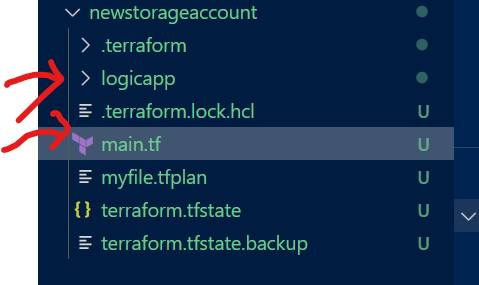
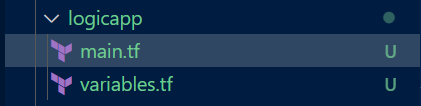
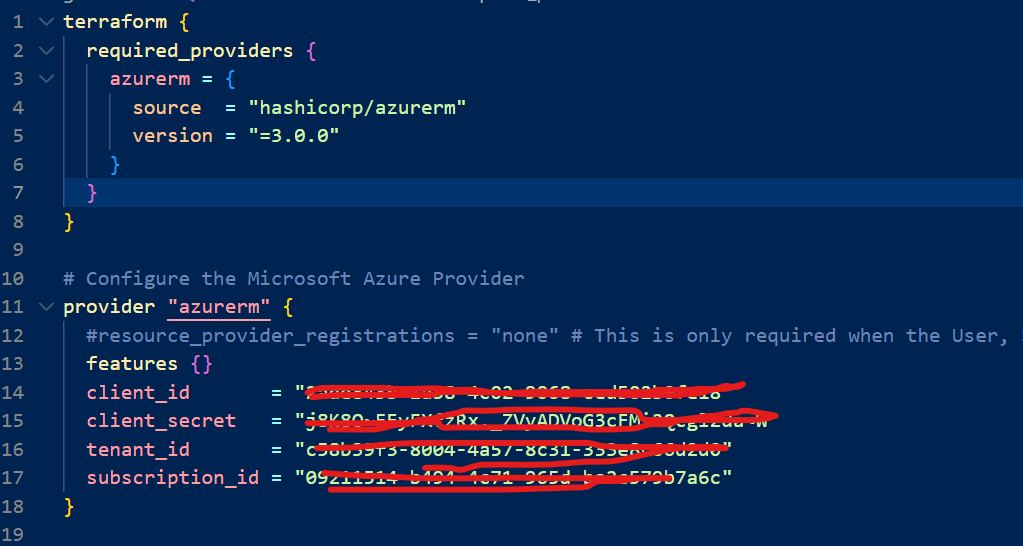
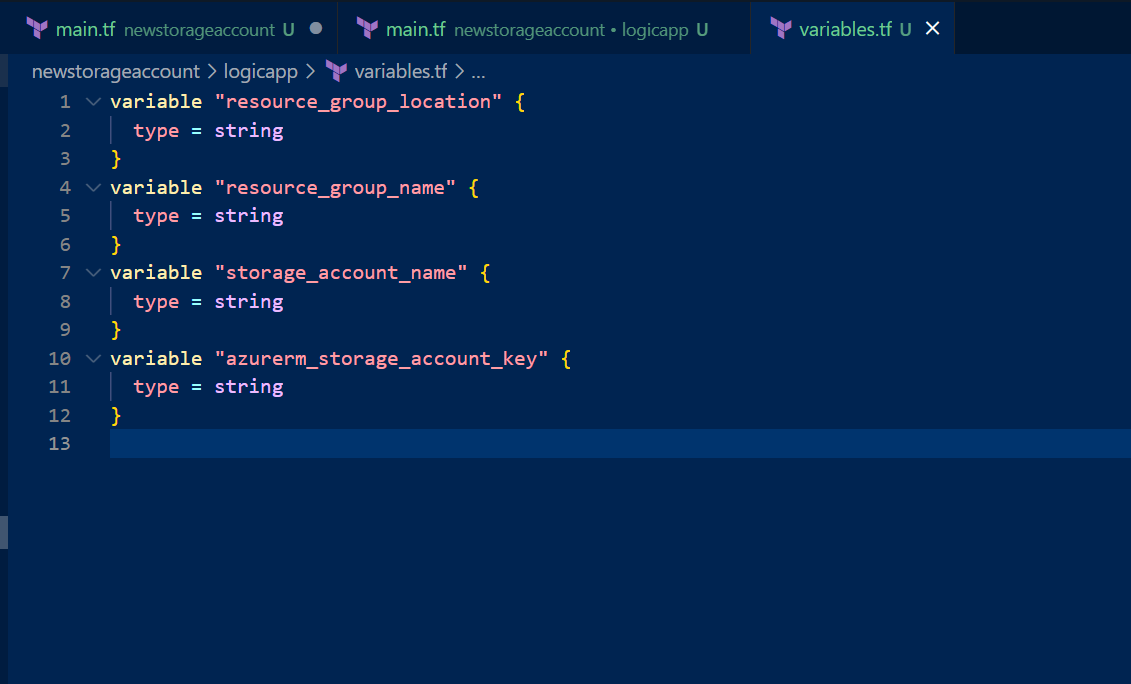
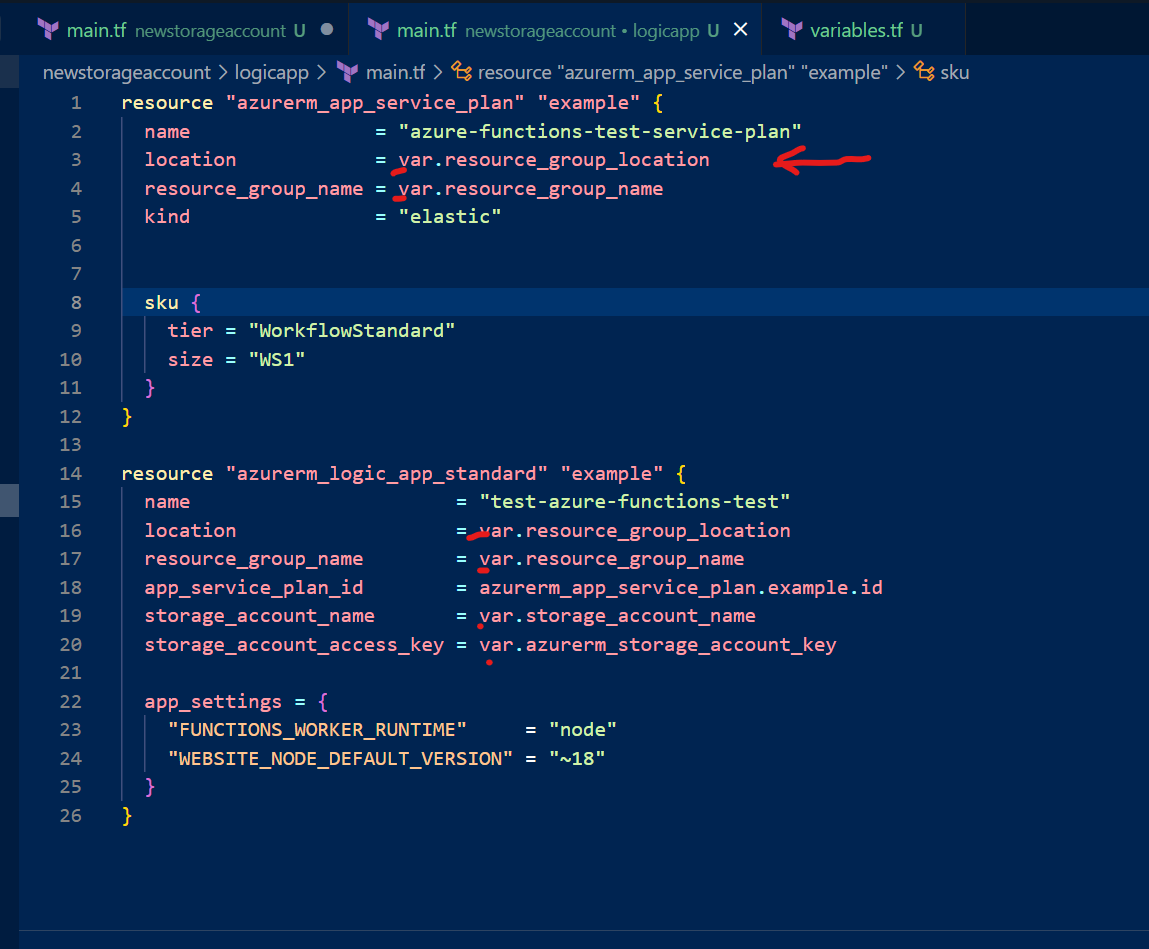
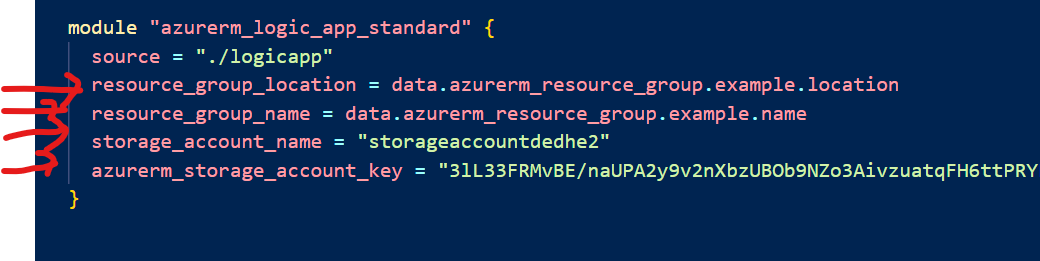
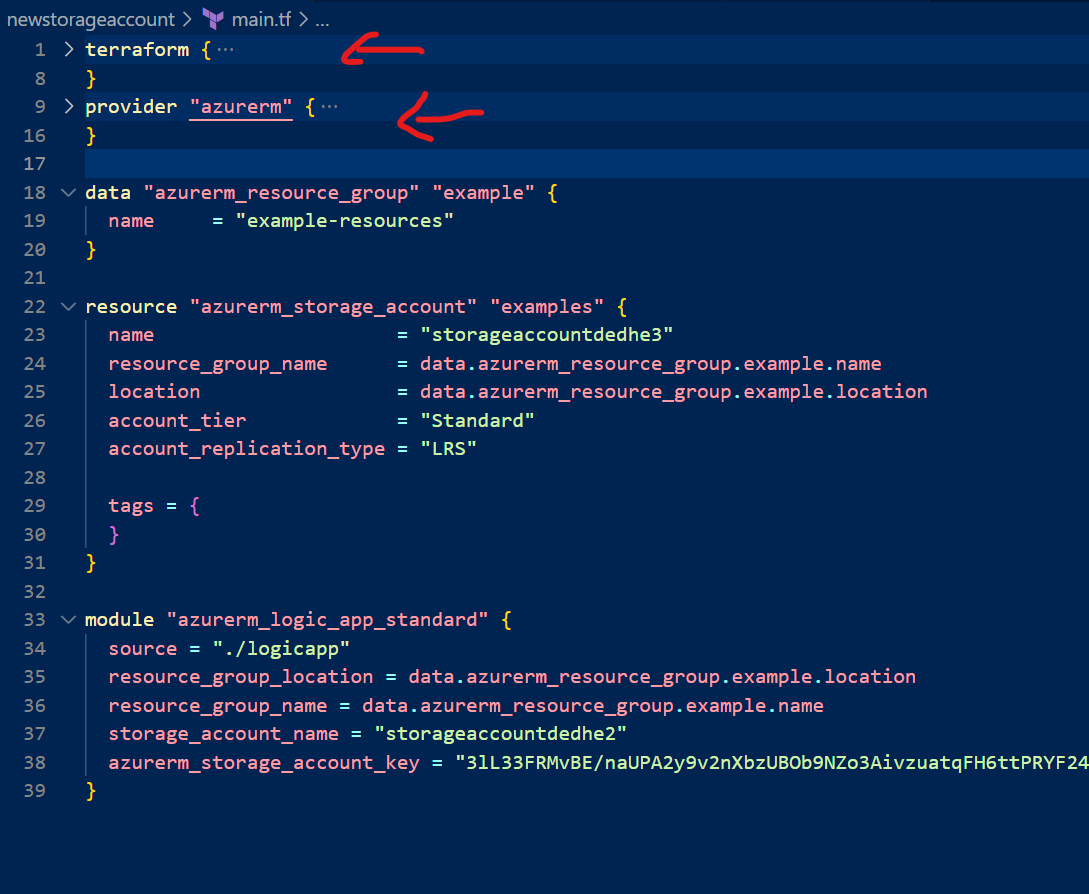
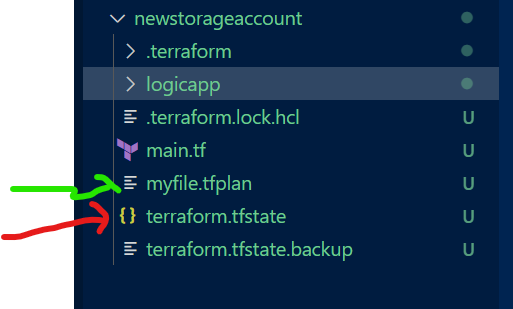
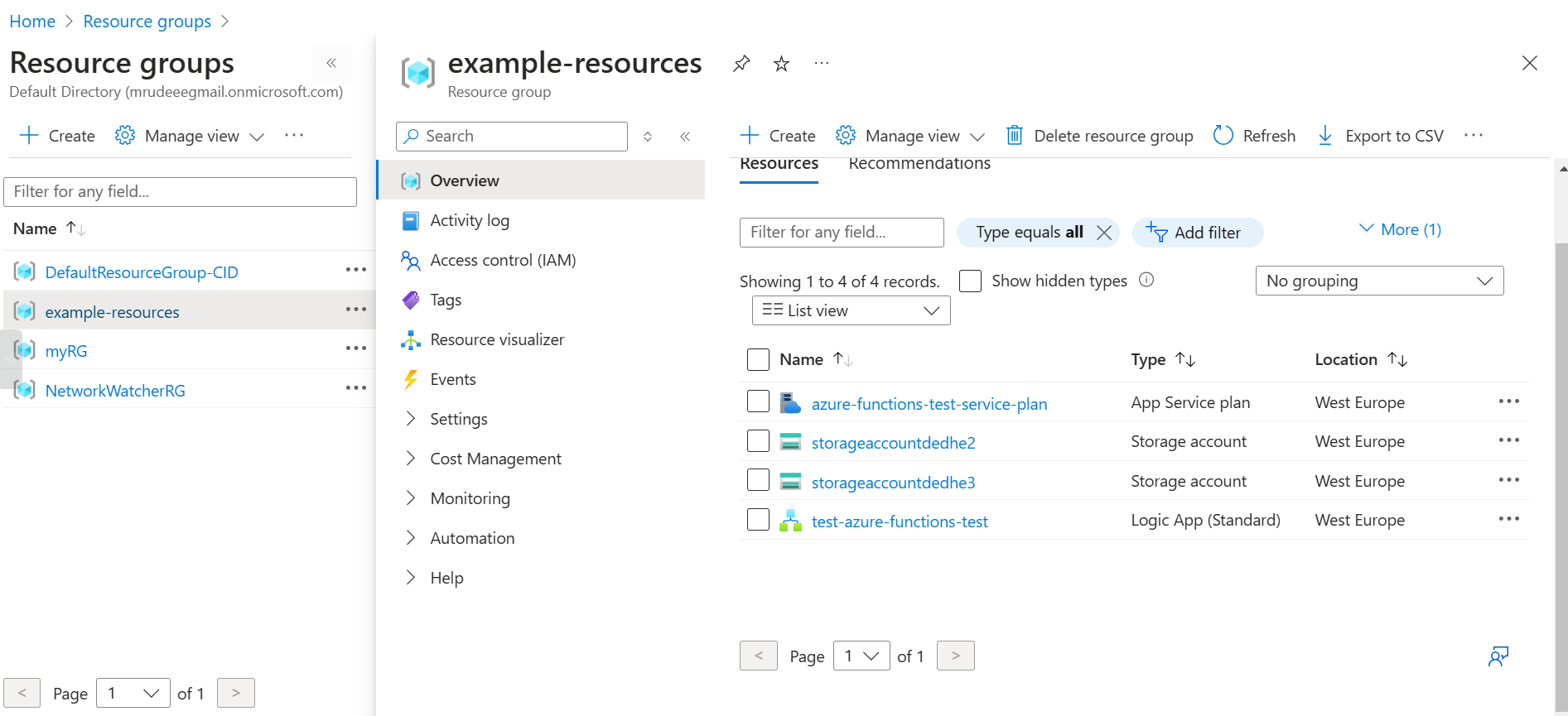
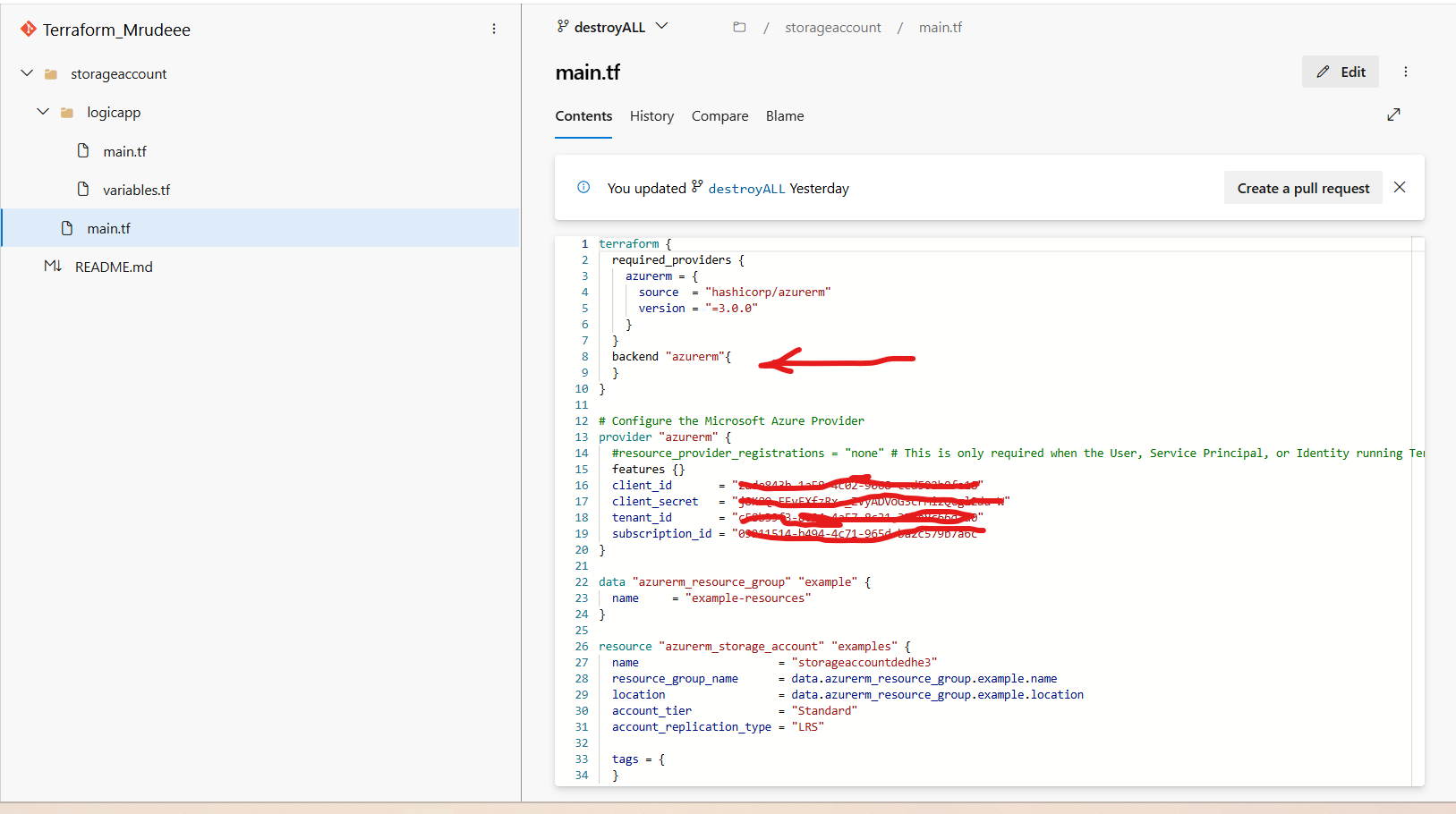
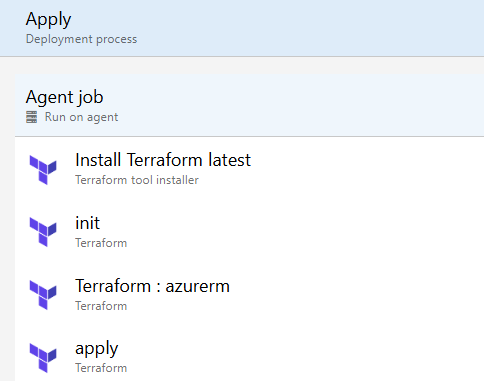
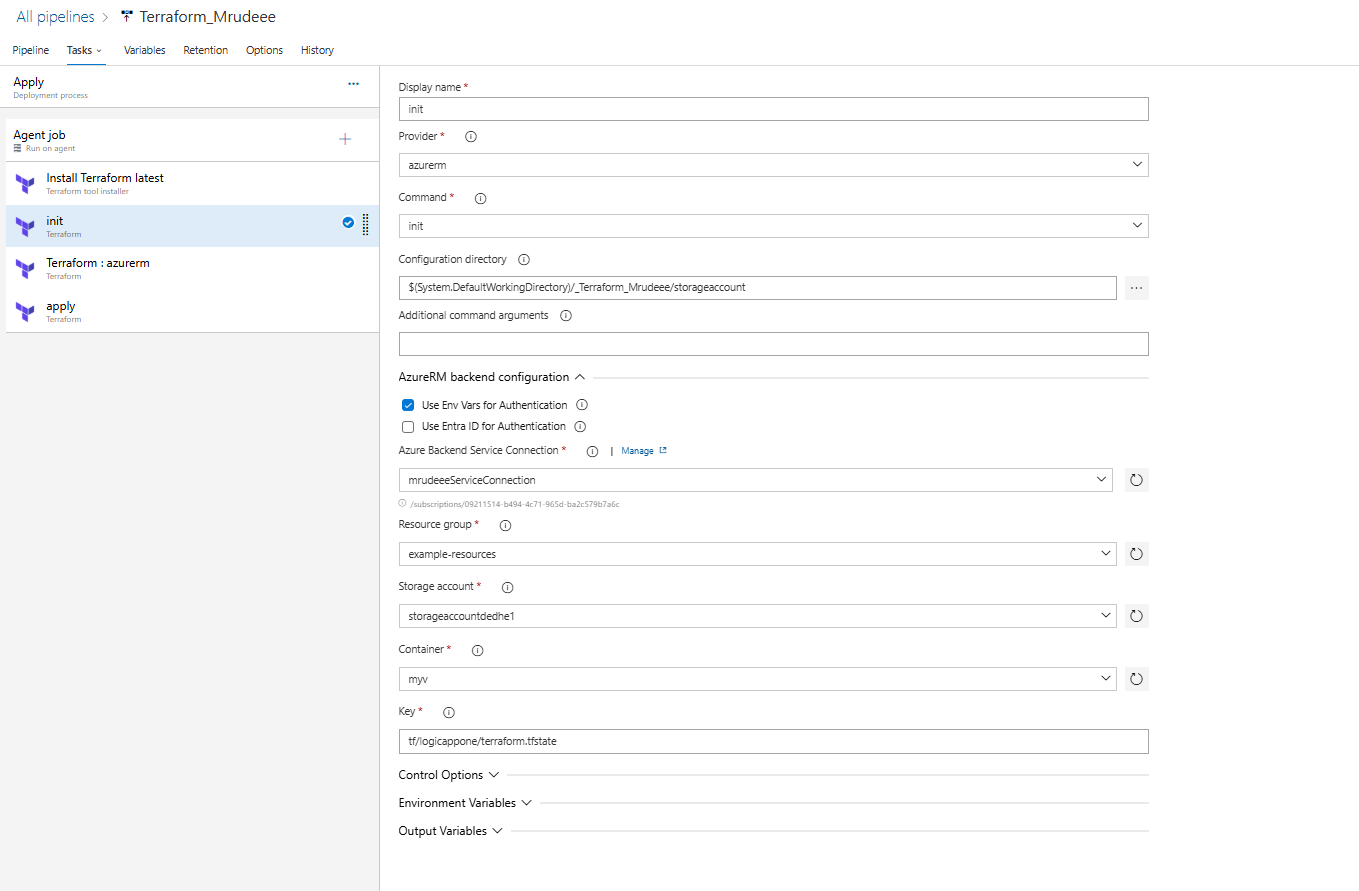
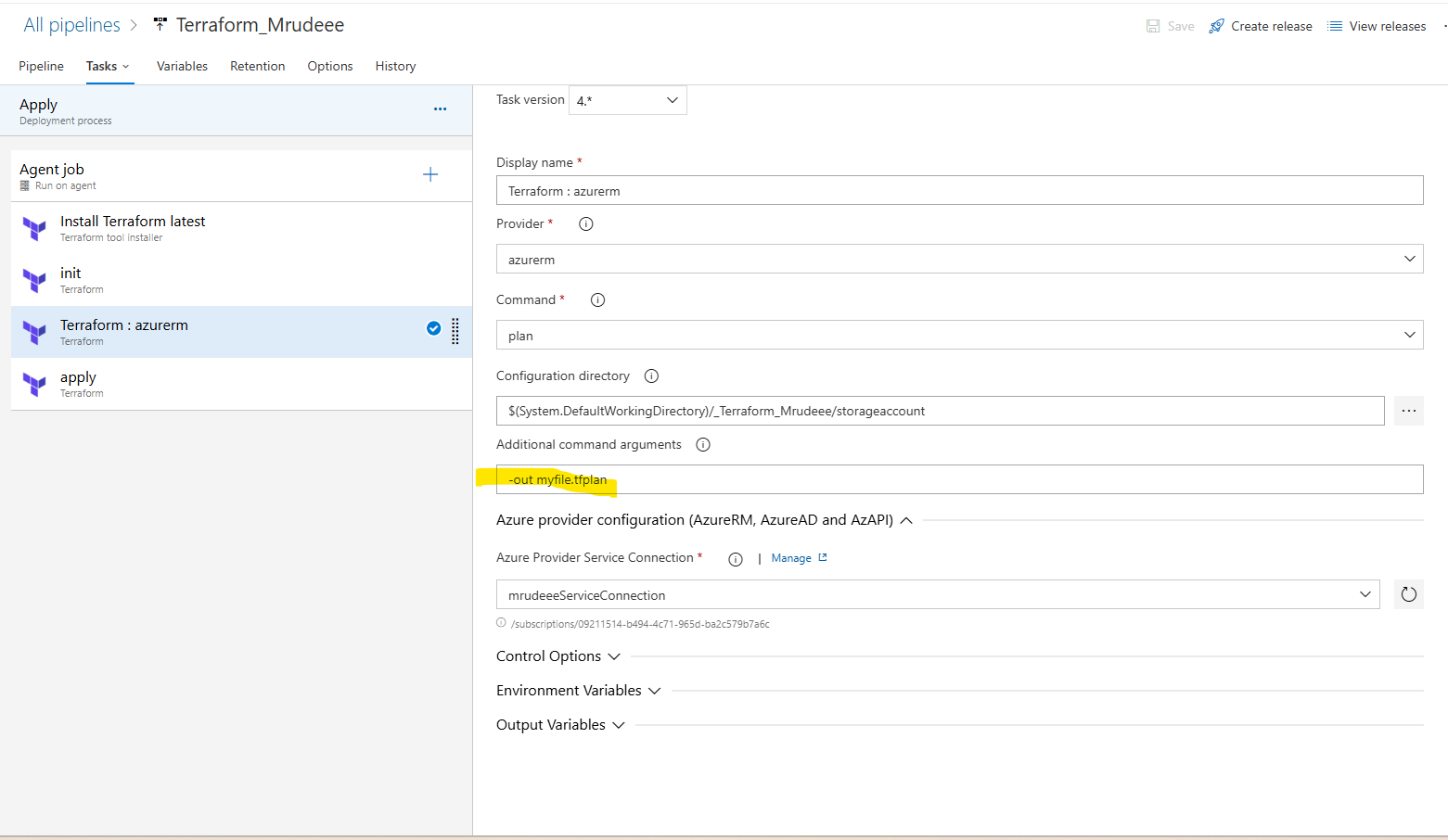
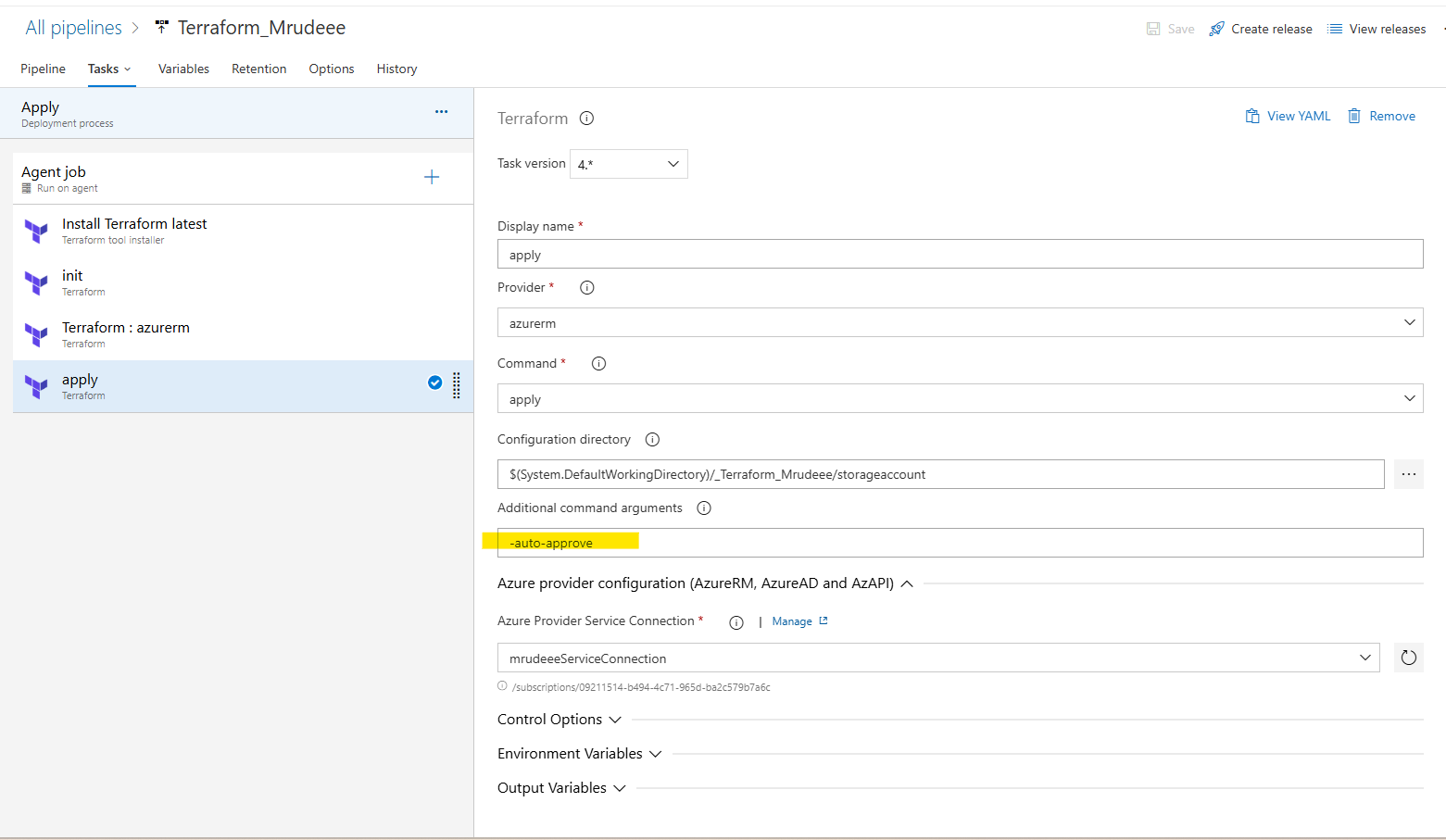
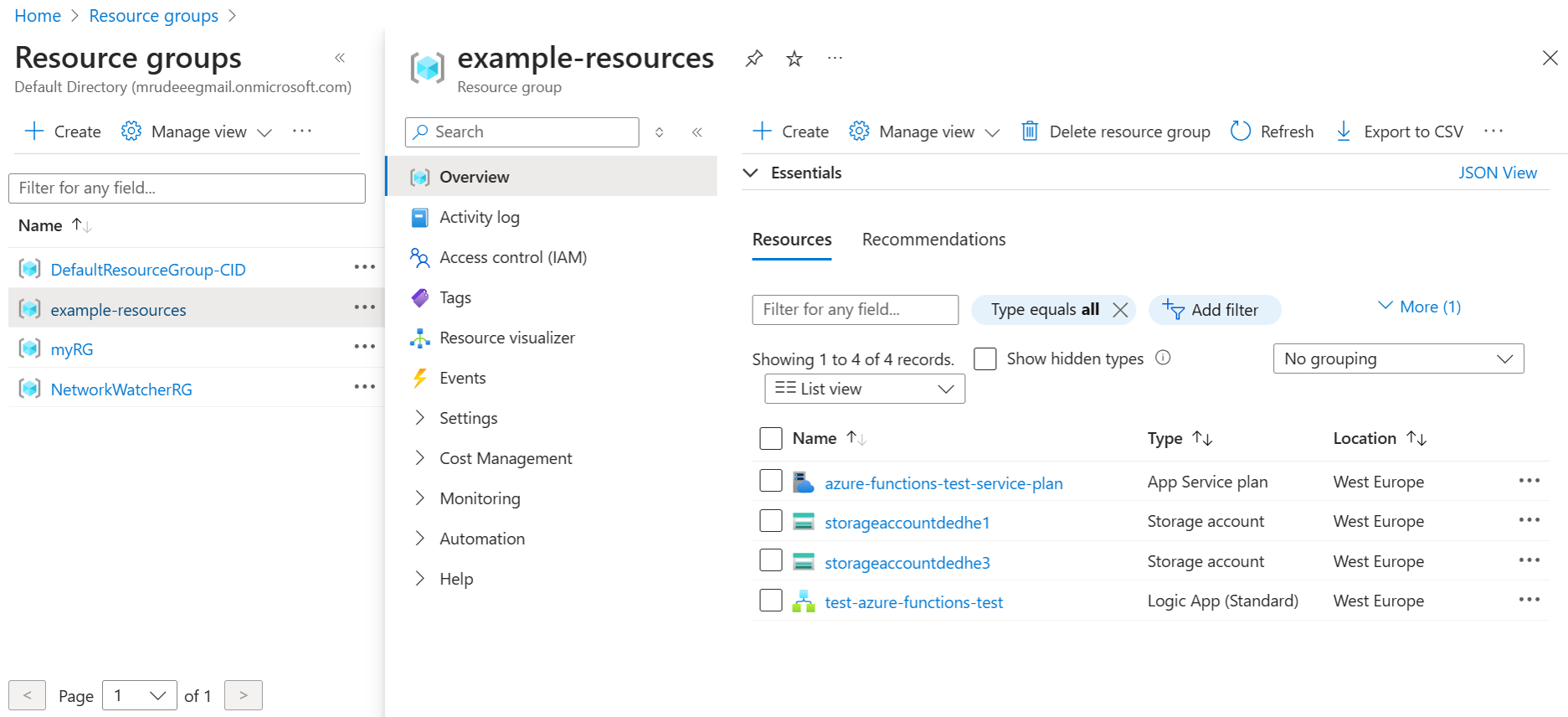
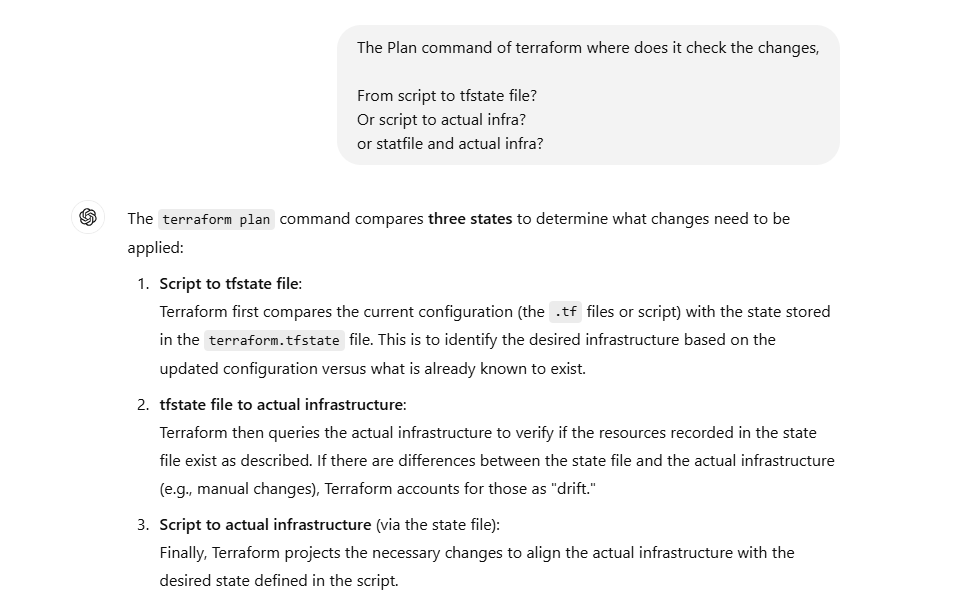
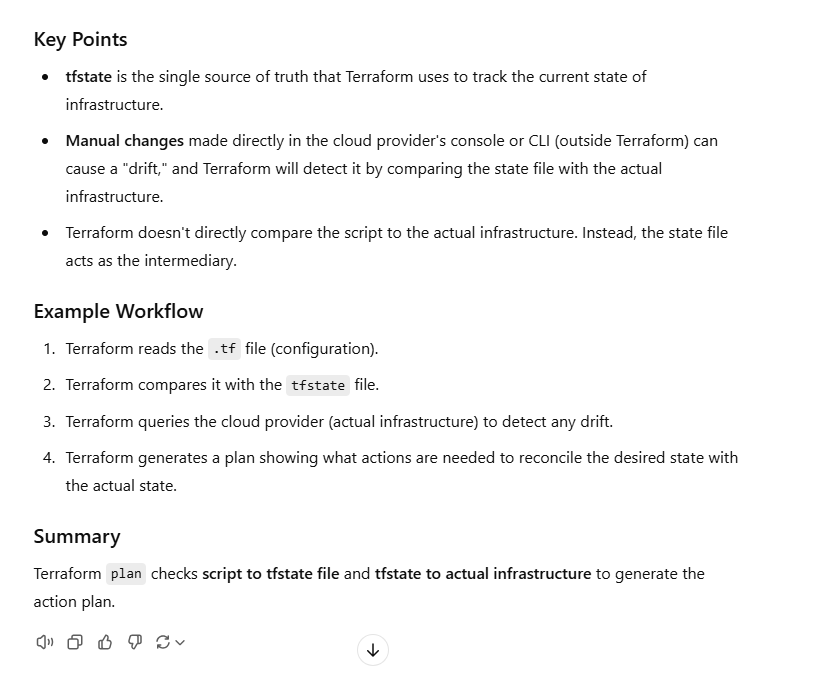
**Terraform Modules & Observations**

1. I have implemented Modules via both Vs Code & Pipeline. This SOP mentions both of them.
2. Terraform Modules via VS code –
3. Create one folder (root folder) in which your main.tf will be there. Inside that folder along side should be another folder. That folder will act as a module. See below, 
4. Now the module folder will also have a main.tf but it must contain variables.tf & optionally output.tf. 
5. The root main.tf must contain the provider blocks (and azurerm backend in case of pipeline structure as we would be using storage account to store the statefile). Also, the credentials of service principal details to connect. But the module main.tf should **not** contain these details. 
6. Now, as you know, the module main.tf & variables.tf should be like this. The variables.tf is mandatory because we need to insert them when we will call them from root main.tf. 
7. Once this is done, now you can call it from the root module.tf, (note, as discussed in earlier SOPs, you must have installed the Terraform extension in the VS code to be able to save file with tf extension). Below is how you can call the module. These pointed out are the same 4 variables which we have mentioned in the above variables.tf
8. Remember the data & resource are two types of declarative ways to initialize the resources in terraform. 1st data only takes the information of **already** created resources, while resource will create **new** infrastructure. Below is the complete root main.tf file. (I have minimized first two blocks) 
9. Remember once all these files are ready you will fire below commands in order in root directory (where your root main.tf module is stored. Also note that you need to have terraform.exe in the environment variables so you could access terraform from everywhere. (Please refer earlier SOPs to install terraform)
   1. **Terraform init** 🡪 To initialize the directory and install the plugins and modules (including our one)
   2. **Terraform plan -out myfile.tfplan** 🡪 This will create tfplan file, which terraform apply command will use to create infrastructure. Note tfplan file is not the state file. It’s just an intermediate file which terraform creates for its own purpose. (It is highlighted in green arrow)
   3. **Terraform apply myfile.tfplan** 🡪 This will actually create the infra and also a statefile (TFSTATE, marked in red arrow. Note, I have found that, even without running Terraform plan command if you run Terraform apply command, it will still create the infra, but it’s dangerous as there can be unexpected behaviors, since terraform apply will have no plan file for reference) 
10. Once you run this you can see, both storage account (root file output) & logic app with its component (module output) gets created. 
11. Terraform Modules via pipelines –
12. Now copy the same code in same way, in Azure DevOps repo, except one difference in root main.tf file, which is that you include backend block, else no statefile would be generated. 
13. Now, create a release, with below tasks, and note in same stage, plan and apply both should be there. Because plan file needs to be passed. If you have two different stages, then plan file is not passed between the two. 
14. Init configurations - 
15. Plan configurations – (Note the output file here I have outed, in earlier SOP, I haven’t, as it was very basic, but this is the correct way) 
16. Apply configurations – (Note, I have set auto-approve, because we don’t need to manually approve the planfile changes) 
17. Now when you run it, the same results are achieved. The tfstate file is also stored in the storage account that we have mentioned in the init task. Next thing is, I ran this, then all the infra got created. Then, I deleted (manually from CLI) only the logic app leaving other componenets intact. Then I ran the same pipeline again, this time the plan task discovered that only logic app needs to be created. And then only logic app was created. This is the use of plan file. Same result. 
18. Now, as I deleted the logic app manually in above step, I naturally got the question, that the changes did not automatically reflect in the TFSTATE file stored in the stroage account, then how come terraform perfectly understood to only create the logic app? How does the plan command actually work? Does it compare changes from the script to the tfstate file, or does it compare it with actual infra? So below is the best answer by ChatGPT.  
19. I guess, this much understanding of Terraform working is enough for you to build further concepts, like how to use count, foreach, azuread provider and some more. I would also like to mention that there are few tools online which would create whole tf file directly for you, if you give them what is the architecture of infrastructre, like <https://www.brainboard.co/>
20. Other SOPs are made on complex concepts, please go through this one for any doubts.