Hewlett Packard Enterprise

HPE Digital Learner Chef and Puppet for DevOps Content Pack

HPE Content Pack number	CP027
Content Pack length	26 Hours
Content Pack category	Category 2
Learn more	**View now**

In this course, you will learn how to automate the configuration, provisioning and management of physical and virtual servers using Chef and Puppet

Why HPE Education Services?

- IDC MarketScape leader 5 years running for IT education and training*
- Recognized by IDC for leading with global coverage, unmatched technical expertise, and targeted education consulting services*
- Key partnerships with industry leaders OpenStack®, VMware®, Linux®, Microsoft®, ITIL, PMI, CSA, and SUSE
- Complete continuum of training delivery options—self-paced eLearning, custom education consulting, traditional classroom, video on-demand instruction, live virtual instructor-led with hands-on lab, dedicated onsite training
- Simplified purchase option with HPE Training Credits

Audience

System administrators, DevOps personnel, operations staff, managers, software developers or testers

Content Pack objectives

- Learn the basics of Chef including vocabulary, installations, roles and tools, and create recipes and cookbooks
- Use Chef Delivery to continuously deliver applications and infrastructure
- Use the Chef development kit and tools such as Test Kitchen, ChefSpec, and Foodcritic
- Use Chef analytics to provide real-time visibility into what is occurring on the Chef server
- Learn advanced Chef administration
- Identify components of the Puppet architecture
- Identify the major Puppet commands and how to execute them
- Learn to configure Puppet and manage files
- Learn to build and administer a complex Puppet installation
- Puppet configuration refactoring

*Realize Technology Value with Training, IDC Infographic 2037, Sponsored by HPE, October 2017

Detailed Content Pack outline

Working with Chef

Working with Chef: Fundamentals

Chef is a configuration management tool that is used to streamline the task of configuring and maintaining company servers. Chef uses recipes or cookbooks to describe how Chef manages server applications and utilities, and how they are to be configured. In this course, you will learn how to create recipes and cookbooks and understand the Chef vocabulary. You will also learn about various Chef installations, Chef roles and Chef tools.

- Describe Chef and how it is used for configuration management
- Recognize that Chef uses an imperative language and distinguish between imperative and declarative languages
- Describe the contents of recipes and cookbook and how Chef uses them when performing configuration
- Describe the terms used by Chef for performing automated configuration management
- Identify other popular DevOps tools used for automated configuration management
- Describe how Chef functions and how it is supported in operating systems such as Linux and Windows
- Compare Chef's cloud-based functionality to traditional operating systems
- Compare the different Chef platforms and decide which one to use
- Recognize the different methods of installing Hosted Enterprise Chef
- Describe how Chef Solo works
- Work with the Chef Administrator's Workstation
- Identify how Enterprise Chef runs, and how it is used
- Distinguish between open source Chef and other Chef installations
- Describe the features of hosted enterprise Chef
- Create a node to be managed by Chef
- Identify how Chef uses Knife to create cookbooks
- Describe how Chef uses per-environment runlists
- Identify how Chef determines if a configuration is out of policy, and how Chef reapplies policy to the desired state
- Describe the different kinds of Chef nodes
- Identify and retrieve Chef node attributes
- Create and upload Chef roles
- Use the Chef development kit and tools such as Test Kitchen, ChefSpec and Foodcritic
- Use Chef analytics to provide real-time visibility into what is occurring on the Chef server
- Use the Chef management console for the management of nodes, roles, data bags, environments and cookbooks
- Use Chef Delivery to continuously deliver applications and infrastructure
- Work with Chef and create a per-environment runlist

Chef - Beyond the Basics

Working with Chef Analytics

The demand for IT personnel having an in depth knowledge of Chef and Chef Analytics is increasing day by day. Chef Analytics is a configuration management automation tool used for auditing changes on a Chef Server. In this course, you will learn the basics of Chef Analytics as well as its installation, configuration and set up of simple alerts and notifications.

- Describe the purpose, architecture, components and network topology of Chef Analytics
- Distinguish between noteworthy Chef Analytics actions and events
- Describe how action logs are generated and recognize the Chef pipeline architecture
- Describe the network and system prerequisites for installing Chef Analytics
- Prepare the Enterprise Chef Server to communicate with Chef Analytics
- List the steps to install Chef Analytics
- List the post installation steps of Chef Analytics that need to be performed on the Enterprise Chef server
- Use and deploy configuration files on the Chef Analytics server
- Describe how hostnames are used in Chef Server/ Chef Analytics architecture
- Verify successful installation of Chef Analytics by using command line utilities
- Log in to the Chef Analytics server and perform simple navigation
- Create a Chef client instance by starting Chef Client on a managed node
- Create an SMTP mail server to send Chef Analytics notifications
- Create simple notifications
- Create simple rules
- Create a simple trigger
- Recognize how audit-mode can be used to enforce server compliance
- Use Chef Tools such as Virtual Box, Vagrant and Test Kitchen
- Recognize how to create a cookbook for server auditing
- Create and apply recipes to a Test Kitchen instance
- Create and use a web server cookbook for auditing
- Describe how to make updates to a web server to enforce compliance
- Create an audit alert when auditing a server for compliance fails
- Define audit triggering rules and describe what happens when an audit fails
- Recognize how audit rules are enforced in an auditing cookbook
- Create a recipe and a cookbook that will enforce compliance on a server and send a notification if the server is not compliant

Chef Advanced Administration

Since Chef is an ace configuration management tool, an in depth knowledge of its advanced features is a fundamental requirement for a skilled DevOps engineer. In this course, you will learn the inner workings of Chef, including managing cookbooks and using troubleshooting tools. This course also covers virtualization, user management and server configuration.

- Inspect cookbooks and manage cookbook dependencies
- Use Chef and Vagrant together for virtualization and automated provisioning
- List the various ways to delete nodes from a Chef server
- Create complex Chef recipes and cookbooks
- Freeze/unfreeze a cookbook and describe the benefits of doing so
- Use roles to group nodes with the same function into environments
- Describe how a Chef client can automatically be started as a daemon
- Use Chef shell to set runtime breakpoints within recipes
- Describe how to handle multiple versions cookbooks
- Use Test Kitchen as a test harness tool for your workflow
- Use ChefSpec to test resources and recipes in a simulated Chef Client run
- Use Foodcritic to find common syntax and best practice problems in your cookbooks
- Describe the rules and best practices for writing Chef DSL
- Prepare and evaluate Cookbooks using best practices
- List the individual steps that occur when the nodes managed by Chef server are provisioned
- Use an environment cookbook to manage application configuration
- Create Chef users
- Use configuration files and packages in a Chef installation
- Implement high availability within the Chef architecture to provide server failover support
- Recognize how Chef will allow users to login with their network credentials (LDAP)
- Describe how the Chef server is authenticated using the Chef server API
- Recognize how the Chef community uses GitHub to share code and to collaborate on projects
- Describe how Bitbucket can be used as a private code repository for Chef cookbooks
- Describe how Chef functionality can be extended into other DevOps tools
- List the places where major Chef training events and conferences are held
- Create an alert and notification process in a control

Mastering DevOps with Chef

First Steps with Chef

How does expressing Infrastructure as Code with Chef accelerate building, deploying and managing infrastructure? This course focuses on executing Chef code with the chef-client and introduces the DSL for writing recipes and cookbooks.

- Describe how automation and version control contribute to DevOps
- Describe the Chef distribution model and the declarative syntax used in writing Chef recipes
- Install the Chef Development Kit (ChefDK)
- Use the package and file resources inside of a Chef Recipe
- Describe Ruby basics such as variables, arrays and objects
- Identify how recipes are packaged and distributed with cookbooks
- Identify best practices of using Git for version control
- Deploy a cookbook using the chef-client in local mode
- Work with system profiling with Ohai and accessing node object attributes
- Use the cookbook file, remote file, and template Chef resources to manage files
- Manage dynamic file creation using the template resource
- Refactor recipes to use node attributes instead of hard-coded values
- Build a simple Apache cookbook that configures a “hello, world” page to serve on the localhost

Chef Server Basics

Using a Chef server can accelerate the process of deploying cookbooks while providing visibility into the state of your infrastructure. This course focuses on Chef server policy and remotely deploying cookbooks.

- Describe why a Chef server is used and the cookbook distribution model
- Create an account on hosted Chef
- Configure a chef-repo using the Chef server starter kit
- Describe the chef-repo components and the user authentication model
- Use the knife command to view Chef server policy and node details
- Upload cookbook policy to the Chef server using knife
- Attach nodes to the Chef server with the knife bootstrap command
- Configure run-lists for nodes in bulk with roles
- Separate cookbook versions using Chef environments
- Search for Chef server policy with knife
- Refactor a recipe to utilize search to create dynamic policy with a Chef server
- Create custom searchable datasets with data bags
- Build a users cookbook that searches through a data bag to configure users and groups

Community Cookbooks and Chef Server Patterns

The best practices of using a Chef Server are directly related to the reusability of cookbook recipes and components. This course focuses on the effective use of community cookbooks with a Chef server.

- Describe the reusability of Chef cookbooks and best practices around utilizing community code
- Search for cookbooks on supermarket.chef.io
- Distinguish the differences between library and application cookbooks
- Distinguish why community cookbooks should be called as dependencies instead of forking upstream code bases
- Configure and install dependencies in the metadata file with berkshelf
- Manage Chef server cookbook versions with berkshelf
- Configure the Berksfile to point at a local dependency instead of a Chef Supermarket dependency
- Use node attribute precedence when overwriting cookbook attribute values
- Assign node attributes at the role or environment level
- Identify the limitations of using roles and explain the purpose of a role cookbook
- Configure the chef-client as a service with a community cookbook
- Configure the logging location on a node directly or as a node attribute with the chef-client cookbook
- Configure a wrapper for the test haproxy community cookbook to redirect traffic to a simple web server

Test-Driven Cookbooks

The most important role of a Cookbook developer is to test code before it enters production. This course focuses on testing Chef cookbooks consistently and effectively.

- Use Behavior-Driven Development (BDD) in the context of Chef cookbook development
- Identify the Chef Development Kit tools used for unit and integration testing
- Generate unit and integration tests inside of a cookbook
- List the components of the .kitchen.yml file
- Configure a Test Kitchen driver to support deploying to physical, virtual or cloud machines
- Execute the chef-client on a virtual machine generated by Test Kitchen
- Define an integration test and verify the results with kitchen login
- Use the InSpec compliance language to write simple integration tests
- Refactor a recipe and run kitchen verify with a test-driven approach
- Execute the RSpec utility to test Chef recipes in memory
- Utilize a simple formula to write ChefSpec tests
- Configure the Pry Ruby gem to insert a breakpoint into a recipe
- Use ChefSpec to check case statement evaluation of node attributes
- Use Test Kitchen to verify a simple Apache cookbook on Ubuntu and Centos

Using Puppet

Implementation and Benefits

Puppet is an open source configuration management tool. Puppet is closely coupled with the DevOps methodology and is implemented with its own declarative language (Ruby). Mostly used by system administrators, Puppet automates the configuration, provisioning and management of physical and virtual servers. With automation, Puppet eliminates the errors that occur with manual tasks. In this course, you will learn the core concepts of Puppet and how to configure Puppet for your organization.

- Describe Puppet and the problems it solves
- Describe the core functional concepts of Puppet and the related vocabulary
- Compare Puppet with other DevOps configuration management tools
- Identify organizations who use Puppet and how Puppet is integrated into their DevOps methodology
- Use declarative, readable Puppet DSL (Domain Specific Language) to describe system resources and state
- Identify the different components of the Puppet architecture
- Describe the basic functionality of Puppet nodes and recognize how they are used
- Distinguish between cloud and network installations
- Describe the features of open source Puppet
- Identify the uses and features of Puppet Enterprise (PE)
- Recognize the different versions of Puppet and their features
- List the general installation requirements of Puppet
- Install Puppet on a Linux system
- Describe how to install Puppet on the Windows operating system
- Identify the major Puppet commands and how to execute them
- List the steps needed to start and stop Puppet
- Work with Puppet to make configuration changes on a machine
- Work with configuring nodes to add hosts to a puppet setup
- Describe the predefined Puppet resources and how they are used
- Describe from a conceptual level how Puppet uses manifests
- Identify the different methods in grouping resources within a manifest
- Describe what a Puppet class is and how it is used
- Recognize how Puppet uses modules
- Recognize how to use Puppet Forge to create and share Puppet modules
- Describe how Puppet uses catalogs
- Define pre-requirements for a Puppet installation on a local machine

Configuration and Programming

Puppet is an open source configuration management tool. Puppet is closely coupled with the DevOps methodology and is implemented with its own declarative language (Ruby). Mostly used by system administrators, Puppet automates the configuration, provisioning and management of physical and virtual servers. With automation, Puppet eliminates the errors that occur with manual tasks. In this course, you will learn how to configure Puppet and manage its resources. Puppet programming, implementation and troubleshooting techniques are discussed and demonstrated.

- Describe how coding is performed in Puppet and how it is managed
- Describe the core concepts of Puppet service management and the related vocabulary
- Identify Puppet resources and how to handle their dependencies
- Identify the contents of a Puppet configuration file and how to manage it
- Identify the Package-File-Service pattern and how it is used in Puppet development
- Work with an end-to-end example of a Puppet task
- Create and set up a user account
- Identify how Puppet uses SSH
- Describe how SSH keys are distributed in a Puppet installation
- Distinguish between and manage user accounts
- Work with exec resources to run commands
- Work with cron resources to run scheduled jobs
- Recognize and deploy file tree structures
- Use different kinds of Puppet templates to create configuration files
- Describe the syntax of flow of control statements
- Use program operators in Puppet
- Identify the different expressions used in Puppet
- Use variables in Puppet
- Recognize how to substitute expressions
- Describe the different kinds of collections
- Work with different kinds of Puppet reports
- Describe the different Puppet modes
- Identify how to print messages
- Work with different methods of monitoring Puppet
- Describe common runtime failures
- Establish a configuration management solution using Puppet

Puppet - Beyond the Basics

Working with Puppet Agent and Puppet Apply

Since the demand for IT personnel having an in depth knowledge of Puppet is exploding, knowing the advanced features of Puppet is an essential requirement for skilled developers and operations managers. In this course, you will learn the intermediate and advanced operations of Puppet as well as Puppet administration. This course also demonstrates the unique interaction between Puppet Server, Puppet Master, Puppet Agent and Puppet Apply.

- Distinguish between the roles of Puppet Server, Puppet Master, Puppet Agent and Puppet Apply
- Summarize the role of the Puppet Server and describe its relationship with Puppet Master
- Install Puppet Server
- Set up initial certificate and domain name for a Puppet Server installation
- Review the configuration steps and describe the contents of each major configuration file
- Recall the role of Ruby, and install and remove Ruby gems
- Describe the types of variables used within Puppet
- Apply control branching statements of Puppet
- Write and call built-in and custom functions
- Write Puppet code to apply specific configurations to specific nodes
- Use Puppet code to access facts
- Describe how Puppet scales as the number of nodes grow
- Describe the certificate signing process between Puppet Master and Puppet Agent in detail
- Describe how the Puppet Agent executes its main manifest
- Describe the communication between Puppet Master and Puppet Agent in detail
- List the start options for Puppet Agent and describe their functionality
- Use the commandline to start Puppet Agent and interpret commandline messages
- Execute Puppet Agent as a service
- Execute Puppet Agent on Linux machines as a scheduled cron job
- Describe how Puppet Agent performs logging and identify logging configuration issues
- Describe the functionality of Puppet Apply and its features
- Describe Puppet Apply's run environment and execute the main manifest
- Describe how Puppet Apply communicates over the network and uses local collections of modules
- Discuss how Puppet Apply handles logging and report handling
- Execute the Puppet resource command to set up a Puppet Apply cron job
- Create a cron job that will execute a Puppet Agent

Building and Administering a Complex Puppet Installation

Since Puppet skills are so much in demand, an in depth knowledge of its functionality is beneficial for a skilled IT professional. In this course, you will learn the interworking of Puppet modules and resources. This course also covers security issues (certificates and SSL), Puppet reporting, virtualization and Hiera.

- Review the best practices in module design and write modules using the module generator
- Use modules and plugins to enhance and extend the functionality of Puppet
- Use the Puppet module installation tool to install and uninstall modules
- Develop a module and deploy it on Puppet Forge
- Use Puppet's group and user resource types to manage group and user accounts
- Use Puppet's file resource type to manage folders and files
- Use Puppet to edit, create and delete a scheduled task
- Use Puppet's package resource type to manage software packages
- Use Puppet to manage operating system services
- Use the internal report handlers to generate reports and analyze YAML
- Create a custom report in Ruby and include it in a Puppet module
- Distinguish between older report formats and the new report format
- Describe the steps needed to configure external SSL termination on the Puppet Server
- Distinguish the functionality of Puppet from Vagrant
- Use Vagrant to create virtual machines
- Prepare a Puppet manifest to be deployed on Vagrant-created virtual machines
- Modify Vagrant configuration to use Puppet for provisioning
- Run Vagrant to create virtual machines provisioned by Puppet
- Recognize the problems and issues that can be fixed by Hiera
- Install Hiera from a package or a gem, and install Puppet functions
- Describe the Hiera global configuration settings and how configuration files are resolved at runtime
- Use Hiera and Puppet together
- Create a Puppet module that will be extrapolated by Hiera
- Create Vagrant virtual machines that will be provisioned by a Puppet manifest
- Compare the different options when configuring an external CA
- Describe how Puppet Server can be configured to use certificates from an existing external CA

Master DevOps with Puppet

Building the Puppet Environment

In this course, you will learn the basic concepts of automated configuration with Puppet. The installation and configuration of Puppet in the cloud are covered. Puppet facts for advanced server configuration are demonstrated.

- Describe the concepts of Puppet and its configuration options
- Install and configure Puppet master on Amazon Web Services EC2
- Install and configure Puppet agent on Amazon EC2
- Build out the Puppet cloud environment and get it all to work
- Use the Puppet Facter utility
- Use core Puppet facts for server configuration
- Use custom Puppet facts for server configuration
- Use the Puppet FACTERLIB variable for custom configuration options
- Use external Puppet facts for advanced configuration options
- Set a Puppet agent to be configured by a Puppet master

Configuring the Puppet Master and Puppet Agent

Most of the functionality of Puppet is contained in the Puppet master and the Puppet agent. In this course, you will learn how to configure Puppet for automated server configuration.

- Describe how Hiera works within Puppet
- Identify how Hiera is installed and configured in different versions of Puppet
- Identify the three layers of Hiera and how they are used
- Create a Puppet test class to be used by Hiera
- Code a module to be used by Puppet
- Create classes to be used by Puppet for configuration
- Create Puppet parameterized classes
- Describe the class inheritance hierarchy
- Describe the Puppet resources types and how they are used
- Write code to use Puppet resources
- Determine how to use attributes and types in Puppet
- Write code to use custom Puppet resource types
- Analyze the Puppet resource abstraction layer
- Design a Puppet master/Puppet agent configuration solution

Configuring Puppet Environments and Managing Files Puppet has robust file functionality and runs in multiple environments. In this course, you will learn how to get Puppet to work with files and how to set up and configure multiple Puppet environments.	• Describe how to use environments on the Puppet master • Identify the methods of setting the client environment in Puppet • Discover how the Puppet search path works • Analyze how to evaluate templates • Define the basics of using Puppet templates • Build and code Puppet templates	• Identify how EPP templates work within Puppet • Describe the Puppet file functions • Analyze the ways to deliver files with Puppet • Define how advanced file functionality works within Puppet • Use Puppet file functionality and environments
Puppet Refactoring Patterns There are different ways to refactor Puppet configurations. In this course, you will learn techniques to mitigate the risk and increase the success of Puppet configuration refactoring as it relates to the Puppet master and the Puppet agent.	• Analyze and use the Puppet Defined function • Identify how to correctly design a custom Puppet function • Code a new Puppet function • Describe how Puppet uses REST • Define the techniques for Puppet REST API Security • Describe how the Puppet file server works • Describe how to extend the functionality of Facter	• Discover the ways Puppet uses local directories • Describe how to create Ruby-based facts • Describe how to provide advanced configuration with modules • Describe how to provide advanced configuration with roles and profiles • Refactor Puppet Configuration

Learn more at
www.hpe.com/ww/digitallearner

www.hpe.com/ww/digitallearner-contentpack

Interested in purchase of this Content Pack as a stand-alone WBT? Contact Us for information on purchasing this Content Pack for individual use.

Follow us:

CP027 A.00, February 2019

Hewlett Packard Enterprise