Puppet Resources

"Resources are the fundamental unit for modeling system configurations. They're like building lacks. Think 'Lego'. Each resource describes some aspect of a system, like a service that must be running, or a package that must be installed. The block of Puppet code that describes a resource is called a 'resource declaration'."

"Resources can be combined together to represent the desired configuration of your system. And order doesn't matter. Puppet resources are written using the declarative modeling language. This is a simple, easy to understand, and easy to write language. Let's take a look at an example."

"This is the syntax for a resource declaration. In this example we're defining a user resource. Every resource follows the same syntax guide. A resource includes the resource type. Curly braces define the resource block. A resource has a title, separated from the body of the resource with a colon. And then the body consists of attribute and value pairs."

"When defining a resource, the type and title must be unique for a given node. Use alphanumerics for the value of attributes. And, you should quote your strings. Every attribute and value pair should end in a comma. This includes the last attribute value pair, which will make maintenance easier."

"Any resource is similar to a class of related things. Every file has a path and an owner. Every user has a name, a user ID, and a group. Which is to say, similar resources can be grouped into types. This is the user resource type."

"Furthermore, the most important resources of an attribute type are usually conceptually identical across operating systems regardless of how the implementations differ. That is, the description of a resource can be abstracted away from its implementation."

"These two insights form Puppet's resource abstraction layer, RAL. The RAL splits resources into types, high-level models, and providers - platform specific implementations - and lets you describe resources in a way that can apply to any system. Each resource - file, package, service, user - has one or more providers. Providers are the interface between the underlying OS and the resource types."

"Puppet uses the RAL to both read and modify the state of resources on a system. Since it's a declarative system, Puppet starts with an understanding of what state a resource should have. To sync the resource, it uses the RAL to query the current state, compare that against the desired state, and then use the RAL again to make any necessary changes."

"Puppet has a number of built-in types, and new native types can be distributed with modules. Puppet's core types include notify, file, package, service, exec, cron, user, and group."

"A resource declaration adds a resource to the catalog, and tells Puppet to manage that resource's state. When Puppet applies the compiled catalog, it will read the actual state of the resource on the target system, compare the actual state to the desired state, and, if necessary, change the system to enforce the desired state."

"Puppet does not allow you to declare the same resource twice. This is to prevent multiple conflicting values from being declared for the same attribute. Puppet uses the title and name to identify duplicate resources. If either of these is duplicated within a given resource type, the compilation will fail."

"Puppet resource is a command line tool for inspecting resources on your system and interacts directly with the RAL. The command format is **puppet**, **resource**, the type of resource you're interested in inspecting, and the title of that resource."

"In the example that we've been looking at, we would enter **puppet resource user gary** on the command line. It would return information about that resource. You'll have a chance to try this command in the exercises following this video."

"Executing Puppet resource, while providing a resource type, will query all known instances of that resource on the system. You can also try this command in the exercises following the video."

"As you can see, using the declarative modeling language, you can define exactly the way that you want your system to look, using resources. Moving forward, you'll begin to see that classes define a collection of resources that are managed together as a single unit."

"As it's been mentioned before, the core of the Puppet language is the resource declaration. A resource declaration describes the desired state for one resource. Multiple resource declarations can be combined together into classes. Manifest, or the Puppet program, can also use these collections of resources along with various other kinds of logic, such as conditional statements, functions, etcetera. These topics will be covered in later classes."

"To complete this course, complete the short quiz and work through any exercises that are found immediately below this video."