Creating a Virtual Private Cloud

Lab overview

Traditional networking is difficult. It involves equipment, cabling, complex configurations, and specialist skills. Amazon Virtual Private Cloud (Amazon VPC) hides the complexity and simplifies the deployment of secure private networks.

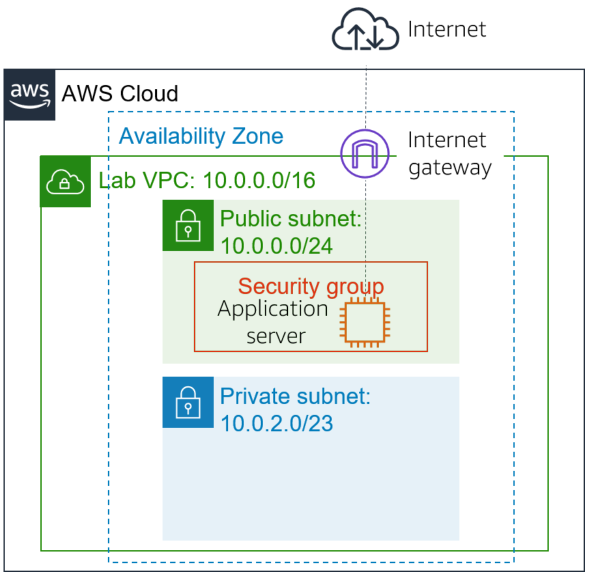
This lab shows you how to build your own virtual private cloud (VPC), deploy resources, and create private peering connections between VPCs.

Objectives

After completing this lab, you should be able to:

- Deploy a VPC
- Create an internet gateway and attach it to the VPC
- Create a public subnet
- Create a private subnet
- Create an application server to test the VPC

At the end of this lab, your architecture will look like the following example:



Task 1: Creating a VPC

You begin by using Amazon VPC to create a new VPC.

A VPC is a virtual network that is dedicated to your Amazon Web Services (AWS) account. It is logically isolated from other virtual networks in the AWS Cloud. You can launch AWS resources, such as Amazon Elastic Compute Cloud (Amazon EC2) instances, into the VPC. You can configure the VPC by modifying its IP address range and can create subnets. You can also configure route tables, network gateways, and security settings.

5. In the AWS Management Console, on the **Services** menu, choose **VPC**.

   The VPC console provides a wizard that can automatically create several VPC architectures. However, in this lab, you create the VPC components manually.

6. In the left navigation pane, choose **Your VPCs**.

   A default VPC is provided so that you can launch resources as soon as you start using AWS. There is also a shared VPC that you use later in the lab. However, you now create your own **Lab VPC**.

   The VPC will have a Classless Inter-Domain Routing (CIDR) range of **10.0.0.0/16**, which includes all IP address that start with 10.0.x.x. It contains more than 65,000 addresses. You later divide the addresses into separate subnets.

7. Choose **Create VPC**.

8. Under **Resources to create**, choose **VPC only**.

9. Configure the following settings:

   - For **Name tag** enter `Lab VPC`
   - For **IPv4 CIDR block**, enter `10.0.0.0/16`

- For **Tenancy**, select **Default**.
- For Tags, ensure that: **Key:** `Name` **Value:** `Lab VPC`

10. Choose **Create VPC**.
11. From the **VPC Details** page, choose the **Tags** tab.

Tags are useful for identifying resources. For example, you can use a tag to identify cost centers or different environments (such as development, test, or production).

12. Choose **Actions** and select **Edit VPC settings**.
13. In the **DNS settings** section, select **Enable DNS hostnames**.

This option assigns a friendly Domain Name System (DNS) name to EC2 instances in the VPC, such as the following:

ec2-52-42-133-255.us-west-2.compute.amazonaws.com

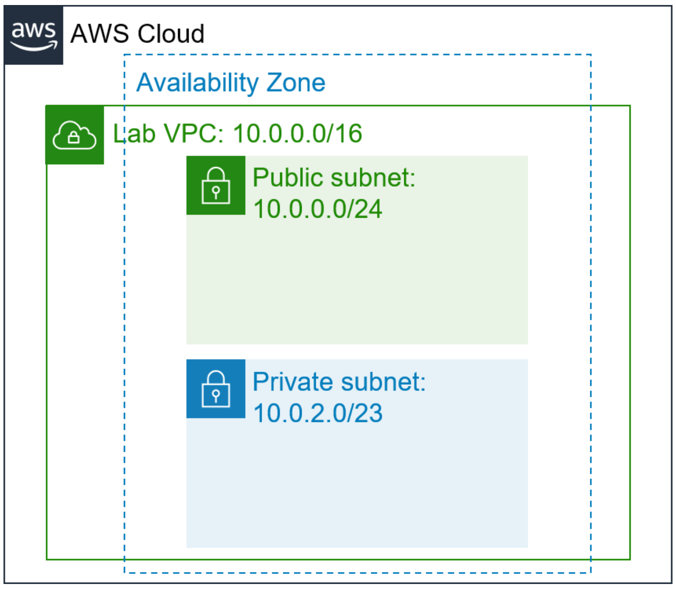
14. Choose **Save**.

Any EC2 instances that are launched into the VPC now automatically receive a DNS hostname. You can also add a more-meaningful DNS name (such as **app.example.com**) later by using Amazon Route 53.

Task 2: Creating subnets

A subnet is a subrange of IP addresses in the VPC. AWS resources can be launched into a specified subnet. Use a *public subnet* for resources that must be connected to the internet, and use a *private subnet* for resources that must remain isolated from the internet.

In this task, you create a public subnet and a private subnet:



Create a public subnet

You use the public subnet for internet-facing resources.

15. In the left navigation pane, choose **Subnets**.
16. Choose **Create subnet** and configure the following settings:
    - For **VPC ID**, choose **Lab VPC**.
    - For **Subnet name**, enter `Public Subnet`
    - For **Availability zone**, select the first Availability Zone in the list. Do not choose **No Preference**.
    - For **IPv4 CIDR block**, enter `10.0.0.0/24`
    - Choose **Create subnet**

The VPC has a CIDR block of **10.0.0.0/16**, which includes all 10.0.x.x IP addresses. The subnet you just created has a CIDR block of **10.0.0.0/24**, which includes all 10.0.0.x IP addresses. They might look similar, but the subnet is smaller than the VPC because of the **/24** in the CIDR range.

You now configure the subnet to automatically assign a public IP address for all instances that are launched in it.

17. Select the check box for **Public Subnet**.

18. Choose **Actions** and select **Edit subnet settings**. Then configure the following option:
    - Under **Auto-assign IP settings**, select **Enable auto-assign public IPv4 address**.
    - Choose **Save**

Though this subnet is named **Public Subnet**, it is not yet public. A public subnet must have an internet gateway, which you attach in the next task.

Create a private subnet

You use the private subnet for resources that must remain isolated from the internet.

19. Use what you learned in the previous steps to create another subnet with the following settings:
    - For **VPC ID**, choose **Lab VPC**.
    - For **Subnet name,** enter `Private Subnet`
    - For **Availability Zone**, select the first Availability Zone in the list. Do not choose **No Preference**.
    - For **IPv4 CIDR block**, enter `10.0.2.0/23`
    - Choose **Create subnet**

The CIDR block of **10.0.2.0/23** includes all IP addresses that start with 10.0.2.x and 10.0.3.x. This is twice as large as the public subnet because most resources should be kept private unless they specifically must be accessible from the internet.

Your VPC now has two subnets. However, the public subnet is totally isolated and cannot communicate with resources outside the VPC. Next, you configure the public subnet to connect to the internet via an internet gateway.

Task 3: Creating an internet gateway

An *internet gateway* is a horizontally scaled, redundant, and highly available VPC component. It allows communication between the instances in a VPC and the internet. It imposes no availability risks or bandwidth constraints on network traffic.

An internet gateway serves two purposes:

- To provide a target in route tables that connects to the internet
- To perform network address translation (NAT) for instances that were assigned public IPv4 addresses

In this task, you create an internet gateway so that internet traffic can access the public subnet.

20. In the left navigation pane, choose **Internet Gateways**.
21. Choose **Create internet gateway** and configure the following settings:
    - For **Name tag**, enter `Lab IGW`
    - Choose **Create internet gateway**

You can now attach the internet gateway to your **Lab VPC**.

22. Choose **Actions** and then **Attach to VPC**, and configure the following settings:
    - For **Available VPCs**, select **Lab VPC**.
    - Choose **Attach internet gateway**

This action attaches the internet gateway to your **Lab VPC**. Although you created an internet gateway and attached it to your VPC, you must also configure the public subnet route table so that it uses the internet gateway.

Task 4: Configuring route tables

23. In the left navigation pane, choose **Route Tables**.

    Several route tables are displayed, but there is only one route table associated with **Lab VPC**. This route table routes traffic locally, so it is a private route table.

24. In the **VPC** column, find the route table that shows **Lab VPC**, and select the check box for this route table. (You can expand the column to see the names.)
25. In the **Name** column, choose and then enter the name `Private Route Table` and choose **Save**
26. In the lower half of the page, choose the **Routes** tab.

There is only one route. It shows that all traffic that is destined for 10.0.0.0/16 (which is the range of the **Lab VPC**) will be routed locally. This route allows all subnets in a VPC to communicate with each other.

You now create a new public route table to send public traffic to the internet gateway.

27. Choose **Create route table** and configure the following settings:
    - For **Name**, enter `Public Route Table`
    - For **VPC**, choose **Lab VPC**.
    - Choose **Create route table**
28. In the **Routes** tab, choose **Edit routes**

    You now add a route to direct internet-bound traffic (0.0.0.0/0) to the internet gateway.
29. Choose **Add route** and then configure the following settings:
    - For **Destination**, enter `0.0.0.0/0`
    - For **Target**, select **Internet Gateway**, and then from the dropdown list select **Lab IGW**.
    - Choose **Save changes**

The last step associates this new route table with the public subnet.

30. Choose the **Subnet associations** tab.
31. In the **Subnets without explicit associations** section, choose **Edit subnet associations**
32. Select the row with **Public Subnet**.
33. Choose **Save associations**

    The public subnet is now public because it has a route table entry that sends traffic to the internet via the internet gateway.

    To summarize, you can create a public subnet by following these steps:
    - Create an internet gateway.
    - Create a route table.
    - Add a route to the route table that directs 0.0.0.0/0 traffic to the internet gateway.
    - Associate the route table with a subnet, which then becomes a public subnet.

Task 5: Creating a security group for the application server

A *security group* acts as a virtual firewall for instances to control inbound and outbound traffic. Security groups operate at the level of the elastic network interface for the instance. Security groups do not operate at the subnet level. Thus, each instance can have its own firewall that controls traffic. If you do not specify a particular security group at launch time, the instance is automatically assigned to the default security group for the VPC.

In this task, you create a security group that allows users to access your application server via HTTP.

34. In the left navigation pane, choose **Security Groups**.
35. Choose **Create security group** and configure the following settings:
    - For **Security group name**, enter `App-SG`
    - For **Description**, enter `Allow HTTP traffic`
    - For **VPC**, choose **Lab VPC**.
    - Choose **Create security group**
36. Choose the **Inbound Rules** tab.

    The settings for **Inbound Rules** determine what traffic is permitted to reach the instance. You configure it to permit HTTP (port 80) traffic that comes from anywhere on the internet (0.0.0.0/0).
37. Choose **Edit inbound rules**
38. Choose **Add rule** and then configure the following settings:
    - For **Type**, choose **HTTP**.
    - From the **Source type** dropdown list, choose **Anywhere IPv4**.
    - For **Description**, enter `Allow web access`
    - Choose **Save rules**

You use this **App-SG** in the next task.

Task 6: Launching an application server in the public subnet

To test that your VPC is correctly configured, you now launch an EC2 instance into the public subnet. You also confirm that you can access the EC2 instance from the internet.

39. On the **Services** menu, choose **EC2**.
40. Choose **Launch instance** and then select **Launch instance** from the dropdown list. Configure the following options:
    - In the **Name and tags** pane, in the **Name** text box, enter `App Server`
    - In the **Application and OS Images (Amazon Machine Image)** section, keep default selection, **Amazon Linux 2**.
    - In the **Instance type** section, keep the default instance type, **t2.micro**.
    - In the **Key pair (login)** section, from the **Key pair name - *required*** dropdown list, choose **Proceed without a key pair (not recommended)**.
    - In the **Network settings** section, choose **Edit**
        - From the **VPC - *required*** dropdown list, choose **Lab VPC**.
        - From the **Subnet** dropdown list, choose **Public Subnet**.
        - Ensure that **Auto-assign public IP** is **Enable**.
    - In the **Firewall (security groups)** section, choose **Select existing security group**
        - From the **Common security groups** dropdown list, choose `App-SG`.
    - In the **Configure storage** section, keep the default storage configuration.
    - Expand the **Advanced details** section.
        - For **IAM instance profile**, choose the role **Inventory-App-Role**.
        - Scroll down to **User data** section, copy and paste the below code in the block.

```
xxxxxxxxxx
#!/bin/bash
# Install Apache Web Server and PHP
yum install -y httpd mysql
amazon-linux-extras install -y php7.2
# Download Lab files
wget https://aws-tc-largeobjects.s3-us-west-2.amazonaws.com/ILT-TF-200-ACACAD-20-EN/mod6-guided/scripts/inventory-app.zip
unzip inventory-app.zip -d /var/www/html/
# Download and install the AWS SDK for PHP
wget https://github.com/aws/aws-sdk-php/releases/download/3.62.3/aws.zip
unzip aws -d /var/www/html
# Turn on web server
chkconfig httpd on
service httpd start
```

    - From the **Summary** section, choose **Launch instance**
41. Choose **View all instances**
42. Wait for the application server to fully launch. It should display the following status:
    - **Instance State:** Running

You can choose refresh occasionally to update the display.

43. Select **App Server**.
44. From the **Details** tab, copy the **Public IPv4 address** address.
45. Open a new browser tab, paste the IP address you just copied, and press Enter.

If you configured the VPC correctly, the Inventory application and this message should appear: **Please configure Settings to connect to database**. You have not configured any database settings yet, but the appearance of the Inventory application demonstrates that the public subnet was correctly configured.

If the Inventory application does not appear, wait for 60 seconds and refresh the page to try again. It can take a couple of minutes for the EC2 instance to boot and run the script that installs the software.