EC2

cloud Found → Mod 6 → Introduction to Amazon EC2

① EC2
Instances → [Launch instance]

Name [cafewebserver]

ubuntu → Ubuntu server 24.04 LTS (HVM). SSD Vol Type

Instance type [t2.micro]

Key pair [vockey]

Network settings → [edit]

VPC → lab VPC

Subnet → Public subnet 1

Auto assign public IP → Enable

Firewall (security group) → [◉ create security group]

Security grp name → [cafeweb sg]

Inbound sec grp rules → SSH - MyIP
Add sg rule → HTTP - Anywhere

Configure Storage

1x [8] GiB [gp3]

[Launch instance]

Instances → ☑ cafe webserver → ☐ public IP & open ⑨

Aws Details

Download PEM

cd Downloads.

⇒ sudo chmod -R 400 labsuser.pem

→ Then download PEM

cd Downloads
ls
sudo chmod 600 labsuser.pem
sudo ssh -i labsuser.pem ubuntu@ Public ip
Yes
sudo apt-get install php apache2 libapache2-mod-php
php-mysql mysql-client mysql-server -y
sudo git clone ____ url - Cafe Dynamic website
cd Cafe_Dynamic_Website/ → cd Mompopcafe
ls
sudo cp -rf */var/www/html/.* → copy all files to /var/www/html
sudo rm -rf /var/www/html/index.html → deletes existing default web apache so that our file will be there

~~[sudo apt-get install php apache2 libapache2-mod-php php-mysql mysql-client]~~

sudo mysql -u root -p
> create database cafedb;
> create user 'msis'@'localhost' identified by 'msis@123';
> grant all privileges on cafedb.* to 'msis'@'localhost';
> exit
cd..
cd mompopdb/
sudo mysql -u msis -p
> show databases;
> use cafedb;
> show tables; //empty set
> source create-db.sql;
> show tables;
> exit

db url = localhost
db name = cafedb
db user = msis
db password = msis@123

sudo systemctl restart apache2

sudo systemctl restart apache2

Instances → public IP → ⑨ - Mobaxterm

Instance → public IP → ⑨ Mobaxterm

cd /var/www/html
ls
sudo nano GetAppParameter.php

VPC

cloud Foundation → Mod5 → Lab2 Build your VPC & Lauch a Webserver

{ VPC
EC2

⑨ VPC
create VPC

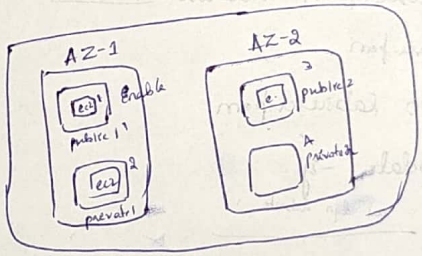
⊙ VPC & more

▸ customize subnets CIDR blocks

10.0.0.0/24	public 1a
10.0.2.0/24	public 1b
10.0.1.0/24	private 1a
10.0.3.0/24	private 1b

NAT gateway → In 1 AZ
VPC endpoints → None

create VPC view vpc



⑨ search ec2 : Lauch public & private instance

public instance

cafewebserver
ubuntu 24.04 VT
t2.micro vockey

Network settings edit
VPC → aiVPC project vpc
public 1a
Enable

security group
websg
ssh anywhere
HTTP anywhere
configure → 8 gp3
Lauch instance

Private instance

db server

ubuntu 24.04 SSSVT

t2.micro vockey

Network settings edit

VPC → our VPC

Private 1a

Disable

Security group

dsg

ssh Custom sg

Mysql / Aurora custom sg

Configure 8 gb3

Lauch instance

Step3: Download PEM

In Terminal: ls | grep 'pem'

cd Downloads

sudo chmod 600 labsuser.pem

sudo scp -i labsuser.pem labsuser.pem ubuntu@ public ip :/home/ubuntu

sudo ssh -i labsuser.pem ubuntu@ public ip

ls // labsuser.pem

sudo chmod 600 labsuser.pem

sudo apt-get update -y

~~sudo git clone~~ url dyn website

sudo apt-get install php apache2 libapache2-mod-php php-mysql mysql-client -y

sudo git clone dyn web url

cd Cafe-Dyn

cd mompop cafe

sudo cp -rf * /var/www/html/.

sudo rm -rf /var/www/html/index.html

sudo systemctl restart apache2

⑨ check Browser public ip

Menu ✓ Order ✗

cd

sudo ssh -i labsuser.pem ubuntu@ private ip

sudo apt-get update -y

sudo apt-get install mysql-server -y

sudo mysql -V

sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf

// bind-address = 0.0.0.0

sudo systemctl restart mysql

sudo mysql -u root -p

> create database cafedb; // → all ip

> create user 'msis'@'%' identified by 'msis@123';

> grant all previleges on cafedb.* to 'msis'@'%';

> exit

$ exit → private instance

webserver → cd Cafe-Dynamic-Website/mompopdb

ls

sudo mysql -h private ip -u msis -p

> use cafedb;

> show tables // empty set

> source create-db.sql

> show tables

> exit

select * from product;

cd /var/www/html/

ls

~~sudo rm index.html~~

sudo nano get-AppParameters.php

edit db_url : "Private ip"

db_name: cafedb

db_user: msis

db_password: msis@123

sudo systemctl restart apache2

→ Menu ✓ Order ✓

RDS

Challenge (cafe): Migrating a Database to Amazon RDS

Cloud Arch → mod 6 →

① EC2 → ☑ cafeserver
copy public ip paste in Browser // Hello from cafe webserver

② RDS
1st create database

◉ Mysql
MySQL 8.0.42

Templates
◉ Sandbox (× dev/test)
Availability & durability: Single-AZ DB instance

settings → DB instance
cafedb

Credentials
master user name: admin
Idfmanaged
Msis1234

Instance conf: ◉ Burstable classes
d3, t3. micro

Storage: General purpose SSD (gp3)
allocated: 20

connectivity → virtual private cloud (VPC)
◉ LabVPC

DB subnet group → existing VPC security groups
dbsg
☐ uncheck default

Monitoring → ☐ uncheck
enable enchanced monitoring

Create

→ ② AWS system Manager
start session → Create session
◉ cafe server → start session

sudo su
su ec2-user
cd /var/www/html
ls
cd cafe
ls
sudo mysql -u root -p
password: paste

> show databases;
use databases;
use cafe_db;
show tables;

→ In RDS
◉ cafedf ← click
connectivity & security
VPC security group
→ edit
Inbound rule

Add rule
Type: MySQL/Aurora
custom: cafesg
→ Save rule

→ exit
mysql -h cafedb.C7A (endpoint) -u admin -p
pass: Msis1234
show databases;
create database cafe-db;
show databases;
exit
cd

→ ② Secrets Manager
→ Secrets
→ password
→ Retrieve secret value
copy

← endpoint

> cd /var/www/html

mysqldump --databases cafe_db -u root -p > cafe.sql

↳ secrets.txt → paste

nano cafe.sql

mysql -h cafe -u admin -p

↳ Mis1234

> use cafe_db;
show tables;
source cafe.sql
show tables;

>exit

sudo systemctl stop mariadb.service
sudo systemctl status mariadb.service "inactive / dead"
sudo systemctl restart httpd

Browser → cafe Menu

④ secrets manager

db URL → retrive → edit → paste (cafedb endpoint) → save
db user → Mis1234
db password → Mis1234
db user → admin Save

IAM - AWS Identity & Access Management

cloud Foundation : Mod 4 → Lab 1 : Introduction to AWS IAM

① IAM

check region → N. Virginia us-east-1

user groups

S3 support → Add users → user-1
EC2 support → Add users → user-2
EC2 Admin → Add users → user-3

users

⇒ user-1 (read only access to S3)

security credentials

copy console sign in link

& paste it in incognito window in Browser

username : user-1
password : Lab-Password1

sign in → S3 bucket → only S3 budget access no access for other like EC2 IAM access denied.

② S3 : Sample bucket is there we can only view this.

⇒ user-2 (Read only Access to EC2)

user-2 → security credentials.

copy console sign in link

& paste it in incognito window in Browser

username : user-2
password : Lab-Password2

③ EC2 → read only access → can't launch an instance throw an error.

cannot start & stop instance.

Instance → Instance state

start instance } Not access
stop instance

check region → us-east-1 N. Virginia

⇒ user-3

user-3 → security credentials

copy console sign in link → = New private window
& paste in incognito window

username: user-3
password: lab-Password3

EC2 → you can start & stop instance

✗ S3 cannot/failed to create a budget error
no view

instance → ☑ LabHost → Instance state ▼
stop instance ✓
start instance ✓

S3 Bucket

Cloud Arch → Mod 4 → challenge lab: creating a static website for the cafe.

S3

create bucket

⊙ general purpose only

Bucket name: angelcafeweb.com

* keep the bucket name unique.

⊙ ACL's disabled

Uncheck ☐ Block all public access

☑ I acknowledge

Bucket versioning → ⊙ Enable ← permissions ACL → Enable

create bucket

Go to created bucket.

⊙ angelcafeweb.com

properties → static website hosting → Edit

⊙ Enable

Index document: index.html

Save changes ✓

Objects → upload → drag drop files

3 files → CafeStaticWebsite

↳ ~~cafe-websites-deployment.pdf~~

✓ { css
images
index.html
~~README.md~~

upload ✓

✗ Permissions:

Object ownership → edit

ACL's enabled → ☑ I acknowledge

Save changes

Objects → check all boxs → Actions ▼
→ Make public using ACL
→ Make public

Go to Properties
→ Static website hosting
→ Bucket website endpoint
URL
copy the link & paste in Browser
cafe website ✓

EBS - Amazon Elastic Block Store
cloud Found^ → Mod 7 → lab 4: working with EBS

① EC2
Instance → ☑ lab → Storage → 8 GiB
Volumes → ☑ vol 8 GiB → create volume
→ create volume
size (GiB) 1 Gib
Availability zone us-east-1a
Key Tag → Key Name Value Myvolume
create volume ✓
☑ Myvolume → Actions ← refresh
attach volume
Instance lab
Device name /dev/sdf
Attach volume ✓

Mounting step:
Instance → ☑ lab → connect → EC2 Instance connect (SSH client)
→ connect

```
df -h
sudo file -s /dev/xvdf          // :data
sudo mkfs -t ext3 /dev/xvdf     // make a fs
sudo file -s /dev/xvdf          // which file
sudo mkdir /mnt/myvolume        // creating a folder to mount
sudo mount /dev/xvdf /mnt/myvolume   // mounting
df -h                           // check volume is mounted
```

60K 92HM 1Y.

cd /mnt/myvolume
sudo touch file.txt
sudo chown -R ec2-user:ec2-user .
cat >> file.txt
This is the ex for ebs volume ^C enter
cat file.txt
// This is the ex for ebs volume

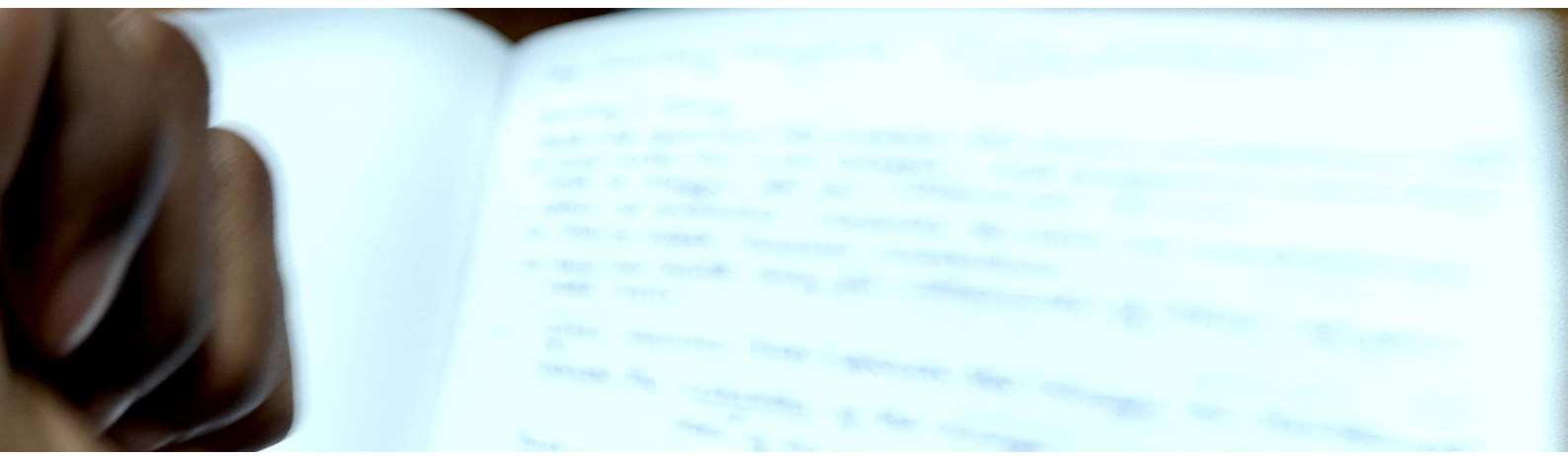
⇒ Go to ec2 → Volumes
☑ My volume → Actions ▾
Create snapshot

⇒ + Add tag
key Name value MyvolumeSnapshot
Create Snapshot ✓

⇒ Go to Snapshot ← refresh
☑ MyvolumeSnapshot → Actions ▾
→ create volume from snapshot
size (GiB) 1
Add tag
key Name value RestoredVolume
create Volume ✓

⇒ Go to volumes
☑ Restored volume → Actions ▾
attach volume
Instance lab
Device name /dev/sde
attach volume ✓

⇒ Go to connect:
cd
df -h
sudo file -s /dev/xvde
sudo mkdir /mnt/resvolume
sudo mount /dev/xvde /mnt/resvolume
cd /mnt/resvolume
ls
// file.txt lost+found ✓



Cafe Inventory Management: Serverless Architecture

Running a Query

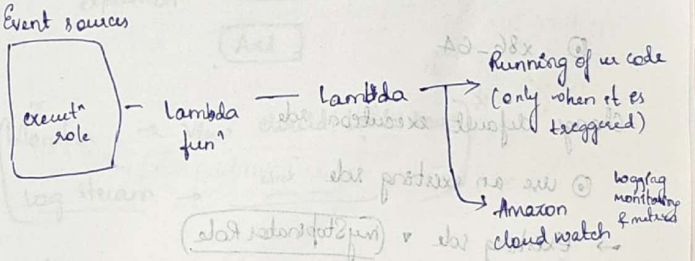
Need not provision & maintain the server in order to run input code
⇒ come with the code /snippet that performs a same task
that is trigger for few milliseconds to run.

- either on particular schedule @ when an event occurs.

⇒ this is called Serverless architecture.

⇒ You are build only for milliseconds of time of your code runs.

- when someone puts / uploads the image to the S3 bucket
  & extracts the metadata of the image → Lambda function
  ↓
  data of the image.



250MB ← should never exceed
Instance id ← to stop all the services / ec2 instances.
IAM role ← should be specified.

- Schedule-based Lambda function.
- Event-based Lambda function
- privacy ↓
  permissions: ↓
  effect, action, resources

AWS Lambda

Cloud Foundation: Mod 6 → Activity: AWS lambda

Event Bridge event → lambda function triggered → EC2 instance stopped

① EC2

Instance1 → copy → Instance ID

② Lambda

Create a function

⊙ Author from scratch

Fun° name: mylambda

Runtime: Python 3.11

⊙ x86_64

Change default execution role

⊙ use an existing role

→ Existing role: myStopInstanceRole

Create function

Goto lab instructions

13th step import boto3 ← copy code

cut paste in lambda_function.py

change region = 'us-east-1'
instance = "paste ID"

changes need to be done

Deploy

EC2 instance → Instance 1 → Instance ID copy

region = 'us-east-1'
instances = ['Instance id']

myLambda
Layers

After deploy ↓

+ Add trigger

Trigger configuration → Event Bridge (cloudWatch Events)

⊙ Create a new rule

Rule name: myrule

Rule type ↓

⊙ Schedule expression

Schedule expression: rate(1 minute)

Add

Monitor → View cloudWatch logs

Log stream → click

After Add ↓

Go to EC2 instance → check instance had stopped

Go to lambda → Monitor → View cloudWatch logs

↓

Log group : Log stream

Cloud Archi → Mod 14 → Guided lab: Implementing a Serverless Architecture on AWS

upload inventoryfile → S3 bucket → lambda fun → DynamoDB table ↔ dashboard app ← Amazon

- Lambda
- S3
- dyna
- Sns

③ Lambda — fetching, reading & writing from S3 to dynamoDB

create a function

load-inventory-lambda

python 3.9

change default

⊙ use a existing

▾ Lambda-Load-Inventory Role — read + write

create fun

copy from directory → paste code

```
import
```

Debby

us-east-1

create bucket

General purpose

inventory-174 → 3 to 4 digit

as it is default settings → create bucket

[S3 → to send event notification for uploading a data deleting post, test]

configure the S3 bucket

properties

↳ Event notification

create even notification

event name: inventoryload — triggered

event types: ☑ All object create events

Destination ⊙ lambda fun

⊙ choose from your lambda fun

Lambda fun: LoadInventoryLambda

save changes

② DynamoDB — attributes colum

tables

↳ Poem

explore items

① S3 bucket ⊙ General purpose buckets

↳ upload ← mybucket ← Objects

↳ upload

③ DynamoDB → run

AWS details → Dashboard link → copy → paste in Browser

Inventory Dashboard

⑧ Lambda

create function

Storenotejec

python 3.9

x86_64

change

① use an

Lambda-check-Stock-Role ▾ create fun^

copy from directory → paste →

Pub sub mechanism

sns → No stock

↳ lambda function when 0 is the subscribers to no stock reads an item from Dyn → 0 → pushes the msg to sns reads as no stock.

⑧ Sns Simple Notification Service

NoStop

NextStep

⊙ Standard

Name NoStop

Displayname CafeInventory

ARN used

Create topic

create subscription

protocol Email

Endpoint shaluja...@gmail.com

create subscription

⑧ Lambda

+ Add trigger

Dynamo

Inventory

Add

Create event

test event

Save

run

Guided lab → .csv files → Copy any 1

touch mydata.csv

nano mydata.csv

S3 → ☑ ☐ } Delete objects permanently delete

↓

upload mystore.es ☑

Dynamo DB → run → Calcutta } files.

Elastic Beanstalk

CloudFound : Mod 6 → Activity - AWS Elastic Beanstalk

- high availability & Fault tolerance
- elastic & scalable Infrastructure.
- cloud watch → continuously monitor the metrics & checks health
- Load Balancer - ELB

Elastic Beanstalk → platform as a service that automatically runs & upload the code

④ Elastic Beanstalk

→ environment → click

platform → untarstalled 9

Health → Enhanced instance health ✓2

Monitoring

→ Environment → → configuration

→ Instance traffic & scaling

Edit

mysql

8.0.40

t3.small

20

admin

Mdis1234

⑨ ec2

Instance

Load Balance

Auto scaling groups → scaling limits (min-max)

2-6

In Elastic Beanstalk

upload & deploy

(tomcat.zip)

Deploy

Beanstalk → Domain → Browser

create environment

app name (cafeapp)

platform (PHP)

Next

Elastic Load Balancing

1 Database, multiple no. of webservers

A service that takes the request from client to a { IP address, ec2 instance, container, λ func } & distributes it among load balancer will have the & monitor the health of the server & devert the traffic

* Can't distribute across one or more availability zones.

Elastic load Balancer →
↳ automatically scales up & down

* used to prevent from Ddos attack by using ELB
* types of load Balancer:
  1. application load
  2. Network load
  3. Gateway load
  4. Classic load

1. Application Load Balancer
* works at the app^n level of the OSI model
* devert the traffic on { IP address, ec2 instance, container, λ fun^n } based on the content requested by ex.
* mainly supports HTTP & HTTPS protocol.

2) Network Load Balancer:
* operates at layer 4 (transport layer)
* devert the traffic on based on IP protocol
* capable of handling millions of requests per second with high traffic low latency
* TCP & UDP protocol

3) Gateway Load Balancer:
* Operates

How Elastic load Balancing works

how to configure.

→ create a process called listener. & on port checks for connection request which receives a traffic & from port then to devert to the

* Target group & devert to target group on that particular protocol.
  ↳ all ec2 instance
  HTTP instance listening on port 80.
* Listen also monitors the health of the target and deverts the traffic only to the healthy target.

load balancing monitoring:

* using cloudWatch metrics:
* Elastic load Balancer: latency metric goes below the threshold to particular alarm.
* Access logs & cloud trail logs.
  ↓ Basic info when access
  ↓ detailed info of the load Balancer.
* cloud Watch.
  ↳ AWS infra^ utilize, capacity, reduce cost automation
* It is a monitoring & observability service of AWS. also it can monitor application running on top of AWS
* cost optimization by collecting the metrics on cloudWatch
* we can collect the metrix
  ↓
  It as a variable
  configure to cloud watch
* cloud watch alarm → proactive (statistical analysis)
  ↳ collect variables and create metrix → create an alarm
* with 5 min goes below the threshold generate a cloudWatch alarm →

* Event : ec2 instance from running to stopped state.
  → define → target to be triggered
  → rule → shedule basis
  or change of state of some resources

Specifications:
Namespace → resource trying to monitor
Metric → CPU utilization 70%
Statistic → statistical threshold ex: above 50%.
Evaluation period → 60 sec → collects the CPU utilization
conditions 5min → 50% for 5 mins it will be above the statistical volume than
> , < before generate an alarm.
Additional conf : ish
Actions : target : SNS notify what actions are taken.
ebs

Scaling

Amazon EC2 auto scaling
* Add more ec2 instance when workload requirement
  @ remove
* automatically scaled up & down
* schedule more resources, (on particular day ex: Wednesday I need more resources)
  → dynamic scaling
  → manual
  → schedule
  → auto-scaling with predictive
  & dynamic can be added.

EC2 auto scaling
* Group of ec2 instance which are running same copy and
  @ web application.
think as
* all servers are single logical Unit
  desired capacity, minimum & maximum

Scale out → add resource
Scale in → remove resources

* continuously monitor all EC2 instances of the group

Launch configuration : It is a template which ec2 what are all is needed, to add ec2 instance

ec2 instance must know the ami
* AMI
* instance
* IAM role
* security groups
* EBS volumes

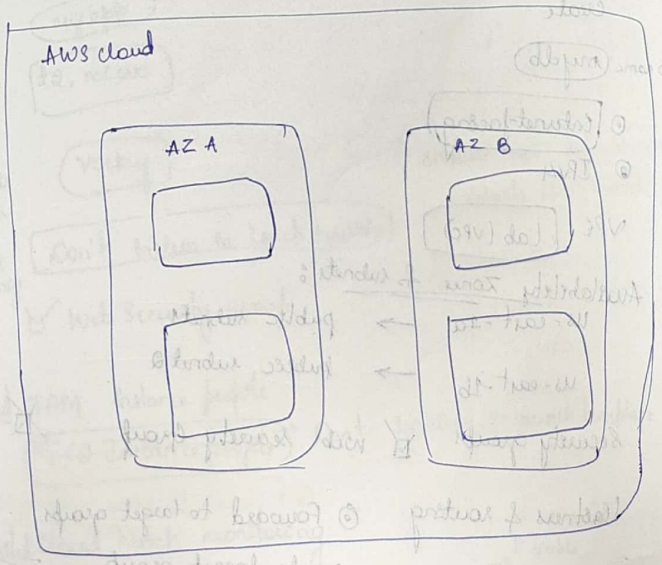
Auto scaling group
when →
* VPC & subnets
* load balance

* we define the fleet of the group
  min, max, desired

when → Manual, Maintain current servers, schedule, predictive, dynamic scaling

VPC

Cloud Foundation : Mod 10 → Section 3 Video 2
Lab 6 Scale & Load Balancer mani



Load Balancer

Mod 10 : Lab 6 Scale & load Balance your Architecture

① EC2

instance webserver 1 → copy public ip & paste in Browser
http:// pub ip

Instance ☑ → Actions
Image & templates
Create image

name → my app
descrip → my app
& Key - Name Value - my app name

Load Balancer
Create a load Balance
App load Bal
Create

Lb name (mydb)
⊙ Internet-facing
⊙ IPv4
VPC (Lab (VPC))
Availability Zones & subnets :
us-east-1a → public subnet1
us-east-1b → public subnet2
Security groups ☑ web security Group ☒ default
Listeners & routing ⊙ Forward to target groups
create target group
create (myec2 target gro) HTTP

→ Basic config
⊙ Instances
tg name (my ec2 target group)

Next
create target group

Firewall → Nacs

create load balancer

⇒ Lauch template

Create
name (myasgtemplate)
descri blank
☑ provide guidance to help me set up

My AMI
⊙ owned by me
(my app)
Instance type (t2. micro)
key pair name (vockey)
Network settings (Don't include in Lauch template)
☑ Web Security group

should not say networking details like subnet1 t2

Advanced IAM Instance profile
(X EC2 Instance profile) Don't include in launch template
Detailed Cloud Watch monitoring
Enable
→ " enable every sec
create lauch template

Launch template

☑ _____ Actions → create auto scaling group

① Name (myasg) ↘ automatically selected launch template (asg template)

Next

② Network (Lab (VPC)) VPC → Availability zones & subnets

☑ us-east-1a private subnet 1

☑ us-east-1b private subnet 2

Next

③ Load balancing

⊙ Attach to an existing load balancer

⊙ choose from your load ba

existing load balancer target groups

(my ec2 target group | HTTP)

Health checks.

Additional health check

☑ Turn on Elastic LB health checks

Next

④ Desired capacity

2

scaling

min 2 max 4

Automatic scaling → ⊙ Target tracking scaling policy

Metric type Average CPU utilization

Target value

Instance warm 60 seconds

Additional settings

☑ Enable instance scale-in protection

Monitoring

☑ Enable group metrics collection within cloudwatch

Next

Next

+Add tag

Key Name Value mywebservers

create Auto scaling group

⑤ ec2 instances ☑ mywebservers

load balancer ☑ myelb DNS name ☐ copy paste in browser

⑥ load test 100% RDS 1a 2b ⑦

⑨ Cloud Watch

Alarms

In alarm → 1 in alarms after 5 min

after this ec2 instance one more instance will

☐ mywebservers be running

cloud → Mod 10 → challenge (cafe lab: creating a Scalable & Highly available environment for the cafe

Each subnets 1.

vpc → Route table

NAT gateway

create NAT gateway

my nat

Public subnet 2

Key Name value my nat

☑ privat subnet 2 244 → No Internet

edit route

Target

my nat ✓

☑ private subnet 4

Network Eng

Ip address: unequely address identify the system in an Network system.

IpvA: 32bit, 4 decimal represent 8 bit

each decimal has $2^8 = 256$

it can vary from → 0 to 255

Ipv6: 128bit,

collection of 8 group & each group has 4 letter & no.

16 bit hexadecimal, can vary from 0 to 0000

Classless Inter-Domain Routing (CIDR)

* Identify the network and its hosts

to define an Ip address of an network

* 24 bits are fixed in 32

Tells you how many bits are fixed

Network identifier (routing prefix) Host /24

192. 0 . 2 . 0

vary from 0 to 255

flexible

fixed

Network address → 192.0.2.0

first Ip address is allocated to router

router: hop from 1 network to another network

switch: connect devices within a network

Hidy gateway: link to within a system to reach internet.

In ubuntu terminal:

route -n

// Destination Gateway Genmask

172.18.181.1

1 allocated to router

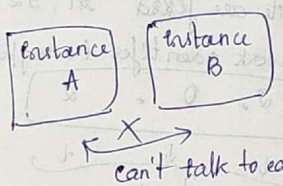
Switch: works in 2 layer of OSI (datalink layer)
↳ looks for MAC address
moves data from one network to another network
→ARP protocol

Router: works in 3 layer of OSI model
↳ looks for IP address
↳ will remember MAC address & IP address
source & destination address.

Amazon VPC
+
* can create our own Networks

your own logical network.

logically isolated.
All has private IP address



* Using only public IP address they can communicate to eachother but using private IP address it can't.

VPC is restricted to region
subnets are limited to availability zones.
↳ for each subnet we need to allocate the CIDR (the range)
↳ devided some set of IP addresses.

→ Not two subnets can have the same CIDR / IP address range
→ we can create a largest IPv4 → x.x.x.x/16 @ 65,536 address
→ we can create a smallest IPv4 → x.x.x.x/28 @ 16 addresses
4 bits are remaining
$2^4 \rightarrow 16$

AWS → always kept reserved
10.0.0.0 → Network address
10.0.0.1 → Internet communication
10.0.0.2 → Domain Name
10.0.0.3 → Future case
10.0.0.255 → Network broadcast address

Public IP address types → ip is dynamic
* Default vpc → will allow public
* for own vpc →
☑ Enable Auto assign public address
* when you stop & start the address, public ip address would have been changed.
↳ to Make it static
Elastic IP address
* can only have 5 elastic ip address for every region/service
limited to
hard limit → never justified
soft limit → can be justified

Elastic network interface
* virtual network interface
* It is scalable & adjustable
* It evaluates whether a traffic to attach or detach
* attributes follow
primary can't be detattched, only which we created or attached can be detached.

NIPK holds all

Route tables & Routes
* Every one Route table can be attatted to multiple or one subnet

Route table →

Destination	Target
10.0.0.0/16	local → by default will be there

Scanned with CamScanner

Route table → set of rules to ~~to~~ has source & destination ip address
* acts at the subnet level

VPC Networking

resource → ec2, lamba, ebs, elastic load balancer.

Internet gateway → allows resource to communicate to outside world.
* Managed highly scalable resource
All resource will have private ip address
NAT → converting private IP address to public IP address & vesa versa

Any traffic of 0.0.0.0/0 → should go through (keyword) NAT gateway.

NAT (Network address translation) Network address transl
* acts as target
* converts private to public & vesa versa in public subnet

NAT gateway.
* use a NAT gateway in any one public subnets.
* Will bolock anyone trying to access the private instance
* Only allows the resources to access to communicate to the private instance.
* NAT instance is an unmanaged service & doesn't provide bandwidth or throughput.
* Route table with target gateway, NAT gateway.
* subnets are for reliability purpose, requirement of client isolation in VPC

VPC sharing:
* multiple accounts ~~entity~~ of different business account, Aws organisation → billing, controlling access
is what services are

VPC sharing
All accounts should belong to same organisation
* One account will create VPC & share subnets of multiple accounts A & B & launch instances (ec2, lambda, ebs)
* owner is the one who owns it willing to share.
* participants can manage, configure, to efficiently use the subnet.
* participants can't delete the other subnets resources.

VPC peering:
* 2 vpc in same region can't speak to each other through private instance but can speak through Internet.
* To speak to each other through same switch
⇒ It is a Network connection b/n two VPC this allows the VPC to communicate eachother through same & update the Route table & ID is enough.
⇒ To communication to each other as if they are in same Network without leaving the Aws Networking. only possible through peering.
why ⇒ ~~only~~ 2 vpc in same region, different region.
⇒ peering b/n VPC's of 2 accounts.
⇒ both accounts can belong to any organization.
security ⇒ No two VPC should have same IP and range
* No transitive peering. A → B → C ✗
A → B ✓ directly to connect
A → C

Aws site-to-site VPN
private network
801Q protocols
VPN connection → secure.
* Customer device DC → capture the VPN
↳ Route table → through virtual private gateway.
↳ secure the services in Aws
private tunnel
↓
encrypted format data.

AWS Direct connect

- private dedicated fiber → data center to AWS service location
- encrypted

VPC endpoints:

↳ VPC connection to connect AWS resources/services to e

ex: S3 bucket, Amazon DynamoDB → Never past out VPC

- without going through the private network of it/self to connect to AWS resources/services.

Interface endpoint → powered by AWS private link.
Gatway endpoint → S3, dynamo DB.

AWS transit Gateway:

do it yourself ↳ Activity

VPC security: Statefull acts at subnet level

Security group: created multiple security groups. attach to instance
↳ restricted to instance level.

→ Based on security group rule the elastic Network Instance case allow @ reject the traffic

- Inbound, outbound.
  - Inbound → In default security all inbound rules are
  - Outbound → Destination
  - Source
- Only said what rules are allowed.
- Security groups are stateful → remembers
  eg: 22 port was one allowed again when sent it remembers identifier & allows the traffic through ssh. even if it is not mentioned in the traffic

Network access control lists (Network ACL's):

(NACL) 1st level

↳ acts at the subnet level.

- only one NACL attached to subnet.
- NACL will restrics the traffics of specified property.
- Lower the no of rule will be evaluated 1st
- heghur the no of rule will be evaluted last.
- Allow / Deny option can be mentioned
- NACL are stateless. → it doesn't remember.
  should specify which is allowed & not allowed.

Difference b/n VPC security & NACL

1. Amazon Route 53:

highly scalable & efficient

- Route 53 is a cloud based, domain Name system service that helps users to direct the traffic to the hosted on AWS resources.
- DNS resolver → this does dns lookup phonbook of dns
  8.8.8
  4.4.4

when user sends a particular domain name. → DNS resolver

this asks for dns service (Amazon Route 53) for IP address & returns the IP address to DNS to user resolver

& DNS resolver will keep a cache of searched result.

Domain name resolution in a Domain name system.

- Amazon Route 53 to route a traffic from a Internet outside the AWS also.
- we can configure the health of AWS resource.
- we can configure with the routing policy traffic.

1. Simple routing → for routing to 1 resource
2. weighted round robin routing → two resource running the same ami

progressing deliver → 2 versions → 75% to version 1 old.
25% to version 2 new

Amazon S3 Intellegent

- monetoring fee, intellegently moves to std & std-IA

Amazon S3 Glacier → archive & preservation, retrieval.

- to storing the data for very long period of time.
- retrieve the data quickly
- data which are not frequently used. healthcare finance
- can't upload directly

Lifecycle policy.

- After 30 days → move to std, 2 months implement, move to Gla 1y → automatically delete it
- Data in transition of data.

Database → long backups, long term storage, disaster recovery folder.

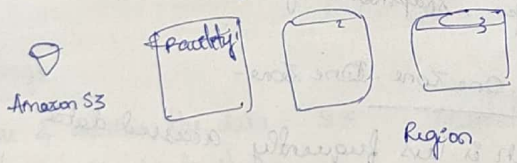
Amazon S3 parts

preservation

Amazon S3 bucket URL's

- bucket → name globally unique.
- DNS compliant.

Data is redundantly stored in the Region.



S3 → to store assets, apph running of any websites.
content delivery, big data analysis → lake house pipelines

Storing backups,

pricing → GB per month.

based on different request has different pricing.

- data transfer in - is free - all post request, upload.
- data transfer out - is paid - get operations.
- If data goes out of region → same region → not paid.

different store

- object storage
- large amount of data
- but latency, performance & throughput is not provided

block storage.

- large amount of data:

but latency, performance & throughput is given

Amazon Elastic file System (EFS)

- A storage that is accessible for nas & sap

ex:

C

Amazon EFS

Cloud Archite , Mod 5 (Amazon EFS)

Guided lab : Introducing Amazon Elastic File System

① EC2 & ⑤ Cloud watch
② EFS

mysql 3306
EFS 3049

EC2 → Instance
EFS client

Security
security groups → copy sg id

Network & Security
↳ Security group
Create security group

name → efsmount target sg
description → efs mount target sg
VPC → Lab Vpc

Inbound rule :

Type: NFS | Source type: Custom | sg: EFS-client ← paste sg id

create security group

① EFS → create file system
customise → default

① Name → myefs
Regional
uncheck ☐ Enable automatic backups ← uncheck
uncheck ☐ Enable encryption of data at rest
Enhanced
Elastic

Tag → Tag key: Name | Tag value: myefs

Next

② Network settings
Network private cloud (VPC)
VPC → Lab VPC
AZ's are automatically selected

Mount target → security group
uncheck → default
select
AZ's 1 us-east-1a
sg - efs mount target sg
Next
③ ④ leave plane → next → create
Refresh
us-east-1b
AZ's 2 → efs mount target sg
Next
Create

P.T.O for cont.

EFS → File System
myefs → View details → Attach
using NFS client
copy

Guided lab → AWS details
→ open instance server URL in new table

EC2 → Instances → EFS client → connect → error

Terminal → cd Downloads/
sudo rm -rf labsuser.pem
→ download key from lab
sudo chmod 600 labsuser.pem
sudo ssh -i labsuser.pem ec2-user@ public ip address 98.89.235.4

Session ID copy paste on browser

$ sudo mkdir /mnt/efs
sudo mount -t nfs4
df -h

// fs data/fs mounted /mnt/efs.

CFA

Cloud Arch : Mod 11 : Guided Lab : Automating Infrastructure deployment with AWS cloud Formation.

IAC - lets you to define your infrastructure & app

After provision, configuration after launching is config Managment
↳ updating, installing

* Terraform - provision, multicloud
Ansible, puppet, chef - config managment, open source tool
* OpsWork - provides chef & puppet version
* mandatory part on JSON & YAML files are resources — Stack
* users can give input during runtime - reconfigurable — list/pool of resources
output ← VPC id, subnet id, bucket id
↓
* deploy in layers { elb, web, database, network } take one output & provide as input.

parameter → makes reusable allowing user to put input of their choice

mapping : ami id is different for different regions. 32 bit os, 64 bit os
↳ maps the key to name value pair
System manager to map key —"—
↳ SSM

Condition → define when a resource is created or property defined

172.18.181.60:8000

⑨ Cloud Formation

Create Stack

- with new resource

⇒ Specify template

⊙ upload a template file

upload a template file

choose file

Next

lab Network

⇒

Stack name: lab-network

Next

⇒ Tags

Key: app

Value: cafe

Stack failure options

⊙ Roll back all stack resources

Next

Review & create

Subnet

Stacks

⊙ lab-network

Change sets → update & add rule ← sec rules ← sd

drift → some resource which are created where its config is changed

idem potency → other update template has same

diff b/n ver1 & ver2

of resource are created again others it doesn't touch the 1st resource.

Create stack

with

⇒ Specify template

⊙ upload a template file

choose file

Next

lab application

⇒ Stack name: lab-application

parameters

ami id

lab-network

⇒

- Port ↓
- config sets

cloud signal → cloud formation

httpd → apache

Deletion policy → take snapshot & delete it @ retain

web server sec. group → only allow inbound to port 80

URL output → enabled = True

⊙ lab-application → update stack

~~Create change~~

- Make a direct update

⊙ Replace existing template

⊙ upload a template file

parameters network

choose file

lab-application 2.yaml

Next

⑨ EC2

Instances → web server → storage

Security group → rd → edit inbound rules

Snapshots

↳ cloud formation → stack
→ lab-application → delete

In EC2 → snapshots → snap —— 100 GiB

cloud formation

→ Infrastructure composer

Menu

→ open → lab network → create template

confirm & continue to cloudformation

C.F → Mod 11 : Challenge (safe) lab: Automating Infrastructure Deployment.

⑨ cloud9

⑨ cloudformation

⑨ S3 bucket

⑨ code commit

⑨ code Pipeline

If getting error delete the stack

@ cloud9 Instance

open in cloud9

touch create-bucket.yaml

S3-bucket-cede-updated.yaml → change BucketName final-mis

copy paste save

aws configure get region // us-east-1

aws cloudformation validate-template --template-body file://create-bucket.yaml. aws-CLI

//parameters

aws cloudformation create-stack --stack-name mybucket --template-body file://create-bucket.yaml

// Stack id

⇒ enabled static website properties. S3 buckets are found

S3-bucket create-stack-create steps.txt CF → Stacks

S5 → General Purpose

1st cmd wget http://

unzip static website.zip & static

cd static/

ls → es images index.html

2nd cmd aws contols --bucket cgl-mis

- ownership

3rd cmd CF → Stack mybucket → output → URL ← click

4th cmd

code commit

Repositories → templates → clone URL → clone HTTPS

copy url

cmd

cd..

pwd

/home/ec2-user/environment

git clone _____

cd CF-templates

clear

git status

cd templates/

ls

CFoun → templates.yaml → duplicate → rename → cafe-network.yaml → open → remove all lines

Directories → cafe-network.yaml → copy → paste → save

→ aws cloudformation validate-template --template-body file://cafe-network.yaml

git add cafe-network.yaml

git commit -m "initial commit of network template" cafe-network.yaml

git push

codecommit repo

cafe-network.yaml

① CF template → network file will created

pipeline → cafenetwork pipeline → deploy

CF → stacks → Stack events

duplicate → templates2.yaml → rename →

directories → cafe-app.yaml

⇒ aws cloudformation validate-temp

file://cafe-app.yaml

code-pipeline (CD service) → create our pipeline

pipeline

Source code → actions → Fetch → save artifact

Build compile → Build test Artifact

test → Test Defined JUnit

Deploy → CloudFormation lambda ECS → use our change set of CF create a stack

Manual approve → manager approval

git add cafe-app.yaml

git commit -m 'updated' cafe-app.yaml

git push

CF → stacks → update-cafe-app → outputs → SDK

→ ip → Browser

- Static website (S3) - Arch m4

Static website hosting

1. Download zip file
extract them

14

papergrid
Date: / /

Cloud Architecture
Module 14
(optional) Guided lab:
Breaking Monolithic Node.js application
in Microservices

Aws details

Copy lab IDE url
Copy Lab IDE password
region us-east-1

On terminal
pwd

Copy curl command.

cd 1-ms-containers

npm install

npm start

new terminal

curl http://localhost:3000
curl http://localhost:3000/api/users
curl http://localhost:3000/api/users/1
curl http://localhost:3000/api/threads
curl http://localhost:3000/api/threads/1
curl http://localhost:3000/api/posts/1/threads/1
Curl http://localhost:3000/api/posts/by-user/1

go back to npm start wala terminal
ctr+c

cd ..
cd 2-containerized-monolith/
ls

docker built -t mb-repo .

Goto ECR
create repository
name : mb-repo
Create

select mb-repo
view push commands
Copy it and paste it in terminal
1. authentication
2. build
3. tag
4. push

go to ECR and refresh

go to ECS
Clusters
Create cluster
name : mb-cluster
Fargate and only managed resource

ECS instance type
t2. medium

Desired capacity
Minimum 2
Maximum 6

Network settings
IDE VPC

use existing security group
ECSSG

Create
task definition
Create new task definition
mb-taskdefinition

uncheck aws Fargate
check amazon EC2 instance

CPU Memory
.5 vCPU 1GB

Container-1
name
mb-container
Image URI
Copy from ECcontainer registry URI
container port 3000

CPU Memory hard limit Memory soft limit
.5 1 1

Create

papergrid
Date: / /

Delivery
Create service
name: mb-taskdefinition-service

Environment
Launch type
EC2

Networking
IDE VPC
Use existing security group
uncheck default
ECSSG
Load Balancing
Check use load balancing
create a new load balancer
name : mb-lb
Target group
Create new target group
name : mb-target
Create

Go to EC2
Load balancers
Copy DNS
paste it in browser / api/users

ECR ECR
Create repo
Mb-users / Mb-threads / Mb-posts
Create

• Static work:-

cd..

cd 3-containerized-microservices

cd users

copy 2, 3 & 4 from ECR Mb-users

view push commands.

cd..

cd threads

copy 2, 3 & 4 from ECR mb-threads

view push commands

cd..

cd posts

copy 2, 3 & 4 from ECR mb-posts

view push commands.

Go to EC2 → Load balancer

copy DNS

DNS/api/users

DNS/api/users/2

DNS/api/threads

/api/posts/create-thread/2

Static website hosting

1. Download zip file
extract them

15

Cloud Architecture
Module 11
Guided lab: Automating Infrastructure
with AWS Cofiguration

Cloud formation
Create stack
with new resources
choose an existing template
upload a template file
choose file
download lab-network.yaml
from aws page
next
stack name: lab-network
next
tags
Name: app value: cafe
submit

Create stack
with new resources
Choose an existing template
Choose file
download lab-application.yaml
from aws page
next
name: vlabs-application
next
Tag
Name: app value: cafe
submit

lab application
update stack
Make a direct update
Replace existing template
lab-application 2.yaml
next
next
submit

EC2
security group
you can see two inbound rules

snapshots
one snapshots are present

Cloud formation → stacks → lab application
click on delete
Snapshot is created because of deletion
policy

Cloud formation → Infrastructure composer
Menu → template file
choose any lab-network & lab application
Create template.

papergrid

Date: / /

Cloud Foundation
Module 10
Lab 6
Scale and load balance your Application
EC2
Instances
web service 1
Actions
image and template
Create image
imagename: myimage
image desc: EC2 AMI
Add tag:
Key: Name
Value: NewAMI
Create image

EC2
load balancer
Create load Balancer
Application Load Balancer
Create
name: NewLB

IPV4

Network mapping
VPC
Lab VPC

Availability zone
us-east-1a
Public subnet 1

us-east-1b
Public subnet 2

Security group
uncheck the default
add web security group

Listeners and routing

Routing actions
• forward to target groups
Create target group

Specify group details
Instances
Target group name: my target group

next

Create target group

Target group
my target group

Create load Balancer.

EC2
Auto scaling group
Instances
Launch templates
Create Launch templates
name: new template

Auto scaling

Auto Scalings group guidance
☑ Provide guidance to help me set up

Application and OS images
My AMI
Own by me
Amazon Machine image (AMI)
my image

Instance type
t2.micro

Key pair
Vockey

Networking setting
select existing group.
web security group SG

Advanced details

Detailed Cloud Watch monitoring
Enable

Create launch template

select the launch template
Action
Create auto scaling group

name : NEWASG
next

Network
VPC
Lab VPC
Availability zones and subnet
us-east-1a (Private subnet 1)
us-east-1b (Private subnet 2)
next
c
Attach to an existing load balancer
• choose from your load balancer
my target group | HTTP

Health check
✓ Turn on elastic load Balancing
health checks
next

Desired capacity
2
Min desried
2
Max desired
4

Auto scaling
Target tracking scaling policy

Instance warmup
60 seconds

Date: / /

Additional settings

☑ enable group metrics :

next

next

Add tag

Value Name

AG Name

next

Create Auto Scaling group

EC2

instances

All

Cloud watch

Alarm

total : 6

Cloud Architecture
Module 14
(optional) Guided Lab:
Breaking Monolithic Node.js application in Microservices

Access details

Copy Lab IDE url
Copy Lab IDE password
region us-east-1

In terminal
pwd

Copy curl command.

cd 1-no-containers

npm install

npm start

new terminal

curl http://localhost:3000
curl http://localhost:3000/api/users
curl http://localhost:3000/api/users/1
curl http://localhost:3000/api/threads
curl http://localhost:3000/api/threads/1
curl http://localhost:3000/api/posts/in-thread/1
curl http://localhost:3000/api/posts/by-user/1

go back to npm start wala terminal
ctr + C

cd ..
cd 2-containerized-monolith/
ls

docker build -t mb-repo .

Goto ECR
create repository
name: mb-repo
create

select mb-repo
view push commands
copy it and paste it in terminal
1. authentication
2. build
3. tag
4. push

go to ECR and refresh

go to ECS
Clusters
Create cluster
name: mb-cluster
Fargate and self managed service

EC2 instance type
t2. medium

Date: / /

Desired capacity

Minimum 2

Maximum 6

Network settings

IDE VPC

use existing security group

ECSSG

Create

task definition

Create new task defenition

mb-taskdefendion

uncheck aws Fargate

check amazon Ec2 instance

CPU	Memory
.5 vCPU	1GB

Containers-1

name

mb-containes

Image URI

Copy from ECcontaines registry URI

containes port 3000

CPU	Memory hard limit	Memory soft limit
.5	1	1

Create

Deploy

Create service

name: mb-taskdefinition-service

Environment

launch type

EC2

Networking

IDE VPC

use existing security group.

uncheck default

ECSSG

Load Balancing

Check use load balancing.

Create a new load balancer.

name: mb-lb

Target group

Create new target group

name: mb-target

Create

Go to EC2

Load balancer

Copy DNS

paste it in browser / api / users

ECR ECR

Create repo

Mb-users / Mb-threads / Mb-posts

Create

cd..

cd 3-containerized-microservices

cd users

copy 2, 3 & 4 from ECR mb-users

aview push commands.

cd..

cd threads

copy 2, 3 & 4 from ECR mb-threads

aview push commands

cd..

cd posts

copy 2, 3 & 4 from ECR mb-posts

aview push commands.

Go to EC2 → Load balancer

copy DNS

DNS/api/users

DNS/api/users/2

DNS/api/threads

/api/posts/in-thread/2

papergrid

Date: / /

Cloud Architecture

Module 11

Guided lab: Automating Infrastructure with AWS Configuration

Cloud formation

Create stack

with new resources

choose an existing template

upload a template file

Choose file

download lab-network.yaml

from aws page

next

stack name: lab-network

next

tags

Name: app value: Cafe

submit

Create stack

with new resources

Choose an existing template

Choose file

download lab-application.yaml

from aws page

next

name: lator-application

next

Tag

Name: app value: cafe

submit

lab application

update stack

Make a direct update

Replace existing template

lab-application 2.yaml

next

next

submit

EC2

security group

you can see two inbound rules

snapshots

one snapshots are present

Cloud formation → stack → lab application

click on delete

Snapshot is created because of deletion policy

Cloud formation → Infrastructure composer

Menu → template file

choose any lab-network & lab applica

Create template.

Cloud Architecture

Module II

Automating Infrastructure Deployment

Date: / /

9

Implementing serverless Architecture on AWS

Cloud Architecture

Module 14

Guided lab: Implementing serverless Architecture on AWS.

Event name: inventory load

Event type

object creation

[V] All object create events

Lambda

Create a function

Function name: load inventory lambda

Runtime: Python 3.9

Change default execution role

⊙ use an existing role

Existing role

lambda-load-Inventory-Role

Create function

Destination

Lambda function

Lambda function

load inventory lambda

Save changes

go download CSV file as

name .csv

paste contents

Code

Code source

copy code from 10. source code

paste it

deploy it

Upload it to the S3 bucket

Lambda

loadinventory lambda

monitor

view cloud watch logs

S3

Create bucket

name: myinventory

Create bucket

Log stream

click on it

my Inventory

properties

Event notification

Create inventory notification

DynamodB

Explore items

inventory

items are displayed.

Guided lab
Aws details
Dashboard link

Lambda
Create function
Function name: Store notification
Runtime: Python 3.9

use existing role
Lambda-check-stock-Role

Create function

Code
Code source
copy code from 49 source code
paste it
deploy it

SNS
Topics
Create topics ⊙ standard:
name: NoStock
display name
Create topic

Create subscription
Details
Protocol
Email

Endpoint
Rs.Prathinay@k.msisimple2025@

Create subscription

go to your mail
Confirm subscription

goto Lambda (Store notification wala)

Trigger
Dynamo DB

Dynamo table
inventory
Add.

New guided lab
cal.csv to
nasa cal.csv

S3 bucket
upload cal.csv

get an email.