Red Hat Satellite 6.15

Managing configurations using Puppet integration

Configure Puppet integration in Satellite and use Puppet classes to configure your hosts

Last Updated: 2024-07-16

Red Hat Satellite 6.15 Managing configurations using Puppet integration

Configure Puppet integration in Satellite and use Puppet classes to configure your hosts

Red Hat Satellite Documentation Team
satellite-doc-list@redhat.com

Legal Notice

Abstract

This guide describes how to enable Puppet integration with Satellite, configure the Puppet agent on hosts, how to import Puppet modules, and how to use the Puppet modules to enforce configurations on hosts managed in your Red Hat Satellite infrastructure.

Table of Contents

PROVIDING FEEDBACK ON RED HAT DOCUMENTATION

We appreciate your feedback on our documentation. Let us know how we can improve it.

Use the **Create Issue** form in Red Hat Jira to provide your feedback. The Jira issue is created in the Red Hat Satellite Jira project, where you can track its progress.

Prerequisites

- Ensure you have registered a Red Hat account.

Procedure

1. Click the following link: **Create Issue**. If Jira displays a login error, log in and proceed after you are redirected to the form.
2. Complete the **Summary** and **Description** fields. In the **Description** field, include the documentation URL, chapter or section number, and a detailed description of the issue. Do not modify any other fields in the form.
3. Click **Create**.

CHAPTER 1. INTRODUCING CONFIGURATION MANAGEMENT USING PUPPET

You can use Puppet to manage and automate configurations of hosts. Puppet uses a declarative language to describe the *desired state* of hosts.

Puppet increases your productivity as you can administer multiple hosts simultaneously. At the same time, it decreases your configuration effort as Puppet makes it easy to verify and possibly correct the state of the hosts.

Additional resources

- Open Source Puppet documentation
- Puppet Forge – a repository of pre-built Puppet modules

1.1. HOW PUPPET INTEGRATES WITH SATELLITE

Puppet uses a server-agent architecture. The Puppet server is the central component that stores configuration definitions. Satellite Server or any Capsules are typically deployed with the Puppet server and Satellite acts as an External Node Classifier (ENC) for such Puppet server. Hosts run the Puppet agent that communicates with the Puppet server.

The Puppet agent collects *facts* about a host and reports them to the Puppet server on each run. You can display the Puppet *facts* in JSON format by running **puppet facts** on a host.

The Puppet server forwards *facts* to Satellite and Satellite stores them for later use. Based on the *facts* and other definitions, Satellite constructs the ENC answer to the Puppet server. The Puppet server compiles a *catalog* based on the ENC answer and sends the *catalog* to the Puppet agent.

The Puppet agent evaluates the system state on the host. If the Puppet agent finds differences, known as *drifts*, between the *desired state* defined in the *catalog* and the actual state, it enforces correction of the state of the host. The Puppet agent then reports correction results back to the Puppet server, which reports them to Satellite.

Puppet modules

The *desired state* of a host is defined in a *catalog*. The *catalog* is compiled from Puppet manifests of one or more Puppet modules assigned to the host. A Puppet module is a collection of classes, manifests, resources, files, and templates. The Puppet modules work as components of host configuration definitions.

Smart Class parameters

You can override parameters of a Puppet module using Smart Class parameters if the module supports the use of parameters. You can define the parameters in your Satellite as *key-value* pairs, which behave similar to host parameters or Ansible variables.

Puppet environments

You can also create multiple Puppet environments to control versions of configuration definitions or to manage variants of the definitions, and to test the definitions before you deploy them on production.

High-level integration steps

Puppet integration with Satellite involves the following high-level steps:

1. Enable Puppet integration.
2. Import Puppet agent packages into Satellite. Puppet agent packages can be managed like any other content with Satellite by enabling Red Hat Repositories and by using activation keys and content views.
3. Install Puppet agent on hosts during provisioning, registration, manually, or by remote job execution.

Additional resources

- Managing content
- Registering Hosts in the *Managing Hosts Guide*
- Configuring and Setting Up Remote Jobs in the *Managing Hosts Guide*

The following procedures outline how to use a Puppet module to install, configure, and manage the *ntp* service to provide examples.

1.2. SUPPORTED PUPPET VERSIONS AND SYSTEM REQUIREMENTS

Before you begin with the Puppet integration, review the supported Puppet versions and system requirements.

Supported Puppet Versions

Satellite supports Puppet 7. Ensure that the Puppet modules used to configure your hosts are compatible with Puppet 7.

System Requirements

Before you begin integrating Puppet with your Satellite, ensure that you meet the system requirements. For more information, see System Requirements for Puppet 7 in the *Open Source Puppet* documentation.

1.3. ENABLING PUPPET INTEGRATION WITH SATELLITE

By default, Satellite does not have any Puppet integration configured. You need to enable the integration as is appropriate for your situation. This means that you can configure Satellite to manage and deploy Puppet server on Satellite Server or Capsule Servers. Additionally, you can deploy Puppet server to Satellite externally and integrate it with Satellite for reporting, facts, and external node classification (ENC).

Procedure

1. Enable Puppet integration and install Puppet server on Satellite Server:

```
# satellite-installer \
--enable-foreman-cli-puppet \
--enable-foreman-plugin-puppet \
--enable-puppet \
--foreman-proxy-puppet true \
--foreman-proxy-puppetca true \
--puppet-server true
```

2. If you want to use Puppet integration on Capsule Servers, enable Puppet integration and install Puppet server on Capsule Servers:

```
# satellite-installer \
--enable-puppet \
--foreman-proxy-puppet true \
--foreman-proxy-puppetca true \
--puppet-server true
```

1.4. INSTALLING AND CONFIGURING PUPPET AGENT DURING HOST PROVISIONING

You can install and configure the Puppet agent on a host during the provisioning process. A configured Puppet agent is required on the host for Puppet integration with your Satellite.

Prerequisites

- Puppet must be enabled in your Satellite. For more information, see Section 1.3, "Enabling Puppet integration with Satellite".
- You enabled and synchronized the **Satellite Client 6** repository to Satellite. For more information, see Importing Content in *Managing content*.
- You created an activation key that enables the **Satellite Client 6** repository for hosts. For more information, see Managing Activation Keys in *Managing content*.

Procedure

1. Navigate to **Hosts** > **Templates** > **Provisioning Templates**.
2. Select a provisioning template depending on your host provisioning method. For more information, see Kinds of Provisioning Templates in *Provisioning hosts*.
3. Ensure the **puppet_setup** snippet is included as follows:

```
<%= snippet 'puppet_setup' %>
```

   Note that this snippet is already included in the templates shipped with Satellite, such as **Kickstart default** or **Preseed default**.

4. Enable the Puppet agent using a host parameter in global parameters, a host group, or for a single host. Add a host parameter named **enable-puppet7**, select the **boolean** type, and set the value to **true**.
5. Set configuration for the Puppet agent.
   - If you use an integrated Puppet server, ensure that you select a Puppet Capsule, Puppet CA Capsule, and Puppet environment when you create a host.
   - If you use a non-integrated Puppet server, either set the following host parameters in global parameters, or a host group, or when you create a host:
     - Add a host parameter named **puppet_server**, select the **string** type, and set the value to the hostname of your Puppet server, such as **puppet.example.com**.

- Optional: Add a host parameter named **puppet_ca_server**, select the **string** type, and set the value to the hostname of your Puppet CA server, such as **puppet-ca.example.com**. If **puppet_ca_server** is not set, the Puppet agent will use the same server as **puppet_server**.
- Optional: Add a host parameter named **puppet_environment**, select the **string** type, and set the value to the Puppet environment you want the host to use.

6. Ensure your host has access to the Puppet agent packages from Satellite Server by using an appropriate activation key.

1.5. INSTALLING AND CONFIGURING PUPPET AGENT DURING HOST REGISTRATION

You can install and configure the Puppet agent on the host during registration. A configured Puppet agent is required on the host for Puppet integration with your Satellite.

Prerequisites

- Puppet must be enabled in your Satellite. For more information, see Section 1.3, "Enabling Puppet integration with Satellite".
- You enabled and synchronized the **Satellite Client 6** repository to Satellite. For more information, see Importing Content in *Managing content*.
- You created an activation key that enables the **Satellite Client 6** repository for hosts. For more information, see Managing Activation Keys in *Managing content*.

Procedure

1. In the Satellite web UI, navigate to **Configure** > **Global Parameters** to add host parameters globally. Alternatively, you can navigate to **Configure** > **Host Groups** and edit or create a host group to add host parameters only to a host group.
2. Enable the Puppet agent using a host parameter in global parameters or a host group. Add a host parameter named **enable-puppet7**, select the **boolean** type, and set the value to **true**.
3. Specify configuration for the Puppet agent using the following host parameters in global parameters or a host group:
   - Add a host parameter named **puppet_server**, select the **string** type, and set the value to the hostname of your Puppet server, such as **puppet.example.com**.
   - Optional: Add a host parameter named **puppet_ca_server**, select the **string** type, and set the value to the hostname of your Puppet CA server, such as **puppet-ca.example.com**. If **puppet_ca_server** is not set, the Puppet agent will use the same server as **puppet_server**.
   - Optional: Add a host parameter named **puppet_environment**, select the **string** type, and set the value to the Puppet environment you want the host to use.

   Until the BZ2177730 is resolved, you must use host parameters to specify the Puppet agent configuration even in integrated setups where the Puppet server is a Capsule Server.
4. Navigate to **Hosts** > **Register Host** and register your host using an appropriate activation key. For more information, see Registering Hosts in *Managing hosts*.

5. Navigate to **Infrastructure** > **Capsules**.
6. From the list in the **Actions** column for the required Capsule Server, select **Certificates**.
7. Click **Sign** to the right of the required host to sign the SSL certificate for the Puppet agent.

1.6. INSTALLING AND CONFIGURING PUPPET AGENT MANUALLY

You can install and configure the Puppet agent on a host manually. A configured Puppet agent is required on the host for Puppet integration with your Satellite.

Prerequisites

- Puppet must be enabled in your Satellite. For more information, see Section 1.3, "Enabling Puppet integration with Satellite".
- The host must have a Puppet environment assigned to it.
- The **Satellite Client 6** repository must be enabled and synchronized to Satellite Server, and enabled on the host. For more information, see Importing Content in *Managing content*.

Procedure

1. Log in to the host as the **root** user.
2. Install the Puppet agent package.
   - On hosts running Red Hat Enterprise Linux 8 and above:

     ```
     # dnf install puppet-agent
     ```

   - On hosts running Red Hat Enterprise Linux 7 and below:

     ```
     # yum install puppet-agent
     ```

3. Add the Puppet agent to **PATH** in your current shell using the following script:

   ```
   . /etc/profile.d/puppet-agent.sh
   ```

4. Configure the Puppet agent. Set the **environment** parameter to the name of the Puppet environment to which the host belongs:

   ```
   # puppet config set server satellite.example.com --section agent
   # puppet config set environment My_Puppet_Environment --section agent
   ```

5. Start the Puppet agent service:

   ```
   # puppet resource service puppet ensure=running enable=true
   ```

6. Create a certificate for the host:

   ```
   # puppet ssl bootstrap
   ```

7. In the Satellite web UI, navigate to **Infrastructure** > **Capsules**.
8. From the list in the **Actions** column for the required Capsule Server, select **Certificates**.
9. Click **Sign** to the right of the required host to sign the SSL certificate for the Puppet agent.
10. On the host, run the Puppet agent again:

```
# puppet ssl bootstrap
```

1.7. PERFORMING CONFIGURATION MANAGEMENT

After you deploy Puppet agent on a host, you can start performing configuration management with Puppet. This involves the following high-level steps:

1. Managing Puppet modules on the Puppet server, that is installing and updating them.
2. Importing Puppet classes and environments from Puppet modules into Satellite.
3. Optional: Creating config groups from Puppet classes.
4. Configuring overrides of Smart Class parameters on various levels.
5. Assigning Puppet classes or config groups to host groups or individual hosts.
6. Configuring intervals for runs of the Puppet agent on hosts and for configuration enforcement runs of the Puppet server.
7. Monitoring configuration management using reports in the Satellite web UI. For more information, see Monitoring Resources in *Administering Red Hat Satellite*.
8. Configuring email notifications. For more information, see Configuring Email Notification Preferences in *Administering Red Hat Satellite*.

After assigning Puppet classes or config groups, Satellite runs configuration management automatically in the configured intervals to enforce Puppet configuration on your hosts, or you can initiate it manually on demand with the **Run Puppet Once** feature. For more information, see Section 9.1, “Running Puppet once using SSH”.

1.8. DISABLING PUPPET INTEGRATION WITH SATELLITE

To discontinue using Puppet in your Satellite, follow this procedure.

Note that the command without the **--remove-all-data** argument removes all Puppet-related data in Satellite database. With the **--remove-all-data** argument, the command additionally removes Puppet server data files, including Puppet environments.

WARNING

If you disable Puppet with the **--remove-all-data** argument, you will not be able to re-enable Puppet afterwards. This is a known issue, see the Bug 2087067.

Prerequisites

- Puppet is enabled on Satellite.

Procedure

1. If you have used Puppet server on any Capsules, disable Puppet server on all Capsules:

```
# satellite-maintain plugin purge-puppet --remove-all-data
```

2. Disable Puppet server on Satellite Server:

```
# satellite-maintain plugin purge-puppet --remove-all-data
```

CHAPTER 2. MANAGING PUPPET MODULES

2.1. INSTALLING A PUPPET MODULE ON SATELLITE SERVER

You can install a pre-built Puppet module from the Puppet Forge. The Puppet Forge is a repository that provides Puppet modules contributed by the community. Puppet modules flagged as *supported* are officially supported and tested by Puppet Inc.

This example shows how to add the *ntp module* to hosts.

Procedure

1. Navigate to forge.puppet.com and search for **ntp**. One of the first modules is *puppetlabs/ntp*.
2. Connect to your Satellite Server using SSH and install the Puppet module:

```
# puppet module install puppetlabs-ntp -i
/etc/puppetlabs/code/environments/production/modules
```

Use the **-i** parameter to specify the path and Puppet environment, for example **production**.

Once the installation is completed, the output looks as follows:

```
Notice: Preparing to install into /etc/puppetlabs/code/environments/production/modules ...
Notice: Created target directory /etc/puppetlabs/code/environments/production/modules
Notice: Downloading from https://forgeapi.puppet.com ...
Notice: Installing -- do not interrupt ...
/etc/puppetlabs/code/environments/production/modules
|-| puppetlabs-ntp (v8.3.0)
  |-- puppetlabs-stdlib (v4.25.1) [/etc/puppetlabs/code/environments/production/modules]
```

An alternative way to install a Puppet module is to copy a folder containing the Puppet module to the module path as mentioned above. Ensure to resolve its dependencies manually.

2.2. UPDATING A PUPPET MODULE

You can update an existing Puppet module using the **puppet** command.

Procedure

1. Connect to your Puppet server using SSH and find out where the Puppet modules are located:

```
# puppet config print modulepath
```

This returns output as follows:

```
/etc/puppetlabs/code/environments/production/modules:/etc/puppetlabs/code/environments/com
mon:/etc/puppetlabs/code/modules:/opt/puppetlabs/puppet/modules:/usr/share/puppet/modules
```

2. If the module is located in the path as displayed above, the following command updates a module:

```
# puppet module upgrade module name
```

CHAPTER 3. IMPORTING PUPPET CLASSES AND ENVIRONMENTS INTO SATELLITE

Import Puppet classes and environments from the installed Puppet modules to Satellite Server or any attached Capsule Server before you assign any of the classes to hosts.

Prerequisites

- Ensure to select **Any Organization** and **Any Location** as context, otherwise the import might fail.

Procedure

1. In the Satellite web UI, navigate to **Configure** > **Puppet ENC** > **Classes** or **Configure** > **Puppet ENC** > **Environments**.
2. Click **Import** in the upper right corner and select which Capsule you want to import modules from. You may typically choose between your Satellite Server or any attached Capsule Server.
3. Select the Puppet environments to import using checkboxes on the left.
4. Click **Update** to import the Puppet environments and classes to Satellite.
5. The import should result in a notification as follows:

> Successfully updated environments and Puppet classes from the on-disk Puppet installation

CHAPTER 4. CREATING A CUSTOM PUPPET ENVIRONMENT

You can create a Puppet environment within your Satellite.

Procedure

1. In the Satellite web UI, navigate to **Configure > Puppet Environments**.
2. Click **Create Puppet Environment** to create a Puppet environment.
3. Enter a **Name**, alphanumeric characters and underscores are allowed, such as **example_environment**.
4. Optional: Set a location context.
5. Optional: Set an organization context.
6. Click **Submit** to create the Puppet environment.

Note that before you run an import of Puppet modules into Satellite, the environment must already exist as the folder **/etc/puppetlabs/code/environments/*example_environment*** on the Puppet server and contain installed Puppet modules.

CHAPTER 5. CREATING A PUPPET CONFIG GROUP

A Puppet config group is a named list of Puppet classes that allows you to combine their capabilities and assign them to hosts at a click. This is equivalent to the concept of profiles in pure Puppet.

Procedure

1. In the Satellite web UI, navigate to **Configure** > **Puppet ENC** > **Config Groups**.
2. Click **Create Config Group**.
3. Select the classes you want to add to the config group.
    a. Choose a meaningful **Name** for the Puppet config group.
    b. Add selected Puppet classes to the **Included Classes** field.
4. Click **Submit** to save the changes.

CHAPTER 6. CONFIGURING PUPPET SMART CLASS PARAMETERS

6.1. PUPPET PARAMETER HIERARCHY

Puppet parameters are structured hierarchically. Parameters at a lower level override parameters of the higher levels:

1. Global parameters
2. Organization parameters
3. Location parameters
4. Host group parameters
5. Host parameters

For example, host specific parameters override the parameter at any higher level, and location parameters only override parameters at the organization or global level. This feature is especially useful when you use locations or organizations to group hosts.

6.2. OVERRIDING A SMART CLASS PARAMETER GLOBALLY

You can configure a Puppet class after you have imported it to Satellite Server. This example overrides the default list of ntp servers.

Procedure

1. In the Satellite web UI, navigate to **Configure** > **Puppet ENC** > **Classes**.
2. Select the **ntp** Puppet class to change its configuration.
3. Select the **Smart Class Parameter** tab and search for **servers**.
4. Ensure the **Override** checkbox is selected.
5. Set the **Parameter Type** drop down menu to *array*.
6. Insert a list of ntp servers as **Default Value**:

```
["0.de.pool.ntp.org","1.de.pool.ntp.org","2.de.pool.ntp.org","3.de.pool.ntp.org"]
```

An alternative way to describe the array is the **yaml** syntax:

```
- 0.de.pool.ntp.org
- 1.de.pool.ntp.org
- 2.de.pool.ntp.org
- 3.de.pool.ntp.org
```

7. Click **Submit** to change the default configuration of the Puppet module **ntp**.

6.3. OVERRIDING A SMART CLASS PARAMETER FOR AN ORGANIZATION

You can use groups of hosts to override Puppet parameters for multiple hosts at once. The following example chooses the *organization* context to illustrate setting context based parameters.

Note that *organization*-level Puppet parameters are overridden by *location*-level Puppet parameters.

Procedure

1. In the Satellite web UI, navigate to **Configure** > **Puppet ENC** > **Classes**.
2. Click a class name to select a class.
3. On the **Smart Class Parameter** tab, select a parameter.
4. Use the **Order** list to define the hierarchy of the Puppet parameters. The individual host (**fqdn**) marks the most and the organization context (**organization**) the least relevant.
5. Check **Merge Overrides** if you want to add all further matched parameters after finding the first match.
6. Check **Merge Default** if you want to also include the default value even if there are more specific values defined.
7. Check **Avoid Duplicates** if you want to create a list of unique values for the selected parameter.
8. The **matcher** field requires an *attribute type* from the *order* list.
9. Optional: Click **Add Matcher** to add more matchers.
10. Click **Submit** to save the changes.

6.4. OVERRIDING A SMART CLASS PARAMETER FOR A LOCATION

You can use groups of hosts to override Puppet parameters for multiple hosts at once. The following examples chooses the *location* context to illustrate setting context based parameters.

Procedure

1. In the Satellite web UI, navigate to **Configure** > **Puppet ENC** > **Classes**.
2. Click a class name to select a class.
3. On the **Smart Class Parameter** tab, select a parameter.
4. Use the **Order** list to define the hierarchy of the Puppet parameters. The individual host (**fqdn**) marks the most and the location context (**location**) the least relevant.
5. Check **Merge Overrides** if you want to add all further matched parameters after finding the first match.
6. Check **Merge Default** if you want to also include the default value even if there are more specific values defined.
7. Check **Avoid Duplicates** if you want to create a list of unique values for the selected parameter.

8. The **matcher** field requires an *attribute type* from the *order* list. For example, you can choose **Paris** as *location* context and set the value to French ntp servers.
9. Optional: Click **Add Matcher** to add more matchers.
10. Click **Submit** to save the changes.

6.5. OVERRIDING A SMART CLASS PARAMETER ON AN INDIVIDUAL HOST

You can override parameters on individual hosts. This is recommended if you have multiple hosts and only want to make changes to a single one.

Procedure

1. In the Satellite web UI, navigate to **Hosts > All Hosts**.
2. Click a host name to select a host.
3. Click **Edit**.
4. On the **Host** tab, select a Puppet **Environment**.
5. Select the **Puppet ENC** tab.
6. Click **Override** to edit the Puppet parameter.
7. Click **Submit** to save the changes.

CHAPTER 7. ASSIGNING A PUPPET CLASS TO A HOST GROUP

Use a host group to assign the *ntp* Puppet class to multiple hosts at once. Every host you deploy based on this host group has this Puppet class installed.

Procedure

1. In the Satellite web UI, navigate to **Configure > Host Groups** to create a host group or edit an existing one.
2. On the **Host Group** tab, set the following parameters:
    a. The **Lifecycle Environment** describes the stage in which certain versions of content are available to hosts.
    b. The **Content View** is comprised of products and allows for version control of content repositories.
    c. The **Environment** allows you to supply a group of hosts with their own dedicated configuration.
3. Navigate to the **Puppet ENC** tab.
4. Add the Puppet class to the *Included Classes* or to the *Included Config Groups* if a Puppet config group is configured.
5. Click **Submit** to save the changes.

CHAPTER 8. ASSIGNING A PUPPET CLASS TO AN INDIVIDUAL HOST

Procedure

1. In the Satellite web UI, navigate to **Hosts** > **All Hosts**.
2. Locate the host you want to add the **ntp** Puppet class to and click **Edit**.
3. Select the **Puppet ENC** tab and look for the ntp class.
4. Click the **+** symbol next to **ntp** to add the *ntp submodule* to the list of *included classes*.
5. Click **Submit** to save your changes.

   TIP

   If the *Puppet classes* tab of an individual host is empty, check if it is assigned to the proper Puppet environment.

6. Verify the Puppet configuration.
   a. Navigate to **Hosts** > **All Hosts** and select the host.
   b. From the top overflow menu, select **Legacy UI**.
   c. Under **Details**, click **Puppet YAML**. This produces output similar as follows:

```
---
parameters:
  // shortened YAML output
classes:
  ntp:
    servers:
'["0.de.pool.ntp.org","1.de.pool.ntp.org","2.de.pool.ntp.org","3.de.pool.ntp.org"]'
environment: production
...
```

7. Verify the ntp configuration.
   Connect to your host using SSH and check the content of **/etc/ntp.conf**.

   This example assumes your host is running *CentOS 7*. Other operating systems may store the ntp config file in a different path.

   TIP

   You may need to run the Puppet agent on your host by executing the following command:

```
# puppet agent -t
```

8. Running the following command on the host checks which ntp servers are used for clock synchronization:

```
# cat /etc/ntp.conf
```

This returns output similar as follows:

```
# ntp.conf: Managed by puppet.
server 0.de.pool.ntp.org
server 1.de.pool.ntp.org
server 2.de.pool.ntp.org
server 3.de.pool.ntp.org
```

You now have a working ntp module which you can add to a host or group of hosts to roll out your ntp configuration automatically.

CHAPTER 9. ENFORCING PUPPET CONFIGURATION ON HOSTS

You can enforce configuration from Satellite either manually on demand (run once) or automatically in configurable intervals.

9.1. RUNNING PUPPET ONCE USING SSH

Assign the proper job template to the *Run Puppet Once* feature to run Puppet on hosts.

Procedure

1. In the Satellite web UI, navigate to **Administer** > **Remote Execution Features**.
2. Select the **puppet_run_host** remote execution feature.
3. Assign the **Run Puppet Once – SSH Default** job template.

Run Puppet on hosts by running a job and selecting category **Puppet** and template **Run Puppet Once - SSH Default**. Alternatively, click **Run Puppet Once** in the **Schedule Remote Job** drop down menu on the host details page.

9.2. UNDERSTANDING INTERVALS OF AUTOMATIC ENFORCEMENT

Satellite considers hosts to be out of sync if the last Puppet report is older than the combined values of **outofsync_interval** and **puppet_interval** set in minutes. By default, the Puppet agent on your hosts runs every 30 minutes, the **puppet_interval** is set to 35 minutes and the global **outofsync_interval** is set to 30 minutes.

The effective time after which hosts are considered out of sync is the sum of **outofsync_interval** and **puppet_interval**. For example, setting the global **outofsync_interval** to 30 and the **puppet_interval** to 60 results in a total of 90 minutes after which the host status changes to **out of sync**.

9.3. SETTING THE PUPPET AGENT RUN INTERVAL ON A HOST

Set the interval when the Puppet agent runs and sends reports to Satellite.

Procedure

1. Connect to your host using SSH.
2. Add the Puppet agent run interval to **/etc/puppetlabs/puppet/puppet.conf**, for example **runinterval = 1h**.

9.4. SETTING THE GLOBAL OUT-OF-SYNC INTERVAL

Procedure

1. In the Satellite web UI, navigate to **Administer > Settings**.
2. On the **General** tab, edit **Out of sync interval** Set a duration, in minutes, after which hosts are considered to be out of sync.

You can also override this interval on host groups or individual hosts by adding the **outofsync_interval** parameter.

9.5. SETTING THE PUPPET OUT-OF-SYNC INTERVAL

Procedure

1. In the Satellite web UI, navigate to **Administer > Settings**, and click the **Config Management** tab.
2. In the **Puppet interval** field, set the value to the duration, in minutes, after which hosts reporting using Puppet are considered to be out of sync.

9.6. OVERRIDING OUT-OF-SYNC INTERVAL FOR A HOST GROUP

Procedure

1. In the Satellite web UI, navigate to **Configure > Host Groups**.
2. Select a host group.
3. On the **Parameters** tab, click **Add Parameter**.
4. In the **Name** field, enter **outofsync_interval**.
5. From the **Type** dropdown menu, select **integer**.
6. In the **Value** field, enter the new interval in minutes.
7. Click **Submit**.

9.7. OVERRIDING OUT-OF-SYNC INTERVAL FOR AN INDIVIDUAL HOST

Procedure

1. In the Satellite web UI, navigate to **Hosts > All Hosts**.
2. Click **Edit** for a selected host.
3. On the **Parameters** tab, click **Add Parameter**.
4. In the **Name** field, enter **outofsync_interval**.
5. From the **Type** dropdown menu, select **integer**.
6. In the **Value** field, enter the new interval in minutes.
7. Click **Submit**.