DevOps AWS

# **What am I going to learn today?**

## **Public IP Address Vs Private IP Address**

## **How AWS Firewall Works**

## **Monolithic Service Vs Micro Service**

## **Microservices advantages**

Before going to dive into the topics we need to understand all the modern applications that are built using 3 Tier Architecture.

1. Web Tier
2. App Tier
3. Database Tier

All these 3 tiers are connected, and the user will connect or request the data through Web Tier

# **Now Let’s understand the Public IP and Private IP**

## Public IP Address:

1. Public IP will be given to you by ISP (Internet Service Provider)
2. Modem will use this public IP to create private IP addresses to connect your devices to the internet.

## Private IP Address:

On the internet, there are billions of devices like computers, phones, and servers, and they all need unique addresses to communicate with each other.

1. Private IP addresses are used inside private networks to help devices find and talk to each other, but they are not used to communicate with devices on the internet directly.
2. Private IP addresses are helpful for keeping our local network separate from the vast internet, and they add an extra layer of security because they are not directly visible to the outside world.

# How AWS Firewall Works?

Before understanding the Firewall concept, we need to know about the security groups.

Security Groups are nothing but a firewall.

In AWS security groups are nothing but a firewall service

## There two different rules in security groups

1. Inbound Rule -> Traffic coming in to your server
   1. Source: Internet i.e any server outside
   2. Destination: your EC2 instance
2. Outbound Rule -> Traffic going out from your server
   1. Source: Your EC2 instance
   2. Destination: Internet i.e any server outside

# Monolithic Vs Microservices

In earlier stages of software development architecture, all the components are built together, and the application becomes very big as it includes front end and backend applications. Maintenance is a big headache.

## Monolithic Service:

A monolithic service is a type of software architecture where all the components of an application are tightly integrated into a single, large, and interconnected system. In other words, all the features and functionalities of the application are packaged together into one unit, making it a self-contained entity.

## Disadvantages of Monolithic Service:

1. **Scalability:** Scaling a monolithic service can be challenging. As the application grows, it may become more resource-intensive and difficult to scale horizontally. Scaling can be limited to vertical scaling (adding more resources to the existing server), which has its limits.
2. **Maintenance Complexity:** As the application grows, maintaining a monolithic service can become cumbersome. Adding new features or fixing bugs might impact other parts of the application, making it more prone to errors and harder to manage.
3. **Team Collaboration:** In larger projects, the monolithic architecture might hinder team collaboration. Different teams might need to work on the same codebase, leading to potential conflicts and coordination challenges.
4. **Deployment Risks:** Since all the components are tightly integrated, deploying updates or changes to one part of the application may affect the entire system. This increases the risk of introducing bugs or downtime during updates.
5. **Technology Limitations:** Monolithic services often involve using the same technology stack throughout the application. This can restrict the ability to adopt newer, more suitable technologies for specific parts of the application.

## Micro Services:

Microservices architecture is a software development approach where a large application is broken down into smaller, independent services that communicate with each other through well-defined APIs. Each service focuses on a specific business capability and can be developed, deployed, and scaled independently.

## Advantages of Microservices Architecture:

1.**Scalability:** Microservices allow for more granular scalability. Instead of scaling the entire application, you can scale only the specific services that need more resources, leading to better resource utilization and cost efficiency.

2.**Flexibility and Agility:** Microservices promote flexibility in development, as each service can be developed using different technologies and frameworks best suited for its purpose. This flexibility allows teams to adopt new technologies and make changes without affecting the entire application.

3.**Independent Deployment:** Microservices can be deployed independently of each other, enabling continuous deployment and faster release cycles. This means that updates, bug fixes, and new features can be rolled out to specific services without affecting the entire system.

4.**Team Independence:** Different teams can work on different services concurrently, which promotes parallel development and allows teams to be more autonomous. This can lead to better productivity and faster development.

5.**Easier Maintenance and Debugging:** With smaller services, maintaining and debugging becomes more manageable. Developers can focus on specific areas of the application without dealing with the complexities of a monolithic codebase.

6.**Improved Performance:** Microservices can lead to improved performance, as services can be optimized individually for their specific tasks, reducing the overhead associated with monolithic applications.

7.**Better Scalability:** Microservices architecture supports horizontal scaling, which allows you to add more instances of a service to handle increased load, resulting in better scalability.