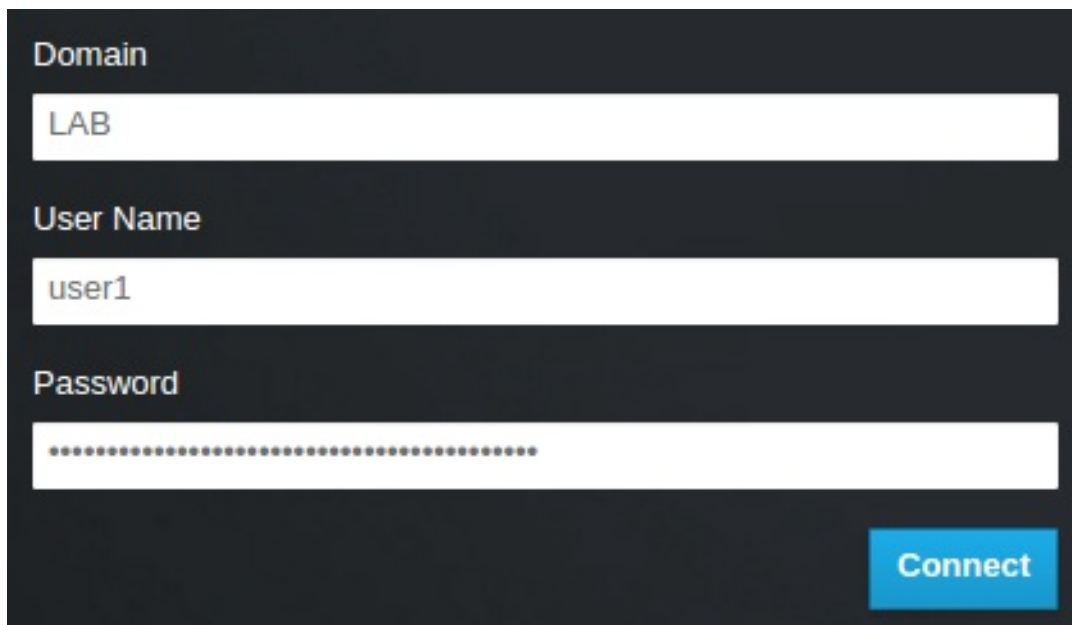
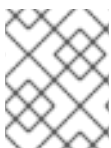
3. Grant users access to Projects by adding them to one or more of these roles. For example, if you want *user1* to be a general user of the *demo* project, you add them to the **_member_** role:

```
# openstack role add --project demo --user
1f24ec1f11aeb90520079c29f70afa060d22e2ce92b2eba7784c841ac418091e _member_
```

Or, if you want *user1* to be an administrative user of the *demo* project, you add them to the *admin* role:

```
# openstack role add --project demo --user
1f24ec1f11aeb90520079c29f70afa060d22e2ce92b2eba7784c841ac418091e admin
```

As a result, *user1* is able to log in to the dashboard by entering their IdM username and password, when also adding **LAB** in the Domain field:


**NOTE**

If users receive the error "Error: Unable to retrieve container list.", and expect to be able to manage containers, then they must be added to the *SwiftOperator* role.

2.9. GRANT ACCESS TO THE DOMAIN TAB

To allow the **admin** user to see the **Domain** tab, you will need to assign it the **admin** role in the **default** domain:

1. Find the **admin** user's UUID:

```
$ openstack user list | grep admin
| a6a8adb6356f4a879f079485dad1321b | admin |
```

2. Add the **admin** role in the **default** domain to the **admin** user:

```
$ openstack role add --domain default --user
a6a8adb6356f4a879f079485dad1321b admin
```

As a result, the **admin** user can now see the **Domain** tab.

2.10. CREATING A NEW PROJECT

After you have completed these integration steps, when you create a new project you will need to decide whether to create it in the **Default** domain, or in the keystone domain you've just created. This decision can be reached by considering your workflow, and how you administer user accounts. The **Default** domain can be thought of as an internal domain, used for service accounts and the **admin** project, so it might make sense for your AD-backed users to be placed within a different keystone domain; this does not strictly need to be the same keystone domain as the IdM users are in, and for separation purposes, there might be multiple keystone domains.

2.10.1. Changes to the dashboard log in process

Configuring multiple domains in Identity Service enables a new *Domain* field in the dashboard login

page.

Users are expected to enter the domain that matches their login credentials. This field must be manually filled with one of the domains present in keystone. Use the *openstack* command to list the available entries.

In this example, IdM accounts will need to specify the **LAB** domain. The built-in keystone accounts, such as *admin*, must specify **Default** as their domain:

```
# openstack domain list
+-----+-----+-----+-----+
| ID                | Name    | Enabled | Description
|
+-----+-----+-----+-----+
| 6800b0496429431ab1c4efbb3fe810d4 | LAB     | True    |
|
| default           | Default | True    | Owns users and
| tenants (i.e. projects) available on Identity API v2. |
+-----+-----+-----+-----+
```

2.10.2. Changes to the command line

For certain commands, you might need to specify the applicable domain. For example, appending **--domain LAB** in this command returns users in the LAB domain (that are members of the *grp-openstack* group):

```
# openstack user list --domain LAB
```

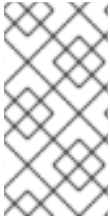
Appending **--domain Default** returns the built-in keystone accounts:

```
# openstack user list --domain Default
```

2.10.3. Test IdM integration

This procedure validates IdM integration by testing user access to dashboard features:

1. Create a test user in IdM, and add the user to the **grp-openstack** IdM group.
2. Add the user to the **_member_** role of the **demo** tenant.
3. Log in to the dashboard using the credentials of the IdM test user.
4. Click on each of the tabs to confirm that they are presented successfully without error messages.
5. Use the dashboard to build a test instance.

**NOTE**

If you experience issues with these steps, perform steps 3-5 with the built-in *admin* account. If successful, this demonstrates that OpenStack is still working as expected, and that an issue exists somewhere within the IdM \leftrightarrow Identity integration settings. See [Section 2.13, “Troubleshooting”](#).

2.11. CONFIGURE FOR HIGH AVAILABILITY

With keystone v3 enabled, you can make this configuration highly available by listing multiple IdM servers in the configuration file for that domain.

1. Add a second server to the **url** entry. For example, updating the **url** setting in the *keystone.LAB.conf* file will have Identity Service send all query traffic to the first IdM server in the list, *idm.lab.local*:

```
url = ldaps://idm.lab.local,ldaps://idm2.lab.local
```

If a query to *idm.lab.local* fails due to it being unavailable, Identity Service will attempt to query the next server in the list: *idm2.lab.local*. Note that this configuration does not perform queries in a round-robin fashion, so cannot be considered a load-balancing solution.

2. Set the network timeout in */etc/openldap/ldap.conf*:

```
NETWORK_TIMEOUT 2
```

In addition, if you have firewalls configured between the controller and the IdM servers, then you should not configure the IdM servers to silently drop packets from the controller. This will allow *python-keystoneclient* to properly detect outages and redirect the request to the next IdM server in the list.

**NOTE**

There might be connection delays while queries are being redirected to the second IdM server in the list. This is because the connection to the first server must first time out before the second is attempted.

2.12. CREATE A RC FILE FOR A NON-ADMIN USER

You might need to create a RC file for a non-admin user. For example:

```
$ cat overcloudrc-v3-user1
# Clear any old environment that may conflict.
for key in $( set | awk '{FS="="} /^OS_/ {print $1}' ); do unset $key ;
done
export OS_USERNAME=user1
export NOVA_VERSION=1.1
export OS_PROJECT_NAME=demo
export OS_PASSWORD=RedactedComplexPassword
export OS_NO_CACHE=True
export COMPUTE_API_VERSION=1.1
export no_proxy=,10.0.0.5,192.168.2.11
export OS_CLOUDNAME=overcloud
export OS_AUTH_URL=https://10.0.0.5:5000/v3
export OS_AUTH_TYPE=password
```



```
export PYTHONWARNINGS="ignore:Certificate has no, ignore:A true
SSLContext object is not available"
export OS_IDENTITY_API_VERSION=3
export OS_PROJECT_DOMAIN_NAME=Default
export OS_USER_DOMAIN_NAME=LAB
```

2.13. TROUBLESHOOTING

2.13.1. Test LDAP connections

Use *ldapsearch* to remotely perform test queries against the IdM server. A successful result here indicates that network connectivity is working, and the IdM services are up. In this example, a test query is performed against the server *idm.lab.local* on port 636:

```
# ldapsearch -D "cn=directory manager" -H ldaps://idm.lab.local:636 -b
"dc=lab,dc=local" -s sub "(objectclass=*)" -w RedactedComplexPassword
```



NOTE

ldapsearch is a part of the *openldap-clients* package. You can install this using `# yum install openldap-clients`.

2.13.2. Test port access

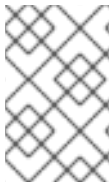
Use *nc* to check that the LDAPS port (636) is remotely accessible. In this example, a probe is performed against the server *idm.lab.local*. Press ctrl-c to exit the prompt.

```
# nc -v idm.lab.local 636
Ncat: Version 6.40 ( http://nmap.org/ncat )
Ncat: Connected to 192.168.200.10:636.
^C
```

Failure to establish a connection could indicate a firewall configuration issue.

CHAPTER 3. INTEGRATE WITH IDM USING NOVAJOIN

Novajoin allows you to enroll your nodes with Red Hat Identity Manager (IdM) as part of the deployment process. As a result, you can integrate IdM features with your OpenStack deployment, including identities, kerberos credentials, and access controls.



NOTE

IdM enrollment through novajoin is currently only available for the undercloud and overcloud nodes. Novajoin integration for overcloud instances is expected to be supported in a later release.

3.1. INSTALL AND CONFIGURE NOVAJOIN IN THE UNDERCLOUD

3.1.1. Add the undercloud to the CA

Before deploying the overcloud, you must add the undercloud to the Certificate Authority (CA):

1. On the undercloud node, install the **python-novajoin** package:

```
$ sudo yum install python-novajoin
```

2. On the undercloud node, run the **novajoin-ipa-setup** script, adjusting the values to suit your deployment:

```
$ sudo /usr/libexec/novajoin-ipa-setup \
  --principal admin \
  --password <IdM admin password> \
  --server <IdM server hostname> \
  --realm <overcloud cloud domain (in upper case)> \
  --domain <overcloud cloud domain> \
  --hostname <undercloud hostname> \
  --precreate
```

In the following section, you will use the resulting One-Time Password (OTP) to enroll the undercloud.

3.1.2. Add the undercloud to IdM

This procedure registers the undercloud with IdM and configures novajoin.

1. The novajoin service is disabled by default. To enable it, add an entry to **undercloud.conf**:

```
enable_novajoin = true
```

2. You need set a One-Time Password (OTP) to register the undercloud node with IdM:

```
ipa_otp = <otp>
```

3. Ensure the overcloud's domain name served by neutron's DHCP server matches the IdM domain (your kerberos realm in lowercase):

```
overcloud_domain_name = <domain>
```

4. Set the appropriate hostname for the undercloud:

```
undercloud_hostname = <undercloud FQDN>
```

5. Set IdM as the nameserver for the undercloud:

```
undercloud_nameservers = <IdM IP>
```

6. For larger environments, you will need to review the novajoin connection timeout values. In **undercloud.conf**, add a reference to a new file called **undercloud-timeout.yaml**:

```
hieradata_override = /home/stack/undercloud-timeout.yaml
```

Add the following options to **undercloud-timeout.yaml**. You can specify the timeout value in seconds, for example, **5**:

```
nova::api::vendordata_dynamic_connect_timeout: <timeout value>
nova::api::vendordata_dynamic_read_timeout: <timeout value>
```

7. Save the **undercloud.conf** file.
8. Run the undercloud deployment command to apply the changes to your existing undercloud:

```
$ openstack undercloud install
```

3.2. INSTALL AND CONFIGURE NOVAJOIN IN THE OVERCLOUD

These sections describe how to register an overcloud node with IdM.

3.2.1. Configure overcloud DNS

For automatic detection of your IdM environment, and easier enrollment, consider using IdM as your DNS server:

1. Connect to your undercloud:

```
$ source ~/stackrc
```

2. Configure the control plane subnet to use IdM as the DNS name server:

```
$ openstack subnet set ctlplane-subnet --dns-nameserver
<idm_server_address>
```

3. Set the **DnsServers** parameter in an environment file to use your IdM server:

```
parameter_defaults:
  DnsServers: ["<idm_server_address>"]
```

This parameter is usually defined in a custom **network-environment.yaml** file.

3.2.2. Configure overcloud to use novajoin

1. To enable IdM integration, create a copy of the `/usr/share/openstack-tripleo-heat-templates/environments/predictable-placement/custom-domain.yaml` environment file:

```
$ cp /usr/share/openstack-tripleo-heat-templates/environments/predictable-placement/custom-domain.yaml \
/home/stack/templates/custom-domain.yaml
```

2. Edit the `/home/stack/templates/custom-domain.yaml` environment file and set the **CloudDomain** and **CloudName*** values to suit your deployment. For example:

```
parameter_defaults:
  CloudDomain: lab.local
  CloudName: overcloud.lab.local
  CloudNameInternal: overcloud.internalapi.lab.local
  CloudNameStorage: overcloud.storage.lab.local
  CloudNameStorageManagement: overcloud.storagemgmt.lab.local
  CloudNameCtlplane: overcloud.ctlplane.lab.local
```

3. Include the following environment files in the overcloud deployment process:

- `/usr/share/openstack-tripleo-heat-templates/environments/enable-internal-tls.yaml`
- `/usr/share/openstack-tripleo-heat-templates/environments/tls-everywhere-endpoints-dns.yaml`
- `/home/stack/templates/custom-domain.yaml`

For example:

```
openstack overcloud deploy \
  --templates \
  -e /usr/share/openstack-tripleo-heat-templates/environments/enable-internal-tls.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/tls-everywhere-endpoints-dns.yaml \
  -e /home/stack/templates/custom-domain.yaml \
```

As a result, the deployed overcloud nodes will be automatically enrolled with IdM.

4. This only sets TLS for the internal endpoints. For the external endpoints you can use the normal means of adding TLS with the `./tripleo-heat-templates/environments/enable-tls.yaml` environment file (which must be modified to add your custom certificate and key). Consequently, your **openstack deploy** command would be similar to this:

```
openstack overcloud deploy \
  --templates \
  -e /usr/share/openstack-tripleo-heat-templates/environments/enable-internal-tls.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/tls-
```

```

everywhere-endpoints-dns.yaml \
  -e /home/stack/templates/custom-domain.yaml \
  -e /home/stack/templates/enable-tls.yaml

```

- Alternatively, you can also use IdM to issue your public certificates. In that case, you need to use the `./tripleo-heat-templates/environments/services/haproxy-public-tls-certmonger.yaml` environment file. For example:

```

openstack overcloud deploy \
  --templates \
  -e ./tripleo-heat-templates/environments/enable-internal-tls.yaml \
  -e /usr/share/openstack-tripleo-heat-templates/environments/tls-
everywhere-endpoints-dns.yaml \
  -e /home/stack/templates/custom-domain.yaml \
  -e ./tripleo-heat-templates/environments/services/haproxy-public-
tls-certmonger.yaml

```

3.3. VALIDATE A NODE IN IDM

- Locate an overcloud node in IdM and confirm that the host entry includes **Keytab: True**:

```

$ ipa host-show overcloud-node-01
Host name: overcloud-node-01.lab.local
Principal name: host/overcloud-node-01.lab.local@LAB.LOCAL
Principal alias: host/overcloud-node-01.lab.local@LAB.LOCAL
SSH public key fingerprint: <snip>
Password: False
Keytab: True
Managed by: overcloud-node-01.lab.local

```

- SSH to the node and confirm that `sss` can query IdM users. For example, to query an IdM user named **susan**:

```

$ getent passwd susan
uid=1108400007(susan) gid=1108400007(bob) groups=1108400007(susan)

```

CHAPTER 4. USING DOMAIN-SPECIFIC LDAP BACKENDS WITH DIRECTOR

Red Hat OpenStack Platform director can configure keystone to use one or more LDAP backends. This approach results in the creation of a separate LDAP backend for each keystone domain.

4.1. SETTING THE CONFIGURATION OPTIONS

For deployments using Red Hat OpenStack Platform director, you need to set the **KeystoneLDAPDomainEnable** flag to **true** in your heat templates; as a result, this will configure the **domain_specific_drivers_enabled** option in keystone (within the **identity** configuration group).



NOTE

The default directory for domain configuration files is set to **/etc/keystone/domains/**. You can override this by setting the required path using the **keystone::domain_config_directory** hiera key and adding it as an **ExtraConfig** parameter within an environment file.

You must also add a specification of the LDAP backend configuration. This is done using the **KeystoneLDAPBackendConfigs** parameter in **tripleo-heat-templates**. You can then specify your required LDAP options. For more information on the available options, see the [Configuration Reference Guide](#).

4.2. CONFIGURE THE DIRECTOR DEPLOYMENT



IMPORTANT

There is currently a known issue with a heat template: the **keystone_domain_config** tag is missing from **keystone.yaml**. For the workaround and further information, see https://bugzilla.redhat.com/show_bug.cgi?id=1519057.

1. Create a copy of the **keystone_domain_specific_ldap_backend.yaml** environment file:

```
$ cp /usr/share/openstack-tripleo-heat-templates/environments/services/keystone_domain_specific_ldap_backend.yaml /home/stack/templates/
```

2. Edit the **/home/stack/templates/keystone_domain_specific_ldap_backend.yaml** environment file and set the values to suit your deployment. For example, these entries create a LDAP configuration for a keystone domain named **testdomain**:

```
parameter_defaults:
  KeystoneLDAPDomainEnable: true
  KeystoneLDAPBackendConfigs:
    testdomain:
      url: ldaps://192.0.2.250
      user: cn=openstack,ou=Users,dc=director,dc=example,dc=com
      password: RedactedComplexPassword
      suffix: dc=director,dc=example,dc=com
      user_tree_dn: ou=Users,dc=director,dc=example,dc=com
```

```
    user_filter: "(memberOf=cn=OSuser,ou=Groups,dc=director,dc=example,dc=com)"
    user_objectclass: person
    user_id_attribute: cn
    user_allow_create: false
    user_allow_update: false
    user_allow_delete: false
```

3. You can also configure the environment file to specify multiple domains. For example:

```
KeystoneLDAPBackendConfigs:
  domain1:
    url: ldaps://domain1.example.com
    user: cn=openstack,ou=Users,dc=director,dc=example,dc=com
    password: RedactedComplexPassword
    ...
  domain2:
    url: ldaps://domain2.example.com
    user: cn=openstack,ou=Users,dc=director,dc=example,dc=com
    password: RedactedComplexPassword
    ...
```

This will result in two domains named **domain1** and **domain2**; each will have a different LDAP domain with its own configuration.