## horizontal line



Deploy Scalable VPC Architecture on AWS Cloud

29.11.2023

**─**

Mahantesh N Hiremath

V-Excel Technology pvt ltd.

Bengaluru

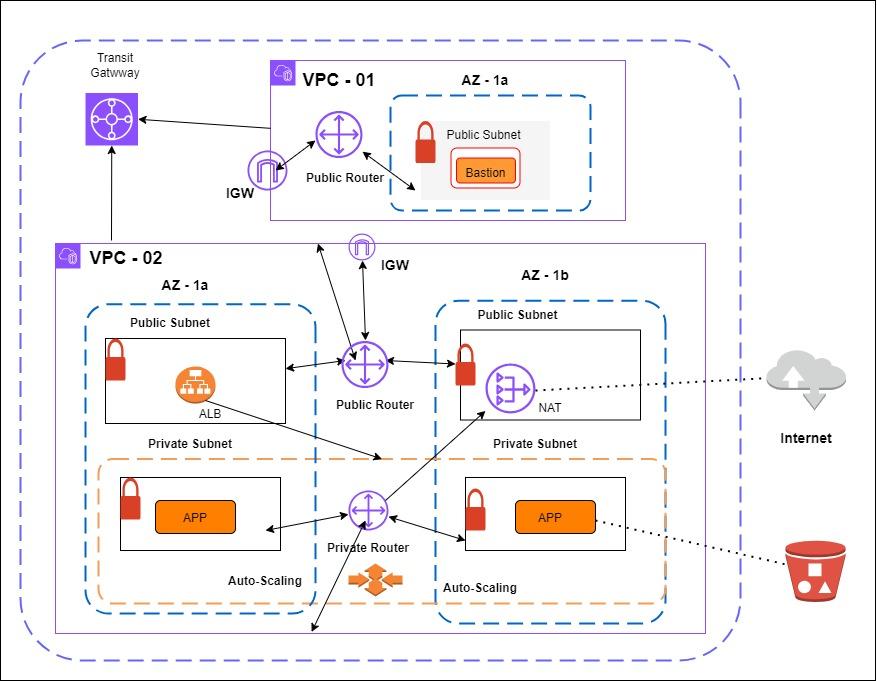
# Table of Content:

1. Goal:
2. Services used:
3. Pre-requirements:
4. Pre-deployment steps:
5. VPC Deployment steps:
6. Validating steps:

# Goals

1. Deploy Scalable Virtual Network Architecture with AWS Management Console.

**Scalable VPC Architecture On AWS Cloud.**

****

# Services used:

1. EC2
2. VPC
3. IAM
4. Cloudwatch
5. S3

# Pre-requirements:

1. It is mandatory to have AWS Account, to establish IT Infrastructure in AWS Cloud.

# Pre-deployment steps:

1. Launch an EC2 instance, with default configurations.
2. Install application dependencies.
3. AWS CLI
4. Install Apache web server
5. Install Cloud-watch-agent
6. Install AWSlogs
7. Install Amazon-SSM-Agent
8. Create “**Golden AMI**”.

**Commands for pre-deployment steps:**

**sudo su -**

1. **AWS CLI:**
2. **AWS cli, comes free installed, if when we use AWS AMI, when we launching the instance.**

**To validate the version installed, we can use —----**

**# aws --version**

1. **Install Apache web server:**
   1. **To Install Apache web server on Amazon Linux, use command–**

**# yum install httpd -y**

**# systemctl enable httpd**

1. **Install Cloud-watch-agent:**
   1. **To Install Cloud watch agent on Amazon Linux, use command-**

**# yum install amazon-cloudwatch-agent**

**# systemctl enable amazon-cloudwatch-agent**

* 1. **Edit (.json) file to push memory matrics to cloudwatch**

1. **Install AWSlogs:**
2. **To install AWSLogs on Amazon Linux, use command-**

**# Yum install awslogs -y**

**# systemctl start awslogsd**

**# systemctl status awslogsd**

1. **Edit awscli.conf —- edit region in it:**
2. **Edit awslogs.conf —- edit log group name in it.**
3. **Create IAM Role:**
4. **With Cloudwatchfull access | S3 full access | Amazon SSM Full Access**
5. **Install Amazon SSM Agent:**
6. **Amazon SSM Agent comes pre-installed, if when we use AWS AMI, when we launching the instance.**

**# yum install amazon-ssm-agent -y**

**# systemctl start amazon-cloudwatch-agent**

**# systemctl status amazon-cloudwatch-agent**

1. **Attach IAM Role IN EC2 instance, go to modify role.**
2. **Check it with**

**# systemctl restart amazon-cloudwatch-agent**

1. **Check it with, its attached to ownership.**

**# cat amazon-cloudwatch-agent.log**

1. **View in cloudwatch,, in metrics,, CWAgent.**
2. **Create Golden AMI.**

Amazon-cloudwatch-agent.json:

{

"metrics":{

"metrics\_collected":{

"mem":{

"measurement":[

"mem\_used\_percent"

],

"metrics\_collection\_interval":60

}

},

"append\_dimensions": {

"InstanceId": "${aws:InstanceId}"

}

}

}

# VPC deployment steps:

1. Build VPC Network ( 192.168.0.0/16 ) for Bastion Host deployment as per architecture shown above.
2. Set-up & Attach Internet-Gateway (IG) to VPC.
3. Create Subnet, ( Public ).
4. Set up Route Table ( Public ), and associate related Subnet to it, and route the default traffic to IGW for inbound/outbound internet Connection.
5. Build VPC Network (172.32.0.0/16) for, Highly available & Auto Scalable Application Servers as per the architecture shown above.
   1. Set-up & Attach Internet-Gateway (IG) to VPC.
   2. Create Subnets, ( 2 Public subnets in 2 different Availability Zones like – 1a, 1b)
   3. Create Subnets, ( 2 Private subnets in 2 different Availability Zones like - 1a, 1b
   4. Set-up NAT Gateway in Public Subnet, and allocate Elastic IP Address to it.
   5. Set-up Route Table (Public), and associate related Subnets to it, and route the default traffic to IGW for inbound/outbound internet connection.
   6. Set-up Route Table (Private), and associate related Subnets to it, and route the default traffic to NAT gateway for Outbound internet Connection.
6. Create Transit-Gateway, and associate Both VPC’s to Transit Gateway, by creating Transit gateway attachments, for private Communication.
7. Create Cloudwatch Log-groups with 2 log streams to store the VPC Flow logs of Both VPC’s.
8. Enable Flow Logs for both VPC’s and push the Flow Logs to Cloudwatch log-groups, and store the logs in respective log stream for each VPC.
9. Create Security group for bastion host allowing port 22 from public.
10. Deploy Bastion Host EC2 instance in the public subnet with Elastic IP Associated.
11. Create S3 Bucket to store application specific configuration.
12. IAM Role granting access to ( Cloudwatch full access ) & with policy attached for S3 Bucket configuration.
13. Create Launch Template with below configurations.
14. Select Golden AMI.
15. Instance Type: t2.micro
16. User-data to pull the code from S3 Bucket to root folder of web-server, and start the httpd service and awslogs.
17. IAM Role granting access to Session Manager, and S3 Bucket created in previous step to pull the configuration.
18. Security group allowing port 22 from Bastion Host and Port 80 from Public.
19. Key-pair.
20. Create Target group, Instance as target type.
21. Create Application Load Balancer (ALB) in Public Subnet, and add Target group as target.
22. Create Auto-scaling group with Min: 1 and Max: 4, with 2 private subnets associated to 1a and 1b Zones.

IAM Policy:

1. To allow access to the S3 Bucket Only.

{

"Version": "2012-10-17",

"Statement": [

{

"Sid": "VisualEditor0",

"Effect": "Allow",

"Action": [

"s3:GetObject",

"s3:ListBucket",

"s3:GetBucketLocation"

],

"Resource": [

"arn:aws:s3::: YOUR\_BUCKET\_NAME/\*",

"arn:aws:s3::: YOUR\_BUCKET\_NAME"

]

}

]

}

User-data:

#!/bin/bash

yum update -y

systemctl start httpd

aws s3 cp s3://ed-web-project1/index.html /var/www/html/

systemctl restart httpd

sudo service awslogsd start

Create a policy with below permissions

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": [

"logs:CreateLogGroup",

"logs:CreateLogStream",

"logs:PutLogEvents",

"logs:DescribeLogGroups",

"logs:DescribeLogStreams"

],

"Resource": "\*"

}

]

}

Create a IAM role with Trusted entity type as AWS service and use case as EC2

And attach the above policy, edit trust relationship of the role with below text

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": "vpc-flow-logs.amazonaws.com"

},

"Action": "sts:AssumeRole"

}

]

}

# Validating Steps:

1. As DevOps Engineer, login to private Instance via Bastion Host.
2. Login to AWS Session Manager and access the EC2 shell from the console.
3. Browse the web application from public internet browser using Laod balancer, Domain Name. verify that page is loaded.