**Terraform Drift Detection and Management:**

1. **Clone the repository:**
2. git clone https://github.com/hashicorp-education/learn-terraform-drift-management
   * **git clone**: This command is used to create a copy of an existing repository from a remote server (in this case, GitHub) to your local machine.
   * **https://github.com/hashicorp-education/learn-terraform-drift-management**: This is the URL of the repository you want to clone.
3. **Navigate to the repository directory:**
4. cd learn-terraform-drift-management/
   * **cd**: Stands for "change directory". This command changes the current working directory to the specified path.
   * **learn-terraform-drift-management/**: This is the directory created by the git clone command.
5. **List the contents of the directory:**
6. ls
   * **ls**: This command lists all files and directories in the current directory.
7. **Generate SSH key:**
8. ssh-keygen -t rsa -C "balu123gamil.com" -f ./key
   * **ssh-keygen**: This command generates a new SSH key pair.
   * **-t rsa**: Specifies the type of key to create, in this case, RSA.
   * **-C "balu123@gmail.com"**: Adds a comment to the key, typically an email address.
   * **-f ./key**: Specifies the filename to save the key to.
9. **List the contents of the directory with detailed information:**
10. ls -lh ./
    * **ls -lh**: Lists files and directories with detailed information (like file size, permissions, etc.) in a human-readable format.
    * **. /**: Refers to the current directory.
11. **Initialize Terraform:**
12. terraform init
    * **terraform init**: Prepares the working directory for other Terraform commands by downloading necessary plugins and setting up the backend.
13. **Apply Terraform configuration:**
14. terraform apply
    * **terraform apply**: Applies the changes required to reach the desired state of the configuration, creating or updating infrastructure.
15. **List Terraform state:**
16. terraform state list
    * **terraform state list**: Lists all resources tracked by Terraform in the current state file.
17. **Create a new security group:**
18. export SG\_ID=$(aws ec2 create-security-group --group-name "sg\_web" --description "allow 8080" --output text | awk '{print $1}')
    * **export SG\_ID=**: Sets the environment variable SG\_ID to the output of the command.
    * **aws ec2 create-security-group**: AWS CLI command to create a new security group.
    * **--group-name "sg\_web"**: Names the security group sg\_web.
    * **--description "allow 8080"**: Provides a description for the security group.
    * **--output text**: Outputs the result in plain text format.
    * **| awk '{print $1}'**: Pipes the output to awk to extract the first field, which is the security group ID.
19. **Modify instance attributes:**
20. aws ec2 modify-instance-attribute --instance-id $(terraform output -raw instance\_id) --groups $SG\_ID
    * **aws ec2 modify-instance-attribute**: AWS CLI command to modify attributes of an EC2 instance.
    * **--instance-id $(terraform output -raw instance\_id)**: Specifies the instance ID, retrieved from Terraform output.
    * **--groups $SG\_ID**: Associates the instance with the security group ID stored in SG\_ID.
21. **Refresh Terraform state:**
22. terraform plan -refresh-only
23. terraform apply -refresh-only
    * **terraform plan -refresh-only**: Creates an execution plan to refresh the state without making any changes.
    * **terraform apply -refresh-only**: Applies the refresh-only plan to update the state file.
24. **Edit main.tf to add a new resource:**
25. vi main.tf
    * **vi main.tf**: Opens the main.tf file in the vi text editor to make changes.
26. **Add the following resource definitions to main.tf:**
27. resource "aws\_security\_group" "sg\_web" {
28. name = "sg\_web"
29. description = "allow 8080"
30. }
31. resource "aws\_security\_group\_rule" "sg\_web" {
32. type = "ingress"
33. to\_port = "8080"
34. from\_port = "8080"
35. protocol = "tcp"
36. cidr\_blocks = ["0.0.0.0/0"]
37. security\_group\_id = aws\_security\_group.sg\_web.id
38. }
    * **resource "aws\_security\_group" "sg\_web"**: Defines a new security group resource named sg\_web.
    * **resource "aws\_security\_group\_rule" "sg\_web"**: Defines a new security group rule for sg\_web to allow traffic on port 8080.
39. **Import the security group into Terraform state:**
40. terraform import aws\_security\_group.sg\_web $SG\_ID
41. terraform import aws\_security\_group\_rule.sg\_web "$SG\_ID"\_ingress\_tcp\_8080\_8080\_0.0.0.0/0
    * **terraform import**: Imports existing infrastructure into Terraform state.
    * **aws\_security\_group.sg\_web $SG\_ID**: Imports the security group with ID SG\_ID.
    * **aws\_security\_group\_rule.sg\_web "$SG\_ID"\_ingress\_tcp\_8080\_8080\_0.0.0.0/0**: Imports the security group rule.
42. **List Terraform state again:**
43. terraform state list
    * **terraform state list**: Lists all resources tracked by Terraform, including the newly imported ones.
44. **Apply Terraform configuration again:**
45. terraform apply
    * **terraform apply**: Applies any changes made to the Terraform configuration.
46. **SSH into the instance:**
47. ssh ubuntu@$(terraform output -raw public\_ip) -i key
    * **ssh ubuntu@$(terraform output -raw public\_ip) -i key**: Connects to the instance using SSH with the generated key.
48. **Test the setup:**
49. curl $(terraform output -raw public\_ip):8080
    * **curl $(terraform output -raw public\_ip):8080**: Sends a request to the instance's public IP on port 8080 to verify the setup.
50. **output:**

Hello World