Puppet on the AWS Cloud

Quick Start Reference Deployment

AWS Quick Start Reference Team

March 2016

This guide is also available in HTML format at
http://docs.aws.amazon.com/quickstart/latest/puppet/.

Contents

About This Guide

This Quick Start reference deployment guide discusses the steps for deploying and testing a Puppet master and Puppet agents on the Amazon Web Services (AWS) cloud. It also provides links for viewing and launching AWS CloudFormation templates that automate the deployment, and a walkthrough on how you can configure Amazon Elastic Compute Cloud (Amazon EC2) instances that act as Puppet agents.

The guide is for IT infrastructure architects, administrators, and DevOps professionals who are planning to implement or extend their Puppet workloads on the AWS cloud.

AWS OpsWorks option This Quick Start is for customers who want to run and manage their own Puppet master infrastructure. However, we recommend that you also take a look at AWS OpsWorks, which is a configuration management service provided by AWS, to determine if it's more suitable for your needs. AWS OpsWorks helps you configure and operate applications of all types and sizes. You can define the application's architecture and the specification of each component, including package installation, software configuration, and resources such as storage. For more information, see the AWS OpsWorks User Guide.

Quick Starts are automated reference deployments for key workloads on the AWS cloud. Each Quick Start launches, configures, and runs the AWS compute, network, storage, and other services required to deploy a specific workload on AWS, using AWS best practices for and security.

Overview

Puppet on AWS

Puppet is a declarative, model-based configuration management solution from Puppet Labs that lets you define the state of your IT infrastructure, and automatically enforces that desired state on your systems. Every step of your software delivery process, from provisioning instances to orchestration and reporting, including production release of software and updates, can be automated. Configuration management tools like Puppet can help you get more done in less time, and can help ensure consistency and reliability across the state of your infrastructure. Puppet uses a client/server model where agent nodes get configuration profiles from the Puppet master, which is a server that controls the configuration information. For more information, see the architecture overview on the Puppet Labs website.

Using this Quick Start, you can launch a Puppet master with a single click to get instant access to the following features:

- Puppet enables you to define configurations that are idempotent, meaning they can be run multiple times without any risks. Once you've developed your configurations, your agents can apply the configuration on a regular interval (30 minutes by default), which will keep your systems in their desired state. If your system state drifts out of the desired configuration, the Puppet agent will re-apply your configuration.
- Puppet gives you cross-platform support for multiple agent types running on a variety of operating systems. For example, if you're already using Puppet for configuration management of Linux systems, you can use Puppet to manage your Microsoft Windows servers as well. This includes EC2 instances running Linux or Windows, and even physical machines you have running in your own data center.
- In addition to using native Puppet modules to configure your systems, you can use code from the Puppet Forge to extend the capabilities of Puppet. Puppet Forge is a repository of modules contributed by the Puppet community. It provides you with reusable code that can automate tasks such as setting up various databases, web servers, and mail servers.

This Quick Start is for users who are looking to move to AWS, or are already running their systems on AWS, and also want to deploy and manage their own Puppet master infrastructure. The goal of this guide is to help you get started with Puppet on AWS, even if you have absolutely no experience with Puppet.

This Quick Start automates the launch of the Puppet master, performs the initial server setup, and creates both Linux and Windows-based Puppet agents within an Amazon Virtual Private Cloud (Amazon VPC). You can follow the walkthrough included in this guide to learn how to apply configurations to Puppet agents and to install a basic web server. The walkthrough will show you the benefits of automating software installation and configuration, and will help you understand how to ensure that your system settings are repeatable, consistent, and always in their desired state.

Quick Links

If you have an AWS account and you're already familiar with AWS and Puppet, you can use the **Launch Quick Start** button to build the architecture shown in Figure 1. The deployment takes approximately 20 minutes. If you're new to AWS or Puppet, please review the implementation details and follow the step-by-step instructions provided later in this guide to launch the Quick Start.

Launch Quick Start

If you want to take a look under the covers, you can choose **View template** to see the AWS CloudFormation template that automates this deployment. The default configuration deploys three servers that use the t2.medium instance type by default, but you can customize the template if you'd like.

View template

Cost and Licenses

You are responsible for the cost of the AWS services used while running this Quick Start reference deployment. There is no additional cost for using the Quick Start. As of the date of publication, the cost for using the Quick Start with default settings is approximately $0.18 an hour. Prices are subject to change. See the pricing pages for each AWS service you will be using or the AWS Simple Monthly Calculator for full details.

This Quick Start deploys Open Source Puppet version 3.8.6 by default. Open Source Puppet is available to download and use under the Apache 2.0 license. You can upgrade to Puppet Enterprise by registering with Puppet Labs and downloading a free trial to manage 10 nodes.

Architecture

Deploying this Quick Start with the **default parameters** builds the following environment in the AWS cloud.

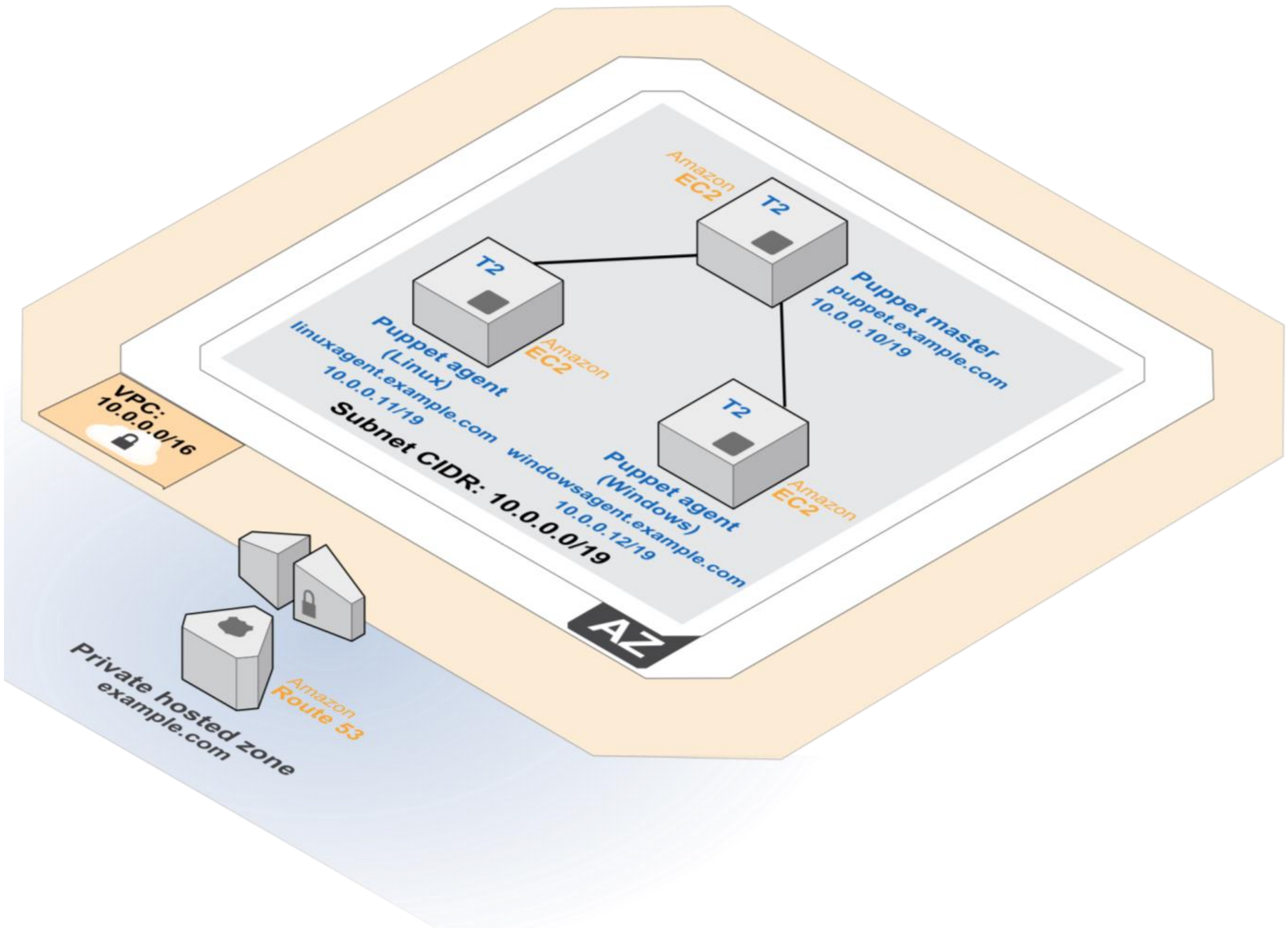


Figure 1: Quick Start Puppet Architecture on AWS

This Quick Start deploys the resources shown in Figure 1 and uses them as follows:

- An Amazon VPC is created in the region you choose when you launch the stack. A single, public VPC subnet is created in the first Availability Zone.
- One Puppet master is deployed into the VPC subnet. During instance launch, the Puppet master is bootstrapped to automatically install all required software along with Puppet modules and manifests that can be used to configure the Puppet agents.
- One Ubuntu Server is deployed into the VPC subnet. You can then follow the walkthrough in this guide to apply a web server configuration that will install and configure the Apache web server and PHP.
- One Windows Server 2012 R2 server is deployed into the VPC subnet. You can follow the walkthrough in this guide to apply a web server configuration that will install and configure the Internet Information Services (IIS) web server and ASP.NET.

Implementation Details

This section discusses the implementation of this Quick Start and explains the considerations for installing and configuring Puppet on AWS. Note that some steps are manual and others are automated for you by this Quick Start.

AWS Services

The core AWS components used by this Quick Start include the following AWS services. (If you are new to AWS, see the Getting Started section of the AWS documentation.)

- Amazon VPC – The Amazon Virtual Private Cloud (Amazon VPC) service lets you provision a private, isolated section of the AWS cloud where you can launch AWS services and other resources in a virtual network that you define. You have complete control over your virtual networking environment, including selection of your own IP address range, creation of subnets, and configuration of route tables and network gateways.
- Amazon EC2 – The Amazon Elastic Compute Cloud (Amazon EC2) service enables you to launch virtual machine instances with a variety of operating systems. You can choose from existing Amazon Machine Images (AMIs) or import your own virtual machine images.
- Amazon Route 53 – Amazon Route 53 is a highly available and scalable cloud Domain Name System (DNS) web service. It is designed to give developers and businesses an extremely reliable and cost-effective way to route end users to Internet or internal applications by translating host names to IP addresses.

Puppet Master Installation

This Quick Start deploys the Puppet master on an EC2 instance that is running Ubuntu 14.04. The installation is automated with a user data script that executes when the instance is launched via AWS CloudFormation. The Open Source version of Puppet is installed using a package called *puppetmaster-passenger*, which is provided by Puppet Labs. This package deploys the Puppet master, including a production-ready web server implementation of Passenger with Apache. For more information about Passenger, see the documentation on the Puppet Labs website.

In addition to installing the Puppet master, this Quick Start downloads preconfigured Puppet modules from Amazon Simple Storage Service (Amazon S3), which will enable you to apply a web server configuration to both the Windows and Linux nodes.

Certificates and DNS Names

The Puppet master acts as a certificate authority (CA), and SSL certificates are used to authenticate communications between the master and agent nodes. Since the Puppet master is a CA, it will generate its own certificates, which will be used to sign agent certificate requests.

Because this Quick Start pre-provisions record sets for each EC2 instance in Amazon Route 53, the Puppet master will use the host name *puppet.example.com* by default. During the automated setup of Puppet, the master's CA certificates will be generated using this host name. This ensures that clients that connect to the master using its predetermined host name will see the correct host name on the certificate. Using the default host name eliminates the need to regenerate the certificates after a typical installation to include the appropriate name.

Puppet agents need to be configured to connect to your Puppet master, and the Quick Start automates that work. If you want to use different host names, you can simply download a copy of the templates, modify them to use your desired host names, and then launch the stack to automatically configure your master and agents. Keep in mind that the Quick Start downloads configuration files, modules, and manifests from Amazon S3 that include these names, so you'll also want to download and modify those if you want to customize your deployment.

The first time the Puppet agent runs on a node, it will send a certificate signing request to the master. Typically, this is not done automatically, and you must sign the agent certificate on the master server before you can start controlling the node.

In this Quick Start, certificate signing requests from the Linux and Windows agents are whitelisted by using the autosign.conf configuration file on the Puppet master. This file includes the names *linuxagent.example.com* and *windowsagent.example.com*. As with DNS name resolution, the Quick Start provisions record sets for these names for you in an Amazon Route 53 private hosted zone, and configures the agents to use these host names within the operating system. Using the autosigning configuration file with this Quick Start enables you to get up and running quickly. However, for production environments you'll likely want to manually sign agent requests, or use Puppet's policy-based interface for autosigning certificates.

Puppet Agent Installation

The Linux agent deployed by this Quick Start also runs Ubuntu 14.04, like the master. The installation of the agent takes place after the master has been deployed. The Quick Start runs a simple user data script when it launches the agent via AWS CloudFormation. This script installs the agent and configures it to point to the master at *puppet.example.com*, and the server automatically requests and signs the agent certificate.

The Windows agent is deployed on an instance running Windows Server 2012 R2. As with Linux, the Quick Start runs a simple user data script to install and configure the agent at launch, after the master has already been deployed. In addition, the Quick Start automatically downloads and installs the *puppetlabs-powershell* and *puppetlabs-windowsfeature* modules from the Puppet Forge. These modules are used within a module manifest that installs the IIS web server with all required components and support for ASP.NET websites.

Managing AWS Resources with Puppet

You can use the AWS module from Puppet Labs to provision, configure, and manage AWS resources in a consistent and repeatable manner. You can use this module to audit AWS resources, launch Auto Scaling groups in the Amazon VPC, perform unit testing, and more. The module supports the following AWS services:

- Amazon EC2
- Amazon VPC
- Elastic Load Balancing
- Auto Scaling
- Security groups
- Amazon Route 53 DNS

To learn more, see Provision, Configure & Manage AWS Resources with Puppet Enterprise on the Puppet Labs website.

Deployment and Configuration Steps

The AWS CloudFormation template provided with this Quick Start bootstraps the AWS infrastructure and automates the deployment of a Puppet master and Puppet agents on the AWS cloud from scratch. Follow the step-by-step instructions in this section to set up your AWS account, customize the template, and deploy the software into your account.

What We'll Cover

The procedure for deploying Puppet on AWS consists of the following steps. For detailed instructions, follow the links for each step.

Prerequisites

- Set up and enable name resolution via DNS.
- Make sure you can use Secure Shell (SSH) or Remote Desktop Protocol (RDP) for remote connections.

Step 1. Prepare an AWS account

- Sign up for an AWS account, if you don't already have one.
- Choose the region where you want to deploy the stack on AWS.
- Create a key pair in the region.
- Review account limits for Amazon EC2 instances and request a limit increase, if needed.

Step 2. Launch the stack

- Launch the AWS CloudFormation template into your AWS account.
- Enter a value for the **KeyPairName** parameter.
- Review the other template parameters, and customize their values if necessary.

Step 3. Configure Puppet agents

- Review the module manifests for the Linux and Windows agents.
- Connect to your Puppet agents via SSH or RDP.
- Apply the configurations to the Puppet agents.

Prerequisites

To enable communication between the Puppet master and Puppet agents, you must set up and enable name resolution via DNS. Agents reach the Puppet master by using a fully qualified DNS name such as *puppet.example.com*.

To provide name resolution within the Amazon VPC created by this Quick Start, the AWS CloudFormation template creates an Amazon Route 53 private hosted zone and provisions record sets for each EC2 instance based on the IP addresses provided through the template parameters at launch.

Using Amazon Route 53 is not a requirement. You can utilize your own DNS server infrastructure and manually create records and configure your instances. If you decide to use your own DNS server, make sure that your EC2 instances will resolve names against your own DNS server infrastructure, and create host (A) records that correspond to each EC2 instance IP address.

In addition to name resolution, a small number of network ports must be open to allow communication between the agents and the Puppet master. The Puppet master must be reachable by agents via TCP port 8140. For this Quick Start, the Puppet master is associated with an EC2 security group that permits inbound access to TCP port 8140 from any address within the VPC CIDR range.

To manage your agents, you must be able to connect remotely via Secure Shell (SSH) or Remote Desktop Protocol (RDP). This Quick Start creates and associates EC2 security groups for remote agent access. The inbound rules include access to TCP port 22 for SSH, and TCP port 3389 for RDP. Additionally, an inbound rule for TCP port 80 is permitted by the CIDR address you define for remote access. This will allow you to verify that your web servers are functional after applying your Puppet configurations on the agents.

Step 1. Prepare an AWS Account

1. If you don't already have an AWS account, create one at http://aws.amazon.com by following the on-screen instructions. Part of the sign-up process involves receiving a phone call and entering a PIN using the phone keypad.
2. Use the region selector in the navigation bar to choose the Amazon EC2 region where you want to deploy Puppet on AWS.

   Amazon EC2 locations are composed of *Regions* and *Availability Zones*. Regions are dispersed and located in separate geographic areas.

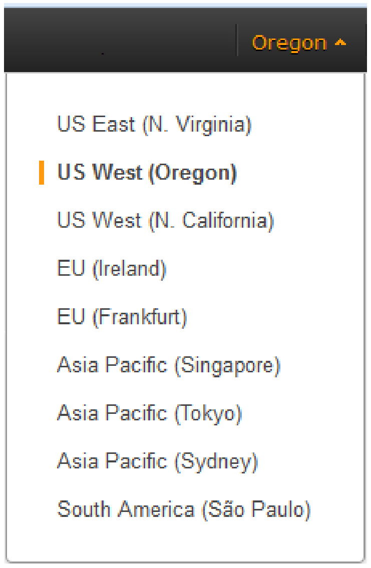


Figure 2: Choosing an Amazon EC2 Region

Tip Consider choosing a region closest to your data center or corporate network to reduce network latency between systems running on AWS and the systems and users on your corporate network.

3. Create a key pair in your preferred region. To do this, in the navigation pane of the Amazon EC2 console, choose **Key Pairs**, **Create Key Pair**, type a name, and then choose **Create**.

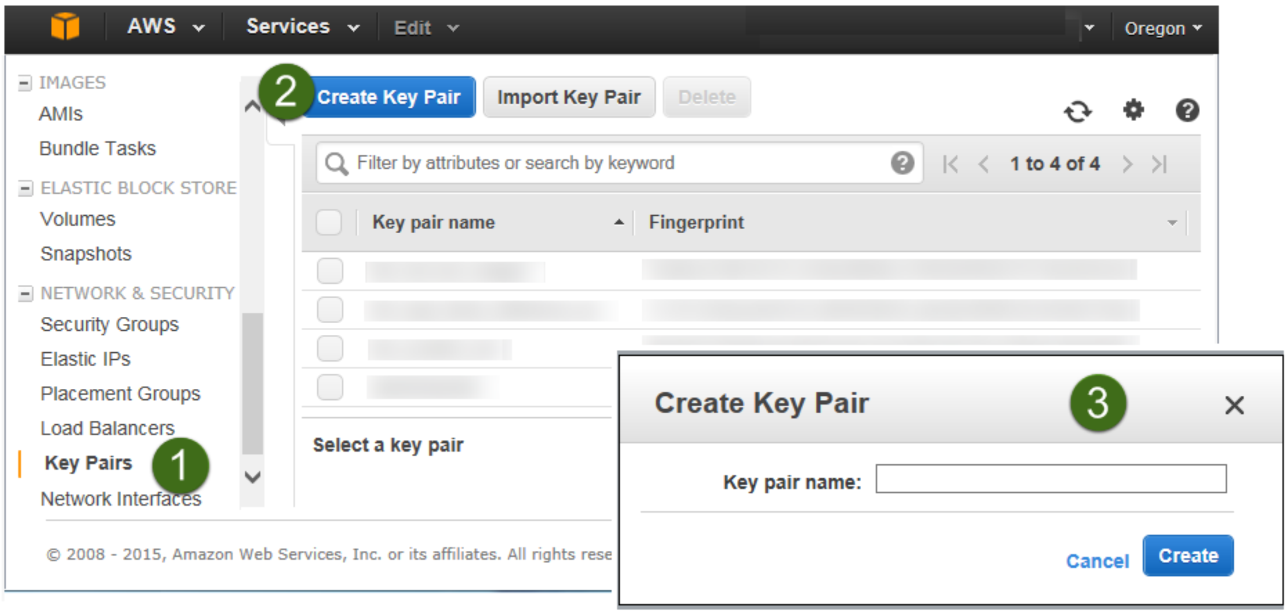
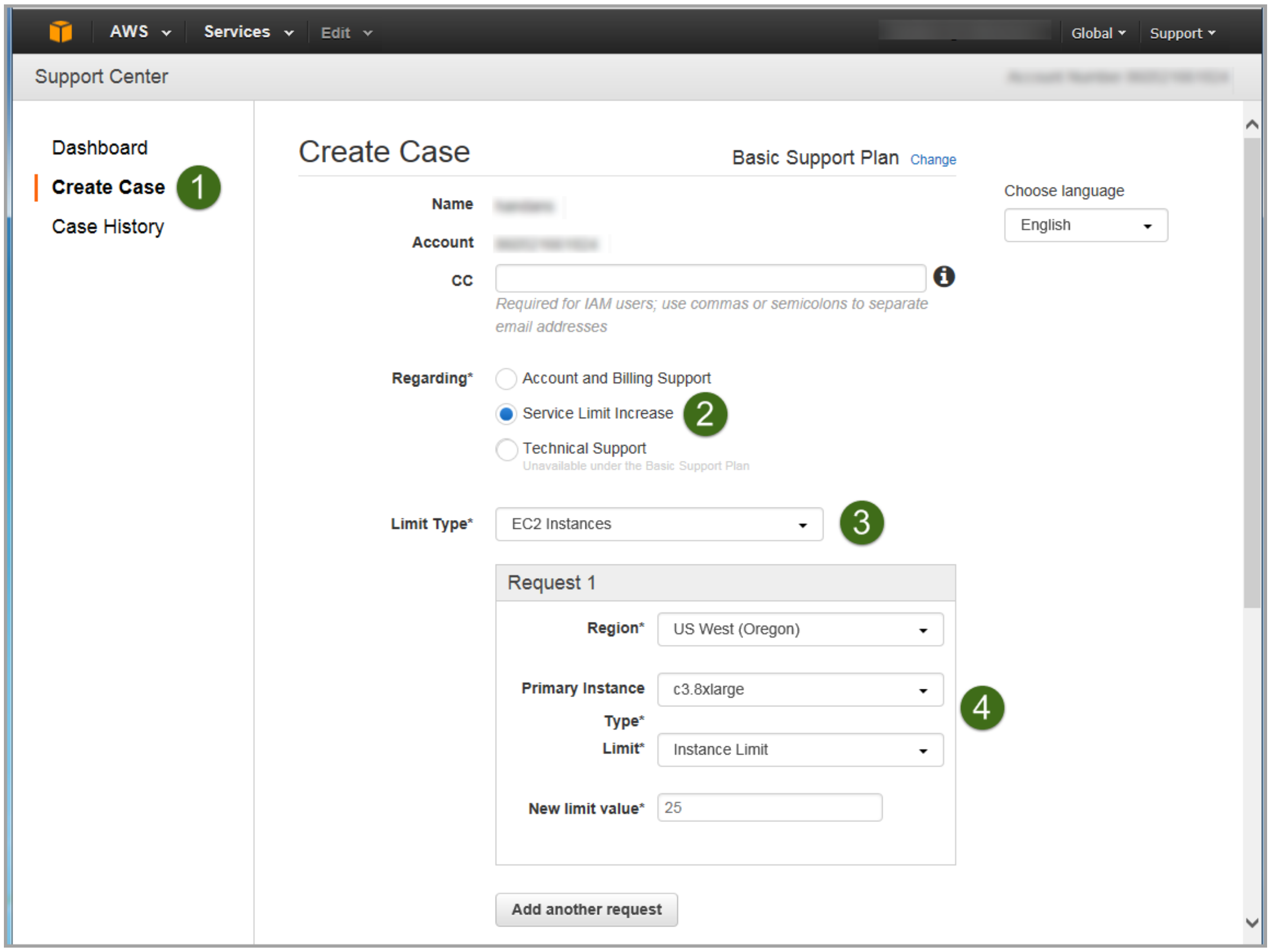


Figure 3: Creating a Key Pair

Amazon EC2 uses public-key cryptography to encrypt and decrypt login information. To be able to log in to your instances, you must create a key pair. With Windows instances, we use the key pair to obtain the administrator password via the Amazon EC2 console and then log in using Remote Desktop Protocol (RDP) as explained in the step-by-step instructions in the *Amazon Elastic Compute Cloud User Guide*. On Linux, we use the key pair to authenticate SSH login.

4. If necessary, request a service limit increase for the Amazon EC2 **t2.medium** instance type. To do this, in the AWS Support Center, choose **Create Case**, **Service Limit Increase**, **EC2 instances**, and then complete the fields in the limit increase form. The current default limit for this instance type is 20 instances.

   You might need to request an increase if you already have an existing deployment that uses this instance type, and you think you might exceed the default limit with this reference deployment. It might take a few days for the new service limit to become effective. To learn more, see Amazon EC2 Service Limits in the AWS documentation.



Create Case — Basic Support Plan Change

Choose language: English

Name: [redacted]
Account: [redacted]
CC:
Required for IAM users; use commas or semicolons to separate email addresses

Regarding*: ( ) Account and Billing Support [● Service Limit Increase] ( ) Technical Support (Unavailable under the Basic Support Plan)

Limit Type*: EC2 Instances

Request 1
Region*: US West (Oregon)
Primary Instance Type*: c3.8xlarge
Limit*: Instance Limit
New limit value*: 25

Figure 4: Requesting a Service Limit Increase

Step 2. Launch the Puppet Stack

The automated AWS CloudFormation template provided with this Quick Start deploys Puppet into an Amazon VPC. Please make sure that you've completed the previous steps before launching the stack.

1. Launch the AWS CloudFormation template into your AWS account.

   Launch Quick Start

   The template is launched in the US West (Oregon) Region by default. You can change the region by using the region selector in the navigation bar.

   This stack takes approximately 20 minutes to create.

   Note You are responsible for the cost of the AWS services used while running this Quick Start reference deployment. There is no additional cost for using this Quick Start. As of the date of publication, the cost for using the Quick Start with default settings is approximately $0.18 an hour, and you can complete the initial deployment for about $0.18. Prices are subject to change. See the pricing pages for each AWS service you will be using in this Quick Start for full details.

   You can also download the template to use it as a starting point for your own implementation.

2. On the **Select Template** page, keep the default URL for the AWS CloudFormation template, and then choose **Next**.

3. On the **Specify Details** page, review the parameters for the template. These are described in the following table.

   Provide a value for the **KeyPairName** parameter. This parameter require your input. For all other parameters, the template provides default settings that you can customize.

Parameter label	Parameter name	Default	Description
Select a key pair	**KeyPairName**	*Requires input*	Public/private key pair, which enables you to connect securely to your instance after it launches. When you created an AWS account, this is the key pair you created in your preferred region.
Source IP for remote access	**RemoteAdminCIDR**	*Requires input*	CIDR block or IP address for SSH and RDP access (e.g., 1.1.1.1/32).
CIDR range for your VPC	**VPCCIDR**	10.0.0.0/16	CIDR block for the VPC.

Parameter label	Parameter name	Default	Description
CIDR range for the subnet in your VPC	**SubnetCIDR**	10.0.0.0/19	CIDR block for the subnet.
IP address for the Puppet master	**PuppetMasterIP**	10.0.0.10	IP address for the instance where the Puppet master is deployed.
IP address for the Linux Puppet agent	**PuppetAgentLinuxIP**	10.0.0.11	IP address for the instance where the Linux Puppet agent is deployed.
IP address for the Windows Puppet agent	**PuppetAgentWindowsIP**	10.0.0.12	IP address for the instance where the Windows Puppet agent is deployed.

Note You can also download the template and edit it to create your own parameters based on your specific deployment scenario.

4. On the **Options** page, you can specify tags (key-value pairs) for resources in your stack and set additional options. When you're done, choose **Next**.
5. On the **Review** page, review and confirm the settings.
6. Choose **Create** to deploy the stack.
7. Monitor the status of the stack. When the status is **CREATE_COMPLETE**, the Puppet cluster is ready.

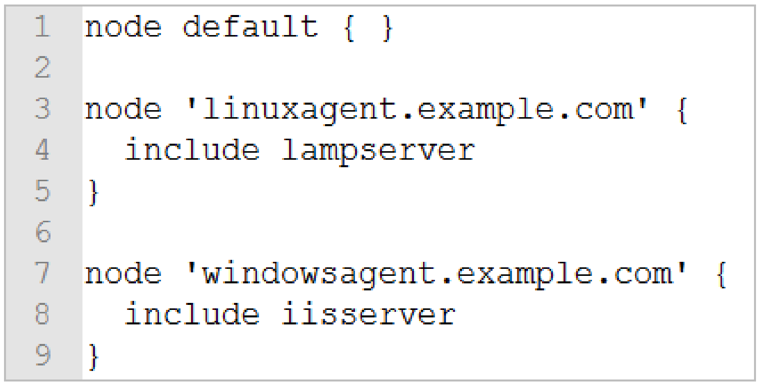
Step 3. Configure Puppet Agents

You can follow this instructions in this section to test your Puppet setup on AWS. We'll take a look at the module manifests for the Puppet agents, apply the configurations, and verify that the configurations were applied successfully.

Review Modules and Manifests

There are a number of ways to apply configurations to your agent nodes (see the Puppet Labs documentation). This Quick Start uses modules for each Linux and Windows node, and downloads these modules from Amazon S3 to the master during the bootstrapping phase.

Puppet programs are called *manifests*, which are developed using Puppet code. (For information about the Puppet language, see the Puppet Labs documentation.) The main manifest is called **site.pp** and is located on the master in **/etc/puppet/manifests**. Figure 5 shows the site.pp manifest used by the master in this Quick Start.



```
1  node default { }
2
3  node 'linuxagent.example.com' {
4    include lampserver
5  }
6
7  node 'windowsagent.example.com' {
8    include iisserver
9  }
```

Figure 5: The Main Manifest

This manifest includes three node declarations:

- **Line 1** – Defines a node block that can be applied by default to any system. We're not performing any common configurations, so there's no code within the curly braces.
- **Line 3** – Defines a node block for an agent named *linuxagent.example.com*. This is the Ubuntu agent launched by the Quick Start. Instead of placing resource definitions in this node block, we're referencing a class from a module called `lampserver`. Using classes is a great way to reduce code duplication. In this case, when the Linux agent applies its configuration, it will use the code from the `lampserver` class to define the state of the system.
- **Line 7** – Defines a node block for an agent named *windowsagent.example.com*. This is the Windows Server 2012 R2 agent launched by the Quick Start. Instead of placing resource definitions in this node block, we're referencing a class from a module called `iisserver`. When the Windows agent applies its configuration, it will use the code from the `iisserver` class to define the state of the system.

Next, let's look at the `lampserver` and `iisserver` classes to see what they do.

The `lampserver` class is defined in a module called `lampserver`. The manifest file for the module is named **init.pp** and is located in **/etc/puppet/modules/lampserver/manifests** on the master.

```
 1 class lampserver {
 2   exec { 'apt-update':
 3     command => '/usr/bin/apt-get update'
 4   }
 5
 6   package { 'apache2':
 7     require => Exec['apt-update'],
 8     ensure => installed,
 9   }
10
11   service { 'apache2':
12     ensure => running,
13   }
14
15   package { 'mysql-server':
16     require => Exec['apt-update'],
17     ensure => installed,
18   }
19
20   service { 'mysql':
21     ensure => running,
22   }
23
24   package { 'php5':
25     require => Exec['apt-update'],
26     ensure => installed,
27   }
28
29   file { '/var/www/html/info.php':
30     ensure => file,
31     content => '<?php  phpinfo(); ?>',
32     require => Package['apache2'],
33   }
34 }
```

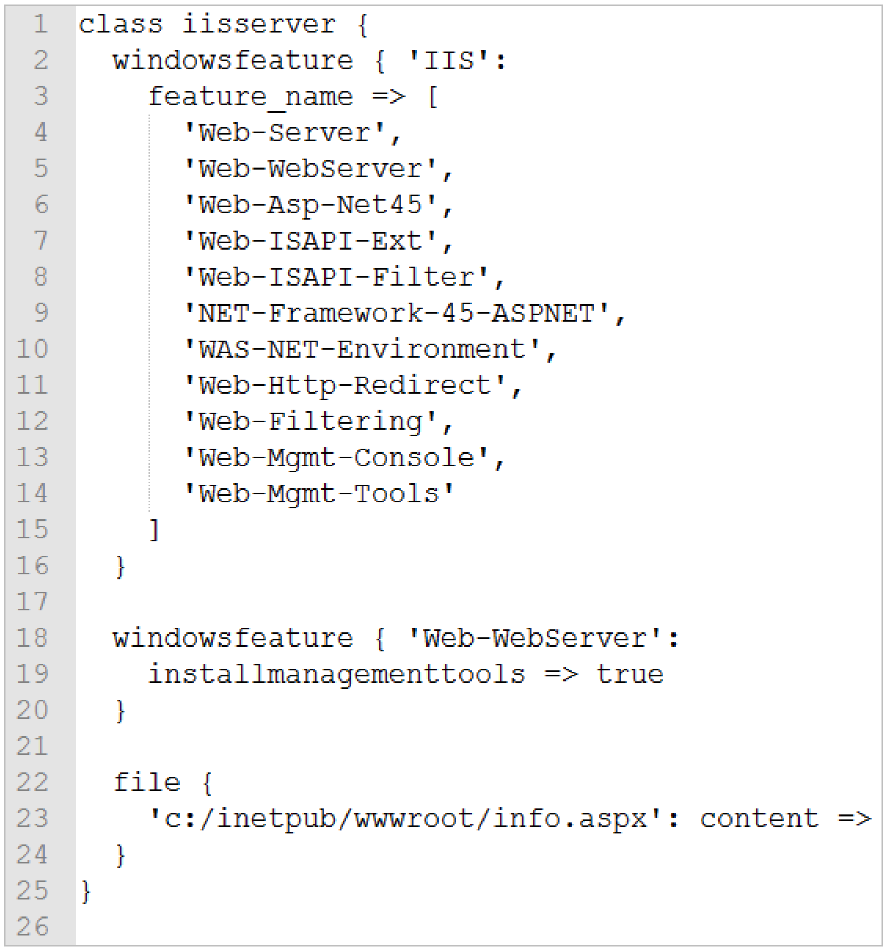
Figure 6: The `lampserver` Class

Note the following about the `lampserver` code shown in Figure 6:

- **Line 1** – This is the class definition for `lampserver`, which is referenced in our main manifest file.

- **Line 2** – The `exec` keyword defines a resource declaration. You use resources to describe the desired state of the system. Here we're using the `exec` resource to execute the `apt-update` command on the node.
- **Line 6** – The `package` resource is used to install Apache 2 on the node. Notice that the `require` statement ensures that `apt-update` has already been run before this resource can be installed.
- **Line 11** – The `service` resource ensures that the Apache 2 service is running.
- **Line 15** – The `package` resource ensures that the MySQL server is installed, as long as `apt-update` has been executed successfully.
- **Line 20** – The `service` resource ensures that MySQL is running.
- **Line 24** – The `package` resource ensures that PHP 5 is installed, as long as `apt-update` has been executed successfully.
- **Line 29** – The `file` resource ensures that a new file called **info.php** is created in the default **apache** root directory. This requires Apache 2 to be installed. PHP code is added to the content of the file to provide an informational page about the web server when the user visits the site in a web browser.

The `iisserver` class is defined in a module called `iisserver`. The manifest file for the module is named **init.pp** and is located in **/etc/puppet/modules/iisserver/manifests** on the master.



```
 1 class iisserver {
 2   windowsfeature { 'IIS':
 3     feature_name => [
 4       'Web-Server',
 5       'Web-WebServer',
 6       'Web-Asp-Net45',
 7       'Web-ISAPI-Ext',
 8       'Web-ISAPI-Filter',
 9       'NET-Framework-45-ASPNET',
10       'WAS-NET-Environment',
11       'Web-Http-Redirect',
12       'Web-Filtering',
13       'Web-Mgmt-Console',
14       'Web-Mgmt-Tools'
15     ]
16   }
17
18   windowsfeature { 'Web-WebServer':
19     installmanagementtools => true
20   }
21
22   file {
23     'c:/inetpub/wwwroot/info.aspx': content =>
24   }
25 }
26
```

Figure 7: The `iisserver` Class

Note the following about the `iisserver` code shown in Figure 7:

- **Line 1** – This is the class definition for `iisserver`, which is referenced in our main manifest file.
- **Line 2** – The `windowsfeature` resource leverages Windows PowerShell to ensure that all the required components for IIS and ASP.NET are installed.
- **Line 18** – The `windowsfeature` resource installs the management tools for IIS administration.
- **Line 22** – The `file` resource ensures that an informational ASP.NET web page called **info.aspx** is present in the web server root directory. The content of this web page is truncated in Figure 7 because of space constraints, but it contains a single page directive that provides information about the server, just like **info.php** on the Linux node.

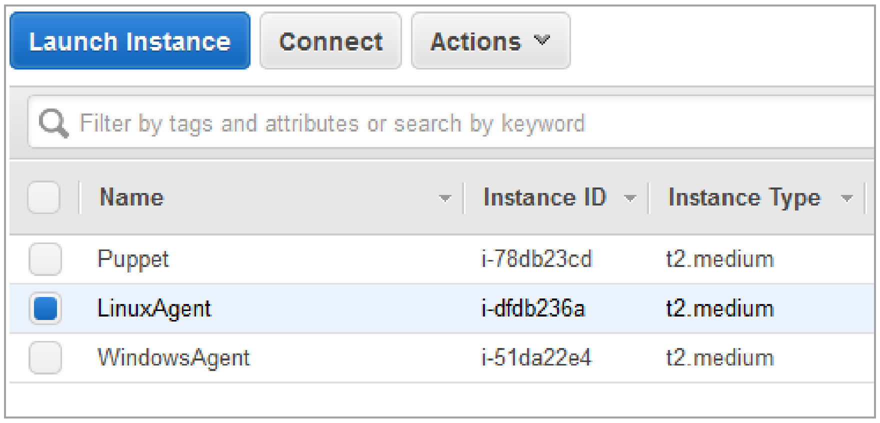
In addition to creating your own modules, you can use manifests directly, or you can leverage pre-existing modules from the Puppet Forge. For details on writing modules and manifests, see Module Fundamentals and the training classes on the Puppet Labs website.

Connect to Puppet Agents

Now that you understand what the sample modules are intended to do, you're ready to connect to your agents remotely.

Linux Agent

You'll need to use SSH to connect to your Linux agent from outside the Amazon VPC. In the Amazon EC2 console, select the EC2 instance tagged **LinuxAgent**, as shown in Figure 8.



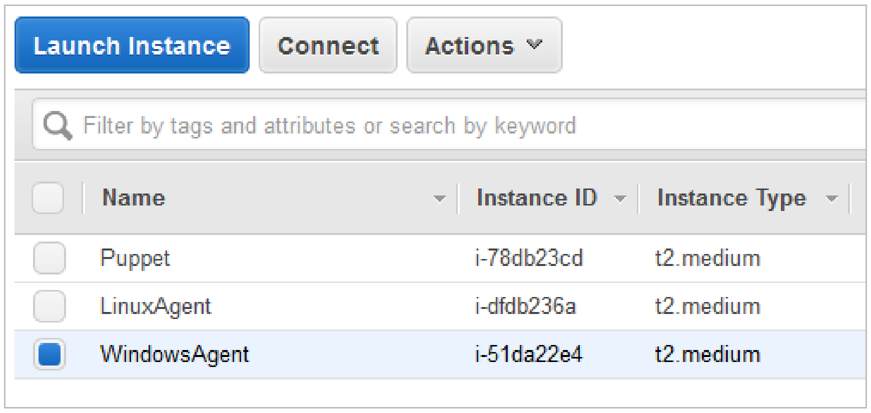
	Name	Instance ID	Instance Type
☐	Puppet	i-78db23cd	t2.medium
☑	LinuxAgent	i-dfdb236a	t2.medium
☐	WindowsAgent	i-51da22e4	t2.medium

Figure 8: Selecting the LinuxAgent Instance

Retrieve the public DNS name for **LinuxAgent**, and follow the instructions in the *Amazon EC2 User Guide for Linux Instances* to connect your SSH client to the instance. You'll need to have your key pair available to establish a remote SSH connection.

Windows Agent

You can use RDP to connect to the Windows agent over the Internet. In the Amazon EC2 console, select the EC2 instance tagged **WindowsAgent**, as shown in Figure 9.



	Name	Instance ID	Instance Type
	Puppet	i-78db23cd	t2.medium
	LinuxAgent	i-dfdb236a	t2.medium
✓	WindowsAgent	i-51da22e4	t2.medium

Figure 9: Selecting the WindowsAgent Instance

Retrieve the public DNS name for **WindowsAgent**, and follow the instructions in the *Amazon EC2 User Guide for Microsoft Windows Instances* to get connected. You'll need to have your key pair available to decrypt the Windows administrator password and establish a remote connection.

Apply Configurations

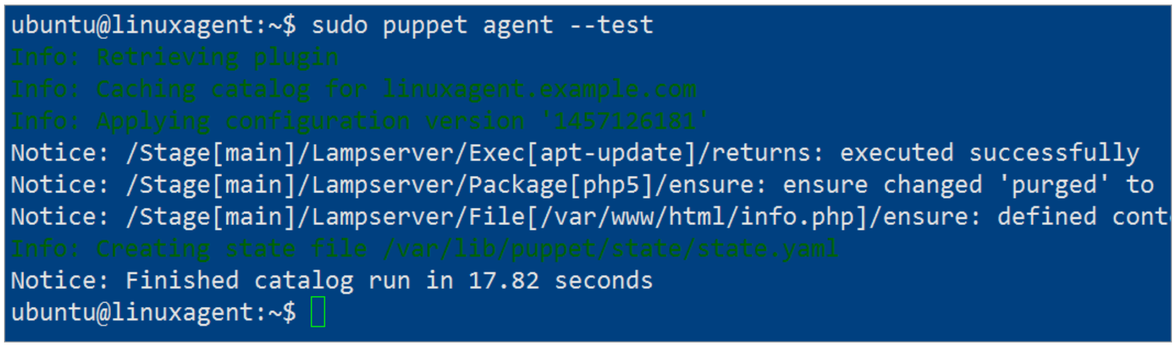
In this section, you'll apply node configurations and verify that everything was configured successfully.

Linux Agent

Once you've connected to your Linux agent via SSH, run the following command to apply the configuration in the `lampserver` module:

```
sudo puppet agent --test
```

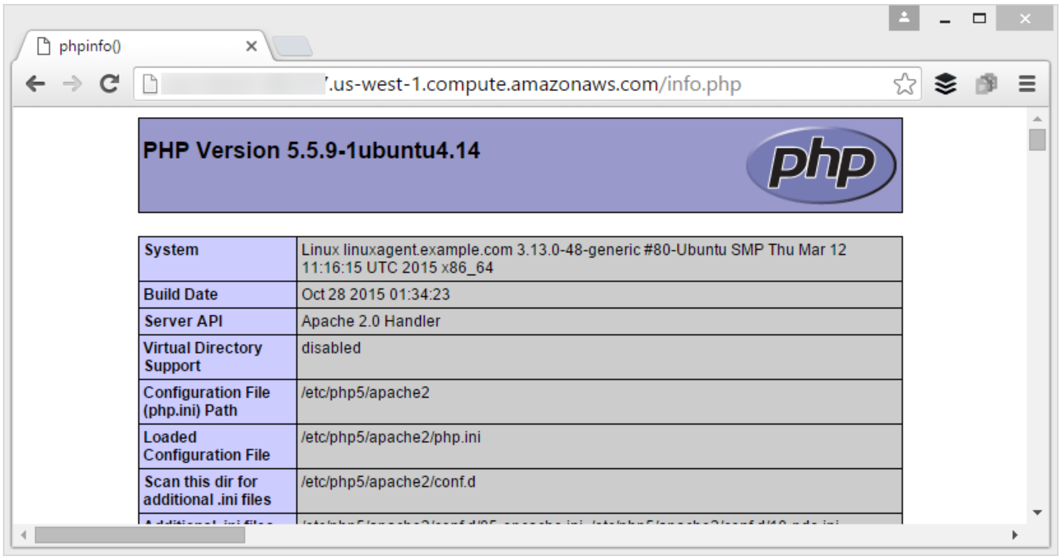
You should see output similar to Figure 10, indicating that the configuration was applied successfully.



```
ubuntu@linuxagent:~$ sudo puppet agent --test
Info: Retrieving plugin
Info: Caching catalog for linuxagent.example.com
Info: Applying configuration version '1457126181'
Notice: /Stage[main]/Lampserver/Exec[apt-update]/returns: executed successfully
Notice: /Stage[main]/Lampserver/Package[php5]/ensure: ensure changed 'purged' to
Notice: /Stage[main]/Lampserver/File[/var/www/html/info.php]/ensure: defined cont
Info: Creating state file /var/lib/puppet/state/state.yaml
Notice: Finished catalog run in 17.82 seconds
ubuntu@linuxagent:~$
```

Figure 10: Linux Puppet Agent Output

Next, open up a web browser and navigate to the info.php page. You'll need to use the public DNS name of the **LinuxAgent** EC2 instance—for example, http://<*public DNS name*>/info.php.



PHP Version 5.5.9-1ubuntu4.14

System	Linux linuxagent.example.com 3.13.0-48-generic #80-Ubuntu SMP Thu Mar 12 11:16:15 UTC 2015 x86_64
Build Date	Oct 28 2015 01:34:23
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d

Figure 11: Testing the Apache Web Server

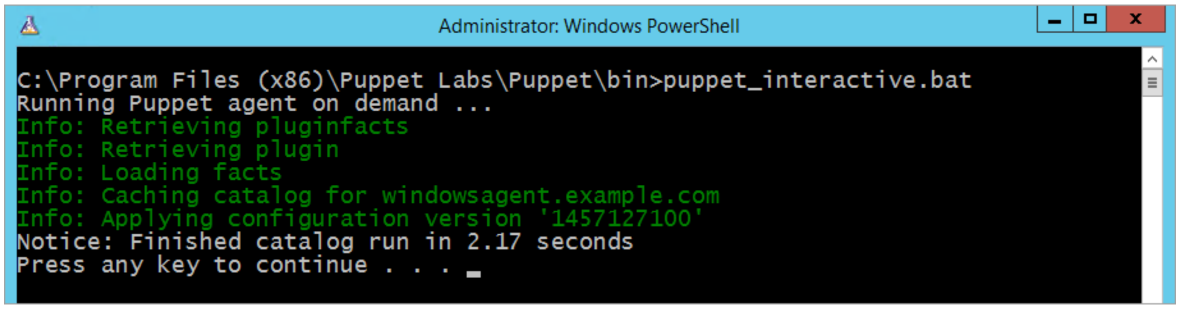
You should see a PHP version page similar to the one shown in Figure 11. This indicates that you've successfully applied the configuration to your Linux agent.

Windows Agent

Once you've connected to your Windows agent via RDP, find the **Start Command Prompt with Puppet** shortcut on the Start screen. Open the context (right-click) menu for the shortcut, and then choose **Run as administrator**. Run the following command to apply the configuration in the `iisserver` module.

```
puppet_interactive.bat
```

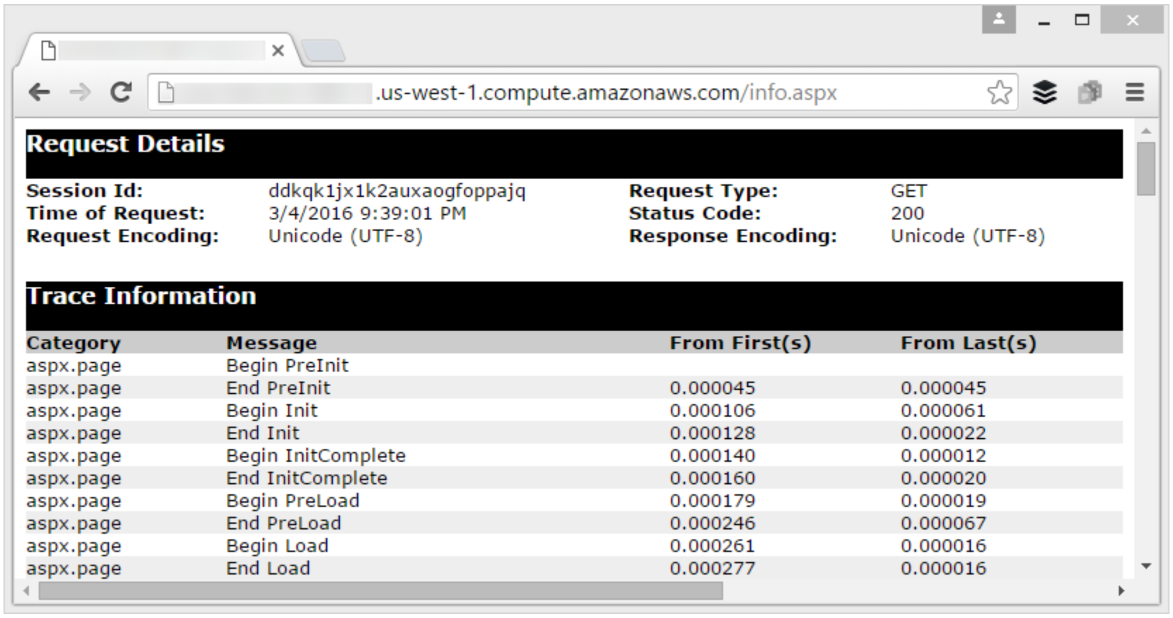
You should see output similar to Figure 12, indicating that the configuration was applied successfully.



```
C:\Program Files (x86)\Puppet Labs\Puppet\bin>puppet_interactive.bat
Running Puppet agent on demand ...
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for windowsagent.example.com
Info: Applying configuration version '1457127100'
Notice: Finished catalog run in 2.17 seconds
Press any key to continue . . .
```

Figure 12: Windows Puppet Agent Output

Finally, open up a web browser and navigate to the info.aspx page. You'll need to use the public DNS name of the **WindowsAgent** EC2 instance—for example, http://<*public DNS name*>/info.aspx.



Request Details

Session Id:	ddkqk1jx1k2auxaogfoppajq	**Request Type:**	GET
Time of Request:	3/4/2016 9:39:01 PM	**Status Code:**	200
Request Encoding:	Unicode (UTF-8)	**Response Encoding:**	Unicode (UTF-8)

Trace Information

Category	Message	From First(s)	From Last(s)
aspx.page	Begin PreInit		
aspx.page	End PreInit	0.000045	0.000045
aspx.page	Begin Init	0.000106	0.000061
aspx.page	End Init	0.000128	0.000022
aspx.page	Begin InitComplete	0.000140	0.000012
aspx.page	End InitComplete	0.000160	0.000020
aspx.page	Begin PreLoad	0.000179	0.000019
aspx.page	End PreLoad	0.000246	0.000067
aspx.page	Begin Load	0.000261	0.000016
aspx.page	End Load	0.000277	0.000016

Figure 13: Testing the IIS Web Server

You should see an IIS version page similar to the one shown in Figure 13. This indicates that you've successfully applied the configuration to your Windows agent.

Security

A *security group* acts as a firewall that controls the traffic for one or more instances. When you launch an instance, you associate one or more security groups with the instance. You add rules to each security group that allow traffic to or from its associated instances. You can modify the rules for a security group at any time. The new rules are automatically applied to all instances that are associated with the security group.

The security groups created and assigned to the individual instances as part of this solution are restricted as much as possible while allowing access to the various functions needed by Puppet. We recommend that you review security groups and further restrict access as needed once Puppet is up and running.

Additional Resources

AWS services

- Amazon EC2
  http://aws.amazon.com/documentation/ec2/
- AWS CloudFormation
  http://aws.amazon.com/documentation/cloudformation/
- Amazon VPC
  http://aws.amazon.com/documentation/vpc/
- Amazon Route 53
  https://aws.amazon.com/documentation/route53/

Puppet resources

- Puppet documentation
  https://docs.puppetlabs.com
- Puppet Labs Training
  https://learn.puppetlabs.com
- Puppet on AWS
  https://puppetlabs.com/solutions/aws
- Provision AWS Infrastructure Using Puppet (blog post)
  https://puppetlabs.com/blog/provision-aws-infrastructure-using-puppet

- Automating AWS with Puppet (video)
  https://youtu.be/eyR0LVjxJAs
- Puppet and AWS: Getting the Best of Both Worlds (video)
  https://puppetlabs.com/presentations/puppet-and-aws-getting-best-both-worlds
- Puppet Forge
  https://forge.puppetlabs.com/
- Puppet Community
  https://puppetlabs.com/community/overview

Quick Start reference deployments

- AWS Quick Start home page
  https://aws.amazon.com/quickstart/
- Quick Start deployment guides
  https://aws.amazon.com/documentation/quickstart/

Send Us Feedback

We welcome your questions and comments. Please post your feedback on the AWS Quick Start Discussion Forum.

Document Revisions

Date	Change	In sections
March 2016	Initial publication	-