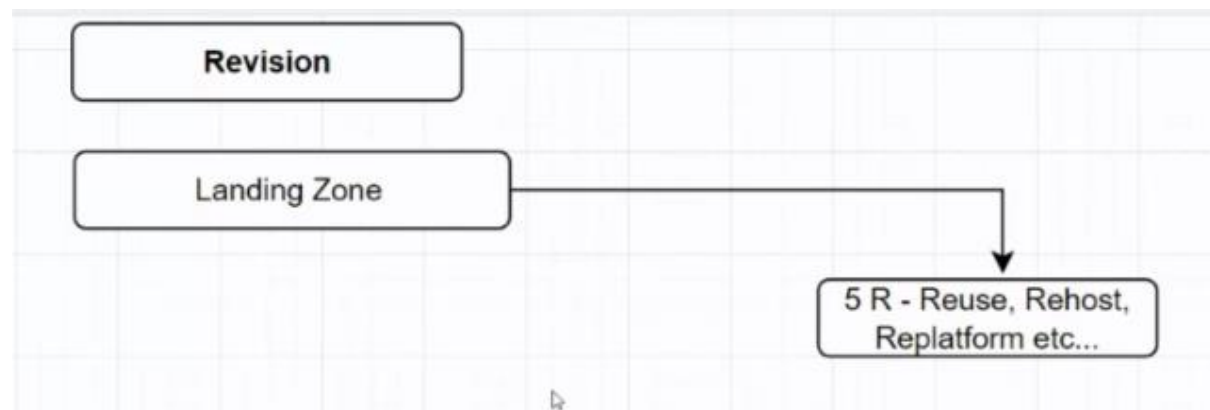
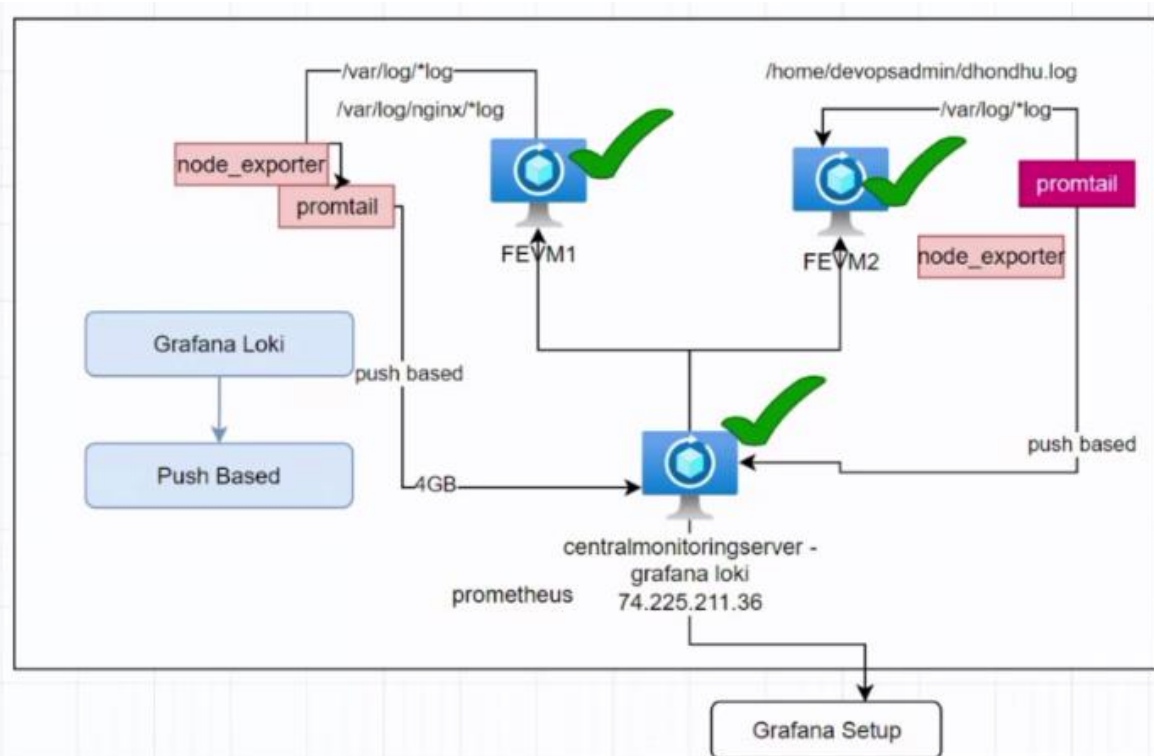


DEEP DIVE IN STORAGE ACCOUNT

1) In monitoring, We had created 3 machines. On 2 machines we had metrics and logs. For metrics we put node exporter. For logs we put promtail. After putting we exported the logs and got stored on central server. Then using yaml we put the value. After storing we shown everything on grafana.



Revision

Difference Between Terraform, ARM Template, and Bicep

Terraform String, Number

Terraform List - Chokor dabba wala - ["one", "two"]

Terraform Map - Ghunghurale Bracket wala {} -

```

{
  rg1 = {
    name = "rg1"
    location = "eastus"
  }
}
    
```

Terraform Foreach + Map concept ke saath Rg, Vnet, Subnet, Keyvault, sab bna lia abhi tk..

+++++

AGENDA - DEEP DIVE IN STORAGE ACCOUNT

```

environments > dev > terraform.tfvars > subnets > subnet3 > vnet
17 subnets = {
18   subnet1 = {
19     rg_name      = "rg-dev-zelectric"
20     vnet_name    = "vnet-zelectric"
21     address_prefixes = ["10.0.1.0/24"]
22   }
23   subnet2 = {
24     name        = "backend-subnet"
25     rg_name     = "rg-dev-zelectric"
26     vnet_name   = "vnet-zelectric"
27     address_prefixes = ["10.0.2.0/24"]
28   }
29   subnet3 = {
30     name        = "AzureBastionSubnet"
31     rg_name     = "rg-dev-zelectric"
32     vnet_name   = "vnet-zelectric"
33     address_prefixes = ["10.0.3.0/24"]
34   }
35 }
36
    
```

AGENDA – DYNAMIC BLOCK

1) **DYNAMIC BLOCK**- A block inside a block and if another block donot have any name then Dynamic block will come into picture. Or Whatever block comes under or inside resource block is called as dynamic block.

2) Create “azurerm_networking” folder under modules folder and write below code into it

```
1 resource "azurerm_virtual_network" "vnet" {
2   name           = "oye-vnet"
3   location        = "centralindia"
4   resource_group_name = "oye-rg"
5   address_space   = ["10.0.0.0/16"]
6
7   subnet {
8     name           = "frontend-subnet"
9     address_prefixes = ["10.0.1.0/24"]
10  }
11
12  subnet {
13    name           = "backend-subnet"
14    address_prefixes = ["10.0.2.0/24"]
15  }
16
17  subnet {
18    name           = "AzureBastionSubnet"
19    address_prefixes = ["10.0.3.0/24"]
20  }
21 }
22
```

3) Run terraform apply then our vnet and subnets will be made

Name	IPAM Pool	IPv4	IPv6	Available IPs	Delegated to	Security group	Route table
frontend-subnet	-	10.0.1.0/24	-	251	-	-	-
AzureBastionSubnet	-	10.0.3.0/24	-	251	-	-	-
backend-subnet	-	10.0.2.0/24	-	251	-	-	-

4) Using for each in code

```
1 variable "vnet" {}
2
3 resource "azurerm_virtual_network" "vnet" {
4   for_each = var.vnet
5   name     = each.value.name
6   location = each.value.location
7   resource_group_name = each.value.resource_group_name
8   address_space = each.value.address_space
9
10  subnet {
11    name           = "frontend-subnet"
12    address_prefixes = ["10.0.1.0/24"]
13  }
14
15  subnet {
16    name           = "backend-subnet"
17    address_prefixes = ["10.0.2.0/24"]
18  }
19
20  subnet {
21    name           = "AzureBastionSubnet"
22    address_prefixes = ["10.0.3.0/24"]
23  }
24 }
25
```

```
1 vnet = {
2   vnet1 = {
3     name           = "oye-vnet"
4     location        = "centralindia"
5     resource_group_name = "chameli-rg"
6     address_space   = ["10.0.0.0/16"]
7   }
8   vnet2 = {
9     name           = "oyel-vnet"
10    location        = "centralindia"
11    resource_group_name = "chameli-rg"
12    address_space   = ["10.0.0.0/16"]
13  }
14 }
```

5) SEARCH – terraform variable block

The screenshot shows the Terraform documentation website. The top navigation bar includes links for Terraform, Install, Tutorials, Documentation, Registry, and Try Cloud. A search bar is located on the right. The left sidebar shows a tree view of the documentation structure, with 'Configuration Language' expanded. The main content area is titled 'Input Variables' and includes a version selector set to 'v1.10.0 (latest)'. The page content explains that input variables allow customizing Terraform modules without altering their source code. It mentions that variables can be set using CLI options and environment variables, and that they should be passed in the 'module' block when used in child modules. A right-hand sidebar lists related topics: 'Input Variables', 'Declaring an Input Variable', 'Arguments' (highlighted with a red circle), 'Using Input Variable Values', and 'Assigning Values to Root Module Variables'.

6) We can custom set also our declared variables as shown below

```
modules > azure_terraform > main.tf > ...  
variable "vnets" {  
  type = map(  
    string = {  
      string = string  
    }  
  )  
}
```

```
modules > azure_terraform > main.tf > variable "vnets" > type  
1 variable "vnets" {  
2   type = map(  
3     string = {  
4       name           = string  
5       location        = string  
6       resource_group_name = string  
7       address_space    = list  
8     }  
9   )  
}
```

```
modules > azure_terraform > main.tf > variable "vnets"  
1 variable "vnets" {  
2   type = map(object(  
3     name           = string  
4     location        = string  
5     resource_group_name = string  
6     address_space    = list(string)  
7   ))  
8   description = "A map of virtual networks with their con  
9 }  
10
```

= { wala map hota hai..
{ wala block hota hai..

7) We can make dynamic blocks under other resources as well like

i) network rule under storage account resource block

```
resource "azurerm_storage_account" "example" {  
  name = "storageaccountname"  
  resource_group_name = azurerm_resource_group.example.name  
  
  location = azurerm_resource_group.example.location  
  account_tier = "Standard"  
  account_replication_type = "LRS"  
  
  network_rules {  
    default_action = "Deny"  
    ip_rules = ["100.0.0.1"]  
    virtual_network_subnet_ids = [azurerm_subnet.example.id]  
  }  
  
  tags = {  
    environment = "staging"  
  }  
}
```

ii) security rule under nsg

```
resource "azurerm_network_security_group" "example" {  
  name = "acceptanceTestSecurityGroup1"  
  location = azurerm_resource_group.example.location  
  resource_group_name = azurerm_resource_group.example.name  
  
  security_rule {  
    name = "test123"  
    priority = 100  
    direction = "Inbound"  
    access = "Allow"  
    protocol = "Tcp"  
    source_port_range = "+"  
    destination_port_range = "+"  
    source_address_prefix = "+"  
    destination_address_prefix = "+"  
  }  
  
  tags = {  
    environment = "Production"  
  }  
}
```

8) Now to put multiple subnet block instead of copying and pasting again and again in code, we have to use concept of dynamic block.

9) SEARCH – azurerm dynamic block

PROVISIONER BLOCKS

```
resource "aws_elastic_beanstalk_environment" "tfenvtest" {
  name                = "tf-test-name"
  application          = aws_elastic_beanstalk_application.tfctest.name
  solution_stack_name = "64bit Amazon Linux 2018.03 v2.11.4 running Go 1.12.6"

  dynamic "setting" {
    for_each = var.settings
    content {
      namespace = setting.value["namespace"]
      name      = setting.value["name"]
      value     = setting.value["value"]
    }
  }
}
```

10)

Dynamic Block - Allow you to generate multiple nested block dynamically
आपको कई नस्टेड ब्लॉक्स को गतिशील रूप से उत्पन्न करने की अनुमति देता है।

1. Storage Account - Network Rules
2. Virtual Network - Subnet
3. NIC - IP configuration

The screenshot shows the Terraform IDE interface. On the left, the Explorer pane shows a project structure with folders like 'Environment', 'dev', 'prod', 'qa', and 'Modules'. The 'azure_rm_networking' module is selected. The main editor displays the Terraform configuration for the 'azure_rm_virtual_network' resource. A red circle highlights the 'dynamic "subnet"' block, which is used to generate multiple subnets dynamically based on the 'var.subnets' variable. The configuration includes variables for 'vnets' and 'subnets', and a resource block for 'azure_rm_virtual_network' with a dynamic block for 'subnet'.

```
1 variable "vnets" {}
2 variable "subnets" {}
3
4 resource "azure_rm_virtual_network" "vnet" {
5   for_each = var.vnets
6   name     = each.value.name
7   location = each.value.location
8   resource_group_name = each.value.resource_group_name
9   address_space = each.value.address_space
10
11   dynamic "subnet" {
12     for_each = var.subnets
13     content {
14       name = subnet.value.name #subnet put as
15       address_prefixes = subnet.value.address_prefixes
16     }
17   }
18 }
19
20
21
```

11) Now we write code as below also means we can mention subnet1 and subnet2 individually for both vnets. And in for_each we will mention it as each.value.subnets.

```
1 variable "vnets" {}
2 variable "subnets" {}
3 resource "azurerm_virtual_network" "vnet" {
4   for_each = var.vnets
5   name     = each.value.name
6   location = each.value.location
7   resource_group_name = each.value.resource_group_name
8   address_space = each.value.address_space
9
10  dynamic "subnet" {
11    for_each = each.value.subnets
12    content {
13      name = subnet.value.name
14      address_prefixes = subnet.value.address_prefixes
15    }
16  }
17 }
18
```

```
1 vnets = {
2   vnet1 = {
3     name           = "dhondhu_vnet"
4     location        = "centralindia"
5     resource_group_name = "dhondhu_rg"
6     address_space    = ["10.0.0.0/16"]
7     subnets = {
8       snet1 = {
9         name           = "snet1"
10        address_prefixes = ["10.0.1.0/24"]
11      }
12       snet2 = {
13         name           = "snet2"
14         address_prefixes = ["10.0.2.0/24"]
15       }
16     }
17   }
18   vnet2 = {
19     name           = "tondu_vnet"
20     location        = "centralindia"
21     resource_group_name = "dhondhu_rg"
22     address_space    = ["10.1.0.0/16"]
23     subnets = {
24       snet1 = {
25         name           = "tondu_subnet1"
26         address_prefixes = ["10.1.1.0/24"]
27       }
28       snet2 = {
29         name           = "tondu_subnet2"
30         address_prefixes = ["10.1.2.0/24"]
31       }
32     }
33   }
34 }
```