Puppet Plug-in for VMware vRealize Automation

Contents

Puppet Plug-in for VMware vRealize Automation..... 3

Puppet Plug-in for VMware vRealize Automation....3
Installing and configuring a reference implementation.... 4
Getting started with vRA on an existing primary server....6
Configure the primary server.... 7
Install and configure the Puppet plug-in....11
Add a primary server endpoint in vRA.... 11
Managing and provisioning infrastructure with vRA and Puppet.... 11
Troubleshooting....16
vRO release notes....17
vRO plug-in 3.3....17
vRO plug-in 3.2....17
vRO plug-in 3.1....18
vRO Puppet plug-in 3.0....18

Puppet Plug-in for VMware vRealize Automation

- Puppet Plug-in for VMware vRealize Automation on page 3

The Puppet Plug-in for vRealize Automation (vRA) provides tools and out-of-the-box components that easily create, provision, and manage application stacks on virtual servers.

- Installing and configuring a reference implementation on page 4

This guide you walks you through installing and configuring a reference implementation of the Puppet plug-in using Puppet Enterprise 2018.1 or newer and vRA 7.3 or newer. This implementation is designed to create a development environment with vRO, vRA, and Puppet running as quickly as possible in order to help you learn how these tools work together.

- Getting started with vRA on an existing primary server on page 6

The following guide provides high level requirements for installing and configuring the Puppet vRO Plug-in using Puppet Enterprise (PE) 2018.1 and vRealize Automation (vRA) 7.x in an existing environment. As existing deployments vary, this guide does not specify implementation details.

- Managing and provisioning infrastructure with vRA and Puppet on page 11

Once you have configured vRO and the Puppet vRO Plug-in, you can use vRealize Automation (vRA) to request servers using blueprints.

- Troubleshooting on page 16
- vRO release notes on page 17

See the vRO Puppet plug-in documentation for more information about the plug-in and instructions on installing and using it.

Puppet Plug-in for VMware vRealize Automation

The Puppet Plug-in for vRealize Automation (vRA) provides tools and out-of-the-box components that easily create, provision, and manage application stacks on virtual servers.

This documentation walks you through setting up a reference implementation of the plug-in, vRO, and vRA, as well as setting up a production implementation to use on an existing primary server in PE. The reference implementation is meant to be a demo and isn't designed for out-of-the-box use in production, but you can modify it to meet your needs. See Installing and configuring a reference implementation on page 4 to get started with the reference implementation. If you want to start fresh and intend to use this in production, go to Getting started with vRA on an existing primary server on page 6.

Once you've completed this guide, you should have a working environment and examples with which you can develop your own Puppet code, vRO workflows, and vRA blueprints.

What the Puppet plug-in can do

With a single click, Puppet, vRealize Orchestrator (vRO), and vRealize Automation (vRealize Automation) can automatically create a VM, install the Puppet agent, autosign its certificate, add Puppet roles and profiles, install the required Puppet modules and the software they configure, and set up the server for immediate use.

If you're new to Puppet and vRO, you can use vRO and vRealize Automation to set up a live, functional Puppet managed system with much less effort than building one manually.

If you're experienced with vRO, vRealize Automation, and Puppet, you can also use this plug-in to model common Puppet workflows as vRO workflows and vRealize Automation blueprints, then deploy them just as easily as other VMs while maintaining the advantages of a Puppet managed infrastructure.

For vRealize Automation Enterprise 7.4 and newer versions of the 7.x line, adding Puppet management to blueprints is built into the vRealize Automation GUI. This plugin is still required for the integration to work. Once you have installed the plugin and configured it with vRealize Automation 7.4 you can drag and drop Puppet management into blueprints.

Installing and configuring a reference implementation

This guide you walks you through installing and configuring a reference implementation of the Puppet plug-in using Puppet Enterprise 2018.1 or newer and vRA 7.3 or newer. This implementation is designed to create a development environment with vRO, vRA, and Puppet running as quickly as possible in order to help you learn how these tools work together.

The reference implementation isn't designed to be used in a production environment. Once you're familiar with how the plug-in works, you can install it into your production vRO/vRA infrastructure and build compatible workflows and blueprints.

The plug-in works with many implementations of Puppet Enterprise, vRO, and vRA. While you can use these instructions to set up this plugin with other versions of Puppet Enterprise and vRO/vRA, we recommend using this reference implementation the first time through.

Note: If you're already experienced with Puppet, vRO, vRA, and the Puppet plug-in, see Managing and provisioning infrastructure with vRA and Puppet on page 11 for a quick reference of properties and usage.

Related information

Managing and provisioning infrastructure with vRA and Puppet on page 11
Once you have configured vRO and the Puppet vRO Plug-in, you can use vRealize Automation (vRA) to request servers using blueprints.

Prerequisites

The Puppet plug-in 3.0 is compatible with the following configurations.

You can use the Puppet plug-in 3.0 with vRA 7.3 Enterprise edition, which includes an advanced GUI experience with drag-and-drop Puppetcomponents on blueprints. To do so, you must have:

- A primary server running Puppet Enterprise 2018.1 or newer
- vRealize Automation Enterprise 7.3
- Either internal/external vRO 7.x appliance (vRA includes an internal vRO appliance)

You can also use the plug-in with any vRA version from 6 to 7.3. To do so, you must have:

- A primary server running Puppet Enterprise 2016.4 or newer
- vRealize Orchestrator 6.x or 7.x (vRA includes an internal vRO appliance)
- vRealize Automation 6.x or 7.x
- vRealize Automation Enterprise 7.3 or newer required for GUI integration

Agent nodes being managed by Puppet must run an operating system supported by the Puppet agent.

Note: The 32-bit version of Microsoft Windows Puppet agent is **not** compatible with vRO plug-in management. You must use the 64-bit (x64) agent.

Network requirements

In order to use vRealize Automation, you must have the correct port configuration for provisioning the primary server, new Linux VMs, and new Windows VMs.

Origin	Destination	Port
vRO	Primary server	SSH (22)
vRO	new Linux VM	SSH (22)
vRO	new Windows VM	WinRM (5985, 5986)

Origin	Destination	Port
new Linux and Windows VMs	• Primary server • Compile masters • Load balancers	• Primary server (8140) • Orchestrator (8142) • MCollective (61613) • RBAC (4433)

Removing previous versions of the Puppet plug-in

The plug-in **does not** currently support upgrades from the previous vRO Puppet plug-in versions.

If you're using any previous version of the plug-in, you **must** completely remove it before installing a newer version. For best results, delete all puppet elements from the vRO GUI client first and then copy this script from the vRO starter content to the appliance and execute it.

Install and configure Puppet Enterprise

For this reference implementation of the vRO plug-in, you must use a new, clean installation of Puppet Enterprise with Code Manager disabled. After running the setup script, you can opt to enable Code Manager.

1. Review the Puppet Enterprise hardware and operating system requirements.
2. Install Puppet Enterprise on a VM or server. This will be the primary server and must be accessible over the network from the vRO appliance or server. An easy way to install PE is to run the installer in text mode. Then there is only one question to answer: the password for the PE Console GUI.
3. Add the Puppet plug-in starter pack content by following the instructions in the README.

   The starter content repository provides reference implementations of Puppet roles and profiles for Linux and Windows web server stacks, utility scripts to prepare the primary server for vRO, and a templated autosigning script. Once you understand how Puppet, vRO, and vRA work together, you can use these reference implementations to help build your own Puppetized vRO/vRA implementations.

   If you're already experienced with Puppet, vRO, and vRA, you can replace this reference implementation with your own code or control repository.
4. Ensure that the primary server has a valid DNS hostname and NTP configured. If you don't have or use a DNS server, provide a valid hostname for the server's IP address in the primary server's hosts file (typically `/etc/hosts`).

   > **Note:**
   >
   > Make sure that a hostname is properly configured on the machines you're installing PE on. To prevent PowerShell authentication failures, Windows nodes should have their domain/forests configured or an appropriate PowerShell proxy configured prior to running install PE agent workflows.
5. Initiate a Puppet run on the primary server by running `sudo puppet agent -t`

   The vRO starter content creates a PE RBAC user and Linux user account on them primary server (both are named `vro-plugin-user`, default password `puppetlabs`) and adds rules to the `sudoers` file allowing it to run commands with elevated privileges as required by the plug-in.

   It also adds the following settings to the primary server's `sshd_config`:

   ```
   PermitRootLogin yes
   PasswordAuthentication yes
   ChallengeResponseAuthentication no
   ```

6. To display role class descriptions in the vRealize Automation web GUI, the starter content installs puppet-strings, a Puppet documentation extraction command built on YARD. If puppet-strings is not installed, you can install it by running:

   ```
   puppet resource package puppet-strings provider=puppet_gem
   ```

   Role class descriptions come from the `@summary` tag in your Puppet code, which `puppet-strings` can digest. The vRO starter content role and profile classes already have this built-in. To do this with your own role classes, add a `@summary` line with a 140 characters or less description. For example:

   ```
   # This role installs a MySQL databse and sample data
   #
   # @summary MySQL database server on Linux with sample data
   class role::linux_mysql_database {
     include profile::linux_baseline
     include profile::mysql
     include profile::sample_data
   }
   ```

7. If you do not allow a sudo-capable user to run commands for vRO — for instance, if you remove the `vro-plugin-user` account or revoke its sudoers privileges:
   a) Provide vRO with remote access to a user account on the primary server with those capabilities, or to the primary server's root user, which is insecure.
   b) Make a user with the same username in PE RBAC.

Install and configure the Puppet Plug-in

For the reference implementation, we recommend using the vRO built into the vRA appliance.

If you choose to install your own vRO, refer to the vRO documentation. If you previously installed version 1.0 or 2.0 of the plug-in, you must completely remove it before installing version 3.2. The vRO starter content repository includesa script to assist with removing the plug-in.

Related topics: Puppet knowledge base: Removing the Puppet plugin from VMware vRealize Automation

Install the Puppet plug-in

Download and install the Puppet plug-in.

1. Download the Puppet plug-in's `.vmoapp` package from the VMware Solution Exchange.
2. Log in to the vRO server's control center at `https://<VRO-SERVER-IP-ADDRESS>:8283/vco-controlcenter`
3. Click the **Plugins** tab.
4. Click **Install plug-in**.
5. Install the Puppet plug-in's `.vmoapp` package downloaded from the VMware Solution Exchange. Read and accept the EULA, then click Install.
6. After the installation confirmation message appears, click **Startup Options** in the message reminding you to restart the Orchestrator server. In some versions of vRO this message may not appear, but you still must restart the Orchestrator server.
7. On the **Startup Options** page, click **Restart** under the Current Status heading.

Getting started with vRA on an existing primary server

The following guide provides high level requirements for installing and configuring the Puppet vRO Plug-in using Puppet Enterprise (PE) 2018.1 and vRealize Automation (vRA) 7.x in an existing environment. As existing deployments vary, this guide does not specify implementation details.

For a greenfield deployment, see Installing and configuring a reference implementation on page 4.

- Configure the primary server on page 7

The Puppet vRO Plug-in requires you install and configure a Puppet Enterprise (PE) 2018.1.0+ primary server, which can be done manually or with a module. Ensure it is accessible on the same network as the vRO appliance.

- Install and configure the Puppet plug-in on page 11
- Add a primary server endpoint in vRA on page 11

Follow the vRA 7.4 docs to start using vRO plugins in vRA.

Configure the primary server

The Puppet vRO Plug-in requires you install and configure a Puppet Enterprise (PE) 2018.1.0+ primary server, which can be done manually or with a module. Ensure it is accessible on the same network as the vRO appliance.

The easiest way to configure the primary server is using the puppetlabs/vra_puppet_plugin_prep module. The module helps you automatically configure SSH, creates an RBAC user, configures autosign, and installs puppet strings. You still need to classify nodes, create a rule, and add the primary server endpoint manually.

If you decide to use the module to set up your primary server, you can skip over the first four setup processes in this guide and proceed to Role class location on page 10.

If you don't want to use the module for setup, the guide walks you through all of the following steps:

- Configuring ssh into the primary server with a password and running commands as root.
- Creating an RBAC user on the primary server. Note as of plug-in version 3.2, RBAC permissions have changed.
- Configuring autosigning to use `challengePassword` in the CSR with a shared secret.
- Installing puppet-strings.
- Making sure role manifests are in the `<environment>/site/role/manifests` directory.
- Checking nodes are classified with the `trustes.extensions.pp_role` in the console or `$trusted['pp_role']` in the `site.pp`.
- Configuring PE environment groups to use the `trused.extensions.pp_environment` fact.

Primary server shell access

The vRO Plug-in needs to be able to ssh into the primary server and run commands. You can run these commands as the root user or a non-root user with sudo.

Configure one of the following on your primary server:

- Root user ssh
  - Enable root ssh access
- Non-root user ssh
  - Create a local user on the primary server
  - Enable ssh with password authentication
  - Enable passwordless sudo for the user

  An example set of the `/etc/sudoers.d/user` file:

```
## This file allows the vRO plugin user 'vro-plugin-user' to remove nodes
 that are destroyed in vRO/vRA.
## This file also disallows the user 'vro-plugin-user' from removing the
 primary server and other PE Internal certs.

Defaults:vro-plugin-user !requiretty
vro-plugin-user ALL = (root) NOPASSWD: /opt/puppetlabs/bin/puppet config
 print *
vro-plugin-user ALL = (root) NOPASSWD: !/opt/puppetlabs/bin/puppet config
 print *[[\:blank\:]]*
vro-plugin-user ALL = (root) NOPASSWD: /opt/puppetlabs/bin/puppet resource
 service puppet ensure=stopped
```

```
vro-plugin-user ALL = (root) NOPASSWD: /opt/puppetlabs/bin/puppet resource
 service puppet ensure=running enable=true
vro-plugin-user ALL = (root) NOPASSWD: /opt/puppetlabs/bin/facter -p
 puppetversion
vro-plugin-user ALL = (root) NOPASSWD: /opt/puppetlabs/bin/facter -p
 pe_server_version
vro-plugin-user ALL = (root) NOPASSWD: /opt/puppetlabs/bin/puppet agent -t
vro-plugin-user ALL = (root) NOPASSWD: /opt/puppetlabs/bin/puppet agent --
test --color=false --detailed-exitcodes
vro-plugin-user ALL = (root) NOPASSWD: /opt/puppetlabs/bin/puppet node
 purge *
vro-plugin-user ALL = (root) NOPASSWD: !/opt/puppetlabs/bin/puppet node
 purge *[[\:blank\:]]*
vro-plugin-user ALL = (root) NOPASSWD: !/opt/puppetlabs/bin/puppet node
 purge pe-internal-mcollective-servers
vro-plugin-user ALL = (root) NOPASSWD: !/opt/puppetlabs/bin/puppet node
 purge pe-internal-peadmin-mcollective-client
vro-plugin-user ALL = (root) NOPASSWD: /bin/ls -1 /etc/puppetlabs/code/
environments/
vro-plugin-user ALL = (root) NOPASSWD: /opt/puppetlabs/bin/puppet strings
 *
vro-plugin-user ALL = (root) NOPASSWD: /bin/find /etc/puppetlabs/code/
environments/*
```

Create an RBAC user

The vRO Plug-in uses an RBAC user to read console data. The username and password need to be the same as the ssh user.

1. Create a new user role for the vRO Plug-in.
   a) In the console, click **Access control** > **User roles**.
   b) In the **Name** field, enter a name for the vRO user.
   c) In the **Description** field, enter a description for the role, for example, vRO Plug-in user role.
   d) Click **Add role**.
2. Assign permissions to the newly created user role to allow for viewing node data.
   a) On the **User roles** page, click **vRO Puppet Plug-in**
   b) Click **Permissions**.
   c) In the **Add a permission** dropdown, add the following permissions:

Type	Action	Instance
Nodes	View node data from PuppetDB	
Agent	Run Puppet on the agent nodes	
Tasks	Run tasks	Tasks: all permitted on all nodes
Job Orchestrator	Start, stop, and view jobs	
Certificate requests	Accept and reject	

   d) Click **Commit**.
3. Create a new RBAC user for the vRO plugin.

   Ensure the name of the user is the same as the ssh user.

   a) In the console, click **Access control** > **Users**.
   b) In the **Full name** field, enter the user name.
   c) In the **Login** field, enter the user name.
   d) Click **Add local user**.

4. Add the new user to the user role you just created.
   a) On the **User role** page, click **User role**.
   b) Click **vRO Puppet user**.
   c) In the **Member users** tab, in the **User name** field, select the user you just created.
   d) Click **Add user**, and commit changes.
5. Set the password for the new user to be the same as the ssh password.
   a) On the **Users** page, click the user's full name.
   b) Click **Generate password reset**.
   c) Copy the link provided in the message and open it in a new browser window.

Autosign configuration

The Puppet vRO Plug-in installs the PE agent on newly deployed nodes.

To avoid manually approving new agent certificates, you can automate this process securely with an autosigning policy. Use the vra_puppet_plugin_prep module to configure autosigning using a shared key when classified on the primary server.

If you want autosigning but don't want to use the vra_puppet_plugin_prep module, you can configure the same shared-key autosigning manually. The csr_attributes.yaml contains the `challengePassword`, which can be used inside the autosigning script to confirm that the certificate can be signed.

The vra_puppet_plugin_prep module autosign script:

```
#!/opt/puppetlabs/puppet/bin/ruby
#
# A note on logging:
#   This script's stderr and stdout are only shown at the DEBUG level
#   of the primary server's logs. This means you won't see the error
 messages
#   in puppetserver.log by default. All you'll see is the exit code.
#
#   https://docs.puppet.com/puppet/latest/ssl_autosign.html#policy-
executable-api
#
# Exit Codes:
#   0 - A matching challengePassword was found.
#   1 - No challengePassword was found.
#   2 - The wrong challengePassword was found.
#
require 'puppet/ssl'

csr = Puppet::SSL::CertificateRequest.from_s(STDIN.read)
valid_pass = 'SecretPassword'

if pass = csr.custom_attributes.find do |attribute|
     ['challengePassword', '1.2.840.113549.1.9.7'].include? attribute['oid']
   end
else
  puts 'No challengePassword found. Not automatically accepting the
 request.'
  exit 1
end

if pass['value'] == valid_pass
  exit 0
else
  puts "challengePassword does not match: #{pass['value']}"
  exit 2
end
```

Install Puppet strings

Install the puppet-strings gem on the primary server by running the following commands:

```
puppet resource package rgen provider=puppet_gem
puppet resource package puppet-strings provider=puppet_gem
```

The puppet-strings gem is responsible for populating the descriptions of the roles in vRA. The role classes need the `@summary` tag to populate this section.

An example role with the `@summary` tag:

```
# This role installs a MySQL database and sample data

#

# @summary MySQL database server on Linux with sample data

class role::linux_mysql_database {

  include profile::linux_baseline

  include profile::mysql

  include profile::sample_data

}
```

Role class location

The Puppet vRO plugin is designed to populate the `$trusted['pp_role']` fact with the role specified in the blueprint.

These roles are populated in the vRA UI by looking for `<environment>/site/role/manifests/**/*.pp` files. The files are then parsed with `puppet-strings` to populate a summary. Classes in other locations in the control repository are not accessible to the Puppet configuration management item in the vRA blueprint. If you are not using the enterprise version of vRA, you can populate any class by setting the `Puppet.RoleClass` property in the blueprint.

See the vRO/vRA property reference for more information.

Node classification

The vRO Plug-in populates the `$trusted['pp_role']` fact with the classname of the role from the request.

Use the `$trusted['pp_role']` fact to classify the node with the class. The reference implementation uses a node group for each role with a rule that matches the role to the `trusted.extensions.pp_role` fact.

Example of a matching rule for the node group:

```
trusted.extensions.pp_role = role::webserver
```

Alternatively, if you use the `site.pp` for managing classification, you can leverage the `$trusted['pp_role']` for inclusion. For example:

```
if $trusted['pp_role'] and defined($trusted['pp_role']) {
  include $trusted['pp_role']
}
```

Use the method that complies with your existing configuration.

Environment group configuration

The PE console is responsible for classifying the node into the correct environment.

In the vRO configuration, each node has the `$trusted['pp_environment']` fact populated with the environment configured in vRA. Use the `trusted.extensions.pp_environment` in the rules for your environment groups. See the environment node group documentation for instructions on creating environment node groups and the rules documentation for adding rules to the environment node group.

An example rule for the development environment group:

```
trusted.extensions.pp_environment = development
```

Install and configure the Puppet plug-in

If you choose to install your own vRO, refer to the vRO documentation. If you have previously installed version 1.0 or 2.0 of the plug-in, you must completely remove it before installing version 3.1.

1. Download the Puppet vRO Plug-in's .vmoapp package from the VMware Solution Exchange.
2. Log in to the vRO server's control center at `https://<VRO-SERVER-IP-ADDRESS>:8283/vco-controlcenter`.
3. Click the **Plugins** tab.
4. Click **Install plug-in**.
5. Install the plug-in's .vmoapp package downloaded from the VMware Solution Exchange.
6. Read and accept the End User Terms and Conditions (EULA).
7. Click **Install**.
8. After the installation confirmation message appears, click the Startup options link in the message reminding you to restart the orchestrator server. In some versions of vRO, this message may not appear, but you must still restart the orchestrator server.
9. On the **Startup options** page, click **Restart**.

Add a primary server endpoint in vRA

Follow the vRA 7.4 docs to start using vRO plugins in vRA.

Run the Add a Puppet Enterprise (PE) primary server workflow provided by the Puppet vRO Plugin with these parameters:

- `Display Name for this Puppet master`: The name for this primary server
- `Hostname or IP Address`: The hostname or IP address of the PE master
- `SSH Port`: 22
- `SSH and RBAC Username`: The user you created
- `SSH and RBAC Password`: The password you configured
- `Use sudo for shell commands on this master?`: yes (no if using the root user)

After you have added the master endpoint, follow the Designing blueprints with Puppet features documentation.

Managing and provisioning infrastructure with vRA and Puppet

Once you have configured vRO and the Puppet vRO Plug-in, you can use vRealize Automation (vRA) to request servers using blueprints.

> **Note:** If you haven't yet installed vRA, refer to the vRA documentation.

Designing blueprints with Puppet features

In the previous version of the starter content we shipped Blueprints that you could install via CloudClient, but with vRealize Automation 7.4 Enterprise and the Puppet plug-in for vRealize Automation 3.2, it is simpler to create a new blueprint from scratch using the new Puppet component in the GUI. Follow these instructions to create your own blueprints.

Note: You can still access the previous version of these docs for consuming those prebuilt blueprints for vRA 7.x here and download them from this branch of the starter content.

- Create a Puppet Enterprise primary server and follow the instructions at the bottom of the starter content README to install the starter content.
- Remove any previous Puppet plug-ins and install the Puppet plug-in for vRealize Automation 3.0 into vRO.
- Follow the vRA 7.4 docs to add a Puppet endpoint (points to your primary server), and create your Puppet Enterprise blueprints using code from this repo that is now on your primary server.
  - Puppet endpoint - username: vro-plugin-user
  - Puppet endpoint - password: puppetlabs
  - Puppet endpoint - use sudo: true
  - Puppet component on blueprints - shared secret (cert autosigning): S3cr3tP@ssw0rd!

Note: For detailed information about designing vRA blueprints, consult the vRA blueprint documentation.

vRO/vRA property reference

The plug-in uses the following properties for blueprint and workflow development.

They can be used when creating traditional IaaS blueprints without the GUI component in vRA 7.4 Enterprise. There is a hierarchy of assignment for these properties. Properties that are set in the GUI override VRA properties set at the VM or tenant level. For certain properties there is a second override version of the property that takes precedence over the non-override version.

Here is a list of vRO JavaScript variables with their corresponding vRA property names, types, and override values where applicable:

puppetRoleClass

vRA property name: Puppet.RoleClass

Override value: Puppet.RoleClass.Override

Type: string

Description: The fully qualified class that implements the node's role.

puppetCodeEnvironment

vRA property name: Puppet.CodeEnvironment

Override value: Puppet.CodeEnvironment.Override

Type: string

Description: The environment on the master in which vRO should look for code.

puppetInstallMaster

vRA property name: Puppet.Master.InstallMaster

Type: string

Description: Optional FQDN or IP Address of the load balancer or compile master to install agent from. Defaults to the Puppet Master selected from vRO inventory.

puppetNodeCertname

vRA property name: Puppet.Node.Certname

Type: string

Description: The agent sets this based on the node's `certname`, which is based on its fully qualified domain name.

puppetNode

vRA property name: Puppet.Node.IPAddress

Type: string

Description: The IP Address of the node to install Puppet Agent on. Will default to the node name if left undefined.

ignoreChanges

vRA property name: Puppet.Node.IgnoreChanges

Type: boolean

Description: Ignores changes in the Puppet Run. If True, Puppet Runs that exits with changes (exit code 2) will still pass. Defaults to false.

puppetAutosignSharedSecret

vRA property name: Puppet.Autosign.SharedSecret

Override value: Puppet.Autosign.SharedSecret.Override

Type: secureString

Description: The shared secret that nodes should provide to the master in order to autosign certificate requests.

sshUsername

vRA property name: Puppet.SSH.Username

Override value: Puppet.SSH.Username.Override

Type: string

Description: Username used to connect to a node via SSH.

sshPassword

vRA property name: Puppet.SSH.Password

Override value: Puppet.SSH.Password.Override

Type: secureString

Description: Password used to connect to a node via SSH.

winRMUsername

vRA property name: Puppet.WinRM.Username

Override value: Puppet.WinRM.Username.Override

Type: string

Description: Username used to connect to a node via WinRM.

useSudo

vRA property name: Puppet.SSH.UseSudo

Override value: Puppet.SSH.UseSudo.Override

Type: boolean

Description: Use sudo commands run on a node via SSH. This requires `NOPASSWD` sudo for the user defined in `sshUsername`.

winRMPassword

vRA property name: Puppet.WinRM.Password

Override value: Puppet.WinRM.Password.Override

Type: secureString

Description: Password used to connect to a node via WinRM.

puppetAgentAccountUser

vRA property name: Puppet.Windows.AgentAccountUser

Override value: Puppet.Windows.AgentAccountUser.Override

Type: string

Description: User for the puppet agent service to run as rather than default of Local System

puppetAgentAccountPassword

vRA property name: Puppet.Windows.AgentAccountPassword

Override value: Puppet.Windows.AgentAccountPassword.Override

Type: secureString

Description: Password for the agent service user

puppetAgentAccountDomain

vRA property name: Puppet.Windows.AgentAccountDomain

Override value: Puppet.Windows.AgentAccountDomain.Override

Type: string

Description: Domain, if any, for the agent service user

UseHTTPS

vRA property name: Puppet.WinRM.UseHTTPS

Type: boolean

Description: If true, use HTTPS for WinRM, if false use HTTP

winRMAuthMethod

vRA property name: Puppet.WinRM.AuthMethod

Type: enum[Basic,Kerberos]

Description: Basic or Kerberos auth method for WinRM

sshKeyPath

vRA property name: Puppet.SSH.KeyPath

Override value: Puppet.SSH.KeyPath.Override

Type: string

Description: A path to the ssh key that can be used instead of password authentication

sshPassphrase

vRA property name: Puppet.SSH.Passphrase

Override value: Puppet.SSH.Passphrase.Override

Type: secureString

Description: A passphrase used for the sshKeyPath

puppetApptier

vRA property name: Puppet.Extensions.Apptier

Override value: Puppet.Extensions.Apptier.Override

Type: string

Description: pp_apptier certificate extension. For example "dev", "uat", "production" etc. Not to be confused with the Puppet code environment.

puppetDepartment

vRA property name: Puppet.Extensions.Department

Override value: Puppet.Extensions.Department.Override

Type: string

Description: pp_department certificate extension. For example, "finance", "digital", etc.

puppetService

vRA property name: Puppet.Extensions.Service

Override value: Puppet.Extensions.Service.Override

Type: string

Description: pp_service certificate extension. For example, "puppet", "corporate_web", etc.

keepFailedVMs

vRA property name: Puppet.Debug.KeepFailedVMs

Type: boolean

Description: To enable returning a successful result despite failures in the module.

vRO/vRA actions reference

The Puppet plug-in ships with several actions that can be used in workflows and integrations with vRA, for instance to populate the contents of input fields or dropdown menus.

For more information, see the vRA documentation for actions.

Action name	Description
escapeShellArgument	Used internally by the plugin to escape a string used in a shell command.
escapePowerShellValue	Used internally by the plugin to escape a string used in a PowerShell command.
escapeJSON	Used internally by the plugin to escape a JSON string for stuctured facts or other uses.
getSectionText	Used internally by the plugin for parsing Error messages.
formatShellArguments	Used internally by the plugin to format and escape a set of strings containing arguments to a shell command. Calls `escapeShellArgument`.
executeCommand	Used internally by the plugin to execute a shell command on a Linux Puppet master.
getMasters	Returns an array of strings containing the UUIDs of all of the Puppet:Master objects in the vRO inventory. Returns `[""]` if there are no Puppet:Master objects.

Action name	Description
getMasterByUUID	Returns a Puppet:Master object given a UUID string. Returns null if there is no object matching that UUID.
getEnvironments	Returns an array of strings which are the environment names on the Puppet:Master specified by a UUID. Returns `[""]` if there are no environments.
getRoleClasses	Returns an array of strings which are role class names on the Puppet:Master specified by a UUID and in a specified environment. Returns `[""]` if there are no role classes there.
getRoleClassesWithDescriptions	Used internally, returns specially formated JSON string used by vRA 7.3 Enterprise with the role classes and their descriptions from a master's environment. Throws an error if no master UUID or environment name provided. Optionally accepts a filter regex string to limit results.

All actions are visible on the "Actions" tab of the Java vRO client when in "Design" view, where you can view the full source code of each action, including parameters and return types.

Encrypting content with eyaml

Securing passwords used in the manifest is beyond the scope of this reference implementation. As a starting point, many Puppet deployments use Hiera, a key/value lookup tool for configuration, with eyaml, or encrypted YAML, to solve this problem.

This solution not only provides secure storage for the password value, but also provides parameterization to support reuse, opening the door to easy password rotation policies across an entire network of nodes.

For information, see the Hiera documentation and the blog post Encrypt your data using Hiera-eyaml.

Troubleshooting

> **Note:** The Puppet Plug-in for VMware vRealize Automation is for vRA 7. Starting in vRA 8, Puppet is integrated directly into vRA by VMware and does not require this plug-in. For details, see the VMware vRealize Automation documentation or contact VMware support.

Configuring Windows domain and forest on a new Puppet agent node

The Puppet vRO Plug-in contains features to install and run a Puppet agent on provisioned nodes. Configure networking on newly provisioned nodes to allow the download of the install and request a certificate from the primary server. Set up the domain and forest configuration on Windows hosts before running the "Install PE Agent on Windows Node" workflow to allow for installation of the agent and potential classification of the node into a defined environment.

Running workflows after upgrading from plug-in version results in errors

If you are trying to upgrade from plug-in version 1.0 or 2.0, note that the plug-in **does not** currently support upgrades from the previous vRO Puppet plug-in versions. If you're using any previous version of the plug-in, you mustcompletely remove it before installing. The vRO starter content repository includesa script to assist with removing the plug-in.

Related topics: Puppet knowledge base: Removing the Puppet plugin from VMware vRealize Automation

vRO release notes

See the vRO Puppet plug-in documentation for more information about the plug-in and instructions on installing and using it.

CAUTION: The plug-in does not support upgrades from previous vRO Puppet plug-in versions. If you're using a previous version of the plug-in, you must completely remove it before installing a newer version. For info on removing the plug-in, see the VMWare knowledge base.

- vRO plug-in 3.3 on page 17
- vRO plug-in 3.2 on page 17

Released 9 July 2019

- vRO plug-in 3.1 on page 18

Released 23 May 2018

- vRO Puppet plug-in 3.0 on page 18

Released May 16, 2017.

vRO plug-in 3.3

Released 16 October 2019

This section contains improvements and fixes in this version.

New in Version 3.3

- Adds `pp_apptier`, `pp_department` and `pp_service` certificate extensions to Puppet agent install workflows.
- Adds param to run Puppet workflows to ignore changes in Puppet run. This support allows those that expect changes or non-zero exit codes from failing the workflow.
- Adds new vRA Properties:
- puppetApptier
- puppetDepartment
- puppetService
- ignoreChanges

Fixes in Version 3.3

- Fixes running Puppet from PE master to support multiple RBAC service URL on HA masters. Automatically selects the first in the list.
- Fixes an infinite retry bug in the run Puppet from PE master workflow.
- Properly purges node certs on failure of installing PE Agent with role if `keepFailedVMs` is `true`.

vRO plug-in 3.2

Released 9 July 2019

This section contains improvements and fixes in this version.

New in this version

- Workflow to run Puppet from the master orchestrator, works with both Linux and Windows nodes managed by the master.
- Separated the “drop facts” functionality into its own workflow.
- Adds an optional vRA parameter `Puppet.Node.IPAddress` to specify an IPAddress for the node to install an agent on. Used in the event that a hostname is not yet defined in DNS.

- Adds an optional vRA parameter, allowing you to specify an FQDN or IP address of a compile master or load balancer to install from, that is a different server to the master defined in the vRO inventory to manage cert signing.
- Installing an agent with a user-defined environment now sets the `environment` in the puppet.conf of the agent.

Fixes in this version

- Fixes a backwards compatibility bug introduced in 3.1 that required users to specify a hostname (FQDN) to the “Install PE Agent on Linux Node with SSH” and “Install PE Agent on Windows Node with PowerShell” workflows. The fix allows for IP addresses again.
- Fixes an issue where a user specified certname wasn't being passed into the agent install script.
- Fixes an issue where the plugin did not manage ssh HostKey checking for the ssh connection to the master.
- Fixes shell escaping for Windows paths.
- Fixes inconsistent behavior leading to failures during “drop facts” functionality when installing an agent on Windows.

vRO plug-in 3.1

Released 23 May 2018

This section contains improvements and fixes in this version.

New in this version

- Support for an SSH key for login to Linux agents.
- First run of Puppet is controlled tightly for better reliability instead of an automatic async run.
- Hostnames are used throughout instead of IP addresses to connect to new agents.
- Many properties have override capability. For example, Puppet.RoleClass.Override
- Accepts Windows MSI installer props so users can set alternate run-as user for Puppet agent (to access network shares, etc).
- Property to exit install workflow with success to preserve hosts for debugging. Puppet.Debug.KeepFailedVMs
- Properties to specify desired Windows auth schemes: Puppet.WinRM.AuthMethod, Puppet.WinRM.UseHTTPS
- No longer dependent on vCenter, which means they might be suitable for AWS, Azure, Openstack provisioning with vRA (not currently with Puppet drag-n-drop component).

Fixes in this version

- sudo is used everywhere in Linux workflows and actions. It was absent in a couple of places previously causing issues on some OSs.
- Remove Master workflow does not hard fail if master not present. Leaves log message and finishes with success so vRA gets no error when removing endpoint.
- Better handling of warnings in Windows to prevent transient errors especially on Windows 2016.
- Resolution of Puppet master restart issues when purging nodes. Now Puppet master handles CRL reload automatically with no downtime (PE 2017.3 and higher).

vRO Puppet plug-in 3.0

Released May 16, 2017.

- Adds support for integration within the vRealize Automation Enterprise GUI in version 7.3 and higher.
- Role classes can be dynamically fetched from the Puppet master.
- vRA GUI displays each role class’s description based on the `@summary` puppet-strings tag if it is present.
- Major version release that drops support for open source Puppet.
- One vRA/vRO instance can support multiple Puppet masters.
- `csr_attributes.yaml` file is deleted after provisioning if you are using the shared-secret autosigning workflow.

- If Puppet agent runs during provision do not succeed, the Puppet agent service is left in running state.
- Adjustments to avoid timeouts in Windows provisioning.
- `Puppet.SSH.useSudo` is now exposed as a custom property.
- Improved error handling in the Add a Puppet Master workflow.
- Improved error message if Puppet:Master is not set.
- Run Puppet workflows no longer perform one more Puppet run than necessary.