**INSTRUCTIONS:**

1. Answer all questions from Section 1, 2 & 3
2. Section 4 is a programming exercise. Prepare complete Terraform IaC with [best practices](https://cloud.google.com/docs/terraform/best-practices-for-terraform) and share your project using GitHub.

**SECTION 1: GIT**

1. if you using git stash, where will it save data? What is diff b/w index and staging area?
2. when would individuals use git rebase, git fast-forward, or a git fetch then push?
3. How to revert already pushed changes ?
4. What is the difference between cherry picking commits vs trying a hard reset. What is the final outcome of the head reference?
5. Explain the difference between git remote and git clone?

**SECTION 2: TERRAFORM**

1. what is the difference between terraform count and for\_each meta data function? and give a scenario-based example to use them?
2. What is Terraform taint ? When to use it? When would you use terraform state rm vs terraform taint?
3. How would you show a diagram of all terraform resources in the state file? When is this useful?
4. Solve this expression:

count                  = var.run\_remote\_environment ? var.TFC\_RUN\_ID !=["Yes"]) : null

1. How would you apply terraform to multiple accounts simultaneously? We want to ensure this follows security best practices.

**SECTION 3: AWS**

1. You have an EC2 instance that has an unencrypted volume. You want to create another Encrypted volume from this unencrypted volume. Which of the following steps can achieve this? How would you share this encrypted volume to another account? What must you ensure to make sure this cross-account encryption is shared?
2. How will you implement service control policy and in which area are you using it?
3. How can you convert a public subnet to private subnet?
4. What is the default route for any newly created route table?
5. How would you ensure routes in the route table DO NOT use the local routes?

**SECTION 4: Programming Exercise**

Exercise 1 (AWS) :

Terraform Code to create the below Infrastructure:

* 1 VPC in us-east-1 region. This should be flexible based on region. If no region is provided, this should be built in us-east-1.
* 2 Subnets with high availability supported in 2 zones
* 1 Route table not including the default one. Routes should not be routed using the local route.
* Autoscaling group with a flexible cool down, deregistration delay, instance warm up.
* 2 EC2 instances created from the autoscaling group
* ALB to load-balance the app servers. Ensure the port is flexible based on the application.
* IAM roles should only be used by the account owner.

**SECTION 1: GIT**

**1.** if you using git stash, where will it save data? What is diff b/w index and staging area?

**ANS**: If you are working on one branch, in between if you get any critical issues or bugs which needs to be fixed on another branch. In this case before I switch to another branch we need to stash it from current branch (stash will store it in temprory area). After completing work on another branch switch it to back to current branch and get stashed changes using git stash pop.

latest stash you created is stored in refs/stash.

older stashes are stored in the reflog.

Index area and staging area both are the same we call it has intermediate area between the working directory (workspace) and remote repository.

Once the changes have made to source to git has to trace the changes if it is the working area its untracked file “git add” command will move the source code (file) from working directory to staging area or index area.

**2.** when would individuals use git rebase, git fast-forward, or a git fetch then push?

**ANS:** We use “git rebase” if you are working on a feature branch and you want to bring the latest changes from the main master branch into your feature branch.

git rebase main == Command to bring all the changes and commit history added to tip of the other branch

git fast-forward === Suppose if you are working on feature branch and want to update it to latest state of the main branch and these you have do without creating a merge commit. In this scenario we can use the command.

git fetch ==== when we run git fetch command it will bring the changes from remote repo and stores it on a separate branch so you can review the changes and merge it to local repo if it is

required and we can push it to remote repository.

**3.** How to revert already pushed changes ?

**ANS:** First we need to check the log to view the commit history and copy the commit hash that you want to revert.

Then if we want to revert the changes which is already pushed we can use “git revert <commit ID> “ commit hash which is taken from the log command. Which creates a new commit and we can see what are the changes made in the previous commit.

**4**. What is the difference between cherry picking commits vs trying a hard reset. What is the final outcome of the head reference?

**ANS:** git cherry-pick command is used to merge specific commit on the branch

git reset --hard command moves the branch pointer and the HEAD to the specified commit and also it will discards all changes in your working directory and staging area.

**5.** Explain the difference between git remote and git clone?

**ANS:** We can say that “git remote” is used for managing remote repository with in your local git repository.

git clone ==== if you bringing the remote repo to working directory for the first time we use git clone command .

**SECTION 2: TERRAFORM**

**1.** what is the difference between terraform count and for\_each meta data function? and give a scenario-based example to use them?

terraform count === If you want to create a multiple instances with similar configuration that is in the module we use count function . If we use count=3 it will create 3 similar type of resource.

terraform for\_each === If we want to create similar type of resource with different configuration like changing name type for that we use for\_each function

**2.** What is Terraform taint ? When to use it? When would you use terraform state rm vs terraform taint?

**ANS:** We use terraform taint if the resource is in an inconsistent state on this condition we will going to taint the resource .

It is usefull when we want the tainted resource should be recreate and it has to function properly.

terraform state rm ===== If you want to replace a resource instance with a new one while maintaining the same terraform configuration . In this scenario we can use the terraform state rm we need to remove the existing resource instance by executing the command it will remove the old instance from the state and terraform will create the new instances from the same configuration.

terraform state rm resource\_type resource\_name

**3.** How would you show a diagram of all terraform resources in the state file? When is this useful?

**ANS:** To show a diagram of all terraform resources in the state file terraform has provided a tool called “graph” which will create a visiual representation of our infrastructure as defined in our configuration files.

It is usefull when we want a clear overview of the resources.

**4.** Solve this expression:

count                  = var.run\_remote\_environment ? var.TFC\_RUN\_ID !=["Yes"]) : null

**ANS:** First will takeIf var.run\_remote\_environment is true and var.TFC\_RUN\_ID is not equal to the string "Yes" then count is set to 1 which will install the one instance.

If var.run\_remote\_environment is true but var.TFC\_RUN\_ID is equal to the string "Yes", then count is set to 0 which will not install any instance.

If var.run\_remote\_environment is false and if count is set to null then it means no instances will be created because when count is null, Terraform will not create any instances.

**5.** How would you apply terraform to multiple accounts simultaneously? We want to ensure this follows security best practices.

**ANS:** If we want to apply terraform to multiple accounts simultaneously we want to follows some of the best practices to overcome any issue.

For the first we need to create a user and we can attach IAM roles for this user grant only the necessary permissions for the resources you want to manage with Terraform.

Secondly we can configure AWS CLI Profiles for each accounts. In this we need to provide the accont details of access key and secret key of both the accounts.

Thirdly in the provider block we can provide the profile details the name which you given in the AWS CLI Profiles. This way we can apply terraform infrastructer.

**SECTION 3: AWS**

**1.** You have an EC2 instance that has an unencrypted volume. You want to create another Encrypted volume from this unencrypted volume. Which of the following steps can achieve this? How would you share this encrypted volume to another account? What must you ensure to make sure this cross-account encryption is shared?

**ANS:** We can fallow some of steps to complete this scenario

First go to EC2 instance which has unencrypted volume and go to storage and click on the volume ID and to the right we have an actions in the actions click on create snapshot. It will create a snapshot

Second we can create an Encrypted volume from snapshot. Go to Snapshots in dashboard select the snapshot you created in the previous step and click on actions and select create volume and choose an appropriate volume size, availability zone, and encryption type (Encrypted) and it will create volume.

Third step we can attach the Encrypted volume to EC2 instance which has unencrypted volume. Go back to the Volumes under EBS in the EC2 Dashboard. Select the newly created encrypted volume and go to actions and select attach volume then attach the volume to the desired EC2 instance.

Go to the EBS in the EC2 Dashboard and select the encrypted volume you want to share. In the actions select the modify volume permissions and in this you give add cross-account permissions and lastly enter the AWS account ID of the account you want to share the volume with.

**2.** How will you implement service control policy and in which area are you using it?

**ANS:** Will implement Service Control Policies in AWS Organizations that will allow us to set permissions and controls to AWS accounts within an organization.

**3.** How can you convert a public subnet to private subnet?

**ANS:** First we need to check at the NACL level. A public subnet is associated with a route table that includes a route how the traffic is going to flow that will send all traffic (0.0.0.0/0) to an Internet Gateway that makes the subnet public. To convert it to a private subnet we should remove this default route. Then it’s become a private subnet.

**4.** What is the default route for any newly created route table?

**ANS:** For any newly created route table in an Amazon (VPC) the default route we called as the local route which is automatically created. This default route ensures that all traffic within the VPC is routed locally to the other instances in the same VPC and also it also establishes connections between instances within the VPC without any route configuration.

**5.** How would you ensure routes in the route table DO NOT use the local routes?  
  
**ANS:** We can create a custom route table and we can define the custom routes in this we can define where the traffic has to be routed with IP address Port and Port numbers and lastly we can associate subnets with route table.

1) Needs to know AWS OU restructure process.

An organization restructure is a change in a company business model and structure.

And restructuring can involve changes to the workforce or introducing new processes.

Here is some best way to restructure OU

1. Evaluate your current structure.

Before restructuring we need to evaluate our current organizational structure.

2. Develop a restructuring plan.

Create a restructure plan by looking management tools and estimating the cost and budget and creating the team to handle all these things.

3. Creating new OU or changing the existing one

4. Add some of the accounts.

5. Perform a test on a smaller scale.

A good strategy can be to implement restructuring in the smaller parts of a business before applying it to the entire organization.

6. Implement new organizational structure.

2 ) Needs to know AWS IAM restructure process.

Restructuring AWS Identity and Access Management involves making changes to your AWS IAM policies, users, groups, roles, and permissions.

I have experience in creating the users and attaching the policies and creating custom policies.

Here is some best way to restructure IAM

1. Review existing IAM structure.

Reviewing our current AWS Identity and Access Management structure which may include IAM policies, users, groups, roles, and permissions.

2. Creating new access control model.

3. Backup existing policies for safer side.

4. Create new IAM policies.

5. Update user and group and role permissions.

6. Test access to user weather assigned roles and permission are based on their roles and everything working fine.

3) Needs to know terraform import process.

Terraform import command is used to bring existing infrastructure resources under Terraform management. Which allows us to import the state of a resource that was created outside of terraform into your terraform configuration. Which helps us to manage that resource along with our Terraform managed infrastructure.

I have imported the resource using terraform import command.

Here an overview of the process to import an existing resource into Terraform.

1. Identify the existing resource.

First we need to identify the existing resource that we want to import into our Terraform configuration.

2. Create a Terraform Configuration.

Secondly we need to write the resource file means main.tf which is similar to resource we want to import.

3. Execute terraform import command.

Run terraform import command to import the existing resource.

terraform import (resource\_type).(resource\_name) (resource\_id) provide proper resource name and resource id and resource type which you want to import.

4. Then last we can execute terraform plan and terraform apply command to preview the changes and to apply the changes.

4. Programming skills.

1. Schell script.

2. Python (learning).

**SECTION 4: Programming Exercise :**

**NOTE :** Above SECTION 4 exercise is available in Github.