Red Hat

Integrate

Ansible Automation Platform 2.6

May 12, 2026

Contents

1 Integrate

Integrate with IBM HashiCorp Terraform

Learn about the supported integrations between IBM HashiCorp products and Red Hat Ansible Automation Platform, the integration workflows, and migration paths to help determine the best options for your environment.

About the Terraform integration

Many organizations find themselves using both Ansible Automation Platform and Terraform Enterprise or HCP Terraform, recognizing that these can work in harmony to create an improved experience for developers and operations teams.

While Terraform Enterprise and HCP Terraform excel at Infrastructure as Code (IaC) for provisioning and de-provisioning cloud resources, Ansible Automation Platform is a versatile, all-purpose automation solution ideal for configuration management, application deployment, and orchestrating complex IT workflows across diverse domains.

This integration directly addresses common challenges such as managing disparate automation tools, ensuring consistent configuration across hybrid cloud environments and accelerating deployment cycles. By bringing together Terraform's declarative approach to infrastructure provisioning with Ansible Automation Platform's procedural approach to configuration and orchestration, users can achieve:

- **Optimized costs:** Reduce cloud waste, minimize manual processes, and combat tool sprawl. This integration can lead to a significant reduction in infrastructure costs and a high return on investment.
- **Reduced risk:** Lower the risk of breaches, enforce policies, and significantly decrease unplanned downtime. The ability to review Terraform plan output before applying it in a workflow, with approval steps, enhances security and compliance.
- **Faster time to value:** Boost developer productivity and deploy new compute resources more rapidly, leading to a faster time to market. This is achieved through unified lifecycle management and automation for Day 0 (provisioning), Day 1 (configuration), and Day 2 (ongoing management) operations.

By enabling direct calls between Ansible Automation Platform and Terraform Enterprise or HCP Terraform, organizations can unlock time to value by creating combined workflows, reduce risk through enhanced product integrations, and enhance Infrastructure-as-Code with Ansible Automation Platform content and practices. This allows for unified lifecycle management, enabling tasks from initial provisioning and configuration to ongoing health checks, incident response, patching, and infrastructure optimization.

Integration options for Terraform

Depending on your existing setup, you can integrate these products from Ansible Automation Platform or from Terraform. Migration paths are provided for community users and for migrating from the `cloud.terraform` collection to `hashicorp.terraform`.

Ansible-initiated workflow

Ansible automation hub collections allow Ansible Automation Platform users to leverage the Terraform Enterprise or HCP Terraform provisioning capabilities.

hashicorp.terraform collection

This collection provides API integration between Ansible Automation Platform and Terraform Enterprise or HCP Terraform. This solution works natively with Ansible Automation Platform and reduces setup complexity because it doesn't require a binary installation and it includes a default execution environment.

cloud.terraform collection

This collection provides CLI integration between Ansible Automation Platform and Terraform Enterprise or HCP Terraform. To use this collection, you must install a binary and create an execution environment.

Although this collection is supported, we recommend using the `hashicorp.terraform` collection instead to take advantage of its API capabilities.

Migration workflows

Community edition users can migrate to Terraform Enterprise or HCP Terraform, and then integrate the Ansible Automation Platform capabilities using the `cloud.terraform` (CLI) collection. However, we recommend using the `hashicorp.terraform` (API) collection instead.

If you are already using the `cloud.terraform` collection, you can migrate to `hashicorp.terraform`.

Terraform-initiated workflow

For existing Terraform Enterprise or HCP Terraform users, Terraform can directly call Ansible Automation Platform at the end of provisioning for a more seamless and secure workflow. This enables Terraform Enterprise or HCP Terraform users to enhance their immutable infrastructure automation with Ansible Automation Platform Day 2 automation capabilities and manage infrastructure updates and lifecycle events.

Use hashicorp.terraform

After installing or migrating to `hashicorp.terraform`, users must create credentials to use with job templates in Ansible Automation Platform.

Create a credential for hashicorp.terraform

Users must create a credential to use with job templates in Ansible Automation Platform.

Before you begin

- You must have a Terraform API token.

Procedure

1. Log in to Ansible Automation Platform.
2. From the navigation panel, select **Automation Execution > Infrastructure > Credentials**, and then select **Create credential**.
3. From the **Credential type** list, select the **HCP Terraform** credential type.
4. In the **Token** field, enter the Terraform API token.
5. (Optional) Edit the **Description** field and select the TF organization from the **Organization** list.
6. Click **Save credential**. You are ready to use the credential in a job template.

Related information
Terraform API tokens

Ansible-initiated workflows and patterns

After you set up authentication with Ansible Automation Platform, there are many possible Ansible-initiated workflows and patterns that you can apply.

Some workflows to consider include:

- **Performing traditional infrastructure set up.** You first configure Ansible Automation Platform to do a task that Terraform cannot manage. Then perform `terraform apply`. For example, configure Ansible Automation Platform to set up the state backend for an initial run. Or use Ansible Automation Platform to set up initial cloud credentials or users to interact with a cloud provider's API.

- **Modifying infrastructure with Terraform.** In this case, turn off Ansible monitoring for the infrastructure that you are modifying. Then perform `terraform apply` with your changes. Finally, turn monitoring back on.
- **Automating `terraform apply` based on an event.** For example, you might want to trigger an event when a ServiceNow ticket is opened or a service catalog order is placed. Set up a webhook with in the Ansible Automation Platform UI so that Terraform is able to receive the event.

Integrate with the cloud.terraform collection

When you integrate with `cloud.terraform`, you must create a credential, build an execution environment, and launch a job template in Ansible Automation Platform.

Create a credential for using cloud.terraform

You can set up credentials directly from the Ansible Automation Platform user interface. The credentials are provided to the execution environment and Ansible Automation Platform reads them from there. This eliminates the need to manually update each playbook.

Before you begin

- You must have a Terraform API token.
- Install the certified `cloud.terraform` collection from automation hub. (You need an Ansible subscription to access and download collections on automation hub.)

Procedure

1. Log in to Ansible Automation Platform.
2. From the navigation panel, select **Automation Execution > Infrastructure > Credential Types**.
3. Click **Create credential type**. The **Create Credential Type** page opens and displays the **Details** tab.
4. For the **Credential Type**, enter a name.
5. In the **Input configuration** field, enter the following YAML parameter and values:

```
fields:
  - id: token
    type: string
    label: token
    secret: true
```

6. In the **Injector configuration** field, enter the following configuration.
   - For Terraform Enterprise, the hostname is the location where you have deployed TFE:

     ```
     env:
       TF_TOKEN_<hostname>:  '{{ token }}'
     ```

   - For HCP Terraform, use:

     ```
     env:
       TF_TOKEN_app_terraform_io:   '{{ token }}'
     ```

7. To save your configuration, click **Create Credential Type** again. The new credential type is created.
8. To create an instance of your new credential type, select **Automation Execution > Infrastructure > Credentials** page, and select **Create credential**.
9. From the **Credential type**, select the name of the credential type you created earlier.
10. In the **Token** field, enter the Terraform API token.
11. (Optional) Edit the **Description** and select the TF organization from the **Organization** list.
12. Click **Save credential**.

Related information
cloud.terraform collection
Terraform API tokens
Terraform CLI configuration

Build an execution environment in Ansible Automation Platform

You must build an execution environment using the automation controller so that Ansible Automation Platform can provide the credentials necessary for using its automation features.

Before you begin

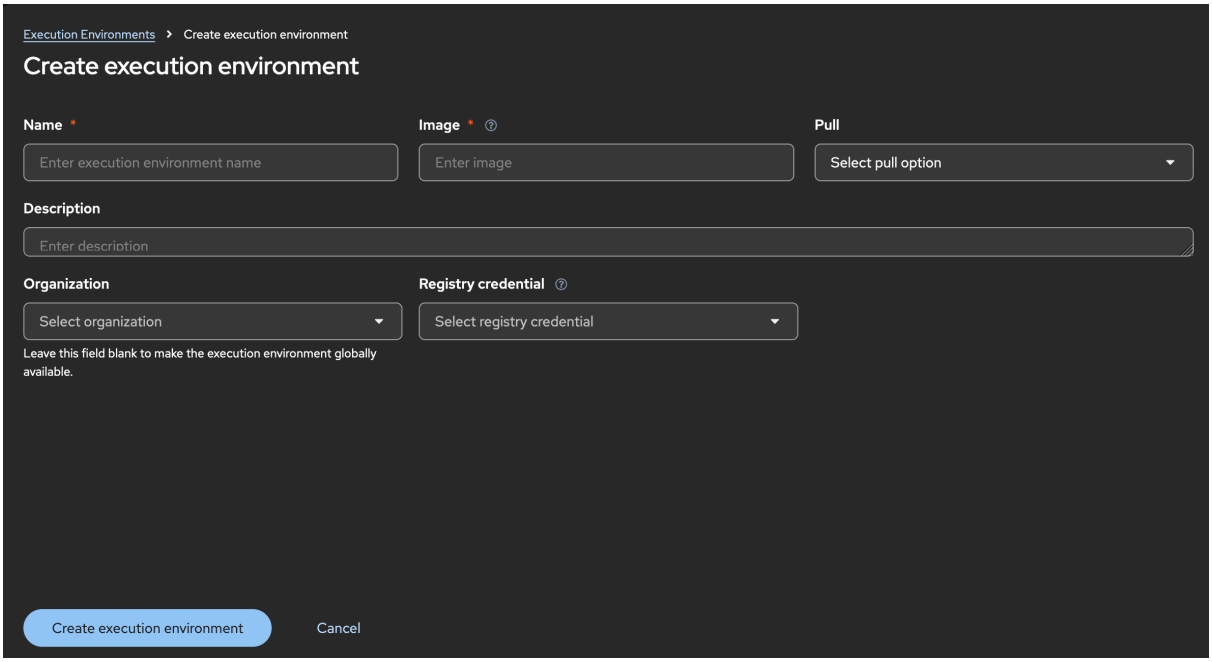
- You need a pre-existing execution environment with the latest version of `cloud.terraform` collection before you can create it using an automation controller. You cannot use the default execution environment provided by Ansible Automation Platform because the default environment does not include the `terraform` CLI binary.

> **NOTE:**
> If you have migrated from Terraform Community Edition, you can continue to use your existing execution environment and update it to the latest version of `cloud.terraform`.

- Install the `terraform` CLI binary in your pre-existing execution environment. See **Additional resources** below for a link to the binary.

Procedure

1. From the navigation panel, select **Automation Execution > Infrastructure > Execution Environments**.
2. Click **Create execution environment**.



3. For **Name**, enter a name for your Ansible Automation Platform execution environment.

4. For **Image**, enter the repository link to the image for your pre-existing execution environment.
5. Click **Create execution environment**. Your newly added execution environment is ready to be used in a job template.

Related information
CLI binary
Red Hat ecosystem catalog
Build a definition file

Create and launch a job template

Create and launch a job template to complete the integration and use the automation features in Ansible Automation Platform.

Procedure

1. From the navigation panel, select **Automation Execution > Templates**.
2. Select **Create template > Create Job Template**.
3. From the **Execution Environment** list, select the environment you created.
4. From the **Credentials** list, select the credentials instance you created previously. If you do not see the credentials, click **Browse** to see more options in the list.
5. Enter any additional information for the required fields.
6. Click **Create job template**.
7. Click **Launch template**.
8. To launch the job, click **Next** and **Finish**. The job output shows that the job has run.

Result

To see that the job has run successfully from the Terraform user interface, select **Workspaces > Ansible-Content-Integration > Run**. The Run list shows the state of the Triggered via CLI job. You can see it go from the Queued to the Plan Finished state.

Related information
Add an execution environment to a job template
Hashicorp Terraform Enterprise documentation

Migrate from `cloud.terraform` to `hashicorp.terraform`

If you are using the existing `cloud.terraform` (CLI-based) collection, you can migrate your existing playbooks to the `hashicorp.terraform` (API-based) collection. The main modules for

`hashicorp.terraform` that you must configure are `hashicorp.terraform.configuration_version` and `hashicorp.terraform.run`.

Configure the `hashicorp.terraform.configuration_version` module

To migrate to the `hashicorp.terraform` collection, you must configure the `hashicorp.terraform.configuration_version` module. This module manages configuration versions in Terraform Enterprise or HCP Terraform.

Before you begin

- Install the Ansible Automation Platform certified `hashicorp.terraform` collection.
- Verify that a valid organization and workspace are correctly set up in Terraform Enterprise or HCP Terraform.

Procedure

1. Replicate your automation tasks from the `cloud.terraform` modules.

   Example

```
- name: Create configuration version with auto_queue_runs to false
  hashicorp.terraform.configuration_version:
    workspace_id: ws-1234
    configuration_files_path: "/usr/home/tf"
    auto_queue_runs: false
    tf_validate_certs: true
    poll_interval: 3
    poll_timeout: 15
    state: present
```

2. Configure the following required parameters:

   - **`workspace_id` or `workspace` + `organization`:** The workspace ID or the workspace name and organization where the configuration version will be created and the file will be uploaded (for `state: present`).
   - **`configuration_files_path`:** The path where the required Terraform Enterprise or HCP Terraform files will be uploaded to create a configuration version (for `state: present`). The module accepts two file types for `configuration_files_path`:

    - **Directory:** Any folder containing Terraform Enterprise or HCP Terraform files. The module auto-creates the .tar.gz file from all contents recursively.
    - **.tar.gz Archive:** Pre-compressed gzip tarball. The module validates TAR format and gzip compression.
  - **`configuration_version_id`:** The configuration version ID that will be archived (`state: archived`). This action deletes the associated uploaded .tar.gz file. Note the following:
    - Only uploaded versions that were created using the API or CLI, have no active runs, and are not the current version for any workspace can be archived.
    - When the `configuration_version_id` is unspecified, Terraform Enterprise or HCP Terraform selects the latest approved `configuration_version_id` in the workspace.
  - **auto_queue_runs:** Determines if Terraform Enterprise or HCP Terraform automatically starts a run after the configuration upload (`true` by default) or requires manual initiation (`false`).
3. Set additional optional parameters as needed.

Related information
[hashicorp.terraform collection]
[Optional parameters]

Configure the `hashicorp.terraform.run` module

The `hashicorp.terraform.run` module lets you manage Terraform Enterprise or HCP Terraform runs using create, apply, cancel, and discard operations. You can trigger plans or apply operations on specified workspaces with customizable settings.

Before you begin

- Ensure that a valid Terraform API token is properly configured to authenticate with your Terraform Enterprise or HCP Terraform environment.
- Verify that a valid organization and workspace are correctly set up in Terraform Enterprise or HCP Terraform.

Procedure

1. Create a run module.

Example

```
- name: Create a destroy run with auto_apply
  hashicorp.terraform.run:
  workspace_id: ws-1234
  run_message: "destroy vpc"
  state: "present"
  tf_token: <your token>
  is_destroy: true
  auto_apply: true
  target_addrs:
    - "aws_vpc.vpc1"
    - "aws_vpc.vpc2"
  poll: true
  poll_interval: 10
  poll_timeout: 30
```

2. Configure the following required parameters:
   - **`workspace_id` or `workspace` + `organization`:** The workspace ID or the workspace name and organization where the configuration version will be created and the file will be uploaded (for `state: present`).
   - **`run_id`:** The unique identifier of the run to apply, cancel, or discard operations.
   - **`tf_token`:** The Terraform API authentication token. If this value is not set, the `TF_TOKEN` environment variable is used.
3. (Optional) Configure the built-in polling options that determine the wait period for Terraform Enterprise or HCP Terraform operations to complete:
   - **`poll: true`:** (Default) Checks the run status every `poll_interval` seconds (default: 5s) until completion or `poll_timeout` (default: 25s) is reached, returning the final status.
   - **`poll: false`:** Returns immediately after initiating the run without waiting.
4. Set additional optional parameters as needed.

Related information
Terraform API tokens
Optional parameters

Examples for `hashicorp.terraform` modules

These before and after examples help users understand how the modules can be configured in a real world environment.

Plan Only

- **Before (`cloud.terraform.terraform`):**

```
- name: Create a plan file using check mode
  cloud.terraform.terraform:
    force_init: true
    project_path: "/usr/home/tf"
    plan_file: "/usr/home/tf/terraform.tfplan"
    state: present
    check_mode: true
    check_destroy: true
    variables:
      environment: prod
```

- **After (`hashicorp.terraform.*`):**
  - The `configuration_version` module:

```
- name: Create configuration version with auto_queue_runs to false
  hashicorp.terraform.configuration_version:
    workspace_id: ws-1234
    configuration_files_path: "usr/home/tf_files"
    auto_queue_runs: false
    tf_validate_certs: true
    poll_interval: 5
    poll_timeout: 10
    state: present
```

  - The `plan_only` run with the run module:

```
- name: Create a plan only run with variables
  hashicorp.terraform.run:
   workspace_id: ws-1234
   run_message: "plan-only vpc creation"
   poll: false
   state: "present"
   tf_token: "{{ tfc_token }}"
   plan_only: true
   variables:
    - key: "env"
      value: "production"
```

Plan and apply

- **Before (`cloud.terraform.terraform`):**
  a. Generate the plan:

```
- name: Plan and Apply Workflow - Step 1 - Generate Plan
  cloud.terraform.terraform:
   force_init: true
   project_path: "/usr/home/tf"
   plan_file: "/usr/home/tf/workflow.tfplan"
   state: present
   check_mode: true
   variables:
     environment: prod
```

  b. Apply the plan:

```
- name: Plan and Apply Workflow - Step 2 - Apply Plan
  cloud.terraform.terraform:
   project_path:  "/usr/home/tf"
   plan_file: "/usr/home/tf/workflow.tfplan"
   state: present
```

- **After (`hashicorp.terraform.run`):**

a. The `configuration_version` module:

```
- name: Create configuration version with auto_queue_runs to false
  hashicorp.terraform.configuration_version:
    workspace_id: ws-1234
    configuration_files_path: "usr/home/tf_files"
    auto_queue_runs: false
    tf_validate_certs: true
    poll_interval: 5
    poll_timeout: 10
    state: present
```

b. The run module with two options for plan and apply workflow:

- **Option 1:** Uses the `auto_apply` parameter to handle both the plan and apply workflows:

```
- name: Create a run with auto_apply
  hashicorp.terraform.run:
    workspace_id: ws-1234
    run_message: "destroy vpc"
    state: "present"
    tf_token: "{{ tfc_token }}"
    auto_apply: true
    poll: true
    poll_interval: 10
    poll_timeout: 30
```

- **Option 2:** Uses two sub-steps to create a `save_plan` run and then apply it:

a. Create the plan:

```
- name: Create a save plan run
  hashicorp.terraform.run:
    workspace_id: ws-1234
    run_message: "save plan vpc creation"
    state: "present"
    tf_token: "{{ tfc_token }}"
    poll: true
    poll_interval: 10
    poll_timeout: 30
    save_plan: true
```

b. Apply the plan. You get the `run_id` from the output of the run module task:

```
- name: Apply the save plan run
  hashicorp.terraform.run:
    run_id: run-1234
    state: "applied"
    tf_token: "{{ tfc_token }}"
    poll: true
    poll_interval: 10
    poll_timeout: 30
```

Migrate from the Terraform community edition

If you want to use Ansible Automation Platform with Terraform Enterprise (TFE) or HCP Terraform and you are currently using Terraform Community Edition (TCE), you must migrate to TFE or HCP Terraform and then update Ansible Automation Platform configurations to work with TFE or HCP Terraform.

Before you begin

- Use the latest supported version of Terraform (1.11 or higher).
- Follow the `tf-migrate` CLI instructions under **Additional resources** below.
- Ensure that the HCP Terraform or TFE workspace is not set to automatically apply plans.

When you migrate from TCE to TFE or HCP Terraform, you are not migrating the collection itself. Instead, you are adapting your existing TCE usage to work with TFE or HCP Terraform.

After you migrate, you must update the Ansible Automation Platform credentials, execution environment, and job templates.

NOTE:
The `cloud.terraform` collection only supports the CLI-driven workflow in HCP Terraform.

Procedure

1. To prevent errors when running playbooks against TFE or HCP Terraform, do the following actions before running a playbook:
    a. Confirm that the Terraform version in the execution environment is the same as the version stated in TFE or HCP Terraform.
    b. Perform an initialization in TFE or HCP Terraform:

```
terraform init
```

    c. If you have a local state file in your execution environment, delete the local state file.
    d. Get a token from HCP Terraform or Terraform Enterprise, which you will use to create the credential in a later step. Ensure the token has the necessary permissions based on the team or user token to execute the desired capabilities in the playbook.
    e. Remove the backend config and files from your playbook definition.
    f. Add the workspace within the default setting in your TF config or an environment variable if you want to define the workspace outside updating the playbook itself.

    NOTE:
    You can add the workspace to your playbook to scale your workspace utilization.

2. From the Ansible Automation Platform user interface, complete the integration with `cloud.terraform`:
    a. Create a credential.
    b. Build an execution environment in Ansible Automation Platform.
    c. Create and launch a job template.
3. (Optional) After the migration is completed and verified, you can run the additional modules and plugins from the collection in your execution environment:
    - Plan Stash module
    - Terraform module
    - Output plugin

- Output lookup plugin
- State inventory plugin

Related information
Integrate with the cloud.terraform collection
Plan Stash module
Terraform module
Output plugin
Output lookup plugin
State inventory plugin
CLI instructions

Integrate from Terraform

If you have already provisioned your environment from Terraform Enterprise, you can use the Terraform official provider to leverage Ansible Automation Platform automation capabilities.

Configure the provider

You must configure the provider to allow Terraform to reference and manage a subset of Ansible Automation Platform resources.

Before you begin

- You have installed and configured Terraform Enterprise or HCP Terraform.
- You have installed the latest release version of `terraform-provider-aap` from the Terraform registry.

> **NOTE:**
> The default latest version on the Terraform registry might be a pre-release version (such as 1.2.3-beta). Select a supported release version, which uses a 1.2.3 format without dashes.

- You have created a username and password or an API token for Ansible Automation Platform. Environment variables are also supported.

> **NOTE:**
> Token authentication is recommended because users can manage tokens for specific integrations (such as Terraform), limit token access, and have full control over token lifecycle.

The provider configuration belongs in the root module of a Terraform configuration. Child modules receive their provider configurations from the root module.

Procedure

1. Create a Terraform configuration (`.tf`) file. Include a `provider` block. The name given in the block header is the local name of the provider to configure. This provider should already be included in a `required_providers` block.

Example

```
 # This example creates an inventory named `My new inventory`
# and adds a host `tf_host` and a group `tf_group` to it,
# and then launches a job based on the "Demo Job Template"
# in the "Default" organization using the inventory created.
#
terraform {
  required_providers {
    aap = {
      source = "ansible/aap"
    }
  }
}


provider "aap" {
  host     = "https://AAP_HOST"
  token = "my-aap-token" # Do not record credentials directly in the Terraform configuration. Provide your token using the AAP_TOKEN environment variable.
}


resource "aap_inventory" "my_inventory" {
  name         = "My new inventory"
  description  = "A new inventory for testing"
  organization = 1
  variables = jsonencode(
    {
      "foo" : "bar"
    }
  )
}
```

```
resource "aap_group" "my_group" {
  inventory_id = aap_inventory.my_inventory.id
  name         = "tf_group"
  variables = jsonencode(
    {
      "foo" : "bar"
    }
  )
}


resource "aap_host" "my_host" {
  inventory_id = aap_inventory.my_inventory.id
  name         = "tf_host"
  variables = jsonencode(
    {
      "foo" : "bar"
    }
  )
  groups = [aap_group.my_group.id]
}


data "aap_job_template" "demo_job_template" {
  name              = "Demo Job Template"
  organization_name = "Default"
}


  # In order for passing the inventory id to the job template execution, the
Inventory on the job template needs to be set to "prompt on launch"
resource "aap_job" "my_job" {
  inventory_id    = aap_inventory.my_inventory.id
  job_template_id = aap_job_template.demo_job_template.id


  # This resource creation needs to wait for the host and group to be created
in the inventory
  depends_on = [
    aap_host.my_host,
    aap_group.my_group
```

```
  ]
}
```

2. Add the configuration arguments, as shown in the previous example. You must configure the host and credentials. A full list of supported schema is available on the Terraform registry for your `aap` provider release version.

    - **`host`:** (String) AAP server URL. Can also be configured using the `AAP_HOST` environment variable.
    - **`insecure_skip_verify`:** (Boolean) If `true`, configures the provider to skip TLS certificate verification. Can also be configured by setting the `AAP_INSECURE_SKIP_VERIFY` environment variable.
    - **`password`:** (String, Case Sensitive) Password to use for basic authentication. Ignored if the token is set. Note that hardcoded credentials are not recommended for security reasons. It is a best practice to use the `AAP_PASSWORD` environment variable instead.
    - **`timeout`:** (Number) Timeout specifies a time limit for requests made to the AAP server. Defaults to 5 if not provided. A timeout of zero means no timeout. Can also be configured by setting the `AAP_TIMEOUT` environment variable.
    - **`token`:** (String, Case Sensitive): Token to use for token authentication. Note that hardcoded credentials are not recommended for security reasons. It is a best practice to use the `AAP_TOKEN` environment variable instead.
    - **`username`:** (String) Username to use for basic authentication. Ignored if the token is set. Can also be configured by setting the `AAP_USERNAME` environment variable.
3. (Optional) You can use expressions in the values of these configuration arguments, but can only reference values that are known before the configuration is applied.
4. (Optional) You can also use an `alias` meta-argument that is defined by Terraform and is available for all provider blocks. `alias` lets you use the same provider with different configurations for different resources.

Related information
Terraform Enterprise and HCP Terraform
Terraform registry
Local names
required_providers block
Expressions
Alias configuration
Terraform State
Associate tokens with applications (2.5 and later)
Create tokens for a user (2.4)
The Module providers Meta-Argument
Module Development: Providers Within Modules

Dynamically override execution settings

As an administrator using Terraform provider, you can configure optional Prompt on Launch parameters in Ansible job and workflow templates to dynamically override default execution settings at runtime.placeholder.

This means that you can use one Ansible template with different Terraform `*.tf` files to deploy many environments. Terraform provides the values when the Ansible job or workflow runs.The Ansible playbooks and templates stored in Ansible Automation Platform remain reusable and independent of the specific Terraform configuration that provisioned them.

You must take the following steps to use Prompt on Launch:

- **Ansible UI:** Select the **Prompt on Launch** checkbox for any of the supported fields in the Ansible job or workflow template.
- **Terraform:** Set the values for the corresponding fields in the `*.tf` file that will launch jobs from that template. If the corresponding value is not set in the `*.tf` file, then the run fails and Ansible Automation Platform sends an error message.

The supported Prompt on Launch settings include:

- **Inventory:** Allows Terraform to specify the inventory that will be used by the Ansible job template.
- **Extra variables:** To use extra variables to pass data or trigger actionable workflows, provide either a JSON or YAML string.
- **Job tag:** Tags to include in the workflow job run.
- **Labels:** Labels can be used to describe a job template, such as dev or test. You can use labels to group and filter job templates and completed jobs in the display.
- **Limit:** The Prompt on Launch option restricts a job template to run against only a specific host or group of hosts within an inventory.

  NOTE: When running as part of a workflow, the workflow job template limit is used instead.

- **Skip tag:** Tags to ignore in the workflow job run.

Related information
Automation job templates
aap_job schema

Use TF Actions and Ansible Automation Platform

Use Terraform (TF) Actions with Ansible Automation Platform to trigger automated configuration after your infrastructure is provisioned.

About Terraform Actions and Ansible Automation Platform

The Terraform (TF) Actions adds an imperative `action` block to the HCL language, letting you execute steps after infrastructure is provisioned without leaving the declarative Terraform workflow. This keeps the entire infrastructure and configuration process visible in your Terraform configuration.

TF Actions can be used to trigger Ansible automation for configuration management, such as sending an event and payload to Ansible Automation Platform to configure newly provisioned virtual machines.

There are two actions implemented with the Terraform provider for Ansible Automation Platform:

- **Launch a job directly:** Runs the job as a direct, immediate execution request to Ansible Automation Platform. You must explicitly define which specific Ansible Automation Platform job template the TF Action should call.
- **Use Event-Driven Ansible:** Sends an event to Ansible Automation Platform, which then uses rulebooks to intelligently decide which playbook to run based on the event's payload. This allows for more dynamic, scalable and reactive automation.

Use TF Actions as a direct job

When you use TF Actions to launch jobs directly with Ansible Automation Platform, the process is streamlined and sequential.

Before you begin

- You have configured the AAP Terraform provider to authenticate with Ansible Automation Platform.
- You have configured the AWS Terraform provider to authenticate with Amazon Web Services.

> **NOTE:**
> The example below uses Amazon Web Services (AWS) and requires an AWS account that might incur charges. You can adapt the pattern to use a different cloud provider.

- You have job templates configured with:
  - Inventory set to **prompt on launch**.
  - A machine credential (private key) matching a public key available in a local file.

The benefit of this approach is a clean, predictable state: the Ansible job launches during the Terraform apply cycle, and Terraform receives a clear, binary status. Note that each change launches a separate job with identical configuration.

This method can be useful when you want to execute Ansible automation against newly provisioned servers. For example, last mile provisioning or applying a routine security patching job on a new host.

Procedure

1. Define the `aap_job_launch` action in your `*.tf` file.
2. Add a lifecycle job block to define which action will be invoked during the proper lifecycle event trigger.

Example

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 6.0"
    }

    aap = {
      source  = "ansible/aap"
      version = "~> 1.4.0"
    }
  }
}

provider "aap" {
  # Configure authentication as needed.
}

provider "aws" {
  region = "us-west-1"
  # Configure authentication as needed.
}
```

```
variable "public_key_path" {
  type        = string
  description = "Local path to a public key file to inject into the VM. Your AAP Job Template must have the matching private key configured as a machine credential."
}

resource "aws_key_pair" "key_pair" {
  key_name   = "aap-terraform-actions-demo-key"
  public_key = file(var.public_key_path)
}

data "aws_ami" "rhel_ami" {
  most_recent = true

  filter {
    name   = "name"
    values = ["RHEL-9*"]
  }

  filter {
    name   = "virtualization-type"
    values = ["hvm"]
  }

  owners = ["309956199498"] # Red Hat
}

resource "aws_instance" "instance" {
  ami                         = data.aws_ami.rhel_ami.id
  instance_type               = "t2.micro"
  associate_public_ip_address = true
  key_name                    = aws_key_pair.key_pair.key_name
}

# Look up Organization ID
```

```
data "aap_organization" "organization" {
  name = "Default"
}

# Create an inventory

resource "aap_inventory" "inventory" {
  name         = "Actions Demo Inventory"
  organization = data.aap_organization.organization.id
}

data "aap_job_template" "job_template" {
  name              = "Demo Job Template"
  organization_name = data.aap_organization.organization.name
}

#
# Direct job launch action example
#

resource "aap_host" "host" {
  inventory_id = aap_inventory.inventory.id
  name         = resource.aws_instance.instance.public_ip
  # Setting a value of 10 for SSH retries because terraform will mark the
  # instance 'created' before it is ready to accept connections from Ansible.
  variables = jsonencode(
    {
      "ansible_ssh_retries" : 10
    }
  )
  # Configure a job launch after the host is created in inventory
  lifecycle {
    action_trigger {
      events  = [after_create]
      actions = [action.aap_job_launch.job_launch]
    }
```

```
  }
}


action "aap_job_launch" "job_launch" {
  config {
    inventory_id        = aap_inventory.inventory.id
    job_template_id     = data.aap_job_template.job_template.id
    wait_for_completion = true
  }
}
```

3. (Required) Change the job template name and the inventory name in this example to your corresponding variables.
4. (Optional) You can set `owners` to the Red Hat RHEL image ID so that the latest image is used each time the job runs.
5. (Optional) Set additional parameters as needed. For example, you can set `wait_for_completion` to `true`, then Terraform will wait until this job is created and reaches any final state before continuing. You can also set `wait_for_completion_timeout_seconds` to control the timeout limit.
6. Update and commit the Terraform code.
7. Execute the Terraform plan and apply it.

Related information
Authenticate with Amazon Web Services
Additional parameters

Use TF Actions with Event-Driven Ansible

Event-Driven Ansible is an automation feature that allows Ansible Automation Platform to react to real-time events, instead of being triggered on a schedule or by a manual request.

Configure an event stream for TF actions

To use TF Actions with Event-Driven Ansible, you must first configure the event stream in Ansible Automation Platform. TF actions will post events to this stream.

Before you begin

- You have configured the AAP Terraform provider to authenticate with Ansible Automation Platform.
- You have configured the AWS Terraform provider to authenticate with Amazon Web Services.

> **NOTE:** The example below uses Amazon Web Services (AWS) and requires an AWS account that might incur charges. You can adapt the pattern to use a different cloud provider.

- You have an Ansible Automation Platform inventory named **EDA Actions Demo Inventory** in the **Default** organization.
- You have job templates configured with:
  - Inventory set to **EDA Actions Demo Inventory**.
  - A machine credential (private key) matching a public key available in a local file.

Procedure

1. Log in to the Ansible Automation Platform user interface.
2. Navigate to **Automation Decisions > Event Streams**.
3. Click **Create event stream**.
4. On the **Create event stream** page, edit the fields:
   - **Name:** A descriptive, unique name for your event stream (such as `Terraform provider_Events`).
   - **Organization:** Select the organization this event stream will belong to (usually `Default` or your specific organization).
   - **Event stream type:** Select the type that matches how you want to receive events. **Basic Event Stream** (username/password) is supported with this integration.
   - **Credential:** Select a credential that you have pre-created for authentication with this event stream.
   - **Headers:** (Optional) Enter comma-separated HTTP header keys that you want to include in the event payload that gets forwarded to the rulebook. Leave this empty to include all headers.
   - **Forward events to rulebook activation:** This option is typically enabled by default. Disabling it is useful for testing and diagnosing your connection and incoming data without inadvertently triggering any automation.
5. Click **Create event stream**. Then navigate to **Automation Decisions > Event Streams** to verify the event stream was created and see the number of events received so far.

   You can also click on the specific stream to see its detailed configuration, including the organization, event stream type, associated credential, and event forwarding settings.
6. Set up a rule book activation. Make sure to:

a. Add the event stream to the rulebook.
b. (Recommended) Select the **Rulebook activation enabled?** option to automatically start the activation after creation.
c. Activate the rulebook.

7. Select **Automation Decisions > Rulebook Activations** to verify that the rulebook is active and check its status.

Related information
Authenticate with Amazon Web Services
Set up a rulebook activation

Configure TF Actions

To connect the event stream to Terraform actions, you configure the main TF file (`*.tf`) in Terraform.

Procedure

1. Add a lifecycle block to call the Event-Driven Ansible event stream to your `*.tf` file. The `after_create` action will trigger the `action.aap_eda_eventstream_post.create` .

Example

The following example shows a `lifecycle` block added to the provisioning of an AWS EC2 server. After the new server is provisioned, the action runs.

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 6.0"
    }

    aap = {
      source  = "ansible/aap"
      version = "~> 1.4.0"
    }
  }
}

provider "aap" {
  # Configure authentication as needed.
```

```
}


provider "aws" {
  region = "us-west-1"
  # Configure authentication as needed.
}


variable "public_key_path" {
  type        = string
  description = "Local path to a public key file to inject into the VM. Your
AAP Job Template must have the matching private key configured as a machine
credential."
}


variable "event_stream_username" {
  type = string
}


variable "event_stream_password" {
  type = string
}


resource "aws_key_pair" "key_pair" {
  key_name   = "aap-terraform-actions-demo-key"
  public_key = file(var.public_key_path)
}


data "aws_ami" "rhel_ami" {
  most_recent = true


  filter {
    name   = "name"
    values = ["RHEL-9*"]
  }


  filter {
    name   = "virtualization-type"
```

```
    values = ["hvm"]
  }

  owners = ["309956199498"] # Red Hat
}

resource "aws_instance" "instance" {
  ami                         = data.aws_ami.rhel_ami.id
  instance_type               = "t2.micro"
  associate_public_ip_address = true
  key_name                    = aws_key_pair.key_pair.key_name
}

# Look up an inventory

data "aap_inventory" "inventory" {
  name              = "EDA Actions Demo Inventory"
  organization_name = "Default"
}

#
# EDA Event launch action example
#

resource "aap_host" "host" {
  inventory_id = data.aap_inventory.inventory.id
  name         = resource.aws_instance.instance.public_ip
  # Configure an EDA eventstream POST after the host is created in inventory
  lifecycle {
    action_trigger {
      events  = [after_create]
      actions = [action.aap_eda_eventstream_post.event_post]
    }
  }
}
```

```
data "aap_eda_eventstream" "eventstream" {
  name = "TF Actions Event Stream"
}


action "aap_eda_eventstream_post" "event_post" {
  config {
    limit             = "all"
    template_type     = "job"
    job_template_name = "Demo Job Template"
    organization_name = "Default"
    event_stream_config = {
      username = var.event_stream_username
      password = var.event_stream_password
      url      = data.aap_eda_eventstream.eventstream.url
    }
  }
}
```

2. (Required) Configure the following parameters:
    - **`event_stream_config`:** (Attributes) Details for the Event-Driven Ansible event stream. You must include:
        - **`username`:** (String) Username to use when performing the POST to the event stream URL
        - **`password`:** (String) Password to use when performing the POST to the event stream URL
        - **`url`:** (String) URL to receive the event POST
    - **`limit`:** (String) Ansible Automation Platform limit for job execution
    - **`organization_name`:** (String) Organization name
    - **`template_type`:** (String) Template type: either `job` or `workflow_job`
3. (Optional) You can set `owners` to the Red Hat RHEL image ID so that the latest image is used each time the job runs.
4. (Optional) Set additional parameters as needed.
5. Configure an action integration with event payload specifications and target rulebook mapping.

   Example:

```
- name: Dispatch TF Workflow Job Template Action
      condition: event.payload.template_type == "workflow"
      throttle:
        once_after: 1 minute
        group_by_attributes:
          - event.payload.workflow_template_name
          - event.payload.limit
          - event.payload.organization_name
      actions:
        - debug:
            msg: "Executing Workflow Template
{{ event.payload.workflow_template_name }}"
        - run_workflow_template:
            name: "{{ event.payload.workflow_template_name }}"
            organization: "{{ event.payload.organization_name }}"
        - debug:
            msg: "Executed Workflow Job Template
{{ event.payload.workflow_template_name }}"
```

Related information
Additional parameters for event stream
Terraform provider documentation
IBM Hashicorp YouTube Demo

Create and apply the plan

After you configure your Terraform plan to include Event-Driven Ansible events, you create and apply the plan to trigger the events.

Procedure

1. Run `terraform init` to initialize your working directory.
2. Use `terraform plan` to commit to create the plan. The following example also saves the plan to a file named `tfplan.out`, but you can specify any name for the plan. Saving the plan is a best practice for automation because the saved plan is strictly enforced.

```
terraform plan -out=tfplan.out
```

3. Review the plan output.

4. Apply the saved plan.

```
terraform apply tfplan.out
```

This creates and sends an event to the specified event stream. As each resource is created, TF actions are invoked and the corresponding Ansible Automation Platform playbooks are executed sequentially.

Result

1. Verify that the runs are updated in the Terraform user interface. Drill down on a resource to see that the action was invoked and a post event was executed.
2. From the Ansible Automation Platform user interface, verify that the event is successfully received by (EDAName} and triggers the appropriate rulebook activation:
   a. Check the **Event Streams** dashboard to see the TF Actions events were received.
   b. Check the **Jobs** dashboard to see the jobs running sequentially and with a **Success** status.
   c. Check the **Inventory** dashboard to see the updates. For example, if you created new servers, check the **Hosts** tab for the Terraform provisioned inventory.

Related information
Terraform workflow for provisioning infrastructure

Example rulebook

The following rulebook example shows how to use TF actions and Event-Driven Ansible to listen for events on a webhook.

```
- name: Listen for events on a webhook
  hosts: all


  ## Define our source for events


  sources:
    - ansible.eda.webhook:
        host: 0.0.0.0
        port: 5000
      filters:
        - ansible.eda.insert_hosts_to_meta:
            host_path: payload.limit
```

```
  ## Define the conditions we are looking for


  rules:
    - name: Dispatch TF Job Template Action
      condition: event.payload.template_type == "job"
      throttle:
        once_after: 1 minute
        group_by_attributes:
          - event.payload.job_template_name
          - event.payload.limit
          - event.payload.organization_name
      actions:
        - debug:
            msg: "Executing Job Template {{ event.payload.job_template_name }}"
        - run_job_template:
            name: "{{ event.payload.job_template_name }}"
            organization: "{{ event.payload.organization_name }}"
        - debug:
            msg: "Executed Job Template {{ event.payload.job_template_name }}"
    - name: Dispatch TF Workflow Job Template Action
      condition: event.payload.template_type == "workflow"
      throttle:
        once_after: 1 minute
        group_by_attributes:
          - event.payload.workflow_template_name
          - event.payload.limit
          - event.payload.organization_name
      actions:
        - debug:
            msg: "Executing Workflow Template
{{ event.payload.workflow_template_name }}"
        - run_workflow_template:
            name: "{{ event.payload.workflow_template_name }}"
            organization: "{{ event.payload.organization_name }}"
        - debug:
```

```
            msg: "Executed Workflow Job Template
{{ event.payload.workflow_template_name }}"
```

Integrate with IBM HashiCorp Vault

The integration of Red Hat Ansible Automation Platform and IBM HashiCorp Vault provides fully automated lifecycle management for Vault.

Authenticate to `hashicorp.vault`

After you install or migrate to the `hashicorp.vault` collection, authentication is configured in the Ansible Automation Platform user interface. An administrator creates a custom credential type to authenticate to Vault. Then users create credentials to use with job templates.

About the Vault integration

Vault lets you centrally store and manage secrets securely. The Ansible Automation Platform certified `hashicorp.vault` collection provides fully automated lifecycle and operation management for Vault. You can create, update, and delete secrets through playbooks.

- **Existing `community.hashi_vault` users:** The `hashicorp.vault` solution is intended to replace unsupported `community.hashi_vault` collection. Use the migration path to keep your existing playbooks.
- **New Vault users:** The `hashicorp.vault` collection is included in the supported execution environment from automation hub.

NOTE:
Although the `hashicorp.vault` and `hashi.terraform` collections work independently of each other and are designed for different tasks, you can use them together in advanced workflows.

Related information
Migrate from community.hashi_vault

Authentication architecture

The `hashicorp.vault` collection manages authentication through environment variables and client initialization. This approach enhances security by preventing sensitive credentials from being passed directly as module parameters within playbook tasks.

The `hashicorp.vault` collection injects credentials into job templates with environment variables, so you get simpler, cleaner task definitions while ensuring that authentication details remain secure.

The following authentication types are supported:

- **appRole authentication:** Use either one of the following methods when using appRole authentication:
  - Set the `VAULT_APPROLE_ROLE_ID` and `VAULT_APPROLE_SECRET_ID` environment variables. When you use environment variables, you must also create a custom credential type and credentials that will be passed to the job template.
  - Directly pass the `role_id` and `secret_id` parameters to the tasks, for example:

```
- name: Create a secret with AppRole authentication
  hashicorp.vault.kv2_secret:
    url: https://vault.example.com:8200
    auth_method: approle
    role_id: "{{ vault_role_id }}"
    secret_id: "{{ vault_secret_id }}"
    path: myapp/config
    data:
      api_key: secret-api-key
```

- **Token authentication:** Set the `VAULT_TOKEN` environment variable.
  Optionally, you can configure parameters for the token. If parameters are not provided, then the module uses environment variables.

Create a custom credential type for Vault

As an admin, you create a secure credential type in Ansible Automation Platform, which is used to authenticate to Vault.

Before you begin

Do *one* of the following:

- **New users:** Install the Ansible Automation Platform certified `hashicorp.vault` collection from Automation hub.
- **`community.hashi_vault` collection users:** Migrate from `community.hashi_vault`.

You can configure role-based (appRole) authentication or allow users to directly provide a token.

Procedure

1. Log in to Ansible Automation Platform.
2. From the navigation panel, select **Automation Execution > Infrastructure > Credential Types**.
3. Click **Create a credential type**. The **Create Credential Type** page opens.
4. Enter a name and a description in the corresponding fields.
5. If you want to configure token authentication for individual users:
    a. For **Input configuration**, enter:

```
fields:
 - id: vault_token
   type: string
   label: Hashicorp Vault Token
   secret: true
```

    b. For **Injector configuration**, enter:

```
env:
   VAULT_TOKEN: '{{ vault_token }}'
```

6. If you want to configure appRole authentication using `role_id` and `secret_id`:
    a. For **Input configuration**, enter:

```
fields:
  - id: vault_approle_role_id
    type: string
    label: Hashicorp Vault appRole Role ID
    secret: true
  - id: vault_approle_secret_id
    type: string
    label: Hashicorp Vault appRole Secret ID
    secret: true
```

b. For **Injector configuration**, enter:

```
env:
    VAULT_APPROLE_ROLE_ID: '{{ vault_approle_role_id }}'
    VAULT_APPROLE_SECRET_ID: '{{ vault_approle_secret_id }}'
```

7. Click **Create credential type**.

Next steps

- Create a custom credential

Related information
Automation hub
Migrate from community.hashi_vault
Vault API documentation

Create a custom credential

Vault users must create a custom credential to use with job templates in Ansible Automation Platform.

Before you begin

- Your administrator has created a Vault credential type.

Procedure

1. Log in to Ansible Automation Platform.

2. From the navigation panel, select **Automation Execution > Infrastructure > Credentials**, and then select **Create credential**.
3. Enter a name and a description in the corresponding fields.
4. (Optional) From the **Organization** list, select an organization.
5. From the **Credential type** list, select a Vault credential type. The fields that display depend on the credential type.
6. Do *one* of the following:
    a. For the token authentication, add your Vault token and edit any fields as needed.
    b. For the appRole authentication method, enter the IDs in the **appRole Role ID** and **appRole Secret ID** fields. Edit any other fields as needed.
7. Click **Save credential**. You are ready to use the credential in a job template.

Migrate from `community.hashi_vault`

If you are using the `community.hashi_vault` collection, you can migrate your existing playbooks to the `hashicorp.vault` collection.

Configure KV1 modules

If you are using KV1 with `community.hashi_vault` collection, configure the corresponding modules in the `hashicorp.vault` collection.

Configure the `hashicorp.vault.kv1_secret` module

Configuring this module is not required, but you can configure the defaults if needed after the migration.

- Configuring this module is not required for migration because there are no corresponding modules in `community.hashi_vault`. However, you might want to configure something other than the defaults for `auth_method` and `state` after the migration. You can use the examples on Ansible automation hub for reference.

Configure the `hashicorp.vault.kv1_secret_info` module

The `hashicorp.vault.kv1_secret_info` module reads KV1 secrets.

The corresponding community.hashi_vault modules are:

- **`community.hashi_vault.vault_kv1_get`:** Retrieves secrets from the HashiCorp Vault KV version 1 secret store.
- **`community.hashi_vault.vault_kv1_get lookup`:** Retrieves secrets from the HashiCorp Vault KV version 1 secret store.

Procedure

1. Replicate the `community.hashi_vault modules` to the following `hashicorp.vault.kv1_secret_secret_info` parameters.

```
    engine_mount_point:
      description: KV secrets engine mount point.
      default: secret
      type: str
      aliases: [secret_mount_path]
    path:
      description:
        - Specifies the path of the secret.
      required: true
      type: str
      aliases: [secret_path]
extends_documentation_fragment:
  - hashicorp.vault.vault_auth.modules
```

2. (Required) Configure the `path` parameter. This is the path to the secret in the `community.hashi_vault.hashi_vault` modules. **Alias:** `secret_path`
3. If needed, configure the optional parameters.

Next steps

- Configure the `hashicorp.vault.kv1_secret_get` lookup plugin.

Related information
Optional parameters

Configure the `hashicorp.vault.kv1_secret_get` lookup plugin

The `hashicorp.vault.kv1_secret_get` lookup plugin module reads KV1 secrets.

The corresponding `community.hashi_vault` modules are:

- **`community.hashi_vault.hashi_vault`:** Retrieves secrets from HashiCorp Vault.

- **`community.hashi_vault.vault_kv1_get lookup :`** Gets secrets from the HashiCorp Vault KV version 1 secret store.

Procedure

1. Replicate `the community.hashi_vault` modules to the following `hashicorp.vault.kv1_secret_get` parameters.

```
 auth_method:
   description: Authentication method to use.
   choices: ['token', 'approle']
   default: token
   type: str
 engine_mount_point:
   description:
     - The KV secrets engine mount point.
   default: secret
   type: str
   aliases: ['mount_point', 'secret_mount_path']
 secret:
   description:
     - The Vault path to the secret being requested.
   required: true
   type: str
   aliases: ['secret_path']
```

2. (Required) Configure the secret parameter. This maps to secret in the `community.hashi_vault.hashi_vault` modules. **Alias:** `secret_path`
3. If needed, configure the optional parameters.

Next steps

- Create a credential type

Related information
Optional parameters

Example: `hashicorp.vault.kv1_secret_info` module

The following migration example shows before and after configurations for the `hashicorp.vault.kv1_secret_info` module.

Example:

Before (community.hashi_vault)

```
- name: Read a kv1 secret from Vault (community collection)
  community.hashi_vault.vault_kv1_get:
    url: https://vault:8201
    token: "{{ vault_token }}"
    path: hello
  register: response
```

After (hashicorp.vault)

```
- name: Read a kv1 secret from Vault (hashicorp.vault collection)
  hashicorp.vault.kv1_secret_info:
    url: https://vault.example.com:8201
    token: "{{ vault_token }}"
    path: sample
```

Example: `hashicorp.vault.kv1_secret_get` lookup

The following migration example shows the KV1 secret get lookup.

Example:

Before (community.hashi_vault)

```
- name: Retrieve a secret from the Vault
  ansible.builtin.debug:
    msg: "{{ lookup('community.hashi_vault.vault_kv1_get', 'hello', url='https://
vault:8201') }}"
```

After (hashicorp.vault)

```
- name: Retrieve a secret from the Vault
  ansible.builtin.debug:
    msg: "{{ lookup('hashicorp.vault.kv1_secret_get',
                    secret='hello',
                    url='https://myvault_url:8201') }}"
```

Configure KV2 modules

If you are using KV2 with `community.hashi_vault` collection, configure the corresponding modules in the `hashicorp.vault` collection.

Configure the `hashicorp.vault.kv2_secret` module

The `hashicorp.vault.kv2_secret` module performs Create, Update, and Delete (CRUD) operations on KV2 secrets through a unified interface.

Before you begin

- Install the Ansible Automation Platform certified `hashicorp.vault` collection.

The corresponding `community.hashi_vault` modules are:

- **`community.hashi_vault.vault_kv2_write`** - Write KV2 secrets.
- **`community.hashi_vault.vault_kv2_delete`** - Delete KV2 secrets.

Procedure

1. Replicate your automation tasks from both of the `community.hashi_vault` modules to the following `hashicorp.vault.kv2_secret` parameters. The `hashicorp.vault.kv2_secret` parameters are similar to `community.hashi_vault`.

```
auth_method:
 description: Authentication method to use
 type: str
 choices: [token, approle]
 default: token
 required: false


cas:
```

```
 description: Perform a check-and-set operation.
 type: int
 required: false

data:
 description: KV2 secret data to write.
 type: dict
 required: true

engine_mount_point:
 description: The path where the secret backend is mounted.
 type: str
 default: secret
 required: false
 aliases: [secret_mount_path]

namespace:
 description: Vault namespace where secrets reside.
 type: str
 default: admin
 aliases: [vault_namespace]

path:
 description: Vault KVv2 path to be written to.
 type: str
 required: true
 aliases: [secret_path]

url:
 description: URL of the Vault service
 type: str
 aliases: [vault_address]
 required: true

versions:
 description: One or more versions of the secret to delete.
```

```
  type: list of int
  required: false


state:
  description: Desired state of the secret
  type: str
  choices: [present, absent]
  default: present
```

2. You must add the `state` parameter to the `hashicorp.vault.kv2_secret` module, as shown above. Valid options are:

    - **`present`:** This is the equivalent of `create` or `update` in the `community.hashi_vault.vault_kv2` modules.
    - **`absent`:** This is the equivalent of `delete secret` in the `community.hashi_vault.vault_kv2` modules.

Next steps

- Configure the `hashicorp.vault.kv2_secret_info` module.

Configure the `hashicorp.vault.kv2_secret_info` module

The `hashicorp.vault.kv2_secret_info` module reads KV2 secrets.

The corresponding `community.hashi_vault` module is:

- **`community.hashi_vault.vault_kv2_get`:** Gets secrets from the HashiCorp Vault KV version 2 secret store.

Procedure

1. Replicate the `community.hashi_vault` modules to the following `hashicorp.vault.kv2_secret_info` parameters.

```
    engine_mount_point:
      description: KV secrets engine mount point.
      default: secret
      type: str
      aliases: [secret_mount_path]
    path:
      description: Path to the secret.
      required: true
      type: str
      aliases: [secret_path]
    version:
      description: The version to retrieve.
      type: int
extends_documentation_fragment:
  - hashicorp.vault.vault_auth.modules
```

2. Configure the required parameters:
    - **`path`:** The path where the secret is located in the community.`hashi_vault.hashi_vault` modules. **Alias:** `secret_path`
    - **`url`:** Maps to `url` in the `community.hashi_vault.hashi_vault` modules. Uses the same aliases as `vault_address`.
3. If needed, configure the optional parameters.

Next steps

- Configure the `hashicorp.vault.kv2_secret_get` lookup plugin.

Related information
Optional parameters

Configure the `hashicorp.vault.kv2_secret_get` lookup plugin

The `hashicorp.vault.kv2_secret_get` lookup plugin module reads KV2 secrets.

The corresponding `community.hashi_vault` modules are:

- **`community.hashi_vault.hashi_vault`:** Retrieves secrets from HashiCorp Vault.
- **`community.hashi_vault.vault_kv2_get` lookup:** Gets secrets from the HashiCorp Vault KV version 2 secret store.

Procedure

1. Replicate the `community.hashi_vault` modules to the following `hashicorp.vault.kv2_secret_get` parameters.

```
 auth_method:

  description: Authentication method to use

  type: str

  choices: [token, approle]

  default: token

  required: false


mount_point:

  description: Vault mount point

  type: str

  required: false

  aliases: [secret_mount_path]


namespace:

  description: Vault namespace where secrets reside.

  type: str

  default: admin

  aliases: [vault_namespace]

secret:

  description: Vault path to the secret being requested in the format
path[:field]

  type: str

  required: true

  aliases: [secret_path]


url:

  description: URL of the Vault service

  type: str

  aliases: [vault_address]

  required: true


version:

  description: Specifies the version to return. If not set the latest is
returned.

  type: int

  required: false
```

2. Use the following guidance to configure the `hashicorp.vault.kv2_secret_get` parameters:

    - **`auth_method`:** Maps identically to `auth_method` in the `community.hashi_vault.hashi_vault` modules.
    - **`mount_point`:** Maps to `mount_point` in the `community.hashi_vault.hashi_vault` modules. **Alias:** `secret_mount_path`.
    - **`namespace`:** Maps to `namespace` in the `community.hashi_vault.hashi_vault` modules. **Alias:** `vault_namespace`.
    - **`secret`:** Maps to `secret` in the `community.hashi_vault.hashi_vault` modules.
    - **`url`:** Maps to `url` in the `community.hashi_vault.hashi_vault` modules. Uses the same aliases as `vault_address`.
    - **`version`:** Maps identically to `version` in the `community.hashi_vault.hashi_vault` modules.

Next steps

- Create a credential type

Examples: `hashicorp.vault.kv2_secret` module

The following migration examples show basic before and after configurations for the `hashicorp.vault.kv2_secret` module.

NOTE:
KV2 delete operations are `soft-delete`.

Example 1: Basic Secret Write/Create

Before (`community.hashi_vault`):

```
- name: Write/create a secret
  community.hashi_vault.vault_kv2_write:
    url: https://vault:8200
    path: hello
    data:
      foo: bar
```

After (`hashicorp.vault`):

```
- name: Write/create a secret
  hashicorp.vault.kv2_secret:
    url: https://vault:8200
    path: hello
    data:
      foo: bar
```

Example 2: Basic Secret Delete

Before (`community.hashi_vault`):

```
- name: Delete the latest version of the secret/mysecret secret.
  community.hashi_vault.vault_kv2_delete:
    url: https://vault:8201
    path: secret/mysecret
```

After (`hashicorp.vault`):

```
- name: Delete the latest version of the secret/mysecret secret.
  hashicorp.vault.kv2_secret:
    url: https://vault:8201
    path: secret/mysecret
    state: absent
```

Example 3: Secret Delete - specific version

Before (`community.hashi_vault`):

```
- name: Delete versions 1 and 3 of the secret/mysecret secret.
  community.hashi_vault.vault_kv2_delete:
    url: https://vault:8201
    path: secret/mysecret
    versions: [1, 3]
```

After (`hashicorp.vault`):

```
- name: Delete versions 1 and 3 of the secret/mysecret secret.
  hashicorp.vault.kv2_secret:
    url: https://vault:8201
    path: secret/mysecret
    versions: [1, 3]
    state: absent
```

Related information
Delete data

Examples: `hashicorp.vault.kv2_secret_info` module

The following migration examples show before and after configurations for the `hashicorp.vault.kv2_secret_info` module.

Example 1: Read a secret with token authentication

Before (community.hashi_vault)

```
- name: Read the latest version of a kv2 secret from Vault
 community.hashi_vault.vault_kv2_get:
    url: https://vault.example.com:8200
    token: "{{ vault_token }}"
    path: myapp/config
  register: response
```

After (hashicorp.vault)

```
- name: Read a secret with token authentication
  hashicorp.vault.kv2_secret_info:
    url: https://vault.example.com:8200
    token: "{{ vault_token }}"
    path: myapp/config
```

Example 2: Read a secret with a specific version

Before (community.hashi.vault)

```
- name: Read version 5 of a secret from kv2
  community.hashi_vault.vault_kv2_get:
    url: https://vault.example.com:8200
    path: myapp/config
    version: 5
```

After (hashicorp.vault)

```
- name: Read a secret with a specific version
  hashicorp.vault.kv2_secret_info:
    url: https://vault.example.com:8200
    path: myapp/config
    version: 1
```

Examples: `hashicorp.vault.kv2_secret_get` lookup

The following migration example shows the KV2 secret get lookup for retrieving the latest version.

Example:

Before (`community.hashi_vault`)

```
- name: Return latest KV v2 secret from path
  ansible.builtin.debug:
    msg: "{{ lookup('community.hashi_vault.hashi_vault', 'secret=secret/data/hello
  token=my_vault_token
  url=http://myvault_url:8200') }}"
```

After (`hashicorp.vault`)

```
name: Return latest KV v2 secret from path
  ansible.builtin.debug:
    msg: "{{ lookup('hashicorp.vault.kv2_secret_get', 'secret=secret/data/hello
  url=http://myvault_url:8200') }}"
```

Manage database secrets with hashicorp.vault

Use the `hashicorp.vault.database_connection` and `hashicorp.vault.database_connection_info` modules to manage the lifecycle of Vault database secrets. These unified modules enable a supported migration from `community.hashi_vault` for consistent, validated creation and retrieval of connection data.

Configure the hashicorp.vault.database_connection module

The hashicorp.vault.database_connection module performs **Create**, **Update**, **Delete**, and **Reset** operations on database connections for plug-ins.

The corresponding community.hashi_vault modules are:

- `vault_database_connection_configure module` : Creates and updates the database connection.
- `vault_database_connection_delete module` : Delete a database connection.
- `vault_database_connection_reset module` : Closes a connection_name and its underlying plugin and restarts it with the configuration stored.

Procedure

1. Map the parameters from your existing `community.hashi_vault` modules to the corresponding `hashicorp.vault.database_connection` parameters.

```
---

module: database_connection

short_description: Manage database secrets engine connections in HashiCorp
Vault.

version_added: 1.2.0

author: my-name

description:

  - This module manages (create, update, delete, and reset) the lifecycle of
database

    connection configurations within the HashiCorp Vault Database secrets
engine.
```

```
options:
 state:
   description:
     - Goal state for the database connection.
     - Use V(present) to create or update the connection.
     - Use V(reset) to trigger a connection reset.
     - Use V(absent) to remove the connection configuration.
   choices: [present, absent, reset]
   default: present
   type: str
 database_mount_path:
   description: Database secret engine mount path.
   type: str
   default: database
   aliases: [vault_database_mount_path]
 name:
   description: The name of the database connection configuration.
   required: true
   type: str
   aliases: [connection_name]
 username:
   description:
     - The username to connect to the database.
   required: false
   type: str
   aliases: [connection_username]
 password:
   description:
     - The password to connect to the database.
   required: false
   type: str
   aliases: [connection_password]
 disable_escaping:
   description: Determines whether special characters in the username and
password fields will be escaped.
   type: bool
```

```
    default: false
  connection_url:
    description: The connection string used to connect to the database.
    type: str
  plugin_name:
    description:
      - The name of the plugin to use for this connection.
      - Required when O(state=present).
    required: false
    type: str
  plugin_version:
    description:
      - The semantic version of the plugin to use for this connection.
    type: str
    required: false
  plugin_options:
    description:
      - Additional parameters specific to the plugin.
      - This should be a dictionary of options required by the specific
database plugin.
    type: dict
  verify_connection:
    description: Specifies if the connection is verified during initial
configuration.
    default: true
    type: bool
  allowed_roles:
    description: A list of roles authorized to use this connection.
    type: list
    elements: str
  root_rotation_statements:
    description:
      - Specifies the database statements to be executed to rotate the root
user's credentials.
      - Refer to the specific Vault database plugin documentation for supported
formatting.
    type: list
    elements: str
```

```
 password_policy:

   description:

     - The name of the password policy to use when generating passwords for
this database.

     - If not specified, Vault uses a default policy (20 characters, mixed
case, number, dash).

   type: str

extends_documentation_fragment:

 - hashicorp.vault.vault_auth.modules
```

2. Configure the following parameters:
    - **Name**: The name of the database connection configuration. Alias: connection_name
    - **plugin_name**: If 0 (state=present) you must include the name of the plugin to use for this connection.
3. **Optional:** Configure the Parameters for your `hashicorp.vault.database_connection` module.

Configure the hashicorp.vault.database_connection_info module

The `hashicorp.vault.database_connection_info` module lists the available database connections or reads the configuration for a specified connection.

The corresponding `community.hashi_vault` modules are:

- `vault_database_connection_read module`: Returns the configuration settings for a connection_name.
- `vault_database_connections_list module`: Returns a list of available connections.

Procedure

1. Replicate the `community.hashi_vault` modules to the following `hashicorp.vault.database_connection_info` parameters.

```
---

module: database_connection_info

short_description: List available connections or read configuration for a
specific connection.

version_added: 1.2.0

author: my-name

description:

  - This module retrieves configuration details for a specific Vault database
connection.

  - When a connection name is provided, it returns its full settings; if the
name is omitted,

    the module returns a comprehensive list of all available database
connections within the specified mount path.

options:

  name:

    description: The name of the database connection to read.

    required: false

    type: str

  database_mount_path:

    description: Database secret engine mount path.

    type: str

    default: database

    aliases: [vault_database_mount_path]

extends_documentation_fragment:

  - hashicorp.vault.vault_auth.modules
```

2. To retrieve the configuration details for a specific database connection, specify a name for the name parameter. If name is omitted, a list of available database connections is returned.
3. **Optional:** Configure the Parameters for your `hashicorp.vault.database_connection_info` module.

Examples for the hashicorp.vault.database_connection module

The following examples show basic configurations for the `hashicorp.vault.database_connection` module.

Example 1: Create a database connection with a PostgreSql plugin

Before (`community.hashi_vault`):

```
- name: Example 1 - Create a database connection (PostgreSQL plugin, default
database mount)
  community.hashi_vault.vault_database_connection_configure:
    url: https://vault.example.com:8200
    namespace: "{{ vault_namespace | default(omit) }}"
    auth_method: userpass
    username: "{{ vault_auth_username }}"
    password: "{{ vault_auth_password }}"
    connection_name: postgres-main
    plugin_name: postgresql-database-plugin
    connection_url:
postgresql://{{'{{username}}'}}:{{'{{password}}'}}@postgres.example.com:5432/appdb?
sslmode=disable
    connection_username: "{{ db_static_user }}"
    connection_password: "{{ db_static_password }}"
    allowed_roles:
      - app-readonly
      - app-readwrite
  register: db_connection_result
```

After (`hashicorp.vault`):

```
- name: Create database connection with PostgreSQL plugin

  hashicorp.vault.database_connection:

    name: postgres-sample-connection

    plugin_name: "postgresql-database-plugin"

    allowed_roles:

      - "readonly"

    connection_url: "host=localhost port=5432 user='{{ username }}'
password='{{ password }}'"

    plugin_options:

      max_open_connections: 5

      max_connection_lifetime: "5s"

    username: "admin_user"

    password: "secure_password"
```

Example 2: Update a database connection with MySQL

Before (`community.hashi_vault`):

```
- name: Example 2 - Update an existing database connection (MySQL plugin)

  community.hashi_vault.vault_database_connection_configure:

    url: https://vault.example.com:8200

    namespace: "{{ vault_namespace | default(omit) }}"

    auth_method: token

    token: "{{ vault_token }}"

    connection_name: mysql-main

    plugin_name: mysql-database-plugin

    connection_url: "{{'{{username}}'}}:{{'{{password}}'}}
@tcp(mysql.example.com:3306)/"

    connection_username: "{{ db_static_user }}"

    connection_password: "{{ db_static_password }}"

    allowed_roles:

      - readonly

      - readwrite

  register: mysql_connection_result
```

After (`hashicorp.vault`):

```
- name: Update a database connection with MySQL
  hashicorp.vault.database_connection:
    name: mysql-sample-connection
    state: present
    connection_url: "mysql://vaultuser:secretpassword@localhost:3306/mydb"
    verify_connection: true
    plugin_name: "mysql-database-plugin"
    database_mount_path: "database-conn-config-integration-tests"
    allowed_roles:
      - "readonly"
      - "readwrite"
    username: "vaultuser"
    password: "secretpassword"
```

Example 3: Reset a database connection

Before (`community.hashi_vault`):

```
- name: Reset a Database Connection with the default mount point
  community.hashi_vault.vault_database_connection_reset:
    url: https://vault:8201
    auth_method: userpass
    username: '{{ user }}'
    password: '{{ passwd }}'
    connection_name: mysql-sample-connection
  register: result
```

After (`hashicorp.vault`):

```
- name: Reset a database connection
  hashicorp.vault.database_connection:
    name: mysql-sample-connection
    state: reset
```

Example 4: Delete a database connection

Before (`community.hashi_vault`):

```
- name: Delete a Database Connection with the default mount point
  community.hashi_vault.vault_database_connection_delete:
    url: https://vault:8201
    auth_method: userpass
    username: '{{ user }}'
    password: '{{ passwd }}'
    connection_name: SomeName
  register: result

- name: Display the result of the operation
  ansible.builtin.debug:
    msg: "{{ result }}"
```

After (`hashicorp.vault`):

```
- name: Delete a database connection
  hashicorp.vault.database_connection:
    name: mysql-sample-connection
    state: absent
```

Example for the hashicorp.vault.database_connection_info module

The following example shows a basic configuration for the `hashicorp.database_connection_inf` o module.

Example: List available database connections

```
- name: Read database connection mysql
  hashicorp.vault.database_connection_info:
    name: mysql
```

Integrate with the external policy engine Open Policy Agent (OPA)

Integrating Ansible Automation Platform with Open Policy Agent (OPA) enforces policies at automation runtime. Use encoded rules to define, manage, and enforce how users interact with the platform. Automate policy management to improve security, compliance, and operational efficiency.

Integrating with OPA helps you to:

- **Enforce policies at runtime**: Stop automation actions that violate organizational policies and provide users with clear information about policy violations before jobs run.
- **Centralize policy management**: Offload policy decisions to your OPA server and manage automation governance rules alongside organizational policies.
- **Improve compliance and security**: Apply policy rules to automation content at the organization, inventory, or job template level to ensure consistent governance.

How Ansible Automation Platform supports OPA integration

When you trigger policy enforcement, policies are retrieved from your OPA server. These policies are applied to automation content before jobs run. If a violation is detected, the action stops and you receive information about the specific policy violation.

Associate policies with organizations, inventories, or job templates to control where policy enforcement occurs.

Implement policy enforcement

Policy enforcement at automation runtime is a feature that uses encoded rules to define, manage, and enforce policies that govern how your users interact with your Ansible Automation Platform instance. Policy enforcement automates policy management, improving security, compliance, and efficiency.

Open Policy Agent, or OPA, is a policy engine that offloads policy decisions from your Ansible instance. When it is triggered, the policy enforcement feature connects to OPA to retrieve policies specified in your configuration, and applies policy rules to your automation content. If OPA detects a policy violation, it will stop the action and give your user information about the policy violation. For more information, see *Open Policy Agent* in the Related Links section.

Prerequisites

Before you can implement policy enforcement in your Ansible Automation Platform instance, you must have:

- Access to an OPA server that is reachable from your Ansible Automation Platform deployment.
- Configured Ansible Automation Platform with settings required for authenticating to your OPA server.
- Some familiarity with OPA and the Rego language, which is the language policies are written in.

For policy enforcement to work correctly, you must both configure the OPA server in your policy settings, and associate a specific policy with a particular resource. For example, a particular organization, inventory, or job template.

NOTE:
OPA API V1 is the only version currently supported in Ansible Automation Platform.

Related information
Open Policy Agent

Configure policy enforcement settings

You can specify how your Ansible Automation Platform instance interacts with OPA by modifying your global settings.

Before you begin

- To configure policy enforcement, you must have administrative privileges.

NOTE:
If you do not configure the OPA server in your policy settings, policy evaluation will not occur when you run the job.

Procedure

1. From the navigation panel, select **Settings > Automation Execution > Policy**.

2. Click **Edit policy settings**.
3. On the Policy Settings page, fill out the following fields:

   OPA Server hostname

   Enter the name of the host that connects to the OPA service.

   OPA server port

   Enter the port that connects to the OPA service.

   OPA authentication type

   Select the OPA authentication type.

   OPA custom authentication header

   Enter a custom header to append to request headers for OPA authentication.

   OPA request timeout

   Enter the number of seconds until the connection times out.

   OPA request retry count

   Enter a figure for the number of times a request can attempt to connect to the OPA service before failing.

4. Depending on your authentication type, you might need to fill out the following fields.
   a. If you selected Token as your authentication type:

      OPA authentication token

      Enter the OPA authentication token.

   b. If you selected Certificate as your authentication type:

      OPA client certificate content

      Enter content of the CA certificate for mTLS authentication.

      OPA client key content

      Enter the client key for mTLS authentication.

      OPA CA certificate content

      Enter the content of the CA certificate for mTLS authentication.

5. Beneath the heading labeled **Options**:

   Use SSL for OPA connection

   Check this box to enable an SSL/TLS connection to the OPA service.

6. Click **Save policy settings**.

Understand OPA packages and rules

An OPA policy is organized in packages, which are namespaced collections of rules. The basic structure of an OPA policy looks like this:

```
package aap_policy_examples  # Package name


import rego.v1  # Import required for Rego v1 syntax


# Rules define the policy logic
allowed := {
    "allowed": true,
    "violations": []
}
```

The key components of the rule's structure are:

Package declaration

This defines the namespace for your policy.

Rules

This defines the policy's logic and the decision that it returns.

These components together form the OPA policy name, with the format `[package]/[rule]`. Enter the OPA policy name when you configure enforcement points.

Configure enforcement points

After you have set up your Ansible Automation Platform instance to communicate with the OPA server, you can set up enforcement points where you want the policy to be applied.

- You can associate a policy with a job template, an inventory, or an organization. Enforcement then occurs in the following ways:

  Organization

  Jobs launched from a template owned by an organization will fail if the policy is violated. This configuration provides broad control over automation within organizational boundaries.

  Inventory

Jobs that use an inventory associated with a policy fail if the policy is violated. This configuration allows you to control access to specific infrastructure resources.

Job template

Jobs launched from a template associated with a policy fail if the job violates the associated policy. This configuration provides granular control over specific automation tasks.

> **NOTE:**
> If you do not associate a policy with a resource, policy evaluation will not occur when you run the related job.

Associate a policy with an organization

Learn how to associate a policy with an organization.

Procedure

1. From the navigation panel, select **Access Management > Organizations**.
2. On the **Organizations** page:
   a. To edit an existing organization, find the organization you want to edit and click the pencil icon to go to the editing screen.
   b. To create a new organization, click **Create organization**.
3. In the field labeled **Policy enforcement**, enter the query path associated with the policy you want to implement. You must format the query path as `package/rule`.
4. Click **Next** and then **Finish** to save your settings.

Associate a policy with an inventory

Learn how to associate a policy with an inventory.

Procedure

1. From the navigation panel, select **Automation Execution > Infrastructure > Inventories**.
2. On the **Inventories** page:
   a. To edit an existing inventory, find the inventory you want to edit and click the pencil icon to go to the editing screen.
   b. To create a new inventory, click **Create inventory**.

3. In the field titled **Policy enforcement.**, enter the query path associated with the policy you want to implement. You must format the query path as `package/rule`.
4. Click **Save inventory** if you are editing an existing inventory, or click **Create inventory** if you are creating a new inventory.

Associate a policy with a job template

Learn how to associate a policy with a job template.

Procedure

1. From the navigation panel, select **Automation Execution > Templates**.
2. On the **Automation Templates** page:
   a. To edit an existing job template, find the job template you want to edit and click the pencil icon to go to the editing screen.
   b. To create a new job template, click **Create template**.
3. In the field titled **Policy enforcement**, enter the query path associated with the policy you want to implement. You must format the query path as `package/rule`.
4. Click **Save job template** if you are editing an existing job template, or click **Create job template** if you are creating a new job template.

Policy enforcement input and output options

Use the following inputs and outputs to craft policies for use in policy enforcement.

Input data

Input	Type	Description
`id`	Integer	The job's unique identifier.
`name`	String	Job template name.
`created`	Datetime (ISO 8601)	Timestamp indicating when the job was created.
`created by`	Object	Information about the user who created the job.

Input	Type	Description
		• `id` (integer): Unique identifier for the user • `username` (string): creator username • `is_superuser` (boolean): indicates if the user is a superuser
`credentials`	List of objects	Credentials associated with job execution. • `id` (integer): the credential's unique identifier • `name` (string): credential name • `description` (string): credential description • `organization` (integer or null): organization identifier associated with the credential • `credential_type` (integer): credential type identifier • `managed` (boolean): indicates if the credential is managed internally • `kind` (string): credential type (such as `ssh` , `cloud` , or `kubernetes`)
`execution_environment`	Object	Details about the execution environment used for the job.

Input	Type	Description
		• `id` (integer): the execution environment's unique identifier • `name` (string): execution environment name • `image` (string): container image used for execution • `pull` (string): pull policy for the execution environment
`extra_vars`	JSON	Extra variables provided for job execution.
`forks`	Integer	The number of parallel processes used for job execution.
`hosts_count`	Integer	The number of hosts targeted by the job.
`instance_group`	Object	Information about the instance group handling the job, including: • `id` (integer): the instance group's unique identifier • `name` (string): the instance group name • `capacity` (integer): the available capacity in the group • `jobs_running` (integer): the number of currently running jobs

Input	Type	Description
		• `jobs_total` (integer): total jobs handled by the group • `max_concurrent_jobs` (integer): maximum concurrent jobs allowed • `max_forks` (integer): maximum forks allowed
`inventory`	Object	Inventory details used in the job execution, including: • `id` (integer): the inventory's unique identifier • `name` (string): The inventory's name • `description` (string): inventory description • `kind` (string): the inventory type • `total_hosts` (integer): the total number of hosts in the inventory • `total_groups` (integer): the total number of groups in the inventory • `has_inventory_sources` (boolean): indicates if the inventory has external sources • `total_inventory_sources` (integer): the

Input	Type	Description
		number of external inventory sources • `has_active_failures` (boolean): indicates if there are active failures in the inventory • `hosts_with_active_failures` (boolean): the number of hosts with active failures • `inventory_sources` (array): external inventory sources associated with the inventory
`job_template`	Object	Information about the job template, including: • `id` (integer): the job template's unique identifier • `name` (string): the job template's name • `job_type` (string): type of job (for example, `run`)
`job_type`	Choice (String)	Type of job execution. Allowed values are: • `run` • `check` • `scan`
`job_type_name`	String	Human-readable name for the job type.
`labels`	List of objects	Labels associated with the job, including:

Input	Type	Description

		• `id` (integer): the label's unique identifier
		• `name` (string): the label name
		• `organization` (object): the organization associated with the label
		  ◦ `id` (integer): the organization's unique identifier
		  ◦ `name` (string): the organization name

`launch_type`	Choice (String)	How the job was launched. Allowed values include: • `manual` : manual • `relaunch` : relaunch • `callback` : callback • `scheduled` : scheduled • `dependency` : dependency • `workflow` : workflow • `webhook` : webhook • `sync` : sync • `scm` : SCM update
`limit`	String	The limit applied to the job execution.

Input	Type	Description
`launched_by`	Object	Information about the user who launched the job, including: • `id` (integer): the user's unique identifier • `name` (string): the user name • `type` (type): the user type (for example, `user` or `system`) • `url` (string): the user's API URL
`organization`	Object	Information about the organization associated with the job, including: • `id` (integer): the organization's unique identifier • `name` (name): the organization's name
`playbook`	String	The playbook used in the job execution.
`project`	Object	Details about the project associated with the job, including: • `id` (integer): the project's unique identifier • `name` (string): the project name • `status` (choice-string): the project status ◦ `successful`: Successful

Input	Type	Description
		◦ `failed` : failed ◦ `error` : error • `scm_type` (string): source control type (such as `git`, or `svn`) • `scm_url` (string): the source control repository URL • `scm_branch` (string): the branch used in the repository • `scm_refspec` (string): RefSpec for the repository • `scm_clean` (boolean): whether the SCM is cleaned before updates • `scm_track_submodules` (boolean): whether submodules are tracked • `scm_delete_on_update` (boolean): whether SCM deletes files on update
`scm_branch`	String	The specific branch to use for SCM.
`scm_revision`	String	SCM revision used for the job.
`workflow_job`	Object	Workflow job details, if the job is part of a workflow.
`workflow_job_template`	Object	Workflow job template details.

The following code block shows example input data from a demo job template launch:

```
{
  "id": 70,
  "name": "Demo Job Template",
  "created": "2025-03-19T19:07:03.329426Z",
  "created_by": {
    "id": 1,
    "username": "admin",
    "is_superuser": true,
    "teams": []
  },
  "credentials": [
    {
      "id": 3,
      "name": "Example Machine Credential",
      "description": "",
      "organization": null,
      "credential_type": 1,
      "managed": false,
      "kind": "ssh",
      "cloud": false,
      "kubernetes": false
    }
  ],
  "execution_environment": {
    "id": 2,
    "name": "Default execution environment",
    "image": "registry.redhat.io/ansible-automation-platform-25/ee-supported-
rhel8@sha256:b9f60d9ebbbb5fdc394186574b95dea5763b045ceff253815afeb435c626914d",
    "pull": ""
  },
  "extra_vars": {
    "example": "value"
  },
  "forks": 0,
  "hosts_count": 0,
  "instance_group": {
    "id": 2,
```

```
    "name": "default",
    "capacity": 0,
    "jobs_running": 1,
    "jobs_total": 38,
    "max_concurrent_jobs": 0,
    "max_forks": 0
  },
  "inventory": {
    "id": 1,
    "name": "Demo Inventory",
    "description": "",
    "kind": "",
    "total_hosts": 1,
    "total_groups": 0,
    "has_inventory_sources": false,
    "total_inventory_sources": 0,
    "has_active_failures": false,
    "hosts_with_active_failures": 0,
    "inventory_sources": []
  },
  "job_template": {
    "id": 7,
    "name": "Demo Job Template",
    "job_type": "run"
  },
  "job_type": "run",
  "job_type_name": "job",
  "labels": [
    {
      "id": 1,
      "name": "Demo label",
      "organization": {
        "id": 1,
        "name": "Default"
      }
    }
```

```
  ],
  "launch_type": "workflow",
  "limit": "",
  "launched_by": {
    "id": 1,
    "name": "admin",
    "type": "user",
    "url": "/api/v2/users/1/"
  },
  "organization": {
    "id": 1,
    "name": "Default"
  },
  "playbook": "hello_world.yml",
  "project": {
    "id": 6,
    "name": "Demo Project",
    "status": "successful",
    "scm_type": "git",
    "scm_url": "https://github.com/ansible/ansible-tower-samples",
    "scm_branch": "",
    "scm_refspec": "",
    "scm_clean": false,
    "scm_track_submodules": false,
    "scm_delete_on_update": false
  },
  "scm_branch": "",
  "scm_revision": "",
  "workflow_job": {
    "id": 69,
    "name": "Demo Workflow"
  },
  "workflow_job_template": {
    "id": 10,
    "name": "Demo Workflow",
    "job_type": null
```

```
  }
}
```

Output data

Input	Type	Description
`allowed`	Boolean	Indicates whether the action is permitted
`violations`	List of strings	Reasons why the action is not permitted

The following code block shows an example of expected output from the OPA policy query:

```
{
  "allowed": false,
  "violations": [
    "No job execution is allowed",
    ...
  ],
  ...
}
```

Integrate with Red Hat Lightspeed (formerly Insights)

Automation controller supports integration with Red Hat Lightspeed.

When a host is registered with Red Hat Lightspeed, it is scanned continually for vulnerabilities and known configuration conflicts. Each problem identified can have an associated fix in the form of an Ansible Playbook.

Red Hat Lightspeed users create a maintenance plan to group the fixes and can create a playbook to mitigate the problems. Automation controller tracks the maintenance plan playbooks through a Red Hat Lightspeed project.

Authentication to Red Hat Lightspeed through Basic Authorization is backed by a special credential, which must first be established in automation controller.

To run a Red Hat Lightspeed maintenance plan, you need a Red Hat Lightspeed project and inventory.

Create Red Hat Lightspeed credentials

To create a Red Hat Lightspeed credential, use the following procedure.

Before you begin

- To use token-based authentication, you must create a Red Hat service account to generate a **Client ID** and **Client secret**.
- Assign this service account to the appropriate **User Access** group with necessary permissions.

To enable integration between Ansible Automation Platform and Red Hat Lightspeed, assign the service account the following permissions:

- **inventory:hosts:read** (included in the Inventory Hosts viewer role)
- **patch:read** (included in the Patch viewer role)
- **remediations:remediation:read** and **playbook-dispatcher:run:read** (included in the Remediations user role)

You might consider associating your service account with an existing user access group that already has the required permissions, or creating a new user access group.

NOTE:
In adherence to security guidelines, service accounts are not automatically included in the default access group. To grant access, you must manually add them to the appropriate user access groups.

If you are not an organization administrator, you can create a service account and then ask your administrator to add your account to the appropriate user access groups.

After you have created a service account and assigned it the appropriate permissions, you can create a new credential for use with Red Hat Lightspeed.

Procedure

1. From the navigation panel, select **Automation Execution > Infrastructure > Credentials**.
2. Click **Create credential**.
3. Enter the appropriate details in the following fields:
    - **Name**: Enter the name of the credential.
    - Optional: **Description**: Enter a description for the credential.
    - Optional: **Organization**: Enter the name of the organization with which the credential is associated, or click the search icon and select it from the **Select organization** window.

- **Credential type**: Enter **Red Hat Lightspeed** or select it from the list.

> **NOTE:**
> Use the **Username** and **Password** fields for Basic authentication. You can leave these blank if using **Client ID** and **Client secret**.

- **Client ID**: Enter the client ID you received when you created your service account.
- **Client secret**: Enter the client secret you received when you created your service account.

4. Click **Create credential**.

   You can now use this credential in an Source an inventory from Red Hat Lightspeed and Red Hat Lightspeed project.

- If you receive a project sync failure, see the steps in Remediating a Red Hat Lightspeed inventory and check your analytics logs.

> **IMPORTANT:**
> You must recreate existing credentials and reassociate them with existing projects and inventory sources to support token-based authentication.

Related information
Creating and Managing Service Accounts
How to use Service Accounts on the Hybrid Cloud Console
Support for token-based authentication via Service Account for Red Hat Lightspeed in Ansible Automation Platform

Create a Red Hat Lightspeed project

Create an automation controller project linked to Red Hat Lightspeed and retrieve remediation playbooks. This streamlines your efforts to address vulnerabilities and keep system configurations.

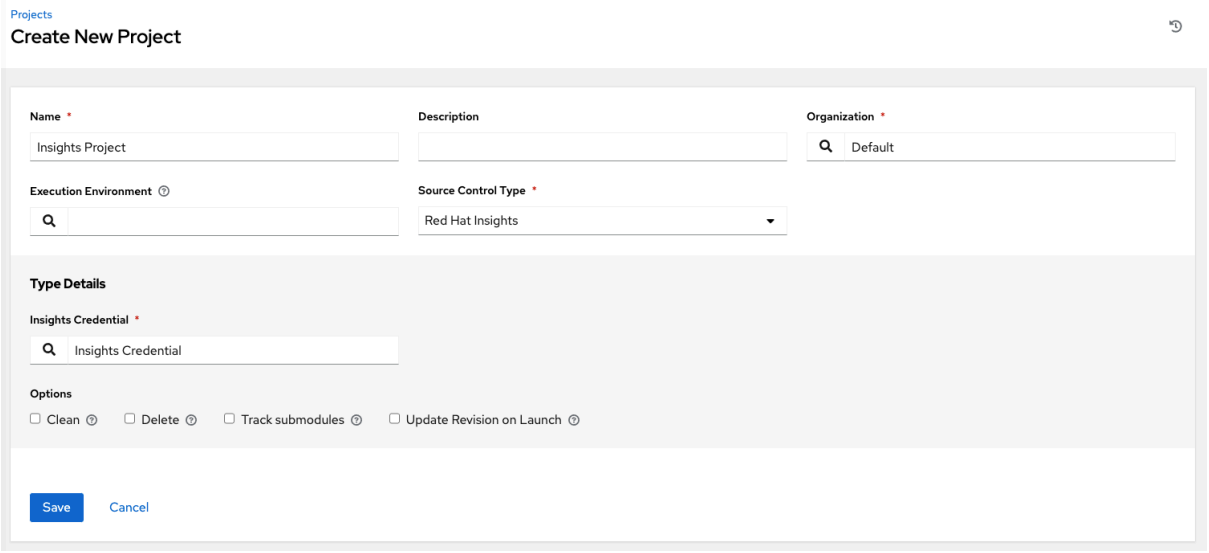
Procedure

1. From the navigation panel, select **Automation Execution > Projects**.
2. Click **Create project**.
3. Enter the appropriate details in the following fields. Note that the following fields require specific Red Hat Lightspeed related entries:

   - **Name**: Enter the name for your Red Hat Lightspeed project.
   - Optional: **Description**: Enter a description for the project.

- **Organization**: Enter the name of the organization with which the credential is associated, or click the search icon and select it from the **Select organization** window.
- Optional: **Execution environment**: The execution environment that is used for jobs that use this project.
- **Source control type**: Select **Red Hat Lightspeed**.
- Optional: **Content signature validation credential**: Enable content signing to verify that the content has remained secure when a project is synced.
- **Red Hat Lightspeed credential**: This is pre-populated with the Red Hat Lightspeed credential you created before. If not, enter the credential, or click the search icon and select it from the **Select Red Hat Lightspeed Credential** window.

4. Select the update options for this project from the **Options** field and provide any additional values, if applicable. For more information about each option click the tooltip icon next to each one.



5. Click **Create project**.

   All SCM and project synchronizations occur automatically the first time you save a new project.

Next steps

If you want them to be updated to what is current in Red Hat Lightspeed, manually update the SCM-based project by clicking the **Update** icon under the project's available actions.

This process syncs your Red Hat Lightspeed project with your Red Hat Lightspeed account solution. Note that the status dot beside the name of the project updates once the sync has run.

Create a Red Hat Lightspeed inventory

The Red Hat Lightspeed playbook contains a `hosts:` line where the value is the hostname supplied to Red Hat Lightspeed, which can be different from the hostname supplied to automation controller.

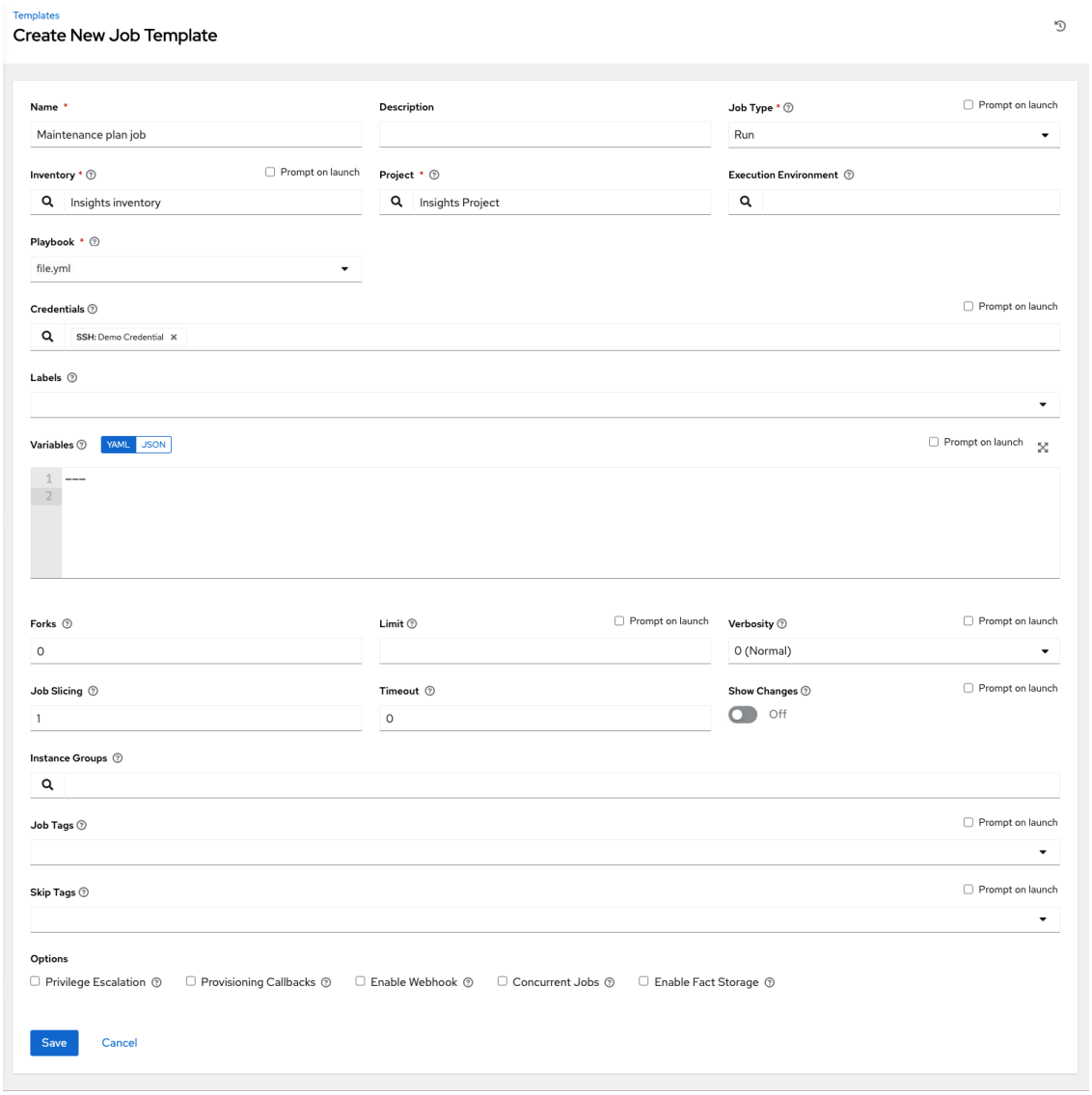
Remediate a Red Hat Lightspeed inventory

Remediation of a Red Hat Lightspeed inventory enables automation controller to run Red Hat Lightspeed playbooks with a single click.

You can do this by creating a job template to run the Red Hat Lightspeed remediation.

Procedure

1. From the navigation menu, select **Automation Execution > Templates**.
2. On the **Templates** list view, click **Create template** and select from the list.
3. Enter the appropriate details in the following fields. Note that the following fields require specific Red Hat Lightspeed related entries:
    - **Name**: Enter the name of your Maintenance Plan.
    - Optional: **Description**: Enter a description for the job template.
    - **Job Type**: If not already populated, select **Run** from the job type list.
    - **Inventory**: Select the Red Hat Lightspeed inventory that you previously created.
    - **Project**: Select the Red Hat Lightspeed project that you previously created.
    - Optional: **Execution Environment**: The container image to be used for execution.
    - **Playbook**: Select a playbook associated with the Maintenance Plan that you want to run from the playbook list.
    - Optional: **Credentials**: Enter the credential to use for this project or click the search ( ) icon and select it from the pop-up window. The credential does not have to be a Red Hat Lightspeed credential.
    - **Verbosity**: Keep the default setting, or select the required verbosity from the list.



4. Click **Create job template**.
5. Click the launch icon to launch the job template.

Result

When complete, the job results in the **Job Details** page.

Red Hat product documentation legal notices

Copyright © Red Hat

Except as otherwise noted below, the text of and illustrations in this documentation are licensed by Red Hat under the Creative Commons Attribution–Share Alike 3.0 Unported license. If you distribute this document or an adaptation of it, you must provide the URL for the original version. Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

XFS is a trademark or registered trademark of Hewlett Packard Enterprise Development LP or its subsidiaries in the United States and other countries.

The OpenStack® Word Mark and OpenStack logo are trademarks or registered trademarks of the Linux Foundation, used under license.

All other trademarks are the property of their respective owners.

GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Copyright © 2007 Free Software Foundation, Inc. link:https://fsf.org/. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program—to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If

such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions. "This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code. The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component,

or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions. All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law. No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work’s users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies. You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions. You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

- a) The work must carry prominent notices stating that you modified it, and giving a relevant date.
- b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".
- c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.
- d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so. A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms. You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

- a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.
- b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

- c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.
- d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.
- e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms. "Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

- a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or
- b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or
- c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or
- d) Limiting the use for publicity purposes of names of licensors or authors of the material; or
- e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or
- f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination. You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies. You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients. Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents. A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom. If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License. Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License. The Free Software Foundation may publish revised and/ or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty. THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES

SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16. If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

```
    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.
```

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

```
    <program>  Copyright (C) <year>  <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type `show c' for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a "copyright disclaimer" for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see link:https://www.gnu.org/licenses/.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read link:https://www.gnu.org/licenses/why-not-lgpl.html.

Apache license

Version 2.0, January 2004

http://www.apache.org/licenses/

Terms and Conditions for use, reproduction, and distribution

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, **"control"** means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or **"Your"**) shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, **"submitted"** means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the

Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as **"Not a Contribution."**

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a **"NOTICE"** text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications,

or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

Copyright 2026. All rights reserved.

www.redhat.com