Puppet Certified Professional Practice Exam (Sample)

Study Guide

Everything you need from our exam experts!

Questions

1. **In Puppet, which directive allows you to include other classes in your manifests?**
   - **A. include**
   - **B. require**
   - **C. import**
   - **D. extent**

2. **Which command will display all installed modules from the Puppet Forge?**
   - **A. puppet module list**
   - **B. puppet module show**
   - **C. puppet list modules**
   - **D. puppet modules installed**

3. **What does the $:: in Puppet signify when used in a variable?**
   - **A. It indicates a local variable**
   - **B. It refers to a top scope variable**
   - **C. It signifies a class variable**
   - **D. It denotes a variable in a specific module**

4. **Where will the autoloader look for the class definition for ssh::server::keys?**
   - **A. /etc/puppetlabs/puppet/modules/ssh/modules/keys.pp**
   - **B. /etc/puppetlabs/puppet/modules/ssh/manifests/key.pp**
   - **C. /etc/puppetlabs/puppet/modules/ssh/manifests/server/keys.pp**
   - **D. /opt/puppetlabs/puppet/modules/ssh/manifests/server/keys.pp**

5. **A class with the same name as the module goes into which file?**
   - **A. lib/self.pp**
   - **B. self.pp**
   - **C. manifests/class**
   - **D. manifests/init.pp**

6. Which of the following are valid parameters for the Puppet module install command?

A. --version, --modulepath, --force, --debug

B. --name, --ensure, --version, --force

C. --installpath, --name, --modulepath, --debug

D. --force, --timeout, --debug, --version

7. True or False: The resource type exec has the ability to listen for relationship notifications such as subscribe and notify.

A. True

B. False

C. Only works with certain resource types

D. Exec can only send notifications

8. In Puppet, what is the primary role of the 'node' declaration?

A. To create modules

B. To define resource collections for specific nodes

C. To set parameters globally

D. To manage data paths

9. True or False: Resource Collectors can only search on attributes which are present in the manifests and cannot read the state of the target system.

A. True

B. False

C. Only partially true

D. Not applicable

10. Which designation is not a valid directory for storing configuration components in a Puppet module?

A. Templates

B. Files

C. Settings

D. Manifests

Answers

1. A
2. A
3. B
4. C
5. D
6. A
7. A
8. B
9. A
10. C

Explanations

1. In Puppet, which directive allows you to include other classes in your manifests?

A. include

B. require

C. import

D. extent

The directive that allows you to include other classes in your Puppet manifests is the "include" directive. This directive is essential for organizing Puppet manifests by enabling the reuse of defined classes within other classes or nodes. When you use the include directive, Puppet ensures that the specified class is evaluated and its resources are applied as part of the catalog compilation process. This functionality enhances modularity and maintainability in your Puppet code, allowing you to break down complex configurations into smaller, manageable sections. It is common practice to define reusable classes for common configurations and then include them in multiple manifests. The other options do not provide the same capability. The "require" directive is used to establish dependencies between resources, ensuring that one resource is applied before another. "Import" is not a recognized directive in Puppet, and "extent" is also not a valid directive. Therefore, "include" is the only correct choice for integrating class definitions into Puppet manifests.

2. Which command will display all installed modules from the Puppet Forge?

A. puppet module list

B. puppet module show

C. puppet list modules

D. puppet modules installed

The command that displays all installed modules from the Puppet Forge is "puppet module list." This command is specifically designed to show a comprehensive list of all the modules that have been installed on the system using Puppet. It provides critical information, such as the name, version, and any other relevant details about the modules. This command retrieves data directly from the Puppet environment and reflects the current state of module installation on the system. It's an essential tool for Puppet users to manage and review their installed modules, ensuring that they are aware of what is currently available for use in their Puppet configurations. The other options do not provide the same functionality. For example, "puppet module show" typically displays details for a specific module rather than listing all installed modules. The other commands either do not exist or do not serve the purpose of listing all installed modules in the manner that "puppet module list" does.

3. What does the $:: in Puppet signify when used in a variable?

A. It indicates a local variable

B. It refers to a top scope variable

C. It signifies a class variable

D. It denotes a variable in a specific module

In Puppet, the use of the $:: prefix before a variable name signifies that the variable is being referenced from the top scope. The top scope is the global context in which Puppet code is evaluated, allowing the variable to be accessed from anywhere within the manifest, regardless of where it is declared. This is particularly useful when you need to ensure that certain global configurations or values are accessible throughout your Puppet code without being restricted to the context in which they were defined. For example, if you have declared a variable at the top scope and you need to reference it from within a class or a defined type, prefixing it with $:: ensures that the correct variable is accessed, rather than any local or scoped variables that may have a similar name. Other options refer to different types of variable scopes. For instance, local variables would not include the $:: prefix; they would be implicitly accessible only within a specific scope such as a class or defined type. Similarly, class variables would typically be referenced with $classname::varname, where "classname" is the name of the class, indicating they are scoped to that particular class. Module-specific variables would not require the $:: syntax as they are typically referenced directly. Thus, $:: uniquely identifies variables within

4. Where will the autoloader look for the class definition for ssh::server::keys?

A. /etc/puppetlabs/puppet/modules/ssh/modules/keys.pp

B. /etc/puppetlabs/puppet/modules/ssh/manifests/key.pp

C. /etc/puppetlabs/puppet/modules/ssh/manifests/server/keys.pp

D. /opt/puppetlabs/puppet/modules/ssh/manifests/server/keys.pp

The autoloader in Puppet is responsible for locating class definitions in specific directories based on the naming conventions used in the Puppet module structure. When looking for the class definition for `ssh::server::keys`, the autoloader follows a pattern that reflects the module's hierarchy. In this case, the autoloader seeks to find the `keys` class within the `server` namespace of the `ssh` module. According to Puppet's conventions, namespaces correspond to directory structures in the module's `manifests` directory. Therefore, it will look for a file named `keys.pp` in a subdirectory that matches the namespace path. The correct location would be `/etc/puppetlabs/puppet/modules/ssh/manifests/server/keys.pp`. This path accurately reflects both the module's base directory (`ssh`), the `server` namespace, as well as the specific file name for the `keys` class. Other options either suggest incorrect file names or incorrect directory structures that don't align with Puppet's expectations for loading classes. For instance, a directory implying a `modules` directory or a file with a different structure would not conform to the autoloader's lookup algorithm for class definitions. Hence, understanding the organizational structure

5. A class with the same name as the module goes into which file?

A. lib/self.pp

B. self.pp

C. manifests/class

D. manifests/init.pp

In Puppet, a class that shares the same name as the module typically resides in the `manifests/init.pp` file. This file serves as the main entry point for a Puppet module. When a module is loaded, Puppet looks for its classes in this specific location. Since the class name and module name align, placing the class definition within `init.pp` ensures that it can be invoked directly by its module name. The structure of Puppet modules is such that the `manifests` directory is intended for defining classes and resources, with `init.pp` often being used to define the primary class of the module. By convention, this design allows for clear organization of code and simplifies the loading and calling of classes. In contrast, `lib/self.pp` and `manifests/class` are not standard locations for defining a class that directly matches the module name. `self.pp` is not recognized as a conventional path in the Puppet module hierarchy. Thus, the organization of your classes in modules is structured to facilitate ease of use and prevent any ambiguity about where to find a class definition.

6. Which of the following are valid parameters for the Puppet module install command?

A. --version, --modulepath, --force, --debug

B. --name, --ensure, --version, --force

C. --installpath, --name, --modulepath, --debug

D. --force, --timeout, --debug, --version

The correct answer identifies valid parameters that can be used with the Puppet module install command. The parameters listed in this answer, which include --version, --modulepath, --force, and --debug, are indeed recognized by the Puppet command infrastructure and serve distinct functions when executing the module installation process. - The --version parameter allows users to specify a particular version of the module they wish to install, ensuring they can control the versioning aspect of their setup. - The --modulepath parameter lets users define a custom directory where Puppet should look for modules. This is useful when managing modules across different environments or particular organizational structures. - The --force parameter allows the installation process to override any existing module, making it possible to replace a currently installed version with a new one without removing the old version first. - The --debug parameter enables detailed logging of the command execution, which is advantageous for troubleshooting and understanding what the Puppet tool is doing under the hood. Each of these parameters supports efficient module management in Puppet, contributing to better control over configurations and deployment practices. The other choices include parameters that do not accurately reflect those recognized by the Puppet module install command or combine correct and incorrect parameters, which diminishes their validity. Therefore, this choice stands out as

7. True or False: The resource type exec has the ability to listen for relationship notifications such as subscribe and notify.

A. True

B. False

C. Only works with certain resource types

D. Exec can only send notifications

The correct response is that the resource type `exec` has the ability to listen for relationship notifications such as `subscribe` and `notify`. In Puppet, the `exec` resource type is used to execute commands on the system. This type can be tied into the notification system, meaning it can be configured to react to changes in other resources. For example, if a file or service changes state, an `exec` resource can be set to `notify` that it should run a command (thus triggering the execution) or it can `subscribe` to changes, meaning it would run when the specified resource changes. This capability allows for greater flexibility in managing configurations, ensuring that certain commands only execute when related resources are modified. This interaction with notifications is fundamental to Puppet's design, enabling orchestration of resource changes in an efficient manner, ensuring that certain actions are taken as required when specific changes occur. Hence, understanding the role of `exec` in this context is crucial for managing Puppet manifests effectively.

8. In Puppet, what is the primary role of the 'node' declaration?

A. To create modules

B. To define resource collections for specific nodes

C. To set parameters globally

D. To manage data paths

The primary role of the 'node' declaration in Puppet is to define resource collections for specific nodes. This declaration acts as a way to specify which resources, classes, and configurations should apply to a particular node (or set of nodes). The node block allows you to tailor the configuration management to the specific needs and characteristics of individual systems. By doing this, Puppet can apply unique settings or resources to each node, ensuring that the configurations align with the specific environment and requirements of that system. When you define a node declaration, you can reference classes and define resources that should be applied only to that node, enabling a highly customized and modular approach to configuration management. This function is crucial in larger environments where resources and requirements can vary significantly from one machine to another. The ability to manage these specificities within node definitions leads to more efficient and effective system management.

9. True or False: Resource Collectors can only search on attributes which are present in the manifests and cannot read the state of the target system.

A. True

B. False

C. Only partially true

D. Not applicable

Resource collectors in Puppet are designed to manage and oversee the resources defined within manifests. However, the assertion that they can only search on attributes present in manifests and cannot read the state of the target system is not accurate. Resource collectors can indeed utilize the state of the target system to make decisions, meaning they can query existing attributes that are currently set on the system. This capability allows Puppet to function more effectively by tailoring its actions according to the actual state of system resources rather than relying solely on predefined manifests. Thus, the true nature of resource collectors allows them to operate interactively with both the defined states in the manifests and the actual system state, enabling a more dynamic and responsive configuration management process. This interaction is what empowers Puppet to ensure that systems are converged towards the desired state as defined in the Puppet code while still being aware of the current state of the system.

10. Which designation is not a valid directory for storing configuration components in a Puppet module?

A. Templates

B. Files

C. Settings

D. Manifests

In Puppet modules, several predefined directories are established for organizing different types of configuration components, each serving a specific purpose. The "Templates" directory is intended for custom templates used with the `template` function to generate configuration files. The "Files" directory is where static files that the Puppet agent can deploy are placed, allowing for easy access to resources. The "Manifests" directory is essential for storing the Puppet manifests, which contain the definitions of resources and the desired state of the system. The designation "Settings," however, does not correspond to any standardized directory structure within a Puppet module. This makes it an invalid choice for storing configuration components, as Puppet's established module directory structure relies exclusively on the other listed directories. Understanding the purpose and function of each specific directory is crucial for effectively organizing a Puppet module and ensuring that the Puppet agent can find and utilize the necessary configuration files correctly.