Terraform Questions:

Terraform Fundamentals (5 questions)

1. What is Terraform and how does it differ from other IaC tools?

   terraform is open source IaC tool, allows users to define and manage infrastructure across few cloud provides like Azure and AWS.
   unlike other IaC tools like ansible and pulumi, terraform is declarative approach to IaC- specify the desired state of the infrastructure.

2. Explain Terraform's declarative nature and state management.

   It means that it specify the desired state of the infrastructures,
   and by state management track the current position of the infrastructure to ensure terraform knows the changes it needs to apply

3. What is the purpose of the Terraform provider?

   Terraform has provides plugins that interact with APIs of cloud provides, allows terraform to connect the resources of the different clouds like AWS and Azure and their services.

4. How does Terraform handle dependency resolution?

   Terraform uses dependency information for determined in the correct order to create the resources. Terraform automatically detect dependencies by using implicit dependency graph. By using depends_on manually, the user can enforce dependencies.

5. What are the key components of a Terraform configuration file?

   the key components are reading the configuration files to understand the desire state of the configuration. State management by tracking the current state and ensure terraform knows the changes needs to by done. Execution plans by determines necessary actions to achieve the desire state based on the differences between the current and the desire state.

State Management & Backend Configuration (3 questions)

6. Explain the difference between terraform refresh, terraform plan, and terraform apply.

   terraform refresh- updates the Terraform state file.
   terraform plan- preview the changes before making them.

terraform apply- provide the resource in your environment.

7. What is the difference between local and remote backends?

   local backend is for small projects, usually personal use, needs manual backups, unlike remote backends that use for production environment, when multiple user works with terraform, automated backup.

8. How can you prevent state corruption when multiple engineers work on the same infrastructure?

   Terraform use state locking - lock your state for all operations that could write state. prevents multiple workers from changing and applying the terraform files at the same time.

Terraform Modules & Reusability (4 questions)

9. What are the benefits of using Terraform modules?

   Modules are able to promote reuse, consistency and maintainability of infrastructure code. It also saves time, by allowing you to use an existing code and not write from scratch.

10. Explain how to pass variables to a Terraform module.

    When using the module, we pass it the required parameters, usually according to the README file that gives us exact information about the variables.
    We need to pass the variable when we declare the module so we can implement it.

11. What is the difference between count and for_each?

    both apply conditional logic.
    count use to define how many instance of a resource are created or can use to conditionally create a resource by: ? 1:0
    for_each use to loop over sets of values, creating resource for each entry. Use for more complex conditions.

    for_each gives more flexibility then count when creating multiple instances of a resource with unique configuration.

12. How do you source a module from a Git repository?

    Going to the git repository that contain the module →
    copy the URL repository (inside "code" button, HTTPS and copy) →
    Going to the folder of the model in the github and see the path and we can copy the location→
    in the .tf file: write name we decide for the module:
    Module "demo"{
        source = "git :: URL // folder location"
    }
    → when we run "terraform init", it downloads the repository and the module resources.

Terraform with AWS (4 questions)

13. How do you create an EC2 instance with Terraform?

    We need to use give the provider and then the resource aws_instance and the dependent resources. There require attributes like ami and instance type. We can define a security group according to what we want to allow ec2 to do, by aws_security_group resource.

    example from lab 101:

```
resource "aws_instance" "vm" {

        ami = "ami-0c02fb55956c7d316"
        instance_type = "t2.micro"
        vpc_security_group_ids = [aws_security_group.sg.id]

        tags = {
            Name = "Moran-vm"
        }

    }
```

14. What are the required fields for defining a VPC in Terraform?

    according to aws_vpc (can find in terraform registry):
    there are no required fields for defining VPC in terraform.
    all the fields are optional.

15. Explain how Terraform manages IAM policies in AWS.

    Terraform can manage all the IAM operation through code. Including permissions.
    By using resource aws_iam_policy, you can provide IAM policy with fields like description, name_prefix, path, policy (json file).

16. How do you use Terraform to provision and attach an Elastic Load Balancer?

    The resource aws_elb provides an elastic load balancer resource
    require fields are:
    availability_zones - required for an EC2-classic ELB.
    subnets - required for a VPC ELB, a list of subnet IDs to attach to the ELB.
    listener – required, a list of listener blocks. Listeners documented below.

    The resource aws_elb_attachment attaches an EC2 instance to ELB.
    require fields are:
    elb – name of the ELB is required.
    instance – require the instance ID that we attach to the alb.

    Means the aws_elb_attachment depend on aws_elb

Debugging & Error Handling (4 questions)

17. What does the terraform validate command do?

    terraform validate minimize the errors in terraform by checking syntax errors and configuration issues before running the command terraform apply an plan.

18. How can you debug Terraform errors effectively?

    use terraform validate for syntax error and terraform plan (to preview if there is misconfigurations before the running),
    TF-LOG before normal terraform command. Terraform provide log levels for debugging with 5 types of log levels: TRACE, DEBUG →INFO, WARN, ERROR.
    read carefully the error messages is always a good idea and using hashicorp terraform can helps avoiding it by highlighting syntax and autocompletion for terraform.

19. What is Terraform’s ignore_changes lifecycle policy used for?

    Terraform's ignore_changes are meant to be used when creating the resource with references to data that may change.
    ignore_changes lifecycle policy prevent Terraform from changing specific properties of a resource, even if they are changed outside of Terraform (maybe change manually in the AWS console).
    should be use carefully.
    inside lifecycle we can add the ignore_changes argument with the require field.

20. How do you import existing AWS infrastructure into Terraform?

    Using terraform import command, it is possible to attach existing infrastructure without recreate, like instance, to resource configuration.
    the strucrue of import is:
    terraform import resource_type_name.resource_name id