

# Oracle Linux Automation Manager 2

## Private Automation Hub Installation Guide



F77465-02  
September 2023



Oracle Linux Automation Manager 2 Private Automation Hub Installation Guide,  
F77465-02

Copyright © 2022, 2023, Oracle and/or its affiliates.

# Contents

## Preface

---

Conventions	v
Documentation Accessibility	v
Access to Oracle Support for Accessibility	v
Diversity and Inclusion	v

## 1 Preparing to Install Private Automation Hub

---

Private Automation Hub Hardware Requirements	1-1
Installation Options	1-1
Set Up Passwordless SSH	1-2
Enabling Access to the Private Automation Hub Packages	1-3
Enabling Repositories with the Oracle Linux Yum Server	1-3
Enabling Channels with ULN and Setting up a Local Mirror	1-4

## 2 Installing Private Automation Hub

---

Installing on a Single Host	2-1
Configuring the Installation Parameter File	2-3
Configuring LDAP Parameters	2-5

## 3 Installing the Builder Utility

---

About the Builder Utility	3-1
Installing Builder	3-1

## 4 Backing up and Restoring Private Automation Hub

---

Offline Backing up Private Automation Hub	4-1
Offline Backing up Private Automation Hub with a Remote Database	4-2
Offline Restoring Private Automation Hub	4-3
Offline Restoring Private Automation Hub with a Remote Database	4-4
Offline Restoring Private Automation Hub to a New Host	4-5



# Preface

[Oracle Linux Automation Manager 2: Private Automation Hub Installation Guide](#) describes how to install Oracle Linux Automation Manager Private Automation Manager in single-host deployments

## Conventions

The following text conventions are used in this document:

Convention	Meaning
<b>boldface</b>	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	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/>.

For information about the accessibility of the Oracle Help Center, see the Oracle Accessibility Conformance Report at <https://www.oracle.com/corporate/accessibility/templates/t2-11535.html>.

## 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

## Preparing to Install Private Automation Hub

This chapter describes the requirements for the systems to be used in an installation of Private Automation Hub.

### Private Automation Hub Hardware Requirements

You can install Private Automation Hub on a single machine in x86-64 Oracle Linux 8 hosts.

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

The preceding list outlines the minimum requirements to run Private Automation Hub. You must determine any additional hardware requirements and capacity based on your operational needs.

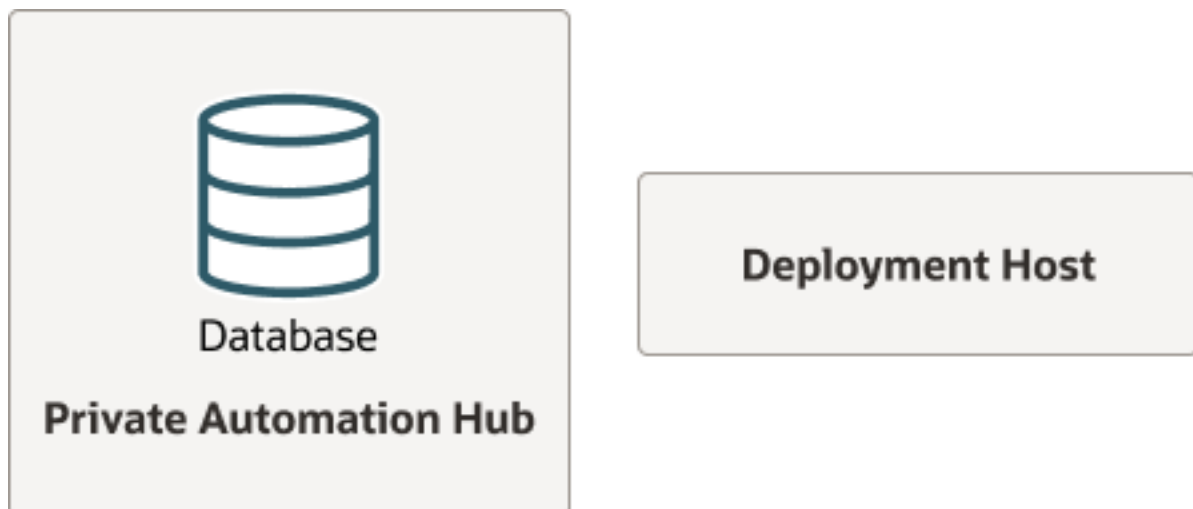
Hosts must be configured to establish and accept an ssh connection. Consider setting up passwordless login between the deployment host and target hosts to simplify the installation process. For more information, see [Set Up Passwordless SSH](#).

### Installation Options

Private Automation Hub provides three installation options:

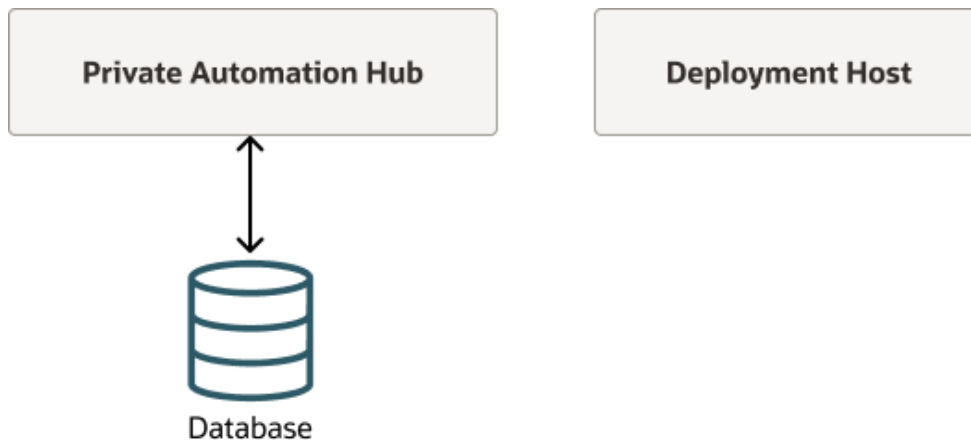
- Standalone installation: All components of are on the same host, including the database deployed from the deployment host.

**Figure 1-1 Standalone Installation with Local Database**



- Standalone installation with remote database: All components are on the same host, with the exception of the database which is on a remote host and deployed from the deployment host.

**Figure 1-2 Standalone Installation with Remote Database**



## Set Up Passwordless SSH

Set up passwordless SSH connections from the deployment host to the target hosts.

Passwordless SSH is used to copy the CA certificates to the nodes. This should be set up for the user on the deployment host that installs Private Automation Hub, for example, for the `opc` or `oracle` user.

Set up passwordless SSH between the deployment host and all target hosts.

For example, one way to set up passwordless SSH to the target hosts, is as follows:

1. On the deployment host, use `ssh-keygen` to generate a public and private key pair. For example:

```
ssh-keygen

Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa): <Enter>
Created directory '/home/user/.ssh'.
Enter passphrase (empty for no passphrase): <Enter>
Enter same passphrase again: <Enter>
...
```

Press `Enter` each time you are prompted to enter a passphrase.

2. Copy the public key into the `~/.ssh/authorized_keys` file for each target host.
3. To avoid authentication verification prompts, add authenticated hosts to the `~/.ssh/known_hosts` file. For example,

```
ssh-keyscan -H <host> >> ~/.ssh/known_hosts
```

4. To verify that the deployment host can access the target system without supplying a password, use `ssh` to log each target system. For example:

```
ssh <remote_user>@<host>
```



For more information on setting up passwordless SSH, see [Oracle Linux: Connecting to Remote Systems With OpenSSH](#).

## Enabling Access to the Private Automation Hub Packages

This section contains information on setting up the locations for the operating system on which you want to install the Private Automation Hub software packages.

The Private Automation Hub packages for Oracle Linux 8 are available on the Oracle Linux yum server in the `ol8_automation2` repository, or on the Unbreakable Linux Network (ULN) in the `ol8_x86_64_automation2` channel. However there are also dependencies across other repositories and channels, and these must also be enabled on each system where Private Automation Hub is installed.

## Enabling Repositories with the Oracle Linux Yum Server

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

To enable the yum repositories:

1. Use the `dnf config-manager` tool to enable the `ol8_baseos_latest` repository.

```
sudo dnf config-manager --enable ol8_baseos_latest
```

 **Note:**

This repository is typically enabled by default.

2. Install `oraclelinux-automation-manager-release-el8`:

```
sudo dnf install oraclelinux-automation-manager-release-el8
```

3. Enable the following yum repositories:

- `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 are using `ol8_UEK6`, use the following command:

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

- If you are using `ol8_UEK7`, use the following command:

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

4. Ensure that no version of `ansible` is present on the system. If any are, uninstall them. For example, the following shows that there are no versions of `ansible` installed:

```
rpm -q ansible
```

The following response should appear:

```
package ansible is not installed
```

## Enabling Channels with ULN and Setting up a Local Mirror

If you are 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 user name and password.
2. On the Systems tab, click the link named for the system in the list of registered machines.
3. Register your deployment and target hosts on ULN. For more information, see [Oracle Linux: Managing Software on Oracle Linux](#).
4. On the System Details page, click **Manage Subscriptions**.
5. 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:
  - o18\_x86\_64\_automation2
  - o18\_x86\_64\_addons
  - o18\_x86\_64\_baseos\_latest
  - o18\_x86\_64\_UEKR6 or o18\_x86\_64\_UEKR7
  - o18\_x86\_64\_appstream
6. Click **Save Subscriptions**.
7. Setup a local ULN mirror for the o18\_x86\_64\_automation2 channel as described in [Oracle Linux: Managing Software on Oracle Linux](#).

 **Note:**

Ensure that you set the `pulp_pkg_repo` variable to point to the location of the ULN mirror. For more information, see [Installing Private Automation Hub](#).

# 2

## Installing Private Automation Hub

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

### Installing on a Single Host

This section provides instructions for installing the Private Automation Hub on a single host where the database is local or on a remote host and assumes that you have setup a passwordless SSH connection.

To set up the host:

1. On the deployment host, login as the user configured with Passwordless SSH to the target host. For more information, see [Set Up Passwordless SSH](#).
2. Ensure python 3.6 is installed on your host. If python 3.6 is not installed, run the following command:

```
sudo dnf install python36
```

3. Install the Private Automation Hub software:

```
sudo dnf install ol-private-automation-hub-installer
```

4. Copy the contents of the /single-node folder to a working directory.

```
cp -r /usr/share/ansible/collections/ansible_collections/oraclelinux/  
private_automation_hub/playbooks/single-node/ ~/single_node
```

5. From the working directory, create a `hosts` file from the `hosts.singlenode.example`. For example,

```
cd ~/single_node  
cp hosts.singlenode.example hosts
```

6. Edit the `hosts` file as follows:

```
all:  
  hosts:  
    hub:  
      ansible_host: <ip_address_or_hostname>  
      ansible_user: <username>
```

In the previous example,

- `<ip_address_or_hostname>` is the IP address or host name of the target node where you want to install Private Automation Hub. This host must be reachable using SSH from the deployment host.

 **Note:**

Valid characters for hostnames are a to z, 0 to 9, and the hyphen (-). A hostname may not start with a hyphen.

- `<username>` is the username running the installer playbook commands on the target node where you want to install Private Automation Hub. This user must have sudo privileges.
- 7. If you want to configure additional installation parameters to use during the installation, setup the installation parameter file as described in [Configuring the Installation Parameter File](#).
- 8. Do one of the following:
  - If you want to install a local database on the same host running Private Automation Hub, run the following command:

```
ansible-playbook single-node-install.yml -i hosts -e  
"olpah_admin_password=<admin_password>  
olpah_db_password=<db_password>"
```

In the previous example, `<admin_password>` and `<db_password>` are the passwords for the default admin user and the database user account.

 **Note:**

If you want to use the parameter file, add the following to the end of the command:

```
-e "@single-node-vars.yml"
```

- If you want to use an existing database on a remote host, run the following command:
  - a. Log into the remote database.
  - b. Create the database user accounts. For example:

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

- c. Enter and confirm the password for the pulp user.

 **Note:**

This must be the same `<db_password>` as specified in the previous step.

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

- d. Create the database instance. For example:

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

- e. From the `single-node-install.yml` file, remove the `pulp_database` role.
- f. Set the database hostname or IP address for the remote database (`existing_db_host: <db_hostname_or_ip_address>`) in the `"@single-node-vars.yml"` variables file. For more information about installing using the parameter file, see [Configuring the Installation Parameter File](#).
- g. Return to the deployment server and run the following command:

```
ansible-playbook single-node-install.yml -i hosts -e
"olpah_admin_password=<admin_password>
olpah_db_password=<db_password>" -e "@single-node-vars.yml"
```

In the previous example, `<admin_password>` and `<db_password>` are the passwords for the default admin user and the database user account.

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

```
https://<ip_address_or_hostname>
```

## Configuring the Installation Parameter File

In some cases, you may want to configure additional parameters when installing Private Automation Hub. If you are configuring extra parameters for a single host installation, edit the `single-node-vars.yml` parameter file.



### Note:

You can configure a parameter file before or after you install Private automation Hub. If you do it after installing Private Automation Hub, then you must complete this step and rerun the playbook as described in [Installing on a Single Host](#).

To configure extra installation parameters in a parameter file, do the following:

1. In the parameter file, add the extra parameters you need. For example:

```
existing_db_host: <db_hostname_or_ip_address>
pulp_pkg_repo: "<local_repo_url>"

olpah_require_content_approval: <True or False>
pulp_api_workers: <Number_of_workers>
connected_olam_controllers: [
  "https://<olam_controller_server_url1>/",
  "https://<olam_controller_server_url2>/",
  ...
]
```

- If you want to use a remote database, add the following parameter to the parameter file. For example,

```
existing_db_host: <db_hostname_or_ip_address>
```

In the previous example, `<db_hostname_or_ip_address>` is the host name or IP address of the remote database.

- If you want to use a remote mirror of ULN or yum repos, add the following parameter to the parameter file. For example,

```
pulp_pkg_repo: "<local_repo_url>"
```

In the previous example, `<local_repo_url>` is the url of the remote mirror repository. The url path should look similar to the following:

```
pulp_pkg_repo: "http://192.0.121.30/yum/OracleLinux/OL8/automation2/$basearch/ol8_x86_64_automation2/"
```

- If you want to enable the approval process for collection uploads, enable the following parameter in a parameter file.

```
olpah_require_content_approval: True
```

 **Note:**

You can do this step before or after you install Private automation Hub. If you do it after installing Private Automation Hub, then you must complete this step and rerun the playbook as described in this procedure.

- If you want to modify the default number of Pulp API workers available for Private Automation Hub, consider setting this value to the same number as the CPU cores as are available on the target instance. For example,

```
pulp_api_workers: 2
```

- If you want to link one or more Oracle Linux Automation Manager control server with Private Automation Hub to enable easier configuration of execution environments in Oracle Linux Automation Manager, add one or more url to the following parameter in a parameter file.

```
connected_olam_controllers: [  
  "https://<olam_controller_server_url1>/",  
  "https://<olam_controller_server_url2>/",  
  ...  
]
```

In the previous example, `<olam_controller_server_url1>` and `<olam_controller_server_url2>` are the urls to the control servers. You can add

more of these URLs depending on the number of control servers you want to make available. The url must include `https://`. For more information about this feature, see [Oracle Linux Automation Manager 2: Private Automation Hub User's Guide](#).

- If you want to integrate Private Automation Hub with an LDAP sever, add the LDAP parameters described in [Configuring LDAP Parameters](#).

## Configuring LDAP Parameters

To configure the LDAP parameters in the parameters file, do the following:

1. Edit the parameter file and add the following required LDAP related parameters to the bottom of the file:

```
#Enable galaxy_ng LDAP Integration
config_ldap: True

# LDAP Binding and Directory Look Up
auth_ldap_server_uri: "<ldap_url>"
auth_ldap_bind_dn: "<ldap_bind>"
auth_ldap_bind_password: "<ldap_bind_password>"
auth_ldap_user_search_base_dn: "cn=users,cn=accounts,dc=example,dc=com"
auth_ldap_user_search_scope: "SUBTREE"
auth_ldap_user_search_filter: "(uid=%(user)s)"
auth_ldap_group_search_base_dn: "cn=groups,cn=accounts,dc=example,dc=com"
auth_ldap_group_search_scope: "SUBTREE"
auth_ldap_group_search_filter: "(objectClass=groupofnames)"
auth_ldap_group_type_class: "django_auth_ldap.config:GroupOfNamesType"
auth_ldap_user_flags_by_group_is_superuser:
"cn=superuserexample,cn=groups,cn=accounts,dc=example,dc=com"
auth_ldap_mirror_groups: False

# LDAP Backend
ldap_logging: True
auth_ldap_start_tls: True
use_galaxy_ldap_self_signed_cert: True
```

In the previous example,

- `config_ldap`  
Set the value of the `config_ldap` parameter to `True` to enable LDAP integration.
- `auth_ldap_server_uri`  
Provide the URI to access your LDAP server in the format: `ldap://<host>` where `<host>` is the host name of the LDAP server. This field is required. For example,

```
ldap://ldap1.example.com
```

If your server uses StartTLS functionality, you can set the protocol to `ldaps` within the URI scheme and enable the `auth_ldap_start_tls` option.

- `auth_ldap_bind_dn`

Provide the Distinguished Name (DN) used to authenticate Oracle Linux Automation Manager against the LDAP server using the Bind operation. This field is required if your LDAP server does not allow anonymous access. For example:

```
uid=admin,cn=users,cn=accounts,dc=example,dc=com
```

- `auth_ldap_bind_password`  
Provide the Bind password for the Bind DN that you provided above.
- `auth_ldap_user_search_base_dn`  
Provide the DN where your users are listed within the directory.
- `auth_ldap_user_search_scope`  
Provide the scope to use when performing an LDAP search query on the base DN where your users are listed. Typically, the scope value is set to either one level deep, ONELEVEL, or to the entire subtree, SUBTREE.
- `auth_ldap_user_search_filter`  
Provide the search filter to be applied when performing an LDAP search query on the base DN where your users are listed. You can use the `%(user)s` syntax to match an attribute or key to the username value that a user provided during authentication.
- `auth_ldap_group_search_base_dn`  
Provide the base DN to use when performing an LDAP search query to determine group membership for a user.
- `auth_ldap_group_search_scope`  
Provide the scope to use when performing an LDAP search query on the base DN where groups are listed in the directory. Typically, the scope value is set to either one level deep, ONELEVEL, or to the entire subtree, SUBTREE.
- `auth_ldap_group_search_filter`  
Provide the search filter to be applied when performing an LDAP search query on the base DN where groups are listed in the directory.
- `auth_ldap_group_type_class`  
Provide an appropriate LDAP group type to define how the LDAP server determines group membership for users when attempting to authorize them. LDAP group types map onto the ObjectClasses that are defined for any groups that are listed on an LDAP server and can vary depending on the LDAP server implementation. The values for this parameter are related to the underlying Django framework and the LDAP ObjectClasses that the framework recognizes. Therefore, values are prefixed with `django_auth_ldap.config`.
- `auth_ldap_user_flags_by_group__is_superuser`  
Any user associated with this group has superuser privileges on Private Automation Hub.



**▲ Caution:**

If you make an error with this value, you cannot log into Private Automation Hub after the installation process completes. You must correct the error and run the installation process again before you can log in. If the LDAP server has not been configured with the specified superuser group yet, you cannot log into Private Automation Hub until the superuser group information has been added to the LDAP server with the exception of the locally defined admin user.

- `auth_ldap_mirror_groups`  
Enabling this feature mirrors LDAP groups associated with a user in Private Automation Hub whenever a user logs in to Private Automation Hub. Set this value to True if you want to enable this feature.
- `ldap_logging`  
Set the `ldap_logging` value to True if you want to retain a log of LDAP activity. Logging can help debug authentication issues.
- `auth_ldap_start_tls`  
If the LDAP server uses StartTLS functionality, you can set the protocol to ldap within the URI scheme used in `auth_ldap_server_uri` and set the `auth_ldap_start_tls` value to True.
- `use_galaxy_ldap_self_signed_cert`  
Set the `use_galaxy_ldap_self_signed_cert` value to True if the certificates used for TLS or SSL on your LDAP server are self-signed and you want to disable validation of the certificate against a CA.

# 3

## Installing the Builder Utility

This chapter shows you how to set up the builder utility on a host running x86-64 Oracle Linux 8.

### About the Builder Utility

The Builder utility is an ansible-builder based tool used for generating containers that Oracle Linux Automation Manager can use as execution environments in control or execution nodes to run playbooks. After creating these custom execution environments, you can upload them to Private Automation Hub so that Oracle Linux Automation Manager execution and control nodes can download them when necessary. For more information about creating container environments and uploading them to Private Automation Hub, see [Oracle Linux Automation Manager 2: Private Automation Hub User's Guide](#).

### Installing Builder

To install the Builder utility, do the following:

1. On a host running x86-64 Oracle Linux 8, setup the repositories as described in [Enabling Access to the Private Automation Hub Packages](#).

 **Note:**

Do not install the Builder utility on any Oracle Linux Automation Manager host being used as a control node, execution node, or private automation hub nodes.

2. Install the Builder utility.

```
sudo dnf install python39-ansible-builder
```

The Builder utility is now installed. For more information about setting up and using the Builder utility to create new container, see [Oracle Linux Automation Manager 2: Private Automation Hub User's Guide](#).

# 4

## Backing up and Restoring Private Automation Hub

The following chapter provides information about backing up and restoring Private Automation Hub. Perform an offline backup where all relevant services are stopped and the data is not changing at the time of the backup; this backup is consistent by definition.

### Offline Backing up Private Automation Hub

To do an offline backup Private Automation Hub, do the following:

1. Create a backup directories. Ensure you have sufficient disk space and consider making the directory persistent.

 **Note:**

For the purposes of this document, we use a folder in the home directory, which is not secure.

```
sudo mkdir -p ~/backup/etc/pulp ~/backup/var/lib/pulp ~/backup/etc/nginx/  
pulp ~/backup/var/lib/pgsql/data
```

2. Stop the following services in the following order:

```
sudo systemctl stop pulpcore  
sudo systemctl stop nginx  
sudo systemctl stop postgresql
```

3. Copy the following folders into the backup directory.

 **Note:**

cp option r ensures that the backup includes all subdirectories and p ensures that all permissions are preserved.

```
sudo cp -rp /etc/pulp/ ~/backup/etc/  
sudo cp -rp /var/lib/pulp ~/backup/var/lib/  
sudo cp -rp /etc/nginx/pulp ~/backup/etc/nginx/  
sudo cp -rp /var/lib/pgsql/data ~/backup/var/lib/pgsql/
```

4. Restart the services in the following order:

```
sudo systemctl start postgresql
sudo systemctl start pulpcore* --all
sudo systemctl start nginx
```

 **Note:**

Consider testing the backup to ensure it works as expected. For more information, see [Offline Restoring Private Automation Hub](#).

## Offline Backing up Private Automation Hub with a Remote Database

To do an offline backup Private Automation Hub with a remote database, do the following:

1. On the Private Automation Hub server, create a backup directories. Ensure you have sufficient disk space and consider making the directory persistent.

 **Note:**

For the purposes of this document, we use a folder in the home directory, which is not secure.

```
sudo mkdir -p ~/backup/etc/pulp ~/backup/var/lib/pulp ~/backup/etc/nginx/pulp
```

2. On the database server, create a backup directory for the following folder. Ensure you have sufficient disk space and consider making the directory persistent.

```
sudo mkdir -p ~/backup/var/lib/pgsql/data
```

3. Stop the following services on the Private Automation Hub server in the following order:

```
sudo systemctl stop pulpcore
sudo systemctl stop nginx
```

4. Stop the following service on the database server:

```
sudo systemctl stop postgresql
```

5. Copy the following folders into the backup directory from the Private Automation Hub server.

 **Note:**

cp option r ensures that the backup includes all subdirectories and p ensures that all permissions are preserved.

```
sudo cp -rp /etc/pulp/ ~/backup/etc/  
sudo cp -rp /var/lib/pulp ~/backup/var/lib/  
sudo cp -rp /etc/nginx/pulp ~/backup/etc/nginx/
```

6. Copy the following folders into the backup directory from the database server.

```
sudo cp -rp /var/lib/pgsql/data ~/backup/var/lib/pgsql/
```

7. Restart the services on the database Server:

```
sudo systemctl start postgresql
```

8. Restart the services on Private Automation Hub in the following order:

```
sudo systemctl start pulpcore* --all  
sudo systemctl start nginx
```

 **Note:**

Consider testing the backup to ensure it works as expected. For more information, see [Offline Restoring Private Automation Hub](#).

## Offline Restoring Private Automation Hub

To do an offline restore of Private Automation Hub, do the following:

 **Note:**

All data entered after taking an offline backup is lost when the backup is restored.

1. If running, stop the following services in the following order:

```
sudo systemctl stop pulpcore  
sudo systemctl stop nginx  
sudo systemctl stop postgresql
```

2. Copy the following folders from the backup directory to the Private Automation Hub server.

 **Note:**

cp option r ensures that the backup includes all subdirectories and p ensures that all permissions are preserved.

```
sudo cp -rp ~/backup/etc/pulp /etc/  
sudo cp -rp ~/backup/var/lib/pulp /var/lib/  
sudo cp -rp ~/backup/etc/nginx/pulp /etc/nginx/  
sudo cp -rp ~/backup/var/lib/pgsql/data /var/lib/pgsql/
```

3. Restart the services in the following order:

```
sudo systemctl daemon-reload  
sudo systemctl start postgresql  
sudo systemctl start pulpcore* --all  
sudo systemctl start nginx
```

## Offline Restoring Private Automation Hub with a Remote Database

To do an offline restore of Private Automation Hub with a remote database, do the following:

 **Note:**

All data entered after taking an offline backup is lost when the backup is restored.

1. Stop the following services on the Private Automation Hub server in the following order:

```
sudo systemctl stop pulpcore  
sudo systemctl stop nginx
```

2. Stop the following service on the database server:

```
sudo systemctl stop postgresql
```

3. Copy the following folders from the backup directory to the Private Automation Hub server.

 **Note:**

cp option r ensures that the backup includes all subdirectories and p ensures that all permissions are preserved.

```
sudo cp -rp ~/backup/etc/pulp /etc/  
sudo cp -rp ~/backup/var/lib/pulp /var/lib/  
sudo cp -rp ~/backup/etc/nginx/pulp /etc/nginx/
```

4. Copy the following folders from the backup directory to the database Server.

```
sudo cp -rp ~/backup/var/lib/pgsql/data /var/lib/pgsql/
```

5. Restart the services on the database Server:

```
sudo systemctl daemon-reload  
sudo systemctl start postgresql
```

6. Restart the services on Private Automation Hub in the following order:

```
sudo systemctl daemon-reload  
sudo systemctl start pulpcore* --all  
sudo systemctl start nginx
```

## Offline Restoring Private Automation Hub to a New Host

To do an offline restore of Private Automation Hub to a new host, do the following:

 **Note:**

All data entered after taking an offline backup is lost when the backup is restored.

1. Ensure the new host is installed with exactly the same configuration and settings as the original host where the backup was taken. This includes running identical installer playbooks and the software versions (for example, same database version and Private Automation Hub version). The host and IP address are the only parameters that need to change when running the playbook. For more information, see [Installing Private Automation Hub](#).

2. If running, stop the following services on the original host in the following order:

```
sudo systemctl stop pulpcore  
sudo systemctl stop nginx  
sudo systemctl stop postgresql
```

3. Copy the following folders from the backup directory to the new Private Automation Hub server using whatever method you need. Copy the backup files to the following directories:

 **Note:**

cp option r ensures that the backup includes all subdirectories and p ensures that all permissions are preserved.

```
sudo cp -rp ~/backup/etc/pulp /etc/
sudo cp -rp ~/backup/var/lib/pulp /var/lib/
sudo cp -rp ~/backup/etc/nginx/pulp /etc/nginx/
sudo cp -rp ~/backup/var/lib/pgsql/data /var/lib/pgsql/
```

4. Update the `/etc/pulp/settings.local.py` on the new host with the following parameters:

```
DATABASES = {'default': {'HOST': '<IP address or host name>',
                        'ENGINE': 'django.db.backends.postgresql', 'NAME': 'pulp', 'USER':
                        'pulp', 'PASSWORD': '<db_password>'}}
TOKEN_SERVER = 'https://<<IP address or host name>>/token/'
and
CONTENT_ORIGIN = 'https://<IP address or host name>'
```

In the previous example, `<IP address or host name>` is the IP address or host name of the standalone Private Automation Hub server and `<db_password>` is the password for the database.

5. Restart the services in the following order:

```
sudo systemctl daemon-reload
sudo systemctl start postgresql
sudo systemctl start pulpcore* --all
sudo systemctl start nginx
```

## Offline Restoring Private Automation Hub to a New Host with Remote Database

To do an offline restore of Private Automation Hub to a new host with remote database, do the following:

 **Note:**

All data entered after taking an offline backup is lost when the backup is restored.

1. Ensure the new host is installed with exactly the same configuration and settings as the original host and remote database where the backup was taken. This includes running identical installer playbooks and the software versions (for example, same database version and Private Automation Hub version). The host and IP address are the only parameters that need to change when running the playbook. For more information, see [Installing Private Automation Hub](#).



2. If running, stop the following services on the Private Automation Hub server in the following order:

```
sudo systemctl stop pulpcore
sudo systemctl stop nginx
```

3. If running, stop the following service on the database server:

```
sudo systemctl stop postgresql
```

4. Copy the following folders from the backup directory to the new Private Automation Hub server using whatever method you need. Copy the backup files to the following directories:

 **Note:**

cp option r ensures that the backup includes all subdirectories and p ensures that all permissions are preserved.

```
sudo cp -rp ~/backup/etc/pulp /etc/
sudo cp -rp ~/backup/var/lib/pulp /var/lib/
sudo cp -rp ~/backup/etc/nginx/pulp /etc/nginx/
```

5. Update the `/etc/pulp/settings.local.py` on the new Private Automation Hub server with the following parameters:

```
DATABASES = {'default': {'HOST': '<database IP address or host name>',
                        'ENGINE': 'django.db.backends.postgresql', 'NAME': 'pulp', 'USER':
                        'pulp', 'PASSWORD': '<db_password>'}}
TOKEN_SERVER = 'https://<<IP address or host name>>/token/'
and
CONTENT_ORIGIN = 'https://<IP address or host name>'
```

In the previous example, `<database IP address or host name>` is the IP address or host name of the database, `<IP address or host name>` is the IP address or host name of the standalone Private Automation Hub server and `<db_password>` is the password for the database.

6. Copy the following folders from the backup directory to the new database server using whatever method you need. Copy the backup files to the following directory:

```
sudo cp -rp ~/backup/var/lib/pgsql/data /var/lib/pgsql/
```

7. Restart the services on the database Server:

```
sudo systemctl daemon-reload
sudo systemctl start postgresql
```

8. Restart the services on Private Automation Hub in the following order:

```
sudo systemctl daemon-reload
sudo systemctl start pulpcore* --all
sudo systemctl start nginx
```