This guide provides information used to deploy automation mesh as part of your Ansible Automation Mesh Platform environment.
This guide provides information used to deploy automation mesh as part of your Ansible Automation Mesh Platform environment.
Legal Notice

Copyright © 2022 Red Hat, Inc.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at http://creativecommons.org/licenses/by-sa/3.0/. In accordance with CC-BY-SA, 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.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

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

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation’s permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

Providing Feedback: If you have a suggestion to improve this documentation, or find an error, please contact technical support at to create an issue on the Ansible Automation Platform Jira project using the Docs component.
Table of Contents

PREFACE ........................................................................................................................................ 3

CHAPTER 1. PLANNING FOR AUTOMATION MESH IN YOUR RED HAT ANSIBLE AUTOMATION PLATFORM ENVIRONMENT ......................................................................................... 4
  1.1. ABOUT AUTOMATION MESH ................................................................................................. 4
  1.2. CONTROL AND EXECUTION PLANES ................................................................................. 4
    1.2.1. Control plane ................................................................................................................. 4
    1.2.2. Execution plane ............................................................................................................ 5
    1.2.3. Peers ............................................................................................................................. 5
    1.2.4. Defining automation mesh node types .......................................................................... 5

CHAPTER 2. AUTOMATION MESH DESIGN PATTERNS .................................................................... 7
  2.1. SINGLE HYBRID NODE INVENTORY FILE EXAMPLE ......................................................... 7
  2.2. MULTIPLE HYBRID NODES INVENTORY FILE EXAMPLE .................................................. 7
  2.3. SINGLE NODE CONTROL PLANE WITH SINGLE EXECUTION NODE .............................. 7
  2.4. STANDARD CONTROL PLANE (3 NODE) AND (N) EXECUTION NODES, FULLY CONNECTED 7
  2.5. STANDARD CONTROL PLANE WITH SINGLE EGRESS TO THE EXECUTION TOPOLOGY 8
  2.6. STANDARD CONTROL PLANE AND EXECUTION TOPOLOGY WITH HOP NODES .......... 8
  2.7. COMPLEX AUTOMATION MESH DESIGN EXAMPLE ......................................................... 9

CHAPTER 3. DEPROVISIONING ..................................................................................................... 11
  3.1. DEPROVISIONING INDIVIDUAL NODES USING THE INSTALLER ................................. 11
  3.2. DEPROVISIONING GROUPS USING THE INSTALLER ....................................................... 11

CHAPTER 4. INSTALLING AUTOMATION MESH ........................................................................... 13
  4.1. ADDITIONAL RESOURCES ............................................................................................... 13
Thank you for your interest in Red Hat Ansible Automation Platform. Ansible Automation Platform is a commercial offering that helps teams manage complex multi-tier deployments by adding control, knowledge, and delegation to Ansible-powered environments.

This guide helps you to understand the installation requirements and processes behind installing Ansible Automation Platform. This document has been updated to include information for the latest release of Ansible Automation Platform.
CHAPTER 1. PLANNING FOR AUTOMATION MESH IN YOUR RED HAT ANSIBLE AUTOMATION PLATFORM ENVIRONMENT

The following topics contain information to help plan an automation mesh deployment in your Ansible Automation Platform environment. The subsequent sections explain the concepts that comprise automation mesh in addition to providing examples on how you can design automation mesh topologies. Simple to complex topology examples are included to illustrate the various ways you can deploy automation mesh.

1.1. ABOUT AUTOMATION MESH

Automation mesh is an overlay network intended to ease the distribution of work across a large and dispersed collection of workers through nodes that establish peer-to-peer connections with each other using existing networks.

Red Hat Ansible Automation Platform 2 replaces Ansible Tower and isolated nodes with automation controller and automation hub. Automation controller provides the control plane for automation through its UI, Restful API, RBAC, workflows and CI/CD integration, while Automation Mesh can be used for setting up, discovering, changing or modifying the nodes that form the control and execution layers.

Automation Mesh introduces:

- Dynamic cluster capacity that scales independently, allowing you to create, register, group, ungroup and deregister nodes with minimal downtime.
- Control and execution plane separation that enables you to scale playbook execution capacity independently from control plane capacity.
- Deployment choices that are resilient to latency, reconfigurable without outage, and that dynamically re-reroute to choose a different path when outages may exist. Mesh routing changes.
- Connectivity that includes bi-directional, multi-hopped mesh communication possibilities which are Federal Information Processing Standards (FIPS) compliant.

1.2. CONTROL AND EXECUTION PLANES

Automation mesh makes use of unique node types to create both the control and execution plane. Learn more about the control and execution plane and their node types before designing your automation mesh topology.

1.2.1. Control plane

The control plane consists of hybrid and control nodes. Instances in the control plane run persistent automation controller services such as the web server and task dispatcher, in addition to project updates, and management jobs.

- Hybrid nodes - this is the default node type for control plane nodes, responsible for automation controller runtime functions like project updates, management jobs and ansible-runner task operations. Hybrid nodes are also used for automation execution.
- Control nodes - control nodes run project and inventory updates and system jobs, but not regular jobs. Execution capabilities are disabled on these nodes.
1.2.2. Execution plane

The execution plane consists of execution nodes that execute automation on behalf of the control plane and have no control functions. Hop nodes serve to communicate. Nodes in the execution plane only run user-space jobs, and may be geographically separated, with high latency, from the control plane.

- **Execution nodes** - Execution nodes run jobs under ansible-runner with podman isolation. This node type is similar to isolated nodes.
- **Hop nodes** - Similar to a jump host, hop nodes will route traffic to other execution nodes. Hop nodes cannot execute automation.

1.2.3. Peers

Peer relationships define node-to-node connections. You can define peers within the [automationcontroller] and [execution_nodes] groups or using the [automationcontroller:vars] or [execution_nodes:vars] groups.

1.2.4. Defining automation mesh node types

You can define node types either by its default value assigned by the inventory group or by using the node_type host variable. Specify the node_type either as part of the inventory group or within the inventory vars group. This section provides examples of how you can define node types in your inventory file. Nodes in [execution_nodes] default execution node_type. Hybrid node types can be overridden to be control type via node_type=control. Execution node type can be overridden to be hop node type via node_type=hop.

**Hybrid node**

Nodes in [automationcontroller] default to the hybrid node type. In the below example, we create a single hybrid node:

```
[automationcontroller]
control-plane-1.example.com
```

**Control node**

Convert hybrid node types to control nodes using node_type=control:

```
[automationcontroller]
control-plane-1.example.com node_type=control
```

**Execution node**

Nodes in the [execution_nodes] inventory group default to the execution node type. In the below example, we create a single execution node:

```
[execution_nodes]
exeuction-node-1.example.com
```

**Hop node**
Convert execution nodes to hop nodes using `node_type=control`:

```
[execution_nodes]
execution-node-1.example.com node_type=hop
```

**Peer connections**

Create node-to-node connections using the `peers=` host variable. In the example below, we connect `control-plane-1.example.com` to `execution-node-1.example.com`, and `execution-node-1.example.com` to `execution-node-2.example.com`.

```
[automationcontroller]
control-plane-1.example.com peers=execution-node-1.example.com

[automationcontroller:vars]
node_type=control

[execution_nodes]
execution-node-1.example.com peers=execution-node-2.example.com
execution-node-2.example.com
```

**Additional resources**

- See the example automation mesh topologies in this guide for more examples of how to implement mesh nodes.
CHAPTER 2. AUTOMATION MESH DESIGN PATTERNS

The automation mesh topologies in this section are intended to provide examples you can use to design a mesh deployment in your environment. Examples range from a single, hybrid node deployment to a complex pattern that deploys numerous automation controllers, employing several execution and hop nodes.

Prerequisites

- You reviewed conceptual information on node types and relationships

2.1. SINGLE HYBRID NODE INVENTORY FILE EXAMPLE

This example inventory file deploys a single hybrid node control plane.

```
[automationcontroller]
control-plane-1.example.com
```

2.2. MULTIPLE HYBRID NODES INVENTORY FILE EXAMPLE

This example inventory file deploys a control plane consisting of multiple hybrid nodes.

```
[automationcontroller]
control-plane-1.example.com ansible_connection=local
control-plane-2.example.com
control-plane-3.example.com
```

2.3. SINGLE NODE CONTROL PLANE WITH SINGLE EXECUTION NODE

This example inventory file deploys a single node control plane and establishes a peer relationship to the execution node.

```
[automationcontroller]
control-plane-1.example.com

[automationcontroller:vars]
node_type=control
peers=execution_nodes

[execution_nodes]
execution-node-1.example.com
```

2.4. STANDARD CONTROL PLANE (3 NODE) AND (N) EXECUTION NODES, FULLY CONNECTED

This example inventory file deploys a control plane consisting of three nodes with fully connected peer execution nodes.

```
[automationcontroller]
control-plane-1.example.com
control-plane-2.example.com
```
2.5. STANDARD CONTROL PLANE WITH SINGLE EGRESS TO THE EXECUTION TOPOLOGY

This example inventory file deploys a control plane consisting of three nodes with a single point of egress to the execution plane.

```
[automationcontroller]
control-plane-1.example.com
control-plane-2.example.com
control-plane-3.example.com peers=execution-node-1.example.com

[automationcontroller:vars]
node_type=control

[execution_nodes]
evaluation-node-1.example.com peers=evaluation-node-2.example.com
evaluation-node-2.example.com peers=evaluation-node-3.example.com
evaluation-node-3.example.com peers=evaluation-node-4.example.com
evaluation-node-4.example.com peers=evaluation-node-5.example.com
evaluation-node-5.example.com peers=evaluation-node-6.example.com
evaluation-node-6.example.com peers=evaluation-node-7.example.com
```

2.6. STANDARD CONTROL PLANE AND EXECUTION TOPOLOGY WITH HOP NODES

This example inventory file deploys a control plane consisting of three nodes and an execution topology that utilizes hop nodes.

```
[automationcontroller]
control-plane-1.example.com
control-plane-2.example.com
control-plane-3.example.com
```
2.7. Complex Automation Mesh Design Example

This example inventory demonstrates a complex automation mesh design pattern that uses groups to associate nodes of the same type in order to reduce noise in the installer configuration. It deploys three automation controllers, two execution nodes and multiple hops with execution nodes.

NOTE

- The control plane contains only control nodes, which are auto-peered together.

- `[automationcontroller:vars]` peers the controllers to the disconnected execution nodes group `[instance_group_disconnected]`, ignoring the hop nodes in `[execution_nodes]`.

- All execution node types are defined in `[execution nodes]`, as well as which execution nodes need a hop. Hop nodes are also defined in this group.

- The `instance_grouo_prefix` automatically adds nodes into the automation controller user interface as an instance group after running the Red Hat Ansible Automation Platform installer.

- The `[local_hop]` group peers to the `[automationcontroller]` group.

- The `[remote_hop]` group peers to the `[local_hop]` group.

Inventory file

```plaintext
[automationcontroller]
control-plane-1.example.com ansible_connection=local
control-plane-2.example.com
control-plane-3.example.com

[automationcontroller:vars]
peers=instance_group_directconnected
node_type=control

[execution_nodes]
execution-node-1.example.com
execution-node-2.example.com
execution-node-3.example.com peers=local_hop
execution-node-4.example.com peers=remote Hop
hop-node-1.example.com node_type=hop
```
hop-node-2.example.com node_type=hop
hop-node-3.example.com node_type=hop

[instance_group_directconnected]
execution-node-1.example.com
execution-node-2.example.com

[instance_group_localhop]
execution-node-3.example.com

[instance_group_multihop]
execution-node-4.example.com

[local_hop]
hop-node-1.example.com
hop-node-2.example.com

[local_hop:vars]
peers=automationcontroller

[remote_hop]
hop-node-3.example.com

[remote_hop:vars]
peers=local_hop
CHAPTER 3. DEPROVISIONING

You can deprovision automation mesh nodes and instance groups using the Ansible Automation Platform installer. The procedures in this section describe how to deprovision specific nodes or entire groups, with example inventory files for each procedure.

3.1. DEPROVISIONING INDIVIDUAL NODES USING THE INSTALLER

You can deprovision nodes from your automation mesh using the Ansible Automation Platform installer. Running the installer will remove all configuration files and logs attached to the node.

**NOTE**

You can deprovision any of your inventory's hosts except for the first host specified in the '[automationcontroller]' group.

**Procedure**

- Append `node_state=deprovision` to nodes in the installer file you want to deprovision.

**Example**

This example inventory file deprovisions two nodes from an automation mesh configuration.

```
[automationcontroller]
126-addr.tatu.home ansible_host=192.168.111.126 node_type=control
121-addr.tatu.home ansible_host=192.168.111.121 node_type=hybrid routable_hostname=121-addr.tatu.home
115-addr.tatu.home ansible_host=192.168.111.115 node_type=hybrid node_state=deprovision

[automationcontroller:vars]
peers=connected_nodes

[execution_nodes]
110-addr.tatu.home ansible_host=192.168.111.110 receptor_listener_port=8928
108-addr.tatu.home ansible_host=192.168.111.108 receptor_listener_port=29182
node_state=deprovision
100-addr.tatu.home ansible_host=192.168.111.100 peers=110-addr.tatu.home node_type=hop
```

3.2. DEPROVISIONING GROUPS USING THE INSTALLER

You can deprovision entire groups from your automation mesh using the Ansible Automation Platform installer. Running the installer will remove all configuration files and logs attached to the nodes in the group.

**NOTE**

You can deprovision any of your inventory's hosts in a group except for the first host specified in the '[automationcontroller]' group.

**Procedure**

- Add `node_state=deprovision` to the [group:vars] associated with the group you wish to deprovision.
- Add `node_state=deprovision` to the `[group:vars]` associated with the group you wish to deprovision.

Example

```
[execution_nodes]
execution-node-1.example.com peers=execution-node-2.example.com
execution-node-2.example.com peers=execution-node-3.example.com
execution-node-3.example.com peers=execution-node-4.example.com
execution-node-4.example.com peers=execution-node-5.example.com
execution-node-5.example.com peers=execution-node-6.example.com
execution-node-6.example.com peers=execution-node-7.example.com
execution-node-7.example.com

[execution_nodes:vars]
node_state=deprovision
```
CHAPTER 4. INSTALLING AUTOMATION MESH

You can install an automation mesh using the instructions found in the Red Hat Ansible Automation Platform Installation Guide. Ansible Automation Platform installations are separated into unique scenarios based on the components deployed. You can incorporate automation mesh designs based on which platform components you intend to deploy in your environment. Review the systems requirements for each component and automation mesh node type before proceeding with your installation.

4.1. ADDITIONAL RESOURCES

- See the Red Hat Ansible Automation Platform Reference Architecture for additional information before installing.