



CA Release Automation
Puppet Integration
Best Practice Guide

Author : Walter Guerrero
Version: 1.2
Filename: CA-RA-Puppet-Best-Practices-GuideV1.2
Date: 7/23/2015

Table of Contents

Copyright Notice	3
Introduction	4
What is Puppet.....	4
Pre-Requisites	4
Generic Modules	4
Module: ABC	5
Module: LinuxFiles	6
Puppet Role and Profile	7
Module: Profile	8
Module: role.....	8
RA Puppet Integration	9
Configuration Management.....	9
Release Automation Application	11
Adding Puppet Roles.....	11
Release Automation Dashboards.....	13
Sample RA Application	17
Create RA Application	17
Create RA Deployment	20
Best Practices	22
Connecting to Puppet Master.....	22
Defining the proper Puppet roles	22
References	24

Copyright Notice

Copyright © 2015 CA, Inc. All rights reserved. All marks used herein may belong to their respective companies. This document does not contain any warranties and is provided for informational purposes only. Any functionality descriptions may be unique to the customers depicted herein and actual product performance may vary.

Introduction

Release Automation 5.5.2 introduces the seamless integration with Puppet by PuppetLabs, a configuration management product. This integration provides all Release Automation users the ability of calling the desired Puppet classes during the Release Automation deployment phase.

What is Puppet

Puppet is an IT automation software manager that allows system administrators to programmatically provision, configure, and manage servers, storage and network devices, regardless of the location of these systems whether in a data center or in the cloud.

For the integration with Release Automation, you can Puppet Enterprise (PE) version 3.7 and above.

Pre-Requisites

A pre-requisite is for the [Puppet Enterprise](#) has been installed and configured in your local systems, once your Puppet Enterprise is up and running and you have installed and configured the necessary Puppet nodes, as well as creating the necessary node groups.

Generic Modules

To show you how to setup Puppet Enterprise to integrate with Release Automation, we need to perform the following steps.

- The systems to use should be Red Hat or CentOS. CentOS is the preferred version of Linux for the Puppet Enterprise installation
- The Puppet Enterprise agent has been installed and configured in these systems

After making sure that the Puppet Enterprise agent has been installed in the necessary Linux systems, we are going to create two modules¹ for verification purposes, you can use your modules, if so desired.

- Change directory to `/etc/puppetlabs/puppet/environments/production/modules`
 - Run the following command `mkdir -p abc/manifests`
 - Run the following command `mkdir -p linuxfiles/manifests`

Now, we are going to create the following “classes” for the different modules.

Puppet classes are named blocks of Puppet code, and they are stored in modules for later use; which are not applied until they are invoked by name.

¹ For additional information in how to create Puppet modules, please check the reference section.

Classes are used to configure large or medium-sized chunks of functionality, such as config files, packages, and services that would be needed to run an application.

Example of a class

```
# A class with no parameters
class base::linux {
  file { '/etc/passwd':
    owner => 'root',
    group => 'root',
    mode  => '0644',
  }
  file { '/etc/shadow':
    owner => 'root',
    group => 'root',
    mode  => '0440',
  }
}
```

Module: ABC

Change directory to the

'/etc/puppetlabs/puppet/environments/production/modules/abc/manifests' and create the files with the contents shown below.

Init.pp

```
class abc {
  notify {'creating files and directory for the abc comp in linux
nodes':}
}
```

Abcdir.pp

```
class abc::abcdir {
  # creating the dir first
  $abc_dir = '/opt/abc_demo'
  file {$abc_dir:
    path => $abc_dir,
    ensure => directory,
    mode => '0755',
    owner => 'root',
    group => 'root'
  }
}
```

Abcfiles.pp

```
class abc::abcfiles {
  # creating a file first
  $abc_file = '/opt/abc_demo/abcdemo_file.txt'
  file {$abc_file:
    path => $abc_file,
    ensure => file,
    content => 'This is a just a simple line added here... \n Another
line here \n',
  }
}
```

```

        mode => '0755',
        owner => 'root',
        group => 'root'
    }
}

```

Module: LinuxFiles

Change directory `/etc/puppetlabs/puppet/environments/production/modules/linuxfiles/manifests'` and create the files with the contents listed below.

Init.pp

```

class linuxfiles {
    notify {'creating files and directories for linux nodes!':}
}

```

Createdir.pp

```

class linuxfiles::createdir {
    # creating a directory first
    $demo_dir = '/opt/ra_demo'
    file {$demo_dir:
        path      => $demo_dir ,
        ensure => directory,
        mode      => '0755',
        owner    => 'root',
        group    => 'root'
    }
}

```

Createfile.pp

```

class linuxfiles::createfile {
    # creating a file first
    $demo_file = '/opt/ra_demo/rademo_file.txt'
    file {$demo_file:
        path      => $demo_file ,
        ensure => file,
        content => 'This is just a test to see what is going on
here...\nHere is a second line to be used...\n',
        mode      => '0755',
        owner    => 'root',
        group    => 'root'
    }
}

```

After creating the above modules, create the node group for the corresponding modules above, named `“abc”` and `“linuxfiles”` in the Puppet Enterprise console.

Create another node group that could be named `“RALinux”`, and make sure that nodes allowed to be part of this node group is based off the entry listed below.

Fact	Operator	Value	Node matches
<input type="text"/>	is	<input type="text"/>	-
osfamily	is	RedHat	2

Figure 1: RAlinux node group node rule

Add all the classes and sub-classes for the “abc” and “linuxfiles” modules under the “Classes” tab in the “RAlinux” node group as shown below.

Class: abc::abcfiles

Parameter

=

Class: abc

Parameter

=

Class: abc::abcdir

Parameter

=

Figure 2: Sample Classes added to RAlinux node group

After you have completed all the above steps, it is recommended that you execute the following command on each of the Linux nodes that you have setup: “*puppet agent -t*”.

This demonstrates that the Puppet modules listed in this guide can be created, added to a node group, and executed via the Puppet command listed above.

Puppet Role and Profile

Role and profile in a Puppet installation are just normal modules with no special features. What sets “role and profiles” apart is the how they will be used. Normal modules are publicly releasable bundles of code that take care of the configuration for a single technology. Whereas “role and profiles” modules are private, site-specific code that configure technology stacks (that is what a profile is) and complete configurations for categories of nodes (which is what roles are).

There is a need to define the necessary Puppet profiles (technology stacks), which reference the necessary modules that you have either created or have been created for you.

Afterwards, you will define the necessary roles (the complete configuration), which will include the necessary profiles to be used.

For the seamless integration with Release Automation to be successful, the following actions in the Puppet Enterprise installation need to be performed.

- In the `/etc/puppetlabs/puppet/environments/production/modules`
 - Create the following directories: `'mkdir -p profile/manifests'`
 - Create the following directories `'mkdir -p role/manifests'`

Please add the necessary entries as shown below to get familiar with the process to be followed for the seamless integration with Release Automation.

Module: Profile

Change directory to the

`/etc/puppetlabs/puppet/environments/production/modules/profile/manifests` and create the files with the contents shown below.

base.pp

```
#
# = Profile: base
#
# Manages base and associated software.
#
# Including the following as examples in how to defined
# default entries. For this example, these entries
# are commented out.
class profile::base {
  #include ::stdlib
  #include ::staging
  #include ::linuxfiles
  #include ::abc
}
```

profile_linuxfiles.pp

```
# this profile makes sure that the linux classes are called correctly
class profile::profile_linuxfiles {
  include linuxfiles
  include linuxfiles::createdir
  include linuxfiles::createfile
}
```

profile_abc.pp

```
# This profile module contains the abc classes specifically
class profile_abc {
  include abc
  include abc::abkdir
  include abc::abcfiles
}
```

Module: role

Change directory `/etc/puppetlabs/puppet/environments/production/modules/role/manifests` and create the files with the contents listed below.

init.pp

```
# role modules init class
class role {
  include profile::base
}
```

role_linuxfiles.pp

```
# this class module will inherit the role class
# and include the profile_linuxfile class
class role::role_linuxfiles inherits role {
  include profile::profile_linuxfiles
}
```

role_abc.pp

```
# this class will include the abc profile class
class role::role_abc inherits role {
  include profile::profile_abc
}
```

After you have completed all the above steps, it is recommended that you execute the following command on each of the Linux nodes that you have setup: “*puppet agent -t*” as a sanity check. This command will perform a configuration update on-demand in the foreground with verbose logging for the given Puppet node(s).

Now you need to install the Release Automation agent in the corresponding Puppet (PE) master. This is a very **important** step.

RA Puppet Integration

Once the Puppet Enterprise (PE) has been installed and configured with the additional modules detailed in the prior sections of this document. This section will cover how to setup Puppet (PE) connection and setup a Release Automation application to execute the necessary configuration for the different server types.

Configuration Management

To connect a Puppet Enterprise master to the given Release Automation installation, you will need to connect to the Release Automation Release Operations Center (ROC), and go to the Administration→Configuration Management dialog.

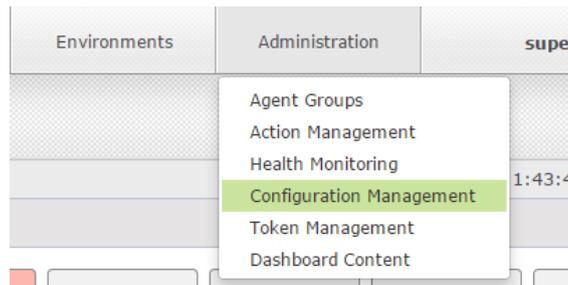


Figure 3: Accessing Configuration Management

Select the “Puppet” option in the Configuration Management dialog.

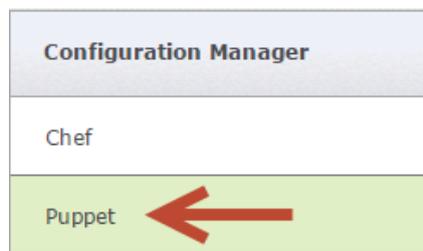


Figure 4: The Puppet option in the Configuration Management dialog

Now, select “+Add New Server” link to bring up the Add Puppet Master Server dialog.



Figure5: The typical entries for a Puppet Enterprise

The entries that you see in the above image are the default values, where the Puppet Enterprise gets installed by default by the Puppet Enterprise installer.

The default values are:

- Execution Location
 - /opt/puppet/bin
- Configuration Location
 - /etc/puppetlabs/puppet

Please click the “Validate” button to verify that Release Automation can communicate with the Puppet master.

Puppet Master Server	Server Status	Execution Location	Configuration Location	
guewa01-U150000.ca.com	Connected	/opt/puppet/bin	/etc/puppetlabs/puppet/	Edit Delete

Figure 6: Typical Puppet master configuration in RA

Release Automation Application

Now that you have connected the Puppet (PE) master with Release Automation, we need to perform the following actions.

- Create an RA application
- Create at least one server type
- Create the necessary architectures and map the server types to it
- Include the necessary components (actions and flows)
- Create the necessary environments
- Create and publish the necessary processes

After creating the logic of the RA application, we need to create a deployment release, so the following steps are necessary.

- Create a template category
- Create a deployment template and add the necessary steps
- Create a deployment
- Create a project
- Create a deployment plan

Adding Puppet Roles

Once you have executed the newly created application successfully, we will need to select the correct Puppet role to use with the Release Automation application.

Select the Environments → Parameter Configuration menu option from the Release Automation ROC.

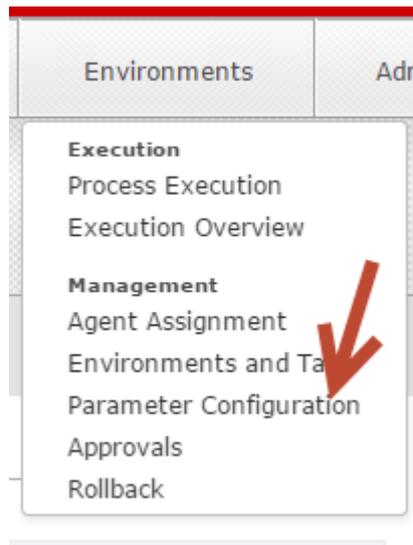


Figure 7: Select Parameter Configuration

Now we need to select the desired environment where the Puppet (PE) role will utilized.

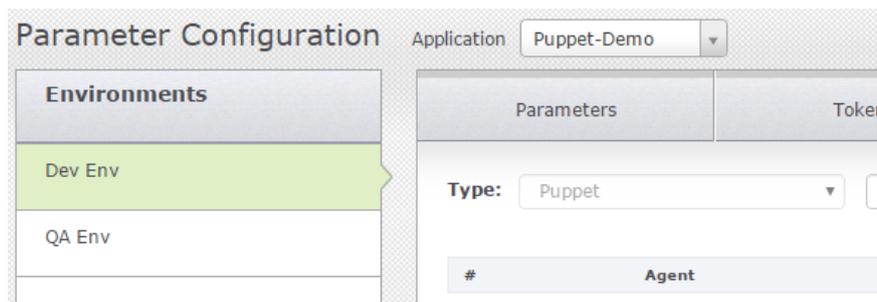


Figure 8: Parameter Configuration Environments

The default view will be the type of configuration that will be taking place, in this case this is defaulting to “Puppet”.

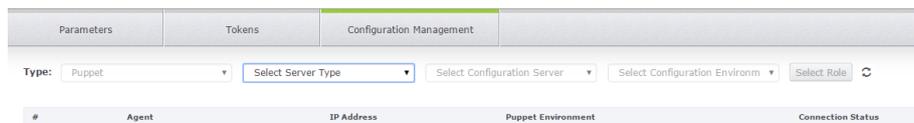


Figure 9: Typical view of the puppet configuration parameter

Now you select the RA server type association, the puppet master, and the puppet environment to utilize.

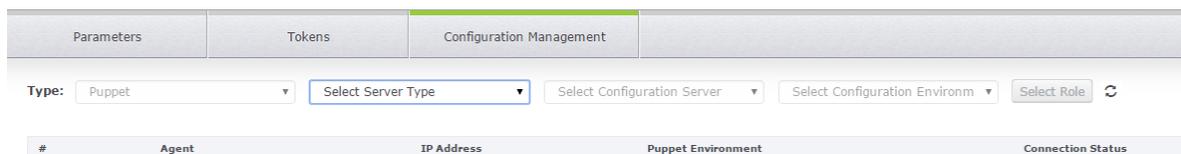


Figure 10: Selecting a server type and puppet environment

Click on the “Select” button and select the desired puppet role that will be used with this Release Automation application.

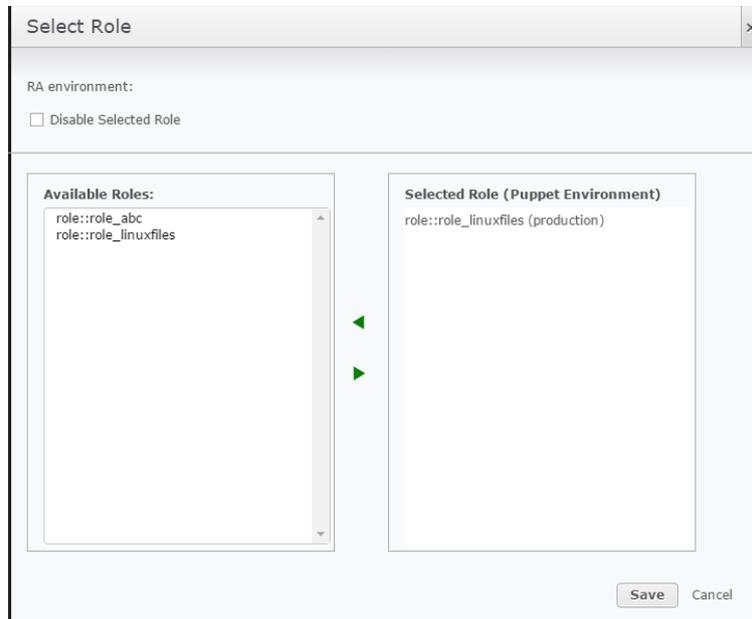


Figure 11: Selecting a Puppet role

Now we can launch a new deployment based off the deployment template that you have already created. As the deployment is executing, you will see in the “Pre-Deployment” tab, how the puppet role that was selected is being used to execute on the designated puppet nodes.

Pre Deployment		Deployment	Post Deployment	Summary
Environment Configuration		Artifact Package		
DESCRIPTION	RESULTS			
Linux	Running			
Puppet Environment	production			
Puppet Role	role::role_linuxfiles			
Puppet Node List (2)	* guewa01-U150000.ca.com * guewa01-U150715			
Configuration Result	Running			

Figure 12: Typical deployment with Puppet integration

After setting up the integration between Release Automation and Puppet (PE), you can setup the desired RA dashboard to show any type of configuration management baseline drift.

Release Automation Dashboards

Release Automation started providing dashboard capabilities starting with release 5.5.0, now that the Puppet seamless integration has been completed, we are going to go over the additional dashboard that are presently available that can provide you with a graphical representation of the different configuration management as executed by Release Automation.

When logging into the Release Operations Center UI you are presented with the default Dashboard as shown below:

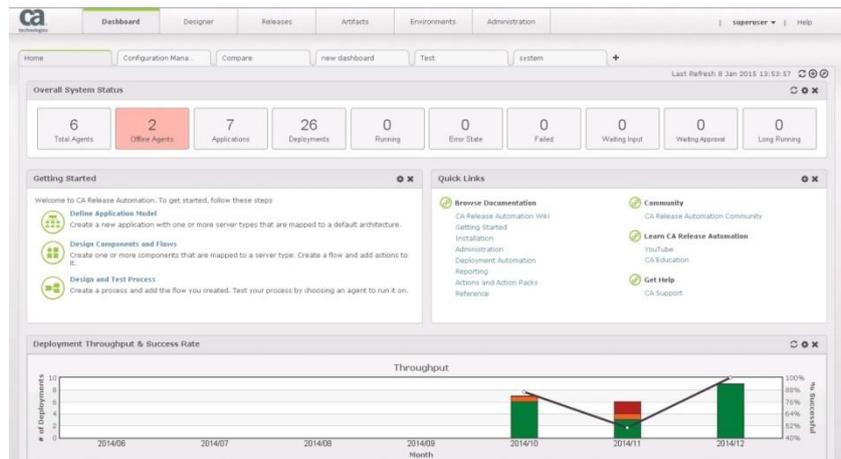


Figure 13: Typical RA dashboard

You can change this default page by deleting the existing widgets and adding your own or you can create a new Dashboard by selecting the “+”, provide a name, and select the layout/format.

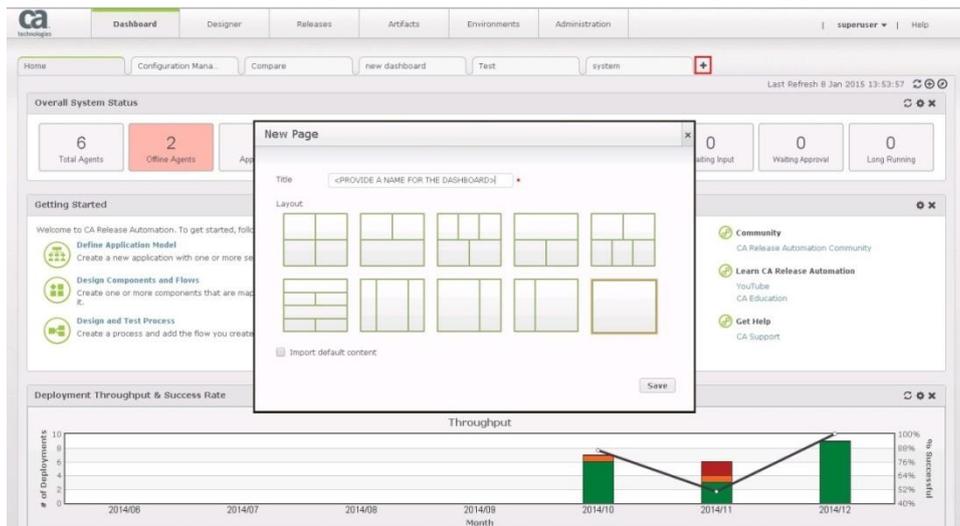


Figure 14: Adding a new dashboard

Then select your new Dashboard and proceed to add Widgets and/or Reports. You can select the “+” in the upper right hand corner or in the center of the page. This will open a new dialog with two tabs to select from, one for Widgets and the other for Reports. There are two Widgets and three Reports that are helpful for reviewing Configuration Management information.

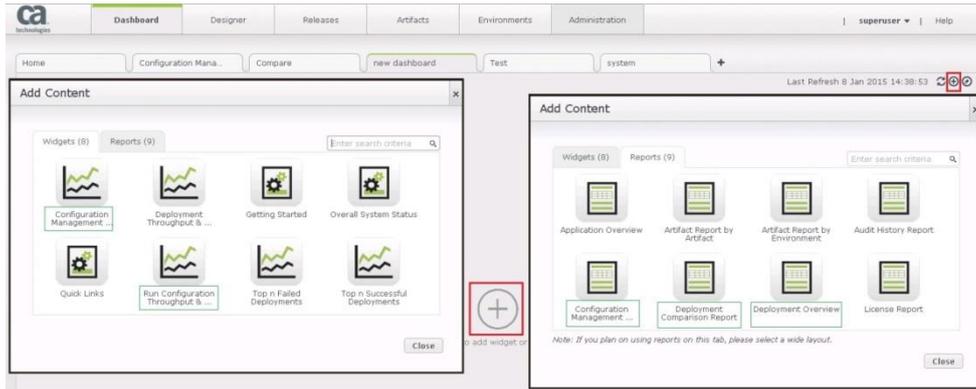


Figure 15: Selecting dashboard content

The Run Configuration Throughput & Success Rate shows the status of the configuration run during the deployment, not the deployment itself. For the data to populate, you are required to specify the configuration manager type, and define the parameters in the same manner as mentioned above.



Figure 16: Sample Configuration throughput

The Configuration Management Deployment Overview report provides information regarding the configuration run during the deployment. As with content configuration mentioned above, you have the same configuration functionality, i.e. the display of one or more Applications, one or more Projects, and one or more environments, for the timeframe specified.

Deployment	Date	Application Name	Drifted	Failed Nodes	Run Status
Puppet-Simple-009	2015-07-10 13:...	Puppet-Demo	no	["guewa01-U122106","guewa01-U150000.ca.com"]	fail
Puppet-Simple-010	2015-07-10 13:...	Puppet-Demo	no	[]	success
Puppet-Simple-011	2015-07-13 12:...	Puppet-Demo	no	[]	success
Puppet-Simple-012	2015-07-13 12:...	Puppet-Demo	no	["guewa01-U150000.ca.com"]	fail
Puppet-Simple-013	2015-07-13 12:...	Puppet-Demo	no	[]	success

Figure 17: Sample configuration management deployment

The Deployment Comparison Report shows a deployment in its different environments for comparison or to identify discrepancies during the deployment. It includes comparative details for parameters and artifacts that are listed through Environment Configuration. The comparisons include, for example, a side-by-side listing of baselines between two deployments. For each server type, the Environment Configuration section compares the following items:

- The RA Environment, the Puppet environment, the Puppet master, Puppet role, Puppet node list, and Puppet role configure result.

Deployment Comparison Report		
+	Deployment Details	4 differences
+	Release Artifacts	0 differences
+	Environment Artifacts	0 differences
+	Environment Tokens	0 differences
+	Environment Parameters	0 differences
+	Release Parameters	0 differences
+	User Input Parameters	0 differences
+	Deployment Plan Properties	0 differences
-	Infrastructure Configuration	0 differences
- Server Type: Linux		
Configuration Type	Puppet	Puppet
RA Environment	Dev Env	Dev Env
Puppet Master	guewa01-U150000.ca.com	guewa01-U150000.ca.com
Puppet Environment	production	production
Puppet Role	role::role_linuxfiles	role::role_linuxfiles
Puppet Node List	guewa01-U150715 guewa01-U150000.ca.com	guewa01-U150000.ca.com guewa01-U150715
Role Configure Result	Success	Success

Figure 18: Sample comparison report

The Deployments Overview Reports provides an overview of current and historical deployments. By default, the report does not show all deployments and you are required to set the Start and End Time filters. As with the Configuration Management Deployment Overview report mentioned above in step 6, Deployment name is a link that will display the deployment information at the time of deployment. However, unlike that same report, where you can select the content configuration,

here you will need to configure the time stamp **filter** as shown below. Once you retrieve the data, you can **Save** the report for review at a later date.

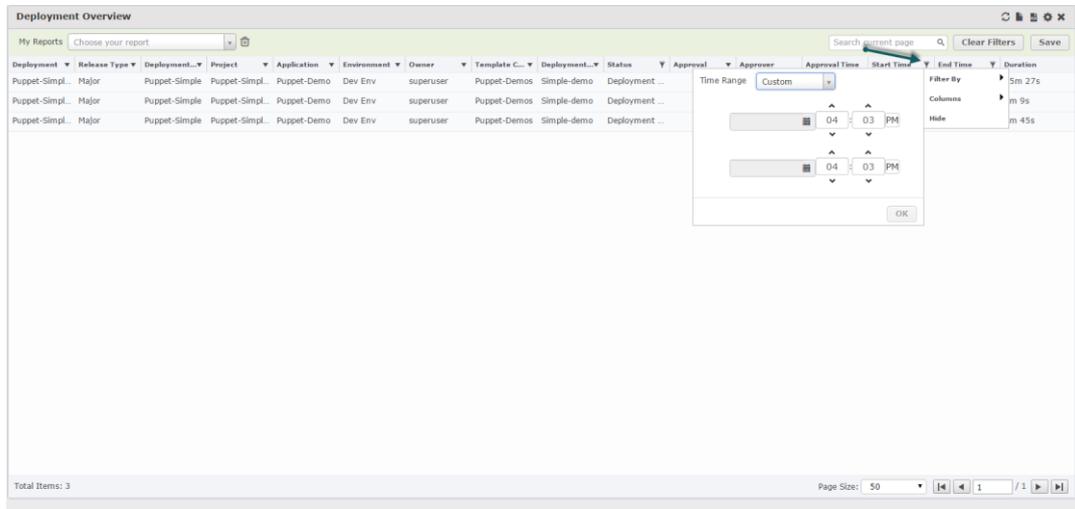


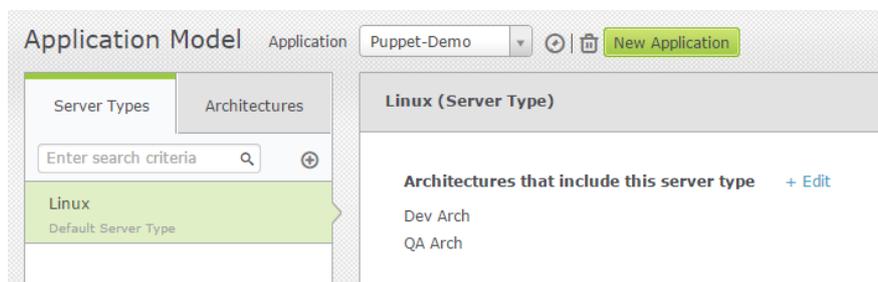
Figure 19: Sample deployment overview

Sample RA Application

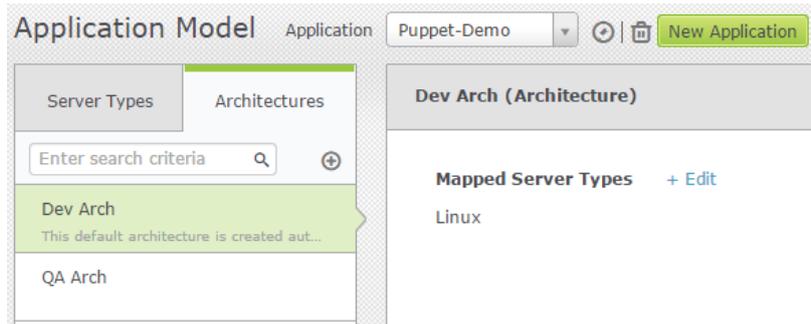
To review that we can accomplish all the steps to integrate the Puppet configuration management system with Release Automation, we are going to create a sample application. Prior to creating this application, we are under the assumption that the connection with the Puppet master has been setup, validated, and it is active.

Create RA Application

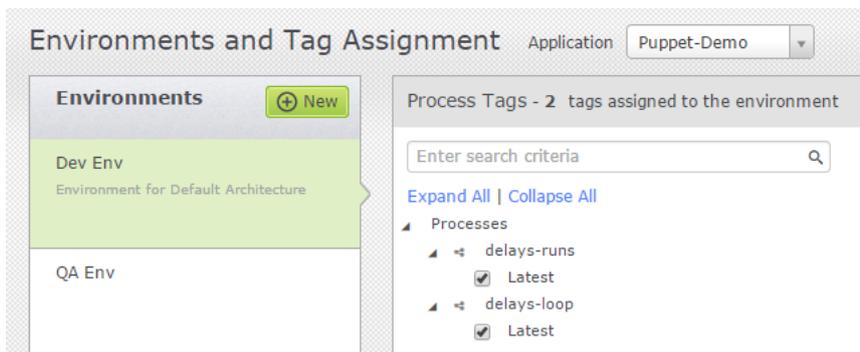
Let's start by creating the sample RA application and call it "Puppet-Demo", let's add the following server type: Linux as shown below.



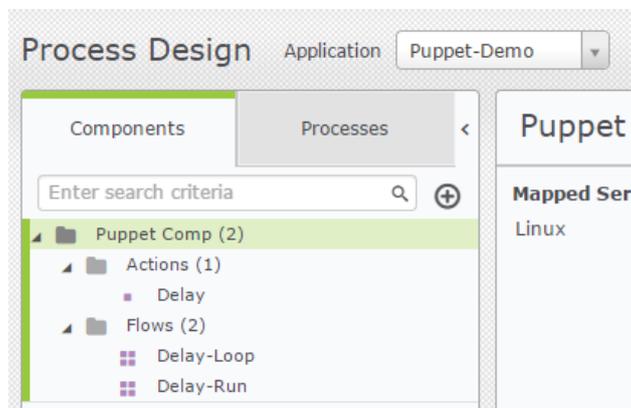
After that, let's add the following architectures: Dev Arch and QA Arch, as shown below.



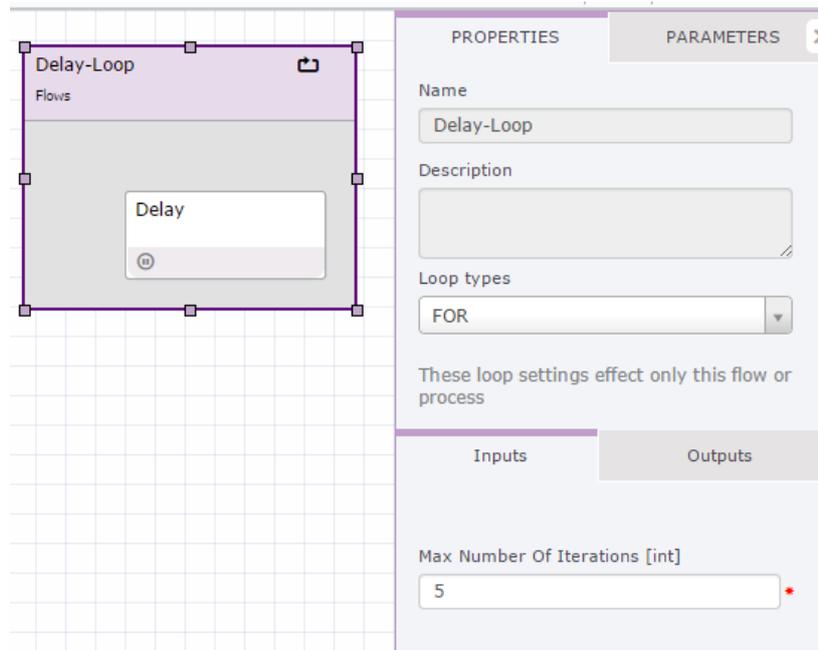
Now let's add the environments as shown below, you might have to rename the default environment to "Dev Env"



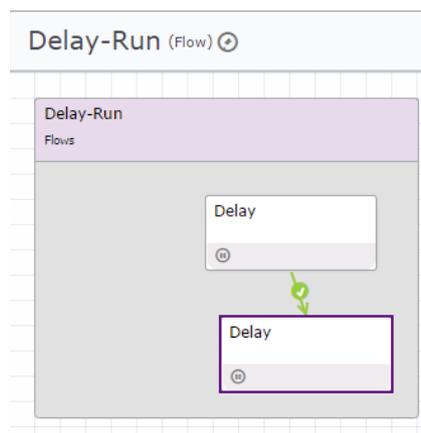
Move to the "Process Design", and add the "Delay" action and the "Delay-Loop" and "Delay-Run" flows as shown below.



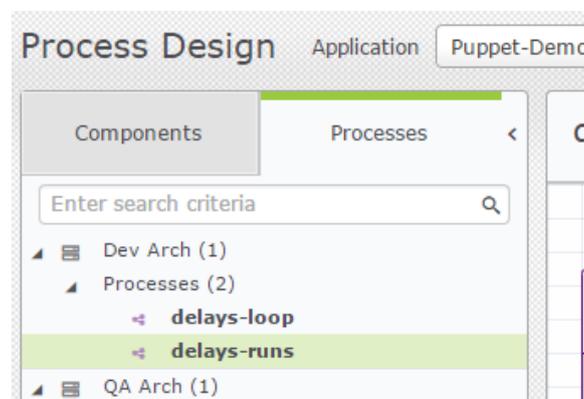
For the "Delay-Loop" flow, add a FOR type loop, and iterate thru it 5 times as shown below.



For the “Delay-Run” flow, just link two Delay actions as shown below.

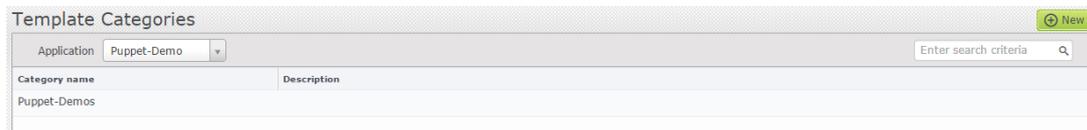


Now, you need to switch to the “Process” tab and create two processes as shown below. These are the same names as the flows that you created previously. Please publish these processes to the “Dev Env”.

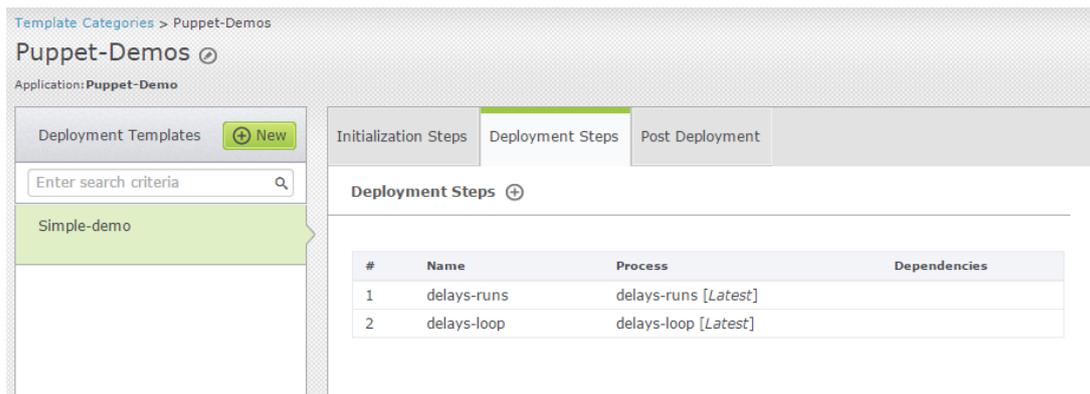


Create RA Deployment

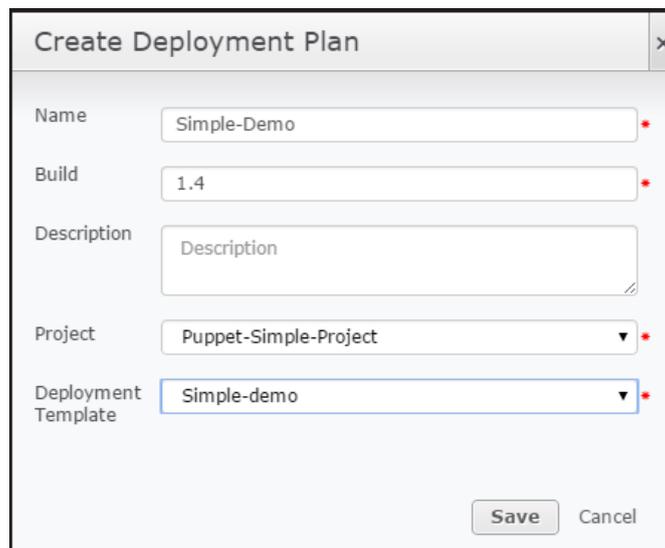
After the RA application processes have been published, we need to create a deployment release, so we start by creating a release template as shown below.



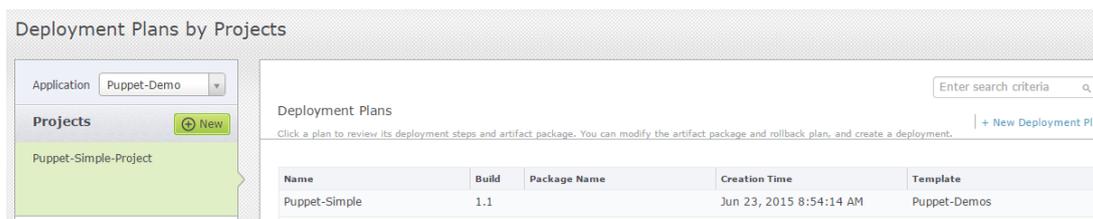
Create a deployment template with the steps shown below.



Now create a deployment plan by clicking on the "Create Deployment Plan" button.



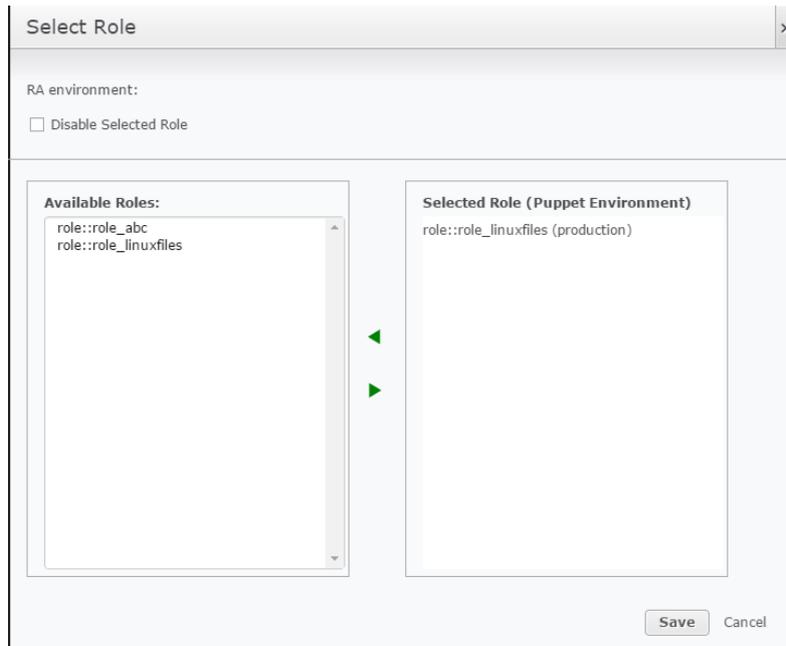
Create a deployment project plan as shown below.



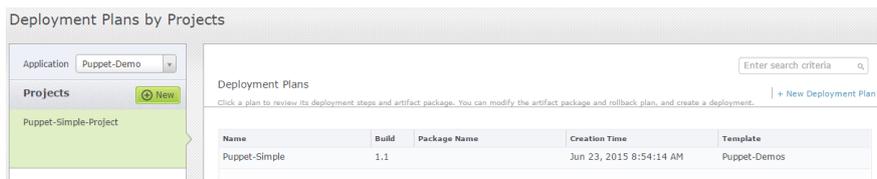
Now we need to assign a Puppet role to the "Dev Env" as shown below.



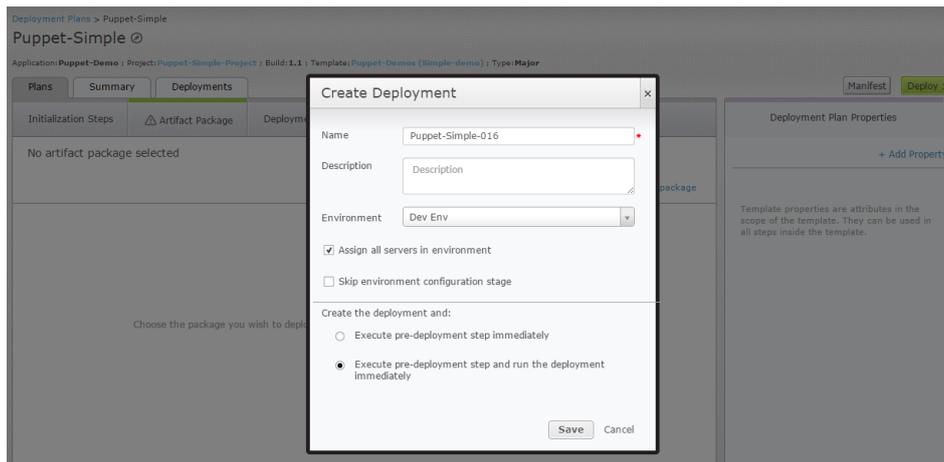
Add the desired Puppet role by clicking on the “Select Role” button, and select the role as shown below, and click the “Save” button.



Let’s go back to the “Releases→Deployment Plans by Projects”, and select the “Puppet-Simple” deployment plan.



Now you need create a deployment as shown below.



The end result of this deployment will be that in the “Pre Deployment” tab, you will see under the Environment Configuration the Puppet (PE) configuration running, and once that has completed running; you can use the dashboard reports detailed in the prior section to see if you have had any type configuration drift.

Pre Deployment		Deployment	Summary
Environment Configuration		Artifact Package	
DESCRIPTION	RESULTS		
Linux	Success		
Puppet Environment	production		
Puppet Role	role::role_linuxfiles		
Puppet Node List (2)	<input checked="" type="checkbox"/> guewa01-U150000.ca.com <input checked="" type="checkbox"/> guewa01-U150715		
Configuration Result	Success		

Best Practices

The following best practices are presented to help in running Puppet configurations via Release Automation.

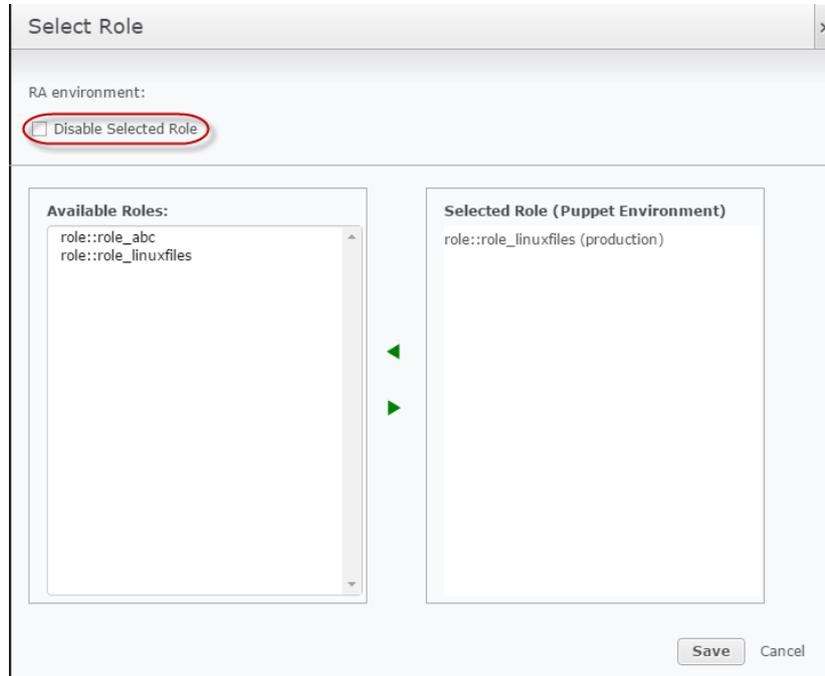
Connecting to Puppet Master

To make sure that you have a valid connection to a puppet master; please make sure that the Release Automation agent is installed in the Puppet master and the Puppet agent. At the same time, you need to make sure that the Puppet installation is the Puppet Enterprise 3.7 or greater and install the puppet agent from the puppet master as defined by Puppet Labs.

Defining the proper Puppet roles

A Puppet role can contain as many Puppet profiles as required, so it becomes of the most importance that you create the proper Puppet roles, since only one role can be associated with the puppet nodes.

The Configuration Run stage during the active deployment can be skipped, by returning to the Environment Configuration steps as shown above and selecting the **Disable Selected Role**. Then select **Save** to save the current Environment Configuration. At this point return to the Deployment and upon selecting **Run**, the Configuration Run will be skipped.



References

Release Automation

<https://wiki.ca.com/display/RA55/How+to+Set+Up+an+Application>

<https://wiki.ca.com/display/RA55/How+to+Create+a+Deployment+Template>

<https://wiki.ca.com/display/RA55/How+to+Create+a+Deployment>

Puppet Labs

http://docs.puppetlabs.com/pe/latest/puppet_overview.html

http://docs.puppetlabs.com/pe/latest/puppet_modules_manifests.html

http://docs.puppetlabs.com/puppet/3.8/reference/lang_resources.html

http://docs.puppetlabs.com/puppet/3.8/reference/lang_relationships.html

http://docs.puppetlabs.com/pe/latest/puppet_assign_configurations.html#assigning-configuration-data-with-role-and-profile-modules

http://docs.puppetlabs.com/puppet/3.8/reference/lang_classes.html