Dated

25.03.2024

IntelliPaat Assignment

Module 5: Case Study - 1

IntelliPaat

AWS Solutions Architect Training

Problem Statement:

You work for XYZ Corporation and based on the expansion requirements of your

corporation you have been asked to create and set up a distinct Amazon VPC for

the production and development team. You are expected to perform the following

tasks for the respective VPCs.

**Production Network:**

1. Design and build a 4-tier architecture.

2. Create 5 subnets out of which 4 should be private named app1, app2,

dbcache and db and one should be public, named web.

3. Launch instances in all subnets and name them as per the subnet that

they have been launched in.

4. Allow dbcache instance and app1 subnet to send internet requests.

5. Manage security groups and NACLs.

**Development Network:**

1. Design and build 2-tier architecture with two subnets named web and db

and launch instances in both subnets and name them as per the subnet

names.

2. Make sure only the web subnet can send internet requests.

3. Create peering connection between production network and development

network.

4. Setup connection between db subnets of both production network and

development network respectively.

Solution

**AWS Solutions Architect Training: VPC Setup for XYZ Corporation**

This document outlines the steps to create and configure separate Amazon VPCs for XYZ Corporation's production and development teams, adhering to the provided requirements.

**I. Production Network**

**1. Design: 4-Tier Architecture**

The production network will employ a 4-tier architecture consisting of:

* **Web Tier (Public)**: Handles user requests and interacts with the application tier.
* **Application Tier (Private)**: Processes user requests and interacts with the database tier. Can be further divided into multiple subnets (app1, app2) for horizontal scaling.
* **Database Cache Tier (Private)**: Caches frequently accessed database data to improve performance. (Optional: Can be combined with app tiers)
* **Database Tier (Private)**: Stores application data.

**2. Subnets:**

Create five subnets within the production VPC:

* **web (Public)**: For web servers accessible from the internet.
* **app1 (Private)**: For application tier instances (can be replicated for app2).
* **dbcache (Private)**: For database cache instances (optional).
* **db (Private)**: For database server instances.

**3. Launching Instances:**

Launch EC2 instances in each subnet and name them according to their subnet (web, app1, dbcache, db).

**4. Internet Access:**

* **dbcache subnet**: Allow outbound internet traffic for potential software updates or license management.
* **app1 subnet**: Allow outbound internet traffic only if the application tier requires access to external resources (e.g., APIs).

**5. Security Groups and Network Access Control Lists (NACLs):**

* Implement security groups to control inbound and outbound traffic for each subnet.
* Web tier security group: Allow inbound traffic from the internet on port 80 (HTTP) or 443 (HTTPS). Restrict all other inbound traffic.
* Application tier security group: Allow inbound traffic only from the web tier security group (e.g., port 80/443) and dbcache/db security group (e.g., database port). Restrict all other inbound traffic.
* dbcache/db security group: Allow inbound traffic only from the application tier security group on the designated database port. Restrict all other inbound traffic.
* Configure NACLs at the VPC level to restrict all inbound traffic by default and allow specific ports only based on security group rules.

**II. Development Network**

**1. Design: 2-Tier Architecture**

The development network will utilize a simpler 2-tier architecture:

* **Web Tier (Public)**: Serves development web application.
* **Database Tier (Private)**: Stores development database data.

**2. Subnets and Instances:**

Create two subnets:

* **web (Public)**: For the development web server.
* **db (Private)**: For the development database server.

Launch instances in each subnet and name them accordingly (web, db).

**3. Internet Access:**

* **web subnet**: Allow outbound internet traffic for development purposes (e.g., software updates).
* **db subnet**: Restrict outbound internet traffic.

**4. Peering Connection and Database Connectivity:**

* Establish a VPC peering connection between the production and development VPCs.
* Configure security groups on both sides to allow communication between the development web tier and the production database tier (if necessary).
* **Important**: Implement security measures to restrict access to production databases from the development environment.

**Note:** This is a high-level overview. For detailed configuration steps, refer to the official AWS documentation for VPC, subnets, security groups, NACLs, VPC peering, and instance launching.

Pls Share the architecture diagram

**VPC Architecture Diagram for XYZ Corporation**

Here's a basic illustration of the VPC architecture for XYZ Corporation:

**Production VPC**

Internet

|

+-----+-----+

| WEB | |

+-----+-----+ (Public Subnet)

|

v

+-----+-----+-----+-----+

| APP1 | APP2 | DBCACHE | DB |

+-----+-----+-----+-----+

| |

v v

+---------+ +---------+

| Dev WEB | (VPC Peering) | Prod DB |

+---------+ +---------+

| |

v v

+---------+ +---------+

| Dev DB | | ... |

+---------+ +---------+

(Development VPC)

**Explanation:**

* The diagram represents two separate VPCs, one for Production and one for Development.
* The Production VPC has a public subnet (WEB) accessible from the internet.
* The Production VPC has private subnets for application tiers (APP1, APP2), database cache (DBCACHE), and database (DB).
* The Development VPC has public and private subnets (WEB, DB) for a simpler 2-tier architecture.
* A VPC peering connection allows communication between the Development WEB subnet and the Production DB subnet (if necessary).
* **Note:** This diagram depicts basic connectivity. Security groups would be implemented to restrict access as per the requirement (e.g., development shouldn't have full access to production databases).

**Additional Notes:**

* This is a simplified diagram, and the actual implementation may involve additional components like security groups, route tables, and internet gateways.
* NACLs (Network Access Control Lists) can be configured at the VPC level to add another layer of security.

CHAT GPT

To design and build a 4-tier architecture in AWS cloud with the specified requirements, we'll use Virtual Private Cloud (VPC), subnets, instances, security groups, and Network Access Control Lists (NACLs). Here's a step-by-step guide to accomplish this:

**1. Design the Architecture:**

* **Tier 1 (Public Tier - Web):**
  + Contains instances accessible from the internet.
* **Tier 2 (Private Tier - App1 & App2):**
  + Contains application servers.
* **Tier 3 (Private Tier - DBCache):**
  + Contains database caching servers.
* **Tier 4 (Private Tier - DB):**
  + Contains database servers.

**2. Create Subnets:**

* **Public Subnet:**
  + Name: web
* **Private Subnets:**
  + Name: app1
  + Name: app2
  + Name: dbcache
  + Name: db

**3. Launch Instances:**

* **Public Subnet (web):**
  + Launch web servers or any other publicly accessible instances.
* **Private Subnet (app1):**
  + Launch application servers.
* **Private Subnet (app2):**
  + Launch application servers.
* **Private Subnet (dbcache):**
  + Launch database caching servers.
* **Private Subnet (db):**
  + Launch database servers.

**4. Allow Internet Requests:**

* **DBCache Instance:**
  + Create a route in its subnet's route table pointing to an internet gateway (IGW).
* **App1 Subnet:**
  + Modify its route table to allow internet access (via an IGW).
* **Configure Security Groups and NACLs:**

**5. Manage Security Groups and NACLs:**

* **Security Groups:**
  + Create security groups for each tier.
  + Allow necessary traffic within each tier (e.g., between app servers and database servers).
  + Allow HTTP/HTTPS traffic to the web servers from the internet (in the web tier).
* **NACLs:**
  + Create network ACLs for each subnet.
  + Define rules to control inbound and outbound traffic at the subnet level.
  + Ensure proper communication between subnets while restricting unnecessary traffic.

**Configuration Example:**

* **Public Subnet (web):**
  + Security Group: Allow HTTP/HTTPS from 0.0.0.0/0
* **Private Subnets (app1, app2, dbcache, db):**
  + Security Groups: Allow traffic within the tier.
  + Network ACLs: Allow necessary traffic between subnets and restrict unwanted traffic.

**Additional Considerations:**

* **Route Tables:**
  + Ensure each subnet is associated with the appropriate route table.
  + Configure routes for internet access where required.
* **Internet Gateway (IGW):**
  + Attach the IGW to the VPC.
* **Elastic IPs (EIPs):**
  + Assign EIPs to instances in the public subnet if needed.

By following these steps and configurations, you'll have a 4-tier architecture deployed in AWS cloud with the specified requirements. Adjustments may be needed based on specific application requirements and security considerations.

pls share the 4 tier design

+-------------------+

| Internet |

| Gateway |

+--------+----------+

|

|

V

+--------------+

| Public |

| Subnet |

| (web) |

+--------------+

|

|

+------------------|------------------+

| | |

V V V

+----------+ +-----------+ +-----------+

| Private | | Private | | Private |

| Subnet | | Subnet | | Subnet |

| (app1) | | (app2) | | (dbcache) |

+----------+ +-----------+ +-----------+

| | |

| | |

V V V

+----------+ +-----------+ +-----------+

| Private | | Private | | Private |

| Subnet | | Subnet | | Subnet |

| (db) | | (db) | | (db) |

+----------+ +-----------+ +-----------+