**KAPPIL – 8KMILES**

1. Working experience with terraform?

I have some conceptual knowledge on terraform, and I have my personal interest to learn that,

It is used to define infrastructure for a variety of cloud providers (e.g. AWS, Azure, Google Cloud, DigitalOcean, etc) using a simple, declarative programming language and to deploy and manage that infrastructure using a few CLI commands.

Why Terraform?

To orchestrate multiple services/providers in a single definition like create instances with a cloud provider, create DNS records with a DNS provider, and register key/value entries in Consul. And you can use a single solution that supports multiple services.

Terraform Components:

Terraform allows you to create infrastructure configurations that affect resources across multiple cloud services and cloud platforms with the following components:

• Configurations: It is a text file which holds the infrastructure resource definitions in .tf or .tf.json formats.

• Providers: Terraform leverages multiple providers to talk to back-end platforms and services, like AWS, Azure, DigitalOcean, Docker, or OpenStack.

Resources: Resources are the basic building blocks of a Terraform configuration. When you define a configuration, you are defining one or more (typically more) resources. These are provider specific and calls to recreate configuration during migrations.

* 1. • Variables: Supports use of variables making configurations more portable and more flexible. Re-use a single configuration multiple times through changing variable values. o provider.tf.json – contains information specific to aws (the provider)
  2. o vars.tf.json – contains variables that will be later used by terraform
  3. o main.tf.json – contains the bulk of terraform configuration (resources)
  4. o output.tf.json – specifies the information that should be output

We define all the data for terraform in four files:

After the authoring of these files, the first step is use the terraform plan command. The plan command lets you see what Terraform will do before actually doing it. This is a great way to sanity check your changes before unleashing them onto the world. The output of the plan command is a little like the output of the diff command: resources with a plus sign (+) are going to be created, resources with a minus sign (-) are going to be deleted, and resources with a tilde sign (~) are going to be modified.

Next is to use terraform apply which actually does the work.

1. VPC architecture and what you have done in it?

Virtual private cloud which we can simply refer to as a private sub-section of AWS which can be controlled by the user who can place the AWS resources within it and thus creating a logically isolated section. The components of a VPC are: subnets, network ACL, Nat Gateway, VP gateways, internet gateway, route table, elastic IP, endpoint, security group, VPN connections and customer gateways.

When you create an AWS account: a VPC with Internet gateway, route table with predefined routes to the default subnets, NACL with predefined rules and subnets to provision resources in.

Internet Gateway: is the one which routes the connection between a VPC and internet. Only one vpc can be attached to one internet gateway. We cannot detach an IGW when there are active resources in VPC.

Route tables: consists a set of rules called routes which are used to determine where the network traffic is directed. You cannot delete a route tables when you have active dependencies. To route the traffic to AWS resources in VPC.

Network access control lists (NACL): an optional layer of security for VPC that acts as a firewall for traffic control in subnets. These have inbound and outbound rules. In default NACL all traffic is allowed. The rules are evaluated based on the rule number. In NACL we have a catch all rule which denies all the traffic by default and this cannot be modified.

**VPC PEERING**

This is primarily used to enable the communication between the servers lying in different VPCs as if there were in the same VPC. It allows the machines to connect using private IP addresses. The phases in VPC peering are:

* • Initiating-request
* • It might Fail
* • Or else it moves to Pending-acceptance
* • Later it might get Expired
* • Or Rejected
* • Or if accepted then it will enter into Provisioning phase
* • And it will enter Active
* • Deleted

***Limitations***

* • Cannot be created between VPCs those having a matching or overlapping IPV4/IPV6 CIDR blocks
* • Cannot be created for VPCs lying in different regions
* • Will not support transitive peering
* • Cannot have more than one peering connection between same VPCs

1. We have one 3-tier web-based app, we are going to use some apache,tomcat server, mysql and front-end as well. What will go into private subnet and what will go into public subnet, how will be your vpc and how will you form for this kind of architecture?

The web-servers like apache and app servers will run on ec2-instances, these gonna running on the public-subnets, the mysql uses the aws rdbms, the db can be the private-subnet where the internal users can access it, to retrieve the information as per the queries, web-site hosting services are hosted in the public-subnet and db’s in the private-subnets.

1. How will you establish connection between the public and private subnets?

we can configure the route tables to access to private subnets, we can also use security groups for the instances to allow only access to which are in the public subnet. We use NAT to access the internet from private subnets. If outside user want to access the public subnet then we can use nat.

1. Instance profile in AWS, why we use it?

Instance profile is used by the apps that are running in it,

How it is helpful?

You are not going to keep any credentials, it is assumed by the instance itself, you are not creating it, any app that is running on the instance only it can use it.

It’s a container for IAM role and you can pass this role information to a EC2-instance when the instance starts. This role gives access permissions to S3 buckets and other repositories where your applications are stored.

We can create IAM instance profile by using AWS CLI

Command to create instance profile is ---

Aws iam create -instance-profile - -instance-profile-name

1. What is the difference between NACL and security groups, when do we use?

|  |  |
| --- | --- |
| **Security Group** | **Network ACL** |
| Operates at the instance level (first layer of defense) | Operates at the subnet level (second layer of defense) |
| Supports allow rules only | Supports allow rules and deny rules |
| Is stateful: Return traffic is automatically allowed, regardless of any rules | Is stateless: Return traffic must be explicitly allowed by rules |
| We evaluate all rules before deciding whether to allow traffic | We process rules in number order when deciding whether to allow traffic |
| Applies to an instance only if someone specifies the security group when launching the instance, or associates the security group with the instance later on | Automatically applies to all instances in the subnets it's associated with (backup layer of defense, so you don't have to rely on someone specifying the security group) |

1. For NACL I have 2 rules, one rule is like xyz ip all it will deny, another is same xyz ip all allow, ?

NACL follows the sequencial order no, whatever rule number you will give first it will give high priority to that only, it does not follow later rules.

1. Tell me the difference between between iam policy and bucket policy?

IAM policies specify what actions are allowed or denied on what AWS resources (e.g. allow ec2:TerminateInstance on the EC2 instance with instance\_id=i-8b3620ec). You attach IAM policies to IAM users, groups, or roles, which are then subject to the permissions you’ve defined. In other words, IAM policies define what a principal can do in your AWS environment.

S3 bucket policies, on the other hand, are attached only to S3 buckets. S3 bucket policies specify what actions are allowed or denied for which principals on the bucket that the bucket policy is attached to (e.g. allow user Alice to PUT but not DELETE objects in the bucket). S3 bucket policies are a type of [access control list](http://en.wikipedia.org/wiki/Access_control_list), or ACL (here I mean “ACL” in the generic sense, not to be confused with S3 ACLs, which is a separate S3 feature discussed later in this post).

Note: You attach S3 bucket policies at the bucket level (i.e. you can’t attach a bucket policy to an S3 object), but the permissions specified in the bucket policy apply to all the objects in the bucket.

IAM policies and S3 bucket policies are both used for access control and they’re both written in JSON using the AWS access policy language, so they can be confused.

1. What is the principal in IAM role?

The principal is basically the user whom you are specifying in that iam,

1. What is the structure of the ARN, how do you create it?

The ARN will create by default when you assign an iam role to a service.

The following are the general formats for ARNs; the specific components and values used depend on the AWS service.

**Copy**

arn:*partition*:*service*:*region*:*account-id*:*resource*

arn:*partition*:*service*:*region*:*account-id*:*resourcetype*/*resource*

arn:*partition*:*service*:*region*:*account-id*:*resourcetype*:*resource*

*partition*

The partition that the resource is in. For standard AWS regions, the partition is aws. If you have resources in other partitions, the partition is aws-*partitionname*. For example, the partition for resources in the China (Beijing) region is aws-cn.

*service*

The service namespace that identifies the AWS product (for example, Amazon S3, IAM, or Amazon RDS). For a list of namespaces, see [AWS Service Namespaces](http://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html#genref-aws-service-namespaces).

*region*

The region the resource resides in. Note that the ARNs for some resources do not require a region, so this component might be omitted.

*account*

The [ID](http://docs.aws.amazon.com/general/latest/gr/acct-identifiers.html) of the AWS account that owns the resource, without the hyphens. For example, 123456789012. Note that the ARNs for some resources don't require an account number, so this component might be omitted.

*resource*, *resourcetype*:*resource*, or *resourcetype*/*resource*

The content of this part of the ARN varies by service. It often includes an indicator of the type of resource—for example, an IAM user or Amazon RDS database —followed by a slash (/) or a colon (:), followed by the resource name itself. Some services allow paths for resource names, as described in [Paths in ARNs](http://docs.aws.amazon.com/general/latest/gr/aws-arns-and-namespaces.html#arns-paths).

1. We are restricting the users who is going to create ec2-instances, only use particular AMI, which we will not allow them to use any other AMI ‘s, how do you do this?

We can assign iam policy related to the scenario, we can specify the instance id in the tagging section we can achieve it. We can give resource based access to the roles, that is only particular user can access particular resources.

1. Have you written CFT?
2. What kind of modifications you have done?

Modifying the type of instances, security groups which are assigned to the instances, launch configuration in the auto-scaling groups, depending on the type of environment like qa,uat and prod as per requirement we will provide the information.

1. We have a link that running on prod, it is running with machine type-a, the traffic is very high or low, then we need to change the instance type, what are the things you would consider before doing this?

We can configure to increase the number of instances in the auto-scaling group as per the requirement, then we can change the type of instances as per the requirement. We can redirect the traffic to the other instances,

1. In auto-scaling the launch configuration will do some steps whenever some instance is coming up, where will be the log data related to the user info, what is the name of the log file?

/etc/userdata/

User-data.txt file the data will go into this. /var/lib/cloud/instance/, inside this path the file will be stored.

Wrt data it will be cloud file.

1. You have worked on elb, route53, when we use each of it and how we will use?

**AWS ROUTE 53**

• Launch an instance

• Create an ELB and choose security group so that it allows port 80, configure health checks, add the ec2 instance.

• Go to route 53, get started, create hosted zone where we enter the domain name (ex- www.aws.com), type public hosted zone

• Creates two records by default – start of authority record and name server records

• Now add these name server records in the domain details of your domain registrar.

• Create naked domain name or apex record (without www – example.com)

• Since we don’t have a public ip of load balancer, we create an alias record and choose alias record and the routing policy (simple, weighted, latency, failover and geolocation) and choose whether you wish to evaluate the health of resources or not.

**AWS ELB**

Classic elastic load balancer: distributes incoming application traffic evenly across the available servers in multiple AZs thus ensuring fault tolerance. In the architecture, the ELB is ahead of route tables. Not available for free tier.

In ELB we have both ALB and CLB. And the load balancer should be associated with an VPC. And a proper protocol too needs to be specified based on the traffic that ELB will be dealing with. We should assign the ELB to a SG. We can also configure the health checks of the resources that ELB is serving by defining the ping parameters like protocol, port and path and under details, we should specify the response timeout, interval and unhealthy threshold which define the health of the resources. And finally we add the resources to the ELB.

Concept of cross-zone balancing which ensures that the traffic is evenly distributed across all the servers in all the available zones. When disabled just ensures that the traffic is just balanced between the zones only. It can be either internet facing or internal facing. And can be connected to an auto-scaling group or instances directly.

**APPLICATION LOAD BALANCER (ALB) VS CLASSIC LOAD BALANCER (CLB)**

* • CLB operates at layer 4 of the OSI (Open System Interconnection) model which implies that it routes the traffic based on the ip address and the port number while the ALB operates at layer 7 of the model which means that it operates not just based on the ip address and the port number but also based on the application-level content.

By default the cross-zone load balancing is enabled in ALB and we need to enable the same in CLB.

• While http and https are supported by both the load balancers, CLB can support TCP and SSL in addition there by enabling the ssl termination at the load balancer level itself.

• ALB support path based routing which enables a listener to forward the requests based on the url path while CLB cannot.

• ALB supports deletion protection which prevents the accidental deletion of the load balancer while CLB doesn’t.

1. What is the difference between classic and app loadbalancer?
2. Which one will perform the health checks?

Classic load balancer will perform the health checks. As per the requirement it will route the traffic.

1. What is the difference between role and environment in chef?

A **role** is a way to define certain patterns and processes that exist across nodes in an organization as belonging to a single job function. Each **role** consists of zero (or more) attributes and a run-list. ... When a **chef**-client runs, it merges its own attributes and run-lists with those contained within each assigned **role**.

An **environment** is a way to map an organization's real-life workflow to what can be configured and managed when using **Chef** server. Every organization begins with a single **environment** called the \_default **environment**, which cannot be modified (or deleted).

**What is a Role?**

In your organization, if your infrastructure grows to meet the demands of higher traffic, there are likely to be multiple, redundant servers that all perform the same basic tasks. For instance, these might be web servers that a load balancer passes requests to. They would all have the same basic configuration and could be said to each satisfy the same "role".

Chef's view of **roles** is almost entirely the same as the regular definition. A role in Chef is a categorization that describes what a specific machine is supposed to *do*. What responsibilities does it have and what software and settings should be given to it.

In different situations, you may have certain machines handling more than one role. For instance, if you are testing your software, one server may include the database and web server components, while in production, you plan on having these on separate servers.

With Chef, this can be as easy as assigning the first server to both roles and then assigning each role to separate computers for your production machines. Each role will contain the configuration details necessary to bring the machine to a fully operational state to fulfill its specific role. This means you can gather cookbooks that will handle package installations, service configuration, special attributes for that role, etc.

**What is an Environment?**

Related to the idea of a role is the concept of Chef **environments**. An environment is simply a designation meant to help an administrator know what stage of the production process a server is a part of. Each server can be part of exactly one environment.

By default, an environment called "\_default" is created. Each node will be placed into this environment unless another environment is specified. Environments can be created to tag a server as part of a process group.

For instance, one environment may be called "testing" and another may be called "production". Since you don't want any code that is still in testing on your production machines, each machine can only be in one environment. You can then have one configuration for machines in your testing environment, and a completely different configuration for computers in production.

In the above example given in roles, you could specify that in your testing environment, the web and database server roles will be on a single machine. In your production environment, these roles should be tackled by individual servers.

Environments also help with the testing process itself. You can specify that in production, a cookbook should be a stable version. However, you can specify that if a machine is part of the testing environment, it can receive a more recent version of the cookbook.

1. Have you written test cases for chef?

We have done with chefspec,

1. What is the role with chef?
2. How the attribute precedence will apply when you run the chef-recipe?

Chef has the precedence for attributes, that when you want to use, at what time we can use.

Before the chef-client run all the attributes collected by ohai, and other attributes mentioned in the cookbooks and environments are rebuilt. After that all the attributes are then merged and applied to a node according to the attribute precedence.

Attribute types:

Default (it resets at the start of chef-client and has lowest precedence),

We have attribute files node or recipe, environment or role. In these high priority goes to role, then environment, then node, then recipe file.

force-default (it ensures that the attributes mentioned in the cookbook takes precedence over a default attribute set by a role or environment),

normal (it has higher attribute precedence than a default one),

override (it has highest precedence than default, force default, normal attributes. A cookbook should ensure that it should use over ride attribute only when required), But in override environment take high priority than role.

force override (it ensures that attribute mentioned in the cookbook takes precedence over override attributes set by role or environment),

automatic (it contains data that is identified by ohai , it cannot be modified and always has highest precedence). When you bootstrap the node the ohai plugin is capture the attributes and put it on the server. Which we cant change the attributes, that has the high priority.

1. Tell me what is automatic attribute?