Day - 1

09 December 2024 10:13

1. Devops & How it work?
2. How it is different from the Dev. & oper.
3. SRE
4. Devops vs SRE
5. Terraform.
6. Terraform Basic Commands.
7. Lab setup.

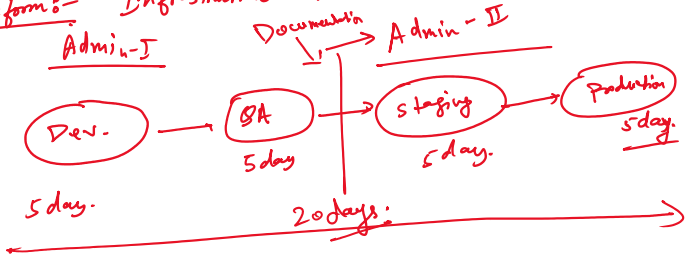
---

1. **Devops:-**

SRE (Site Reliability Engineer):-

① Metrics → CPU ≥ 80%.
② logs. → Detail with timestamp.
③ Traces →

Terraform:- Infrastructure as Code (IaC)

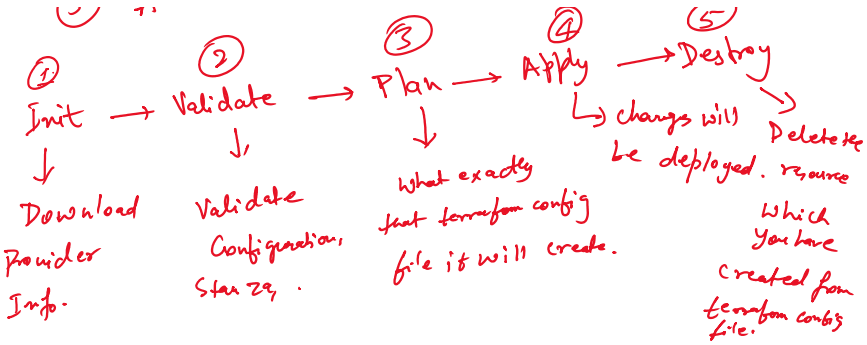


① Time Consumption.
② Scale up / scale down.
③ Reuse Code.
④ Resource.

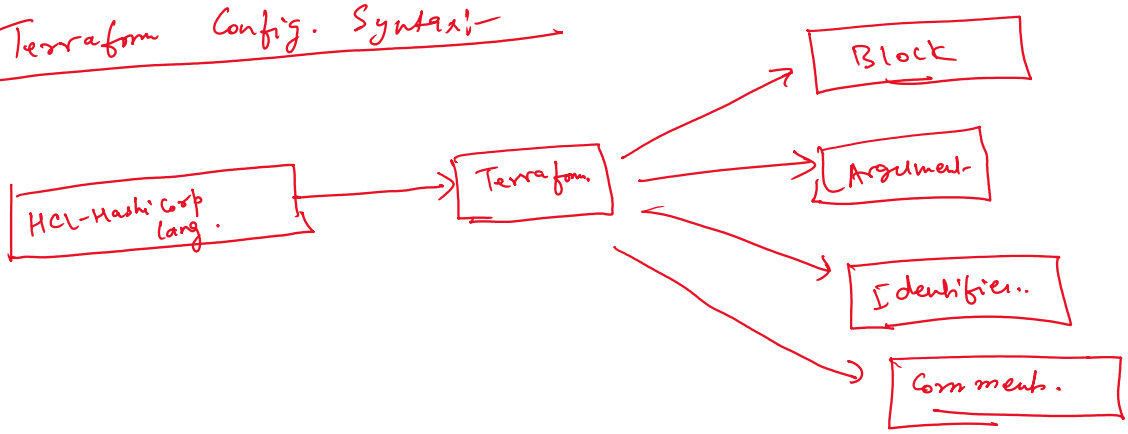
Adv. of Terraform:-

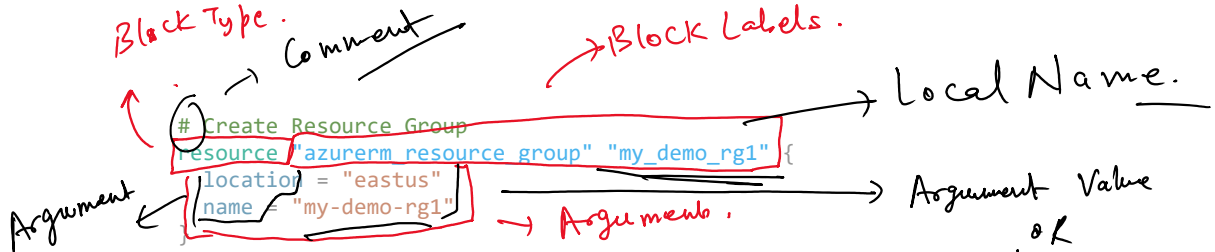
① Visibility → Code clear reference.
② Scalability → GUI →
③ Stability → Write Once, Reuse code Multiple times
④ Security → Defined template.
⑤ Audit → Monitoring the Activity of Terraform.

① Validate → ② Plan → ③ Apply → ④ Destroy → ⑤



Terraform Config. Syntax:-





Top level Block & Block inside Block.

Top level Block - Resource, Provider.

Comment → # → 1

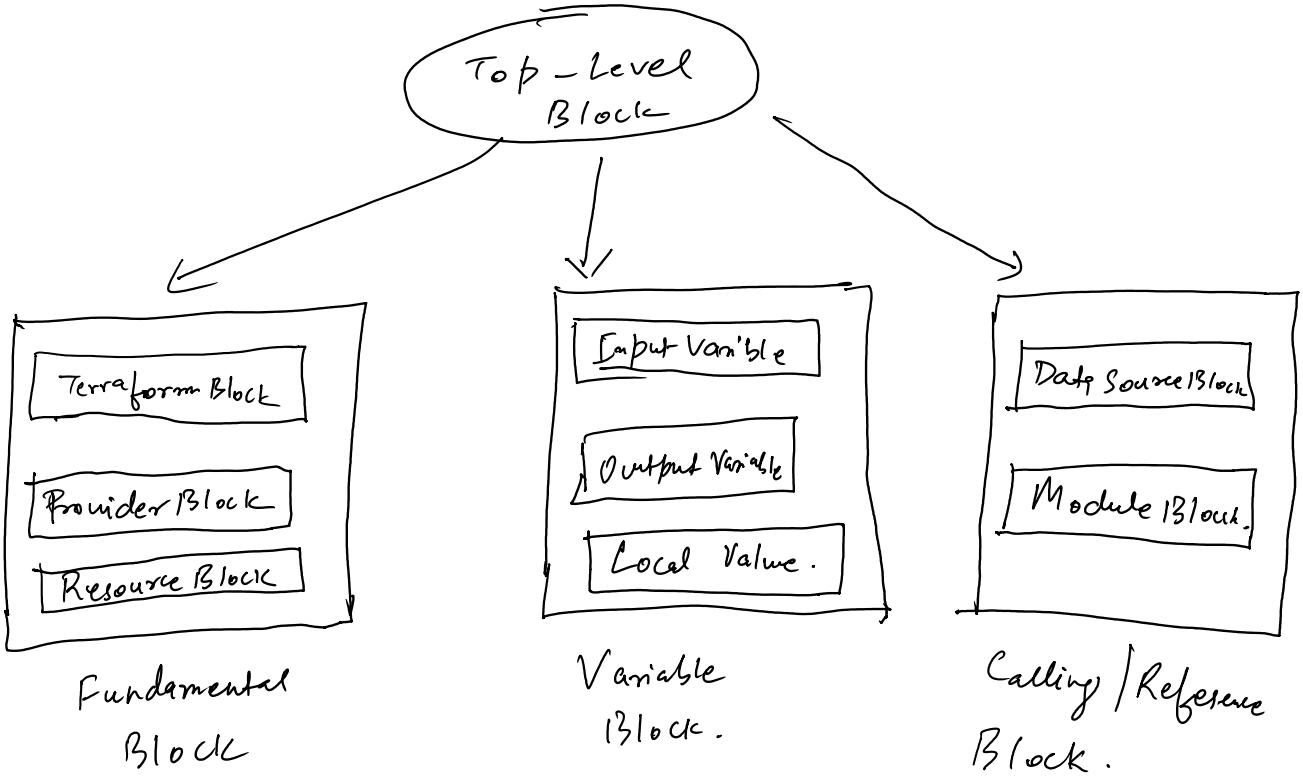
/* */ → 2

To comment the line of code .tf file

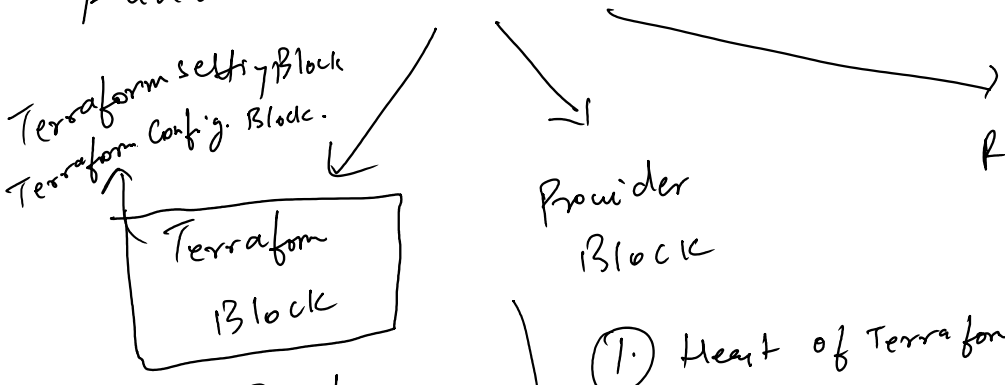
* Top level Block:- That is used outside any block in tf Config. file.

Top block → resource.

Top block → resource.



Fundamental Block :-



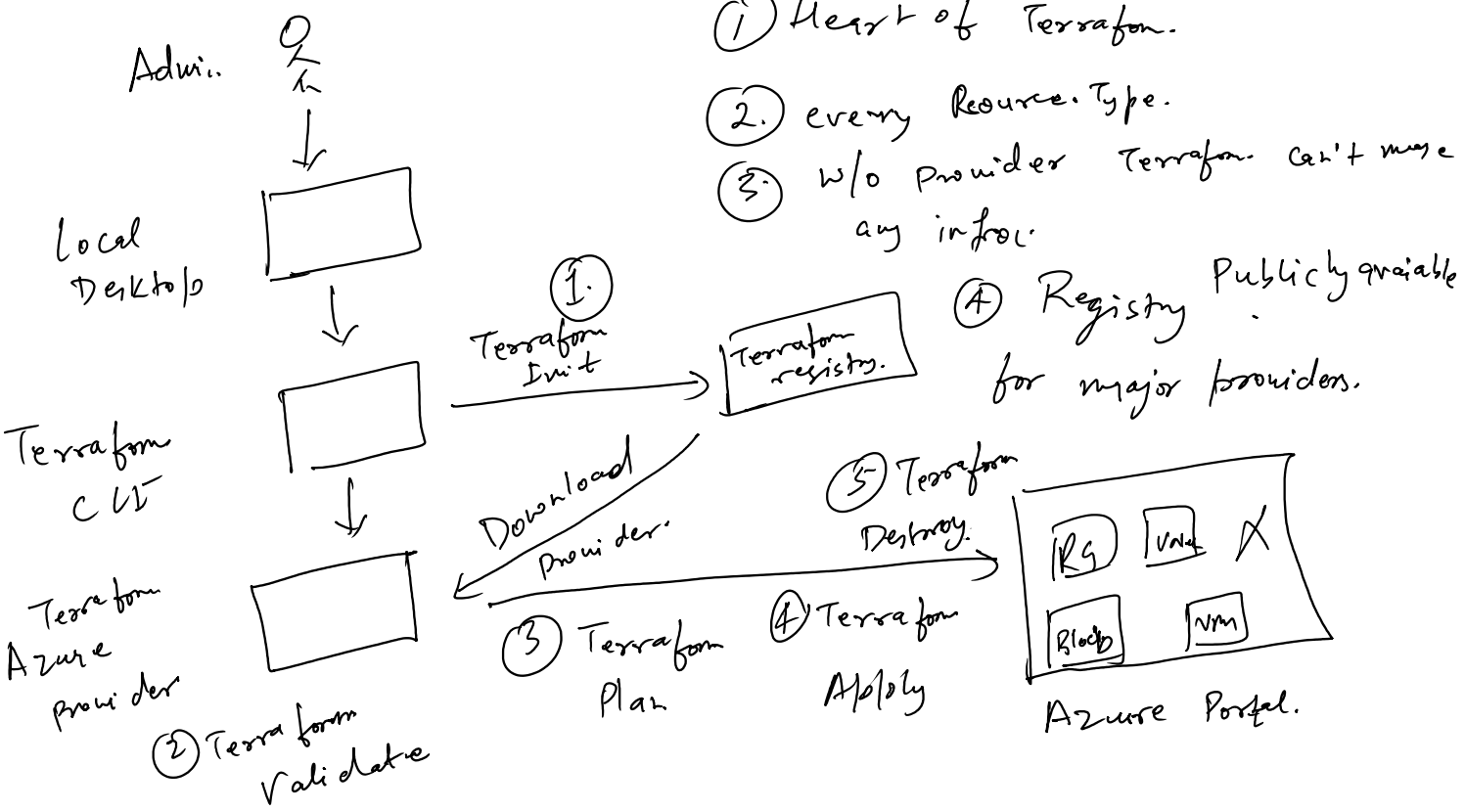
(1) Config. Terraform.
(2) Version of Terraform.
(3) Provider Req. & Version.
(4) Terraform Backend.

(1) Heart of Terraform.
(2) Terraform to connect Provider Block.
(3) Install provider config.
(4) Root Modules.

(1) Info. / object initialize in service / provider.
(2) How?
(3) Action!

Terraform Provider :-

Terraform



1. Heart of Terraform.
2. every Resource Type.
3. W/o Provider Terraform can't manage any infra.
4. Registry Publicly available for major providers.

Provider Block:-

1. Provider Requirement.
2. Provides Configuration.
3. Dependency Lock file.

registry.terraform.io / hashicorp / azurerm.

[<hostname>] / <Namespace> / <Type.

Local Name.

```
# Terraform Settings Block
terraform {
  required_version = ">= 1.0.0"
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = ">= 2.0" # Optional but recommended in production
    }
  }
}
# Configure the Microsoft Azure Provider
provider "azurerm" {
  features {}
  subscription_id = "#######"
}
```

Terraform Registry.

- Official
- Verified
- Community
- Archieved.

Version = " >= 2.0.0 "

Version = 4.13.0

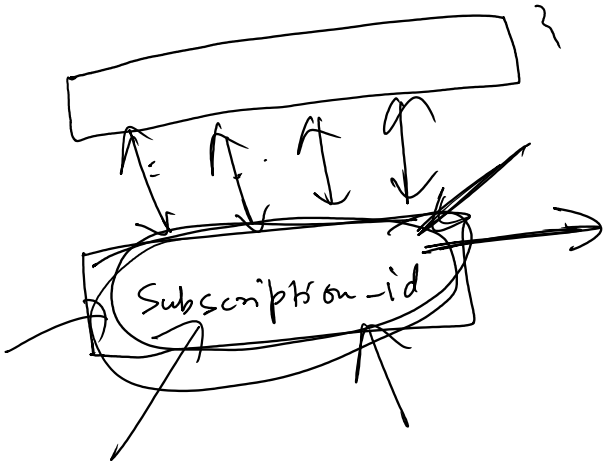
Constraint = >= 2.0.0

Multiple Providers

1. Multiple Config. for the same provider.
2. Multiple Region for a same provider.

```
provider "azurerm"
{
  features{}
  alias = "provider2-WestUS"
}
```

Resourse "rg" "name-rg1" <prov.name>.<alias>
{
provider = azurerm.provider2-WestUS"



* Dependency lock file :-

.terraform.lock.hcl → Version Control.

(1.) Not Available; create most latest lock file as per .tf Config. file.

(2.) Every time same version is been used to run your code across your team.

(a) random.

(b) storage.

azurerm
V 1.44.0

account_encryption_type = "Microsoft.storage"

.terraform.lock.hcl → 1.44.0

Error → V > 2.0 ⇒ V 4.--- Don't support error

Day - 2

10 December 2024 10:17

1. Resource Syntax & Behaviour.
2. Meta Argument
   - depends-on
   - Count
   - for-each
   - lifecycle
3. Input Variable
4. element, length, slot.
5. Output Variable.

Resource Behaviour :-

Terraform Resource
- Create
- Destroy
- Update in-place.
- Destroy & recreate.

Version → RG → VNet → subnet → PublicIP → Network interface

Meta Arguments
- depends-on
- for-each.
- Count
- life cycle.

1) Depends-on :-

a) hidden Resources.

b) Explict Dependecy

c) All resource Block & Module.

Use Case:

1) Random String.
2) Az- resource group.
3) Az VNet
4) Azure Subnet
5) Azure PublicIP
6) Azure Nic

(2) Meta Argument - for_each :-

(1) Map or set of string. One instance for each members of that map.

(2) each.key
each.value.

rg-2 = "USEAst"
rg-3 = "US West"

toset for_each = toset["Vivek", "amra"])

Vivek

each.key =
each.value = amra

for_each = {
dev = "myapp1"
}

each.key = dev
each.value = myapp1

toset() function :-

1. Convert arg. to set value.
2. toset(["Bang", "Mumbai", "Delhi"])
3. Convert toset(["Bang", "Mumbai", 123, 456])

"123", "456", "Bang", "Mumbai"

4. Convert toset([1, 4, 5, 7, 8])

1, 4, 5, 7, 8

(3) Meta Argument : Lifecycle.

(1) Create_before_destroy.

(2) Prevent_destroy.

2. Prevent_destroy.

3. ignore_changes

1. Create_before_destroy :-

Destroy → re create

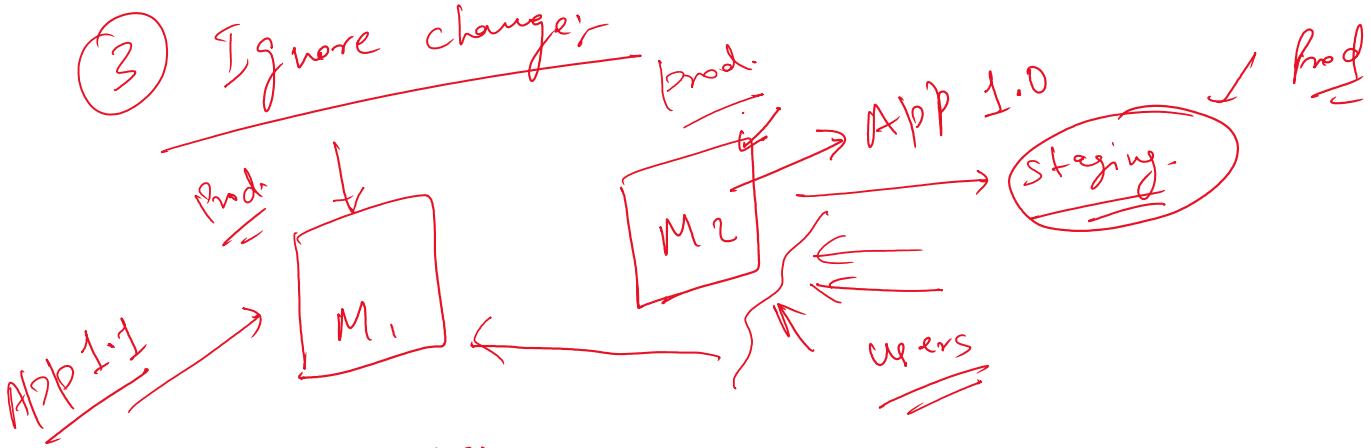
Arg. nested in any of the resource block.

```
lifecycle {
    create_before_destroy = true;
    {
```

2 Prevent_destroy :-

```
lifecycle
{
prevent_destroy = true;
}
```

3 Ignore changes :-

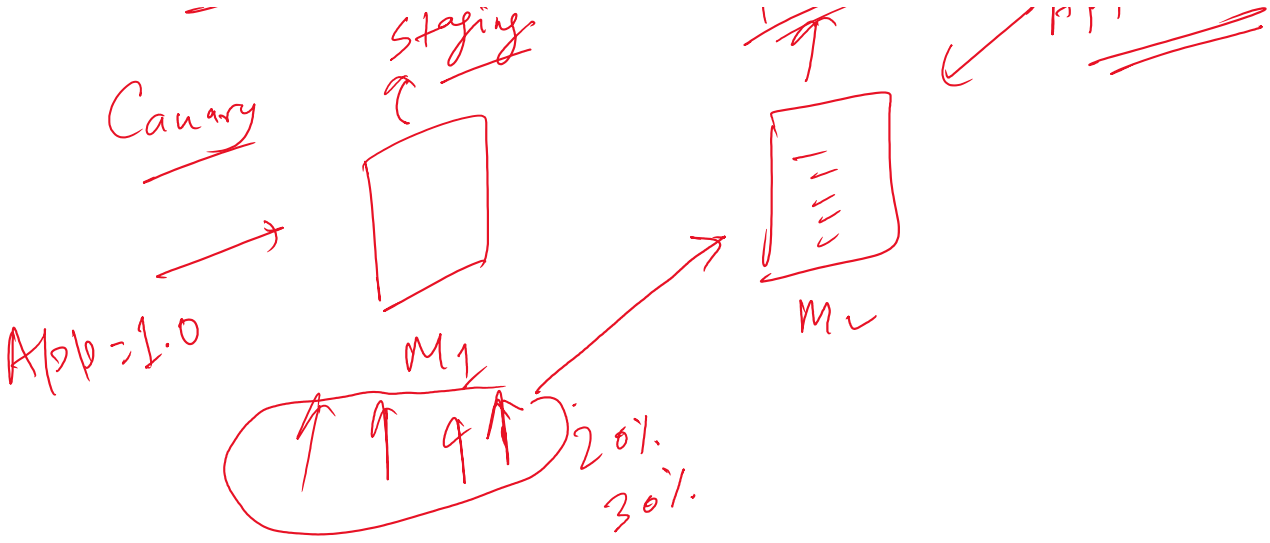


Blue-green

Staging

Prod.

App = 1.1




```
lifecycle {
    ignore_change = [ tag, —, — ]
}
```

④ Meta Argument - Count:-

*** (1) Cannot use the for_each & count both the Meta Arg. in one resource Block.

(2.) Count = whole No.

(3) No. of infra.

Count = 2

Count.index = [0, 1]

*Input Variables:-

Define | Initialize the Variable.

.tfvar → Override the Variable | default value in the Variable stanza.

.tfvar → Over...

defined in the Varia...
vic-variable.tf

terraform.tfvars
└ Value of the Variable, it will override the default value.

* Input Validation →

1. length()
2. substr()
3. Contains()
4. lower()
5. Regex()
6. can()

Day - 3

11 December 2024 10:27

1. Input Variable - Sensitive Values.
2. Output Variable.
3. Local Values.
4. Conditional Expression.
5. DataSource.
6. Azure static Website
   - Portal.
   - terraform..
7. Child Module.

---

1. Input Variable - Sensitive Data.
2. Output Variable:- In the output Variable

[for o in var.list :o.id]

Var.list[*].id for(i=0; i<10; i++)

Splat expression → more concise way to express a common operation.

Local Values:

1. Avoid the repeation.
2. Any modules.
3. DRY → Don't repeat yourself
4. Shorten the calling.
5.

```
locals {
    service :"audit"
}
```

```
locals {
  tag = {
    env. = dev
  }
}
```

Conditional expression:-

```
if ( ___ )
{
  true;
}
else { false; }
```

Condition ? true : false.

```
if (a>b)
{ return a;
}
else { return b; }
```

a>b ? a : b.

Azure Static Web site :-

1. Version.tf
2. main.tf
3. Variable.tf
4. terraform.tfVars
5. Output.tf

1. Version.tf → Provider → azurerm, random.
2. main.tf → Resource group, storage Account
3. Variable.tf → list of variable
4. terraform.tfVars → Variable Values.
5. Output.tf → plan → Value captured.

Child Module

1. Child Module:-

Root
↳ child
(1) Inheritance.
(2) call Block from one tf project to other tf Project.

Block
module " _____ "
{
source = " _____ "
}

Module Block:-

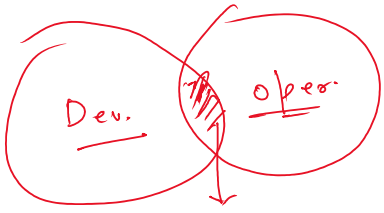
Module-block-project
azure-static-website.
source
variable.tf
main.tf
output.tf
Version.tf
→ Module
r1 → root
r1- version.tf
r1- output.tf
r1- static-website.tf
r1- variable.tf

r1- static-website.tf → module " n1 " {
source = " ./ " → ./azure-static-website
module " " ↔ source → relative
(module)
output.tf
Value = var. azu- resou- id
main.tf
resource block
attribute name
resourcetype
rg - locatnaca - id -

r1- output.tf
value = module . module-name . resource_id.
rb . LN . Dan

Azure Devops:-

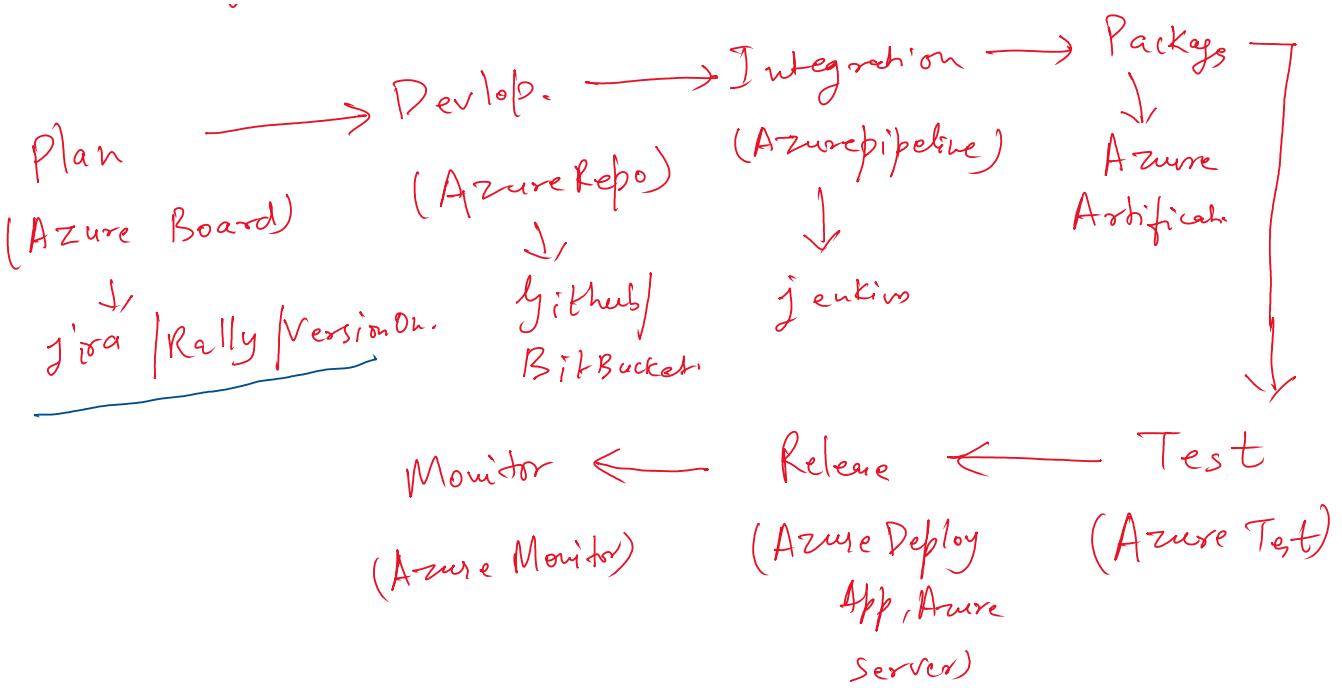
Dev + oper.
↓
Agile



Devops

jira / rally → Azure Board.
Git / BitBucket → Azure Repos.
Jenkin → Azure pipeline.

Devlob. → Integration → Packages



AKS
(Azure Kub. Service)
Operation
Ansible, Kub, Chef

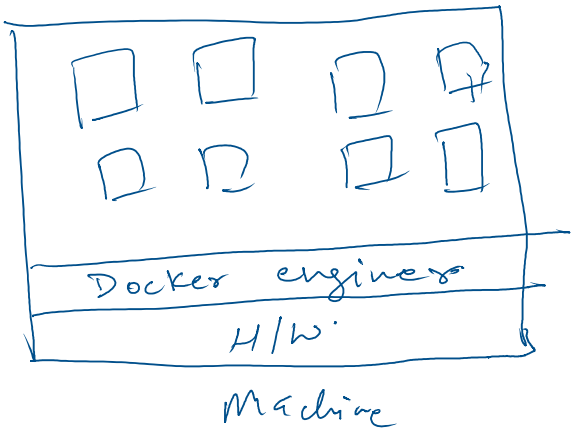
Build → Maven, sbt

Test → jmeter, Junit

Release → jenkin, Codeship

Deploy → Docker, AWS

Monitor → splunk, Nagios, DataDog.



Trigger

Trigger →

stage →

Stage → VM → Agent Pool

Agent Pool

Login vm → install Agent Pool

./config.cmd → User →

Access Token.

./run.cmd

install docker.

Relogin → sudo systemctl restart docker

docker pull hello-world

./run.cmd.

Day - 5

13 December 2024 10:33

1. Virtual Machine using terraform.
2. Sentinel Policies.
3. RBAC.
4. Remote Backend Concept & state lock.
5. Terraform Import
6. Demonstrate how to setup pipeline for terraform task.
7. Azure Key Vault.

1. Terraform state:- → Detail of execution.

terraform.tfstate after you execute the terraform apply.

State lock will lock the .tfstate file to avoid the multiple change parallelely. It will push one change at a time.

RG → Storage Account → Container → tfstatefile

initialize at .tf file in our terraform end.

2. Terraform Import:-

1. Able to import existing infra.
2. Resource which is already available & it's been used by Terraform.
3. Infra. → Terraform.

terraform import azurerm_resource_group.vkrg2 /subscriptions/<Subscription_ID/resourceGroups/vkrg2

terraform import azurerm_resource_group.example /subscriptions/<SUBSCRIPTION_ID>/resourceGroups/<RESOURCE_GROUP_NAME>

* Sentinel Policies:-

1. Allowed-providers.sentel.
2. enforce-mandatory-tags.sentel
3. limit-proposed-monthly cost.sentel
4. Restrict-vm-publishers.sentel.
5. restrict-vm-size.sentel.

sentinel.hcl.

5. restrict - v...

Sentinel.hcl.

Enforcement level:-

1. Advisory.
2. Soft - Mandatory
3. Hard - Mandatory

Common function:-

1. tf config - function.
2. tf plan - function.
3. tf run - function.
4. tf state - function.

Sentinel.hcl

1. tf-plan-function
2. enforcement level policy wise

module

ent.policy ('tfplan')

Allow.policy

restrict.policy

module "____"

{

source = "____"

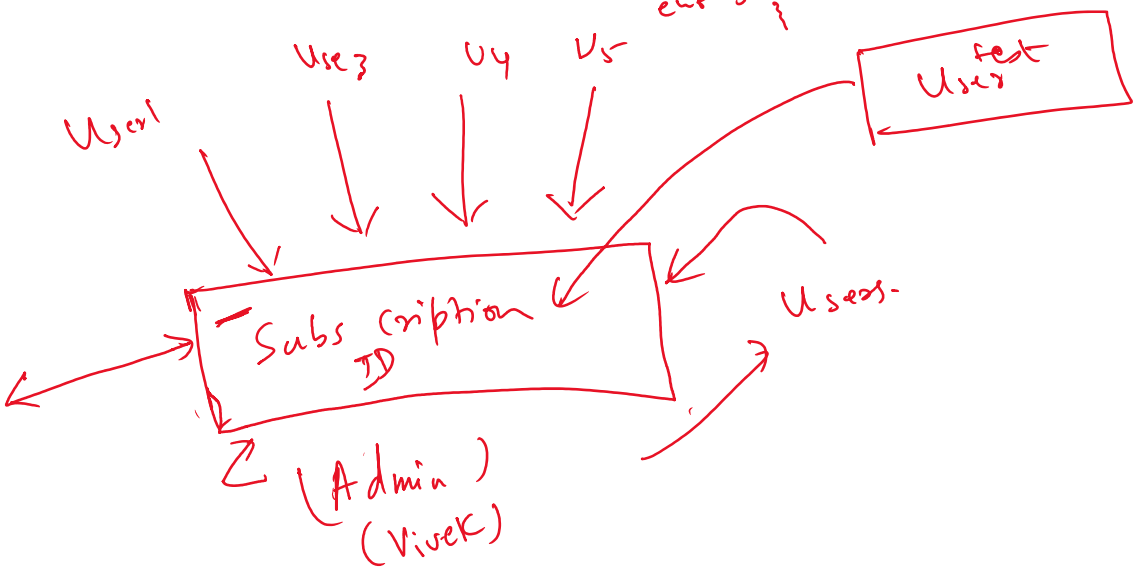
}

policy "____"

{

ent type = "adv/sft/hard"

}



VM:-

1. Variable - Input / Output
2. Local.
3. Backend
4. terraform.tfvars.
7. VM

(2) Local.
(3) Backend

(1) RG.
(2) Vnet.
(3) Subnet

(4) Public IP
(5) Virtual NI
(6) NSG
(7) V...

Tag → Conditional statement
↳ dev →
Prod →

(4) Lifecycle.

(1) Meta Argument → (1) depends
[RG → Vnet → Subnet] (2) for_each.
(3) Count.

(1) Depends :- public IP

(2) for_each :- Var. —— [*]. id

(3) count :- Count = 2

Count. Index [0]
[1]

(4) Lifecycle :-
Alter
(1) Create then destroy
(2)