Red Hat OpenStack Platform 16.1

Integrate with Identity Service

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

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

OpenStack Team
rhos-docs@redhat.com
Abstract

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

MAKING OPEN SOURCE MORE INCLUSIVE ................................................................. 4

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION ........................................... 5

CHAPTER 1. INTEGRATING ACTIVE DIRECTORY ......................................................... 6
  1.1. CONFIGURING ACTIVE DIRECTORY DOMAIN SERVICES ............................... 6
  1.2. CONFIGURE THE LDAPS CERTIFICATE ...................................................... 7
  1.3. CONFIGURE IDENTITY SERVICE ............................................................. 8
      1.3.1. Configure the controller ................................................................. 8
      1.3.2. Allow Active Directory group members to access Projects .................... 14
      1.3.3. Allow Active Directory users to access Projects ................................ 16
  1.4. GRANT ACCESS TO THE DOMAIN TAB ...................................................... 17
  1.5. CREATING A NEW PROJECT .................................................................. 18
  1.6. CHANGES TO THE DASHBOARD LOG IN PROCESS ...................................... 18
  1.7. CHANGES TO THE COMMAND LINE .......................................................... 18
  1.8. TEST AD DS INTEGRATION .................................................................... 18
  1.9. CREATE A RC FILE FOR A NON-ADMIN USER .......................................... 19
      1.10. TROUBLESHOOTING ........................................................................ 19
           1.10.1. Test LDAP connections ............................................................... 19
           1.10.2. Test the Certificate Trust Configuration ....................................... 20
           1.10.3. Test port access .................................................................. 20

CHAPTER 2. IDENTITY MANAGEMENT INTEGRATION ............................................... 21
  2.1. KEY TERMS ......................................................................................... 21
  2.2. ASSUMPTIONS ...................................................................................... 21
  2.3. IMPACT STATEMENT ........................................................................... 21
      2.3.1. High Availability options .............................................................. 21
      2.3.2. Outage requirements .................................................................. 22
      2.3.3. Firewall configuration .................................................................. 22
      2.3.4. Configure the IdM server .............................................................. 22
      2.3.5. Configure the LDAPS certificate .................................................... 23
  2.4. CONFIGURE IDENTITY SERVICE ............................................................ 23
      2.4.1. Configure the controller .............................................................. 24
      2.4.2. Allow IdM group members to access Projects .................................... 28
      2.4.3. Allow IdM users to access Projects ................................................ 29
  2.5. GRANT ACCESS TO THE DOMAIN TAB .................................................... 31
  2.6. CREATING A NEW PROJECT .................................................................. 31
      2.6.1. Changes to the dashboard log in process ......................................... 31
      2.6.2. Changes to the command line ....................................................... 32
      2.6.3. Test IdM integration .................................................................. 32
  2.7. CREATE A RC FILE FOR A NON-ADMIN USER ........................................ 32
  2.8. TROUBLESHOOTING ........................................................................... 33
      2.8.1. Test LDAP connections ............................................................... 33
      2.8.2. Test port access .................................................................. 33

CHAPTER 3. INTEGRATE WITH IDM USING NOVAJOIN ............................................. 34
  3.1. INSTALL AND CONFIGURE NOVAJOIN IN THE UNDERCLOUD .................. 34
      3.1.1. Adding the undercloud to the CA .................................................... 34
      3.1.2. Adding the undercloud to IdM ....................................................... 34
  3.2. INSTALL AND CONFIGURE NOVAJOIN IN THE OVERCLOUD .................. 35
      3.2.1. Configure overcloud DNS .............................................................. 35
      3.2.2. Configure the overcloud to use novajoin ....................................... 36
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3. VALIDATE A NODE IN IDM</td>
<td>37</td>
</tr>
<tr>
<td>3.4. CONFIGURE DNS ENTRIES FOR NOVAJOIN</td>
<td>37</td>
</tr>
<tr>
<td>CHAPTER 4. IMPLEMENTING TLS-E WITH ANSIBLE</td>
<td>39</td>
</tr>
<tr>
<td>4.1. CONFIGURING TLS-E ON THE UNDERCLOUD</td>
<td>39</td>
</tr>
<tr>
<td>4.2. CONFIGURING TLS-E ON THE OVERCLOUD</td>
<td>40</td>
</tr>
<tr>
<td>CHAPTER 5. USING DOMAIN-SPECIFIC LDAP BACKENDS WITH DIRECTOR</td>
<td>42</td>
</tr>
<tr>
<td>5.1. SETTING THE CONFIGURATION OPTIONS</td>
<td>42</td>
</tr>
<tr>
<td>5.2. CONFIGURE THE DIRECTOR DEPLOYMENT</td>
<td>42</td>
</tr>
</tbody>
</table>
MAKING OPEN SOURCE MORE INCLUSIVE

Red Hat is committed to replacing problematic language in our code, documentation, and web properties. We are beginning with these four terms: master, slave, blacklist, and whitelist. Because of the enormity of this endeavor, these changes will be implemented gradually over several upcoming releases. For more details, see our CTO Chris Wright’s message.
Providing Feedback on Red Hat Documentation

We appreciate your input on our documentation. Tell us how we can make it better.

Using the Direct Documentation Feedback (DDF) function

Use the Add Feedback DDF function for direct comments on specific sentences, paragraphs, or code blocks.

1. View the documentation in the Multi-page HTML format.
2. Ensure that you see the Feedback button in the upper right corner of the document.
3. Highlight the part of text that you want to comment on.
4. Click Add Feedback.
5. Complete the Add Feedback field with your comments.
6. Optional: Add your email address so that the documentation team can contact you for clarification on your issue.
7. Click Submit.
CHAPTER 1. INTEGRATING ACTIVE DIRECTORY

Identity service

The OpenStack Identity (keystone) service provides authentication and authorization for Red Hat OpenStack Platform (RHOSP).

You can integrate the Identity service with Microsoft Active Directory Domain Service (AD DS), Red Hat Identity Management (IdM), and LDAP.

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

By integrating the Identity service with AD DS, you allow AD DS users to authenticate to Red Hat OpenStack Platform (RHOSP) to access resources. RHOSP service accounts, such as the Identity Service and the Image service, and authorization management remain in the Identity Service database. Permissions and roles are assigned to the AD DS accounts using Identity Service management tools.

1.1. CONFIGURING ACTIVE DIRECTORY DOMAIN SERVICES

Prerequisites

- 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.

NOTE

Implemented AD DS with a high availability or load balancing solution to avoid a single point of failure.

Procedure

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

   ```powershell
   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:
3. Create a group for RHOSP 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 RHOSP administrators.

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

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

Send this value to the RHOSP administrators.

### 1.2. 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 -inform der -in addc.lab.local.cer -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 `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`:
   ```bash
   # 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:
   ```bash
   # cp addc.lab.local.pem /etc/pki/ca-trust/source/anchors/
   # update-ca-trust
   ```

1.3. 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*.

1.3.1. 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/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 service. For example: `sudo systemctl restart tripleo_keystone`

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

1. Configure SELinux:

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

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

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

2. Create the `domains` directory:

   ```bash
   # 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`.

   ```bash
   # 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
   ```
4. Enable multiple domains in dashboard. Add these lines to `/var/lib/config-data/puppet-generated/horizon/etc/openstack-dashboard/local_settings`:

```python
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. For director-based deployments, see Chapter 5, Using domain-specific LDAP backends with director.
```

Restart the horizon container to apply the settings:

```bash
$ sudo systemctl restart tripleo_horizon
```

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:

      ```bash
      $ 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=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
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
use_tls                  = False
tls_cacertfile                  = /etc/pki/ca-trust/source/anchors/anchorsaddc.lab.local.pem
query_scope                  = sub
chase_referrals                  = false

[identity]
driver = ldap

Explanation of each setting:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>The AD Domain Controller to use for authentication. Uses LDAPS port 636.</td>
</tr>
<tr>
<td>user</td>
<td>The Distinguished Name of an AD account to use for LDAP queries. For example, you can locate the Distinguished Name value of the svc-ldap account in AD using `Get-ADUser svc-ldap</td>
</tr>
<tr>
<td>password</td>
<td>The plaintext password of the AD account used above.</td>
</tr>
<tr>
<td>suffix</td>
<td>The Distinguished Name of your AD domain. You can locate this value using `Get-ADDomain</td>
</tr>
<tr>
<td>Setting</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>user_tree_dn</td>
<td>The Organizational Unit (OU) that contains the OpenStack accounts.</td>
</tr>
<tr>
<td>user_objectclass</td>
<td>Defines the type of LDAP user. For AD, use the <strong>person</strong> type.</td>
</tr>
<tr>
<td>user_filter</td>
<td>Filters the users presented to Identity Service. As a result, only members of the <strong>grp-openstack</strong> group can have permissions defined in Identity Service. This value requires the full <strong>Distinguished Name</strong> of the group: `Get-ADGroup grp-openstack</td>
</tr>
<tr>
<td>user_id_attribute</td>
<td>Maps the AD value to use for user IDs.</td>
</tr>
<tr>
<td>user_name_attribute</td>
<td>Maps the AD value to use for <strong>names</strong>.</td>
</tr>
<tr>
<td>user_mail_attribute</td>
<td>Maps the AD value to use for user email addresses.</td>
</tr>
<tr>
<td>user_pass_attribute</td>
<td>Leave this value blank.</td>
</tr>
<tr>
<td>user_enabled_attribute</td>
<td>The AD setting that validates whether the account is enabled.</td>
</tr>
<tr>
<td>user_enabled_mask</td>
<td>Defines the value to check to determine whether an account is enabled. Used when booleans are not returned.</td>
</tr>
<tr>
<td>user_enabled_default</td>
<td>The AD value that indicates that an account is enabled.</td>
</tr>
<tr>
<td>user_attribute_ignore</td>
<td>Defines user attributes that Identity Service should disregard.</td>
</tr>
<tr>
<td>group_objectclass</td>
<td>Maps the AD value to use for <strong>groups</strong>.</td>
</tr>
<tr>
<td>group_tree_dn</td>
<td>The Organizational Unit (OU) that contains the user groups.</td>
</tr>
<tr>
<td>group_filter</td>
<td>Filters the groups presented to Identity Service.</td>
</tr>
<tr>
<td>group_id_attribute</td>
<td>Maps the AD value to use for group IDs.</td>
</tr>
</tbody>
</table>
### 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:

   ```bash
   # 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:

   ```bash
   # sudo systemctl restart tripleo_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:

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

   b. Get the ID value of the `admin` user:
c. Get the ID value of the `admin` role:

```
# openstack role list
+----------------------------------+-----------------+
| ID                               | Name            |
+----------------------------------+-----------------+
| 01d92614cd224a589bdf3b171af5488 | admin           |
| 034e4620ed3d45969dfe8992af001514 | member          |
| 0aa377a807df4149b0a8c69b9560b106 | ResellerAdmin   |
| 9369f2b754443f199c6d6b96479b1fa | heat_stack_user |
| cfeac5760d9c948e7b362abc1d06e557f | reader          |
| d5cb454559e44b47aaa8821df4e11af1 | swiftoperator   |
| ef3d3f510a474d6c860b4098ad658a29 | service         |
+----------------------------------+-----------------+
```

```
c.
```

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 785c70b150ee4c778e4de088070b4cf
```

```
e.
```

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.
```

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

1.3.2. 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 1.3.3, “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 list --domain LAB
   +------------------------------------------------------------------+---------------------+
   | ID                                                               | Name                |
   +------------------------------------------------------------------+---------------------+
   | 185277be62ae17e498a69f98a59b66934fb1d6b7f745f14f5f68953a665b8851 | grp-openstack       |
   | a8d17f19f464c4548c18b97e4aa331820f9d3be52654aa094e698a9182cb88 | grp-openstack-admin |
   | d971bb3bd5e64a454cb0cc7af4c0773e78d61b5f81321809f8323216938cae8 | grp-openstack-demo  |
   +------------------------------------------------------------------+---------------------+
   ```

2. Retrieve a list of roles:

   ```
   # openstack role list
   +----------------------------------+-----------------+
   | ID                               | Name            |
   +----------------------------------+-----------------+
   | 01d92614cd224a589bdf3b171a5f5488 | admin           |
   | 03fe4620ed3d45699e6892a001514    | member          |
   | 0aa377a87a7d4198ba8c695060106    | ResellerAdmin   |
   | 9369f2b754433f99c6d6b96479b1fa   | heat_stack_user |
   | cfe5760d9c9f87b326abc1d06e557f  | reader          |
   | d5cb45459e4f47aa8821df4e11af1    | swiftoperator   |
   | ef3bf010a474d6c860b4098ad658a29  | service         |
   +----------------------------------+-----------------+
   ```

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
   d971bb3bd5e64a454cb0cc7af4c0773e78d61b5f81321809f8323216938cae8 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**.

1.3.3. 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:

   # openstack user list --domain LAB

   +----------------------------------------+----------+
   | ID                                    | Name     |
   +----------------------------------------+----------+
   | 1f24ec1f11aeb90520079c29f70a060d22e2ce2b2eba7784c841ac418091e | user1    |
   | 12c062faddc5f8b065434df6f6f6f03eb9259537c93b41122458686e9a38bf12 | user2    |
   | afaf48031eb54c3e44e4cbb35f5b612084033ff70f63c22873d181fdae2e73c | user3    |
   | e47fc21dcf0d9716d2663766023e2d8dc15a6d9b01453854a898cabb2396826e | user4    |

2. Retrieve a list of roles:

   # openstack role list

   +------------------|----------+
   | ID               | Name     |
   +------------------|----------+
   | 01d92614cd22a589b8f3b171af95488 | admin    |
   | 034e4620ed3d45969df8992a0f01514 | member   |
   | 0aa377a807df4149b0a8c69b9560b106 | ResellerAdmin |
   | 9369d7f754443f199c66b96479b1f7 | heat_stack_user |
   | cfe5760d9c9487b362abc1d06e557f | reader   |
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:

```bash
# 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:

```bash
# 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:

![Dashboard Login](image)

**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.

### 1.4. 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:

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

2. Add the `admin` role in the `default` domain to the `admin` user:
As a result, the **admin** user can now see the **Domain** tab.

### 1.5. 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.

### 1.6. 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
```

<table>
<thead>
<tr>
<th>ID</th>
<th>Name</th>
<th>Enabled</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>6800b0496429431ab1c4efbb3fe810d4</td>
<td>LAB</td>
<td>True</td>
<td></td>
</tr>
<tr>
<td>default</td>
<td>Default</td>
<td>True</td>
<td>Owns users and projects available on Identity API v2.</td>
</tr>
</tbody>
</table>

### 1.7. 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
```

### 1.8. 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 project.

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 1.10, “Troubleshooting”.

1.9. 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
```

1.10. TROUBLESHOOTING

1.10.1. 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 `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 openldap-clients package. You can install this using `# dnf install openldap-clients`

### 1.10.2. 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.

### 1.10.3. 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
^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.

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:

<table>
<thead>
<tr>
<th>Source</th>
<th>Destination</th>
<th>Type</th>
<th>Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenStack Controller Node</td>
<td>Red Hat Identity Management</td>
<td>LDAPS</td>
<td>TCP 636</td>
</tr>
</tbody>
</table>

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:

```bash
# 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.

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

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

```bash
# 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:
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.

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 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 `ldapsearch`, 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/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 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
use_tls = False
query_scope = sub
chase_referrals = false
tls_cacertfile = /etc/pki/ca-trust/source/anchors/anchorsca.crt

[identity]
driver = ldap

Explanation of each setting:

<table>
<thead>
<tr>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>url</strong></td>
<td>The IdM server to use for authentication. Uses LDAPS port 636.</td>
</tr>
<tr>
<td><strong>user</strong></td>
<td>The account in IdM to use for LDAP queries.</td>
</tr>
<tr>
<td><strong>password</strong></td>
<td>The plaintext password of the IdM account used above.</td>
</tr>
<tr>
<td><strong>user_filter</strong></td>
<td>Filters the users presented to Identity Service. As a result, only members of the <code>grp-openstack</code> group can have permissions defined in Identity Service.</td>
</tr>
<tr>
<td><strong>user_tree_dn</strong></td>
<td>The path to the OpenStack accounts in IdM.</td>
</tr>
<tr>
<td><strong>user_objectclass</strong></td>
<td>Defines the type of LDAP user. For IdM, use the <code>inetUser</code> type.</td>
</tr>
<tr>
<td><strong>user_id_attribute</strong></td>
<td>Maps the IdM value to use for user IDs.</td>
</tr>
<tr>
<td><strong>user_name_attribute</strong></td>
<td>Maps the IdM value to use for names.</td>
</tr>
<tr>
<td><strong>user_mail_attribute</strong></td>
<td>Maps the IdM value to use for user email addresses.</td>
</tr>
<tr>
<td><strong>user_pass_attribute</strong></td>
<td>Leave this value blank.</td>
</tr>
</tbody>
</table>
NOTE

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:

```bash
# 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:

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

b. Get the ID value of the admin user:

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

c. Get the ID value of the admin role:

```bash
# openstack role list
+----------------------------------+---------------+
| ID                               | Name          |
+----------------------------------+---------------+
| 544d48aaffde48f1b3c31a52c35f01f9 | SwiftOperator |
| 6d005d783bf0436e882c55c62457d33d | ResellerAdmin |
| 785c70b150ee4e4778fe4de088070b4cf | 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:

```bash
$ openstack role add --domain 6800b0496429431ab1c4efbb3fe810d4 --user 3d75388d351846c6a880e53b2508172a 785c70b150ee4e4778fe4de088070b4cf
```
8. Restart the keystone service to apply the changes:

   ```
   $ sudo systemctl restart tripleo_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:

   ```
   $ openstack group list --domain LAB
   +--------------------------------+-------------+
   | ID                             | Name        |
   +--------------------------------+-------------+
   | 185277be62ae17e498a69f98a59b66934fb1d6b7f745f14f5f68953a665b8851 | grp-openstack |
   | a8d17f19f464c4548c18b97e4aa331820f9d3be52654aa8094e698a9182cbb88 | grp-openstack-admin |
   | d971bb3bd5e64a454cbd0cc7af4c0773e78d61b5f81321809f8323216938cae8 | grp-openstack-demo |
   +--------------------------------+-------------+
   ```
2. Retrieve a list of roles:

```bash
$ openstack role list
+----------------------------------+---------------+
| ID                               | Name          |
|----------------------------------+---------------+
| 0969957bce5e4f678ca6cef00e1abf8a | ResellerAdmin |
| 1fcb3c9b50aa46ee8196aaecc2b76b7 | admin         |
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_      |
| d3570730eb4b4780a7fed97eba197e1b | SwiftOperator |
```

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:

```bash
$ openstack role add --project demo --group
d971bb3bd5e64a454cb00cc7af4c0773e78d61b5f81321809f8323216938cae8 _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:

![Login Form](image)

### 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:

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

```bash
+------------------------------------------------------------------+
<table>
<thead>
<tr>
<th>Domain</th>
<th>User Name</th>
<th>Password</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAB</td>
<td>user1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
2. Retrieve a list of roles:

```
# openstack role list
+----------------------------------+---------------+
| ID                               | Name          |
+----------------------------------+---------------+
| 544d48aafde48f1b3c31a52c35f01f9  | SwiftOperator |
| 6d005d783b0436e882c55c62457d33d  | 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 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.

2.5. 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:

   $ openstack domain list
   +----------------------------------+---------+---------+-----------------------------------------------------------------+---+
   | ID                               | Name    | Enabled | Description                                                          |
   +----------------------------------+---------+---------+-----------------------------------------------------------------+---+
   | 6800b0496429431ab1c4efbb3fe810d4 | LAB     | True    | Owns users and projects available on Identity API v2. |
   | default                          | Default | True    | |
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` project.
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 ↔ Identity integration settings. See Section 2.8, “Troubleshooting”.

2.7. 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

2.8. TROUBLESHOOTING

2.8.1. 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 `# dnf install openldap-clients`.

2.8.2. 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
^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. Adding the undercloud to the CA

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

Procedure

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, and adjust 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

   Use the resulting One-Time Password (OTP) to enroll the undercloud.

3.1.2. Adding the undercloud to IdM

Register the undercloud with IdM and configure novajoin. Configure the following settings in the [DEFAULT] section of the `undercloud.conf` file.

Procedure

1. Enable the `novajoin` service:

   [DEFAULT]
   enable_novajoin = true

2. Set a One-Time Password (OTP) so that you can register the undercloud node with IdM:
3. Ensure that the overcloud domain name that the neutron DHCP server serves matches the IdM domain. For example, your kerberos realm in lowercase:

```plaintext
overcloud_domain_name = <domain>
```

4. Set the appropriate hostname for the undercloud:

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

5. Set IdM as the nameserver for the undercloud:

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

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

```plaintext
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:

```plaintext
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:

```bash
$ 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, you can use IdM as your DNS server:

Procedure

1. Connect to your undercloud:

```bash
$ source ~/stackrc
```

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

```bash
$ openstack subnet set ctlplane-subnet --dns-nameserver <idm_server_address>
```
3. Set the **DnsServers** parameter in an environment file to use your IdM server:

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

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

### 3.2.2. Configure the overcloud to use novajoin

**Procedure**

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

   ```bash
   $ 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:

```yaml
parameter_defaults:
  CloudDomain: lab.local
  CloudName: overcloud.lab.local
  CloudNameInternal: overcloud.internalapi.lab.local
  CloudNameStorage: overcloud.storage.lab.local
  CloudNameStorageManagement: overcloud.storagemgmt.lab.local
  CloudNameCtIplane: 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-everywhere-endpoints-dns.yaml`
   - `/home/stack/templates/custom-domain.yaml`

   For example:

   ```bash
   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
   ```

   The deployed overcloud nodes will be enrolled with IdM automatically.

4. This sets TLS only 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 you must modify to add your custom certificate and key):
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 must use the `/usr/share/openstack-tripleo-heat-templates/environments/services/haproxy-public-tls-certmonger.yaml` environment file:

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`:
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 overcloud --a-rec 172.17.4.101
```
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 TLS-e 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.

NOTE

If you are implementing TLS-e for an existing environment, it is still required to run commands such as `openstack undercloud install`, the `openstack overcloud deploy` commands. These are procedures are idempotent and will only adjust your existing deployment configuration to match updated templates and configuration files.

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

1. 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:

```bash
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

```bash
undercloud_nameservers = $IDM_SERVER_IP_ADDR
overcloud_domain_name = example.com
```

6. Deploy the undercloud:

```bash
openstack undercloud install
```

**Verification**

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

1. List the hosts in IdM:

   ```bash
   $ kinit admin
   $ ipa host-find
   ```

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

   ```bash
   ls /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.

**NOTE**

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

```yaml
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 value for the `IdMInstallClientPackages` parameter should be set to `false` unless you have preprovisioned nodes.

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 add and 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
  IdMServer: freeipa-0.redhat.local
  IdMDomain: redhat.local
  IdMInstallClientPackages: False

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:

```
openstack overcloud deploy \n-e ${DEFAULT_TEMPLATES}/environments/ssl/tls-everywhere-endpoints-dns.yaml \n-e ${DEFAULT_TEMPLATES}/environments/services/haproxy-public-tls-certmonger.yaml \n-e ${DEFAULT_TEMPLATES}/environments/ssl/enable-internal-tls.yaml \n-e ${CUSTOM_TEMPLATES}/tls-parameters.yaml \n... 
```

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 keyston::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_ldap_backend.yaml environment file:

```bash
$ 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:

```yaml
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:

```yaml
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.