AWS Cloud Patricioner

Created	@December 17, 2022 10:44 AM
Class	
Type	
Reviewed	☐
Materials	

IAM Section

▼ **Summary**

- Users: mapped to a physical user, has a password for AWS Console
- Groups: contains users only
- Policies: json document that outlines permissions for users or groups
- Roles: for EC@ instances or AWS services
- Security: MFA + password policy
- AWS CLI: manage your AWS services using the command-line
- AWS-SDK: manage your AWS services using a programming language
- Access Keys: access AWS using the CLI or SDK
- Audit: IAM Credentials Reports & IAM Access Advisor

▼ **AWS CLI**

- AWS — version → shows to us the version of CLI
- AWS iam list-users → shows to us the lis of users

▼ **CoudShell**

- ls → list files on environment
- echo `name` > `name.extension` -> Create a new file on environment
- We can download this files
- cat → show what is in our file
- pwd → full path to my file
    - Than I will copy this path to start my download

▼ **IAM Roles for Service**

- Some AWS service will need to perform actions on your behalf
- To do so, we will assign permissions to AWS Services with IAM Roles
- Common Roles:
    - EC2 Instance Roles
    - Lambda Function Roles
    - Roles for CloudFormation

▼ **IAM Security Tools**

> IAM Credentials Report (account-level)

- a report that list all your account’s users and the status of their various credentials

> IAM Access Advisor (user-level)

Access advisor shows the service permissions granted to a user and when those services were last accessed.

You can use this information to revise your policies.

▼ IAM Guidelines & Best Practices

- Don’t use the root account except for AWS account setup
- One physical user = One AWS user
- Assign users to groups and assign permissions to groups
- Create a strong password policy
- Use and enforce the use of MFA
- Create and use Roles for giving permissions to AWS services
- Use Access Keys for Programmatic Access (CLI/SDK)
- Audit permissions of your account with the IAM Credentials **Report**
- **Never share IAM users & Access Keys**

▼ Shared Responsibility Model for IAM

> AWS

- Infrastructure (global network security)
- Configuration and vulnerability analysis
- Compliance validation

> YOU

- Users, Groups, Policies management and monitoring
- Enable MFA on all accounts
- Rotate all your keys often

- Use IAM tools to apply appropriate permissions
- Analyze access partners & review permissions

EC2 (Elastic Compute Cloud)

▼ **Section - Summary**

- EC2 Instance: AMI (OS) + Instance Size (CPU + RAM) + Storage + security groups + EC2 User Data
- Security Groups: Firewall attached to the EC2 instance
- EC2 User Data: Script launched at the first of an instance
- SSH: start a terminal into our EC2 Instances (port 22)
- EC2 Instance Role: link to IAM roles
- Purchasing Options: On-Demand, Spot, Reserved (Standard + Convertible + Scheduled), Dedicated Host, Dedicated Instance

▼ **Amazon EC2**

- It mainly consists in the capability of:
  - Renting Virtual machines (EC2)
  - Storing data on virtual drives (EBS)
  - Distributing load across machines (ELB)
  - Scaling the services using an auto-scaling group (ASG)

▼ **EC2 sizing & configuration options**

- Operating System (OS): Linux, Windows or Mac OS
- How much compute power & cores (CPU)

- How much RAM
- How much storage space:
  - Network-attached (EBS & EFS)
  - hardware (EC2 instance store)
  - By default, the root storage is terminated when you finish an instance.
  - You attach other volumes as you want in anytime. However, you need to create an storage in the same AZ - It’s possible to attach other storage volumes with diferentes AZ, but it’s out of scope from this course.
- Network card: speed of the card, public IP address
- Firewall rules: security group
- Bootstrap script (configure at first launch): EC2 User Data

▼ EC2 User Data

It is possible to bootstrap our instances using an EC2 User data script.

- bootstrapping means launching commands when a machine starts
- That script is only run once at the instance first start
- EC2 user data is used to automate boot tasks such as:
  - Installing updates
  - Installing software
  - Downloading common files from the internet
  - Anything you can think of
- The EC2 User Data Script runs with the root user

▼ EC2 Instance Types – Compute Optimized

• Great for compute-intensive tasks that require high performance
processors:
• Batch processing workloads
• Media transcoding
• High performance web servers
• High performance computing (HPC)
• Scientific modeling & machine learning
• Dedicated gaming servers

▼ **EC2 Instance Types – Memory Optimized**

• Fast performance for workloads that process large data sets in memory
• Use cases:
• High performance, relational/non-relational databases
• Distributed web scale cache stores
• In-memory databases optimized for BI (business intelligence)
• Applications performing real-time processing of big unstructured data

▼ **EC2 Instance Types – Storage Optimized**

• Great for storage-intensive tasks that require high, sequential read and write
access to large data sets on local storage
• Use cases:
• High frequency online transaction processing (OLTP) systems
• Relational & NoSQL databases
• Cache for in-memory databases (for example, Redis)
• Data warehousing applications
• Distributed file systems

▼ **Security Groups - Good to know**

- Can be attached to multiple instances
- Locked down to a region / VPC combination
- Does live “outside” the EC2 – if traffic is blocked the EC2 instance won’t see it
- It’s good to maintain one separate security group for SSH access
- If your application is not accessible (time out), then it’s a security group issue
- If your application gives a “connection refused“ error, then it’s an application error or it’s not launched
- All inbound traffic is blocked by default
- All outbound traffic is authorized by default

▼ **SSH for Windows**

- We need de ‘.pem’ file to allow our connection
- On PowerShell, go to our directory file and paste the command line
  - ssh -i .\course-key-pair.pem ec2-user@’the-instance’s-public-ip’
- To logout, execute the command exit

▼ **EC2 Instances Purchasing Options**

- On-Demand Instances – short workload, predictable pricing, pay by second
- Reserved (1 & 3 years)
  - Reserved Instances – long workloads

    - Convertible Reserved Instances – long workloads with flexible instances
- Savings Plans (1 & 3 years) –commitment to an amount of usage, long workload
- Spot Instances – short workloads, cheap, can lose instances (less reliable)
- Dedicated Hosts – book an entire physical server, control instance placement
- Dedicated Instances – no other customers will share your hardware
- Capacity Reservations – reserve capacity in a specific AZ for any duration

▼ **Shared Responsibility Model for EC2**

AWS

- Infrastructure (global network security)
- Isolation on physical hosts
- Replacing faulty hardware
- Compliance validation

USER

- Security Groups rules
- Operating-system patches and updates
- Software and utilities installed on the EC2 instance
- IAM Roles assigned to EC2 & IAM user access management
- Data security on your instance

EC2 Instance Storage Section

- **It’s a network drive (i.e. not a physical drive)**
  - It uses the network to communicate the instance, which means there might be a bit of latency
  - It can be detached from an EC2 instance and attached to another one quickly
- **It’s locked to an Availability Zone (AZ)**
  - An EBS Volume in us-east-1a cannot be attached to us-east-1b
  - To move a volume across, you first need to snapshot it
- **Have a provisioned capacity (size in GBs, and IOPS)**
  - You get billed for all the provisioned capacity
  - You can increase the capacity of the drive over time
- **Controls the EBS behavior when an EC2 instance terminates**
  - By default, the root EBS volume is deleted (attribute enabled)
  - By default, any other attached EBS volume is not deleted (attribute disabled)
  - This can be controlled by the AWS console / AWS CLI
  - Use case: preserve root volume when instance is terminated
- **EBS Snapshots**
  - Make a backup (snapshot) of your EBS volume at a point in time
  - Not necessary to detach volume to do snapshot, but recommended
  - Can copy snapshots across AZ or Region
- **EBS Snapshots Features**

- **EBS Snapshot Archive**
  - Move a Snapshot to an ”archive tier” that is 75% cheaper
  - Takes within 24 to 72 hours for restoring the archive
- **Recycle Bin for EBS Snapshots**
  - Setup rules to retain deleted snapshots so you can recover them after an accidental deletion
  - Specify retention (from 1 day to 1 year)

AMI

▼ **Overview**

- AMI = Amazon Machine Image
- AMI are a **customization** of an EC2 instance
  - You add your own software, configuration, operating system, monitoring…
  - Faster boot / configuration time because all your software is pre-packaged
- AMI are built for a **specific region** (and can be copied across regions)
- You can launch EC2 instances from:
  - **A Public AMI**: AWS provided
  - **Your own AMI**: you make and maintain them yourself
  - **An AWS Marketplace AMI**: an AMI someone else made (and potentially sells)

▼ **AMI Process (from an EC2 instance)**

- Start an EC2 instance and customize it

- Stop the instance (for data integrity)
- Build an AMI – this will also create EBS snapshots
- Launch instances from other AMIs

▼ **EC2 Image Builder**

- Used to automate the creation of Virtual Machines or container images
- => Automate the creation, maintain, validate and test EC2 AMIs
- Can be run on a schedule (weekly, whenever packages are updated, etc…)
- Free service (only pay for the underlying resources)

▼ **EC2 Instance Store**

- EBS volumes are network drives with good but “limited” performance
- **If you need a high-performance hardware disk, use EC2 Instance Store**
- Better I/O performance
- EC2 Instance Store lose their storage if they’re stopped (ephemeral)
- Good for buffer / cache / scratch data / temporary content
- Risk of data loss if hardware fails
- Backups and Replication are your responsibility

▼ **EFS – Elastic File System**

. Managed NFS (network file system) that can be mounted on 100s of EC2
• EFS works with Linux EC2 instances in multi-AZ
• Highly available, scalable, expensive (3x gp2), pay per use, no capacity planning

▼ **EFS Infrequent Access (EFS-IA)**

. Storage class that is cost-optimized for files not accessed every day
• Up to 92% lower cost compared to EFS Standard
• EFS will automatically move your files to EFS-IA based on the last time they were accessed
• Enable EFS-IA with a Lifecycle Policy
• Example: move files that are not accessed for 60 days to EFS-IA
• Transparent to the applications accessing EFS

▼ **Shared Responsibility Model for EC2 Storage**

AWS

. Infrastructure
• Replication for data for EBS volumes & EFS drives
• Replacing faulty hardware
• Ensuring their employees cannot access your data

ME

. Setting up backup / snapshot procedures
• Setting up data encryption
• Responsibility of any data on the drives
• Understanding the risk of using EC2 Instance Store

▼ **Amazon FSx – Overview**

. Launch 3rd party high-performance file systems on AWS
• Fully managed service

▼ **Amazon FSx for Windows File Server**

. A fully managed, highly reliable, and scalable Windows native shared file system

• Built on Windows File Server
• Supports SMB protocol & Windows NTFS
• Integrated with Microsoft Active Directory
• Can be accessed from AWS or your on-premise infrastructure

▼ **Amazon FSx for Lustre**

. A fully managed, high-performance, scalable file storage for High Performance Computing (HPC)
• The name Lustre is derived from “Linux” and “cluster”
• Machine Learning, Analytics, Video Processing, Financial Modeling, …
• Scales up to 100s GB/s, millions of IOPS, sub-ms latencies

▼ **EC2 Instance Storage - Summary**

- **EBS volumes:**
  - • network drives attached to one EC2 instance at a time
  - • Mapped to an Availability Zones
  - • Can use EBS Snapshots for backups / transferring EBS volumes across AZ
- **AMI**: create ready-to-use EC2 instances with our customizations
- **EC2 Image Builder:** automatically build, test and distribute AMIs
- **EC2 Instance Store**:
  - High performance hardware disk attached to our EC2 instance
  - Lost if our instance is stopped / terminated
- **EFS**: network file system, can be attached to 100s of instances in a region
- **EFS-IA**: cost-optimized storage class for infrequent accessed files

- **FSx for Windows:** Network File System for Windows servers
- **FSx for Lustre**: High Performance Computing Linux file system

Elastic Load Balancing & Auto Scaling Groups Section

▼ **Scalability & High Availability**

- Scalability means that an application / system can handle greater loads by adapting.
- There are two kinds of scalability:
  - Vertical Scalability
  - Horizontal Scalability (= elasticity)
- **Scalability is linked but different to High Availability**

▼ **Vertical Scalability**

- Vertical Scalability means increasing the size of the instance
- For example, your application runs on a t2.micro
- Scaling that application vertically means running it on a t2.large
- Vertical scalability is very common for non distributed systems, such as a database.
- There’s usually a limit to how much you can vertically scale (hardware limit)

▼ **Horizontal Scalability**

- Horizontal Scalability means increasing the number of instances / systems for your application

- Horizontal scaling implies distributed systems.
- This is very common for web applications / modern applications
- It’s easy to horizontally scale thanks the cloud offerings such as Amazon EC2

▼ **High Availability**

- High Availability usually goes hand in hand with horizontal calling
- High availability means running your application / system in at least 2 Availability Zones
- The goal of high availability is to survive a data center loss (disaster)

▼ **High Availability & Scalability For EC2**

- Vertical Scaling: Increase instance size (= scale up / down)
  - From: t2.nano - 0.5G of RAM, 1 vCPU
  - To: u-12tb1.metal – 12.3 TB of RAM, 448 vCPUs
- Horizontal Scaling: Increase number of instances (= scale out / in)
  - Auto Scaling Group
  - Load Balancer
- High Availability: Run instances for the same application across multi AZ
  - Auto Scaling Group multi AZ
  - Load Balancer multi AZ

▼ **Scalability vs Elasticity (vs Agility)**

- Scalability: ability to accommodate a larger load by making the hardware stronger (scale up), or by adding nodes (scale out)
- Elaticity: once a system is scalable, elasticity means that there will be some “auto-scaling” so that the

system can scale based on the load. This is “cloud-friendly”: pay-per-use, match demand, optimize costs

- Agility: (not related to scalability - distractor) new IT resources are only a click away, which means that you reduce the time to make those resources available to your developers from weeks to just minutes.

▼ **What is load balancing?**

- Load balancers are servers that forward internet traffic to multiple servers (EC2 Instances) downstream

▼ **Why use a load balancer?**

- Spread load across multiple downstream instances
- Expose a single point of access (DNS) to your application
- Seamlessly handle failures of downstream instances
- Do regular health checks to your instances
- Provide SSL termination (HTTPS) for your websites
- High availability across zone

▼ **Why use an Elastic Load Balancer?**

An ELB (Elastic Load Balancer) is a managed load balancer
• AWS guarantees that it will be working
• AWS takes care of upgrades, maintenance, high availability
• AWS provides only a few configuration knobs
• It costs less to setup your own load balancer but it will be a lot more effort on your end (maintenance, integrations)
• 4 kinds of load balancers offered by AWS:
• Application Load Balancer (HTTP / HTTPS only) – Layer 7
• Network Load Balancer (ultra-high performance, allows for TCP) – Layer 4
• Gateway Load Balancer – Layer 3
• Classic Load Balancer (retired in 2023) – Layer 4 & 7

▼ **What’s an Auto Scaling Group?**

- In real-life, the load on your websites and application can change
- In the cloud, you can create and get rid of servers very quickly
- The goal of an Auto Scaling Group (ASG) is to:
  - Scale out (add EC2 instances) to match an increased load
  - Scale in (remove EC2 instances) to match a decreased load
  - Ensure we have a minimum and a maximum number of machines running
  - Automatically register new instances to a load balancer
  - Replace unhealthy instances
- Cost Savings: only run at an optimal capacity (principle of the cloud)

▼ **Auto Scaling Groups – Scaling Strategies**

- Manual Scaling: Update the size of an ASG manually
- Dynamic Scaling: Respond to changing demand
  - Simple / Step Scaling
    - When a CloudWatch alarm is triggered (example CPU > 70%), then add 2 units
    - When a CloudWatch alarm is triggered (example CPU < 30%), then remove 1
  - Target Tracking Scaling
    - Example: I want the average ASG CPU to stay at around 40%
  - Scheduled Scaling
    - Anticipate a scaling based on known usage patterns
    - Example: increase the min. capacity to 10 at 5 pm on Fridays

- Predictive Scaling
  - Uses Machine Learning to predict future traffic ahead of time
  - Automatically provisions the right number of EC2 instances in advance
  - Useful when your load has predictable time based pattern

▼ **ELB & ASG – Summary**

- **High Availability** vs **Scalability** (vertical and horizontal) vs **Elasticity** vs **Agility** in the Cloud
- **Elastic Load Balancers (ELB)**
  - Distribute traffic across backend EC2 instances, can be Multi-AZ
  - Supports health checks
  - 3 types: Application LB (HTTP – L7), Network LB (TCP – L4), Classic LB (old)
- Auto Scaling Groups (ASG)
  - Implement Elasticity for your application, across multiple AZ
  - Scale EC2 instances based on the demand on your system, replace unhealthy
  - Integrated with the ELB