Terraform Fundamentals (5 questions)

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

A: Terraform is a tool that's used to write Infrastructure as code. Using declarative configuration files you can define and manage infrastructure. Terraform is better for infra thanks to its state file that tracks your actions and the ability to “plan” your infra before “applying”. Ansible for example requires an ssh connection to a usually dynamic inventory(list of hosts) and is better for maintenance and configuring a vm.

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

A: Terraform’s declarative nature lets you specify what the infrastructure should end up as (desired end state), and then terraform will determine the actions needed to reach your outcome.

Terraform's state management creates a state file that keeps a record of deployed resources. This lets terraform compare desired to actual infra and create a solid accurate plan, so that you'll be able to apply changes predictably

3. What is the purpose of the Terraform provider?

A: The Terraform provider translates your Terraform configurations into API calls that create and manage resources to the cloud you configured when you defined your provider.

4. How does Terraform handle dependency resolution?

A: Terraform manages dependencies by checking the relationships between resources and organizing them into a dependency graph so that it can apply changes in the correct order.

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

A: Key components of terraform config: Provider(cloud connection), resources(infra to create), data(check existing info on provider), variables(custom vars), outputs(prints and can be used as in the moment vars), modules (not key, but can be very useful for common actions). MUST END WITH *.tf*

State Management & Backend Configuration (3 questions)

1. Explain the difference between terraform refresh, terraform plan, and terraform Apply.

A: Refresh - updates the state file to what actually exists

Plan - preview the changes you’ve made to your infra since your last apply

Apply - apply the changes, whether its creating updating or destroying old infra.

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

A: Local backend will store state file on your machine, while remote backend will save the state file remotely like on s3

3. How can you prevent state corruption when multiple engineers work on the same Infrastructure?

A: Use a remote backend that supports state locking so that only one person can make changes at a time. Can be stored on s3 or other shared storage solutions.

Terraform Modules & Reusability (4 questions)

1. What are the benefits of using Terraform modules?

A: modules help organize common procedures, and are reusable/ output customizable

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

A: You declare a variable in your module, then give it a value inside the module block like so:

```
module "BLA" {
  source        = "./mod"
  Var1 = "bla bla"
  Var2 = 5
  …}
```

3. What is the difference between count and for_each?

A: count creates X amount of identical resources

for_each creates a resource for every item in a map or a set (you can use this to give specific values)

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

A: set the module's source to the repository URL:

```
module "BLA" {
  source = "git::https://github.com/username/terraform-bla-module.git"
  …}
```

Terraform with AWS (4 questions)

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

A: After setting up your terraform environment(terraform init), you need to configure the aws provider. Then define a resource "aws_instance" with instance type, and ami included in the resource block. If you want to choose nondefault values you may define your subnet, security group, key, and more.

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

A: The only required field in VPC definition is *cidr_block*

for example:

```
resource "aws_vpc" "myvpc"{  cidr_block = "10.0.0.0/16" }
```

3. Explain how Terraform manages IAM policies in AWS.
A: By keeping IAM policies in your code you are holding the standard and any changes made outside the code would revert back the moment you run the next apply

4. How do you use Terraform to provision and attach an Elastic Load Balancer?
A: define the ELB, but you also need to define a listener, a target group, and then attach your instances to the target group.

Debugging & Error Handling (4 questions)

1. What does the terraform validate command do?
A: terraform validate checks your code instead of performing the check against your aws resources or making a change

2. How can you debug Terraform errors effectively?
A: Terraform validate, and terraform plan are great tools for debugging problems with your code, and checking with the aws resources themselves with *terraform plan*

3. What is Terraform's ignore_changes lifecycle policy used for?
A: Terraform's ignore_changes lifecycle policy is used if you want to ignore certain changes that may happen without needing to make terraform try and change it back.
For example, adding this to an aws_instance block:

```
  lifecycle {
    ignore_changes = [associate_public_ip_address]
 }
```

This will stop terraform from trying to reset the IP if you or aws change it.

4. How do you import existing AWS infrastructure into Terraform?
A: using the *terraform import <RESOURCE> <RESOURCE_ID>* command.
This will import an existing resource using its ID, and link it to a resource you defined. If you want to continue using it you have to fill out its fields so that terraform doesn't flag it as changes.