AWS 'Architecting on AWS' training course DAY ONE (24/05/2023 – 26/05/2023)

Course objectives:

1. Identify AWS services, compare features, and explore best practices to architect resilient, secure, and highly available IT solutions on AWS

Course layout:

1. https://online.vitalsource.com/reader/books/200-ARCHIT-74-EN-SG-E/pageid/0 (Lecture slides link - Theory)
2. https://us-east-1.student.classrooms.aws.training/class/oa6PnQtm5WkBM7Gv3XdVaA (Labs link - Practical labs)
3. https://explore.skillbuilder.aws/learn/course/external/view/elearning/8319/architecting-on-aws-online-course-supplement (Additional learning resources - optional)

Content:

For details – pls refer to the actual lecture notes/slides

This summary below is just a short & summarized version of the main points in the course.

1. Module 1 (Architecting Fundamentals)
    a. Benefits of using AWS – pay-as-you-go, provides a lot of various computing resources, performance, secure, reliable, optimize costs
    b. AWS Infrastructure:
        i. AWS has multiple **Data Centers** that host thousands of servers/network equipment →
        ii. which are organized in many different **Availability Zones** →
        iii. and are located in many **Regions** around the globe →
        iv. There are also many AWS **Local Zones** (Think of it as a 'Minimized version' of an AWS data center that are usually available in selected regions. Think of it as you can spin up Local Zones to deploy AWS resource in countries that are NOT in the usual AZ/regions… this is done to improve latency cause sometimes its better for latency wise to spin up a LZ that's nearer to your customers rather than a AZ/region that's far from your customers) →
        v. and **Edge Locations** (cache data & deliver content faster to your customers, helps to improve latency as well aka pushing your content to locations closer to your customers – e.g. CDN like CloudFront, services like Route53)
    c. Factors that impact region selection in AWS (aka which region should you select?):
        i. Governance/Regulatory laws
        ii. Latency
        iii. Cost
        iv. Service Availability
    d. AWS Well-Architected Framework:
        i. Security, Performance Efficiency, Cost optimization, Operational excellence, Reliability, Sustainability
    e. Lab 1 – Learn how to use AWS Management Console (GUI) and the CLI to create a S3 bucket and upload an object to it. Commands of interest:
        i. aws s3 ls (list all the available S3 instances you created)
        ii. aws s3 mb s3://labclibucket-NUMBER (make a S3 bucket with that name)

iii. aws s3 cp /home/ssm-user/HappyFace.jpg s3://labclibucket-NUMBER (copy a file named HappyFace.jpg to your bucket)
iv. aws s3 ls s3://labclibucket-NUMBER (list all your objects stored in your bucket)

2. Module 2 (Account Security)
    a. AWS Root user:
        i. Has full access to ALL AWS services
        ii. SHOULD NOT be used for day-to-day interactions with AWS
        iii. CANNOT be restricted in a single account model
    b. AWS IAM users:
        i. Create different IAM (Identity and Access Management) groups
        ii. Under each group, can have different individual users that will interact daily with your AWS services
        iii. E.g. [Dev IAM group] [Marketing IAM group] [IT Support IAM group]
        iv. Under each group, can have different individual users with varying access to your AWS services.
    c. Ways to access AWS services:
        i. AWS Management Console - GUI
        ii. Programmatic access – CLI or SDKs:
            1. Access key ID and Access key values – these 2 things are used when programmatically accessing AWS services using CLIs or SDKs.
            2. Follow Least Privileges Principle – give access key id, values to ONLY those who need it
            3. DO NOT check in all these access key id & values to public repos/documentation!
    d. IAM Roles
        i. Think of IAM Roles as like "temporary" access to certain AWS services only
        ii. We can easily assign IAM roles and revoke those roles when those users no longer need access to those AWS services
    e. Principals:
        i. A principal is someone that can make a request for an action, or operation or AWS resource.
        ii. 4 types of principals: IAM user, IAM role, AWS service (e.g. SDK/CLIs), Identity Provider (idP) or federated user (verify the user's identity through your own company's active users directory or via 3rd party services like Fb, Google etc).
    f. Differences between IAM identity-based policies vs IAM resource-based policy:
        i. Identity-based policies are attached to a particular identity in AWS. It dictates which AWS resources that identity can access. WHAT YOU CAN INVOKE?
        ii. Resource-based policies are attached to a particular AWS resource instead. It dictates which identity/who can access that resource. WHO CAN INVOKE THE RESOURCE?
        iii. Reference: https://sonalake.com/latest/identity-vs-resource-based-aws-iam-policies/
    g. AWS policies are IMPLICIT DENY by default… we can configure to explicitly deny or explicitly allow also
    h. Identity based policies + Resource based policies make your AWS Architecture DEFENSIVE and SECURE!

3. Module 3 (Networking 1):
    a. An IP Address identifies a location within a network
    b. Examples of IP addresses – IPv4 (32 bits → 2 to the power of 32 combinations) and IPv6 (128 bits → 2 to the power of 128 combinations)
    c. E.g. if IPv4: **172.31**.2.15 (the **bold** part identifies the **network**, the underlined part identifies the location of the host)
    d. CIDR (Classless Inter-Domain Routing):
        i. CIDR notation defines the range of IP addresses for a network
        ii. E.g. 10.22.0.0/28 → IPv4 has 2 power of 32 combinations → if CIDR is /28 → means 32-28 = 4 bits left → means 2 to power of 4 = 16 possible Ips
        iii. Refer to slide 120 in lecture notes for more info
    e. VPC fundamentals:
        i. VPC (Virtual Private Cloud) – this is your network environment in the cloud (think of it like a container/box that encompasses all the components below)
        ii. Subnets – a subset of your VPC CIDR block; for simplicity think of it as a range of IP addresses within your VPC (subset of VPC). Can be public subnet or private subnet. It's basically an IP address within the possible combinations of IP addresses in your VPC. In short, a subnet is a sub-range of IP addresses within a network.
        iii. Internet gateway (allows communication between resources in your VPC and the internet – part of public subnet. An Internet GW perform NAT by mapping a public and private addresses → translates a source IP)
        iv. Route table (A set of rules that the VPC uses to route the network; route tables basically control where network traffic is directed; can have public route table or private route table – attached to public or private subnets respectively)
        v. Elastic IP address (a fixed IP address you can purchase)
        vi. Elastic network interface
        vii. NAT gateway (Network Address Translation gateway – this is needed so that your AWS resources in your private subnet can connect to external services outside your VPC; NAT gateway has to be within a public subnet; It essentially maps/converts the private IP addresses of your AWS resources in the private subnet to a public IP address so that resources outside VPC can be accessed – the key is to PROTECT your private IP addresses – In summary: NAT gateway allows internet traffic initiated by private subnet instances)
        viii. Good explanation regarding the differences between Internet GW vs NAT GW:
            1. Internet Gateway (IGW) allows instances with public IPs to access the internet AND the internet to access those instances. 2 way traffic.
            2. NAT Gateway (NGW) allows instances with no public IPs (only private IPs) to access the internet. 1 way traffic.
            3. https://medium.com/awesome-cloud/aws-vpc-difference-between-internet-gateway-and-nat-gateway-c9177e710af6
            4. https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Internet_Gateway.html (Cltrl F 'Connect to the internet using an internet gateway' + 'IP addresses and NAT' – see SS below)

IP addresses and NAT

To enable communication over the internet for IPv4, your instance must have a public IPv4 address. You can either configure your VPC to automatically assign public IPv4 addresses to your instances, or you can assign Elastic IP addresses to your instances. Your instance is only aware of the private (internal) IP address space defined within the VPC and subnet. The internet gateway logically provides the one-to-one NAT on behalf of your instance, so that when traffic leaves your VPC subnet and goes to the internet, the reply address field is set to the public IPv4 address or Elastic IP address of your instance, and not its private IP address. Conversely, traffic that's destined for the public IPv4 address or Elastic IP address of your instance has its destination address translated into the instance's private IPv4 address before the traffic is delivered to the VPC.

5.

f. A network access control list (ACLs) acts as a firewall at the subnet boundary. By default, it allows all inbound and outbound traffic.
g. A security group (SG) is a virtual firewall at the instance level. It also controls inbound and outbound traffic:
    i. 1) [Web Server aka the server that hosts your FE logic] ➔ can have a Web Security Group (e.g. Inbound rule – allow HTTPS port 443, Source: 0.0.0.0/0 (any)
    ii. 2) [App Server aka the server that hosts your BE logic] ➔ can have a App Security Group (e.g. Inbound rule – allow HTTP port 80, Source: Web tier – only Web Server can access)
    iii. 3) [Database] ➔ Can have a Database Security Group (e.g. Inbound rule – allow TCP port 3306, Source: App tier – only App Server can access)
h. Network ACL (**STATELESS** FIREWALL) vs Security Group (**STATEFUL** FIREWALL)

4. Module 4 (Compute)
    a. Various computing resources available in AWS – e.g. EC2, Serverless, ECS/EKS containers, AWS Fargate, AWS Inferentia, AWS Graviton processors
    b. AWS EBS (Elastic Block Storage):
        i. Two main types: SSD and HDD
        ii. SSD – better performance (faster read/writes) VS HDD – slower performance, but more cost friendly (good for large & infrequently accessed data)
    c. Each EC2 instance has their ephemeral block storage instance (basically each EC2 instance has their own temporary block storage – after the instance is shut down, that storage is cleared, so cannot use this as a solution for your storage needs).
    d. AWS Lambda (Serverless compute option):
        i. AWS handles the server management for you fully (that's why is "serverless" as per say)
        ii. You only focus on the code/application logic
        iii. Convenient, and operates on a pay-per-run
    e. Architecting on AWS - Lab 2 - Build your Amazon VPC infrastructure:
        i. This lab teaches how to set up a VPC AWS architecture
        ii. Search for 'VPC' in AWS Management Console and create a VPC from there
        iii. At the 'VPC' AWS service, the left navigation bar/panel will contain all the necessary links that you need to click to configure your whole AWS Architecture on the cloud
        iv. In summary:
            1. Go to 'Your VPCs' >
            2. 'Subnet' (create public + private subnet) >

3. ‘Internet gateway’ (need to attach IGW to VPC first, then add/configure the Public Route Table of the Public Subnet to use the IGW) >
4. ‘Route Tables’ (Need to configure your route table to allow access to route traffic from the Internet – by adding 0.0.0.0 route to IGW destination, AND THEN go to ‘Subnet Associations’ tab, and attach to your public subnet to make it REALLY PUBLIC) >
5. ‘Security Groups’ (Create a SG for your EC2 instance in the public subnet) > Go to ‘EC2 Dashboard’ (time to create a EC2 instance, and attach it to our VPC/public subnet and also attach that SG to it) >
6. Go to ‘Networking’ tab in the instance dashboard (Copy your IPv4 public DNS domain and copy paste in a new browser tab to test if the HTTP connection works) >
7. Connect to that public EC2 instance through ‘Connect’ > ‘Session Manager’ > in the bash terminal type in ` curl -I https://aws.amazon.com/training/` (essentially just making a request in the command line to get some response data) >
8. Go back to ‘VPC’ > ‘NAT Gateways’ (left panel/menu) > Create a NAT Gateway >
9. ‘Route tables’ > Create a private route table for your private subnet > Add in the route 0.0.0.0 to allow traffic out of your private subnet to the NAT gateway > Go to ‘Subnet Associations’ > And associate your private subnet to this particular private route table >
10. Time to create a SG for your private subnet > Go to ‘Security Groups’ on left menu > ‘Create Security Group’ for your private subnet (make sure you configure the correct ‘Inbound rules’ → in this lab, we ONLY ALLOW inbound access to your private SG from public SG) >
11. Launch an EC2 instance in private subnet + attach it to VPC/private subnet + attach the private SG >
12. Search for EC2 in searchbar > Go to EC2 dashboard > Launch a new EC2 instance (MAKE SURE you set it up properly)
13. Go to ‘Connect’ to connect to your private EC2 instance > use ‘Session Manager’ to access your private EC2 instance’s bash terminal/CLI > Do ‘cd ~’ to change the working directory and use the command ‘curl -I https://aws.amazon.com/training/’ to get a response back > if get a response > means OK, you successfully connected to a private instance using Session Manager
14. Optional tasks: Copy your private EC2 instance’s private IPv4 address > Connect to your public EC2 instance via Session Manager > do ‘ping <PRIVATE_INSTANCE_IPV4_ADDRESS>’ > Notice that the ping request to your private instance FAILS > We have to figure out the **incorrect inbound rule in the private SG** (notice only got packets transmitted but none is received – that’s why fail)
15. Optional tasks: Go to ‘VPC’ > ‘Security Groups’ > select that Private SG > Add a new inbound rule ‘Custom ICMP – IpV4’, source: ‘Public SG’ > ‘Save Rules’. You should now see the ping command succeed. The reason is because **the ping command is a type of ICMP traffic hence we needed to add that ICMP rule for our private SG then it can work**