Lab Guide

AWS Training Program

AWS Certified Solution Architect Associate

Lab 01 – Create an EC2 instance

Task 1 – Create the EC2 instance

1. Select the EC2 service go to instances menu and launch an instance
2. Select a free tier eligible Linux or Windows image from “Quick Start” AMIs
3. Select the instance type - free tier eligible t2.micro instance type
4. Configure the instance details
    a. Select the VPC as the default VPC
    b. Select an availability zone you like
    c. Ensure “Auto-assign Public IP” is enabled
    d. Keep the rest as defaults
5. Add storage – Keep the defaults
6. Add tags
    a. Add the “Name” as a tag and value as the name of the EC2 instance
7. Configure security group – Select the default security group of the VPC
8. Launch the instance
9. From the pop-up window select “Create a new keypair” and download the key pair (the .pem file)

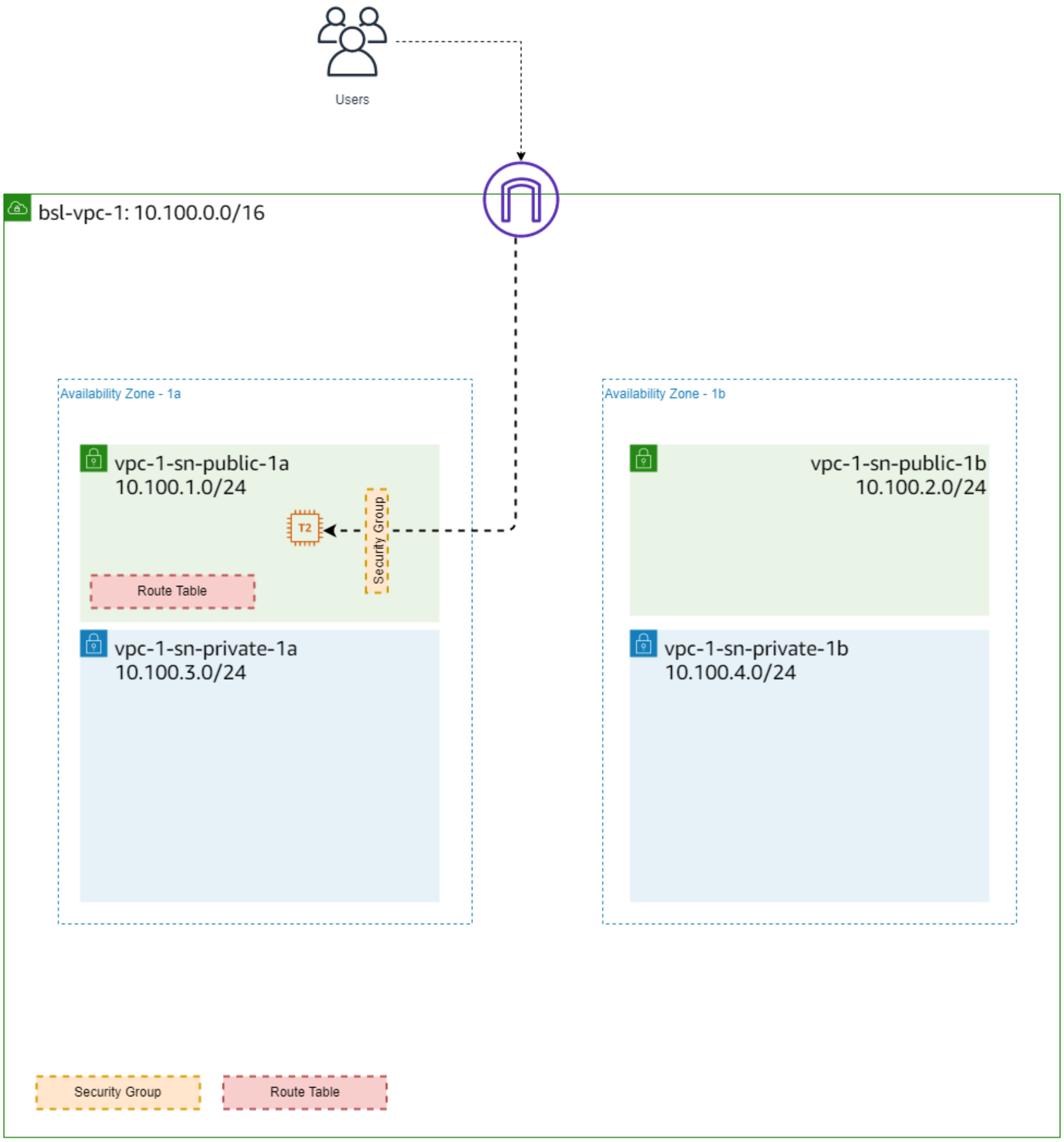
Note : We will have to change the default security group assigned to the instance to allow traffic from outside (From internet). Do the following to allow traffic form internet.

1. Go to VPC -> Security Group and select the default security group of the default VPC
2. Add an inbound rule to allow traffic from anywhere
    a. select source as anywhere and type as SSH (If connecting to Linux instance)

Task 2 - Connect to EC2 instance from your local machine

1. If you are connecting from Linux use the .pem file to connect
    a. Change the .pem file permission to be 400
        i. chmod 400 <key_name.pem>
    b. Connect to eh instance using ssh
        i. ssh -i <key_name>.pem ec2-user@<Public IP>
2. If you are connecting from windows and using putty
    a. Use Puttygen to convert the .pem file to .ppk
    b. Use .ppk file to connect through the putty

Lab 02 – Build a 2-tier application infrastructure



Task 1 - Delete the default VPC

a. Select a region (Should have the default VPC)
b. Notice the following
    a. Each default VPC already have following networking components inside it
        i. Subnet for each AZ
        ii. A Main routing table
        iii. A default NACL attached to subnets
        iv. Security group allowing all outbound traffic and inbound with in the security group
        v. An internet gateway
. Got to Actions and delete the default VPC

Task 2 - Create the new VPC structure

a. Create a VPC
    a. VPC CIDR : 10.100.0.0/16, Name : bsl-lab-vpc-1, Tenancy - Default
    b. Make sure DNS resolution and DNS hostnames are enabled for the VPC (Under "Actions" menu)
b. Check the following
    a. Default route table, NACL and a SG must appear in relevant screens
        i. Route table - "10.100.0.0/16 -> Local" route has been added automatically
        ii. NACL - All in and outbound traffic allowed
        iii. SG - All traffic within the SG allowed & all outbound traffic allowed
    . No subnets, or internet gateway are present
c. Create 4 subnets
    a. Name : vpc-1-sn-public-1a , CIDR : 10.100.1.0/24
    b. Name : vpc-1-sn-public-1b , CIDR : 10.100.2.0/24
    c. Name : vpc-1-sn-private-1a , CIDR : 10.100.3.0/24
    d. Name : vpc-1-sn-private-1b , CIDR : 10.100.4.0/24
    e. Ensure "Enable auto-assign public IPv4 address" setting is enabled for public subnets

Task 3 - Create the EC2 instance

a. Create a EC2 instance inside vpc-1-sn-public-1a
b. Download the keypair for the region or use an already existing keypair

Task 4 - Create an Internet Gateway

a. Create a internet gateway
    a. Create IGW and attached it to the VPC - IGW Name : vpc-1-igw
    b. Attached it to the VPC
b. Create a route table
    a. Name : vpc-1-rtb-igw
    b. Notice that the local route has been already added and cannot be deleted
    c. Add a new route : 0.0.0.0/0 --> IGW

c. Edit the subnet to point to the new route table
    a. Edit the "route table association" to the new one

Task 5 - Create a Security Group to allow traffic from outside

a. Create a new security group to allow traffic from internet to public subnet
    a. Name : vpc-1-sg-public-access, allow inbound - SSH, ICMP

Task 6 - Connect to EC2 instance from your local machine

b. If you are connecting from Linux use the .pem file to connect
```
ssh -i bsl-lsb-london.pem  ec2-user@18.130.56.162
```
b. If you are connecting from windows and using putty
    a. Use Puttygen to convert the .pem file to .ppk
    b. Use .ppk file to connect through the putty

Lab 03 – Configure AWS CLI

Task 1 – Download and install aws cli

a. Verify you have python 2.6 or above
    i. `python –version OR python3 --version`
b. Download the awscli bundle zip
    i. `wget https://s3.amazonaws.com/aws-cli/awscli-bundle.zip`

c. Unzip and run the installer
    i. `unzip awscli-bundle.zip`
    ii. `python3 install -b /home/bsl/.local/bin/aws`
    iii. `aws --version`

Task 2 – Create access keys

1. Login to iam and create an access key and secret key pair for a user

Task 3 – Configure aws cli

1. Configure the aws cli

```
# aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-2
Default output format [None]: json
```

Task 4 – Execute commands using awscli

```
#aws sts get-caller-identity
#aws iam list-account-aliases
#aws s3 ls
#aws ec2 describe-instances
#aws ec2 describe-instances --output json
#aws ec2 describe-instances --output text
#aws ec2 describe-instances --output table
#aws ec2 describe-regions | jq .Regions
#aws ec2 describe-instances | jq .Reservations[].Instances[].InstanceId
```

Lab 04 – Creating IAM Users, Roles and Policies

Task 1 – Allow user access through IAM policies

a. Go to IAM and create a policy
    a. Name : ec2-start-stop
    b. Actions : ec2:StartInstances + ec2:StopInstances + ec2:DescribeInstances
    c. Resources : the EC2 instance ID
b. Create a user and assign the policy
    a. Directly assign the “ec2-start-stop“ policy to user
c. Login as the new user and verify that user can only start and stop a particular EC2 instnace

Task 2 – Allow user access through roles

a. Go to IAM and create a role
    a. Trusted entity type : Another AWS account
    b. Name : ec2-role
    c. Account ID : AWS account ID
b. Create a policy to assume the role
    a. Service : sts
    b. Actions : AssumeRole
    c. Resources : role ARN of “ec2-role”
    d. Name : assume-ec2-role
c. Create a new user
    a. Create a new user with “assume-ec2-role” policy attached
d. Login as the new user and switch the role to “assume-ec2-role”

Task 2 – Allow EC2 instances access s3 using service roles

a. Go to IAM and create a role
    a. Trusted entity type : AWS Service -> EC2
    b. Name : ec2-s3
    c. Select the “AmazonS3FullAccess”
b. Go to EC2 instance and attached the policy
c. Verify the access
    a. Login to Ec2 install as a normal user
    b. Clear all the AWS cli access credentials
    c. Check the access (use the command “aws s3 ls”)

Lab 05 – Create and configure S3 buckets

Task 1 – Create S3 bucket and put and get object

a. Go to S3 service and create a bucket
    a. Note that the bucket name should be unique
b. Upload files to s3 bucket
    a. Note the S3 storage tiers and Encryption options
c. Download files from s3 buckets

Task 2 – Use the AWS CLI to work with S3

```
aws s3 mb s3://hglw-lab-test                                    #create a bucket in s3

aws s3 ls                                                       #list buckets in s3
aws s3 ls s3://bsl-aws-test                                     #list object inside a bucket

aws s3 cp s3-file-1.txt s3://bsl-aws-test                       #upload an object to s3
aws s3 cp s3://bsl-aws-test/s3-file-1.txt .              #download an s3 object

aws s3 cp aws-training s3://hglw-lab-test --recursive    #upload a directory to s3
aws s3 sync . s3://hglw-lab-test                         #sync a directory with s3
aws s3 sync s3://hglw-lab-test s3://bsl-aws-test                #sync two s3 buckets

aws s3 rm s3://hglw-ol --recursive                              #empty the s3 bucket. delete all objects
aws s3 rb s3://hglw-web                                         #delete a bucket. bucket should be empty
```

Task 3 – Enable public/anonymous access for objects

a. Go to “Permissions” and enable public access for the bucket
b. Go to the object enable public access for everyone
c. Now you use the object URL to access/download the object
d. Revert the changes again and disable all public access

Task 4 – Use S3 buckets policies to control access

a. Remove any existing IAM permission for the user from IAM
b. Make sure that you cant do any actions on s3 buckets
c. Go to the bucket -> Permissions -> Bucket policy
d. Write a bucket policy to make a user have full access to bucket
    a. Can use the visual policy generator
    b. Change the Principle and Resource accordingly

```
{
  "Id": "Policy1589603026088",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1589603023750",
      "Action": "s3:*",
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::hglw-lab-test/*",
      "Principal": {
        "AWS": [
          "arn:aws:iam::933325688911:user/lab-user"
        ]
      }
    }
  ]
}
```

Task 5 – Enable/disable s3 features

a. Enable object versioning
    a. To go bucket -> Properties
    b. Click on versioning and enable versioning
    c. Notice that now you get a versions show/hide button on overview tab
b. Enable server access logging
    a. To go bucket -> Properties
    b. Click on Server access logging and enable logging
    c. Provide a another s3 bucket as a target and a prefix
c. Enable encryption for s3
    a. To go bucket -> Properties -> Default Encryption
    b. Enable AES-256 - Server-Side Encryption with Amazon S3-Managed
    c. Keys (SSE-S3)
d. Add a lifecycle rule
    a. To go bucket -> Management -> Add lifecycle rule
    b. Create lifecycle rule to move all object in the bucket to Standard-IA storage after 30 days

Task 6 – Enable static web hosting

a. Create a bucket to host the website
b. Go to properties and enable "Static website hosting"
c. Enable web hosting for the bucket and enter the index and error html page names if different from the defaults.
d. Copy website/html files the file to the bucket
e. Enable public access to the bucket
f. Configure the following bucket policy allow public read access to objects in the bucket

a. Change the bucket name accordingly

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::hglw-web/*"
    }
  ]
}
```

g. Use the endpoint in “Static website hosting” to access the site

Lab 06 – Create and configure MySQL RDS instance

Task 1 – Create the subnets and security groups

a. Create a security group to allows access to RDS
b. Create 2 subnets in the VPC (vpc-1-sn-db-1a, vpc-1-sn-db-1b)
c. Create a DB subnet group and add the two DB subnets - bsl-lab-rds-sn-group

Task 2 – Create the RDS instance

d. Create the RDS instance
    i. Standard Create (Name : bsl-lab-mysql-01)
    ii. MySQL
    iii. MySQL Community - Latest Version
    iv. Free-tier
    v. DB instance identifier and password
    vi. DB instance class - Default Selected for Free tier
    vii. Enable storage autoscaling - Disabled
    viii. Multi-AZ deployment - Disabled
    ix. Additional connectivity configuration
        1. Select the newly created DB subnet group
        2. Select the security group
        3. Publicly accessible - NO
    x. Password authentication
    xi. Additional configuration
        1. Initial database name – bsldb

Task 3 – Connect to the RDS instance

```
mysql -u admin -p -h bsl-lab-mysql-01.cexyx42bwdqh.ap-south-1.rds.amazonaws.com
> MySQL [(none)]> show databases
```

Lab 07 – Configure CloudTrail and CloudWatch Logs

Task 1 – Browse and search the event history for events

a. Go to CloudTrail -> Event history
b. Apply the filters and make sure you can view the event history

Task 2 – Create a trail in CloudTrail to logs all API calls

a. Go to CloudTrail -> Trails
b. Create a trail with the following settings
    i. Name : all-ct-events
    ii. Apply to all regions : Yes
    iii. Select all s3 buckets
    iv. Keep the defaults for other settings
    v. Create a new s3 buckets for the logs

Task 3 – Create a log group in CloudWatch

a. Go to CloudWatch -> Log groups and create a log group

Task 4 – Configure CloudTrail to send logs to CloudWatch

a. Go to CloudTrail -> Select the trail - > CloudWatch Logs
b. Enter the name and configure continue

Task 5 – Verify you can see the logs in CloudWatch Logs

a. Go to CloudWatch -> Log Groups -> Select the log group
b. Select the log stream and then view and search as necessary for events

Lab 08 – Trigger Lambda function from CloudWatch

In this lab we will trigger a lambda function to stop all the running EC2 instances at 10pm everyday. We will be using CloudWatch Events and Lambda.

Task 1 – Create the lambda function

a. Create the lambda function with following settings
   i. Author from scratch
   ii. Function Name : lambda-ec2-shut
   iii. Select runtime as Python 3.7
b. Lambda will automatically create a execution role with the necessary permissions

Task 2 – Create the IAM role for lambda

c. Go to IAM and create the policy with the following permissions
d. Create a role name “lamnda-execution-role-ec2” and attach the above policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:Start*",
                "ec2:Stop*"
            ],
            "Resource": "*"
        }
    ]
}
```

Task 3 – Write the lambda code and test

e. Use the following code and do the necessary changes
   i. Change the regions and instance ids
   ii. Save the function
f. Go to Permissions and attached the execution role created

g. Test the function and make sure the code is working as expected
    i. Create a test event with any name you like
    ii. Execute the test

```
import boto3

region = 'ap-south-1'
instances = ['i-049fb50f0bc0d14a6', 'i-05ddfc3ea555eff10', 'i-
06590bdb57862b3ed']
ec2 = boto3.client('ec2', region_name=region)

def lambda_handler(event, context):
    print(event)
    print(context)
    ec2.stop_instances(InstanceIds=instances)
    print('stopped your instances: ' + str(instances))
```

Task 4 – Create a CloudWatch event to trigger lambda

a. Go to CloudWatch -> Events -> Rules
b. Create a rule with following settings
    a. Schedule event
    b. Cron expression : 0 22 * * ? *
    c. Use the GMT time in cron expression
c. Add a target
    a. Select the lambda service and then the function name
d. Give a name and save the rule
e. Make sure the lambda gets triggers at the specified time

Lab 09 – Using SNS with CloudWatch and Lambda

Task 1 – Create a SNS topic

a. Go to SNS -> Topics and create a topic
b. Enter a name and a display name and keep everything else on default

Task 2 – Create a subscription

a. Go to SNS -> Subscription and create a subscription
b. Select
    a. Topic ARN : SNS topic ARN
    b. Protocol : AWS Lambda
    c. Endpoint: Lambda ARN
    d. Keep everything else as default

Task 3 – Test the SNS topic

a. Go to SNS -> Topics and select the SNS topic and then publish message
b. Enter a name for the subject and for the message body and publish the message
c. Make sure the lambda got executed

Task 4 – Trigger SNS topic through CloudWatch

a. Go to CloudWatch -> Rules and create a rule
b. Create a event patters as you wish and add the SNS topic as the target
c. Trigger the event make sure the lambda gets executed on the event

Note : CloudWatch can directly trigger lambda and no need to use SNS as a middleman. This is only to understand the flow and the service integration techniques. Only reason you might want this architecture is if you want to trigger a lambda from a CloudWatch rule or event in another regions.

Lab 10 – Configure DNS – Route53

Task 1 – Verify the current DNS client settings

d. EC2 DNS is pointing to the .2/+2 AWS provided DNS
    a. Execute “cat /etc/resolv.conf” to confirm
e. EC2 instances can by default resolve both internal (inside VPC) and external DNS name through AWS provided DNS
    a. nslookup google.com
    b. nslookup ip-10-100-3-172.ap-south-1.compute.internal
f. EC2 public DNS name can also be resolved from outside
    a. nslookup ec2-13-126-4-26.ap-south-1.compute.amazonaws.com

Task 2 – Create a private hosted DNS zone (PHZ)

a. Go to Route53 - > Hosted zones
b. Create a hosted zone with following settings
    a. Domain name : lab.local
    b. Type : Private Hosted Zone for Amazon VPC
    c. VPC : VPC for the current EC2 instances
c. Create a records set with the following settings
    a. Name : app1.lab.local
    b. Value : <IP address of the EC2 instance>
    c. Routing policy : simple
d. Make sure you can resolve the DNs name from side the EC2 instances in the VPC
    a. nslookup app1.lab.local
e. Make sure to delete the Private Hosted Zone after the lab
    a. Delete the record sets first
    b. Delete the PHZ

Lab 11 – Using DynamoDB

ProductID	ProductName	ProductDescription	Weight	Price	Quantity
1000	Dhal	Dhal 500G Packet	500	70	100
1001	Sugar	Sugar 1Kg Packet	1000	150	200
1002	Rice	Rice Samba 1Kg Packet	1000	120	400

Task 1 – Create a DynamoDB table

a. Go to DynamoDB service -> Tables and create table
b. Use the following settings
    a. Table Name : EMPLOYEE-INFO
    b. Add the Primary Key as NIC
    c. Add a sort key as FirstName
    d. Use default Settings
c. Once table is created go to items tab and create items
    a. Add the ProductID and ProductName values for all 3 items
d. Search for products using filters
    a. Use both scan and query mechanism for search
e. Select item and edit and add other attributes for the products
f. Search the products based on the other attributes

Lab 12 – Configuring Elastic Load Balancing

Note : This lab uses RdHat 8 AMI

Task 1 – Launch an EC2 instance

a. Go to EC2 service and launch an Linux EC2 instance
b. Place the EC2 instance in a private subnet

Task 2 – Configure ngnix webserver to server

a. Connect to the EC2 instance and configure the ngnix webserver to serve some static HTML content as follows

```
dnf install nginx
systemctl start nginx
curl -I 10.100.3.200
systemctl enable nginx
cp /usr/share/nginx/html/index.html /usr/share/nginx/html/index.old.html
echo "Ngnix Web Server - lab-web-1a" > /usr/share/nginx/html/index.html
curl  10.100.3.200
```

Task 3 – Create an AMI from the EC2 instance

a. Stop the current Ec2 instance
b. Select the instance and create an AMI from Actions -> Image -> Create Image
c. Start the EC2 instance after the image creation completes

Task 4 – Create and configure another EC2 instance from the AMI

a. Lanuch another EC2 instance from the newly created AMI
    a. Select another availability zone
    b. Configure an IP address manually
b. Make sure ngnix is working
c. Change the HTML content to reflect the web server host

```
curl -I 10.100.4.200
echo "Ngnix Web Server - lab-web-1b" > /usr/share/nginx/html/index.html
curl  10.100.4.200
```

Task 5 – Create the ELB

a. Go to EC2 -> Load Balancing -> Target group and create a target group
    i. Target group name : lab-tg
    ii. Target type : instance
    iii. Protocol : HTTP
    iv. Port : 80
    v. VPC : Select the same VPC of the EC2 instances
    vi. Keep everything else in defaults
b. Go to the Target tab in target groups and register the targets
    i. Go to Edit and add the 2 web instances and add to registered
    ii. Make sure two instances are shown under Registered targets in Targets tab
c. Go to EC2 -> Load Balancing -> Load Balancers
d. Create the load balancer with the following settings
    i. Select Application Load Balancer
    ii. Name : lab-elb
    iii. Scheme : internet-facing
    iv. Listener : HTTP port 80
    v. Under Availability Zones select the Public Subnets in each AZ
    vi. Select the security groups allowing inbound traffic from internet
e. Under Configuring routing select the target group create earlier
f. Ensure Security groups settings are correct
    i. Ensure SG in the ELB allows HTTP/HTTP ports
    ii. Ensure SG attached to instances allows traffic from ELB

Task 6 – Access the pages and verify the configuration

a. Go to EC2 -> Load Balancing -> Load Balancers and select the load balancer
b. Copy the DNS and try to access it through a web browser
c. Refresh it few times and make sure the you are being server from both web server instances

Task 7 – Cleanup the resources

a. Make sure to cleanup the all the following resources
    i. 2 Instances
    ii. Load balancer
    iii. Target groups
    iv. AMI
    v. Snapshot used for AMI creation

Lab 13 – Configuring Automatic Scaling Groups

Task 1 – Launch an EC2 instance

c. Go to EC2 service and launch an Linux EC2 instance
d. Place the EC2 instance in a private subnet

Task 2 – Configure ngnix webserver to server

b. Connect to the EC2 instance and configure the ngnix webserver to serve some static HTML content as follows

```
dnf install nginx
systemctl start nginx
curl -I 10.100.3.200
systemctl enable nginx
cp /usr/share/nginx/html/index.html /usr/share/nginx/html/index.old.html
echo "Ngnix Web Server - lab-web-1a" > /usr/share/nginx/html/index.html
curl  10.100.3.200
```

Task 3 – Create an AMI from the EC2 instance

d. Stop the current Ec2 instance
e. Select the instance and create an AMI from Actions -> Image -> Create Image
f. Start the EC2 instance after the image creation completes

Task 4 – Create a launch configuration

a. Go to EC2 -> Auto Scaling -> Launch configuration and Create launch configuration with the following settings
    a. Select the AMI created earlier
    b. Name : lab-asg-launch-config
    c. Select a security group with the right access

Task 5 – Create an auto scaling group

a. Go to EC2 -> Auto Scaling -> Auto Scaling group and create auto scaling group with the following settings
    a. Select the launch configuration created earlier
    b. Enter a group and select 2 private subnets from 2 AZs
    c. Select “Use scaling policies to adjust the capacity of this group ”and configure the group size
    d. Select a SNS topic to send the notifications
    e. Select the target group created earlier

Task 6 – Access the pages and verify the configuration

a. Go to EC2 -> Load Balancing -> Load Balancers and select the load balancer
b. Copy the DNS and try to access it through a web browser
c. Refresh it few times and make sure the you are being server from both web server instances
d. Go to the EC2 -> instances and note that you have the instances created by auto scaling

Note : You will have to create some load for the auto scaling to trigger. If you are using CPU utilization use the something similar to following to generate some load.

```
while true ; do
echo "I am still running"
done
```