

Home = Shivam Kumar Singh +
Registration No: 22BSA10166

2-A

i) Required Provider code

→ First need to specify and Provide for AWS In your Terraform configuration. this is usually done. In a file named 'main.tf' or similar.

ii) Provide "aws" {

region = "us-east-1" // change this to your desired AWS region

ii) code for creating one Instance of 't3.micro',
named as 'web server 1'.

→ Next you can Add your resource block to create the EC2 Instance:

resource "aws_instance" "web server 1" {

ami = "ami-0c55b959c6fafa7f0" # Replace

with a valid AMI ID for your region.

instance-type = "t3.micro"

tag = {

name = "web server 1"

}

complete example (1-B)

→ combining both parts your complete 'main.tf' file would look like this:

```
1) provider "aws" {  
  region = "us-east-1" # change this to your region  
}  
  
resource "aws_instance" "web_server" {  
  ami           = "ami-0c558f59c6baf020" # change this to your AMI  
  instance_type = "t3.micro"  
  tags = {  
    Name = "web_server" # change this to your tag  
  }  
}
```

→ Addition step:

1) Initialize Terraform: Run 'terraform init' in the directory where your 'main.tf' file is located to initialize the terraform working directory.

2) Plan the deployment: Run 'terraform plan' to see that resource will be created.

3) Apply the configuration: Run 'terraform apply' to create the EC2 instance.

4) Destroy the resource → when you are done you can run 'terraform destroy' to clean up the resource created by terraform.

Q2

1) ^(2-a) In terraform command that show the sequence of step before actually making change to the Infrastructure. Is 'terraform plan' this command generate an execution Plan. which describe what action terraform will take to change the current Infrastructure to match the desired state defined in your configuration files.

1) 'terraform plan'

Syntax

1) terraform plan

* The 'terraform plan' command create an execution plan which is show you what action Terraform will take to change the current state to match the desired state defined in your configuration.

2) 'terraform init'

Syntax

1) terraform init

→ it setup the backend. Installs the required provider Insight plugins. and prepare the directory for other Terraform commands.

Example usage

1) Initialize the Terraform working directory:

i) Terraform init

2) show the execution plan

i) Terraform plan

* → these commands are fundamental to using terraform effectively. ensuring that you understand what changes will be made before ~~applying~~ applying them and that you are working environment is set up correctly.

2-6

Any

2-6

1) cloud provider support

- * ① Terraform ÷ multi-cloud supports, allowing user to manage resource across various cloud provider (AWS, Azure, Google cloud, etc - cloud formation: AWS specific, designed exclusively for managing AWS resources.

2) Syntax and language:

- * Terraform ÷ use Hashicorp configuration language HCL, which is known for its readability and simplicity, making it easier for beginners to learn.
- * cloud formation: utilizes JSON or YAML, which can be more complex and may present a steeper learning curve for those unfamiliar with these formats.

3) state management

- * ① Terraform: maintains a state file that tracks the current state of infrastructure, allowing for the efficient management and update.

- * ② cloud formation ÷ automatically manages the state of resource with in AWS, abstracting away the complexity of state management from the user.

3) (a) Ans

(34)

Ans → Isolation in virtualization is a critical concept that helps ensure that security, stability, and efficient resources management of virtual machines (VMs). Here are several reasons why isolation is important and how it contributes to VM security.

1) Security

* Separation of Environment:

⊛ Isolation ensures that each VM operates in its own environment, preventing unauthorized access to data and resources of other VMs. If one VM is compromised, isolation limits the attacker's ability to access or affect other VMs on the same host.

⊛ Containment of threats: If a VM exhibits a security breach → (malware infection). Isolation helps contain the threats within that VM. The containment prevents the spread of malware or other malicious activity to other VMs or the host system.

2) Resource management

⊛ Resource Allocation → Isolation allows for controlled allocation of resources (CPU, memory, storage)

where one vm could monopolize resource and degrade the performance of the system and other vm's.

(*) Performance stability : By Isolating vm's performance issue in one vm (such as high cpu) do not impact the performance of other vm's, ensuring that each vm can operate efficiently and predictably.

3) Fault tolerance and reliability.

If a vm crashes or experience a failure, isolation does not affect the stability of other vm's running on the same physical host. this lead to improve overall systems reliability.

4) Compliance and Governance (:-)

5) Infrastructure security :-

3-1

Ans

3-1

Agility refers to the Ability to move quickly and easily adapt to change. and respond promptly to new Information or circumstances. In a broader context, particularly in business and technology, agility encompasses a mindset and a set of practices that enable organization to be flexible, responsive and adaptive to changing market condition - customer needs and Technological advancement.

→ 1) Flexibility:

Agility Allows organization to pivot quickly in response to change in the environment whether those change are driven by market dynamics, customer feedback, or technological innovations. this flexibility is crucial for maintaining competitiveness.

2) Speed:
Agile organization promote speed in decision-making and execution. they aim to reduce the time it takes to bring product to market. enabling them to capitalize on market opportunities more quickly than their agile competition.

Q5

5

Ans

→ The terraform command used to check the correctness of the configuration is 'terraform validate'. This command is used to validate the configuration files in a directory and ensure that they are syntactically valid and internally consistent.

→ Syntax

1) terraform validate [options] [DIR]

Options

* DIR: This is an optional argument. If specified, terraform will validate the configuration files in the given directory. If no directory is specified, terraform will validate the configuration files in the current directory.

Example

if you want to validate configuration files ~~in~~ in current directory.

1) terraform validate

→ if you want to validate in specific directory.

syntax

1) `terraform validate my-terraform-config`

out put

* If the configuration is valid, you will see a message indicating that the configuration is valid.

* If there are any issues, Terraform will provide an error message detailing what is wrong with the configuration files, helping you to correct them before proceeding with other terraform commands.