Oracle Linux Automation Manager 2.3

Installation Guide

G32840-01
June 2025

Oracle Linux Automation Manager 2.3 Installation Guide,

G32840-01

Contents

Preface

Oracle Linux Automation Manager 2.3: Installation Guide describes how to install Oracle Linux Automation Manager in single-host deployments or clustered deployments.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

Oracle Linux Automation Manager Requirements

This chapter describes requirements for the systems to be used in an installation of Oracle Linux Automation Manager.

Oracle Linux Automation Manager Hardware Requirements

You can install Oracle Linux Automation Manager on a single machine or in a clustered setup in x86-64 Oracle Linux 8 or Oracle Linux 9 hosts.

Oracle Linux Automation Manager doesn't require specific hardware; however, certain operations are memory intensive and require a certain amount of disk space and CPU. A minimum configuration is:

- 4 GB RAM
- 40 GB disk space (170 GB is recommended)
- Two core CPU

These are minimum requirements to run Oracle Linux Automation Manager. You must decide any extra hardware requirements and capacity based on operational needs. For more information, see the upstream documentation.

2

Planning the Installation

This chapter provides information about planning an installation.

Oracle Linux Automation Manager Node Architecture

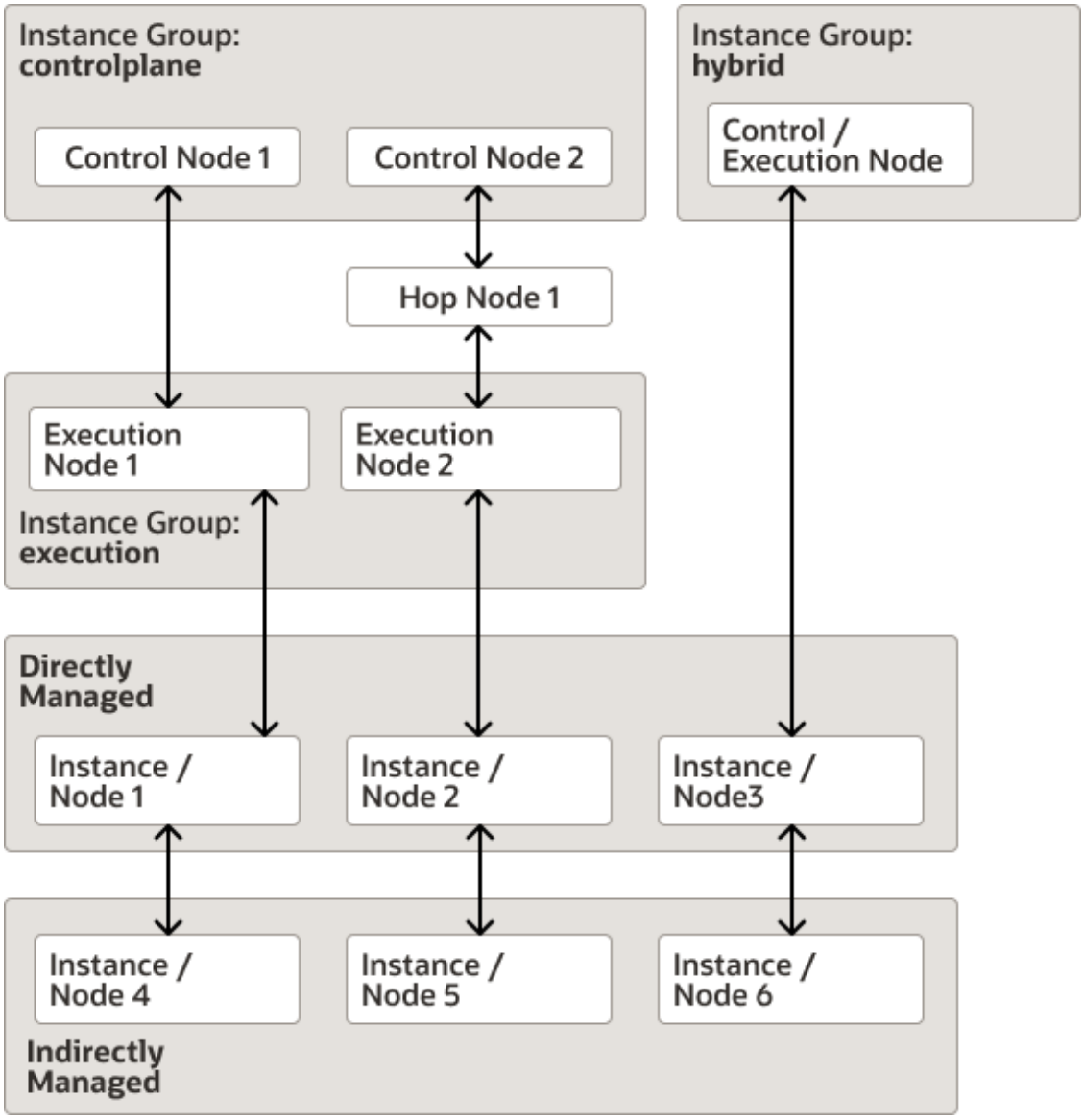
Oracle Linux Automation Manager works with a Service Mesh that provides a multiservice network that links nodes within a secure mesh. The Service Mesh can include up to 20 nodes that have the following node types:

- **Control Nodes**: These nodes provide management functions such as launching system jobs, inventory updates, and project synchronizations. Control nodes use ansible-runner which in turn uses Podman to run jobs within the **Control Plane Execution Environment**. The **Control Plane Execution Environment** references the `olam-ee` container image found on the Oracle Linux Container Registry. Control nodes don't run standard jobs.
- **Execution Nodes**: These nodes run standard jobs using ansible-runner which in turn uses Podman to run playbooks within **OLAM EE** execution environments. The **OLAM EE** execution environment references the `olam-ee` container image found on the Oracle Linux Container Registry. Execution nodes don't run management jobs. Execution nodes can also run custom execution environments that you can create using the Builder utility. For more information about custom execution environments, see Oracle Linux Automation Manager 2.3: Private Automation Hub Installation Guide. For more information about using Podman and the Oracle Linux Container Registry, see Oracle Linux: Podman User's Guide.
- **Hybrid Nodes**: Hybrid nodes combine the functions of both control nodes and execution nodes into one node. Hybrid nodes are supported in single host Oracle Linux Automation Manager deployments, but not in clustered multihost deployments.
- **Hop Nodes**: You can use hop nodes to connect control nodes to execution nodes within a cluster. Hop nodes can't run playbooks and don't appear in instance groups. However, they do appear as part of the service mesh.

Oracle Linux Automation Manager can manage various different instance types, such as devices, servers, databases, network equipment, and so on. In general, Oracle Linux Automation Manager manages the following instance types:

- **Directly Managed Instances**: Directly managed instances (nodes) are any virtual, physical, or software instances that Oracle Linux Automation Manager manages over an ssh connection.
- **Indirectly Managed Instances**: Indirectly managed instances (nodes) include any identifiable instance not directly connected to Oracle Linux Automation Manager, but managed by a device that's directly connected to Oracle Linux Automation Manager.

For example, the following image illustrates all node and instance types and some ways that they can be related to one another.



Installation Options

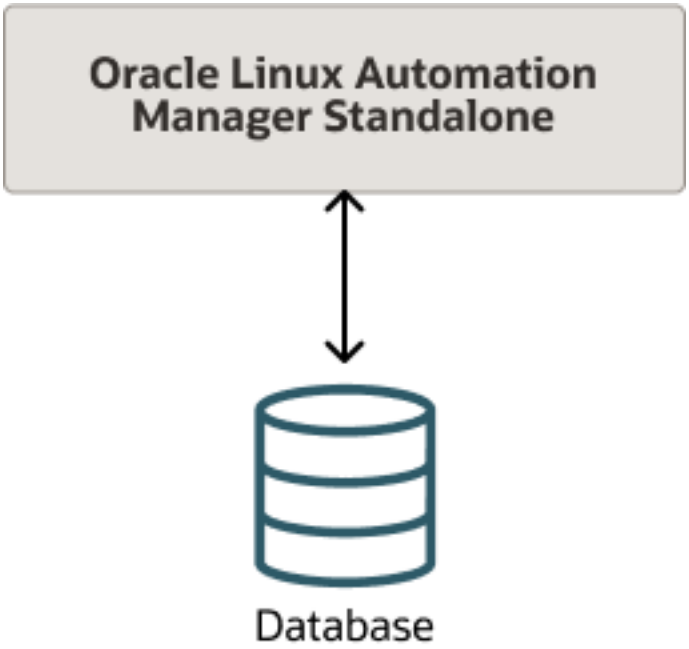
Oracle Linux Automation Manager provides three installation options:

- Standalone Installation: All components are on the same host, including the database.

Figure 2-1 Standalone Installation with Local Database

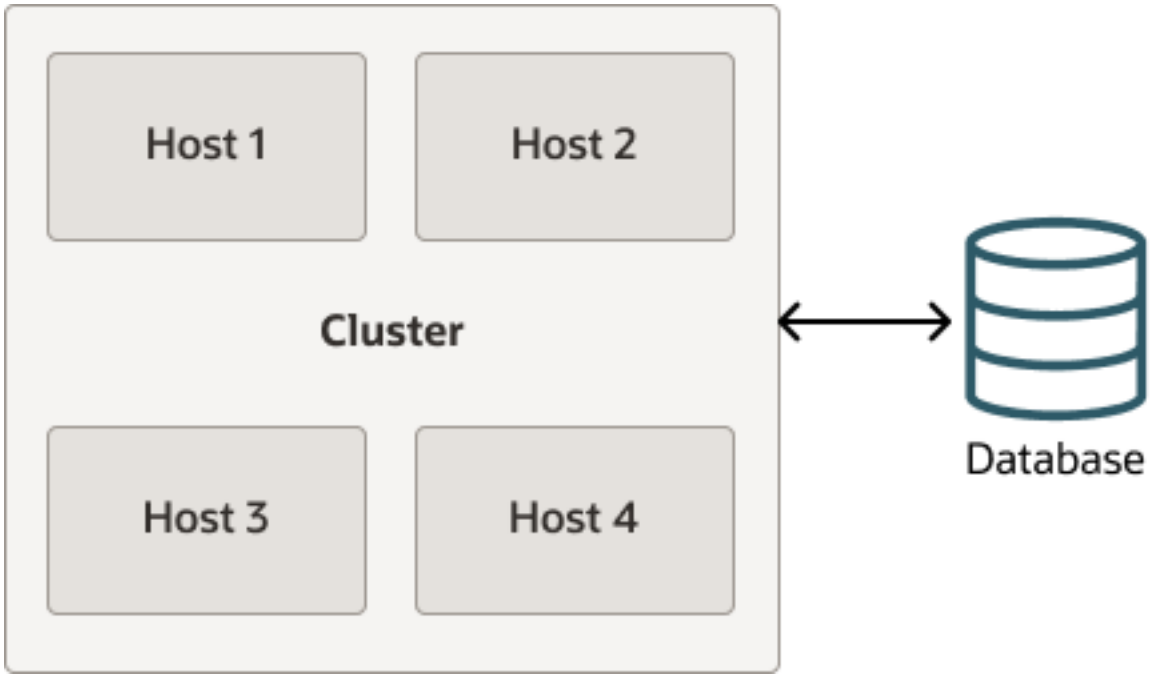
- Standalone Installation with Remote Database: All components are on the same host, except for the database which is on a remote host.

Figure 2-2 Standalone Installation with Remote Database



- Clustered Installation with Remote Database: Clustered installation can contain up to 20 nodes with one or more control node, one or more execution nodes, and one or more hop nodes all connected to one database. For example, the following shows a cluster with two control plane nodes and two execution plane nodes, each on separate hosts, and all connected to a remote database.

Figure 2-3 Clustered Installation with Remote Database

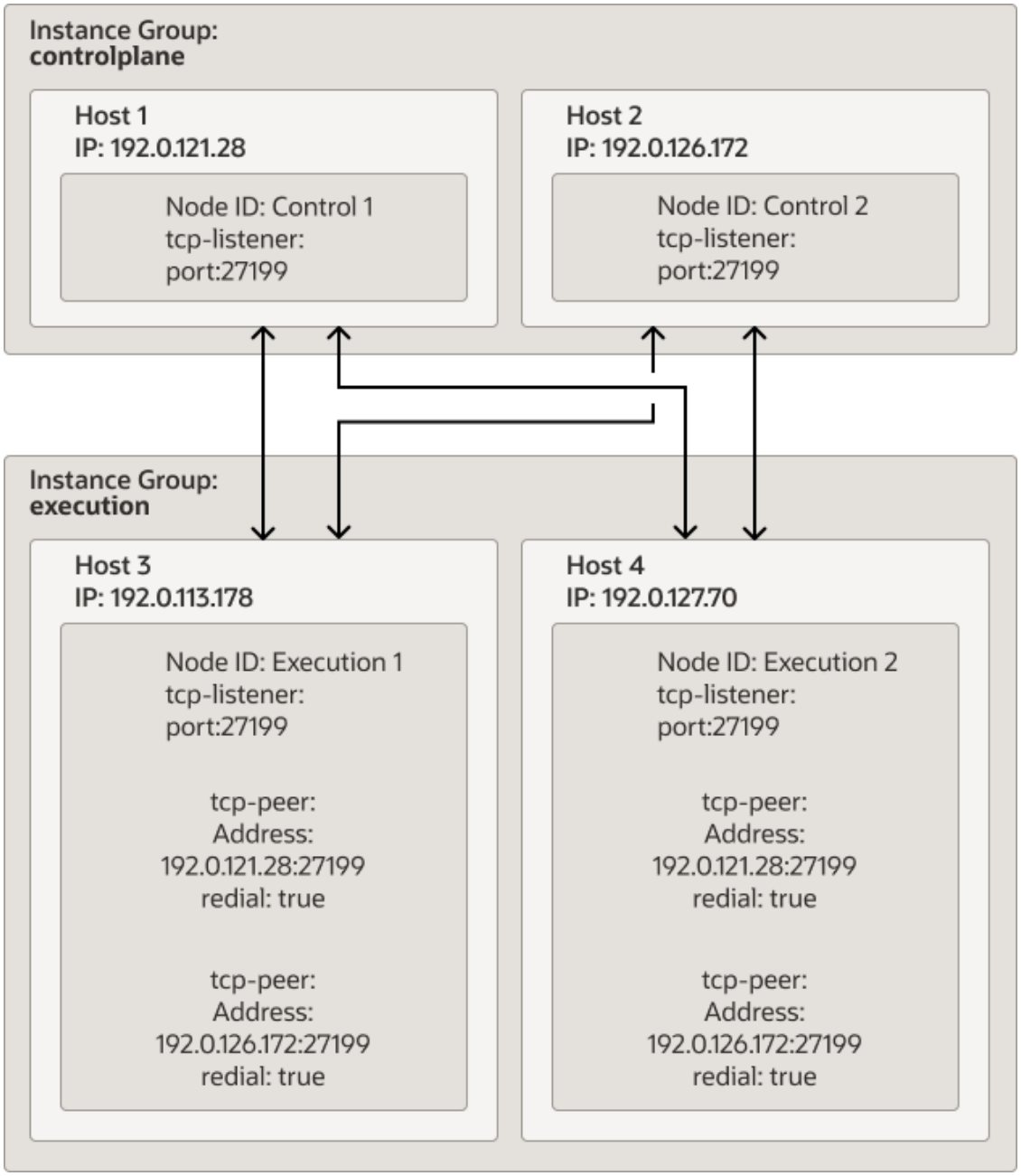


Service Mesh Topology Examples

You can configure the Oracle Linux Automation Manager Service Mesh topology in various ways.

Example 1: Design the Service Mesh such that you have at least one backup control plane node and one backup execution plane node (for example, two control and two execution nodes). Each execution plane node would have communication with both control plane nodes in case one of the control plane node were to fail. If the first execution plane node were to fail, the control plane node would switch to the second execution plane node.

Figure 2-4 Clustered Installation with Remote Database



The high level steps to configure the Service Mesh are as follows:

1. Configure the `/etc/receptor/receptor.conf` file with the Node ID, tcp-listener, and tcp-peer addresses as required for each node. For more information about this task, see Configuring and Starting the Control Plane Service Mesh and Configuring and Starting the Execution Plane Service Mesh.
2. From a control plane node, log in as the awx user, and run the awx-manage command to do the following:

a. Provision each host's IP address or host name, and designate it as a control plane or execution plane node type. For example, the following commands provision two control plane and two execution plane nodes as illustrated in the figure above:

```
awx-manage provision_instance --hostname=192.0.121.28 --
node_type=control
awx-manage provision_instance --hostname=192.0.126.72 --
node_type=control
awx-manage provision_instance --hostname=192.0.113.178 --
node_type=execution
awx-manage provision_instance --hostname=192.0.127.70 --
node_type=execution
```

b. Register each node to either the `controlplane` or the `execution` instance group, based on the type of node you designated for each node. The awx-manage command refers to instance groups as queuenames. For example, the following commands create the controlplane and execution instance groups and associates the two control plane and two execution plane nodes to each instance group as illustrated in the figure above:

```
awx-manage register_queue --queuename=controlplane --
hostnames=192.0.121.28
awx-manage register_queue --queuename=controlplane --
hostnames=192.0.126.72
awx-manage register_queue --queuename=execution --
hostnames=192.0.113.178
awx-manage register_queue --queuename=execution --hostnames=192.0.127.70
```

c. Add the receptor address record for the `controlplane` and the `execution` nodes. For example, the following commands adds the controlplane and execution node addresses as illustrated in the figure above:

```
awx-manage add_receptor_address --instance=192.0.121.28 --
address=192.0.121.28 --port=27199 --canonical
awx-manage add_receptor_address --instance=192.0.126.72 --
address=192.0.126.72 --port=27199 --canonical
awx-manage add_receptor_address --instance=192.0.113.178 --
address=192.0.113.178 --port=27199 --canonical
awx-manage add_receptor_address --instance=192.0.127.70 --
address=192.0.127.70 --port=27199 --canonical
```

d. Register the peer relationship between each node. Note that when you register a peer relationship between a source IP address to a target IP address, the peer relationship establishes bidirectional communication. For example, the following commands registers the host IP address of the execution nodes as the source and each tcp-peer connection are the targets, which are the control plane nodes:

```
awx-manage register_peers 192.0.113.178 --peers 192.0.121.28
awx-manage register_peers 192.0.113.178 --peers 192.0.126.172
awx-manage register_peers 192.0.127.70 --peers 192.0.121.28
awx-manage register_peers 192.0.127.70 --peers 192.0.126.172
```

The command must be run for each link you want to establish between nodes.

e. Register each instance group as the default queuename for either the control plane or the execution plane. This ensures that only control type jobs go to the control plane instance group and only Oracle Linux Automation Engine jobs go to execution plane instance group. To do this, you must create a custom settings file in `/etc/tower/conf.d/filename.py` and add the following parameters:

```
DEFAULT_EXECUTION_QUEUE_NAME = 'execution'
DEFAULT_CONTROL_PLANE_QUEUE_NAME = 'controlplane'
```

For more information about these steps, see Configuring the Control, Execution, and Hop Nodes.

Example 2: Deploy as many control and execution plane nodes as you require such that you build in fail over in case any control or execution plane node fails. Ensure you don't exceed the 20 node limit for the cluster. Some options you can consider are:

- Sometimes you might have an execution node that can't be directly connected to a control plane node. In such cases you can connect the execution node to another execution node that's connected to the control node. This does introduce a risk such that if the intermediate execution node were to fail, then the connected execution node would become inaccessible to the control node.
- Sometimes you might have an execution node that can't be directly connected to a control plane node. In such cases you can connect the execution node to a hop node that's connected to the control node. This does introduce a risk such that if the intermediate hop node were to fail, then the connected execution node would become inaccessible to the control node.
- Establishing a peer relationship between control plane nodes. This ensures that control plane nodes are always directly accessible to one another. If no such relationship is established, then control plane nodes are aware of each other through connected execution plane nodes. For example, control A connects to control B through execution A which is connected to both.

Tuning Instances for Playbook Duration

Oracle Linux Automation Manager monitors jobs for status changes. For example, some job statuses are Running, Successful, Failed, Waiting, and so on. Normally the playbook triggers status changes as it runs. However, sometimes, the playbook gets stuck in the Running or Waiting state. When this happens, a reaper process automatically changes the state of the task from Running or Waiting to Failed. The default timer for when the reaper changes the status of a stuck job to the Failed state is 60 seconds.

If you have jobs that are designed to run longer than 60 seconds, then create or edit a custom settings file in `/etc/tower/conf.d/filename.py` and set the **REAPER_TIMEOUT_SEC** parameter. Specify a time in seconds that is longer than the duration that your playbooks with the longest duration is expected to run. This avoids scenarios where the reaper mistakenly sets a long running playbook to the Failed state because the **REAPER_TIMEOUT_SEC** value has expired.

A possible scenario could occur if you run many short and long duration playbooks with a reaper that has a long timeout value. If one or more of the short duration playbooks run for longer than expected, (for example, because of a network outage making it impossible for these playbooks to complete) the reaper continues to track the status of the stuck short duration playbooks until they either get unstuck and transition to the Successful state or until the reaper timeout value is reached. This scenario should cause no performance difficulties if only a few such failures were to occur. However, if hundreds of such failures were to occur at

the same time, Oracle Linux Automation Manager would waste resources on tracking these stuck jobs and could degrade the performance of the host processing the jobs.

For more information about setting the **REAPER_TIMEOUT_SEC** parameter, see Setting up Hosts.

3
Preparing the Database and Hosts

The following chapter provides information about setting up the network firewalls, database, and hosts for an Oracle Linux Automation Manager installation. This chapter also discusses how to enable the repositories to install the Oracle Linux Automation Manager packages.

Setting Up the Network

This section contains information about the generic networking requirements for Oracle Linux Automation Manager hosts, and the database host. An example of how to set up the network to enable the communication between the Oracle Linux Automation Manager host and the inventory hosts in an environment is also included.

Setting Up the Firewall Rules

Oracle Linux 8 and 9 installs and enables `firewalld`, by default. Example commands to open the ports and to set up the firewall rules are provided below.

On the Oracle Linux Automation Manager hosts, run the following firewalld commands:

```
sudo firewall-cmd --add-port=27199/tcp --permanent
sudo firewall-cmd --add-service=http --permanent
sudo firewall-cmd --add-service=https --permanent
sudo firewall-cmd --reload
```

> **Note:**
>
> Port 27199 provides a TCP listener port for the Oracle Linux Automation Manager service mesh and must be open on each node in the mesh. The HTTP and HTTPS ports are for the NGINX server.

If you choose to install a remote database, open the following port on the host running the database:

```
sudo firewall-cmd --add-port=5432/tcp --permanent
sudo firewall-cmd --reload
```

Enabling Access to the Oracle Linux Automation Manager Packages

This section contains information on setting up the locations for the OS on which you want to install the Oracle Linux Automation Manager software packages.

Enabling Channels with ULN

If you're registered to use ULN, use the ULN web interface to subscribe the system to the appropriate channels.

To subscribe to the ULN channels:

1. Log in to https://linux.oracle.com with your ULN username and password.
2. On the Systems tab, click the link named for the system in the list of registered machines.
3. On the System Details page, click **Manage Subscriptions**.
4. On the System Summary page, select each required channel from the list of available channels and click the right arrow to move the channel to the list of subscribed channels. Subscribe the system to the following channels:
   - For Oracle Linux 8 instances, subscribe to the following:
     - `ol8_x86_64_automation2.3`
     - `ol8_x86_64_addons`
     - `ol8_x86_64_baseos_latest`
     - `ol8_x86_64_UEKR6` or `ol8_x86_64_UEKR7`
     - `ol8_x86_64_appstream`
   - For Oracle Linux 9 instances, subscribe to the following:
     - `ol9_x86_64_automation2.3`
     - `ol9_x86_64_addons`
     - `ol9_x86_64_baseos_latest`
     - `ol9_x86_64_UEKR7` or `ol9_x86_64_UEKR8`
     - `ol9_x86_64_appstream`
5. Click **Save Subscriptions**.

Enabling Repositories with the Oracle Linux Yum Server

If you're using the Oracle Linux yum server for system updates, enable the required yum repositories.

To enable the yum repositories:

1. Do one of the following:
   - For Oracle Linux 8, use the `dnf config-manager` tool to enable the `ol8_baseos_latest` repository.

     ```
     sudo dnf config-manager --enable ol8_baseos_latest
     ```
   - For Oracle Linux 9, use the `dnf config-manager` tool to enable the `ol9_baseos_latest` repository.

     ```
     sudo dnf config-manager --enable ol9_baseos_latest
     ```
2. Do one of the following:
   - For Oracle Linux 8, install `oraclelinux-automation-manager-release-el8`:

     ```
     sudo dnf install oraclelinux-automation-manager-release-el8-2.3
     ```
   - For Oracle Linux 9, install `oraclelinux-automation-manager-release-el9`:

     ```
     sudo dnf install oraclelinux-automation-manager-release-el9-2.3
     ```
3. Enable the following yum repositories including the Oracle Linux Automation Manager release 2 repositories:

- For Oracle Linux 8, enable the following:
  - `ol8_addons`
  - `ol8_UEKR6` or `ol8_UEKR7`
  - `ol8_appstream`

  Use the `dnf config-manager` tool to enable the yum repositories and do one of the following:
  - If you're using ol8_UEK6, use the following command:

    ```
    sudo dnf config-manager --enable ol8_addons ol8_UEKR6 ol8_appstream
    ```

  - If you're using ol8_UEK7, use the following command:

    ```
    sudo dnf config-manager --enable ol8_addons ol8_UEKR7 ol8_appstream
    ```

- For Oracle Linux 9, enable the following:
  - `ol9_addons`
  - `ol9_UEKR7` or `ol9_UEKR8`
  - `ol9_appstream`

  Use the `dnf config-manager` tool to enable the yum repositories and do one of the following:
  - If you're using ol9_UEK7, use the following command:

    ```
    sudo dnf config-manager --enable ol9_addons ol9_UEKR7 ol9_appstream
    ```

  - If you're using ol9_UEK8, use the following command:

    ```
    sudo dnf config-manager --enable ol9_addons ol9_UEKR8 ol9_appstream
    ```

Setting Up a Local or Remote Database

To setup a local or remote Postgresql database instance on Oracle Linux 8 or 9 for Oracle Linux Automation Manager single host or multihost configurations, do the following:

1. Install and configure Oracle Linux 8 or 9 on a host.
2. If the database is remote, open the database port in the firewall as described in Setting Up the Firewall Rules.
3. Enable the `postgresql 16` module stream.

   ```
   sudo dnf module reset postgresql
   sudo dnf module enable postgresql:16
   ```

> **Note:**
>
> For more information about the Postgresql 16 life cycle, see the appendix discussing the application life cycle for stream modules in Oracle Linux: Managing Software on Oracle Linux.

4. Install the database.

```
sudo dnf install postgresql-server
```

5. Initialize the database:

```
sudo postgresql-setup --initdb
```

6. In the `/var/lib/pgsql/data/postgresql.conf` file, switch the password storage mechanism from `md5` to `scram-sha-256`. For example, the following command makes the switch for you:

```
sudo sed -i "s/#password_encryption.*/password_encryption = scram-
sha-256/"  /var/lib/pgsql/data/postgresql.conf
```

7. Start the database using the following command that also ensures that the database restarts in case the host restarts:

```
sudo systemctl enable --now postgresql
```

8. Ensure the database is running:

```
sudo systemctl status postgresql
```

9. Create the database user accounts. For example:

```
sudo su - postgres -c "createuser -S -P awx"
```

10. Enter and confirm the password for the awx user.

```
Enter password for new role:
Enter it again:
```

11. Create the database.

```
sudo su - postgres -c "createdb -O awx awx"
```

12. As the root user, in the `/var/lib/pgsql/data/pg_hba.conf` file add the following line:

```
host  all  all 0.0.0.0/0 scram-sha-256
```

13. As the root user, in the `/var/lib/pgsql/data/postgresql.conf` file in the `# CONNECTIONS AND AUTHENTICATION` section, a line with the text `listen_addresses =` followed by the IP address or host name of the database in single quotes. For example:

```
listen_addresses = '<IP address or host name>'
```

```
#listen_addresses = 'localhost'         # what IP address(es) to listen on;
                                        # comma-separated list of
addresses;
                                        # defaults to 'localhost'; use '*'
for all
                                        # (change requires restart)
#port = 5432                            # (change requires restart)
```

In the previous example, <*IP address or hostname*> is the IP address or host name of the database.

14. Calculate and update the memory requirements parameters using the following:

```
max_connections = 1024
shared_buffers = total_mem_mb*0.3
work_mem = total_mem_mb*0.03
maintenance_work_mem = total_mem_mb*0.04
```

In the previous example, *total_mem_mb* is the total memory size in megabytes of the system hosting the database server. For example, if the total available memory on the system were 18 000 MB, then this worksheet would include the following:

```
max_connections = 1024
shared_buffers = 18000*0.3
work_mem = 18000*0.03
maintenance_work_mem = 18000*0.04
```

The final numbers to add are as follows:

```
max_connections = 1024
shared_buffers = 5400MB
work_mem = 540MB
maintenance_work_mem = 720MB
```

15. Add the calculated values to the `/var/lib/pgsql/data/postgresql.conf` file.
16. Restart the database.

```
sudo systemctl restart postgresql
```

17. You're now ready to set up hosts as described in Setting up Hosts.

Setting up Hosts

This section provides information for setting up one or more hosts intended to run Oracle Linux Automation Manager in any of the configurations listed in Installation Options.

To set up one or more hosts:

1. If you haven't already done so, enable the `postgresql 16` module stream.

```
sudo dnf module reset postgresql
sudo dnf module enable postgresql:16
```

> **Note:**
>
> For more information about the Postgresql 16 life cycle, see the appendix discussing the application life cycle for stream modules in Oracle Linux: Managing Software on Oracle Linux.

2. Install Oracle Linux Automation Manager.

   ```
   sudo dnf install ol-automation-manager
   ```

3. If you're creating a cluster, choose the `/etc/tower/SECRET_KEY` from one node and replace the value of the `/etc/tower/SECRET_KEY` on all other nodes with the value from the chosen node. Ensure the file user and group ownership is `awx:awx` on all nodes. All nodes must have the same value in their `/etc/tower/SECRET_KEY` file.
4. Edit the `/etc/redis.conf` file on Oracle Linux 8 or the `/etc/redis/redis.conf` file on Oracle Linux 9 to include the following lines:

   ```
   unixsocket /var/run/redis/redis.sock
   unixsocketperm 775
   ```

5. Create a custom settings file in the `/etc/tower/conf.d` folder. For example:

   ```
   sudo touch /etc/tower/conf.d/olam.py
   ```

6. Configure the `CLUSTER_HOST_ID` field.

   ```
   CLUSTER_HOST_ID = "hostname or ip address"
   ```

   In the previous example, *hostname or ip address* is the hostname or IP address of the system running Oracle Linux Automation Manager. If hostname is used, the host must be resolvable.

   > **Note:**
   >
   > The settings you specify in custom settings files supersede the settings in `/etc/tower/settings.py`, preventing software updates from changing the custom settings.

7. Add the following `DATABASE` fields to the custom settings file:

   ```
   DATABASES = {
       'default': {
           'ATOMIC_REQUESTS': True,
           'ENGINE': 'awx.main.db.profiled_pg',
           'NAME': 'awx',
           'USER': 'awx',
           'PASSWORD': 'password',
           'HOST': 'database hostname or ip address',
           'PORT': '5432',
       }
   }
   ```

   In the previous example, *database hostname or ip address* is the hostname or IP address of the local or remote database. If hostname is used, the host must be resolvable. *password* is the password for your database, if you have configured one.

8. If you have playbooks designed to run longer than the default reaper timeout of 60 seconds, add the `REAPER_TIMEOUT_SEC` parameter to the custom settings file to increase the timeout. For example,

   ```
   REAPER_TIMEOUT_SEC=<longest_playbook_time>
   ```

   In the previous example, <*longest_playbook_time*> is number of seconds that exceeds the duration of the longest playbook runtime.

9. Set the ownership and file permissions for the custom settings file.

   ```
   sudo chown awx:awx /etc/tower/conf.d/filename.py
   sudo chmod 0640 /etc/tower/conf.d/filename.py
   ```

   In the previous example, filename is the name of the custom settings file. For example,

   ```
   sudo chown awx:awx /etc/tower/conf.d/olam.py
   sudo chmod 0640 /etc/tower/conf.d/olam.py
   ```

10. Run the following commands on all hosts:

    ```
    sudo su -l awx -s /bin/bash
    podman system migrate
    podman pull container-registry.oracle.com/oracle_linux_automation_manager/olam-
    ee:2.3-ol8
    exit
    ```

> **Note:**
>
> After you finish installing Oracle Linux Automation Manager, you can configure whether you want the Execution Environments to always pull the 2.3 `olam-ee` container image when running playbooks, or use some other option or custom image. For more information about these options, see Oracle Linux Automation Manager 2.3: User's Guide. For more information about Private Automation Hub, see Oracle Linux Automation Manager 2.3: Private Automation Hub User's Guide .

> **Note:**
>
> The previous command assumes that you are pulling the olam-ee image directly from the Oracle Container Registry. If you are using Private Automation Hub or have set up a custom container registry, you can pull the image from there instead. In addition, you can configure Oracle Linux Automation Manager to always pull from that container registry by changing the container registry path in a custom settings file (`/etc/tower/conf.d/filename.py`). Add the following fields to the custom settings file, then replace the Oracle Container Registry path with your custom container registry path:
>
> ```
> GLOBAL_JOB_EXECUTION_ENVIRONMENTS = [{'name': 'OLAM EE (2.3)',
> 'image': 'container-registry.oracle.com/
> oracle_linux_automation_manager/olam-ee:2.3-ol8'}]
> CONTROL_PLANE_EXECUTION_ENVIRONMENT = 'container-
> registry.oracle.com/oracle_linux_automation_manager/olam-ee:2.3-
> ol8'
> ```

11. Run the following commands on one control host (in a clustered deployment) or on the single host (in single host deployment):

    ```
    sudo su -l awx -s /bin/bash
    awx-manage migrate
    awx-manage createsuperuser --username admin --email email
    ```

    In the previous example, *email* is the email address of the admin user.

12. Enter and repeat the password for the admin user.

    ```
    Password:
    Password (again):
    ```

13. Exit the awx user.

    ```
    exit
    ```

14. On all hosts, generate SSL certificates for NGINX:

> **Note:**
>
> The following instruction explains how to create a self-signed certificate for use by NGINX as part of Oracle Linux Automation Manager. We recommend that on production systems you use CA signed certificates for this purpose. For more information on working with SSL certificates, see Oracle Linux: Managing Certificates and Public Key Infrastructure.

    ```
    sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout /etc/tower/
    tower.key -out /etc/tower/tower.crt
    ```

15. Remove any default configuration for NGINX. Edit `/etc/nginx/nginx.conf` to contain the following configuration:

    ```
    user nginx;
    worker_processes auto;
    ```

```
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;

# Load dynamic modules. See /usr/share/doc/nginx/README.dynamic.
include /usr/share/nginx/modules/*.conf;

events {
    worker_connections 1024;
}

http {
    log_format  main  '$remote_addr - $remote_user [$time_local] "$request" '
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile            on;
    tcp_nopush          on;
    tcp_nodelay         on;
    keepalive_timeout   65;
    types_hash_max_size 2048;

    include             /etc/nginx/mime.types;
    default_type        application/octet-stream;

    # Load modular configuration files from the /etc/nginx/conf.d directory.
    # See http://nginx.org/en/docs/ngx_core_module.html#include
    # for more information.
    include /etc/nginx/conf.d/*.conf;
}
```

> **Note:**
>
> For advanced NGINX users, the Oracle Linux Automation Manager NGINX configuration file is located in `/etc/nginx/conf.d/ol-automation-manager-nginx.conf`. For example, you might use a different version of TLS or have different ciphers configured. If you have an existing customized NGINX setup, ensure that you also apply the `ol-automation-manager-nginx.conf` settings.

16. You're now ready to install Oracle Linux Automation Manager in a cluster or on a single host. For more information, see Installing Oracle Linux Automation Manager on a Single-Host Deployment and Installing Oracle Linux Automation Manager in a Clustered Deployment.

4

Installing Oracle Linux Automation Manager on a Single-Host Deployment

This chapter shows you how to set up a host and install the Oracle Linux Automation Manager software and includes an option for using a remote or local database.

Installing on a Single Host

This section provides instructions for installing the Oracle Linux Automation Manager on a single host where the database is local or on a remote host.

To set up the host:

1. On the Oracle Linux Automation Manager host, run the following commands:

   ```
   sudo su -l awx -s /bin/bash
   ```

2. Enter the following command:

   ```
   awx-manage provision_instance --hostname=<hostname or IP address> --node_type=hybrid
   ```

   In the previous example, *hostname or IP address* is the hostname or IP address of the system running Oracle Linux Automation Manager. If hostname is used, the host must be resolvable.

3. Run the following command to register the default execution environments, which are:
   - Control Plane Execution Environment
   - OLAM EE: (2.3)

   ```
   awx-manage register_default_execution_environments
   ```

4. Run the following command to create the default queue for standard jobs that run playbooks:

   ```
   awx-manage register_queue --queuename=default --hostnames=<hostname or IP address>
   ```

5. Run the following command to create the controlplane queue for Oracle Linux Automation Manager management type jobs.

   ```
   awx-manage register_queue --queuename=controlplane --hostnames=<hostname
   or IP address>
   ```

6. Exit the awx shell environment.

   ```
   exit
   ```

7. Remove any default configuration for Receptor. Edit `/etc/receptor/receptor.conf` to contain the following configuration:

   ```
   ---
   - node:
       id: <IP address>
   ```

```
- log-level: debug

- tcp-listener:
    port: port_number

- control-service:
    service: control
    filename: /var/run/receptor/receptor.sock

- work-command:
    worktype: local
    command: /var/lib/ol-automation-manager/venv/awx/bin/ansible-runner
    params: worker
    allowruntimeparams: true
#    verifysignature: true
```

In the previous example, *hostname or IP address* is the IP address of the host and *port_number* is the port number that this node is listening on. For example, you can call the host single1-192.0.121.30 where you provide a node name and the IP address of the node. And you could configure the tcp-listener list listen on port 27199.

8. Start the service:

```
sudo systemctl enable --now ol-automation-manager.service
```

9. Run the following command to preload data, such as:
   - Demo Project
   - Default Galaxy Credentials
   - Demo Organization
   - Demo Inventory
   - Demo Job template
   - And so on

```
sudo su -l awx -s /bin/bash
awx-manage create_preload_data
```

10. Exit the awx shell environment.

```
exit
```

11. The host is now ready. Using a browser, you can now sign in as the admin user.

```
https://<hostname or IP address>
```

5

Installing Oracle Linux Automation Manager in a Clustered Deployment

This chapter discusses how to prepare hosts in an Oracle Linux Automation Manager multihost deployment. When you prepare the hosts, you must install the Oracle Linux Automation Manager software packages and configure them as part of the Oracle Linux Automation Manager service mesh. Configure and start the Service Mesh nodes before configuring and starting the control plane and execution plane nodes.

Configuring and Starting the Control Plane Service Mesh

You configure each node in the control plane of a cluster by editing the `/etc/receptor/receptor.conf` file. This file contains the following elements:

- **node ID**: The node ID must be the IP address or host name of the host.
- **log-level**: Available options are: **Error**, **Warning**, **Info** and **Debug**. Log level options provide increasing verbosity, such that Error generates the least information and Debug generates the most.
- **tcp-listener port**: This is the port that the node listens for incoming tcp peer connections configured on other nodes. For example, if the node ID represents a control node that listens on port 27199, then all other nodes that want to establish a connection to this control node would require that port 27199 is specified in the tcp-peer element in the `/etc/receptor/receptor.conf` file.
- **control-service**: All nodes in a cluster run the control service which reports status and launches and monitors work.
- **work-command**: This element defines the type of work that can be done on a node. For control plane nodes, the work type is always Local. The command it runs is the Ansible Runner tool which provides an abstraction layer for running Ansible and Ansible playbook tasks and can be configured to send status and event data to other systems. For more information about Ansible Runner, see https://ansible-runner.readthedocs.io/en/stable/.

On each host intended for use as a control plane node, do the following:

1. Remove any default configuration for Receptor and edit `/etc/receptor/receptor.conf` to contain the following configuration control plane specific information:

```
---
- node:
    id: <IP address or host name>

- log-level: info

- tcp-listener:
    port: <port_number>

- control-service:
    service: control
    filename: /var/run/receptor/receptor.sock
```

```
- work-command:
    worktype: local
    command: /var/lib/ol-automation-manager/venv/awx/bin/ansible-runner
    params: worker
    allowruntimeparams: true
    verifysignature: false
```

In the previous example, *IP address or hostname* is the IP address or hostname of the node and *port_number* is the port number that this node is listening on. For example, you can use something such as control1-192.0.121.28 where you provide a node name and the IP address of the node. And you could configure the tcp-listener list listen on port 27199. The worktype parameter must be `local` in control plane nodes.

2. Start the Oracle Linux Automation Manager mesh service.

```
sudo systemctl start receptor-awx
```

3. Verify the Service Mesh. For more information, see Viewing Service Mesh Status for a Cluster Node.

> **Note:**
>
> At this point in the process, the peer relationships between service mesh nodes haven't been established yet. Status information only exists for the individual servers running the Service Mesh.

Configuring and Starting the Execution Plane Service Mesh

You configure each node in the execution plane of a cluster by editing the `/etc/receptor/receptor.conf` file. This file contains the following elements:

- **node ID**: The node ID must be the IP address or hostname of the host.
- **log-level**: Available options are: **Error**, **Warning**, **Info** and **Debug**. Log level options provide increasing verbosity, such that Error generates the least information and Debug generates the most.
- **tcp-listener port**: This is the port that the node listens for incoming tcp peer connections configured on other nodes. For example, if the node ID represents an execution node that listens on port 27199, then all other nodes that want to establish a connection to this execution node would require that port 27199 is specified in the tcp-peer element of the `/etc/receptor/receptor.conf` file.
- **tcp-peer**: This element must include the hostname and port number of the host it's connecting with. For example, if this execution node needs to connect to more than one control plane node to provide redundancy, you would need to add tcp-peer elements for each control plane node that the execution node connects with. In the address field, enter the host name or IP address of the control plane node, followed by the port number. The redial element, if enabled, tries to periodically reestablish a connection to the host if connectivity fails.
  You can also configure tcp-peer elements to include the hostnames and port numbers of other execution nodes or hop nodes based on the service mesh topology requirements.
- **control-service**: All nodes in a cluster run the control service which reports status and launches and monitors work.

- **work-command**: This element defines the type of work that can be done on a node. For execution plane nodes, the work type is always `ansible-runner`. The command it runs is the Ansible Runner tool which provides an abstraction layer for running Ansible and Ansible playbook tasks and can be configured to send status and event data to other systems. For more information about Ansible Runner, see https://ansible-runner.readthedocs.io/en/stable/.

On each host intended for use as an execution plane node, do the following:

1. Remove any default configuration for Receptor and edit `/etc/receptor/receptor.conf` to contain the following configuration execution plane specific information:

```
---
- node:
    id: <IP address or hostname>

- log-level: debug

- tcp-listener:
    port: <port_number>

- tcp-peer:
    address: <hostname or IP address>:<target_port_number>
    redial: true

- tcp-peer:
    address: <hostname or IP address>:<target_port_number>
    redial: true

- control-service:
    service: control
    filename: /var/run/receptor/receptor.sock

- work-command:
    worktype: ansible-runner
    command: /var/lib/ol-automation-manager/venv/awx/bin/ansible-runner
    params: worker
    allowruntimeparams: true
    verifysignature: false
```

   In the previous example,

   - *IP address or hostname* is the IP address or hostname of the node.
   - *port_number* is the port number that this node is listening on.
   - *target_port* is the port number of the peer node that you're configuring this node to connect with.
   - *hostname or IP address* is the hostname or IP address of the execution, control, or hop node being connected with.
   - The worktype parameter must be `ansible-runner` in execution plane nodes.

   If the execution environment is associated with more than one control, execution, or hop node, enter other `- tcp-peer:` nodes for instances that the execution host is associated with.

2. Start the Oracle Linux Automation Manager mesh service.

```
sudo systemctl start receptor-awx
```

3. Verify the Service Mesh. For more information, see Viewing Service Mesh Status for a Cluster Node.

> **Note:**
>
> At this point in the process, the peer relationships between service mesh nodes haven't been established yet. Status information only exists for the individual servers running the Service Mesh.

Configuring and Starting the Hop Nodes

You configure each hop node in the cluster by editing the `/etc/receptor/receptor.conf` file. This file contains the following elements:

- **node ID**: The node ID must be the IP address or hostname of the host.
- **log-level**: Available options are: **Error**, **Warning**, **Info** and **Debug**. Log level options provide increasing verbosity, such that Error generates the least information and Debug generates the most.
- **tcp-listener port**: This is the port that the node listens for incoming tcp peer connections configured on other nodes. For example, if the node ID represents an execution node that listens on port 27199, then all other nodes that want to establish a connection to this execution node would require that port 27199 is specified in the tcp-peer element in the `/etc/receptor/receptor.conf` file.
- **tcp-peer**: This element must include the hostname and port number of the host it is connecting with. For example, you might configure a hop node to connect to a control node as the intermediate node between the control node and an execution node. In the address field, enter the host name or IP address of the control plane node, followed by the port number. The redial element, if enabled, tries to periodically reestablish a connection to the host if connectivity fails.
- **control-service**: All nodes in a cluster run the control service which reports status and launches and monitors work.
- **work-command**: This element defines the type of work that can be done on a node. Hop nodes don't run playbooks. However, you must configure the default fields. The work type for hop nodes is always `ansible-runner`.

On each host intended for use as a hop node, do the following:

1. Remove any default configuration for Receptor and edit `/etc/receptor/receptor.conf` to contain the following configuration with hop node specific information:

```
---
- node:
    id: <node IP address or hostname>

- log-level: debug

- tcp-listener:
    port: <port_number>

- tcp-peer:
    address: <control hostname or IP address>:<target_port_number>
    redial: true

- tcp-peer:
```

```
    address: <control hostname or IP address>:<target_port_number>
    redial: true

- control-service:
    service: control
    filename: /var/run/receptor/receptor.sock

- work-command:
    worktype: ansible-runner
    command: /var/lib/ol-automation-manager/venv/awx/bin/ansible-runner
    params: worker
    allowruntimeparams: true
    verifysignature: false
```

In the previous example,

- *node IP address or hostname* is the IP address or hostname of the node.
- *port_number* is the port number that this node is listening on.
- *target_port* is the port number of the peer node that you're configuring this node to connect with.
- *control hostname or IP address* is the hostname or IP address of the control nodes that the hop node is connecting with.

If the hop node is associated with more than one control node, enter other `- tcp-peer:` nodes for each instance that the hop node is associated with.

2. Start the Oracle Linux Automation Manager mesh service.

```
sudo systemctl start receptor-awx
```

3. Verify the Service Mesh. For more information, see Viewing Service Mesh Status for a Cluster Node.

> **Note:**
>
> At this point in the process, the peer relationships between service mesh nodes haven't been established yet. Status information only exists for the individual servers running the Service Mesh.

Configuring the Control, Execution, and Hop Nodes

To configure the control plane, execution plane, and hop nodes, on one control plane host do the following steps which applies to all Oracle Linux Automation Manager instances:

1. Run the following commands:

```
sudo su -l awx -s /bin/bash
```

2. Do the following:
   - Repeat the following command for each host you want to choose as control node type, changing the IP address or host name each time you run the command:

     ```
     awx-manage provision_instance --hostname=<control hostname or IP address> --
     node_type=control
     ```

In the previous example, *control hostname or IP address* is the hostname or IP address of the system running Oracle Linux Automation Manager. The host name or IP address must match with the host name or IP addressed used when you configured the `/etc/receptor/receptor.conf` file node ID (see Configuring and Starting the Control Plane Service Mesh). If hostname is used, the host must be resolvable.

- Repeat the following command for each host you want to choose as execution node type, changing the IP address or host name each time you run the command:

  ```
  awx-manage provision_instance --hostname=<execution hostname or IP address> --
  node_type=execution
  ```

  In the previous example, *execution hostname or IP address* is the hostname or IP address of the system running Oracle Linux Automation Manager. The host name or IP address must match with the host name or IP addressed used when you configured the `/etc/receptor/receptor.conf` file node ID (see Configuring and Starting the Execution Plane Service Mesh). If hostname is used, the host must be resolvable.

- Repeat the following command for each host you want to choose as the hop node type, changing the IP address or host name each time you run the command:

  ```
  awx-manage provision_instance --hostname=<hop hostname or IP address> --
  node_type=hop
  ```

  In the previous example, *hop hostname or IP address* is the hostname or IP address of the system running Oracle Linux Automation Manager. The host name or IP address must match with the host name or IP addressed used when you configured the `/etc/receptor/receptor.conf` file node ID (see Configuring and Starting the Hop Nodes). If hostname is used, the host must be resolvable.

3. Run the following command to register the default execution environments, which are:
   - Control Plane Execution Environment
   - OLAM EE: (2.3)

   ```
   awx-manage register_default_execution_environments
   ```

4. Run the following command to create the controlplane instance groups (specified as a queue in the command) and associate it to a control plane host. Repeat the command with the same queue name for each control plane host in the cluster:

   ```
   awx-manage register_queue --queuename=controlplane --hostnames=<control hostname or
   IP address>
   ```

5. Run the following command to create instance groups and associate it to an execution plane host. Repeat the command with the same queue name for each execution plane host in your cluster:

   ```
   awx-manage register_queue --queuename=execution --hostnames=<execution hostname or
   IP address>
   ```

6. Run the following command to add the receptor address record for the control and the execution nodes.

   ```
   awx-manage add_receptor_address --instance=<execution hostname, control
   hostname, or IP address> --address=<execution hostname, control hostname,
   or IP address> --port=27199 --canonical
   ```

7. Run the `awx-manage list_instances` command to ensure each host you registered are available under the correct instance group. For example, the following shows the IP addresses of two control plane and three execution plane nodes running under the

controlplane and execution instance groups. The nodes aren't running yet, and therefore don't show available capacity or heartbeat information.

```
awx-manage list_instances
[controlplane capacity=0]
    192.0.119.192 capacity=0 node_type=control version=?
    192.0.124.44 capacity=0 node_type=control version=?

[execution capacity=0]
    192.0.114.137 capacity=0 node_type=execution version=ansible-runner-???
    192.0.117.98 capacity=0 node_type=execution version=ansible-runner-???
    192.0.125.241 capacity=0 node_type=execution version=ansible-runner-???

[ungrouped capacity=0]
    192.0.123.77 node_type=hop version=ansible-runner-???
```

8. Run the following command to register the Oracle Linux Automation Manager service mesh peer relationship between each node in the cluster:

   ```
   awx-manage register_peers <execution or hop hostname or IP address> --
   peers <execution, hop, or control hostname or IP address>
   ```

   This command must be run for each pair of nodes to requiring a peer relationship. For example, the peer relationships being established in the example described in Service Mesh Topology Examples shows the command being run twice for each execution node so that each execution node is connected to a different control node. This ensures that each execution node always has a backup control node if one of the control nodes were to fail.

   Other topologies are possible, such as those where an isolated execution node must peer to a hop node, and the hop node must peer to a control node. In this case the command must be run one time to peer the execution node with the hop node, and again to peer the hop node with the control node.

9. Exit the awx shell environment.

   ```
   exit
   ```

10. For each control and execution plane host, create a custom settings file in `/etc/tower/conf.d/filename.py` and include the following:

    ```
    DEFAULT_EXECUTION_QUEUE_NAME = 'execution'
    DEFAULT_CONTROL_PLANE_QUEUE_NAME = 'controlplane'
    ```

Starting the Control, Execution, and Hop Nodes

To start the control, execution, and hop nodes, do the following:

1. Start the service on each node:

   ```
   sudo systemctl enable --now ol-automation-manager.service
   ```

2. On one control plane node, run the following command to preload data, such as:
   - Demo Project
   - Default Galaxy Credentials
   - Demo Organization

- Demo Inventory
- Demo Job template
- And so on

```
sudo su -l awx -s /bin/bash
awx-manage create_preload_data
```

> **Note:**
>
> You only need to run this command one time because the preloaded data persists in the database that all cluster nodes connect with.

3. Run the `awx-manage list_instances` command to ensure that the control and execution plane nodes are now running and show available capacity and display heartbeat information. For example, the following shows all control and execution plane instances running, with available capacity, and active heartbeat information.

```
awx-manage list_instances
[controlplane capacity=270]
    192.0.119.192 capacity=135 node_type=control version=24.6.1
heartbeat="2022-09-22 14:38:29"
    192.0.124.44 capacity=135 node_type=control version=24.6.1
heartbeat="2022-09-22 14:39:09"

[execution capacity=405]
    192.0.114.137 capacity=135 node_type=execution version=24.6.1
heartbeat="2022-09-22 14:40:07"
    192.0.117.98 capacity=135 node_type=execution version=24.6.1
heartbeat="2022-09-22 14:40:35"
    192.0.125.241 capacity=135 node_type=execution version=24.6.1
heartbeat="2022-09-22 14:40:55"

[ungrouped capacity=0]
       192.0.123.77 node_type=hop heartbeat="2024-09-20 13:26:44"
```

4. Exit the awx shell environment.

```
exit
```

Configuring TLS Verification and Signed Work Requests

We recommend that you secure the Service Mesh communication within the cluster with TLS verification and signed work requests sent between cluster nodes. TLS verification ensures secure communication in the Service Mesh network and signed work requests ensure secure job execution.

The following procedure enables TLS for an existing Oracle Linux Automation Manager cluster. Complete the following tasks before doing this procedure:

- Setting up Hosts
- Configuring and Starting the Control Plane Service Mesh
- Configuring and Starting the Execution Plane Service Mesh
- Configuring and Starting the Hop Nodes

- Configuring the Control, Execution, and Hop Nodes
- Starting the Control, Execution, and Hop Nodes

To configure TLS verification and signed work requests, do the following:

1. On each host in the cluster (each execution, hop, and control plane nodes), create a file for TLS settings. For example, `/etc/tower/conf.d/tls.py`.
2. To enable signed work add the following text to `/etc/tower/conf.d/tls.py`:

   ```
   RECEPTOR_NO_SIG = False
   ```

3. Set the ownership and permissions for the custom settings file:

   ```
   sudo chown awx:awx /etc/tower/conf.d/tls.py
   sudo chmod 0640 /etc/tower/conf.d/tls.py
   ```

4. From one of the control nodes, in the `/etc/tower` folder, run the following:

   ```
   sudo mkdir -p certs
   sudo receptor --cert-init commonname="test CA" bits=2048 outcert=certs/
   ca.crt outkey=certs/ca.key
   ```

5. Do the following for each node in the cluster:
   - If you're using IP addresses for the `node_id` field, run the following commands to create the certs folder and generate TLS certificates:

     ```
     node=<node_id>; sudo receptor --cert-makereq bits=2048
     commonname="$node test cert"  ipaddress=$node nodeid=$node
     outreq=certs/$node.csr outkey=certs/$node.key
     node=<node_id>; sudo receptor --cert-signreq req=certs/$node.csr
     cacert=certs/ca.crt cakey=certs/ca.key outcert=certs/$node.crt
     ```

     In the previous example, *node_id* is the IP address of the node you're creating keys for that you set in the `/etc/receptor/receptor.conf` file for the execution, hop, or control plane nodes.
   - If you're using a host name for the `node_id` field, run the following commands to create the certs folder and generate TLS certificates:

     ```
     node=<node_id>; sudo receptor --cert-makereq bits=2048
     commonname="$node test cert"  dnsname=$node nodeid=$node
     outreq=certs/$node.csr outkey=certs/$node.key
     node=<node_id>; sudo receptor --cert-signreq req=certs/$node.csr
     cacert=certs/ca.crt cakey=certs/ca.key outcert=certs/$node.crt
     ```

     In the previous example, *node_id* is the host name of the node you are creating the keys for that you set in the `/etc/receptor/receptor.conf` file for the execution, hop, or control plane nodes.
6. After the second command, type `yes` to confirm that you want to sign the certificate. For example, the following generates certificates for a cluster with two hosts:

   ```
   node=192.0.250.40; sudo receptor --cert-makereq bits=2048
   commonname="$node test cert"  ipaddress=192.0.250.40 nodeid=$node
   outreq=certs/$node.csr outkey=certs/$node.key
   node=192.0.250.40; sudo receptor --cert-signreq req=certs/$node.csr
   ```

```
cacert=certs/ca.crt cakey=certs/ca.key outcert=certs/$node.crt
Requested certificate:
  Subject: CN=192.0.250.40 test cert
  Encryption Algorithm: RSA (2048 bits)
  Signature Algorithm: SHA256-RSA
  Names:
    IP Address: 192.0.250.40
    Receptor Node ID: 192.0.250.40
Sign certificate (yes/no)? yes

node=192.0.251.206; sudo receptor --cert-makereq bits=2048
commonname="$node test cert" ipaddress=192.0.251.206 nodeid=$node
outreq=certs/$node.csr outkey=certs/$node.key
node=192.0.251.206; sudo receptor --cert-signreq req=certs/$node.csr
cacert=certs/ca.crt cakey=certs/ca.key outcert=certs/$node.crt
Requested certificate:
  Subject: CN=192.0.251.206 test cert
  Encryption Algorithm: RSA (2048 bits)
  Signature Algorithm: SHA256-RSA
  Names:
    IP Address: 192.0.251.206
    Receptor Node ID: 192.0.251.206
Sign certificate (yes/no)? yes
```

7. From the `/etc/tower/certs` folder, run the following commands to generate certificates for work request signing and verification:

```
sudo openssl genrsa -out signworkprivate.pem 2048
sudo openssl rsa -in signworkprivate.pem -pubout -out signworkpublic.pem
```

8. From the cd `/etc/tower/` folder, run the following command to change the certs folder ownership and all files within the folder:

```
sudo chown -R awx:awx certs
```

9. Check that you have all the files you need in the `/etc/tower/certs` folder. For example, the following shows the generated key information for a four node cluster.

```
ls -al
total 68
drwxr-xr-x. 2 awx awx 4096 Sep 12 18:26 .
drwxr-xr-x. 4 awx awx  132 Sep 12 16:49 ..
-rw-------. 1 awx awx 1180 Sep 12 18:19 192.0.113.178.crt
-rw-------. 1 awx awx 1001 Sep 12 18:19 192.0.113.178.csr
-rw-------. 1 awx awx 1679 Sep 12 18:19 192.0.113.178.key
-rw-------. 1 awx awx 1176 Sep 12 18:20 192.0.121.28.crt
-rw-------. 1 awx awx 1001 Sep 12 18:20 192.0.121.28.csr
-rw-------. 1 awx awx 1675 Sep 12 18:20 192.0.121.28.key
-rw-------. 1 awx awx 1180 Sep 12 18:20 192.0.126.172.crt
-rw-------. 1 awx awx 1001 Sep 12 18:19 192.0.126.172.csr
-rw-------. 1 awx awx 1679 Sep 12 18:19 192.0.126.172.key
-rw-------. 1 awx awx 1176 Sep 12 18:19 192.0.127.70.crt
-rw-------. 1 awx awx 1001 Sep 12 18:19 192.0.127.70.csr
-rw-------. 1 awx awx 1675 Sep 12 18:19 192.0.127.70.key
-rw-------. 1 awx awx 1107 Sep 12 16:54 ca.crt
```

```
-rw-------. 1 awx awx 1679 Sep 12 16:54 ca.key
-rw-------. 1 awx awx 1675 Sep 12 18:26 signworkprivate.pem
-rw-r--r--. 1 awx awx  451 Sep 12 18:26 signworkpublic.pem
```

10. On each node in the cluster, in the `/etc/tower` folder, create a certs folder and change the ownership and group of the certs folder to `awx:awx`:

```
sudo mkdir -p certs
sudo chown -R awx:awx certs
```

11. Copy over the ca.crt, node specific .crt, csr, and key files, and the signworkprivate.pem, and signworkpublic.pem files to each node in the cluster.

12. For each control plane node, add the following lines to the `/etc/receptor/receptor.conf` file:

```
---
- node:
    id: <IP address or host name>

- log-level: debug

# Add the tls: control that specifies the tls-server name for the listener
- tcp-listener:
    port: 27199
    tls: controller

# Add the TLS server configuration
- tls-server:
    name: controller
    cert: /etc/tower/certs/<IP address or host name>.crt
    key: /etc/tower/certs/<IP address or host name>.key
    requireclientcert: true
    clientcas: /etc/tower/certs/ca.crt

- control-service:
    service: control
    filename: /var/run/receptor/receptor.sock

# Add the work-signing and work-verification elements
- work-signing:
    privatekey: /etc/tower/certs/signworkprivate.pem
    tokenexpiration: 30m

- work-verification:
    publickey: /etc/tower/certs/signworkpublic.pem

# Set verifysignature to true.
- work-command:
    worktype: local
    command: /var/lib/ol-automation-manager/venv/awx/bin/ansible-runner
    params: worker
    allowruntimeparams: true
    verifysignature: true
```

In the previous example, *IP address or host name* is the host name or IP address of the control plane host. If host name is used, the host must be resolvable.

13. For each execution plane node, add the following lines to the `/etc/receptor/receptor.conf` file:

```
---
- node:
    id: <execution IP address or host name>

- log-level: debug

- tcp-listener:
    port: 27199

# Add tls: client that specifies the tls-client name.
- tcp-peer:
    address: <hostname or IP address>:27199
    redial: true
    tls: client

- tcp-peer:
    address: <hostname or IP address>:27199
    redial: true
    tls: client

# Add the tls-client element.
- tls-client:
    name: client
    rootcas: /etc/tower/certs/ca.crt
    insecureskipverify: false
    cert: /etc/tower/certs/<execution IP address or host name>.crt
    key: /etc/tower/certs/<execution IP address or host name>.key

- control-service:
    service: control
    filename: /var/run/receptor/receptor.sock

# Add the work-verification element.
- work-verification:
    publickey: /etc/tower/certs/signworkpublic.pem

# Set verifysignature to true.
- work-command:
    worktype: ansible-runner
    command: /var/lib/ol-automation-manager/venv/awx/bin/ansible-runner
    params: worker
    allowruntimeparams: true
    verifysignature: true
```

In the previous example,

- *execution IP address or host name* is the IP address or host name of the node
- *hostname or IP address* is the host name or IP address of the execution, control, or hop node you're peering with.

14. (If required) For each hop node, add the following lines to the `/etc/receptor/receptor.conf` file:

```
---
- node:
    id: <node IP address or hostname>

- log-level: debug

# Add the tls: control that specifies the tls-server name for the listener
- tcp-listener:
    port: 27199
    tls: controller

# Add tls: client that specifies the tls-client name.
- tcp-peer:
    address: <control hostname or IP address>:27199
    redial: true
    tls: client

# Add the tls-client element.
- tls-client:
    name: client
    rootcas: /etc/tower/certs/ca.crt
    insecureskipverify: false
    cert: /etc/tower/certs/<node IP address or hostname>.crt
    key: /etc/tower/certs/<node IP address or hostname>.key

- work-verification:
    publickey: /etc/tower/certs/signworkpublic.pem

# Add the work-signing and work-verification elements
- work-signing:
    privatekey: /etc/tower/certs/signworkprivate.pem
    tokenexpiration: 30m

# Add the TLS server configuration
- tls-server:
    name: controller
    cert: /etc/tower/certs/<node IP address or hostname>.crt
    key: /etc/tower/certs/<node IP address or hostname>.key
    requireclientcert: true
    clientcas: /etc/tower/certs/ca.crt

- control-service:
    service: control
    filename: /var/run/receptor/receptor.sock


# Set verifysignature to true.
- work-command:
    worktype: local
    command: /var/lib/ol-automation-manager/venv/awx/bin/ansible-runner
    params: worker
```

```
allowruntimeparams: true
verifysignature: true
```

In the previous example,

- *node IP address or host name* is the IP address or host name of the node
- *control hostname or IP address* is the host name or IP address of the control plane host you're peering with.

15. On each node, restart the Service Mesh and Oracle Linux Automation Manager.

```
sudo systemctl daemon-reload
sudo systemctl restart receptor-awx
sudo systemctl restart ol-automation-manager
```

16. Verify the Service Mesh. See Viewing the Service Mesh for more information.

6

Adding or Removing Nodes to an Existing Cluster

This chapter provides instructions for adding or removing nodes to and from an existing cluster.

Adding a New Control Plane Node to a Cluster

To add new control node to a cluster, do the following:

1. Prepare the new hosts, as described in Setting Up the Network and Enabling Access to the Oracle Linux Automation Manager Packages.
2. Configure the host, following the instructions in Setting up Hosts. Don't run the `awx-manage migrate` or `awx-manage createsuperuser`. These only need to be run when initially creating the cluster.
3. Set up the service mesh for the control plane node, by following the instructions in Configuring and Starting the Control Plane Service Mesh.
4. Set up the service mesh for the execution plane nodes you want to connect to your new control plane node, by following the instructions in Configuring and Starting the Execution Plane Service Mesh.
5. Set up the hop nodes you want to connect to your new control plane node, by following the instructions in Configuring and Starting the Hop Nodes .
6. Provision the node as the control node type, register the node to an appropriate instance group (called a queuename in the command), and establish the peer relationships between the execution, hop, and the control nodes as described in Configuring the Control, Execution, and Hop Nodes.
7. Start the control plane node as described in Starting the Control, Execution, and Hop Nodes. Don't run the command to create preloaded data.
8. If required, apply TLS verification and signed work requests as described in Configuring TLS Verification and Signed Work Requests.

Adding a New Execution Plane Node to a Cluster

To add a new execution node to a cluster, do the following:

1. Prepare the new hosts, as described in Setting Up the Network and Enabling Access to the Oracle Linux Automation Manager Packages.
2. Configure the host, following the instructions in Setting up Hosts. Don't run the `awx-manage migrate` or `awx-manage createsuperuser`. These only need to be run when initially creating the cluster.
3. Set up the service mesh for the execution plane node, by following the instructions in Configuring and Starting the Execution Plane Service Mesh.
4. Provision the node as the execution node type, register the node to an appropriate instance group (called a queuename in the command), and establish the peer relationships

between the execution node and the control plane nodes or between the execution node and the hop nodes as described in Configuring the Control, Execution, and Hop Nodes.

5. Start the execution plane node as described in Starting the Control, Execution, and Hop Nodes. Don't run the command to create preloaded data.
6. If required, apply TLS verification and signed work requests as described in Configuring TLS Verification and Signed Work Requests.

Adding a New Hop Node to a Cluster

To add new hop node to a cluster, do the following:

1. Prepare the new hosts, as described in Setting Up the Network and Enabling Access to the Oracle Linux Automation Manager Packages.
2. Configure the host, following the instructions in Setting up Hosts. Don't run the `awx-manage migrate` or `awx-manage createsuperuser`. These only need to be run when initially creating the cluster.
3. Set up the hop nodes you want to connect to your control plane nodes, by following the instructions in Configuring and Starting the Hop Nodes .
4. Set up the execution nodes you want to connect to your new hop node, by following the instructions in Configuring and Starting the Execution Plane Service Mesh.
5. Provision the node as the hop node type, and for any new execution nodes, register the execution node to the `execution` instance group (called a queuename in the command), and establish the peer relationships between the execution, hop, and the control nodes as described in Configuring the Control, Execution, and Hop Nodes.
6. Start the hop node and execution nodes as described in Starting the Control, Execution, and Hop Nodes. Don't run the command to create preloaded data.
7. If required, apply TLS verification and signed work requests as described in Configuring TLS Verification and Signed Work Requests.

Removing a Node from a Cluster

To remove a node from a cluster, do the following:

1. Log on the node you want to remove.
2. Stop Oracle Linux Automation Manager on the node.

   ```
   sudo systemctl disable ol-automation-manager.service --now
   ```

3. Stop the service mesh.

   ```
   sudo systemctl disable receptor-awx --now
   ```

4. Delete the `/etc/tower/SECRET_KEY` file.
5. Open the `/etc/tower/settings.py` file and remove the database password from DATABASES node or remove any configuration that provides a password for your database, if you are using alternative approaches. Check for passwords in any custom settings files in `/etc/tower/conf.d`.

6. From any control plane node, verify that the node you want to remove no longer shows capacity or heartbeat information. For example, the following shows the node with IP address 192.0.124.44 has zero capacity and no heartbeat information.

```
sudo su -l awx -s /bin/bash
awx-manage list_instances
[controlplane capacity=126]
        192.0.119.192 capacity=126 node_type=control version=24.6.1
heartbeat="2022-10-20 06:55:44"
        192.0.124.44 capacity=0 node_type=control version=24.6.1

[execution capacity=126]
        192.0.114.137 capacity=126 node_type=execution version=24.6.1
heartbeat="2022-10-20 06:56:20"
```

7. Deprovision the instance from the cluster.

```
awx-manage deprovision_instance --hostname=<IP address or host name>
```

   In the previous example, *<IP address or host name>* is the host you want to remove from the cluster.

8. Check the status of the remaining control and execution plane nodes to verify that the deprovisioned instance no longer appears. For example, the deprovisioned node with IP address 192.0.124.44 from the previous example no longer appears:

```
awx-manage list_instances
[controlplane capacity=126]
        192.0.119.192 capacity=126 node_type=control version=24.6.1
heartbeat="2022-10-20 06:55:44"

[execution capacity=126]
        192.0.114.137 capacity=126 node_type=execution version=24.6.1
heartbeat="2022-10-20 06:56:20"
```

9. Exit the awx shell environment.

```
exit
```

10. If required, remove any `tcp-peer` nodes pointing to the deprovisioning node in the `/etc/receptor/receptor.conf` files of the remaining cluster nodes, the restart the nodes.

```
sudo systemctl restart receptor-awx
```

7

Viewing the Service Mesh

This chapter describes methods to view service mesh information.

Viewing Service Mesh Status for a Cluster Node

This section provides instructions for obtaining Service Mesh status information about a node in an Oracle Linux Automation Manager cluster, such as:

- **Node ID**: The node ID must be the IP address of the host.
- **System Information**: Such as CPU count and system memory.
- **Connections**: A list of IP address or host names that the node is connected with and the number of hops required to reach them, listed as the cost. Cost is defined by each customer.
- **Known Node** and **Known Node Connections**: A list of all known nodes in the cluster and the further connections known to each node listed.
- **Route**: This parameter list the route by which a node connects to another node. If the node is the same, then node is directly connected. If the nodes are different, then there is one or more hop or execution plane nodes between the nodes.
- **Node Service**: The Control Service run on every node in the cluster. It reports node status and monitors work being performed on the node.
- **Node Work Types**: The work types are Local for control plane nodes and ansible-runner for execution plane nodes.

To view service mesh status, from any host in the cluster, do the following:

1. Run the following command to obtain status information about the service mesh:

   ```
   sudo receptorctl  --socket /var/run/receptor/receptor.sock status
   ```

   For example, the following command shows the status for a four host cluster where peer relationships have been established:

   ```
   sudo receptorctl  --socket /var/run/receptor/receptor.sock status
   Node ID: 192.0.121.28
   Version: +g
   System CPU Count: 4
   System Memory MiB: 15583

   Connection         Cost
   192.0.113.178    1
   192.0.127.70     1

   Known Node         Known Connections
   192.0.113.178    {'192.0.121.28': 1, '192.0.126.172': 1}
   192.0.121.28     {'192.0.113.178': 1, '192.0.127.70': 1}
   192.0.126.172    {'192.0.113.178': 1, '192.0.127.70': 1}
   ```

```
192.0.127.70     {'192.0.121.28': 1, '192.0.126.172': 1}

Route              Via
192.0.113.178    192.0.113.178
192.0.126.172    192.0.113.178
192.0.127.70     192.0.127.70

Node               Service   Type       Last Seen             Tags
192.0.113.178    control   Stream     2022-09-02 18:06:14   {'type':
'Control Service'}
192.0.121.28     control   Stream     2022-09-02 18:06:33   {'type':
'Control Service'}
192.0.126.172    control   Stream     2022-09-02 18:06:20   {'type':
'Control Service'}
192.0.127.70     control   Stream     2022-09-02 18:06:25   {'type':
'Control Service'}

Node               Work Types
192.0.113.178    ansible-runner
192.0.121.28     local
192.0.126.172    local
192.0.127.70     ansible-runner
```

Viewing Service Mesh Cluster Status

Using the `api/v2/mesh_visualizer/` API call or a graphical user interface, you can view status information about each node in the service mesh cluster and details about available links setup between each node from the perspective of Oracle Linux Automation Manager.

To get cluster node and link details, do the following:

1. Do one of the following:
   - To view a graphical representation of cluster node and link details in release 2.3 and later, go to

     `https://<hostname or IP address>/#/topology_view`

     In the previous example, *<hostname or ip address>* is the hostname or IP address of the system running Oracle Linux Automation Manager . If hostname is used, the host must be resolvable.
   - To use the API to retrieve cluster node and link details, do the following:
     a. Log in to the Oracle Linux Automation Manager server with a user account.

        `https://<hostname or IP address>/api/login/`

        In the previous example, *<hostname or ip address>* is the hostname or IP address of the system running Oracle Linux Automation Manager . If hostname is used, the host must be resolvable.
     b. In the response area, click one of the /api/v2 links to perform a GET request that lists all available resources.
     c. Click the `/api/v2/mesh_visualizer/` link.

The Mesh Visualizer get response appears. For example:

```
HTTP 200 OK
Allow: GET, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept
X-API-Node: 192.0.121.28
X-API-Product-Name: AWX
X-API-Product-Version: 24.6.1
X-API-Time: 0.019s

{
    "nodes": [
        {
            "id": 1,
            "hostname": "192.0.121.28",
            "node_type": "control",
            "node_state": "healthy"
        },
        {
            "id": 2,
            "hostname": "192.0.127.70",
            "node_type": "execution",
            "node_state": "healthy"
        },
        {
            "id": 3,
            "hostname": "192.0.126.172",
            "node_type": "hop",
            "node_state": "healthy"
        }
    ],
    "links": [
        {
            "source": "192.0.127.70",
            "target": "192.0.121.28"
        },
        {
            "source": "192.0.126.172",
            "target": "192.0.121.28"
        },
        {
            "source": "192.0.127.70",
            "target": "192.0.126.172"
        }
    ]
}
```

8

Installing Oracle Linux Automation Manager CLI

You can install the Oracle Linux Automation Manager CLI on the same machine you installed the Oracle Linux Automation Manager server or on another Oracle Linux 8 machine. For information about installing the Oracle Linux Automation Manager CLI, see Oracle Linux Automation Manager 2.3: CLI and API Reference Guide.

9

Upgrading and Migrating Oracle Linux Automation Manager

The following chapter provides instructions for upgrading Oracle Linux Automation Manager. You can also migrate from single hybrid setup to a clustered setup if required for increased reliability and capacity.

Upgrading the Database to Version 16

To upgrade a database to version 16:

1. For each Oracle Linux Automation Manager 2.2 nodes, sign in to a terminal.
2. Stop Oracle Linux Automation Manager.

   ```
   sudo systemctl stop ol-automation-manager
   ```

3. Log in to the local or remote user account that hosts the database.

   ```
   sudo su - postgres
   ```

4. Export the database using the following command that creates a script file containing all the necessary SQL commands and input data to restore the databases. For example, this command creates the `olam.dump` file in the chosen backup directory.

   ```
   pg_dumpall > /var/tmp/olam.dump
   ```

5. Exit the user account that hosts the database.

   ```
   exit
   ```

6. Stop the database server.

   ```
   sudo systemctl stop postgresql
   ```

7. Remove (and optionally backup) existing database data directory. For example, the following command removes and creates a backup file in the home directory.

   ```
   sudo mv /var/lib/pgsql/data/ ~/data.old
   ```

8. Remove the current version of the database.

   ```
   sudo dnf remove postgresql
   ```

9. Enable the `postgresql 16` module stream.

   ```
   sudo dnf module reset postgresql
   sudo dnf module enable postgresql:16
   ```

> **Note:**
>
> For more information about the Postgresql 16 life cycle, see the appendix discussing the application life cycle for stream modules in Oracle Linux: Managing Software on Oracle Linux.

10. Install the database.

```
sudo dnf install postgresql-server
```

11. Initialize the database:

```
sudo postgresql-setup --initdb
```

12. In the `/var/lib/pgsql/data/postgresql.conf` file, switch the password storage mechanism from `md5` to `scram-sha-256`. For example, the following command makes the switch for you:

```
sudo sed -i "s/#password_encryption.*/password_encryption = scram-
sha-256/"  /var/lib/pgsql/data/postgresql.conf
```

13. Start the database and import the `/var/tmp/olam.dump` file.

```
sudo systemctl start postgresql
sudo su - postgres
psql -d postgres -f /var/tmp/olam.dump
exit
```

14. Reapply the password to the database user account:

```
sudo -u postgres psql
\password awx
```

15. Enter and confirm the password for the awx user.

```
Enter new password for user "awx":
Enter it again:
exit
```

16. Run the following command to see if the database is available:

```
sudo su - postgres -c "psql -l |grep awx"
```

    Output similar to the following is displayed:

```
awx | awx | UTF8 | libc | en_US.UTF-8 | en_US.UTF-8 | | |
```

17. As the root user, in the `/var/lib/pgsql/data/pg_hba.conf` file add the following line:

```
host  all  all 0.0.0.0/0 scram-sha-256
```

18. As the root user, in the `/var/lib/pgsql/data/postgresql.conf` file in the `# CONNECTIONS AND AUTHENTICATION` section, a line with the text `listen_addresses =` followed by the IP address or host name of the database in single quotes. For example:

```
listen_addresses = '<IP address or host name>'

#listen_addresses = 'localhost'         # what IP address(es) to listen on;
                                        # comma-separated list of
addresses;
                                        # defaults to 'localhost'; use '*'
for all
                                        # (change requires restart)
#port = 5432                            # (change requires restart)
```

In the previous example, *<IP address or hostname>* is the IP address or host name of the database.

19. If you have a remote database, check that the `5432/tcp` port is open on the host running the database. For example, the following commands shows that the ports parameter has the correct port open:

```
sudo firewall-cmd --list-all


public (active)
target: default
icmp-block-inversion: no
interfaces: enp1s0
sources:
services: cockpit dhcpv6-client ssh
ports: 5432/tcp
protocols:
forward: no
masquerade: no
forward-ports:
source-ports:
icmp-blocks:
rich rules:
```

20. If the remote database host isn't open, run the following command:

```
sudo firewall-cmd --add-port=5432/tcp --permanent
sudo firewall-cmd --reload
```

21. Calculate and update the memory requirements parameters using the following:

```
max_connections = 1024
shared_buffers = total_mem_mb*0.3
work_mem = total_mem_mb*0.03
maintenance_work_mem = total_mem_mb*0.04
```

In the previous example, *total_mem_mb* is the total memory size in megabytes of the system hosting the database server. For example, if the total available memory on the system were 18 000 MB, then this worksheet would include the following:

```
max_connections = 1024
shared_buffers = 18000*0.3
work_mem = 18000*0.03
maintenance_work_mem = 18000*0.04
```

The final numbers to add are as follows:

```
max_connections = 1024
shared_buffers = 5400MB
work_mem = 540MB
maintenance_work_mem = 720MB
```

22. Add the calculated values to the `/var/lib/pgsql/data/postgresql.conf` file.
23. Start the database server.

```
sudo systemctl restart postgresql
```

24. Restart all Oracle Linux Automation Manager 2.2 servers.

```
sudo systemctl start ol-automation-manager
```

25. Verify that Oracle Linux Automation Manager functionality continues as before the database upgrade. See Viewing the Service Mesh for various ways to perform these checks.

Upgrading Release 2.2 to Release 2.3

Use this procedure to upgrade Oracle Linux Automation Manager from 2.2 to 2.3:

1. Backup the Oracle Linux Automation Manager nodes.
2. Ensure all hosts have been updated to the latest software versions. For example,

```
sudo dnf update
```

3. Ensure that the database is upgraded. See Upgrading the Database to Version 16.
4. Log into a control node or a hybrid node.

```
sudo su -l awx -s /bin/bash
```

5. Repeat the following command for every hybrid, control, execution, and hop node in the cluster.

```
awx-manage provision_instance --hostname=<hostname or ip address> --
listener_port=<listener_port> --node_type=<node_type>
```

In the previous example, the following variables are:

- *hostname or IP address* is the hostname or IP address of the system running Oracle Linux Automation Manager. The host name or IP address must match the host name

or IP addressed used when you configured the `/etc/receptor/receptor.conf` file node ID (see Configuring and Starting the Control Plane Service Mesh). If hostname is used, the host must be resolvable.

- *listener_port* is the port number of the node, for example, 27199.
- *node_type* is the node type. Valid values are, `control`, `execution`, `hop`, or `hybrid`.

6. Exit the hybrid or control node session:

```
exit
```

7. For each Oracle Linux Automation Manager 2.2 node (control, execution, hop, or hybrid), sign in to a terminal and do the following:

   a. Stop the system:

```
sudo systemctl stop receptor-awx
sudo systemctl stop ol-automation-manager
```

   b. If not already created, create a custom settings file in the `/etc/tower/conf.d` folder. The settings you specify in custom settings files supersede the settings in `/etc/tower/settings.py`, preventing software updates from changing the custom settings. To create a custom settings file, do the following:

      i. Create a custom settings file in the `/etc/tower/conf.d` folder. For example:

```
touch /etc/tower/conf.d/olam.py
```

      ii. Add any Oracle Linux Automation Manager specific setting to the new file. For example,the CLUSTER_HOST_ID, DATABASES are some parameters you might want to add. For more information, see Setting up Hosts.

      iii. Set the ownership and file permissions for the custom settings file. For example,

```
sudo chown awx:awx /etc/tower/conf.d/olam.py
sudo chmod 0640 /etc/tower/conf.d/olam.py
```

   c. Update the system:

```
sudo dnf update oraclelinux-automation-manager-release-el8-2.3
sudo mv /etc/yum.repos.d/oraclelinux-automation-manager-ol8.repo /etc/
yum.repos.d/oraclelinux-automation-manager-ol8.repo.OLD
sudo mv /etc/yum.repos.d/oraclelinux-automation-manager-
ol8.repo.rpmnew /etc/yum.repos.d/oraclelinux-automation-manager-ol8.repo
sudo dnf update
```

   d. If after the update, the `/etc/tower/settings.py.rpmnew` appears, do the following:

```
sudo mv /etc/tower/settings.py /etc/tower/settings.py.OLD
sudo mv /etc/tower/settings.py.rpmnew /etc/tower/settings.py
```

> **Note:**
>
> Consider adding any custom settings made to the `/etc/tower/settings.py.OLD` file to the `/etc/tower/conf.d` directory in a similar way as described in step 7.b if you haven't already done so.

e. If the `/etc/nginx/conf.d/ol-automation-manager-nginx.conf.rpmnew` file appears, then custom modifications were detected in the `/etc/nginx/conf.d/ol-automation-manager-nginx.conf` file. The changes for the upgrade are contained in the `ol-automation-manager-nginx.conf.rpmnew` file. In the custom `ol-automation-manager-nginx.conf` file find the following:

```
    location /favicon.ico { alias /var/lib/awx/venv/awx/lib/python3.9/
site-packages/awx/ui/build/static/media/favicon.ico; }

    location /static/ {
        alias /var/lib/awx/venv/awx/lib/python3.9/site-packages/awx/ui/
build/static/;
    }
```

Change them as follows:

```
    location /favicon.ico { alias /var/lib/awx/venv/awx/lib/python3.11/
site-packages/awx/ui/build/static/media/favicon.ico; }

    location /static/ {
        alias /var/lib/awx/venv/awx/lib/python3.11/site-packages/awx/ui/
build/static/;
    }
```

f. Pull the 2.3 olam-ee image using.

> **Note:**
>
> If you have set the `GLOBAL_JOB_EXECUTION_ENVIRONMENTS` and `CONTROL_PLANE_EXECUTION_ENVIRONMENT` variables as described in Setting up Hosts, ensure that you update these to point to the `olam-ee:2.3-ol8` container image. Note that if the `olam-ee:2.3-ol8` container image is on a Private Automation Hub server, change the podman pull command as required on the variable and in the following commands.

```
sudo su -l awx -s /bin/bash
podman system migrate
podman pull container-registry.oracle.com/
oracle_linux_automation_manager/olam-ee:2.3-ol8
awx-manage makemigrations --merge
awx-manage migrate
awx-manage register_default_execution_environments
exit
```

8. Restart the following services on all nodes:

```
sudo systemctl restart nginx
sudo systemctl start receptor-awx
sudo systemctl start ol-automation-manager
```

Migrating a Single Instance Release 2.3 Deployment to a Clustered Deployment

To migrate a single host instance deployment of Oracle Linux Automation Manager (hybrid node) to a clustered deployment (control, execution, and hop node clusters), do the following:

1. In a terminal on the hybrid node, stop Oracle Linux Automation Manager.

```
sudo systemctl stop ol-automation-manager
```

2. Create a database dump file.

```
sudo su - postgres
pg_dumpall > /var/tmp/olam.dump
```

3. Complete the procedures for setting up a remote database in Setting Up a Local or Remote Database with the following exceptions:
   a. Before starting the procedure, copy over the dump file to the remote database. For example, using scp.
   b. After starting and checking the status of the database in step 7 and 8, import the dump file:

```
sudo su - postgres
psql -d postgres -f /dirwithbackup/olam.dump
exit
```

   c. Skip steps 9 through 11 for creating the database user account and creating the database because these are already part of the dump file.
   d. Continue the procedure at step 12 until you restart the database.
4. On the remote database, reapply the password to the database user account:

```
sudo -u postgres psql
\password awx
```

5. Enter and confirm the password for the awx user.

```
Enter new password for user "awx":
Enter it again:
exit
```

6. If the single instance hybrid node had a local database, do the following for a remote database configuration:

a. Return to the Oracle Linux Automation Manager server, and create a custom settings file. For example:

```
sudo touch /etc/tower/conf.d/olam.py
```

b. Add the following `DATABASES` fields to the custom settings file:

```
DATABASES = {
    'default': {
        'ATOMIC_REQUESTS': True,
        'ENGINE': 'awx.main.db.profiled_pg',
        'NAME': 'awx',
        'USER': 'awx',
        'PASSWORD': 'password',
        'HOST': 'database hostname or ip address',
        'PORT': '5432',
    }
}
```

In the previous example, *database hostname or ip address* is the hostname or IP address of the remote database. If hostname is used, the host must be resolvable. *password* is the password for your remote database, if you have configured one.

7. Set the ownership and file permissions for the custom settings file:

```
sudo chown awx:awx /etc/tower/conf.d/filename.py
sudo chmod 0640 /etc/tower/conf.d/filename.py
```

In the previous example, filename is the name of the custom settings file. For example,

```
sudo chown awx:awx /etc/tower/conf.d/olam.py
sudo chmod 0640 /etc/tower/conf.d/olam.py
```

8. Stop the local database.

```
sudo systemctl stop postgresql
```

9. Open the ports used for the Service Mesh.

```
sudo firewall-cmd --add-port=27199/tcp --permanent
sudo firewall-cmd --reload
```

10. Start Oracle Linux Automation Manager.

```
sudo systemctl start ol-automation-manager
```

11. Backup and remove the existing database data directory. For example, the following command creates a backup file in the home directory and then removes the database:

```
sudo mv /var/lib/pgsql/data/ ~/data.old
sudo dnf remove postgresql
```

12. Run the following command:

```
sudo su -l awx -s /bin/bash
```

13. Run the following commands.

```
awx-manage deprovision_instance --hostname=<hostname or IP address>
awx-manage unregister_queue --queuename default
awx-manage unregister_queue --queuename tower
awx-manage provision_instance --hostname=<hostname or IP address> --
node_type=control
awx-manage register_queue --queuename=controlplane --hostnames=<hostname
or IP address>
awx-manage add_receptor_address --instance=<hostname or ip address> --
address=<hostname or ip address> --port=<listener_port> --canonical
exit
```

    In the previous example, *hostname or IP address* is the hostname or IP address of the system running Oracle Linux Automation Manager. The host name or IP address must match the host name or IP addressed used when you configured the `/etc/receptor/receptor.conf` file node ID (see Configuring and Starting the Control Plane Service Mesh). If hostname is used, the host must be resolvable. You only need to unregister tower if it exists and is no longer used. *listener_port* is the port number of the node, for example, 27199.

14. In the custom settings file (for example, `/etc/tower/conf.d/olam.py`), append the `DEFAULT_EXECUTION_QUEUE_NAME` and `DEFAULT_CONTROL_PLANE_QUEUE_NAME` fields:

```
DEFAULT_EXECUTION_QUEUE_NAME = 'execution'
DEFAULT_CONTROL_PLANE_QUEUE_NAME = 'controlplane'
```

15. If you have playbooks designed to run longer than the default reaper timeout of 60 seconds, add the `REAPER_TIMEOUT_SEC` parameter to the custom settings file to increase the timeout. For example,

```
REAPER_TIMEOUT_SEC=<longest_playbook_time>
```

    In the previous example, <*longest_playbook_time*> is number of seconds that exceeds the duration of the longest playbook runtime.

16. Restart Oracle Linux Automation Manager.

```
sudo systemctl restart  ol-automation-manager.service
```

17. The original upgraded node is now converted into a control node. You must now add one more execution node for the upgraded cluster to be fully functional. For all other members of the cluster, follow the procedures described in Preparing the Database and Hosts, except for setting up a remote database, because this is already completed. Then, follow the procedures for installing and configuring all other hosts as part of the cluster, as described in Installing Oracle Linux Automation Manager in a Clustered Deployment.

Migrating Playbooks to Oracle Linux Automation Engine Release 2.3

Test any Oracle Linux Automation Engine release 2.2 playbooks to verify whether they function with Oracle Linux Automation Manager release 2.3. You might need to update playbooks because the upstream projects have made changes such as, the number of modules, some

modules have become collections, and some modules have been merged into other modules or collections.

Verify that the playbooks are using the correct versions of python as described in Oracle Linux Automation Manager 2.3: Release Notes. For custom execution environments, review the overview section in Oracle Linux Automation Manager 2.3: Private Automation Hub User's Guide.