**AWS Cloud Practitioners Essentials -**

1- Overview

2- What is cloud computing - Use and manage a scalable online data center

3 Scalability - resize resource’s as needed

4- Elasticity - Create scale and shut down new resource’s as needed

5 auto scale and elastic load balancing allow you to have this whole process automated

6 fault tolerance- the built in redundancy that helps you run even when you have errors

7 Your data is always owned by you - in AWS

8 Using AWS services-

**AWS Management Console - UI - Navigation and usability**

Aws services tools

AWS resource groups help cluster these services

Resource groups are user specific but can be shared

you can also tag the resource’s in groups for searching

It make time for your feature updates to take effect

AWS CLI - control from command line- flexibility to create scripts

make commands like to any remotely running EC2 instance

AWS SDK’s - Software development kits to use AWS with - the ability to use AWS in existing application’s

the SDK allows you to manage you infrastructure as code

All of these perform their actions by accessing AWS API's

**AWS EC2**

What is EC2 Elastic compute Cloud - amazon server instance tools pay as you go

How to build and configure a ec2 instance

login AWS

choose region

launch ec2 wiz

select AMI-

The OS of the server or image if you have one the server will use

select instance type

hardware selection - like t2 micro or 64 core hard drive

config network

how many instances running - 1 -to- 100000 max

setting for cloud and sub net - Note default is good here unless client or boss says otherwise

default auto assign settings which give us a DSC2 address- everything else usually good as well

Configure storage

set the gigs of storage and the type of volume

you can have multiple volumes configured at once

Make sure you set volumes to be deleted automatically if you choose at any point to terminate the instance

then add the TAG - name what you want

Security rules- SSH connectivity- or simple web coactivity

Save SSH key to computer

Last Review and launch your new EC2 instance

Configure key pairs

by creating a new key pair you get the info on how to connect to your services

Launch and connect

Hit launch to launch - once pending changes to running we can connect with the instance

highlight and copy public DNS -

launch a tool called Putty that you have downloaded on to your computer -

Host name = ec2-user@ ( paste in the DNS )

Then use Putty gen + where you saved your SSH key from before to make a ppk file

set this new ppk file as the SSH security key in putty and you should be able to log in with no problems

**AWS EBS Volumes**

EBS - Volumes to store your eC2 instances

pay as you go -

take snapshots of volumes at times for backup

full encryption options for data transit

EBS volumes can change the type need to hold all the EC2 instances

using EBS volumes

Volumes are located on the side bar when you are accessing EC2

Volumes must be in the same availability zones as their EC2 instances

Volume Config

General purpose SSD is best Volume type as it only changes for GB used

Snapshot ID section lets you restore a snapshot to a volume

after that just click create Volume

actions attach volume - and then specify instance attaching and device

from there you can go to your instance copy the SSH commands and run it in your terminal

on mac the command lsblk -will let you see all the instances you have running in this volume

by attaching and mounting a specific instance with root permissions you can begin cd’ing in and start accessing or creating files

You can also select and detach any volume at any time -

it will still be available once you have detached it

you can also tag your volumes

tags on AWS resources let you examine billing based on those tags

**AWS S3 -**

the data in S3 is not associated with ant specific server it’s all over the world

you can retrieve data any time anywhere - several terabytes in size

by default none of your data is shared publicly

you can also encrypt data in transit

we take an object like a video - and give it a key

backups are automatic when you make a bucket

S3 will scale data size and availability of requests

Access - you can access the data from the console the AWS CLI or AWS sdk's

you can also access your data through a rest service endpoint

this is an http request / (region specific endpoint)/ Object Key

S3 bucket names must be unique-

Use cases -

1 storing application assets - like server logs

2 for static web hosting s3 can serve up the content of your files

3 cross region replication makes it great a database backup tool

4 Staging area for Big data - you can use Red shift EMR amazon Athena or import export large data with snowball

5 It may be the most useful single tool in AWS

How to create a new S3 bucket

go to s3 section of AWS

hit create bucket

set name- must be DNS compliant

set region-make sure matches any EC2 instance that might be trying to access the data

and this is all we need to get started -

Interacting with the S3 bucket in UI

click on your target bucket

once you hit upload you can easily drag and drop files in the bucket

Interacting with the S3 bucket in CLI

target a file

use the (s3 copy command) to copy an object that lives in your s3 and give it a key to add it to that s3 instance

you can also sync a folder with the s3 bucket and the CLI will take those files and check to see if they exist in the bucket

if they don’t it will upload them - this is the best way to ensure no duplicate objects are in your s3 buckets

From here if we ssh into an EC2 instance we can see if it has been given IAM role to manage buckets what buckets its managing

using ls for macs onto that instance will allow you to see all the files in that bucket

from here you can use copy commands to bring the files back from s3 over to your main computer or ec2 instance

you can use the sync commands in reverse as well -

Back in the UI

when accessing a file in s3 you have options

you can modify properties and permissions you can also see attributes of the file

**AWS Global Infrastructure-**

Regions - locations hosting availability zones

these determine optimizing latency and reducing costs

you could deploy major resource’s in one location then deploy tools to interact with it in another

if one zone goes down others can pick up the request-

you should deploy your content across different locations whenever possible

edge locations are locations with quick access to high population areas

**Amazon VPC virtual private cloud**

this allows you to create a virtual private network within the aws cloud

vpc allows you complete control over the security and configurations - such as routing rules-

numerous services work directly with Amazon VPC

amazon ec2's are deployed into your amazon VPC-

amazon RDS or regional database service deploy directly into VPC -others work similarly to this

Amazon VPC's live within regions and can span across multiple AZ - availability zones

Subnets -

these allow the VPC to span multiple AZ's but fewer is recommended to make the service as simple as possible

Route tables control traffic between subnets and the internet- all sub nets have access to each other’s

subnets are either public or private - this means access to the internet or no access

Building a VPC

go to VPC section of AWS

select region

select name of VPC

set the IP address

next build a subnet - name it

specify subnet locations

continue to create all subnets-

add in an internet gateway to the subnet's you want to be public

name the gateway

**AWS Security Groups**

act as built in fire walls

control access to instances

you can have different security groups with different levels of access

security group will be on the left of your service

clicking on it will allow you to adjust security group settings on your security groups or create new ones

Creating a security group- Name - description - Rules

default - inbound traffic in denied all outbound traffic is allowed -you can edit this

The best practice is to figure out what traffic is required for your service and only allow it

you can also set - type - protocol - port rage - Source - Description

by setting sources to all 0 you allow all ip address’s

by adding multiple rules you can designate between things like http and https requests security settings

**AWS Application Load Balancer**

replaces the classic load balancer with new features

Supported Protocols- Cloud watch metrics - Access Logs - Health Checks

additional features like path and host based routing and native ipv6 support included

Application load balancer allows you route different requests to the same ec2 instance or other service target

you set up rules to target what you want

ALB -terms

Listeners - listen for requests

target -is the target of the requests

Target Groups- hold multiple targets for requests

Creating an Application load balancer

go to EC2 in aws services

go to a running instance

copy the ip address of the instance

if they are sites you can check to see if they are listening correctly with the ip address plus domain name

next go to load balancers on the left side

click create load balancer

you can then choose application load balancer or the classic load balancer

set name

set scheme- internet facing

set ip address

set listeners- set them to the ports you want to listen for

than choose the availability zone

then set a key

then configure security settings

then configure security group

then configure routing - for target groups

next we can set the heath check settings - every 10secs is good

next we register the targets

then add to register

then we get to the review page - where we see everything we have just configured

after that we can just hit create

with two targets you must create a target group

click create target group

set name

set request type

set health check

use view and edit rules to change things like forwarding address targets

next we can check to see if it is working by copying the DNS name and checking the http request in the browser

to adjust the listeners go to the listeners tab

**AWS Auto Scaling**

checks the available EC2 instances to manage your workload

cloud watch checks values but only Auto Scaling can add more to your network

you can set the conditions for auto scaling for the days you need more performance versus less

scaling out is adding more scaling in is adding less

1 create launch configuration

2 create an auto scaling group - min/max instances

3 policy- do you scale on a time schedule or based on use

4 dynamic auto scaling can be set to go off on cloud watch alarms

5 you should have alarms to both add and remove instances

Let’s Build a demo

open EC2

choose auto scaling groups

choose ec2 conditions

go to launch instance conditions

choose instance type

give launch condition a name

choose an existing key pair

choose properties for auto scaling group

specify instances

specify subnet deployment

set configure scaling policy’s to adjust capacity of group

set min max instance values

set target tracking policy - like cpu usage 60%

hit crate auto scaling group

go to the instance tab you will see pending your min instances already being generated

if you go back to the auto scaling group you can modify these min max instances based on launch config

**Amazon Route 53**

DNS service or Domain name service to route end users to endpoints

how does it work - user puts in address redirects to route 53 sends back info on a target address they need to go

you an create hosted zones to help route 53 send your users back to the right address

Let’s create a DNS service

go to services on the left choose hosted zones

hit create hosted zone

set the domain name

set type - like public hosted zone

create a new record set - choose sub domain

choose routing policy

then hit create

Now you can see your instance resolving names to your desired instance target address

Route 53 supports a wide variety of DNS resolution options -

works with ipv4 and ipv6 instances

**AWS RDS- (Relational database service)**

running your own relational database has many factors - making in cost time and money

AWS provides a service that automatically handles all the major admin tasks of your db.

with RDS you can just worry about your data

scaling data backup and maintenance are all covered by amazon

what is a RD instance- a individual db. that you can manage scale and control all from aws services

you can run it using the VPC service-

you can run it on its own -

when you configure it you can choose the availability zone used

when you create an amazon RDS a standby back up copy is created as well all interactions are synced to the standby copy

its recommended to run your RDS in multiple AZ's

Amazon RDS supports the creation of read replicas - for my SQL - Maria DB and others

using read replicas you can also scale out beyond the capacity of a single database instance

read replicas can be promoted to be the master db. instance

amazon RDS is perfect for most web application’s, standalone db's, e commerce sites as well as online games

the service can be managed with the AWS UI, CLI, or any API

**AWS Lambda**

aws lambda is a compute service that lets you run code without provisioning or managing services

scales and mages you code

no servers to manage - Continuous Scaling - Sub second Metering

aws lambda runs your code on a highly available compete infrastructure - this provides all administration

capacity provisioning and auto scaling are also handled

also aws lambda supports a variety of programing langue’s like python and C#

best use is to run code in response to events

you can respond to s3 changes or even http requests

you can also build server-less application’s that respond to aws lambda functions

auto deploy with aws code pipeline and aws code deploy

To use -

1 Configure Lambda environment -

2 Upload code

3 Watch it run

config Lambda environment

In the lambda section you can see all lambda functions you have created

choose entry type- Runtime language - and the code handler

set environment vars - for encryption - set tags - choose an execution role- set memory execution types

then set the trigger policy such as cloud watch to trigger your code

next the monitoring page will show you when your function has run

you can also use aws lambda combined aws kinesis to process real time steaming data - and much much more

you can also build aws lambda to handle your extract transform and load pipelines

you can also use aws lambda to build the backend for your IOT devices

you can also use to run the entire backend of your site when combined with other aws services

AWS lambda is thought of as the connective tissue between all other aws services

**AWS Elastic Beanstalk**

AWS beanstalk is a platform as a service

so you can place your code over to launch your application quickly

choose your instance type choose your data base choose your auto scaling levels

update your application Access server logs Enable HTTPS load balancer

EB supports a large range of platforms as well as code languages

this is the bean stalk stack

1 your code-

2 Application service

3 HTTP service- Operating system

4 Language interpreter

5 Host

create application - upload version - Launch Environment - Mange Environment

Update your application as easily as you deploy it

Use case - launch python website

go to elastic beanstalk in the dashboard

create a new application- and then name it - give it a description

choose the environment

set up your environment configuration

than set up your environment’s base config including - tier - platform an application code

advanced config can let you change this like software instances and load balancing

then hit create- and the log will show you all the pieces of your application being generated

after your done you have a dashboard that shows you your whole application hosting

even the url for your site if that is what you were creating