

Red Hat OpenStack Platform 16.1

Integrate with Identity Service

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

Last Updated: 2020-10-18

Red Hat OpenStack Platform 16.1 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 © 2020 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, the Red Hat 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 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
1.3.2. Outage requirements	6
1.3.3. Firewall configuration	6
1.4. CONFIGURE ACTIVE DIRECTORY DOMAIN SERVICES	6
1.5. CONFIGURE THE LDAPS CERTIFICATE	8
1.6. CONFIGURE IDENTITY SERVICE	8
1.6.1. Configure the controller	8
1.6.2. Allow Active Directory group members to access Projects	14
1.6.3. Allow Active Directory users to access Projects	16
1.7. GRANT ACCESS TO THE DOMAIN TAB	17
1.8. CREATING A NEW PROJECT	18
1.9. CHANGES TO THE DASHBOARD LOG IN PROCESS	18
1.10. CHANGES TO THE COMMAND LINE	18
1.11. TEST AD DS INTEGRATION	18
1.12. CONFIGURE HIGH AVAILABILITY	19
1.13. CREATE A RC FILE FOR A NON-ADMIN USER	19
1.14. TROUBLESHOOTING	20
1.14.1. Test LDAP connections	20
1.14.2. Test the Certificate Trust Configuration	20
1.14.3. Test port access	21
CHAPTER 2. IDENTITY MANAGEMENT INTEGRATION	22
CHAPTER 2. IDENTITY MANAGEMENT INTEGRATION	22 22
2.1. KEY TERMS	22
2.1. KEY TERMS 2.2. ASSUMPTIONS	22 22
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT	22 22 22
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options	22 22 22 22
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options 2.3.2. Outage requirements	22 22 22 22 22 23
2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options 2.3.2. Outage requirements 2.3.3. Firewall configuration	22 22 22 22 23 23
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options 2.3.2. Outage requirements 2.3.3. Firewall configuration 2.3.4. Configure the IdM server	222222222323
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options 2.3.2. Outage requirements 2.3.3. Firewall configuration 2.3.4. Configure the IdM server 2.3.5. Configure the LDAPS certificate	22 22 22 23 23 23 24
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options 2.3.2. Outage requirements 2.3.3. Firewall configuration 2.3.4. Configure the IdM server 2.3.5. Configure the LDAPS certificate 2.4. CONFIGURE IDENTITY SERVICE	22 22 22 23 23 23 24 24
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options 2.3.2. Outage requirements 2.3.3. Firewall configuration 2.3.4. Configure the IdM server 2.3.5. Configure the LDAPS certificate 2.4. CONFIGURE IDENTITY SERVICE 2.4.1. Configure the controller	22 22 22 23 23 23 24 24 25
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options 2.3.2. Outage requirements 2.3.3. Firewall configuration 2.3.4. Configure the IdM server 2.3.5. Configure the LDAPS certificate 2.4. CONFIGURE IDENTITY SERVICE 2.4.1. Configure the controller 2.4.2. Allow IdM group members to access Projects	22 22 22 23 23 23 24 24 25 29
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options 2.3.2. Outage requirements 2.3.3. Firewall configuration 2.3.4. Configure the IdM server 2.3.5. Configure the LDAPS certificate 2.4. CONFIGURE IDENTITY SERVICE 2.4.1. Configure the controller 2.4.2. Allow IdM group members to access Projects 2.4.3. Allow IdM users to access Projects	22 22 22 23 23 23 24 24 25 29
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options 2.3.2. Outage requirements 2.3.3. Firewall configuration 2.3.4. Configure the IdM server 2.3.5. Configure the LDAPS certificate 2.4. CONFIGURE IDENTITY SERVICE 2.4.1. Configure the controller 2.4.2. Allow IdM group members to access Projects 2.4.3. Allow IdM users to access Projects 2.5. GRANT ACCESS TO THE DOMAIN TAB	22 22 22 23 23 23 24 24 25 29 30 32
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options 2.3.2. Outage requirements 2.3.3. Firewall configuration 2.3.4. Configure the IdM server 2.3.5. Configure the LDAPS certificate 2.4. CONFIGURE IDENTITY SERVICE 2.4.1. Configure the controller 2.4.2. Allow IdM group members to access Projects 2.4.3. Allow IdM users to access Projects 2.5. GRANT ACCESS TO THE DOMAIN TAB 2.6. CREATING A NEW PROJECT	22 22 22 23 23 23 24 24 25 29 30 32
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options 2.3.2. Outage requirements 2.3.3. Firewall configuration 2.3.4. Configure the IdM server 2.3.5. Configure the LDAPS certificate 2.4. CONFIGURE IDENTITY SERVICE 2.4.1. Configure the controller 2.4.2. Allow IdM group members to access Projects 2.4.3. Allow IdM users to access Projects 2.5. GRANT ACCESS TO THE DOMAIN TAB 2.6. CREATING A NEW PROJECT 2.6.1. Changes to the dashboard log in process	22 22 22 23 23 23 24 24 25 29 30 32 32
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options 2.3.2. Outage requirements 2.3.3. Firewall configuration 2.3.4. Configure the IdM server 2.3.5. Configure the LDAPS certificate 2.4. CONFIGURE IDENTITY SERVICE 2.4.1. Configure the controller 2.4.2. Allow IdM group members to access Projects 2.4.3. Allow IdM users to access Projects 2.5. GRANT ACCESS TO THE DOMAIN TAB 2.6. CREATING A NEW PROJECT 2.6.1. Changes to the dashboard log in process 2.6.2. Changes to the command line	22 22 22 23 23 23 24 25 29 30 32 32 32 33
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options 2.3.2. Outage requirements 2.3.3. Firewall configuration 2.3.4. Configure the IdM server 2.3.5. Configure the LDAPS certificate 2.4. CONFIGURE IDENTITY SERVICE 2.4.1. Configure the controller 2.4.2. Allow IdM group members to access Projects 2.4.3. Allow IdM users to access Projects 2.5. GRANT ACCESS TO THE DOMAIN TAB 2.6. CREATING A NEW PROJECT 2.6.1. Changes to the dashboard log in process 2.6.2. Changes to the command line 2.6.3. Test IdM integration	22 22 22 23 23 24 24 25 29 30 32 32 32 33
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options 2.3.2. Outage requirements 2.3.3. Firewall configuration 2.3.4. Configure the IdM server 2.3.5. Configure the LDAPS certificate 2.4. CONFIGURE IDENTITY SERVICE 2.4.1. Configure the controller 2.4.2. Allow IdM group members to access Projects 2.4.3. Allow IdM users to access Projects 2.5. GRANT ACCESS TO THE DOMAIN TAB 2.6. CREATING A NEW PROJECT 2.6.1. Changes to the dashboard log in process 2.6.2. Changes to the command line 2.6.3. Test IdM integration 2.7. CONFIGURE FOR HIGH AVAILABILITY	22 22 22 23 23 23 24 24 25 29 30 32 32 32 33 33 33
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options 2.3.2. Outage requirements 2.3.3. Firewall configuration 2.3.4. Configure the IdM server 2.3.5. Configure the LDAPS certificate 2.4. CONFIGURE IDENTITY SERVICE 2.4.1. Configure the controller 2.4.2. Allow IdM group members to access Projects 2.4.3. Allow IdM users to access Projects 2.5. GRANT ACCESS TO THE DOMAIN TAB 2.6. CREATING A NEW PROJECT 2.6.1. Changes to the dashboard log in process 2.6.2. Changes to the command line 2.6.3. Test IdM integration 2.7. CONFIGURE FOR HIGH AVAILABILITY 2.8. CREATE A RC FILE FOR A NON-ADMIN USER	22 22 22 23 23 24 24 25 29 30 32 32 33 33 33
2.1. KEY TERMS 2.2. ASSUMPTIONS 2.3. IMPACT STATEMENT 2.3.1. High Availability options 2.3.2. Outage requirements 2.3.3. Firewall configuration 2.3.4. Configure the IdM server 2.3.5. Configure the LDAPS certificate 2.4. CONFIGURE IDENTITY SERVICE 2.4.1. Configure the controller 2.4.2. Allow IdM group members to access Projects 2.4.3. Allow IdM users to access Projects 2.5. GRANT ACCESS TO THE DOMAIN TAB 2.6. CREATING A NEW PROJECT 2.6.1. Changes to the dashboard log in process 2.6.2. Changes to the command line 2.6.3. Test IdM integration 2.7. CONFIGURE FOR HIGH AVAILABILITY 2.8. CREATE A RC FILE FOR A NON-ADMIN USER 2.9. TROUBLESHOOTING	22 22 22 23 23 23 24 24 25 29 30 32 32 32 33 33 33 34

	3.1. INSTALL AND CONFIGURE NOVAJOIN IN THE UNDERCLOUD	36
	3.1.1. Add the undercloud to the CA	36
	3.1.2. Add the undercloud to IdM	36
	3.2. INSTALL AND CONFIGURE NOVAJOIN IN THE OVERCLOUD	37
	3.2.1. Configure overcloud DNS	37
	3.2.2. Configure overcloud to use novajoin	38
	3.3. VALIDATE A NODE IN IDM	39
	3.4. CONFIGURE DNS ENTRIES FOR NOVAJOIN	39
С	HAPTER 4. IMPLEMENTING TLS-E WITH ANSIBLE	41
	4.1. CONFIGURING TLS-E ON THE UNDERCLOUD	41
	4.2. CONFIGURING TLS-E ON THE OVERCLOUD	42
С	HAPTER 5. USING DOMAIN-SPECIFIC LDAP BACKENDS WITH DIRECTOR	44
	5.1. SETTING THE CONFIGURATION OPTIONS	44
	5.2 CONFIGURE THE DIRECTOR DEDLOVMENT	11

PREFACE

Identity Service (codename *keystone*) provides authentication and authorization for Red Hat OpenStack Platform 16.1.

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 director, see Chapter 5, *Using domain-specific LDAP backends with director*. This is because the 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.

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 1.12, "Configure high availability" for more information.

1.3.2. 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.

1.3.3. Firewall configuration

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

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

1.4. 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:

 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-Idap -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"

- 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.

1.5. CONFIGURE THE LDAPS CERTIFICATE



NOTE

When using multiple domains for LDAP authentication, you might receive various errors, such as **Unable to retrieve authorized projects**, or **Peer's Certificate issuer is not recognized**. This can arise if keystone uses the incorrect certificate for a certain domain. As a workaround, merge all of the LDAPS public keys into a single **.crt** bundle, and configure all of your keystone domains to use this file.

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/pki/ca-trust/source/anchors



NOTE

Optionally, if you need to run diagnostic commands, such as **Idapsearch**, 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

1.6. CONFIGURE IDENTITY SERVICE

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



NOTE

If you are using director, note that the 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 5, *Using domain-specific LDAP backends with director*.

1.6.1. Configure the controller



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/puppetgenerated/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 service. For example: sudo systemctl restart tripleo_keystone

Perform this procedure on each OpenStack node running the keystone service:

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 **dnf 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



If you are using director, note 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 Chapter 5, Using domain-specific LDAP backends with director.

4. Enable multiple domains in dashboard. Add these lines to /var/lib/config-data/puppet-generated/horizon/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 director, note that /var/lib/config-data/puppet-generated/horizon/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.

Restart the horizon container to apply the settings:

- \$ sudo systemctl restart tripleo_horizon
- 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:





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
                  = RedactedComplexPassword
password
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
user_enabled_default = 512
user_attribute_ignore = password,tenant_id,tenants
group_objectclass
                     = group
                   = OU=labUsers,DC=lab,DC=local
group_tree_dn
group_filter
                 = (CN=grp-openstack*)
group_id_attribute
group name attribute
                      = name
use tls
                = False
                     =/etc/pki/ca-trust/source/anchors/anchorsaddc.lab.local.pem
tls_cacertfile
query scope
                     = sub
chase referrals
                        = false
[identity]
driver = Idap
```

Explanation of each setting:

Setting	Description
url	The AD Domain Controller to use for authentication. Uses LDAPS port 636 .
user	The Distinguished Name of an AD account to use for LDAP queries. For example, you can locate the Distinguished Name value of the svc-Idap account in AD using Get-ADuser svc-Idap select DistinguishedName
password	The plaintext password of the AD account used above.
suffix	The Distinguished Name of your AD domain. You can locate this value using Get- ADDomain select DistinguishedName

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

Setting	Description
group_name_attribute	Maps the AD value to use for group names.
use_tls	Defines whether TLS is to be used. This needs to be disabled if you are encrypting with LDAPS rather than STARTTLS.
tls_cacertfile	Specifies the path to the .crt certificate file.
query_scope	Configures Identity Service to also search within nested child OUs, when locating users that are members of the grp-openstack group.
chase_referrals	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/puppetgenerated/keystone/etc/keystone/domains/keystone.LAB.conf

- 7. Restart the keystone service to apply the changes:
 - # sudo podman 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:

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

1.6.2. 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 Section 1.6.3, "Allow 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.
- Have a designated project in mind. This example uses a project called demo, created using openstack project create --domain default --description "Demo Project" demo.

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 listdom	nain LAB +	
ID	Name	
•	f98a59b66934fb1d6b7f745f14	•
'	7e4aa331820f9d3be52654aa	8094e698a9182cbb88 grp-
	cc7af4c0773e78d61b5f81321	809f8323216938cae8 grp-
+	+	+

2. Retrieve a list of roles:

3. 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:



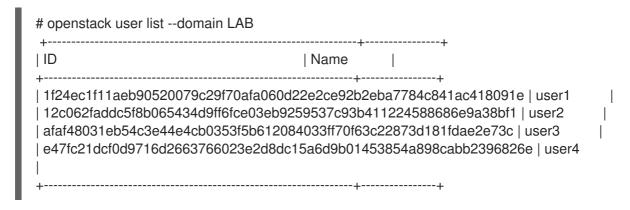


If users receive the error **Error: Unable to retrieve container list.**, and expect to be able to manage containers, then they must be added to the **SwiftOperator** role.

1.6.3. 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:

1. Retrieve a list of AD users:



2. Retrieve a list of roles:

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 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.

1.7. 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.

1.8. 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 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.

1.9. 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:

1.10. 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

1.11. 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.



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 \longleftrightarrow Identity integration settings. See Section 1.14, "Troubleshooting".

1.12. 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.

 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.

2. Set the network timeout in /etc/openIdap/Idap.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.

1.13. 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

1.14. TROUBLESHOOTING

1.14.1. 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 **Idapsearch** 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 **addc.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

Idapsearch is a part of the **openIdap-clients** package. You can install this using **# dnf install openIdap-clients**

1.14.2. Test the Certificate Trust Configuration

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

/etc/openIdap/Idap.conf

TLS_CACERTDIR /etc/openIdap/certs



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

This setting must not be permanently configured

/etc/openIdap/Idap.conf

TLS_REQCERT allow

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

1.14.3. 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 director, see Chapter 5, *Using domain-specific LDAP backends with director*. This is because the 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.



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 2.7, "Configure for high availability" for more information.

2.3.2. Outage requirements

- The Identity Service will need to be restarted in order to add the IdM back end.
- 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.

2.3.3. Firewall configuration

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

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

2.3.4. 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 [radministrator]: 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

4. Login as svc-ldap user and perform the password change when prompted:

-

kinit svc-ldap

2.3.5. Configure the LDAPS certificate



NOTE

When using multiple domains for LDAP authentication, you might receive various errors, such as **Unable to retrieve authorized projects**, or **Peer's Certificate issuer is not recognized**. This can arise if keystone uses the incorrect certificate for a certain domain. As a workaround, merge all of the LDAPS public keys into a single **.crt** bundle, and configure all of your keystone domains to use this file.

 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 OpenStack node that runs the keystone service. For example, this command uses *scp* to copy ca.crt to the node named *node.lab.local*:
 - # scp /etc/ipa/ca.crt root@node.lab.local:/root/
- 3. On the OpenStack node, convert the .crt to .pem:
 - # openssl x509 -in ca.crt -out ca.pem -outform PEM
- 4. Copy the .crt to the certificate directory. This is the location that the keystone service will use to access the certificate:
 - # cp ca.crt/etc/pki/ca-trust/source/anchors



NOTE

Optionally, if you need to run diagnostic commands, such as **Idapsearch**, you will also need to add the certificate to the RHEL certificate store. For example:

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

2.4. CONFIGURE IDENTITY SERVICE

These steps prepare Identity Service for integration with IdM.



NOTE

If you are using director, note that the 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 5, *Using domain-specific LDAP backends with director*.

2.4.1. 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/puppetgenerated/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 systemctl restart tripleo_keystone

Perform this procedure on the controller running the keystone service:

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 Identity Service to use multiple back ends:



NOTE

You might need to install **crudini** using **dnf 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 director, note 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 Chapter 5, Using domain-specific LDAP backends with director.

4. Enable multiple domains in dashboard. Add these lines to /var/lib/config-data/puppet-generated/horizon/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 director, note that /var/lib/config-data/puppet-generated/horizon/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.

Restart the horizon container to apply the settings:

\$ sudo systemctl restart tripleo_horizon

- 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
 - 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 =
group_tree_dn
                      = cn=groups,cn=accounts,dc=lab,dc=local
group_objectclass
                         = groupOfNames
group_id_attribute
                      = cn
group_name_attribute
                        = cn
group_member_attribute = member
group_desc_attribute
                      = description
                = False
use_tls
query_scope
                      = sub
chase_referrals
                         = false
tls_cacertfile =/etc/pki/ca-trust/source/anchors/anchorsca.crt
[identity]
driver = Idap
```

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



Integration with an IdM group will only return direct members, and not nested groups. As a result, queries that rely on

LDAP_MATCHING_RULE_IN_CHAIN or

memberof:1.2.840.113556.1.4.1941: will not currently work with IdM.

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

chown 42425:42425 /var/lib/config-data/puppetgenerated/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:

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 podman restart 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

2.4.2. Allow IdM group members to access Projects

To allow authenticated users access to OpenStack resources, the recommended method is to authorize certain IdM 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 IdM groups are granted roles in Projects. As a result, IdM users that are members of these IdM groups will be able to access pre-determined Projects.



NOTE

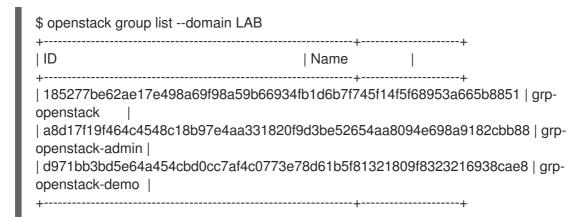
If you would prefer to manually manage the authorization of individual IdM users, see the Section 2.4.3, "Allow IdM users to access Projects".

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

- Create a group named grp-openstack-admin in IdM.
- Create a group named **grp-openstack-demo** in IdM.
- Add your IdM users to one of the above groups, as needed.
- Add your IdM users to the **grp-openstack** group.
- Have a designated project in mind. This example uses a project called demo, created using openstack project create --domain default --description "Demo Project" demo.

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

1. Retrieve a list of IdM groups:



2. Retrieve a list of roles:

3. Grant the IdM 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 IdM username and password, when also entering **LAB** in the Domain field:





NOTE

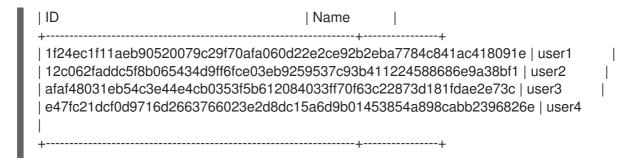
If users receive the error **Error: Unable to retrieve container list.**, and expect to be able to manage containers, then they must be added to the **SwiftOperator** role.

2.4.3. Allow IdM users to access Projects

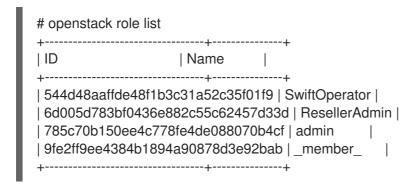
IdM users that are members of the **grp-openstack** IdM group can be granted permission to log in to a *Project* in the dashboard:

1. Retrieve a list of IdM users:

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



2. Retrieve a list of roles:



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 entering **LAB** in the **Domain** field:





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.

2.5. 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.

2.6. 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 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.

2.6.1. 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:

2.6.2. 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

2.6.3. 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 \longleftrightarrow Identity integration settings. See Section 2.9, "Troubleshooting".

2.7. CONFIGURE FOR HIGH AVAILABILITY

You can make this configuration highly available by listing multiple IdM servers in the configuration file for that domain.

 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.

2.8. 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

2.9. TROUBLESHOOTING

2.9.1. Test LDAP connections

Use *Idapsearch* 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:

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



NOTE

Idapsearch is a part of the *openIdap-clients* package. You can install this using **# dnf install openIdap-clients**.

2.9.2. 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 **python3-novajoin** package:
 - \$ sudo dnf install python3-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. Configure the following settings in **undercloud.conf** (within the **[DEFAULT]** section):

1. The novajoin service is disabled by default. To enable it:

```
[DEFAULT]
enable_novajoin = true
```

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

3. Set the overcloud's domain name to be served by neutron's DHCP server:

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/ssl/enable-internal-tls.yaml
 - /usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-everywhereendpoints-dns.yaml
 - /home/stack/templates/custom-domain.yaml
 For example:

от схаттртс.

openstack overcloud deploy \

- --templates \
- -e /usr/share/openstack-tripleo-heat-templates/environments/ssl/enable-internal-tls.yaml \backslash
- -e /usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-everywhere-endpoints-dns.yaml \setminus
 - -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 /usr/share/openstack-tripleo-heat-templates/environments/ssl/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/ssl/enable-internal-tls.yaml \
- -e /usr/share/openstack-tripleo-heat-templates/environments/ssl/tls-everywhere-endpoints-dns.yaml \
- -e /home/stack/templates/custom-domain.yaml \
- -e /home/stack/templates/enable-tls.yaml

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

openstack overcloud deploy \

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



NOTE

You can no longer use novajoin to implement TLS everywhere (TLS-e) on a pre-existing deployment. For information on hardening an existing deployment of Red Hat OpenStack Platform using TLS-e, see Chapter 4, Implementing TLS-e with Ansible.

3.3. VALIDATE A NODE IN IDM

1. 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

2. SSH to the node and confirm that sssd 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)

3.4. CONFIGURE DNS ENTRIES FOR NOVAJOIN

If you use the **haproxy-public-tls-certmonger.yaml** template to issue public certificates for endpoints, then you will need to manually create DNS entries for the VIP endpoints used by Novajoin:

1. Identify the overcloud networks. You can expect to locate these in /home/stack/virt/network/network-environment.yaml:

parameter defaults:

ControlPlaneDefaultRoute: 192.168.24.1

ExternalAllocationPools:

- end: 10.0.0.149 start: 10.0.0.101

InternalApiAllocationPools:

- end: 172.17.1.149

start: 172.17.1.10 StorageAllocationPools: - end: 172.17.3.149 start: 172.17.3.10

StorageMgmtAllocationPools:

- end: 172.17.4.149 start: 172.17.4.10

2. Create a list of virtual IP addresses (VIP) for each overcloud network. For example: /home/stack/virt/public_vip.yaml

parameter_defaults:
ControlFixedIPs: [{'ip_address':'192.168.24.101'}]
PublicVirtualFixedIPs: [{'ip_address':'10.0.0.101'}]
InternalApiVirtualFixedIPs: [{'ip_address':'172.17.1.101'}]
StorageVirtualFixedIPs: [{'ip_address':'172.17.3.101'}]
StorageMgmtVirtualFixedIPs: [{'ip_address':'172.17.4.101'}]

RedisVirtualFixedIPs: [{'ip_address':'172.17.1.102'}]

3. Add DNS entries to IdM for each of the VIPs. You may also need to create new zones. The following example demonstrates DNS record and zone creation for IdM:

ipa dnsrecord-add lab.local overcloud --a-rec 10.0.0.101
ipa dnszone-add ctlplane.lab.local
ipa dnsrecord-add ctlplane.lab.local overcloud --a-rec 192.168.24.101
ipa dnszone-add internalapi.lab.local
ipa dnsrecord-add internalapi.lab.local overcloud --a-rec 172.17.1.101
ipa dnszone-add storage.lab.local
ipa dnsrecord-add storage.lab.local overcloud --a-rec 172.17.3.101
ipa dnszone-add storagemgmt.lab.local
ipa dnsrecord-add storagemgmt.lab.local

CHAPTER 4. IMPLEMENTING TLS-E WITH ANSIBLE

Red Hat recommends the new ansible-based **tripleo-ipa** method over the default **novajoin** method to configure your undercloud and overcloud with TLS-e. You can use the following procedure to implement TLSe on either a new installation of Red Hat OpenStack Platform, or an existing deployment you wish to configure with TLS-e. You must use this method if you deploy Red Hat OpenStack Platform with TLS-e on pre-provisioned nodes.

4.1. CONFIGURING TLS-E ON THE UNDERCLOUD

Prerequisites

Ensure that all configuration steps for the undercloud, such as the creation of the stack user, are complete. For more details, see Director Installation and Usage for more details

Procedure

Configure the hosts file
 Set the appropriate search domains and the nameserver on the undercloud in /etc/resolv.conf.
 For example, if the deployment domain is example.com, and the domain of the FreeIPA server is bigcorp.com, then add the following lines to /etc/resolv.conf:

search example.com bigcorp.com nameserver \$IDM_SERVER_IP_ADDR

- 2. Install required software:
 - sudo yum install -y python3-ipalib python3-ipaclient krb5-devel
- 3. Export environmental variables with values specific to your environment.:

```
export IPA_DOMAIN=bigcorp.com
export IPA_REALM=BIGCORP.COM
export IPA_ADMIN_USER=$IPA_USER
export IPA_ADMIN_PASSWORD=$IPA_PASSWORD
export IPA_SERVER_HOSTNAME=ipa.bigcorp.com
export UNDERCLOUD_FQDN=undercloud.example.com
export USER=stack
export CLOUD_DOMAIN=example.com
```



NOTE

The IdM user credentials must be an administrative user that can add new hosts and services.

4. Run the **undercloud-ipa-install.yaml** ansible playbook on the undercloud:

ansible-playbook \
--ssh-extra-args "-o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null" \
/usr/share/ansible/tripleo-playbooks/undercloud-ipa-install.yaml

5. Add the following parameters to undercloud.conf

undercloud_nameservers = \$IDM_SERVER_IP_ADDR overcloud_domain_name = example.com

6. Deploy the undercloud:

openstack undercloud install

Verification

Verify that the undercloud was enrolled correctly by completing the following steps:

1. List the hosts in IdM:

\$ kinit \$ ipa host-find

2. Confirm that /etc/novajoin/krb5.keytab exists on the undercloud.

Is /etc/novajoin/krb5.keytab



NOTE

The **novajoin** directory name is for legacy naming purposes only.

4.2. CONFIGURING TLS-E ON THE OVERCLOUD

When you deploy the overcloud with TLS everywhere (TLS-e), IP addresses from the Undercloud and Overcloud will automatically be registered with IdM. This feature is provided as a technology preview. This feature is available in this release as a *Technology Preview*, and therefore is not fully supported by Red Hat. It should only be used for testing, and should not be deployed in a production environment. For more information about Technology Preview features, see Scope of Coverage Details.



NOTE

To disable automatic IP address registration, set the **IDMModifyDNS** heat parameter to false:

parameter_defaults: IdMModifyDNS: false

- 1. Before deploying the overcloud, create a YAML file **tls-parameters.yaml** with contents similar to the following. The values you select will be specific for your environment:
 - The **DnsServers** parameter should have a value that reflects the IP address of the IdM server.
 - If the domain of the IdM server is different than the cloud domain, include it in the DnsSearchDomains parameter. For example: DnsSearchDomains: ["example.com", "bigcorp.com"]

- The shown value of the **OS::TripleO::Services::IpaClient** parameter overrides the default setting in the **enable-internal-tls.yaml** file. You must ensure the **tls-parameters.yaml** file follows **enable-internal-tls.yaml** in the **openstack overcloud deploy** command.
- If you are running a distributed compute node (DCN) architecture with cinder configured as active-active, you must set the **EnableEtcdInternalTLS** parameter to **true**.

parameter_defaults:

DnsSearchDomains: ["example.com"]

DnsServers: ["192.168.1.13"] CloudDomain: example.com

CloudName: overcloud.example.com

CloudNameInternal: overcloud.internalapi.example.com CloudNameStorage: overcloud.storage.example.com

CloudNameStorageManagement: overcloud.storagemgmt.example.com

CloudNameCtlplane: overcloud.ctlplane.example.com

resource_registry:

OS::TripleO::Services::IpaClient: /usr/share/openstack-tripleo-heat-templates/deployment/ipa/ipaservices-baremetal-ansible.yaml

2. Deploy the overcloud. You will need to include the tls-parameters.yaml in the deployment command:

DEFAULT_TEMPLATES=/usr/share/openstack-tripleo-heat-templates/ CUSTOM TEMPLATES=/home/stack/templates

openstack overcloud deploy \

- -e \${DEFAULT_TEMPLATES}/environments/ssl/tls-everywhere-endpoints-dns.yaml \
- -e \${DEFAULT TEMPLATES}/environments/services/haproxy-public-tls-certmonger.yaml \
- -e \${DEFAULT_TEMPLATES}/environments/ssl/enable-internal-tls.yaml \
- -e \${CUSTOM TEMPLATES}/tls-parameters.yaml \

...

3. Confirm each endpoint is using HTTPS by querying keystone for a list of endpoints:

openstack overcloud endpoint list

CHAPTER 5. 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.

5.1. 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**, where you can then specify your required LDAP options.

5.2. CONFIGURE THE DIRECTOR DEPLOYMENT

1. Create a copy of the **keystone domain specific Idap 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
```

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

KeystoneLDAPBackendConfigs: domain1:

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