TNGS Learning Solutions
AWS Solutions Architect
Online Course
Elastic Compute (EC2)
Compute Service
Fundamentals

keyPair

- In AWS, a **key pair** refers to a set of cryptographic keys used for securely accessing and managing EC2 instances (virtual machines).
- AWS uses key pairs as part of its security model to ensure that only authorized individuals or systems can access and control AWS resources.

keyPair

- **Public Key:** This part of the key pair is intended to be shared openly. It is used to encrypt data or verify a digital signature.
- **Private Key:** The private key is kept secret and should never be shared. It is used to decrypt data or create digital signatures.
- AWS stores the public key while the private key remains with the user.

keyPair

- **SSH Access:** AWS key pairs are used to securely access EC2 instances using SSH (Secure Shell). When you create an EC2 instance, you associate an AWS key pair with it. To access the instance, you must have the private key corresponding to the public key stored on the instance.
- **Authentication:** Key pairs can be used to authenticate users and applications when interacting with AWS services programmatically, such as using the AWS Command Line Interface (CLI) or AWS SDKs.
- **Data Encryption:** Key pairs can also be used to encrypt and decrypt data in storage or during transmission.

Creating an AWS Key Pair

- To create an AWS key pair, you can either use the AWS Management Console or the AWS CLI.
- Here's an example of how to create a key pair using the AWS CLI:

```
aws ec2 create-key-pair --key-name MyKeyPair
```

- AWS will generate a key pair, store the public key, and provide the private key to you as a downloadable file.
- Be sure to secure the private key since it grants access to resources.

Instance Type

- In AWS, an **instance type** refers to a predefined configuration for a virtual server (EC2).
- Instance types define the combination of **computing**, **memory, storage, and network capacity** (CPU, RAM) available to the virtual machine.
- AWS offers a wide range of instance types to cater to various workload requirements, from small web applications to high-performance computing clusters.

Categories of Instance Types

- AWS categorizes its instance types into families based on their intended use cases and characteristics.
  - General Purpose
  - Compute Optimized
  - Memory Optimized
  - Storage Optimized
  - GPU Instances
  - FPGA Instances.

Categories of Instance Types

- Each instance family includes different instance sizes or types, typically identified by a name that includes the family name and a size indicator (e.g., t2.micro, m5.large, c4.xlarge).
- Larger sizes within a family generally offer more resources (vCPU, RAM) and, consequently, higher performance and cost.

Instance Types

- Each instance type has specific characteristics, including the number of virtual CPUs (vCPUs), memory (RAM), and network performance.
- Some instance types also include features like GPU (Graphics Processing Unit) for parallel processing or local instance storage for temporary data.

Instance Types Use Cases

- Different instance types are optimized for specific use cases. For example:
  - General Purpose instances are suitable for a wide range of applications, including web servers and small to medium-sized databases.
  - Compute Optimized instances are designed for CPU-intensive workloads, like data analysis and scientific computing.
  - Memory Optimized instances are tailored for memory-intensive applications, such as in-memory databases and data analytics.
  - GPU instances are ideal for tasks like machine learning, deep learning, and rendering that require high-performance graphics processing.
  - Storage Optimized instances are optimized for high-capacity, low-latency storage requirements.

Instance Types Pricing

- AWS instance types come with different pricing options, including On-Demand, Reserved Instances (RIs), Spot Instances, and Savings Plans.
- Pricing depends on factors like the instance type, region, and payment model.
- Click on the link below for instance type pricing models

https://aws.amazon.com/ec2/pricing/on-demand/

Elastic IP

- An **Elastic IP (EIP)** is a public IPv4 address provided by AWS that can be associated with an Amazon EC2 instance or certain other AWS resources.
- EIPs are designed to make it easy to host applications and services that require a consistent public IP address, even when the underlying infrastructure changes.
- Elastic IPs are public IPv4 addresses, which means they can be used to communicate over the internet.

Dynamic and Static IP Addresses:

- Unlike standard EC2 instances that receive dynamic public IP addresses that change when the instance is stopped or terminated, EIPs are static.
- Once allocated, an EIP remains associated with your AWS account until you release it.

Elastic IP Use Cases

- Elastic IPs are commonly used for scenarios where you need a fixed and publicly routable IP address for your AWS resources, such as:
  - Hosting a website or web application with a consistent IP address.
  - Running email servers or DNS servers.
  - Implementing network appliances like firewalls and load balancers.
  - Maintaining IP reputation for outgoing emails or network communication.

Elastic IP Billing and Costs

- AWS charges a small hourly fee for each allocated Elastic IP address that is not associated with a running instance. However, if an EIP is attached to a running instance, there is no additional cost beyond the standard EC2 instance pricing.

Security Groups

- A **Security Group** is a fundamental component of the network security model that acts as a virtual firewall for controlling inbound and outbound traffic to AWS resources, primarily Amazon EC2 instances (virtual machines).
- Security Groups are used to define rules that specify the allowed traffic sources, destination ports, and protocols for a group of resources.
- Each EC2 instance can be associated with one or more Security Groups.

Security Groups

- Security Group rules can be stateful, meaning if you allow inbound traffic from a specific IP address, the corresponding outbound response traffic is automatically allowed. You do not need to create separate outbound rules.
- Security Groups are stateless for outbound traffic, so you need to define rules for outbound traffic separately if needed.

Security Groups

- When you launch an EC2 instance, it is automatically associated with a default Security Group.
- By default, all inbound traffic to an instance is denied, and all outbound traffic is allowed.
- You can modify the default Security Group rules to suit your requirements.
- You can equally create a custom security group for your instance.
- You can associate one or more security group to an instance and likely associate same security group to multiple instances

Security Groups

- Security Group rules are evaluated based on the principle of "least privilege." If a rule explicitly allows traffic, it is permitted. Otherwise, traffic is denied by default.
- When multiple Security Groups are associated with an instance, the rules are effectively combined, and the most permissive rule is applied.
- Security Group rule updates take effect immediately, and changes to rules do not require instance restarts or downtime.

Security Groups Use Cases

- Security Groups are used for various security-related tasks, such as:
  - Restricting SSH (Secure Shell) access to a specific set of trusted IP addresses.
  - Allowing only specific IP addresses to access web applications running on EC2 instances.
  - Defining network access control policies for databases, web servers, and other resources.

Network ACLs vs. Security Groups

- While Security Groups control traffic at the instance level, Network Access Control Lists (Network ACLs) operate at the subnet level. Network ACLs provide an additional layer of control for traffic between subnets within a Virtual Private Cloud (VPC).

Elastic block storage (EBS)

- Amazon Elastic Block Store (Amazon EBS) is a block-level type storage that is designed for use with Amazon EC2 instances.
- EBS allows you to create and attach persistent block storage volumes to your EC2 instances, providing scalable and durable storage for your data and applications.
- Amazon EBS provides block-level storage volumes, which are similar to hard drives or SSDs.
- These volumes can be attached to EC2 instances to store data and files.

Types of EBS Volumes

- Amazon EBS offers several types of storage volumes, each optimized for different use cases:
  - **General Purpose (SSD):** Provides a balance of price and performance and is suitable for a wide range of workloads.
  - **Provisioned IOPS (SSD):** Designed for I/O-intensive applications that require high and consistent input/output operations per second (IOPS).
  - **Throughput Optimized (HDD):** Optimized for high-throughput workloads that require sequential read and write access to data.
  - **Cold HDD:** Ideal for infrequently accessed data that needs cost-effective storage.
  - **Magnetic (HDD):** Provides the lowest cost per gigabyte and is suitable for low-latency, infrequent access workloads.

EBS Volumes

- EBS volumes can be easily created, attached, and resized as needed. You can scale your storage up or down without affecting the EC2 instance.
- Amazon EBS volumes are designed for high durability and availability. They are replicated within a specific Availability Zone (AZ) to ensure data redundancy.
- EBS volumes can be snapshotted, creating a point-in-time copy of the data. Snapshots are stored in Amazon S3 and can be used for data backup, recovery, and cloning.
- EBS volumes support encryption at rest using AWS Key Management Service (KMS). You can encrypt both new and existing volumes for added security.

EBS Volumes

- EBS volumes are AZ-specific, so they can only be attached to EC2 instances in the same AZ.
- Amazon EBS plays a critical role in providing scalable and reliable block storage for EC2 instances in AWS.

EBS Snapshots

- Amazon Elastic Block Store (EBS) Snapshots are point-in-time copies of EBS volumes in AWS.
- These snapshots capture the entire state of an EBS volume, including its data, configuration, and metadata, at the moment the snapshot is created.
- EBS snapshots are a crucial tool for data backup, recovery, and data management in AWS.

EBS Snapshots

- EBS snapshots serve as a reliable means of backing up your EBS volumes. You can create snapshots of your volumes to protect your data against data loss, accidental deletions, or volume failures.
- EBS snapshots are incremental, meaning that only the data that has changed since the last snapshot is saved. This efficiency reduces storage costs and speeds up the snapshot creation process.

EBS Snapshots

- EBS snapshots are regional resources, meaning they are specific to an AWS region. You can copy snapshots to other regions for redundancy or disaster recovery purposes.
- EBS snapshots can be encrypted using AWS Key Management Service (KMS) keys. Encrypted snapshots provide an additional layer of security for your data.

EBS Snapshots

- You can define snapshot lifecycle policies using AWS Data Lifecycle Manager to automate the creation, retention, and deletion of snapshots based on defined schedules and policies.
- You can use EBS snapshots to create new EBS volumes. These volumes can be used for various purposes, including disaster recovery, testing, or scaling your application.

EBS Snapshots

- You can share EBS snapshots with other AWS accounts, making them useful for collaboration and sharing data across accounts.
- EBS snapshots have associated storage costs based on the volume's size and the duration of retention. It's important to manage and delete unnecessary snapshots to control costs.

Bootstrapping a Server

- Bootstrapping a server refers to the process of setting up and configuring a server automatically, often from a minimal or bare-bones state, to prepare it for its intended purpose.
- This process typically involves installing necessary software, configuring settings, and ensuring that the server is ready to perform its designated tasks.
- Bootstrapping can be particularly useful in cloud computing environments, where servers can be provisioned and configured on-demand.

Bootstrapping a Server

- **Provisioning the Server**: This involves creating a new server instance, whether it's a virtual machine in a cloud environment (e.g., AWS EC2, Azure VM) or a physical server.
- **Operating System Installation**: Install the chosen operating system onto the server if it's not already pre-installed. This may include selecting the appropriate version and configuring basic settings such as the hostname.
- **Security**: Immediately secure the server by applying necessary updates, patches, and security configurations. This step is crucial to protect the server from vulnerabilities.

Bootstrapping a Server

- **User Access and SSH Keys**: Set up user accounts, including at least one with administrative privileges. Configure SSH key-based authentication for secure remote access.
- **Networking Configuration**: Configure network settings, including IP addresses, DNS servers, and firewall rules, to ensure proper network communication.
- **Software Installation**: Install the necessary software and dependencies for your server's intended role. This can include web servers (e.g., Apache, Nginx), databases (e.g., MySQL, PostgreSQL), application servers (e.g., Tomcat, Node.js), and any custom software.

Bootstrapping a Server

- **Configuration Management**: Use configuration management tools like Ansible, Puppet, or Chef to automate the configuration of software and services. This ensures consistency and repeatability across multiple servers.
- **Application Deployment**: If your server runs applications, scripts, or services, automate their deployment and configuration as part of the bootstrapping process. This may involve deploying code from version control repositories.
- **Logging and Monitoring**: Configure logging and monitoring tools to track server performance and detect issues promptly. Tools like Prometheus, Grafana, or AWS CloudWatch can help with this.

Bootstrapping a Server

- **Backup and Recovery**: Implement backup solutions to ensure data can be restored in case of data loss or server failure. EBS snapshots (as mentioned in the previous response) are often used for data backup in AWS environments.
- **Testing**: Thoroughly test the server to ensure that it's functioning as expected. Automated tests can help catch issues early in the bootstrapping process.
- **Documentation**: Document the server's configuration, including any customizations and settings, for future reference. This documentation is crucial for troubleshooting and maintenance.

Bootstrapping a Server

- **Scaling and Load Balancing**: If necessary, set up auto-scaling and load balancing to handle increased traffic and demand. Services like AWS Auto Scaling and Elastic Load Balancing can help with this.
- **Automation Scripts**: Consider using automation scripts or configuration management tools to automate the entire bootstrapping process. Infrastructure as Code (IAC) practices can make server provisioning and management more efficient and reproducible.
- **Monitoring and Maintenance**: Continuously monitor the server's performance and security and perform routine maintenance tasks, such as applying updates and patches.

Bootstrapping a Server

- Remember that the specific steps and tools you use for bootstrapping may vary depending on your server's purpose, the chosen cloud provider, and your organization's preferences.
- The goal is to streamline the server setup process, reduce manual interventions, and ensure consistency and reliability across your server infrastructure.