TCS Wings 1 (T9) Practice: 100 Ansible MCQs

Preparation Material

Here are 100 moderate-difficulty questions covering core Ansible concepts, modules, playbooks, roles, and best practices.

Core Concepts & Inventory

1. **What is the primary characteristic of Ansible's architecture?**

A) Agent-based

B) Agentless

C) Master-Slave

D) Proxy-based

Answer: B
Explanation: Ansible is agentless. It communicates with managed nodes over SSH (or WinRM for Windows) without requiring any client software to be installed on them.

2. **What does "idempotency" mean in the context of Ansible?**

A) A task can only be run once.

B) A task can be run multiple times without changing the result beyond the initial execution.

C) A task must always change the state of the system.

D) A task is executed on all hosts simultaneously.

Answer: B
Explanation: Idempotency ensures that applying an operation multiple times has the same effect as applying it once. If a file is meant to be present, Ansible will create it if it's missing, but do nothing if it's already there.

3. **What is the default location for the Ansible inventory file?**

A) `/etc/ansible/inventory`

B) `/etc/ansible/hosts`

C) `/ansible/hosts`

D) `/opt/ansible/inventory`

Answer: B
Explanation: By default, Ansible looks for its inventory at `/etc/ansible/hosts`.

4. **Which of the following is NOT a valid format for an Ansible inventory?**

A) INI

B) YAML

C) JSON

D) XML

Answer: D
Explanation: Ansible natively supports INI and YAML for static inventories. Dynamic inventories can return JSON, but XML is not a supported format.

5. **Consider the following INI inventory:**

```
[webservers]
web1.example.com
web2.example.com

[dbservers]
db1.example.com

[datacenter:children]
webservers
dbservers
```

Which host group will target `web1.example.com`, `web2.example.com`, and `db1.example.com`?

A) `[all]`

B) `[datacenter]`

C) `[webservers:dbservers]`

D) Both A and B

Answer: D
Explanation: The `[all]` group implicitly contains every host. The `[datacenter:children]` group definition explicitly groups `webservers` and `dbservers` under the `datacenter` group.

6. **What is the purpose of the `ansible.cfg` file?**

A) To define managed hosts.

B) To store encrypted variables.

C) To configure Ansible's behavior, such as inventory location, privilege escalation, and logging.

D) To define playbook tasks.

Answer: C
Explanation: `ansible.cfg` is the configuration file used to customize Ansible's settings.

7. **What is an Ansible "Control Node"?**

A) Any server managed by Ansible.

B) The machine on which Ansible is installed and from which playbooks are run.

C) A server that stores Ansible facts.

D) A server that requires an agent to be installed.

Answer: B
Explanation: The control node is your management machine where you have Ansible installed and from which you execute commands and playbooks.

8. **How can you define a variable for a single host in an INI inventory?**

A) `[web1.example.com:vars]`

B) `web1.example.com ansible_port=2222`

C) `set_fact: host="web1.example.com"`

D) `vars: web1.example.com: ansible_port=2222`

Answer: B
Explanation: You can place variables directly after the hostname in an INI inventory.

9. **In a YAML inventory, how are child groups defined?**

A) `parent_group: [child1, child2]`

B) `parent_group: children: { child1: {}, child2: {} }`

C) `parent_group: contains: [child1, child2]`

D) `children: parent_group: [child1, child2]`

Answer: B
Explanation: In YAML inventories, parent groups have a `children:` key that contains a dictionary of child groups.

10. **What is a "Dynamic Inventory"?**

A) An inventory file that is written in YAML.

B) An inventory file that changes automatically based on `ansible-pull`.

C) A script or program that Ansible calls to get inventory data from external sources (e.g., AWS, Azure, vSphere).

D) An inventory file that uses host patterns.

Answer: C
Explanation: A dynamic inventory is an executable that Ansible runs, which then fetches inventory information (hosts, groups, variables) from cloud providers, CMDBs, or other APIs.

Ad-hoc Commands

11. **Which command would check connectivity to all hosts in the `webservers` group?**

A) `ansible webservers -a "ping"`

B) `ansible webservers -m ping`

C) `ansible webservers -check ping`

D) `ansible-playbook webservers -m ping`

Answer: B
Explanation: The `-m` flag specifies the module to use (in this case, `ping`). `-a` is for module arguments, which `ping` doesn't require.

12. **What is the purpose of the `-a` flag in an `ansible` ad-hoc command?**

A) To specify the inventory file.

B) To pass arguments to the module.

C) To ask for the SSH password.

D) To run in asynchronous mode.

Answer: B
Explanation: `-a` (or `-args`) is used to provide the arguments for the module being executed.

13. **How would you run the command `uptime` on all hosts in your inventory?**

A) `ansible all -m uptime`

B) `ansible all -m shell -a "uptime"`

C) `ansible all -a "uptime"`

D) `ansible all -m command -a "uptime"`

Answer: D
Explanation: While B would also work, `command` is the default module. `command` is safer as it doesn't process shell features like pipes (`|`) or redirects (`>`). `uptime` is a simple command that doesn't need the shell. (Note: C is also a shortcut for D).

14. **What does the `-b` flag (or `-become`) signify in an `ansible` command?**

A) It runs the command in the background.

B) It backs up files before changing them.

C) It specifies privilege escalation (e.g., to `root` via `sudo`).

D) It forces a connection even if the host key is unknown.

Answer: C
Explanation: `-become` tells Ansible to use privilege escalation (like `sudo`, `su`, etc.) to execute the task as a different user, typically `root`.

15. **What is the key difference between the `command` and `shell` modules?**

A) `command` is idempotent, `shell` is not.

B) `command` does not process shell environment variables or operators (like `|`, `>`, `&`), while `shell` does.

C) `command` must be used in playbooks, `shell` in ad-hoc commands.

D) `command` runs as root, `shell` runs as the user.

Answer: B
Explanation: `command` is safer and more predictable as it executes the command directly. `shell` runs the command through `/bin/sh` (or a specified shell), allowing for shell features.

Playbooks & Syntax

16. **What language are Ansible Playbooks written in?**

A) JSON

B) Python

C) YAML

D) INI

Answer: C
Explanation: Playbooks are YAML files.

17. **What is the correct way to start a YAML file?**

A) `--` (three dashes)

B) `###` (three hashes)

C) `<?yaml version="1.0"?>`

D) `{` (an open brace)

Answer: A
Explanation: A YAML file optionally begins with `--` (document start) and can end with `...` (document end).

18. **A "Play" in an Ansible Playbook is...**

A) A single module to be executed.

B) A file containing a list of tasks.

C) A mapping of hosts to a set of tasks.

D) A collection of roles.

Answer: C
Explanation: A Play is the core unit of a playbook. It defines which hosts to target (e.g., `hosts: webservers`) and the tasks to run on them.

19. **Examine this playbook snippet:**

```
---
- name: My First Play
  hosts: webservers
  tasks:
    - name: Install nginx
      apt:
        name: nginx
        state: present
```

What is the element `apt`?

A) A task

B) A module

C) A parameter

D) A play

Answer: B
Explanation: `apt` is the name of the module being called by the task "Install nginx". `name: nginx` and `state: present` are parameters for the `apt` module.

20. **What does `hosts: all` in a play mean?**

A) It targets all hosts defined in the `[all]` group in the inventory.

B) It targets all hosts in the `all.yml` file.

C) It requires a dynamic inventory.

D) It targets all hosts Ansible can ping.

Answer: A
Explanation: `all` is a special group that includes every host found in the inventory file.

21. **What is the purpose of the `name:` directive in a play or task?**

A) It is a required identifier for Ansible to run.

B) It is used to define a variable.

C) It provides a human-readable description that is shown in the command-line output.

D) It specifies the name of the module to use.

Answer: C
Explanation: The `name:` attribute is for documentation and logging. It makes the playbook output much easier to read and debug.

22. **What is the correct syntax for a YAML list?**

A) `items: { "item1", "item2" }`

B) `items: "item1, item2"`

C)
```
items:
- item1
- item2
```

D) `items: (item1, item2)`

Answer: C
Explanation: YAML lists (arrays) are typically represented by items on new lines, each prefixed with a dash and a space (`- `).

23. **What does the `gather_facts: no` directive in a play do?**

A) It prevents Ansible from collecting system information (e.g., OS, IP address) from the managed nodes.

B) It stops the play from running.

C) It tells Ansible to use a cached set of facts.

D) It only gathers facts from the `[all]` group.

Answer: A
Explanation: By default, Ansible starts every play by "gathering facts." Setting `gather_facts: no` skips this step, which can speed up playbooks if you don't need `ansible_facts`.

24. **How do you run a playbook named `site.yml`?**

A) `ansible site.yml -m playbook`

B) `ansible-playbook site.yml`

C) `ansible all -a site.yml`

D) `run-playbook site.yml`

Answer: B
Explanation: The `ansible-playbook` command is used to execute playbook files.

25. **What is the purpose of the `-check` flag in `ansible-playbook`?**

A) It checks the playbook for syntax errors.

B) It runs the playbook in "dry-run" mode, reporting what *would* change without actually making changes.

C) It checks if the managed hosts are reachable.

D) Both A and B.

Answer: D
Explanation: `-check` (or `-C`) performs a syntax check *and* a dry run. It connects to the hosts and runs the modules, but tells them not to make persistent changes.

26. **What does the `-limit` flag do?**

A) It limits the number of tasks that can run.

B) It limits the playbook execution to a subset of the hosts defined in the play's `hosts:` line.

C) It limits the amount of CPU Ansible can use.

D) It limits the playbook to only running handlers.

Answer: B
Explanation: `-limit` is used to restrict a playbook run to specific hosts or groups, e.g., `ansible-playbook site.yml -limit web1.example.com`.

Modules

27. **Which module is used to install software packages on a Red Hat-based system (like RHEL, CentOS)?**

A) `apt`

B) `package`

C) `yum` or `dnf`

D) `software`

Answer: C
Explanation: `yum` is the traditional module for RHEL-based systems. `dnf` is the modern equivalent (RHEL 8+). `apt` is for Debian/Ubuntu. `package` is a generic module that auto-detects.

28. **Consider this task:**

```
- name: Ensure 'conf' directory exists
  file:
    path: /etc/myapp/conf
```

```
state: directory
mode: '0755'
```

What does this task do?

A) It creates a file named `conf` with permissions `0755`.

B) It creates a directory named `conf` if it doesn't exist and sets its permissions.

C) It deletes the `/etc/myapp/conf` directory.

D) It creates an empty file named `directory` inside `/etc/myapp/conf`.

Answer: B
Explanation: `state: directory` ensures that the path exists and is a directory. `state: touch` would create a file. `state: absent` would delete it.

29. **Which module would you use to copy a file from the control node to a managed node?**

A) `fetch`

B) `copy`

C) `file`

D) `rsync`

Answer: B
Explanation: `copy` moves files from the control node *to* the managed node. `fetch` moves files *from* the managed node *to* the control node.

30. **Which module is used to render a Jinja2 template file on the managed node?**

A) `copy`

B) `file`

C) `template`

D) `jinja2`

Answer: C
Explanation: The `template` module copies a file from the control node, but first processes it through the Jinja2 templating engine, allowing for variable substitution.

31. **What is the purpose of the `service` module?**

A) To install a new service.

B) To edit service configuration files.

C) To ensure a service is started, stopped, restarted, or enabled.

D) To check the network port of a service.

Answer: C
Explanation: The `service` module (or the more modern `systemd` module) manages services (daemons) on the target node.

32. **This task ensures the `ntpd` service is running and enabled at boot:**

```
- name: Ensure ntpd is running and enabled
  service:
    name: ntpd
    state: started
    enabled: yes
```

This task is an example of...

A) Idempotency

B) A non-idempotent task

C) A handler

D) Fact gathering

Answer: A
Explanation: This task is idempotent. If `ntpd` is already running and enabled, Ansible will report "ok" and make no changes. If it's stopped, it will start it and report "changed".

33. **What is the `debug` module used for?**

A) To run a playbook in step-by-step mode.

B) To print the values of variables to the console during a playbook run.

C) To check for syntax errors in a playbook.

D) To connect to a remote debugger.

Answer: B
Explanation: The `debug` module is essential for troubleshooting. You can use it to print strings or, more commonly, the content of variables (`debug: var=my_variable`).

34. **Which module is used to add or modify a single line in a file, based on a regular expression?**

A) `blockinfile`

B) `lineinfile`

C) `replace`

D) `file`

Answer: B
Explanation: `lineinfile` is perfect for ensuring a specific line exists (or doesn't exist, or is modified) in a file, often finding its place using `regexp=`.

35. **You want to add a block of text surrounded by markers (e.g., `# BEGIN ANSIBLE BLOCK`) to `/etc/hosts`. Which module is best?**

A) `template`

B) `copy`

C) `blockinfile`

D) `lineinfile`

Answer: C
Explanation: `blockinfile` manages a multi-line block of text in a file, surrounded by customizable marker lines, making it idempotent and safe.

36. **What does the `stat` module do?**

A) It starts a service.

B) It gathers `ansible_facts`.

C) It retrieves information (metadata) about a file, such as its size, permissions, and whether it exists.

D) It reports system statistics like CPU and memory.

Answer: C
Explanation: `stat` is used to check the status of a file or directory, similar to the `stat` command in Linux.

37. **Which module would you use to run a command that involves shell redirection (e.g., `echo "hello" > /tmp/hello.txt`)?**

A) `command`

B) `shell`

C) `raw`

D) `script`

Answer: B
Explanation: The `>` redirection operator is a shell feature. The `command` module would not process it. `shell` is required for this.

38. **What is the `raw` module used for?**

A) For running commands that require shell processing.

B) For installing new modules on the control node.

C) For running commands on a managed node that does not have Python installed.

D) For formatting output as raw JSON.

Answer: C
Explanation: Most Ansible modules require Python on the target node. `raw` is a last resort that sends raw SSH commands, useful for bootstrapping a machine (e.g., installing Python).

39. **Which module is used to download a file from a URL to the managed node?**

A) `get_url`

B) `fetch`

C) `download`

D) `uri`

Answer: A
Explanation: `get_url` downloads a file from an HTTP, HTTPS, or FTP source and places it on the managed node. `uri` is for making general web requests (like to an API), not just downloading files.

40. **You need to install a package but are unsure if the host is Debian-based or Red Hat-based. Which module is the best choice?**

A) `apt`

B) `yum`

C) `package`

D) `command`

Answer: C
Explanation: The `package` module is a generic wrapper that automatically detects the underlying package manager (like `apt`, `yum`, `dnf`) and uses the correct one.

Variables & Facts

41. **What is the correct syntax for a variable in a Jinja2 template?**

A) `$my_variable`

B) `{{ my_variable }}`

C) `(my_variable)`

D) `%{my_variable}`

Answer: B
Explanation: Jinja2 (Ansible's templating language) uses double curly braces `{{ ... }}` for variable substitution.

42. **Where would you define variables that apply to all hosts in the `webservers` group?**

A) `group_vars/webservers.yml`

B) `host_vars/webservers.yml`

C) `webservers/vars.yml`

D) `inventory.yml`

Answer: A
Explanation: Ansible automatically loads variables from files in the `group_vars/` directory that are named after a group.

43. **What is an "Ansible Fact"?**

A) A variable defined in a playbook.

B) A piece of system information (e.g., IP address, OS version) discovered by Ansible on a managed node.

C) A task that is always run.

D) An encrypted string from Ansible Vault.

Answer: B
Explanation: Facts are data points collected by the `setup` module during the "Gathering Facts" stage. They are stored in the `ansible_facts` variable.

44. **How would you access the primary IPv4 address of a managed node inside a template?**

A) `{{ ansible_facts.ipv4.address }}`

B) `{{ ansible_facts.default_ipv4.address }}`

C) `{{ ansible_ipv4_address }}`

D) `{{ ip_address }}`

Answer: B
Explanation: While many facts are available in `ansible_facts`, common ones are also "injected" as top-level variables. `ansible_default_ipv4.address` is the standard way to get the main IP.

45. **What is the purpose of the `vars:` section in a play?**

A) To define variables that are available to all plays.

B) To define variables that are available only within that play.

C) To import variables from a file.

D) To list all facts gathered from the hosts.

Answer: B
Explanation: Variables defined in the `vars:` block of a play are scoped to that play.

46. **What is the `register` keyword used for?**

A) To register a new module.

B) To register a new host in the inventory.

C) To capture the output (stdout, stderr, return code, etc.) of a task into a variable.

D) To register a service with `systemd`.

Answer: C
Explanation: `register:` saves the resulting JSON data from a module's execution into a variable you name.

47. **Consider this task:**

```
- name: Check file content
  shell: cat /etc/hosts
  register: host_file_content

- debug:
    var: host_file_content.stdout
```

What will the `debug` task output?

A) The literal string "host_file_content.stdout"

B) A JSON object with all task results.

C) The standard output of the `cat /etc/hosts` command.

D) An error, as `.stdout` is not a valid key.

Answer: C
Explanation: The `register` keyword saves the task's output to `host_file_content`. The standard output is stored in the `.stdout` key of that variable.

48. **What is the variable precedence order (from lowest to highest)?**

A) Play `vars:`, Role defaults, Inventory `group_vars:`, Extra vars (`-e`)

B) Role defaults, Inventory `group_vars:`, Play `vars:`, Extra vars (`-e`)

C) Extra vars (`-e`), Play `vars:`, Inventory `group_vars:`, Role defaults

D) Inventory `group_vars:`, Role defaults, Extra vars (`-e`), Play `vars:`

Answer: B
Explanation: This is a key concept. Role defaults (`defaults/main.yml`) have the lowest priority. Extra vars (`-e` on the command line) have the highest priority and will override everything else.

49. **How do you define variables on the command line?**

A) `ansible-playbook site.yml -vars "key=value"`

B) `ansible-playbook site.yml -v "key=value"`

C) `ansible-playbook site.yml -e "key=value"`

D) `ansible-playbook site.yml -define "key=value"`

Answer: C
Explanation: `-e` or `-extra-vars` is used to pass variables (as "key=value" pairs or JSON/YAML) on the command line.

50. **What is the `vars_prompt` section used for?**

A) To prompt the user for input during playbook execution and save it as a variable.

B) To print the value of a variable to the screen.

C) To define variables that are high-priority.

D) To validate the syntax of variables.

Answer: A
Explanation: `vars_prompt` allows you to ask the user interactive questions (e.g., "Enter the new password:") before the tasks run.

51. **What is the `set_fact` module used for?**

A) To define a new fact on the control node.

B) To define a new variable for a host *during* a play's execution.

C) To gather facts from a host.

D) To write variables to a file.

Answer: B
Explanation: `set_fact` creates a new variable in the `ansible_facts` dictionary for the *current host* it's running on. This variable is then available for subsequent tasks in the play.

Handlers & Conditionals

52. **What is a "Handler" in Ansible?**

A) A task that runs only at the beginning of a play.

B) A task that is triggered by a `notify` directive and runs at the end of a play.

C) A module that handles errors.

D) A special task for handling variables.

Answer: B
Explanation: Handlers are tasks (e.g., "Restart apache") that only run if a task that `notify`'s them reports a "changed" status. They run after all tasks in the play are complete.

53. **Examine this playbook:**

```
- hosts: web
  tasks:
    - name: Update config
      copy:
        src: httpd.conf
        dest: /etc/httpd/conf/httpd.conf
      notify: Restart apache

  handlers:
    - name: Restart apache
      service:
        name: httpd
        state: restarted
```

If the `httpd.conf` file on the control node is identical to the one on the managed node, will the "Restart apache" handler run?

A) Yes, handlers always run.

B) No, because the `copy` task will report "ok" (not "changed").

C) Yes, because the `notify` directive forces it to run.

D) Only if the `httpd` service is already stopped.

Answer: B
Explanation: Because of idempotency, the `copy` module will see the files are identical and report "ok" (green). Handlers are only triggered by a "changed" (yellow) status.

54. **What is the `when` keyword used for?**

A) To define a handler.

B) To specify a time for the playbook to run.

C) To run a task conditionally, based on the evaluation of an expression.

D) To loop over a list of items.

Answer: C
Explanation: The `when:` clause provides a condition. If the condition evaluates to `true`, the task runs. If `false`, the task is skipped.

55. **Consider this task:**

```
- name: Install Apache on Debian
  apt:
    name: apache2
    state: present
  when: ansible_facts['os_family'] == "Debian"
```

On which host will this task run?

A) All hosts.

B) Only on hosts where the OS is "Debian".

C) Only on hosts in the "Debian" group.

D) It will not run, `os_family` is not a valid fact.

Answer: B
Explanation: The `when` condition checks the `ansible_facts` variable. The task will be skipped on any host where `ansible_facts['os_family']` is not "Debian" (e.g., "RedHat").

56. **What is the correct way to check if a registered variable `result` indicates a failure?**

A) `when: result.failed == true`

B) `when: result is failed`

C) `when: result.rc != 0`

D) All of the above are common patterns.

Answer: B
Explanation: Ansible provides `is failed`, `is successful`, `is changed`, and `is skipped` as convenient Jinja2 tests for registered variables. `when: result is failed` is the most idiomatic way.

57. **What is the purpose of `failed_when:`?**

A) To define a condition that, if true, will *force* a task to fail, even if it ran successfully (e.g., return code 0).

B) To define a condition that, if true, will *prevent* a task from failing, even if it returns an error.

C) To specify which hosts have failed.

D) To run a task when a previous task has failed.

Answer: A
Explanation: `failed_when:` lets you override the default success/failure logic. For example, if a `shell` command exits 0 but you see "ERROR" in its output, you can use `failed_when: "'ERROR' in command_result.stdout"` to make the task fail.

58. **What is the purpose of `changed_when:`?**

A) To define a condition that, if true, will force a task to report "changed".

B) To define a condition that, if true, will prevent a task from reporting "changed".

C) Both A and B.

D) To check when a file was last changed.

Answer: C
Explanation: It's used to control the "changed" status. By default, `command` and `shell` tasks always report "changed". You can use `changed_when: false` to make them idempotent, or `changed_when: "'user created' in result.stdout"` to make it "changed" only when specific output is seen.

59. **How do you run handlers immediately, in the middle of a play, instead of at the end?**

A) You cannot; handlers always run at the end.

B) By using the `run_handlers` module.

C) By adding a task: `- meta: flush_handlers`

D) By setting `force_handlers: true` in `ansible.cfg`.

Answer: C
Explanation: `meta: flush_handlers` is a special task that triggers any pending notified handlers to run immediately at that point in the play.

60. **If two tasks notify the same handler, how many times will the handler run?**

A) Twice, once for each notification.

B) It depends on the `handler_run_count` setting.

C) Only once, after all tasks in the play are finished.

D) It will fail, as a handler can only be notified once.

Answer: C
Explanation: Handlers are de-duplicated. No matter how many tasks notify a specific handler, it will only run once at the end of the play.

Loops & Blocks

61. **What is the modern (Ansible 2.5+) keyword for creating a loop?**

A) `with_items`

B) `loop`

C) `foreach`

D) `iterate`

Answer: B
Explanation: `loop` is the preferred, modern keyword for loops. `with_items` is the older, still-functional syntax.

62. **Consider this task:**

```
- name: Add several users
  user:
    name: "{{ item }}"
    state: present
  loop:
```

```
    - alice
    - bob
    - charlie
```

What does this task do?

A) It creates one user named "item".

B) It creates three users: `alice`, `bob`, and `charlie`.

C) It creates one user named "alice, bob, charlie".

D) It fails, as `item` is not defined.

Answer: B
Explanation: The `loop` keyword iterates over the provided list. In each iteration, the current list element is available in the `item` variable.

63. **What is the `item` variable in a loop?**

A) A special variable that always contains the number `1`.

B) The default variable name that holds the current value from the loop's list.

C) A variable you must define in `vars:`.

D) The total number of items in the loop.

Answer: B
Explanation: `item` is the default "loop variable" used by Ansible.

64. **How can you change the name of the loop variable from `item` to `username`?**

A) `loop_var: username`

B) `set_loop_var: username`

C) `loop_control: { loop_var: username }`

D) `loop(username):`

Answer: C
Explanation: The `loop_control` directive is used to change loop behavior, including setting the `loop_var` name.

65. **What is a `block` in an Ansible playbook?**

A) A way to group tasks, often for applying a common `when` condition or for error handling.

B) A directive to block network access.

C) A module for managing `blockinfile`.

D) A way to define a reusable set of tasks, similar to a role.

Answer: A
Explanation: A `block` is a logical grouping of tasks. It's most commonly used with `rescue` and `always` for exception handling.

66. **Examine this playbook structure:**

```
- name: Attempt a risky operation
  block:
    - name: Risky task
      command: /opt/risky_script.sh
  rescue:
    - name: Run this on failure
      debug:
        msg: "The risky script failed!"
  always:
    - name: Run this no matter what
      debug:
        msg: "Cleaning up."
```

If the `Risky task` fails, which task(s) will run next?

A) Only "Run this on failure"

B) Only "Run this no matter what"

C) "Run this on failure" *then* "Run this no matter what"

D) The play will halt immediately, and neither will run.

Answer: C
Explanation: This structure is Ansible's `try...catch...finally` equivalent. `rescue` runs *only* if a task in the `block` fails. `always` runs *always*, whether the `block` or `rescue` sections succeeded or failed.

67. **How would you apply a `when` condition to a group of tasks?**

A) You must add the `when` condition to every single task.

B) Put the tasks in a `block` and apply the `when` condition to the `block` itself.

C) Use `when_group:`

D) You must put the tasks in a separate file and `include` it.

Answer: B
Explanation: A `when` condition on a `block` is applied to all tasks *within* that block, which is much cleaner than repeating the condition.

Roles

68. **What is the primary purpose of an Ansible "Role"?**

A) To define privilege escalation (e.g., root).

B) To package and reuse a collection of tasks, handlers, variables, and files for a specific purpose (e.g., "webserver").

C) To define a host's function (e.g., `role: db`).

D) To encrypt sensitive data.

Answer: B
Explanation: Roles are the primary mechanism for abstracting, encapsulating, and reusing Ansible code.

69. **What is the name of the command-line tool used to create a new role directory structure?**

A) `ansible-role-init`

B) `ansible-galaxy init`

C) `ansible-new-role`

D) `ansible-scaffold`

Answer: B
Explanation: `ansible-galaxy init my_new_role` will create the standard directory skeleton for a role.

70. **In a role, which directory contains the main list of tasks to be executed?**

A) `main/tasks.yml`

B) `tasks/main.yml`

C) `tasks.yml`

D) `role.yml`

Answer: B
Explanation: Ansible looks for `tasks/main.yml` inside the role directory for the main task list.

71. **What is the difference between `vars/main.yml` and `defaults/main.yml` in a role?**

A) `vars` are for task variables, `defaults` are for host variables.

B) `vars/main.yml` variables have a *high* priority and cannot be easily overridden. `defaults/main.yml` variables have the *lowest* priority and are meant to be overridden.

C) `vars` are for strings, `defaults` are for numbers.

D) `vars/main.yml` is for public variables, `defaults/main.yml` is for private variables.

Answer: B
Explanation: `defaults` are for "default" values. `vars` are for "role-internal" variables that the user is not expected to change.

72. **How do you use a role named `nginx` in a play?**

A)
```
- hosts: web
  roles:
    - nginx
```

B)
```
- hosts: web
  tasks:
    - import_role: nginx
```

C)
```
- hosts: web
  tasks:
    - ansible.builtin.include_role:
        name: nginx
```

D) All of the above are valid ways.

Answer: D
Explanation: A is the classic, static way. B (`import_role`) and C (`include_role`) are for using roles dynamically within the `tasks:` section. `import_role` is static (parsed at playbook start), while `include_role` is dynamic (parsed when encountered).

73. **What is "Ansible Galaxy"?**

A) The configuration file for roles.

B) A free, public repository for finding, downloading, and sharing Ansible roles.

C) A graphical UI for Ansible.

D) The command to initialize a new role.

Answer: B
Explanation: Ansible Galaxy (galaxy.ansible.com) is the community hub for sharing roles, and `ansible-galaxy` is the command-line tool to interact with it.

74. **How would you install the `geerlingguy.nginx` role from Ansible Galaxy?**

A) `ansible-galaxy get geerlingguy.nginx`

B) `ansible-galaxy download geerlingguy.nginx`

C) `ansible-galaxy install geerlingguy.nginx`

D) `ansible-galaxy clone geerlingguy.nginx`

Answer: C
Explanation: `ansible-galaxy install ...` is the command to download a role from the public repository.

75. **What is the purpose of the `meta/main.yml` file in a role?**

A) To define the role's tasks.

B) To define role metadata, such as author, license, and dependencies on other roles.

C) To define variables that have meta-priority.

D) To define the main function of the role.

Answer: B
Explanation: The `meta/` directory holds metadata. `meta/main.yml` is primarily used to list `dependencies`, which are other roles that must be run *before* this one.

76. **In a play, what is the execution order between `tasks:`, `pre_tasks:`, and `post_tasks:`?**

A) tasks -> pre_tasks -> post_tasks

B) pre_tasks -> post_tasks -> tasks

C) pre_tasks -> tasks -> post_tasks

D) It depends on the `gather_facts` setting.

Answer: C
Explanation: `pre_tasks` run before any roles or tasks. `tasks` (and roles) run next. `post_tasks` run after all tasks/roles are complete.

77. **How can you pass a variable to a role?**

A)
```
- hosts: web
  roles:
    - role: nginx
      nginx_port: 8080
```

B)
```
- hosts: web
  roles:
    - nginx
  vars:
    nginx_port: 8080
```

C) By defining it in `group_vars/web.yml`

D) All of the above.

Answer: D
Explanation: All of these are valid ways to pass variables to a role, which will override the role's `defaults/main.yml`.

Ansible Vault

78. **What is the purpose of Ansible Vault?**

A) To speed up playbook execution.

B) To store and manage encrypted files, such as files containing secrets or passwords.

C) To provide a graphical dashboard for Ansible.

D) To back up playbook files.

Answer: B
Explanation: Ansible Vault provides a way to encrypt sensitive data (files or individual variables) at rest, so they can be safely committed to source control.

79. **What command is used to create a new encrypted file named `secrets.yml`?**

A) `ansible-vault new secrets.yml`

B) `ansible-vault create secrets.yml`

C) `ansible-vault encrypt secrets.yml`

D) `ansible-vault make secrets.yml`

Answer: B
Explanation: `ansible-vault create` will prompt for a password and then open an editor for the new, encrypted file.

80. **You have an existing, unencrypted file `vars/prod.yml` that you want to encrypt. What command do you use?**

A) `ansible-vault create vars/prod.yml`

B) `ansible-vault encrypt vars/prod.yml`

C) `ansible-vault edit vars/prod.yml`

D) `ansible-vault lock vars/prod.yml`

Answer: B
Explanation: `encrypt` is used to encrypt an existing *plaintext* file.

81. **How do you run a playbook `site.yml` that uses an encrypted file?**

A) `ansible-playbook site.yml -vault-password`

B) `ansible-playbook site.yml -vault-id @prompt`

C) `ansible-playbook site.yml -ask-vault-pass`

D) `ansible-playbook site.yml -decrypt`

Answer: C
Explanation: `-ask-vault-pass` will interactively prompt you for the vault password before running the playbook.

82. **How can you store the vault password in a file so you don't have to type it?**

A) It is not possible; you must always type it.

B) By creating a `.vault_pass` file and using `ansible-playbook site.yml -vault-password-file .vault_pass`

C) By setting the `ANSIBLE_VAULT_PASS` environment variable.

D) Both B and C are valid methods.

Answer: D
Explanation: Using a password file (`-vault-password-file`) or an environment variable are common ways to automate vault usage, especially in CI/CD pipelines.

83. **What command is used to edit an *already encrypted* file?**

A) `ansible-vault encrypt`

B) `ansible-vault create`

C) `ansible-vault edit`

D) `ansible-vault decrypt`

Answer: C
Explanation: `ansible-vault edit` will prompt for the password, decrypt the file into a temporary location, open your editor, and then re-encrypt the file when you save and quit.

84. **What is the command to change the password for a vault-encrypted file?**

A) `ansible-vault changepass`

B) `ansible-vault password`

C) `ansible-vault rekey`

D) `ansible-vault update`

Answer: C
Explanation: `ansible-vault rekey` is used to change the password of an existing encrypted file.

Strategy & Advanced Concepts

85. **What is a common "strategy" setting in a play, and what does it do?**

A) `strategy: linear` (the default) - Runs tasks on hosts one by one.

B) `strategy: free` - Runs tasks on all hosts in parallel without waiting for others.

C) `strategy: fast` - Skips tasks that have failed.

D) `strategy: parallel` - Runs one task at a time, but on all hosts.

Answer: B
Explanation: The default strategy is `linear`, where Ansible finishes a task on *all* hosts before moving to the *next* task. `strategy: free` allows hosts to run ahead, executing tasks as fast as they can, which can be much faster but harder to follow.

86. **What does the `serial:` keyword in a play control?**

A) The order of tasks.

B) How many hosts to run the play on at a time (e.g., in a rolling update).

C) The serial number of the playbook run.

D) The connection type (serial vs. SSH).

Answer: B
Explanation: `serial:` is used for rolling updates. `serial: 1` would run the *entire* play on one host, then the next, etc. `serial: "30%"` would run on 30% of the hosts at a time.

87. **What is `ansible-pull`?**

A) An ad-hoc command to fetch files from managed nodes.

B) A mode of operation where managed nodes fetch a playbook from a Git repository and run it locally.

C) A command to download roles from Ansible Galaxy.

D) A command to check for new Ansible facts.

Answer: B
Explanation: `ansible-pull` inverts the standard "push" model. It's useful for large-scale or disconnected environments.

88. **What does "delegation" mean in Ansible?**

A) Running a task on a different host than the one targeted by the play.

B) Assigning a role to another user.

C) Using `become:` to get root privileges.

D) Skipping a task.

Answer: A
Explanation: Using `delegate_to: localhost` is common. For example, you might loop over your `webservers` group, but use `delegate_to: localhost` to add each one to a load balancer monitoring pool from the control node.

89. **Consider this task:**

```
- name: Add web server to load balancer
  elb_target:
    ...
  delegate_to: localhost
```

Where does this task execute?

A) On all hosts in the `webservers` group.

B) On the host named `localhost` in the inventory.

C) On the Ansible control node (which is what `localhost` usually refers to).

D) It fails, as `delegate_to` is not a valid keyword.

Answer: C
Explanation: `delegate_to: localhost` makes the task run on the machine executing the `ansible-playbook` command, even if the play targets `hosts: webservers`.

90. **What is the purpose of `run_once: true`?**

A) It ensures a task only runs on the *first* host in the group and is skipped for all others.

B) It makes the entire playbook run only one time.

C) It prevents a handler from running more than once.

D) It skips the task if it has been run successfully before.

Answer: A
Explanation: This is useful for tasks that only need to happen once, like registering a server with an API or creating a database. It's often combined with `delegate_to: localhost`.

91. **What is "check mode"?**

A) Another name for `ansible-vault`.

B) A mode where Ansible checks syntax but doesn't connect to hosts.

C) A "dry-run" mode (`-check`) where Ansible reports what *would* change.

D) A mode for checking network connectivity (`-m ping`).

Answer: C
Explanation: Check mode is a critical feature for safely testing playbooks before applying them.

92. **Which module is used to wait for a port to become open on a host?**

A) `port_check`

B) `wait_for`

C) `service`

D) `shell`

Answer: B
Explanation: `wait_for` is a powerful module that can pause a play and wait for a condition, such as a port being open, a file existing, or text to appear in a file.

93. **Consider this task:**

```
- name: Wait for web server to come up
  wait_for:
    port: 80
    host: "{{ inventory_hostname }}"
    delay: 5
    timeout: 30
```

What does this task do?

A) It opens port 80 on the host.

B) It pauses the play for 30 seconds.

C) It checks if port 80 is open, retrying every 5 seconds for a maximum of 30 seconds.

D) It connects to port 80 and waits for 5 seconds.

Answer: C
Explanation: This task is used to pause the play until a service (like a web server) on the target host is ready to accept connections on port 80.

94. **Which module allows you to `include` another task file, but dynamically during execution?**

A) `import_tasks`

B) `include_tasks`

C) `include`

D) `load_tasks`

Answer: B
Explanation: `include_tasks` is dynamic. This means it's processed *during* the play, so it can be used with `loop:` and `when:`. `import_tasks` is static, meaning it's "pasted" into the playbook at parse time, *before* execution.

95. **What is the difference between `import_tasks` and `include_tasks`?**

A) `import_tasks` is for roles, `include_tasks` is for plays.

B) `import_tasks` is static (parsed at the start), `include_tasks` is dynamic (parsed during execution).

C) `import_tasks` is for local files, `include_tasks` is for remote files.

D) There is no difference; they are aliases.

Answer: B
Explanation: This is a key difference. Use `import_tasks` for static playbook structure. Use `include_tasks` when you need to conditionally include tasks or loop over them.

96. **Which host pattern selects all hosts in the `webservers` group *except* for `web3.example.com`?**

A) `webservers - web3.example.com`

B) `webservers:!web3.example.com`

C) `webservers AND NOT web3.example.com`

D) `webservers:-except:web3.example.com`

Answer: B
Explanation: The `!` operator in a host pattern means "not". `webservers:&dbservers` would mean "hosts in *both* webservers and dbservers".

97. **What is the purpose of the `ansible_connection` variable?**

A) To set the connection timeout.

B) To define the connection plugin to use (e.g., `ssh`, `winrm`, `docker`).

C) To check the connection status.

D) To define the SSH user.

Answer: B
Explanation: This inventory variable tells Ansible *how* to connect to a host (e.g., `ansible_connection=winrm` for Windows, `ansible_connection=docker` for a Docker container).

98. **Which `ansible.cfg` setting defines the default user for SSH connections?**

A) `remote_user`

B) `ansible_user`

C) `ssh_user`

D) `connection_user`

Answer: A
Explanation: In `ansible.cfg`, `remote_user` sets the default username to use for connecting to managed nodes.

99. **How can you force a task to run, even in check mode?**

A) `check_mode: false` on the task.

B) `force_run: true` on the task.

C) `run_in_check: yes`

D) You cannot run tasks in check mode.

Answer: A
Explanation: If you have a task that *must* run (e.g., a `stat` check to see if a file exists, which is safe to run), you can add `check_mode: false` to that specific task to override the global `-check` flag.

100. **What is the `ignore_errors: true` directive used for?**

A) It tells Ansible to ignore syntax errors in the playbook.

B) It tells Ansible to continue running the play, even if that specific task fails.

C) It tells Ansible to ignore connection errors.

D) It tells Ansible to not report "changed" status.

Answer: B
Explanation: By default, if a task fails, Ansible stops execution on that host. `ignore_errors: true` allows the play to continue, which is useful if you *expect* a task might fail (e.g., trying to remove a file that might not exist).