AWS EC2

What is EC2 and What is virtual server in cloud?

We'll also try to automate creation of virtual servers.

Things to do in AWS console to introduce EC2 instance

(Step1: We need to create security group, Step2: we need to create key pair Step3: Pair the Key-pair file with the EC2 instance, Step4: )

We need to login using IAM user.

To setup EC2 instance in cloud one should know about regions, availability zones, VPC subnet, Security groups and EC2.

Virtual server is basically a server in cloud.

Whenever we have server in Datacenter, that server is basically a physical server. And in those blade servers/ physical servers in the datacenters we deploy our applications and databases. (Landlord)

When we deploy our applications or databases on the cloud , we deploy them to servers but these servers are in cloud and that is called virtual servers (tenant). Virtual servers are servers in the cloud which we make use of.

In AWS virtual servers are called EC2 (Elastic Compute Cloud).

What are regions? What problem does it solve? What is its purpose?

In AWS we have number of regions around the world, we can put our resources in US, we can put in Asia, Canada, France etc… By having multiple regions users get better availability, and low latency for users. By using different servers for different regions, user will get response from applications which are nearest to them, So they'll have better performance, and by spreading the application user gets better availability as well. For Ex: If one of the servers go down we can get response from servers from other regions as well.

Steps involved in creating virtual server/ EC2 server in AWS/ HTTP server in AWS.

* First thing we need choose when creating EC2 instance/ Virtual server is **which region** we'd want to create it in.
* Amazon Machin Image. **(AMI)**. We need to copy the ID of the AMI. (ami-01816d07b1128cd2d)
* Next we would need **instance type**. We earlier chose the software, now we need to choose the H/W. In our example we choose t2.micro which is free eligible.
* Next we need to configure instance details. **(network and subnet) (VPC)**
* We choose the basic storage provided by default.
* We can add the tags if needed.
* Configure **security group**.

Note: Whenever we create EC2 instance a default VPC will be created in each of the regions as well.

Note: Whenever we create resources in a data center, they are already protected by a physical firewall. Nobody can reach these resources from outside our corporate network. Now to do this cloud we use VPC.

Whenever we create EC2 instance, one of the first things we do is which subnet it should get into. Remember subnet is inside VPC. Whenever we create an EC2 instance inside private n/w then only the resources within that VPC can talk to that EC2 instance. However, if the instance is in public subnet then outside resources can talk.

**CIDR block** is used, when we want to allow access from anywhere. Below example allow access from any system everywhere.

CIDR ["0.0.0.0/0"]

Tips:

Port numbers for various network communication protocols:

HTTP -----> Port 80

HTTPS -----> Port 443

SSH -----> Port 22

Example code for creating EC2 instance / HTTP server

// creating a http server / EC2 instance / virtual server.

resource "aws\_instance" "http\_server" {

ami = "ami-01816d07b1128cd2d" // AMI specifies about S/W. What is the S/W, /what is the OS, What is the application that needs to be installed on top of it etc.

key\_name = "VenkatTestKeyPair" // What is the key name that needs to be associated with the EC2 instance.

instance\_type = "t2.micro" // instance type specifies about H/W. We need to consider cost factors here.

// vpc\_security\_groups\_ids = ["sg-046b03ec770f0ecf3"] // isntead of hardcoding the sg value we cna fetch from this same file as we earlier used sg. this can eb done via var.

vpc\_security\_groups\_ids = [aws\_security\_group.http\_server\_sg.id] // Security groups that we need to associate with this instance. We may have to specify array of ids.

// We can associate multiple security groups with same EC2 instance, So we need to associate with a list, list of security group IDs

subnet\_id = "subnet-01c086cdefb212d63" // Specifies, which subnet should this EC2 be created in.

}

Step1: Go to EC2 in AWS console and press launch instance.

A screenshot of a computer

Description automatically generated

v Application and OS Images (Amazon Machine Image) 
Info 
An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or 
Browse for AMIS if you don't see what you are looking for below 
Q Search our full catalog including 1000s of application and OS Images 
Recents 
Amazon 
Linux 
aws 
Quick Start 
macos 
Ubuntu 
ubuntuØ 
Windows 
Microsoft 
Red Hat 
Red Hat 
SUSE Linux 
SUSE 
Debian 
Browse more AMIS 
Including AMIS from 
AWS, Marketplace and 
the Community 
Free tier eligible 
Amazon Machine Image (AMI) 
Amazon Linux 2023 AMI 
ami-01816d07b1128cd2d (64-bit (x86), uefi-preferred) / ami-02dcfe5d1d39baa4e (64-bit (Arm), uefi) 
Virtualization: hvm ENA enabled: true Root device type: ebs 
Description 
Amazon Linux 2023 is a modern, general purpose Linux-based OS that comes with 5 years of long term support. It is optimized for AWS and designed to provide a 
secure, stable and high-performance execution environment to develop and run your cloud applications. 
v Summar 
Number of insti 
Software Image 
Amazon Linux 
ami-01816d07b11 
Virtual server t) 
t2.mjcro 
Firewall (securi' 
New security grc 
Storage (volum 
I volume(s) - 8 ( 
O Free tier: 
(or t3.mic 
instance 
public IPv 
storage, 2 

v Summary 
Number of instances 
Info 
Software Image (AMI) 
Amazon Linux 2023 AMI 2023.6.20241212.0 
kernel-6. I 
ami-01816d07b1128cd2d 
Virtual server type (instance type) 
t2.mjcro 

Step2:

Key-Pair

Whenever we want to connect to EC2 instance, lets say I want to do SSH to my EC2 instance then we need the key-pair.

We need the key-pair to be associated with the EC2 instance, at the time of the creation.

So first step is to create a key-pair, then associate with an EC2 instance and then we'll be able to use it when we want to connect to the EC2 instance.

Step3:

We need to associate the AWS EC2 instance that we create with the key pair file in the local system.

By following the steps mentioned in section "Steps involved in creating virtual server/ EC2 server in AWS/ HTTP server in AWS." we can create EC2 instance.

Now if we want add a html file into the EC2 instance then we need to follow the below steps:

* We need to connect to the http-server. Key-pair helps us in doing this.
* We then need to execute commands inside EC2. To execute few commands inside EC2 instance we need provisioner.

 On a Overall setup in terms if terraform below steps are required to implement http-server/ EC2 instance in AWS

Step1:

Security Group:

resource "aws\_security\_group" "http\_server\_sg" {

name = "http\_server\_sg"

vpc\_id = "vpc-0e728fef0720adff5" // which vpc should this sg should be part of.

// to configure the incoming traffic source. To allow/ block incoming traffic from a particular source.

// below block is for incoming HTTP

ingress {

from\_port = 80

to\_port = 80

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"] // CIDR blocks is a list and so we mention "[]" to allow multiple ip address ranges.

}

// Below block is for incoming ssh

ingress {

from\_port = 22

to\_port = 22

protocol = "tcp"

cidr\_blocks = ["0.0.0.0/0"] // CIDR blocks is a list and so we mention "[]" to allow multiple ip address ranges.

}

// To configure the outgoing traffic. What are the other things that we can talk to.

// typically when we creat a sg by default it can talk to every one. default egress to everything.

// but by default terraform disables this feature. So we need to take care of enabling it.

egress {

from\_port = 0

to\_port = 0

protocol = -1

cidr\_blocks = ["0.0.0.0/0"]

}

tags = {

name = "http\_server\_sg"

}

}

Step 2:

Key Pair:

Why do we require Key-Pair, Whenever we need to connect to EC2 instance or whenever we do SSH into EC2 instance, we need Key pair for this.

We need a keypair to be associated with the EC2 instance, at the time of creation of EC2 instance.

So we need a Keypair and associate it with EC2 instance.

Go to the AWS console and search for EC2 and look for KeyPairs in the left side window, Then create a Keypair.

Keep it inside the host system and ensure change access permission via CHMOD.

For ex: $~/aws/aws\_keypair/VenkatTestKeyPair.pem

Step 3:

Now we need to start implementing EC2/ http-server related code in terraform.

// Ex-03

// creating a http server / EC2 instance / virtual server.

resource "aws\_instance" "http\_server" {

ami = "ami-01816d07b1128cd2d" // AMI specifies about S/W. What is the S/W, /what is the OS, What is the application that needs to be installed on top of it etc.

key\_name = "VenkatTestKeyPair" // What is the key name that needs to be associated with the EC2 instance.

instance\_type = "t2.micro" // instance type specifies about H/W. We need to consider cost factors here.

vpc\_security\_group\_ids = [aws\_security\_group.http\_server\_sg.id] // Security groups that we need to associate with this instance. We may have to specify array of ids.

// We can associate multiple security groups with same EC2 instance, So we need to associate with a list, list of security group IDs

// vpc\_security\_groups\_ids = ["sg-046b03ec770f0ecf3"] instead of hardcoding the sg value we can fetch from this same file as we earlier used sg. this can eb done via var.

subnet\_id = "subnet-01c086cdefb212d63" // Specifies, which subnet should this EC2 be created in. These subnets can be fetched from VPC page in AWS console.

connection {

type = "ssh"

host = self.public\_ip

user = "ec2-user"

// private\_key = "~/aws/aws\_keys/VenkatTestKeyPair.pem"

// private\_key = file(var.aws\_key\_pair)

private\_key = file("./VenkatTestKeyPair.pem")

}

// Thsi section will execute set of commands inside EC2 instance.

provisioner "remote-exec" {

inline = [

"sudo yum install httpd -y", // install httpd

"sudo service httpd start", // start

"echo Welcome to Venkats AWS Practice Lab - Virtual Server is at ${self.public\_dns} | sudo tee /var/www/html/index.html" // Copy a file

]

}}

(How is regions can be compared to CDNs?)

We know that multiple regions around the world and each of these regions have multiple availability zones. For ex; Us east zone will have multiple availability zones, similarly for India multiple availability zone. So on…

These availability zones are within a region but physically separated data centers. The reason for availability zones are again as we earlier saw is to increase availability.

**What is AMI?**

Ami involved in choosing what is the software configuration user would need to launch the instance that they want. And that involves specifying the OS, and details of the S/W one would need on that specific OS.

In the example we're seeing we choose Amazon basic linux (Amazon Linux 2 AMI). As this is part of free tier.

**What is instance type?**

It involves choosing how many CPUs, How much memory do we need, What kind of hard disk we'd want, and what kind of n/w performance we'd want.

What are instance details and what are the things we need to take care in this step?

In Configure instance details 2 important choices we need to make when configuring instance details are, network and subnet.

**Why do we need n/w and subnet?**

Whenever we create resource in data centers they are already protected by a physical firewall. Which means nobody can reach these resources from outside your corporate n/w.

To achieve this cloud, we create VPC. (In data power we create virtual frontside handlers and backside handlers)

Virtual private Cloud (VPC) is kind of your own n/w in cloud.

In VPC we can create multiple subnets. We can create public subnet or private subnet.

Whenever we want to create a EC2 instance, one of the first things we need to choose is which subnet it would need to go into.

If we create EC2 instance in a pvt subnet, then only the resource within inside that VPC n/w can talk to EC2 instance. Resources from outside will not be able to talk to it.

If we create in public subnet, even resource from outside will be able to talk, send request to that specific EC2 instance.

Typically databases are put inside pvt subnet as it protects and don’t allow access from outside VPC.

However, we put things like HTTP servers, Webservers, which requires outside access, these things we put into public subnet.

So VPS is our own n/w in AWS and inside that VPC network we create different subnets, which we can use to put right resources into those.

**What is security group? Why is it required?**

A security group is another way we can control traffic to our EC2 instance.

Lets consider this example:

A virtual server is created inside a subnet, even if we allow traffic inside this subnet, I can decide not to allow certain traffic to a specific EC2 instance.

For instance a subnet allows all http and https traffic to a specific instance, now we have an option to allow only https traffic to another instance. That’s what security group will allow us to do.

By specifying a security group on our EC2 instance, we can filter and restrict traffic kinds to a specific EC2 instance.

Now note here that these security groups are configured for an instance.

With a security group we'll be able to say, I would want to allow this kind of traffic this protocol, to this port.

Is the security group related to throttling/ rate-limiting?

Learn and recollect how this is taken care in datapower.

A security group is a set of firewall rules that control the traffic for our instance. On this page, we can add rules to allow specific traffic to reach our instance.

For example, if we want to setup a web server and allow internet traffic to reach our instance, add rules that allow unrestricted access to HTTP and HTTPS ports.