# Using Automation Dashboard

This guide describes how to install and configure Automation Dashboard to evaluate automation usage across your environments and the savings associated with it.

## Providing feedback on Red Hat documentation

If you have a suggestion to improve this documentation, or find an error, you can contact technical support at https://access.redhat.com to open a request.

# 1. View key usage metrics with Automation Dashboard

By effectively using Automation Dashboard, you can gain valuable insights into your Ansible Automation Platform usage and drive continuous improvement in your automation practices.

## 1.1. About the Automation Dashboard

The Automation Dashboard utility is a web-based container application that provides key metrics related to **job execution**, **efficiency**, and the **value derived from automation**.

Automation Dashboard uses automation metrics to supply automation usage data from Ansible Automation Platform. This data helps you compare the cost of performing tasks manually to the cost of performing tasks through automation, allowing you to show what savings are achievable through automation.

### 1.1.1. Key benefits

Automation Dashboard helps you:

- **Get a clear overview** of the automation occurring in your environment.
- **Track metrics** such as time saved and errors reduced, to quantify the benefits of automation.
- **Analyze job execution times and failure rates** to pinpoint areas for automation improvement.
- **Use the generated data** to make informed decisions about automation strategy, resource allocation, and prioritization of automation projects.

## 1.2. Installing Automation Dashboard

Install Automation Dashboard to collect and analyze key metrics related to job execution, efficiency, and automation savings across your Ansible Automation Platform deployments.

*Prerequisites*

- One of the following tested configurations:
  - RHEL 9 x86 or ARM based physical or virtual host.
  - With an external database: PostgreSQL v15 database.

  **IMPORTANT** Do not attempt to install Automation Dashboard on the same host(s) as Ansible Automation Platform.

- Automation Dashboard installation has been tested with the following configuration:
  - 80 GB hard drive (depending on data growth)
  - 4 vCPUs x 16 GB RAM
  - Disk IOPS - 3000
  - Handle up to 10,000 jobs/month and 47M summaries/month
  - Connects to three Ansible Automation Platform deployments of the same version
- Access to *baseos* and *Ansible Automation Platformstream* repository packages for the RHEL 9 host.
- A non-root login account to the RHEL 9 host for installation.
  - This requires passwordless `sudo` access to root.
  - By default, we use the `$HOMEDIR` of the user account.
- URL details for access to your Ansible Automation Platform instances.
- An Ansible Automation Platform OAuth2 token, which is used for communication between the Ansible Automation Platform instances and Automation Dashboard.
- Access to download the installation bundle providing installation components for the Automation Dashboard.
- Open firewall access to allow for bidirectional communication between AAP instances and the Automation Dashboard.
  - This includes HTTPS/443 (or your Ansible Automation Platform configured port) from the dashboard to the Ansible Automation Platform instance(s).
  - Port 8447 is the default ingress port for the Automation Dashboard. This port can be configured during installation.
  - RHEL firewall ports that might block 5432 to PostgreSQL.
- A supported version of `ansible-core` installed on supported RHEL versions.

*Procedure*

1. Download the latest installer tar file from access.redhat.com. Navigate to Downloads > Red Hat Ansible Automation Platform Product Software.
2. Copy the installation source file to your RHEL 9 host.
3. Extract the installation source. This will require ~500Mb. of disk space. Throughout this example we will use the ec2-user home directory: `/home/<username>`.

```
tar -xzvf ansible-automation-dashboard-containerized-setup-bundle-0.1-x86_64.tar.gz
cd ansible-automation-dashboard-containerized-setup/
```

4. Verify that the installation of necessary software by running the following commands:

```
cd ansible-automation-dashboard-containerized-setup
sudo dnf install ansible-core
ansible-galaxy collection install -r requirements.yml
```

5. Create an application `client_id/client_secret` in your Ansible Automation Platform instance:
   a. Create an OAuth2 application using the following steps :
      i. **For Ansible 2.4**:
         - Navigate to https://AAP_CONTROLLER_FQDN/#/applications
      ii. **For Ansible 2.5 and 2.6**:
         - Navigate to https://AAP_GATEWAY_FQDN/access/applications
      iii. Add the following information:
         - **Name**: automation-dashboard-sso
         - **Authorization grant type**: authorization-code
         - **Organization**: Default
         - **Redirect URIs**: https://AUTOMATION_DASHBOARD_FQDN/auth-callback
         - **Client type**: Confidential

      **NOTE** The values for **Name**, **Organization**, and HTTPS port number for Ansible Automation Platform are configurable. The examples provided in this document assume use of port 443.

   b. Save the `client_id` and `client_secret information` inputs into the inventory file.
   c. Create an Ansible Automation Platform access token:
      i. Navigate to https://AAP_GATEWAY_FQDN/#/users/<id>/tokens, and create a token using the following information:
         - OAuth application: automation-dashboard-sso
         - Scope: read
      ii. Store this access token value. `clusters.yaml` uses this access token.

6. Copy the example inventory and change it before running the installation program.

```
cp -i inventory.example inventory
vi inventory
```

**IMPORTANT**

- This is an example tested inventory containing default values for Ansible Automation Platform 2.4, 2.5, and 2.6.
- You must change the following values to use this inventory configuration in your environment:
  - Change the RHEL 9 host occurrences from `host.example.com` to your FQDN host
  - Change the phrase `TODO` to match your passwords within all `_admin_password` or `_pg_password` values.
- For more information, see the Inventory variables section of this document.

```
# This is our Automation Dashboard front-end application
[automationdashboard]
host.example.com ansible_connection=local

# These are required vars for the installation and should not be removed
[automationdashboard:vars]
# Configure AAP OAuth2 authentication.
# aap_auth_provider_name - name as shown on login page.
aap_auth_provider_name=Ansible Automation Platform
# aap_auth_provider_protocol - http or https
aap_auth_provider_protocol=https
# AAP version - 2.4, 2.5 or 2.6
aap_auth_provider_aap_version=2.5
# aap_auth_provider_host - AAP IP or DNS name, with optional port
aap_auth_provider_host=my-aap.example.com
# aap_auth_provider_check_ssl - enforce TLS check or not.
aap_auth_provider_check_ssl=true
# aap_auth_provider_client_id and aap_auth_provider_client_secret -
# they are obtained from AAP when OAuth2 application is created in AAP.
aap_auth_provider_client_id=TODO
aap_auth_provider_client_secret=TODO


# Specify amount of old data to synchronoize after installation.
# The initial_sync_days=N requests N days of old data, counting from "today".
# The initial_sync_since requests data from the specified data until "today".
# If both are specified, the initial_sync_since will be used.
initial_sync_days=1
# initial_sync_since=2025-08-08

# Hide warnings when insecure https request are made.
# Use this if your AAP uses self-signed TLS certificate.
# show_urllib3_insecure_request_warning=False

# Force clean install-like
# dashboard_update_secret=true
# HTTP/HTTPS settings
# nginx_disable_https=true
# Change nginx_http_port or nginx_https_port if you want to access dashboard on a
different TCP port.
# nginx_http_port=8083
# nginx_https_port=8447
# TLS certificate configuration
# The dashboard_tls_cert needs:
#   - contain server certificate, intermediate CA certificates and root CA
certificate.
#   - the server certificate must be the first one in the file.
# dashboard_tls_cert=/path/to/tls/dashboard.crt
# dashboard_tls_key=/path/to/tls/dashboard.key

# Enable Django DEBUG.
# django_debug=True

[database]
host.example.com ansible_connection=local

[all:vars]
postgresql_admin_username=postgres
postgresql_admin_password=TODO

# AAP Dashboard - mandatory
# --------------------------
dashboard_pg_containerized=True
dashboard_admin_password=TODO
dashboard_pg_host=host.example.com
dashboard_pg_username=aapdashboard
dashboard_pg_password=TODO
dashboard_pg_database=aapdashboard
#
bundle_install=true
# <full path to the bundle directory>
bundle_dir='{{ lookup("ansible.builtin.env", "PWD") }}/bundle'
```

7. Run the installation program.

```
ansible-playbook -i inventory ansible.containerized_installer.dashboard_install
```

*Verification*

For reference, see the following example output:

```
PLAY RECAP
*************************************************************************************
*******************************************************
ec2-54-147-26-173.compute-1.amazonaws.com : ok=126  changed=51   unreachable=0
```

```
failed=0    skipped=42   rescued=0    ignored=0
localhost                  : ok=12   changed=0    unreachable=0    failed=0
skipped=9    rescued=0    ignored=0
```

Alternative configurations are possible (for example, the database for Automation Dashboard can be set on a different host). This requires additional changes to variables in the inventory file. Consult the Inventory variables section of this document for available variables.

## 1.3. Integrating Automation Dashboard with your Ansible Automation Platform

Integrate your Ansible Automation Platform instances into the Automation Dashboard configuration to collect and visualise data and gain insights into your automation.

*Prerequisites*

- You have installed Automation Dashboard.
- You have verified that Automation Dashboard is running on HTTPS port 8447 on your Red Hat Enterprise Linux host.

> **NOTE**
> - This verification requires your Ansible Automation Platform login details.
> - Port 8447 is enabled by default, but this is configurable.

*Procedure*

1. Configure a personal access token. For more information, see Configuring access to external applications with token-based authentication.
2. Create or update the `clusters.yaml` file. You must include the `refresh_token`, `client_id`, and `client_secret` to enable persistent authentication and automatic token rotation.

```
clusters:
  - protocol: https
    address: <aap_controller_url>
    port: 443
    access_token: <access_token_string>
    refresh_token: <refresh_token_string>
    client_id: <client_id_string>
    client_secret: <client_secret_string>
    verify_ssl: false
    name: <unique_cluster_name>
```

3. You can add one or more Ansible Automation Platform instances (of the same Ansible Automation Platform version) into the Automation Dashboard configuration for pulling and combining data by using the following:

> **NOTE** If you only have one Ansible Automation Platform instance, then remove the second entry.

```
---
clusters:
  - protocol: https         <--- Normally https
    address: my-aap.example.com  <--- Can use IP or FQDN without http(s)://
    port: 443               <--- Normally 443
    access_token: sampleToken   <--- Your preconfigured Ansible Automation Platform
read access token
  Platform read access token
    refresh_token: myRefreshToken
    client_id: myClientID
    client_secret: myClientSecret
    verify_ssl: false       <--- Can be used when using self signed certs
    sync_schedules:
      - name: Every 5 minutes sync
        rrule: DTSTART;TZID=Europe/Ljubljana:20250630T070000
FREQ=MINUTELY;INTERVAL=5
        enabled: true

  - protocol: https
    address: aap2.example.com
    port: 443
    access_token: WRn2swiqg5spEwUndDkrJoCeg4Qwuw
    verify_ssl: true
    sync_schedules:
      - name: Every 5 minutes sync
        rrule: DTSTART;TZID=Europe/Ljubljana:20250630T070000
FREQ=MINUTELY;INTERVAL=5
        enabled: true
```

**NOTE** The access_token, refresh_token, and client_secret are stored in the Automation Dashboard database. These values are encrypted for security.

4. Load and activate your Automation Dashboard configuration:

```
podman cp clusters.yaml automation-dashboard-web:/
podman exec -it automation-dashboard-web /venv/bin/python manage.py setclusters
/clusters.yaml
```

**NOTE** **Token Rotation:** The Automation Dashboard rotates tokens internally for security but does not update your local clusters.yaml file. If you must re-run the setclusters command later, your defined tokens may be expired.

To retrieve the current valid tokens, run the following command and save the output to a new file:

```
podman exec -it automation-dashboard-web /venv/bin/python manage.py
```

```
getclusters --decrypt > clusters_current.yaml
```

### 1.3.1. Synchronizing data to Automation Dashboard

Synchronize data from your connected Ansible Automation Platform clusters to the Automation Dashboard to view the latest automation metrics.

*Prerequisites*

- You have installed Automation Dashboard.
- You have configured the `clusters.yaml` file with your platform details.

*Procedure*

1. Identify the running Automation Dashboard container:

   ```
   $ podman ps | grep automation-dashboard-web
   ```

2. Run the synchronization command using one of the following methods:

   - **Interactive mode:** Run `syncdata` without arguments. You must confirm the synchronization request when prompted.

     ```
     $ podman exec -it automation-dashboard-web /venv/bin/python manage.py syncdata
     ```

   - **Noninteractive mode:** Use the `--since` and `--until` flags to define the date range. This method bypasses the user prompt and is required for automated scripts or cron jobs.

     ```
     $ podman exec -it automation-dashboard-web /venv/bin/python manage.py syncdata
     --since=2024-01-01 --until=2024-06-30
     ```

*Verification*

1. Verify that the terminal displays the success message: `Successfully created Sync task for Cluster <cluster_url>`.
2. Refresh Automation Dashboard in your browser to view the updated metrics.

### 1.3.2. Verifying cluster access tokens

After you configure and load your cluster data, verify the stored access tokens for debugging purposes.

*Procedure*

1. Use the `getclusters` management command with the `--decrypt` option to display the stored `access_token` and `refresh_token` in plain text.
2. Run the following command inside the `automation-dashboard-web` container:

```
podman exec -it automation-dashboard-web /venv/bin/python ./manage.py getclusters
--decrypt
```

3. Review the output to confirm that the stored tokens are correct and up-to-date.

*Example*

```
clusters:
  - protocol: https
    address: my-aap.example.com
    port: 443
    access_token: sampleToken
    refresh_token: myRefreshToken
    client_id: myClientID
    client_secret: myClientSecret
    verify_ssl: false
    sync_schedules:
      - name: Every 5 minutes sync
        rrule: DTSTART;TZID=Europe/Ljubljana:20250630T070000 FREQ=MINUTELY;INTERVAL=5
        enabled: true
```

**NOTE** Displaying the encrypted `access_token` and `refresh_token` in plain text for debugging requires the `--decrypt` flag. Do not use this command on unsecured systems.

You can write output produced by `./manage.py getclusters --decrypt` to a file `clusters.yaml` and use it as input for `./manage.py setclusters clusters.yaml`.

*Verification*

If you come across error messages during installation, consult the following table:

| Issue | Possible Cause | Solution |
|---|---|---|
| 401 error | This is an unauthorized access message indicating authentication errors such as wrong credentials or tokens. | Verify that your access token is correct in `clusters.yaml` |

| | | |
|---|---|---|
| 401 error | A temporary 401 error is expected behavior when the token expires, followed immediately by trying to refresh. | If the automatic token refresh fails (for example, due to invalid `client_secret` or `refresh_token`), use the `getclusters --decrypt` command to manually verify that the credentials stored in the database match those in your source `clusters.yaml` file. If they do not match, re-run the `setclusters` command with the correct configuration. You can only use the refresh token once. If you need to execute `setclusters` because of invalid access token, create new access and refresh tokens, and use them in your source `clusters.yaml`. |
| 404 error | This is a “not found” message indicating that something is not configured correctly or pointing to the correct endpoint. | Verify that your Ansible Automation Platform instance URLs used in `clusters.yaml` are correct. |

A successful installation should be running the following three container services:

```
podman ps --all --format "{{.Names}}"

postgresql
automation-dashboard-task
automation-dashboard-web
```

You can check your container logs by running the following:

```
journalctl CONTAINER_NAME=container (where container equals one of postgresql
automation-dashboard-task or automation-dashboard-web)

For example:
journalctl CONTAINER_NAME=automation-dashboard-task
May 22 13:02:07 automation-dashboard automation-dashboard-task[1607]: [wait-for-
migrations-dashboard.sh] Waiting for database migrations...
May 22 13:02:07 automation-dashboard automation-dashboard-task[1607]: [wait-for-
migrations-dashboard.sh] Attempt 1
May 22 13:02:10 automation-dashboard automation-dashboard-task[1607]: INFO 2025-05-22
13:02:10,636 periodic 2 140568371550016 Starting sync task.
May 22 13:02:10 automation-dashboard automation-dashboard-task[1607]: INFO 2025-05-22
13:02:10,636 periodic 2 140568371550016 Retrieving clusters inform>
May 22 13:02:10 automation-dashboard automation-dashboard-task[1607]: INFO 2025-05-22
13:02:10,747 periodic 2 140568371550016 Retrieved 1 clusters.
May 22 13:02:10 automation-dashboard automation-dashboard-task[1607]: INFO 2025-05-22
13:02:10,761 periodic 2 140568371550016 Retrieving data from clust>
```

```
May 22 13:02:10 automation-dashboard automation-dashboard-task[1607]: INFO 2025-05-22
13:02:10,761 connector 2 140568371550016 Checking Ansible Automation Platform version
at h>
May 22 13:02:10 automation-dashboard automation-dashboard-task[1607]: INFO 2025-05-22
13:02:10,761 connector 2 140568371550016 Checking if is Ansible Automation Platform
2.5 at>
May 22 13:02:10 automation-dashboard automation-dashboard-task[1607]: INFO 2025-05-22
13:02:10,761 connector 2 140568371550016 Pinging api https://ec2-3>
May 22 13:02:10 automation-dashboard automation-dashboard-task[1607]: INFO 2025-05-22
13:02:10,761 connector 2 140568371550016 Executing GET request to >
May 22 13:02:13 automation-dashboard automation-dashboard-task[1607]: ERROR 2025-05-22
13:02:13,820 connector 2 140568371550016 GET request failed with >
May 22 13:02:13 automation-dashboard automation-dashboard-task[1607]: INFO 2025-05-22
13:02:13,821 connector 2 140568371550016 Checking if is Ansible Automation Platform
2.4 at>
May 22 13:02:13 automation-dashboard automation-dashboard-task[1607]: INFO 2025-05-22
13:02:13,821 connector 2 140568371550016 Pinging api https://ec2-3>
May 22 13:02:13 automation-dashboard automation-dashboard-task[1607]: INFO 2025-05-22
13:02:13,821 connector 2 140568371550016 Executing GET request to >
May 22 13:02:16 automation-dashboard automation-dashboard-task[1607]: ERROR 2025-05-22
13:02:16,892 connector 2 140568371550016 GET request failed with ...
```

The following log snippet shows a successful token refresh:

**NOTE** This log snippet omits timestamps and hostname for brevity.

**Example**

```
journalctl CONTAINER_NAME=automation-dashboard-task
INFO Checking if is AAP 2.5 ... 2.6 at https://app.example.com:443
INFO Pinging api https://app.example.com:443/api/gateway/v1/ping/
INFO Detected AAP version AAP 2.6 at https://app.example.com:443
INFO Executing GET request to
https://app.example.com:443/api/controller/v2/organizations/?page_size=100&page=1
ERROR GET request failed with status 401
INFO Token refresh POST request succedded with status 201
ERROR GET after reauth response.status_code=200
INFO Executing GET request to
https://app.example.com:443/api/controller/v2/job_templates/?page_size=200&page=1
Executing GET request to
https://app.example.com:443/api/controller/v2/jobs/?page_size=100&page=1&order_by=fini
shed&finished__gt=2025-10-23T13:01:09.768681Z
```

Check how the services are running by using `systemd`:

```
systemctl status --user

 automation-dashboard
```

```
     State: running
     Units: 76 loaded (incl. loaded aliases)
      Jobs: 0 queued
    Failed: 0 units
     Since: Thu 2025-05-22 13:02:07 UTC; 22min ago
   systemd: 252-51.el9
    CGroup: /user.slice/user-1000.slice/user@1000.service
            ├─app.slice
            │ ├─automation-dashboard-task.service
            │ │ └─1607 /usr/bin/conmon --api-version 1 -c
84e46532e8ca31b0cadb037479289d030103aa01b7a1591e62b83b17f031e47d -u
84e46532e8ca31b0cadb037479>
            │ ├─automation-dashboard-web.service
            │ │ └─1608 /usr/bin/conmon --api-version 1 -c
d060f3e3fb2b4c4c5c588149253beed83c78ccc9c9a8c1bf4c96157142a210dc -u
d060f3e3fb2b4c4c5c58814925>
            │ ├─dbus-broker.service
            │ │ ├─1621 /usr/bin/dbus-broker-launch --scope user
            │ │ └─1624 dbus-broker --log 4 --controller 9 --machine-id
612db98503014199bfd8c788c8d3da58 --max-bytes 100000000000000 --max-fds 2500000000>
            │ └─postgresql.service
            │   └─1614 /usr/bin/conmon --api-version 1 -c
eec61745cb6fc3a89a4f7475d7ef63b5899699157d943c2f16a3243311927bef -u
eec61745cb6fc3a89a4f7475d7>
            ├─init.scope
            │ ├─1093 /usr/lib/systemd/systemd --user
            │ └─1128 "(sd-pam)"
            └─user.slice
              ├─libpod-
84e46532e8ca31b0cadb037479289d030103aa01b7a1591e62b83b17f031e47d.scope
              │ └─container
              │   ├─1619 /usr/bin/dumb-init -- /usr/bin/launch_dashboard_task.sh
              │   └─1681 /venv/bin/python periodic.py
              ├─libpod-
d060f3e3fb2b4c4c5c588149253beed83c78ccc9c9a8c1bf4c96157142a210dc.scope
              │ └─container
              │   ├─1617 /usr/bin/dumb-init -- /usr/bin/launch_dashboard_web.sh
              │   ├─1646 /usr/bin/python3.9 /usr/local/bin/supervisord -c
/etc/supervisord_dashboard_web.conf
              │   ├─1877 /bin/bash /usr/local/bin/stop-supervisor
              │   ├─1878 "nginx: master process nginx -g daemon off;"
              │   ├─1879 /venv/bin/uwsgi /etc/tower/uwsgi.ini
              │   ├─1880 "nginx: worker process"
              │   ├─1881 "nginx: worker process"
              │   ├─1882 "nginx: worker process"
              │   ├─1883 "nginx: worker process"
              │   ├─1884 /venv/bin/uwsgi /etc/tower/uwsgi.ini
              │   ├─1885 /venv/bin/uwsgi /etc/tower/uwsgi.ini
              │   ├─1886 /venv/bin/uwsgi /etc/tower/uwsgi.ini
              │   ├─1887 /venv/bin/uwsgi /etc/tower/uwsgi.ini
              │   └─1888 /venv/bin/uwsgi /etc/tower/uwsgi.ini
              ├──libpod-
eec61745cb6fc3a89a4f7475d7ef63b5899699157d943c2f16a3243311927bef.scope
              │  └──container
              │    ├──1623 postgres
              │    ├──1869 "postgres: logger "
              │    ├──1871 "postgres: checkpointer "
              │    ├──1872 "postgres: background writer "
              │    ├──1873 "postgres: walwriter "
              │    ├──1874 "postgres: autovacuum launcher "
              │    ├──1875 "postgres: stats collector "
              │    ├──1876 "postgres: logical replication launcher "
              │    └──1889 "postgres: aapdashboard aapdashboard 172.31.28.99(39338)
idle"
              └──podman-pause-b6c4e853.scope
                └──1359 catatonit -P
```

## 1.4. Uninstalling Automation Dashboard

Uninstall Automation Dashboard and its dependencies by using a single command, ensuring a clean removal from your host.

*Procedure*

- Run the following command to uninstall Automation Dashboard and its dependencies, including the PostgreSQL database container:

  ```
  ansible-playbook -i inventory
  ansible.containerized_installer.dashboard_uninstall
  ```

## 1.5. Filter and save automation data for reporting

Automation Dashboard provides filtering options to analyze your Ansible Automation Platform automation runs. You can select one or more filtering options to customize your report, select a time period and a currency, and save your report to your Automation Dashboard.

### 1.5.1. Filters

Use the following filtering options to customize your report:

- **Template:** select one or more Job Templates
- **Organization:** select one or more Organizations
- **Project:** select one or more Projects
- **Label:** select one or more automation projects by label.

**NOTE**

You must preconfigure and assign labels to Ansible Automation Platform for display in Automation Dashboard. For more information on configuring labels, see Creating a job template.

### 1.5.2. Time period and currency

After selecting your filters, select a time period for analysis, and select a currency to show automation savings.

- A shorter time period is useful when considering specific automation use cases.
- A longer time period is useful when considering overall platform usage and automation growth.
- Changing from one currency to another does not convert the value of that currency. You must manually change the manual and automation cost figures to reflect whichever currency you select.

### 1.5.3. Saving a report

Use **Save as Report** to save this report to your Automation Dashboard. You can retrieve it at any time by using **Select a Report**.

## 1.6. Summary of top and overview usage

Automation Dashboard displays a summary of the top and overview usage for your selected report. This includes the following data:

- **Total number of successful jobs**: The number of automation jobs completed successfully.
- **Total number of failed jobs**: The number of automation jobs that encountered errors. Analyzing these failures can help improve efficiency.
- **Total number of unique hosts automated**: This is the number of Controller inventory records you have automated.
- **Total hours of automation**: The cumulative time that Ansible Automation Platform spent running jobs.
- **Number of times jobs were run**: The total number of individual job executions.
- **Number of hosts jobs are running on**: The total number of hosts that jobs are executed upon.
- **Top 5 projects**: The top five automation projects based on the number of running jobs.
- **Top 5 users**: The top five users of Ansible Automation Platform, with a breakdown of the total number of jobs run by each user.

NOTE Scheduled jobs can affect these results, because they do not represent a real, logged-in user.

## 1.7. Analyzing costs and savings

The costs and savings analysis compares the cost of manual automation compared to the cost of automation execution by using Ansible Automation Platform to calculate the total savings derived from automation execution.

*Procedure*

1. Use the **Average cost per minute to manually run the job** field to enter the average cost per minute for an engineer to manually run jobs.
2. Use the **Average cost per minute of running on Ansible Automation Platform** field to enter an average cost per minute of running a job using Ansible Automation Platform.
3. Select **Time taken to create automation into calculation** to include the costs associated with creating the initial or ongoing automation execution.

   Automation Dashboard supplies the following data:

   - **Cost of manual automation**: This number represents the estimated cost of manually performing all of the automated tasks. This is an estimated value, not an actual expenditure. It represents the potential expenses that the organization would incur without automation. The calculation is based on the time taken to manually run each job, multiplied by a labor cost rate.
   - **Cost of automated execution**: This is the cost of automation execution using Ansible Automation Platform. This cost includes the resources consumed by Ansible Automation Platform, such as server time, processing power, and any other operational expenses associated with automation execution. It represents the actual cost incurred for automation execution.
   - **Total savings/cost avoided**: This is the difference between the **Cost of manual automation** and the **Cost of automated execution**. This is a key metric for demonstrating the return on investment attainable by using Ansible Automation Platform.
   - **Total hours saved/avoided**: This figure is calculated by adding host executions and automation creation time, then subtracting the running time in minutes.
   - **Time taken to manually execute (min)**: This metric represents the amount of time it would take for a user to perform the task manually on a host. It is an input provided to compare the value of manual execution and automated execution using the time taken by the organization to manually automate.

**NOTE** You can export the data from your cost and savings analysis as a CSV.

## 1.8. Cost and savings analysis metrics

The costs and savings analysis generates the following metrics to quantify the return on investment (ROI) derived from automation execution.

*Table 1. Costs and Savings Output Metrics*

| Metric | Description |
|---|---|
| **Cost of manual automation** | The estimated cost of manually performing all automated tasks. This estimated value is based on the time taken to manually run each job, multiplied by a labor cost rate. |

| Metric | Description |
|---|---|
| **Cost of automated execution** | The actual cost incurred for automation execution using Ansible Automation Platform. This includes the resources consumed by Ansible Automation Platform, such as server time and processing power. |
| **Total savings/cost avoided** | The difference between the **Cost of manual automation** and the **Cost of automated execution**. This is a key metric for demonstrating the return on investment. |
| **Total hours saved/avoided** | The figure calculated by adding host executions and automation creation time, then subtracting the running time in minutes. |
| **Time taken to manually execute (min)** | The amount of time it would take for a user to perform the task manually on a host. This input compares the value of manual execution and automated execution. |

# 2. Appendix

The inventory variables required by the Automation Dashboard installer are described in the following table:

## 2.1. Inventory variables

The following variables control how Automation Dashboard interacts with remote hosts.

| Inventory variable | Description |
|---|---|
| `aap_auth_provider_name` | Natural language name shown on the login page. Default: `Ansible Automation Platform` |
| `aap_auth_provider_protocol` | Enter http or https |
| `aap_auth_provider_aap_version` | Enter one value - the version of Automation Dashboard , valid values are 2.4, 2.5 or 2.6 |
| `aap_auth_provider_host` | Ansible Automation Platform IP or DNS name, with optional port |
| `aap_auth_provider_check_ssl` | Enforce TLS check or not |
| `aap_auth_provider_client_id` | Ansible Automation Platform OAuth2 application `client_id`. Required for SSO authentication. |
| `aap_auth_provider_client_secret` | Ansible Automation Platform OAuth2 application `client_secret` |
| `initial_sync_days` | Requests x number of days of old data, counting from "today" |
| `initial_sync_since` | Requests data from the specified data until "today" |
| `dashboard_update_secret` | Forces regeneration of autogenerated Podman secrets. Store the password for database access by using a Podman secret. Set `dashboard_update_secret` to true if you changed the `dashboard_pg_password` in inventory. |
| `nginx_disable_https` | Enables use of http instead of https for Automation Dashboard |

| Inventory variable | Description |
|---|---|
| nginx_http_port | Configures the HTTP port for Automation Dashboard |
| nginx_https_port | Configures the HTTPS port for Automation Dashboard |
| dashboard_tls_cert | TLS server certificate for dashboard |
| dashboard_tls_key | TLS server certificate key for dashboard |
| postgresql_admin_username | Admin username to access PostgreSQL database |
| postgresql_admin_password | Admin password to access PostgreSQL database |
| registry_username | Username used to pull container images from registry.redhat.io. Building the bundled installer requires this variable.<br><br>**NOTE** End users can omit this variable. |
| registry_password | Password used to pull container images from registry.redhat.io. Building the bundled installer requires this variable.<br><br>**NOTE** End users can omit this variable. |
| dashboard_pg_containerized | Configures the installation program to install and configure the database as a container on the same host as Automation Dashboard. True is also the only supported value. |
| dashboard_admin_password | The password for the Automation Dashboard administrator user. The username is always admin. |
| dashboard_pg_host | The hostname or IP address of the database host. |
| dashboard_pg_username | The database user for Automation Dashboard. |
| dashboard_pg_password | The password for the dashboard_pg_username. |
| dashboard_pg_database | The database schema name for Automation Dashboard. |
| bundle_install | Indicates the required container images are already included in the installation bundle (tar file). It must be true. |
| bundle_dir | The directory where the installation bundle unpacks + /bundle (for example: /home/<username>/ansible-automation-dashboard-containerized-setup/bundle). The default value is relative to the current directory (PWD) and should work without modification. |