Puppet Infrastructure Automation Quiz (100 Questions)

DevOps Learning Module

This quiz covers fundamental concepts related to Puppet infrastructure automation. Choose the best answer for each question.

1. What is the main executable that runs on a Puppet-managed node to retrieve and apply configurations?

   A. `puppet-master`

   B. `puppet apply`

   C. `puppet agent`

   D. `facter`

   Answer: C
   Explanation: The `puppet agent` service (or command) runs on a node. It communicates with the Puppet master, sends facts, retrieves a catalog, and applies that catalog to the node.

2. What is a "Manifest" in Puppet?

   A. A file containing node-specific data (like `ohai` data).

   B. A file with the `.pp` extension that contains Puppet code (resources, classes, etc.).

   C. The compiled configuration that a node receives from the master.

   D. A file defining module metadata.

   Answer: B
   Explanation: Manifests are the files, ending in `.pp` (for Puppet Program), where you write your Puppet code to define the desired state of your resources.

3. Which Puppet component gathers data (facts) about a node, such as its OS, IP address, and hostname?

   A. `puppetdb`

   B. `hiera`

   C. `facter`

   D. `puppet agent`

   Answer: C
   Explanation: `facter` is Puppet's system inventory tool. It runs on the agent and collects "facts" about the node, which are then sent to the master and available as variables in manifests.

4. What is a "Catalog" in Puppet?

   A. A collection of modules on the Puppet Forge.

   B. A Ruby file containing resource definitions.

   C. A JSON document compiled by the master that describes the desired state for a specific node.

   D. The main manifest file (`site.pp`).

   Answer: C
   Explanation: The Puppet master compiles all the relevant manifests, modules, and Hiera data for a specific node into a single JSON document called the "Catalog." The agent then applies this catalog.

5. What is the default data lookup system in Puppet used for separating data from code?

   A. Data Bags

   B. Hiera

   C. Facter

   D. PuppetDB

   Answer: B
   Explanation: Hiera is Puppet's built-in key-value lookup tool. It allows you to store configuration data (in YAML or JSON) in a hierarchical structure, separate from your Puppet code.

6. How do you declare a `package` resource in Puppet to ensure `nginx` is installed?

   A. `package { 'nginx': ensure => 'installed' }`

   B. `package 'nginx' do ensure 'installed' end`

   C. `Package('nginx', 'installed')`

   D. `resource 'package', 'nginx', ensure: 'installed'`

   Answer: A
   Explanation: Puppet's declaration syntax is `resource_type { 'title': attribute => value, }`. `ensure => 'installed'` is the standard way to make sure a package exists.

7. What is a "Puppet Module"?

   A. A single `.pp` file.

   B. A JSON file defining node attributes.

   C. A self-contained, reusable bundle of manifests, files, templates, and data.

   D. A plugin for the Puppet master.

   Answer: C
   Explanation: Modules are the primary way to organize and reuse Puppet code. A module typically manages a specific piece of software (like `ntp` or `apache`).

8. What is the "Puppet Forge"?

   A. The component that compiles catalogs.

   B. A community-driven repository of publicly available Puppet modules.

   C. The testing framework for Puppet.

   D. A tool for storing secret data.

   Answer: B
   Explanation: The Puppet Forge is the official repository for finding, sharing, and installing modules created by the Puppet community and Puppet, Inc.

9. How do you ensure a service (like `nginx`) is running and enabled to start on boot?

   A. `service { 'nginx': ensure => 'running', enabled => 'true' }`

   B. `service { 'nginx': ensure => ['running', 'enabled'] }`

   C. `service { 'nginx': ensure => 'running', enable => true }`

   D. `service { 'nginx': state => 'running', boot => true }`

   Answer: C
   Explanation: The `service` resource uses the `ensure` attribute for the running state (`'running'` or `'stopped'`) and the `enable` attribute (`true` or `false`) for boot-time behavior.

10. What does idempotency mean in Puppet?

    A. A manifest can be run on any operating system.

    B. A catalog can be applied multiple times, but it will only make changes if the node is not in the desired state.

    C. The Puppet agent always runs, even if not needed.

    D. The code must be written in Ruby.

    Answer: B
    Explanation: Idempotency is a core principle. It means that applying a catalog is a "convergent" operation. You declare the desired *state*, and Puppet only takes action if the *current state* is different.

11. What command-line tool is used to install modules from the Puppet Forge?

    A. `puppet module install`

    B. `puppet install module`

    C. `puppet forge get`

    D. `puppet module get`

    Answer: A
    Explanation: The `puppet module install` command (e.g., `puppet module install puppetlabs-ntp`) is used to download and install modules into your module path.

12. Which file defines a module's metadata and dependencies?

    A. `manifests/init.pp`

    B. `Puppetfile`

    C. `metadata.json`

    D. `Modulefile`

    Answer: C
    Explanation: The `metadata.json` file is the standard way to declare a module's name, version, author, and dependencies on other modules from the Forge.

13. What is the "main manifest"?

    A. The `manifests/init.pp` file in a module.

    B. The primary entry point for Puppet, defined on the master, often called `site.pp`.

    C. A manifest that contains only `package` resources.

    D. The catalog file.

    Answer: B
    Explanation: The main manifest (often `site.pp`) is the top-level file the Puppet master uses to start compiling a catalog. It typically contains node definitions.

14. How do you use a class (e.g., `ntp`) from a module in your `site.pp`?

    A. `use 'ntp'`

    B. `import 'ntp'`

    C. `include ntp`

    D. `require ntp`

    Answer: C
    Explanation: The `include` function is the standard way to declare a class. It ensures the class is added to the catalog and applied. It's idempotent (a class can only be included once).

15. What is the difference between `include ntp` and `class { 'ntp': }`?

    A. There is no difference.

    B. `include` is for roles; `class` is for nodes.

    C. `include` cannot pass parameters; `class { }` (resource-like declaration) can.

    D. `class { }` is the old, deprecated syntax.

    Answer: C
    Explanation: `include` is for simple class inclusion. The resource-like `class { 'ntp': ... }` syntax allows you to pass parameters to the class, overriding its Hiera defaults.

16. What is a "metaparameter" in Puppet?

    A. A parameter that is defined in `metadata.json`.

B. A parameter that is only available for `package` resources.

C. A parameter (like `require` or `notify`) that is available for all resource types.

D. A parameter that stores Hiera data.

Answer: C
Explanation: Metaparameters are not specific to any one resource type. They handle relationships between resources. Common examples are `require`, `before`, `notify`, and `subscribe`.

17. What does `require => Package['nginx']` mean in a `service` resource?

A. The service will restart when the package changes.

B. The service will be managed *before* the package.

C. The service will *only* be managed if the package is *not* installed.

D. The service will be managed *after* the package has been successfully managed.

Answer: D
Explanation: `require` creates a dependency. It ensures that the `Package['nginx']` resource (and any of its dependencies) are applied *before* the `service` resource.

18. What does `notify => Service['nginx']` mean in a `file` resource?

A. The file will be updated if the service restarts.

B. The file will only be created if the service is running.

C. If the file's content changes, it will send a notification to the `Service['nginx']` resource, typically triggering a restart.

D. The file will be created *after* the service.

Answer: C
Explanation: `notify` creates a subscription. If the `file` resource changes (i.e., it's not "up-to-date"), it "notifies" the `service` resource, which will then "refresh" (e.g., restart).

19. What is the opposite of `notify`?

A. `require`

B. `before`

C. `listen`

D. `subscribe`

Answer: D
Explanation: `notify` is a "push" relationship (from the file *to* the service). `subscribe` is a "pull" relationship (e.g., `Service['nginx'] { subscribe => File['/etc/nginx.conf'] }`).

20. How do you create a file with specific content directly in a manifest?

A. `file { '/tmp/foo': content => 'My content' }`

B. `file { '/tmp/foo': source => 'My content' }`

C. `file { '/tmp/foo': text => 'My content' }`

D. `file { '/tmp/foo': ensure => 'My content' }`

Answer: A
Explanation: The `content` attribute is used to specify the exact content of a file directly, as opposed to `source`, which points to a file in the module.

21. What is the `source` attribute of a `file` resource used for?

A. To specify the content of the file.

B. To point to a file in the module's `files/` directory.

C. To specify a URL to download the file from.

D. To point to an `.erb` template file.

Answer: B
Explanation: The `source` attribute uses a Puppet-specific URL (e.g., `'puppet:///modules/mymodule/my`
to copy a static file from the module's `files/` directory.

22. What is the `content` attribute of a `file` resource used for, when managing a dynamic file?

A. To specify the content of the file.

B. To point to a file in the module's `files/` directory.

C. To point to a remote URL.

D. To specify a template file using the `template()` or `epp()` function.

Answer: D
Explanation: To manage a dynamic file, you set the `content` attribute to the *output* of a function like `template('mymodule/mytemplate.erb')` (for EPP templates, `epp()`).

23. What is an "EPP" template?

A. An "Embedded Perl" template.

B. An "Embedded Puppet" template, a modern, Puppet-native alternative to ERB.

C. A "External Policy" template.

D. A "Encrypted Puppet" template.

Answer: B
Explanation: EPP (Embedded Puppet) templates are similar to ERB but use Puppet's language syntax (e.g., `<%= $variable %>`) instead of Ruby's. They are generally preferred over ERB.

24. What is the `exec` resource used for?

A. To execute an arbitrary command.

B. To install a package.

C. To define a class.

D. To execute a Ruby block.

Answer: A
Explanation: Much like Chef's `execute`, Puppet's `exec` resource is a general-purpose tool for running shell commands. It must be made idempotent.

25. How do you make an `exec` resource idempotent?

    A. `exec { 'mycommand': idempotent => true }`

    B. `exec { 'mycommand': onlyif => 'test command' }`

    C. `exec { 'mycommand': unless => 'test command' }`

    D. Both B and C.

Answer: D
Explanation: `onlyif` runs the `exec` *only if* its test command returns 0 (success). `unless` runs the `exec` *unless* its test command returns 0 (success). These are guards.

26. What is the "Roles and Profiles" pattern?

    A. A deprecated method of managing nodes.

    B. A way to buy Puppet Enterprise.

    C. A best-practice design pattern for organizing Puppet code.

    D. A feature of Hiera.

Answer: C
Explanation: It's a pattern where "Profiles" (e.g., `profile::webserver`) combine modules to build a service, and "Roles" (e.g., `role::webserver::production`) combine profiles to define a complete node.

27. What is "PuppetDB"?

    A. A key-value store that replaces Hiera.

    B. A database that stores all catalogs, facts, and reports from all nodes.

    C. A test framework.

    D. The main manifest file.

Answer: B
Explanation: PuppetDB is a backend database that collects and stores all data from your Puppet infrastructure. It enables powerful queries, like "which nodes are running an old version of OpenSSL?"

28. What is `puppet apply`?

    A. A command to apply for a job at Puppet.

    B. A command to apply a local manifest file, "server-less" (like `chef-apply`).

    C. A command to apply a hotfix to the Puppet master.

    D. The same as `puppet agent`.

Answer: B
Explanation: `puppet apply` compiles and applies a single manifest file (or directory) locally. It does not require a Puppet master and is useful for testing or simple, single-node setups.

29. What is the "noop" mode in Puppet?

    A. A mode that does nothing.

    B. A "dry-run" mode (like `-why-run` in Chef) that simulates a run and reports what *would* change.

    C. A mode that only runs on one node.

    D. A mode that skips all `exec` resources.

    Answer: B
    Explanation: Running `puppet agent -t -noop` (or just `-noop`) puts the agent in "no-operation" mode. It gets a catalog and simulates the run, but makes no changes.

30. What does the `~>` (tilde-arrow) operator do?

    A. It is a "chaining" operator, equivalent to `require`.

    B. It is a "notification" operator, equivalent to `notify`.

    C. It is an "assignment" operator.

    D. It is a "Hiera lookup" operator.

    Answer: B
    Explanation: The "wavy-rocket" or "tilde-arrow" operator is a shorthand for notification. `File['a'] ~> Service['b']` is the same as `File['a'] { notify => Service['b'] }`.

31. What is a "Defined Type" in Puppet?

    A. A custom resource type that can be instantiated multiple times (e.g., `apache::vhost`).

    B. A class.

    C. A built-in resource type like `package`.

    D. A Hiera data type.

    Answer: A
    Explanation: A Defined Type (or "define") is a block of Puppet code that you can reuse multiple times with different parameters. It's like a class, but it can be declared more than once.

32. What is the main configuration file for the Puppet master?

    A. `puppet.conf`

    B. `master.conf`

    C. `site.pp`

    D. `puppetdb.conf`

Answer: A
Explanation: `puppet.conf` (usually in `/etc/puppetlabs/puppet/`) is the main configuration file for all Puppet components, including the master, agent, and apply.

33. How do you reference a "fact" (like `osfamily`) in a manifest?

    A. `$::osfamily`

    B. `$facts['osfamily']`

    C. `ohai['osfamily']`

    D. `facter('osfamily')`

Answer: A
Explanation: Top-scope variables (facts) are accessed using the `$::` prefix, such as `$::operatingsystem` or `$::ipaddress`. (Modern Puppet also uses `$facts['...']`).

34. What is the purpose of the `file` resource's `ensure` attribute?

    A. `ensure => 'installed'`

    B. `ensure => 'file'` or `'directory'` or `'absent'`

    C. `ensure => 'content'`

    D. `ensure => 'mode'`

Answer: B
Explanation: The `ensure` attribute for a `file` resource defines its *type* or *state*. `'file'` ensures it's a file, `'directory'` a directory, and `'absent'` ensures it is deleted.

35. What is "Puppet Bolt"?

    A. An agent-less, task-based tool for running ad-hoc commands and scripts over SSH or WinRM.

    B. A part of the Puppet master.

    C. A tool for building modules.

    D. A security and compliance tool.

Answer: A
Explanation: Bolt is Puppet's answer to ad-hoc task running (like Ansible). It's agentless and designed for tasks, scripts, and plans, not for continuous state enforcement.

36. What is a `Puppetfile`?

    A. The main manifest.

    B. A file used by `r10k` or `code-manager` to define and list module dependencies.

    C. A file defining Hiera data.

    D. A file containing a list of all nodes.

Answer: B
Explanation: A `Puppetfile` is used to declare which modules (and which versions, from

Git or the Forge) should be in an environment. `r10k` reads this file to deploy the modules.

37. What is `r10k`?

    A. A Puppet master component.

    B. A tool for managing and deploying Puppet environments and modules from Git.

    C. A reporting tool.

    D. A database.

    Answer: B
    Explanation: `r10k` is a tool that reads a `Puppetfile` and creates environments on your Puppet master, populating each with the correct module versions from Git or the Forge.

38. What is "code-manager"?

    A. A text editor for Puppet code.

    B. The Puppet Enterprise (PE) version of `r10k`, used for code deployment.

    C. A tool for refactoring manifests.

    D. A module for managing source code.

    Answer: B
    Explanation: `code-manager` is the built-in, automated version of `r10k` in Puppet Enterprise. It automatically deploys code to masters when triggered by a Git webhook.

39. How do you set file permissions to `0644` on a `file` resource?

    A. `file { '/tmp/foo': permissions => '0644' }`

    B. `file { '/tmp/foo': mode => '0644' }`

    C. `file { '/tmp/foo': chmod => '0644' }`

    D. `file { '/tmp/foo': access => '0644' }`

    Answer: B
    Explanation: The `mode` attribute is used to define the file system permissions for a file or directory.

40. What does the `user` resource manage?

    A. Puppet Enterprise console users.

    B. Local user accounts on a node.

    C. Hiera data.

    D. Users in PuppetDB.

    Answer: B
    Explanation: The `user` resource is used to create, delete, and manage local user accounts on the agent node.

41. What is the purpose of the `group` resource?

    A. To group nodes in Puppet Enterprise.

    B. To manage local groups on a node.

    C. To group resources in a manifest.

    D. To define a role.

    Answer: B
    Explanation: Similar to the `user` resource, the `group` resource manages local groups on the agent node.

42. How do you set a variable in a manifest?

    A. `let $my_var = 'value'`

    B. `var $my_var = 'value'`

    C. `$my_var = 'value'`

    D. `set $my_var = 'value'`

    Answer: C
    Explanation: Puppet uses a simple dollar-sign (`$`) prefix for variable assignment, e.g., `$package_name = 'nginx'`.

43. What is the `cron` resource used for?

    A. To manage cron jobs on a node.

    B. To schedule Puppet agent runs.

    C. To time how long a catalog compilation takes.

    D. To run an ad-hoc command.

    Answer: A
    Explanation: The `cron` resource is an abstraction for managing entries in the system's cron table, ensuring scheduled tasks are present, absent, or have the correct command.

44. What is a "node definition" in `site.pp`?

    A. `node 'web1.example.com' { ... }`

    B. `define node 'web1.example.com' { ... }`

    C. `host 'web1.example.com' { ... }`

    D. `agent 'web1.example.com' { ... }`

    Answer: A
    Explanation: A node definition is a block of code that applies *only* to a node with a matching name (certname). It's where you would typically `include` roles for that node.

45. What is the `default` node definition?

    A. The definition for the Puppet master.

B. A node definition that applies to *any* node that doesn't have a more-specific node definition.

C. A node that is not in any environment.

D. A deprecated feature.

Answer: B
Explanation: The `node default { ... }` block acts as a fallback. Any node that connects to the master and doesn't match a named node definition will get the code inside `default`.

46. What is "data-binding" in Puppet?

A. The connection between an agent and a master.

B. The automatic lookup of class parameters from Hiera.

C. The relationship between a `package` and a `service`.

D. A security feature.

Answer: B
Explanation: Data-binding is the feature where Puppet automatically looks up values for class parameters in Hiera. For `class ntp($servers)`, Puppet will automatically search Hiera for `ntp::servers`.

47. How is the Hiera hierarchy typically defined?

A. In `hiera.yaml`

B. In `site.pp`

C. In `puppet.conf`

D. In `metadata.json`

Answer: A
Explanation: The `hiera.yaml` file defines the "hierarchy" of data files (e.g., look in `nodes/%{certname}.yaml` first, then `environments/%{environment}.yaml`, then `common.yaml`).

48. What does `ensure => absent` do on a `package` resource?

A. Installs the package.

B. Upgrades the package.

C. Ensures the package is uninstalled.

D. Purges the package and its configuration files.

Answer: C
Explanation: `ensure => 'absent'` is the idempotent way to make sure a package is not installed on the system.

49. What is the difference between `ensure => absent` and `ensure => purged` for a `package`?

A. There is no difference.

B. `absent` only works for `apt`.

C. `purged` also removes system-wide configuration files associated with the package.

D. `purged` is for services, `absent` is for packages.

Answer: C
Explanation: `absent` typically just uninstalls the package (e.g., `apt-get remove`). `purged` will also remove configuration files (e.g., `apt-get purge`).

50. What is the `file` resource's `recurse => true` attribute used for?

A. To create parent directories.

B. To manage all files *within* a directory.

C. To delete a directory and its contents.

D. To follow symbolic links.

Answer: B
Explanation: When used on a directory, `recurse => true` tells Puppet to manage the permissions, owner, and group of all files and subdirectories *inside* that directory.

51. What is the "environment" in Puppet?

A. A collection of Hiera data.

B. An isolated set of manifests and modules, typically corresponding to a Git branch.

C. A security setting.

D. The `/etc/puppetlabs/puppet` directory.

Answer: B
Explanation: Environments (like "production", "development") allow you to test code changes (new modules, new manifests) on a subset of nodes before promoting them to production.

52. How does an agent know which environment to use?

A. It is set in Hiera.

B. It is set in the agent's `puppet.conf` file.

C. It is defined in the `site.pp` file.

D. The master assigns it based on the node's role.

Answer: B
Explanation: The agent's `puppet.conf` file contains an `environment = ...` setting, which tells the master which environment's manifests and modules to use when compiling its catalog.

53. What is the `file` resource's `owner` attribute used for?

A. To set the file's user owner.

B. To set the file's group owner.

C. To set the file's permissions.

D. To define who can read the file.

Answer: A
Explanation: The `owner` attribute specifies the user (by name or UID) that should own the file.

54. What is the `file` resource's `group` attribute used for?

A. To set the file's user owner.

B. To set the file's group owner.

C. To group files together.

D. To define which group can execute the file.

Answer: B
Explanation: The `group` attribute specifies the group (by name or GID) that should own the file.

55. What is the purpose of the `mount` resource?

A. To mount an ISO file.

B. To manage filesystem mount points (e.g., in `/etc/fstab`).

C. To mount a remote directory via SSH.

D. To install `nfs-utils`.

Answer: B
Explanation: The `mount` resource is an abstraction for managing mount points. It can ensure an entry is in `/etc/fstab` and that the filesystem is mounted.

56. What is the `host` resource used for?

A. To manage entries in the `/etc/hosts` file.

B. To define a new node in `site.pp`.

C. To ping a remote host.

D. To configure the node's hostname.

Answer: A
Explanation: The `host` resource provides an idempotent way to manage entries in the local `/etc/hosts` file.

57. What is the default log level for the `puppet agent`?

A. `debug`

B. `info`

C. `notice`

D. `warn`

Answer: C
Explanation: The default log level is `notice`, which shows summaries of changes. `info` is more verbose, and `debug` is extremely verbose.

58. What does `puppet agent -t` do?

    A. Runs the agent in test (noop) mode.

    B. Runs the agent once, in the foreground, with verbose logging.

    C. Runs the agent's built-in tests.

    D. Triggers a catalog compilation on the master.

    Answer: B
    Explanation: The `-t` flag (or `-test`) is the standard way to run the agent manually. It runs once, shows all log messages, and exits. It is *not* the same as `-noop`.

59. What is "Puppet lint"?

    A. A tool that cleans up old node data.

    B. A tool that checks Puppet code for style guide violations and potential errors.

    C. The cache directory for Puppet.

    D. A component of PuppetDB.

    Answer: B
    Explanation: `puppet-lint` is a static analysis tool that checks your `.pp` files against the official Puppet style guide, helping to maintain clean and consistent code.

60. What is "Onceover"?

    A. A tool that runs `puppet agent -t` on all nodes.

    B. A testing tool that runs `puppet-lint`, `rspec-puppet`, and `beaker` tests.

    C. A code-coverage tool.

    D. A Hiera encryption tool.

    Answer: B
    Explanation: Onceover is a comprehensive testing framework that simplifies the process of running all the common types of tests (lint, syntax, unit, integration) for your control-repo or modules.

61. What is "rspec-puppet"?

    A. A tool for running ad-hoc commands.

    B. A tool for "unit testing" Puppet manifests.

    C. A tool for "integration testing" (like Test Kitchen).

    D. A tool for linting.

Answer: B
Explanation: `rspec-puppet` is a unit testing framework. It allows you to test your compiled catalogs in isolation, without ever creating a real node.

62. What is "Beaker"?

    A. A unit testing tool.

    B. An "acceptance testing" (integration testing) framework that provisions nodes and applies Puppet code.

    C. A Hiera backend.

    D. A replacement for `facter`.

    Answer: B
    Explanation: Beaker is Puppet's integration testing framework (similar in role to Test Kitchen). It spins up nodes, runs Puppet, and then lets you run tests to verify the node's state.

63. What is the purpose of the `stage` metaparameter?

    A. To define which environment a class is in.

    B. To create large-scale "run-stages" (like `pre`, `main`, `post`) to order groups of classes.

    C. To define a class as "beta" or "production".

    D. To stage a change before applying it.

    Answer: B
    Explanation: Stages allow you to create broad ordering. For example, you can put all your `apt_update` classes in a `pre` stage to ensure they run before the `main` stage.

64. In Puppet, what is a "class"?

    A. A single resource.

    B. A collection of resources, logic, and variables grouped together as a unit (e.g., `class ntp { ... }`).

    C. A type of node.

    D. A Hiera data file.

    Answer: B
    Explanation: A class is the fundamental unit of code in a module. For example, the `ntp` module has an `ntp` class that manages the `ntp` package, file, and service.

65. What is the "autoloader" in Puppet?

    A. A tool that automatically runs `puppet agent`.

    B. The system that automatically loads classes, defined types, and functions from modules.

    C. The tool that installs module dependencies.

    D. The Hiera data-binding system.

Answer: B
Explanation: When you `include ntp`, you don't have to specify *where* that class is. The autoloader knows to look in the module path for a module named `ntp` and load its `manifests/init.pp`.

66. What is the `file` resource's `purge => true` attribute used for?

    A. To delete the file.

    B. To purge the package associated with the file.

    C. When set on a directory, to delete any files in that directory that are *not* managed by Puppet.

    D. To clear the content of the file.

Answer: C
Explanation: `purge` is a powerful and dangerous attribute. It's used to ensure that a directory contains *only* the files Puppet knows about, deleting all others.

67. What is the "control repository" (or "control-repo")?

    A. A Git repository that contains the `Puppetfile` and `site.pp` to define the entire infrastructure.

    B. The Puppet master's `/etc/puppetlabs/puppet` directory.

    C. A repository of all Puppet modules.

    D. The PuppetDB database.

Answer: A
Explanation: The control-repo is the "single source of truth" for your Puppet infrastructure. It defines the environments (as branches) and the modules (in the `Puppetfile`).

68. What is the standard name for the main class in a module (e.g., the `ntp` module)?

    A. `manifests/default.pp`

    B. `manifests/main.pp`

    C. `manifests/ntp.pp`

    D. `manifests/init.pp`

Answer: D
Explanation: The `manifests/init.pp` file is special. It must contain the class that has the *same name* as the module (e.g., `class ntp { ... }`).

69. How do you write a conditional `if` statement in Puppet?

    A. `if ($::osfamily == 'RedHat') { ... }`

    B. `if $::osfamily == 'RedHat' then ... end`

    C. `test { $::osfamily == 'RedHat': ... }`

    D. `when $::osfamily == 'RedHat' { ... }`

Answer: A
Explanation: Puppet's language includes standard `if`, `elsif`, and `else` conditional statements, which use brace-delimited code blocks.

70. What is a "selector" in Puppet?

    A. A way to select which nodes to run on.

    B. A conditional statement that returns a value (like a ternary operator).

    C. A metaparameter for selecting files.

    D. A Hiera lookup function.

    Answer: B
    Explanation: A selector is like a switch statement that returns a value. Example: `$pkg = $::osfamily ? { 'RedHat' => 'httpd', 'Debian' => 'apache2' }`.

71. What does the `puppet resource` command do?

    A. It lists all available resource types.

    B. It generates Puppet code by inspecting the current state of a resource on the system.

    C. It runs a single resource from a manifest.

    D. It deletes a resource.

    Answer: B
    Explanation: `puppet resource` is a powerful tool for introspection. For example, `puppet resource user root` will show you the current state of the `root` user in Puppet code format.

72. What is the `augeas` resource used for?

    A. To manage user accounts.

    B. To parse and modify configuration files in a structured, idempotent way.

    C. To install packages.

    D. To manage a "house-cleaning" (Augean stables) task.

    Answer: B
    Explanation: Augeas is a tool that parses text-based config files into a tree. The `augeas` resource lets you make very specific, idempotent changes to files that don't have a dedicated resource type.

73. What is the purpose of the `tag` metaparameter?

    A. To tag a resource with a keyword, which can be used to filter logs or limit runs.

    B. To tag a module for upload to the Forge.

    C. To tag a Git commit for `r10k`.

    D. To tag a file with an MD5 hash.

Answer: A
Explanation: You can add tags to resources (e.g., `tag => 'webserver'`) and then run the agent with `-tags webserver` to *only* apply resources with that tag.

74. What is a "Puppet Plan"?

    A. A Hiera data file.

    B. A list of manifests to be compiled.

    C. A set of ordered, procedural tasks, written in the Puppet language, and run via Bolt.

    D. A new name for a role.

    Answer: C
    Explanation: Plans are part of Puppet Bolt. While resources are "declarative" (what state), plans are "procedural" (what to do). They are used to coordinate complex, multi-step tasks.

75. What is the `notify` resource in Puppet?

    A. It's a metaparameter, not a resource.

    B. A resource that sends an email.

    C. A resource that prints a message to the agent's log (for debugging).

    D. A resource that restarts a service.

    Answer: C
    Explanation: The `notify` *resource* (not to be confused with the metaparameter) is a simple resource for logging. Example: `notify { 'This is a debug message': }`.

76. What is Hiera Eyaml?

    A. A new version of Hiera.

    B. A command to validate Hiera data.

    C. A "backend" for Hiera that allows you to encrypt individual values inside your YAML files.

    D. A text editor plugin.

    Answer: C
    Explanation: `hiera-eyaml` is a tool for managing secrets. It allows you to encrypt values in your Hiera YAML files so they can be safely committed to Git.

77. What is the "node_name" setting in `puppet.conf`?

    A. The URL of the Puppet master.

    B. The agent's unique identifier, which must match its certificate name (certname).

    C. The environment the node is in.

    D. The "role" of the node.

Answer: B
Explanation: This setting defines the node's identity when it talks to the master. It defaults to the node's FQDN (fully qualified domain name).

78. What is "External Node Classifier" (ENC)?

    A. A tool that automatically writes `site.pp`.

    B. A script or application that tells the master which classes and parameters to apply to a node.

    C. A replacement for Facter.

    D. The Puppet Enterprise console.

    Answer: B
    Explanation: Instead of using `site.pp`, you can use an ENC. The master runs this script, passing it the node's name. The script returns YAML or JSON defining the node's configuration.

79. What is the `puppet parser validate` command used for?

    A. To check the syntax of `.pp` manifest files.

    B. To check Hiera data.

    C. To validate the `puppet.conf` file.

    D. To run unit tests.

    Answer: A
    Explanation: This command is a simple syntax checker. It's a good first step before running `puppet-lint` or committing code.

80. What does the `noop => true` metaparameter do?

    A. It makes the resource run in "dry-run" mode.

    B. It makes the *entire* agent run in "dry-run" mode.

    C. It skips this resource.

    D. It is not a valid metaparameter.

    Answer: A
    Explanation: You can set `noop => true` on a *single* resource to prevent Puppet from making changes, even if the rest of the run is not in `-noop` mode.

81. What is the `puppetca` command used for (on the master)?

    A. To manage the Certificate Authority (CA) and sign or revoke agent certificates.

    B. To create a new catalog.

    C. To apply a catalog.

    D. To clean the agent's certificate.

Answer: A
Explanation: When a new agent connects, it generates a certificate signing request (CSR). You use `puppetca -list` to see requests and `puppetca -sign <node>` to approve them.

82. How does an agent clean its old certificate if it needs to be re-registered?

    A. It happens automatically.

    B. `puppet agent -clean-cert`

    C. By deleting the `ssldir` (e.g., `/etc/puppetlabs/puppet/ssl`).

    D. `puppet cert clean`

    Answer: C
    Explanation: The agent's private key and signed certificate are stored in its `ssldir`. To force a re-registration, you must stop the agent, delete this directory, and restart the agent.

83. What is the default port for the Puppet master?

    A. 8080

    B. 443

    C. 8140

    D. 5000

    Answer: C
    Explanation: The Puppet master daemon listens on port 8140 for agent connections, certificate requests, and catalog retrievals.

84. What is the `runinterval` setting in `puppet.conf`?

    A. The timeout for a run.

    B. The frequency (e.g., `30m`) at which the `puppet agent` daemon runs.

    C. The time between the compile and converge phases.

    D. The network latency.

    Answer: B
    Explanation: When `puppet agent` is running as a daemon, `runinterval` (e.g., `30m` or `1800`) sets how long it sleeps between runs.

85. What is a "Custom Fact"?

    A. A Hiera data point.

    B. A fact that you write yourself (in Ruby or as a structured file) and place in a module.

    C. A variable set in `site.pp`.

    D. A fact that is encrypted.

Answer: B
Explanation: If Facter doesn't provide a piece of information you need (e.g., the application version installed), you can write a custom fact. Puppet will distribute it via modules and `facter` will execute it.

86. Where are custom facts placed within a module?

    A. `lib/facter/`

    B. `facts.d/`

    C. `lib/puppet/facts/`

    D. Both A and B.

    Answer: D
    Explanation: Custom facts can be written as Ruby scripts (in `lib/facter/`) or as structured data files like YAML or text files (in `facts.d/`).

87. What is a "Puppet Function"?

    A. A reusable block of Puppet code (like a `define`)

    B. A reusable block of Ruby or Puppet code that performs logic and returns a value.

    C. A custom fact.

    D. A metaparameter.

    Answer: B
    Explanation: Functions (like `template()`, `merge()`, or `lookup()`) are used during compilation to perform logic and return a value. You can write your own in Ruby or the Puppet language.

88. What is the purpose of `puppet-strings`?

    A. A tool for managing strings in Hiera.

    B. A tool that generates documentation from inline comments in manifests.

    C. A function for string concatenation.

    D. A tool for testing templates.

    Answer: B
    Explanation: `puppet-strings` is the official documentation tool. It parses special comments (like YARD-docs) in your `.pp` files to generate HTML documentation for your modules.

89. What is the `file` resource's `validate_cmd` attribute?

    A. A command to run *before* updating the file to validate the new content.

    B. A command to run *after* updating the file to validate it.

    C. A command to validate the file's permissions.

    D. A command to validate the `file` resource syntax.

Answer: A
Explanation: This is a safety check. For example, on a `sudoers` file, you can set `validate_cmd => '/usr/sbin/visudo -cf %'` (where `%` is the new file) to prevent a bad config.

90. What is the purpose of the `run_stage` setting in `puppet.conf`?

    A. It defines the `main` stage.

    B. It is not a valid setting.

    C. It specifies which "run-stage" the agent should run.

    D. It defines which environment to run.

    Answer: C
    Explanation: By default, an agent runs all stages. You can set `run_stage = pre` on an agent to make it *only* run resources in the `pre` stage. This is a very advanced and rare use case.

91. What is the `hiera()` lookup function used for?

    A. To define the Hiera hierarchy.

    B. To perform an explicit Hiera lookup in a manifest.

    C. To validate Hiera YAML files.

    D. To encrypt Hiera data.

    Answer: B
    Explanation: While automatic data-binding is preferred, you can use `$val = hiera('mykey')` (or the modern `lookup('mykey')`) to manually query Hiera from within your code.

92. What is the `merge_behavior` in Hiera?

    A. How Hiera merges data from different hierarchy levels (e.g., "deep" merge for hashes).

    B. How Puppet merges catalogs.

    C. How Git branches are merged.

    D. A way to combine modules.

    Answer: A
    Explanation: By default, Hiera does a "first-found" lookup. You can specify a `lookup_options` to change this to a "merge" behavior, (e.g., to combine two hashes or build an array).

93. What is the `ssh_authorized_key` resource used for?

    A. To install SSH.

    B. To manage SSH keys in a user's `.ssh/authorized_keys` file.

    C. To manage the SSH server config.

    D. To generate an SSH key.

Answer: B
Explanation: This is a very common resource type. It provides an idempotent way to add or remove specific public keys from a user's `authorized_keys` file.

94. How can you make a resource run *only* if its "noop" mode change is detected?

    A. This is not possible.

    B. `noop_only => true`

    C. `subscribe_noop => true`

    D. `onlyif => $noop_mode`

Answer: B
Explanation: This is an advanced feature. A resource with `noop => true` will *still* report if it *would* have changed. Another resource can subscribe to this "noop" change, allowing for complex "pre-flight" checks.

95. What is the `puppet-scl` package?

    A. A package for managing SELinux.

    B. A package that provides a self-contained "Software Collection" of Puppet and its Ruby dependencies.

    C. A package for scaling the Puppet master.

    D. A package for running Puppet in a container.

Answer: B
Explanation: On systems like RHEL/CentOS, this package provides all Puppet components (Ruby, gems, etc.) in a self-contained environment (`/opt/puppetlabs/`) to avoid conflicts with system Ruby.

96. What is the "transactional" nature of a Puppet run?

    A. It means Puppet generates a report.

    B. It means Puppet "transacts" data with PuppetDB.

    C. It means that if one resource fails, the entire run stops.

    D. It means that resources are applied in a specific, predictable order.

Answer: D
Explanation: Puppet builds a graph of all resources and their dependencies. It then applies them in an order that respects those dependencies.

97. What is the purpose of the `before` metaparameter?

    A. It is the opposite of `require`.

    B. It is the same as `require`.

    C. It runs the resource before the compile phase.

    D. It is a conditional guard.

Answer: A
Explanation: `Package['a'] { before => Service['b'] }` is the same as `Service['b'] { require =` It's just a more readable way to express the same dependency.

98. What is the `file_line` resource (from the `puppetlabs-stdlib` module)?

    A. A resource to count lines in a file.

    B. A resource to ensure a specific line of text exists (or does not exist) in a file.

    C. A resource to read a line from a file.

    D. A resource to create a file with one line.

    Answer: B
    Explanation: This is a very useful resource from the "standard library" module. It provides an idempotent way to manage a single line in a configuration file (e.g., `ensure => 'present', line => '...'`)

99. What is the "certname" of a node?

    A. The node's unique, cryptographic identifier, used for its SSL certificate.

    B. The node's role.

    C. The node's environment.

    D. The node's operating system.

    Answer: A
    Explanation: The certname (usually the FQDN) is the node's unique ID. The master uses this name to find the node definition and sign its certificate.

100. What is the purpose of the `puppet.conf [main]` section?

    A. It contains settings only for the `main` stage.

    B. It contains settings that apply to *all* Puppet commands (master, agent, etc.).

    C. It contains settings for the `site.pp` manifest.

    D. It contains settings for the primary Puppet master.

    Answer: B
    Explanation: Settings in the `[main]` section (like `ssldir`) are the defaults. Other sections (like `[master]` or `[agent]`) can override them.