**Terraform items for Interview:**

1. **Modules - Types:**
   * **Module:** A self-contained package of Terraform configurations that can be used as a reusable component.
   * **Root Module:** The main directory containing the primary Terraform configuration files.
   * **Child Module:** Modules called from the root module or other modules.
2. **Dynamic Block:**
   * Allows dynamic nested blocks within resource, variable, and other declarations.
   * Useful for handling repeated configurations.
3. **Optional Attribute:**
   * Allows specifying optional attributes in resource configurations.
   * Useful for making certain resource properties conditional.
4. **Locking Mechanisms (terraform state lock/unlock/release/delete):**
   * **Lock:** Used to prevent multiple Terraform commands from executing concurrently.
   * **Unlock:** Releases a previously acquired lock.
   * **Release:** Similar to unlock, releases a lock acquired by a previous Terraform run.
   * **Delete Lock File:** Manually delete the lock file.
5. **Sensitive Information:**
   * The **sensitive** attribute is used to mark sensitive data like passwords, preventing them from being displayed in logs or outputs.
6. **Key Vault Integration:**
   * Integration with Azure Key Vault allows Terraform to retrieve secrets securely.
7. **Terraform Workflow (init, plan, apply):**
   * **terraform init**: Initializes a new or existing Terraform working directory.
   * **terraform plan**: Creates an execution plan for changes to infrastructure.
   * **terraform apply**: Applies the changes to reach the desired state.
8. **Current Version:**
   * Check the current installed version of Terraform: **terraform version**
9. **Terragrunt Tool:**
   * A thin wrapper for Terraform that supports locking, remote state management, and other advanced features.
   * Useful for managing configurations across multiple folders.
10. **Import:**
    * The **terraform import** command associates an existing resource with a Terraform resource configuration.
11. **Multiple Users in Terraform:**
    * Terraform uses provider authentication mechanisms for cloud providers.
    * Users need appropriate credentials to access the desired infrastructure.
12. **Multi-Subscription Deployment:**
    * For multi-subscription scenarios, use provider configurations for each subscription.
13. **Taint (terraform taint):**
    * Marks a resource for recreation on the next **terraform apply**.
    * Useful for forcing the recreation of specific resources.
14. **Formatting/Validation (terraform fmt/validate):**
    * **terraform fmt**: Formats the code to adhere to the standard style.
    * **terraform validate**: Checks the configuration files for syntax errors.
15. **Branching Strategy or Folder Structure (Dev/QA/Prod):**
    * Commonly, separate folders or branches are used for different environments (e.g., dev, qa, prod).
16. **Backend Statefile:**
    * Specifies where to store the Terraform state file, such as a remote backend or local file.
17. **Depends On:**
    * Specifies explicit dependencies between resources.
18. **count[index] vs for-each[map/list]:**
    * **count**: Specifies the number of instances of a resource.
    * **for-each**: Allows dynamic creation of resources based on a map or list.
19. **Another Use Case of Count - Boolean/True-False:**
    * **count** can be used with a boolean expression to conditionally create resources.
20. **Map vs List:**
    * **Map:** Key-value pairs.
    * **List:** Ordered collection of elements.
21. **Providers Files:**
    * Configuration files specifying provider settings, credentials, and configurations.
22. **Component - Resource, Providers, Data, Var, Output, Backend, Module, Init/Plan/Apply:**
    * Components of Terraform configurations, each serving a specific purpose in infrastructure management.
23. **Lifecycle:**
    * Allows defining custom behavior before or after resource creation or deletion.
24. **Destroy Specific Resource:**
    * Use **terraform destroy -target=resource\_type.resource\_name** to destroy a specific resource.
25. **Root and Child Module:**
    * The root module is the main directory containing the primary Terraform configuration files.
    * Child modules are reusable configurations called from the root module or other modules.
26. **Var and Output:**
    * **var**: Input variables.
    * **output**: Displayed values after applying the Terraform configuration.
27. **Workspace - Default/Dev/QA/Prod:**
    * Workspaces allow managing multiple environments within the same Terraform configuration.
28. **Resource Graph:**
    * Visual representation of the dependencies between resources.
29. **Null Resource = Optional:**
    * The **null\_resource** allows defining a resource with no direct counterpart in the infrastructure.
30. **Provisioners - Exec, Local, Zip, Remote:**
    * Tools to configure and set up resources during or after creation.
31. **Implicit and Explicit Depends:**
    * **Implicit:** Dependencies inferred by Terraform based on resource references.
    * **Explicit:** Dependencies specified using the **depends\_on** attribute.