



Key Takeaways

AWS Services

Introduction to AWS

AWS stands for Amazon Web Services

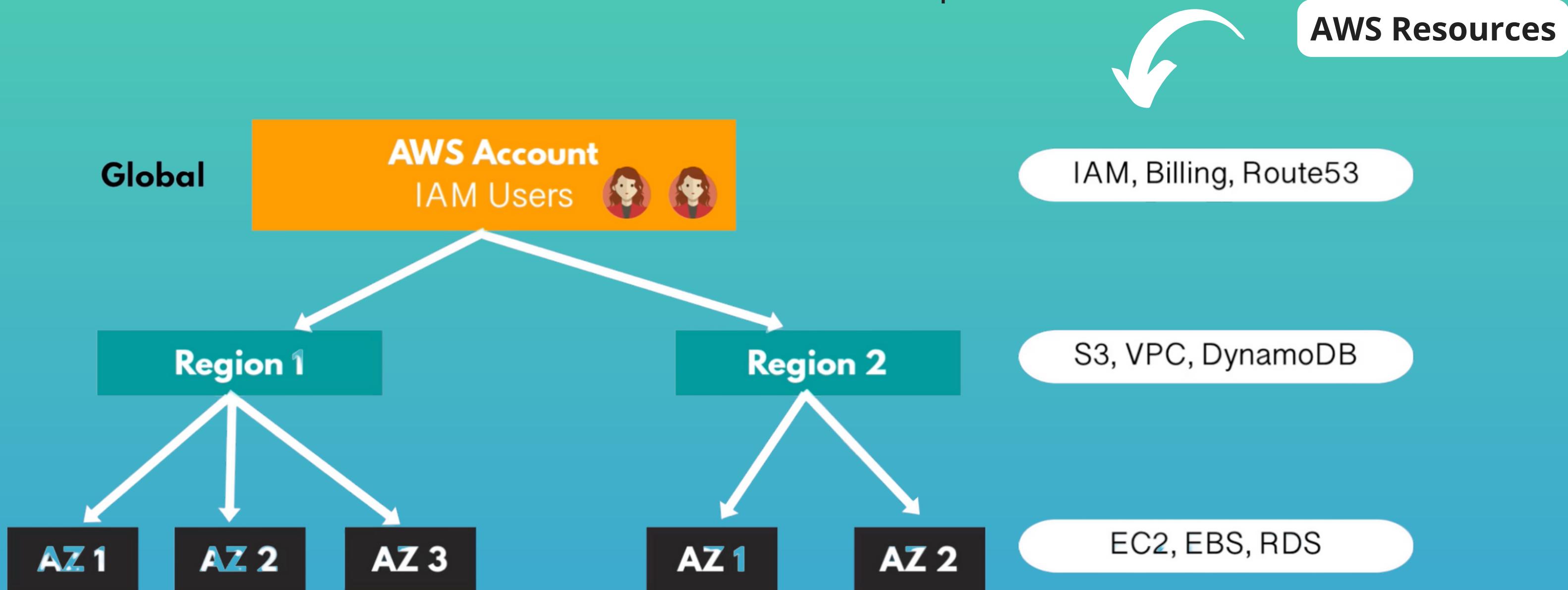
Currently the most popular cloud platform, with a huge **collection of services**:

The screenshot shows the AWS Management Console interface. On the left, there's a sidebar with a navigation menu. The 'Recently visited' section is highlighted in orange. Below it are 'Favorites' and 'All services'. A large list of services is shown under 'All services', including Analytics, Application Integration, AWS Cost Management, Blockchain, Business Applications, Compute, Containers, Customer Enablement, Database, Developer Tools, End User Computing, Front-end Web & Mobile, Game Development, and Internet of Things. To the right, a 'Recently visited' panel is open, listing Console Home, VPC, EC2, AWS Organizations, AWS Health Dashboard, Elastic Beanstalk, IAM, and AWS Cost Explorer. Further right, the 'Welcome to AWS' dashboard is visible, featuring sections for 'Getting started with AWS', 'Training and certification', and 'What's new with AWS?'. The top right corner shows the user's location as Frankfurt and their email as admin @ 3306-7354-7330.



AWS Account and Services Scope

- In AWS you have **3 scopes**: Global, Region and Availability Zones
- Different resources will be created in one of those scopes



Create an AWS Account - 1

- Register on AWS

Sign up using your email address

1. Open the [Amazon Web Services \(AWS\) home page](#).

2. Choose **Create an AWS Account**.

Note: If you signed in to AWS recently, choose **Sign in to the Console**. If **Create a new AWS account** isn't visible, first choose **Sign in to a different account**, and then choose **Create a new AWS account**.

3. In **Root user email address**, enter your email address, and then choose **Verify email address**. An AWS verification code will be sent to your email address.

Tip: For **Root user email address**, use a corporate email address (such as info admins@example.com) or email box if your account is for a company. This way, if an individual's corporate email address (for example, your personal email address) changes, your company can retain access to the AWS account even if the individual leaves the company. The email address you choose must be valid and active. Be sure that you protect access to these distribution lists by using strong, unique passwords for your everyday tasks. It's a best practice to [enable MFA for your account](#) to secure your AWS resources.

Add a payment method

On the **Billing information** page, enter the information about your payment method, and then choose **Verify and Add**.

If you're signing up for an Amazon Web Services India Private Limited (AWS India) account, then you must provide your CVV for the verification process. You might also have to enter a one-time password, depending on your bank. AWS India charges your payment method two Indian Rupees (INR), as part of the verification process. AWS India refunds the two INR after the verification is complete.

If you want to use a different billing address for your AWS billing information, choose **Use a new address**. Then, choose **Verify and Continue**.

Important: You can't proceed with the sign-up process until you add a valid payment method.

Create an AWS Account - 2

- For first time registration, you get **1-year free of basic resources**

The screenshot shows the AWS Free Tier landing page. At the top, there's a navigation bar with links: AWS Free Tier, Overview, Free Tier Categories, How to Create an Account, Featured Offers for Business, FAQs, and Terms and Conditions. Below this is a section titled "Explore Top Product Categories" with icons for Compute, Database, Storage, Containers, Web & Mobile Apps, Serverless, and Machine Learning. A large orange circle with a white exclamation mark is overlaid on the page.

Free Tier details

Filter by: [Clear all filters](#)

Tier Type

- Featured
- 12 Months Free
- Always Free
- Trials

Product Categories

- Analytics
- Application Integration
- Business Productivity
- Compute
- Containers
- Customer Engagement

Search free tier products

COMPUTE		STORAGE		DATABASE	
Free Tier	12 MONTHS FREE	Free Tier	12 MONTHS FREE	Free Tier	12 MONTHS FREE
Amazon EC2		Amazon S3		Amazon RDS	
750 Hours		5 GB		750 Hours	
per month		of standard storage		per month of database usage (applicable DB engines)	
Resizable compute capacity in the Cloud.		Secure, durable, and scalable object storage infrastructure.		Managed Relational Database Service for MySQL, PostgreSQL, MariaDB, or SQL Server.	
750 hours per month of t4g.small instance		5 GB of Standard Storage			

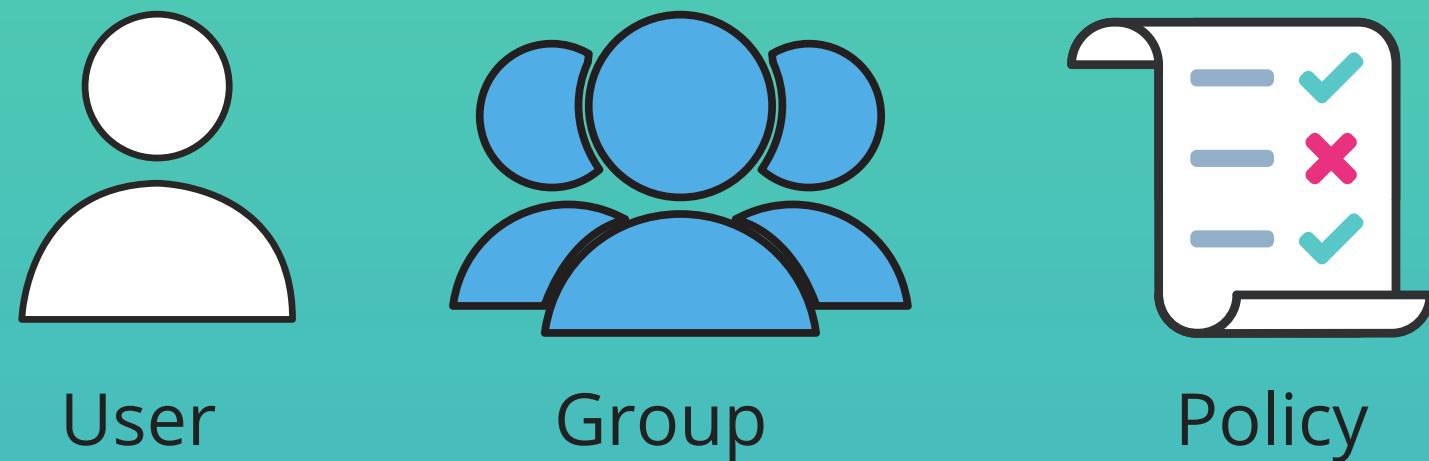
Some services are **NOT included in the free tier!**

Delete Resources that you don't need anymore

Core AWS Services

Identity and Access Management (IAM) - 1

- With the IAM service you can specify **who can access which services and resources**
- Create and manage AWS Users and Groups
- Assign policies (set of permissions)



- ROOT user is created **by default**
- ROOT user has **unlimited privileges**
- Best Practice: **Create an admin user** with less privileges that manages the whole AWS account



Identity and Access Management (IAM) - 2

Different Types of IAM Users

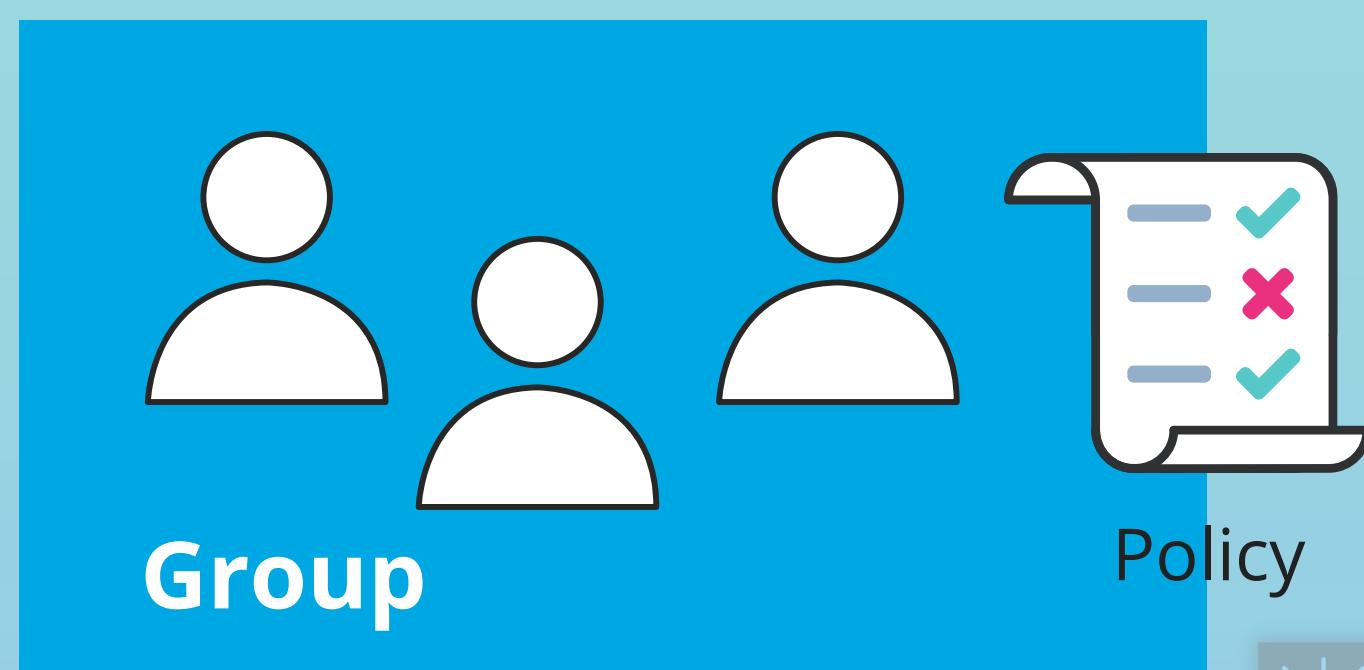
1. Human Users

2. **System Users:** For example Jenkins needs permission to deploy Docker containers on AWS



Groups

- For granting access to multiple IAM users



Identity and Access Management (IAM) - 3



IAM Roles

- An IAM role is similar to an IAM user
- Instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it
- Policies cannot be assigned to AWS services directly
- So a role is used to **grant AWS services access to other AWS services**



IAM User



EC2



ECS



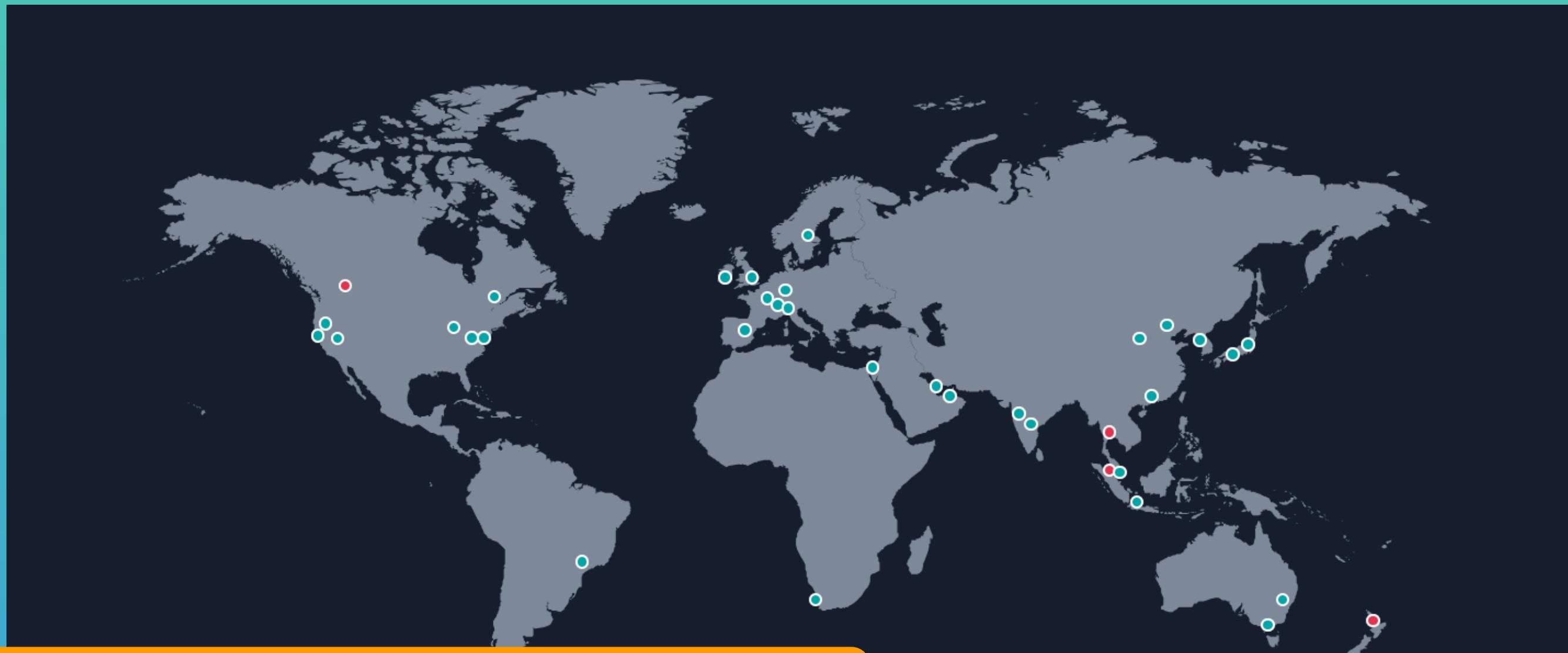
EKS

How to:

1. Create IAM Role
2. Assign Role AWS Service
3. Attach Policies to that Role

Regions & Availability Zones (AZ) - 1

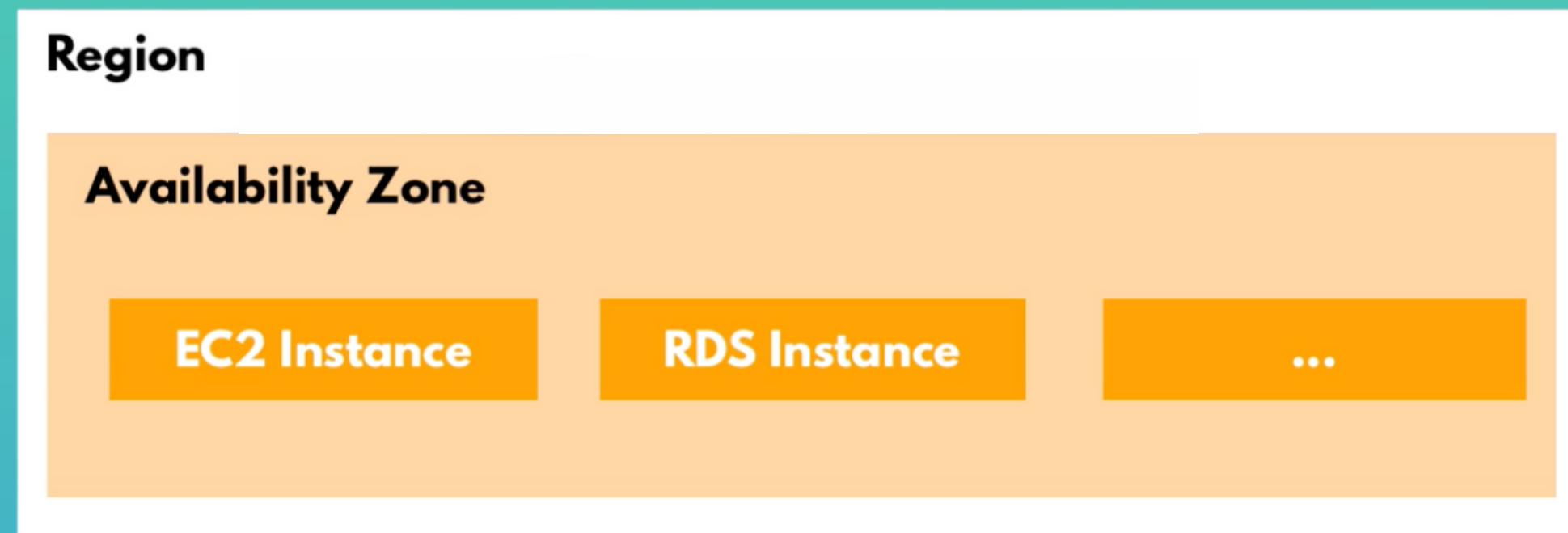
- Cloud providers have physical data centers
- Data centers are located all over the world
- **Region** = physical location, where data centers are clustered



You have to select which region you want your server in

Regions & Availability Zones (AZ) - 2

- Each group of logical data centers is called an Availability Zone
- **Availability Zones** = one or more discrete data centers



Regions & Availability Zones	
Region	Availability Zone
US East (N. Virginia)	us-east-1
US East (Ohio)	us-east-2
US West (N. California)	us-west-1
US West (Oregon)	us-west-2
Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (Osaka)	ap-northeast-3
Asia Pacific (Seoul)	ap-northeast-2
Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2
Asia Pacific (Tokyo)	ap-northeast-1
Canada (Central)	ca-central-1
Europe (Frankfurt)	eu-central-1
Europe (Ireland)	eu-west-1
Europe (London)	eu-west-2
Europe (Paris)	eu-west-3
Europe (Stockholm)	eu-north-1
South America (Sao Paulo)	sa-east-1

Virtual Private Cloud (VPC)

- VPC is **your own isolated network** in the cloud



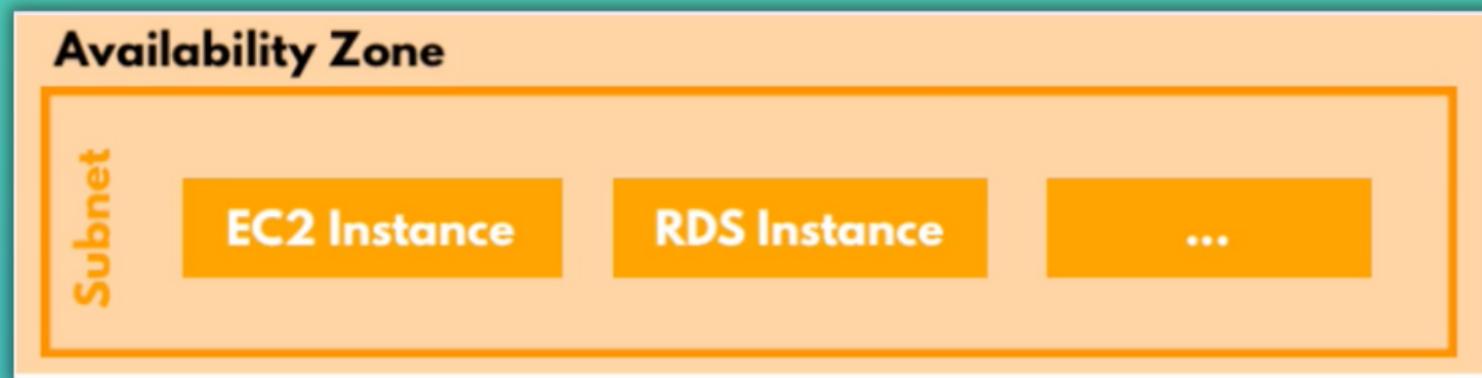
- You have a default VPC in each Region
- A VPC **spans all the AZs (Subnets)** in that Region
- Multiple VPCs in different Regions
- A VPC is just like a **virtual representation of physical network infrastructure**: Server setup, network configuration moved to cloud



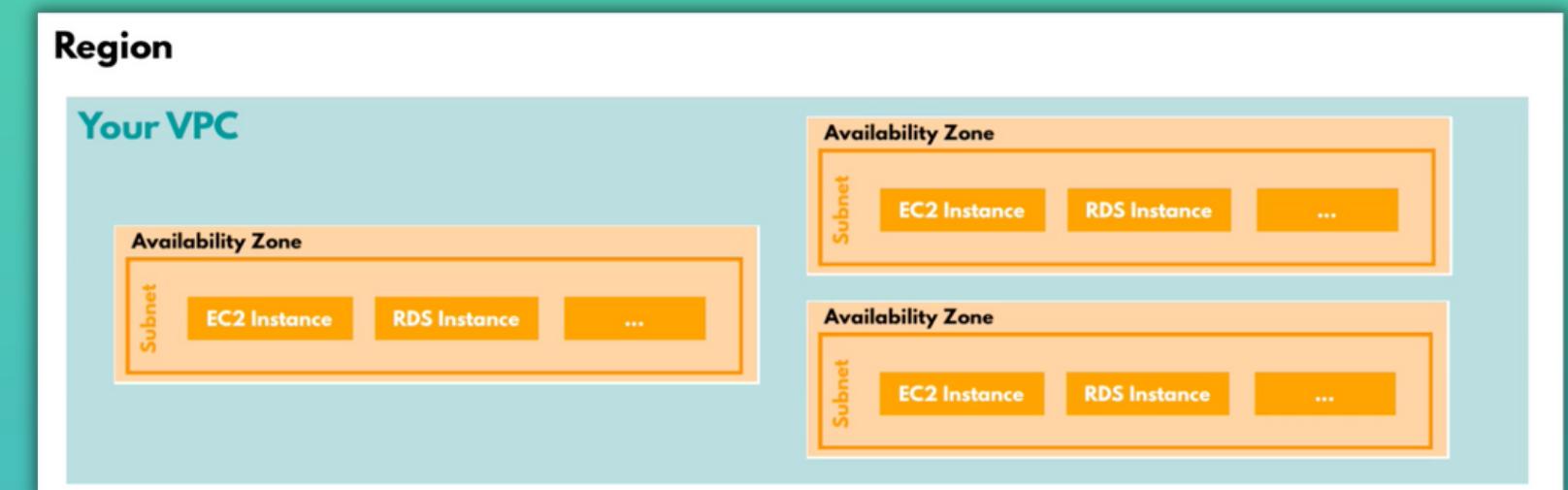
Your resources always have to run in a VPC!

Subnet - 1

- A Subnet is a **range of IP addresses** in your VPC
- It's like a **private network inside a network**:

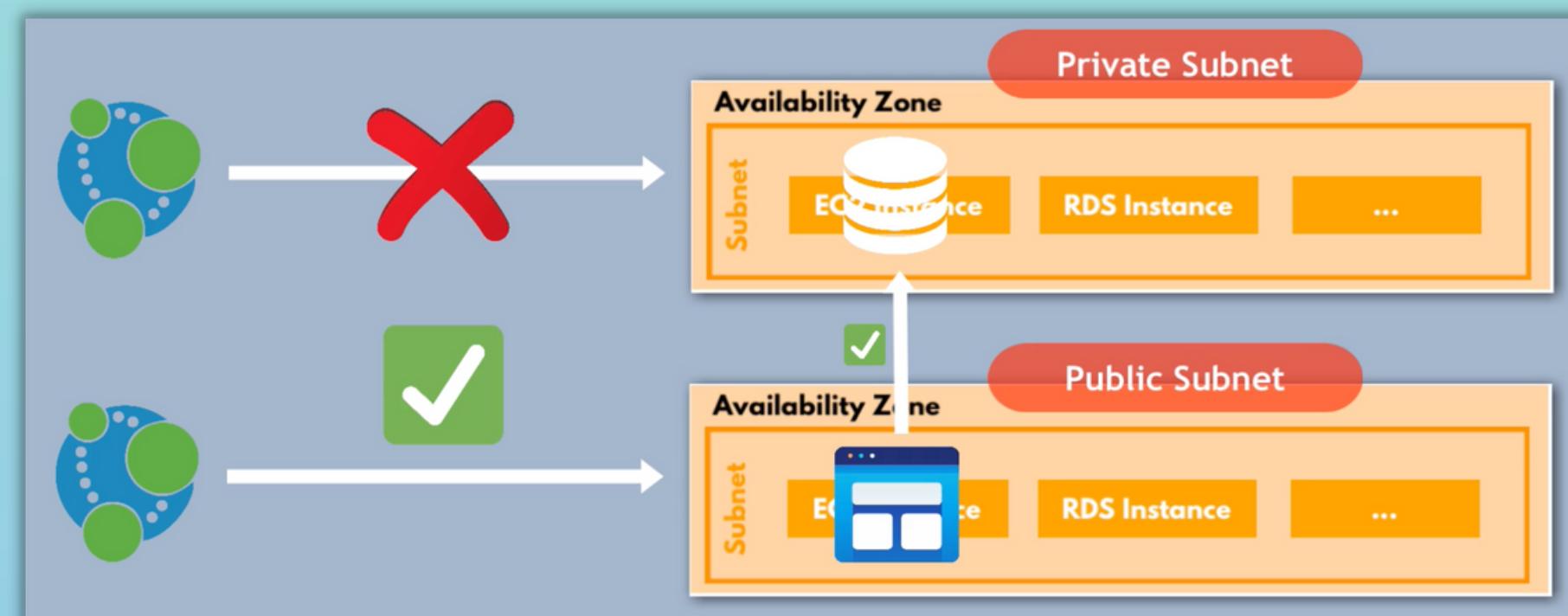
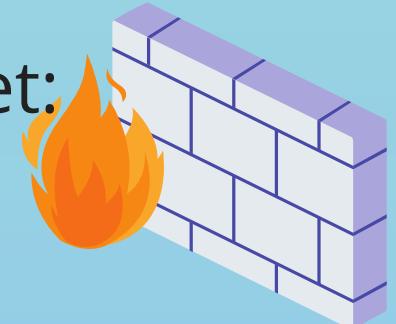


- You have a subnet for each Availability Zone:



Private and Public Subnets

- Based on your firewall configuration you can have a private and/or public subnet:



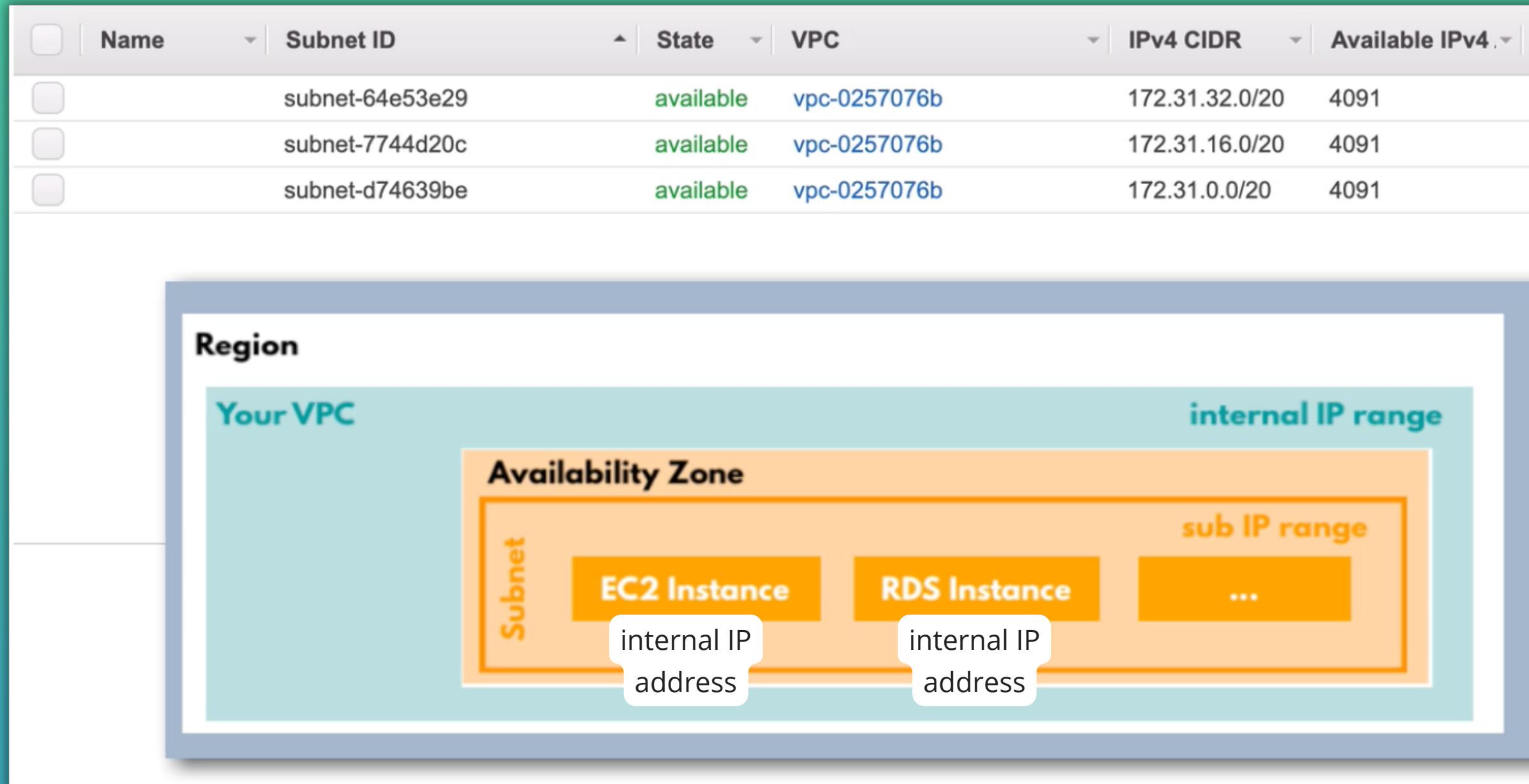
Subnet - 2

- A subnet has a default range of **internal IP addresses**
- When you create a new resource such as an EC2 instance, an IP address is assigned within this subnet's IP range
- This allows **communication inside the VPC**

Internet Gateway

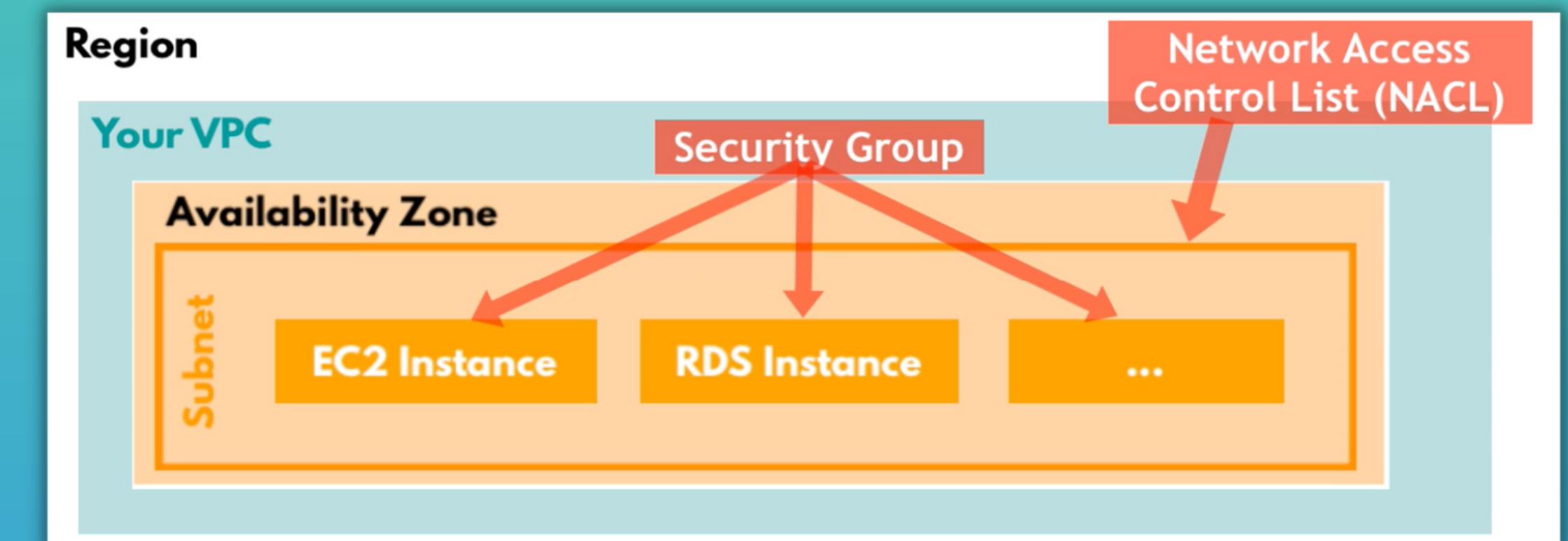
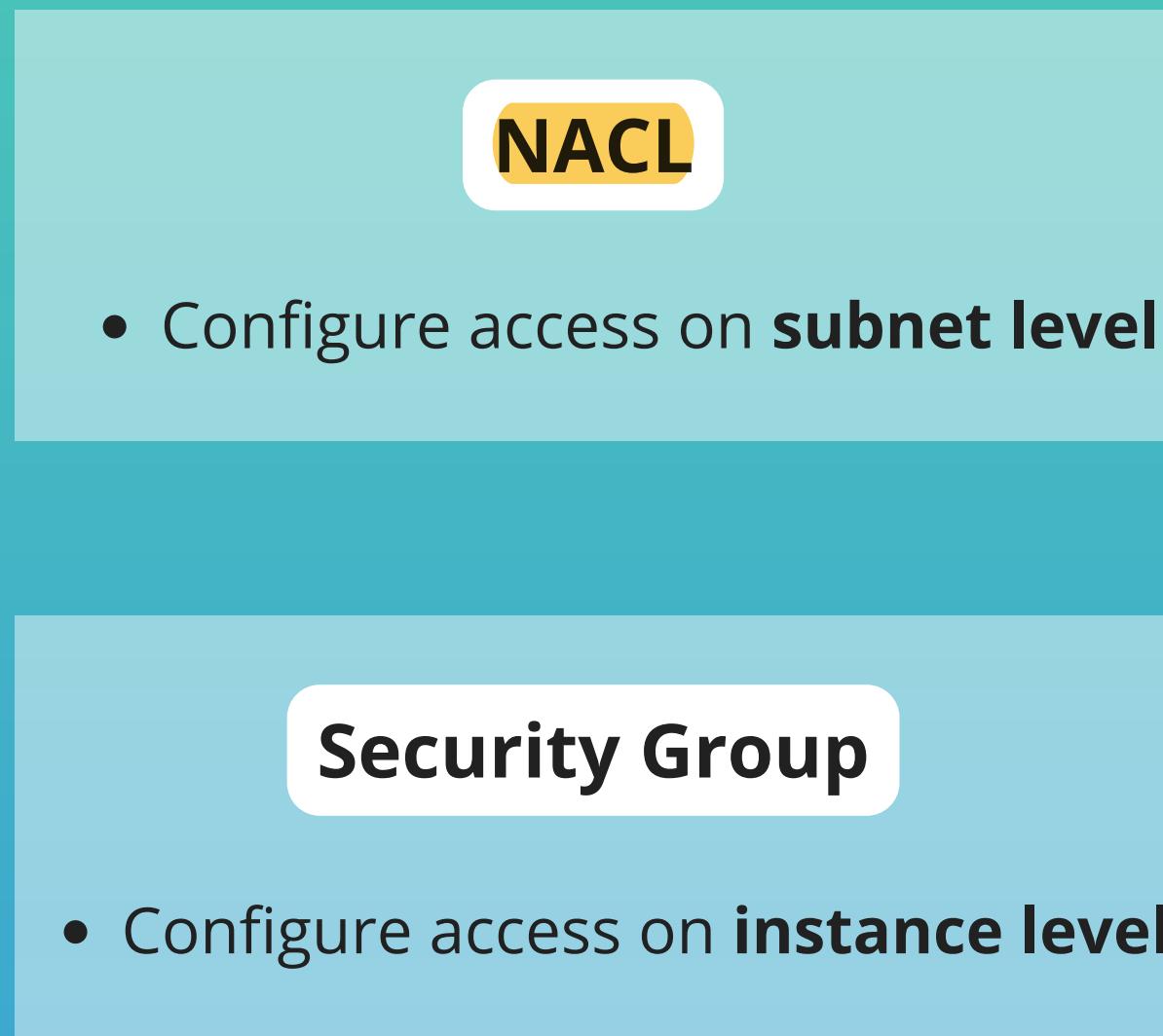
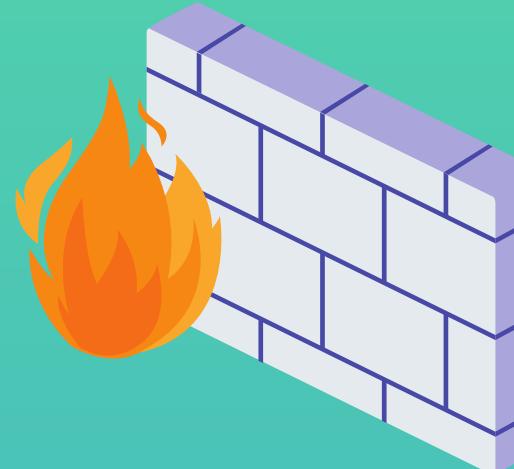


- Using an internet gateway you can **connect the VPC** or its subnets to the **internet**



Security - Controlling Access

- Of course you need to **secure your resources**:
 - Control access to your VPC
 - Control access to your individual server instances



CIDR Block

- When you create a subnet, you **specify the IPv4 CIDR block for the subnet**, which is a subset of the VPC CIDR block
- Defines a range of IP addresses
- CIDR = Classless Inter-Domain Routing

How to choose a CIDR Block:

 Subnet Calculator

172.31.0.0

Input 172.31.0.0/16	Input IP 172.31.0.0	Input Long 2887712768	/16	Calculate
CIDR 172.31.0.0/16	CIDR IP Range 172.31.0.0 - 172.31.255.255	CIDR Long Range 2887712768 - 2887778303		
IPs in Range 65,536	Mask Bits 16	Subnet Mask 255.255.0.0		

Available Subnets:

- /32 1 IPs
- /31 2 IPs
- /30 4 IPs
- /29 8 IPs
- /28 16 IPs
- /27 32 IPs
- /26 64 IPs
- /25 128 IPs
- /24 256 IPs
- /23 512 IPs

Subnets (3) Info		Actions
<input type="text"/> Filter subnets		
VPC	IPv4 CIDR	
vpc-0257076b	172.31.0.0/20	
vpc-0257076b	172.31.16.0/20	
vpc-0257076b	172.31.32.0/20	

VPC CIDR block (IP range)	
Subnet	sub CIDR block
Subnet	sub CIDR block

Visual Subnet Calculator

Enter the network you wish to subnet:

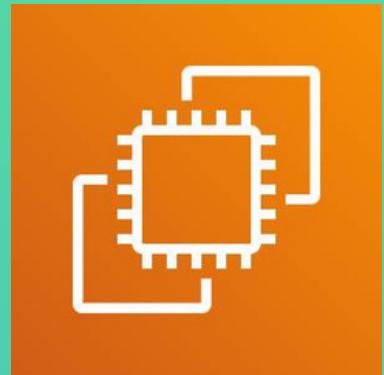
Network Address Mask bits
192.168.0.0 / 16 Update Reset

Show columns: Subnet address Netmask Range of addresses Useable IPs Hosts Divide Join

Click below to split and join subnets.
If you wish to save this subnetting for later, bookmark [this hyperlink](#).

Subnet address	Range of addresses	Useable IPs	Hosts	Divide	Join
10.0.0.0/18	10.0.0.0 - 10.0.63.255	10.0.0.1 - 10.0.63.254	16382	Divide	/18
10.0.64.0/18	10.0.64.0 - 10.0.127.255	10.0.64.1 - 10.0.127.254	16382	Divide	/18
10.0.128.0/18	10.0.128.0 - 10.0.191.255	10.0.128.1 - 10.0.191.254	16382	Divide	/18
10.0.192.0/18	10.0.192.0 - 10.0.255.255	10.0.192.1 - 10.0.255.254	16382	Divide	/18

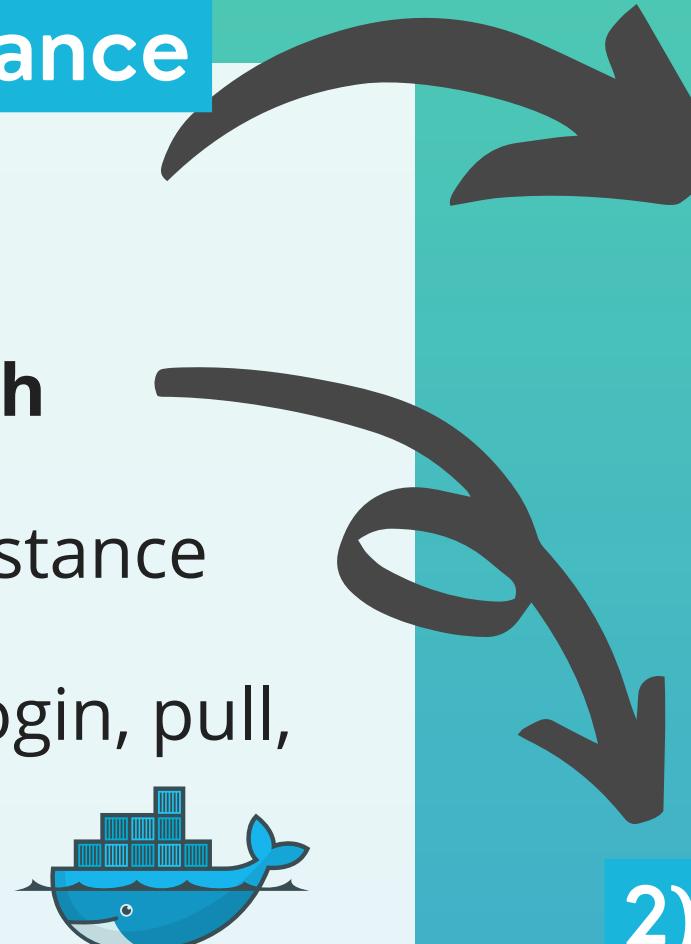
Elastic Compute Cloud (EC2)



- A **virtual server** in AWS, providing computing capacity

Steps to deploy a web application on an EC2 instance

1. Create an EC2 instance on AWS
2. Connect to EC2 instance with **ssh**
3. Install Docker on remote EC2 instance
4. Run Docker container (docker login, pull, run) from private repository
5. Configure EC2 Security Group to access application externally from the browser



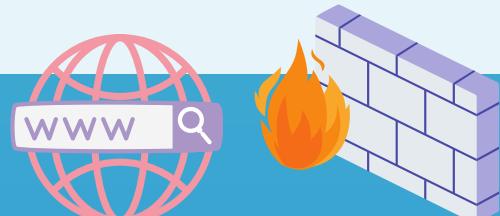
1) Steps to launch EC2 instance

1. Choose OS Image
2. Choose capacity
3. Network configurations
4. Add storage
5. Add tags
6. Configure Security Group



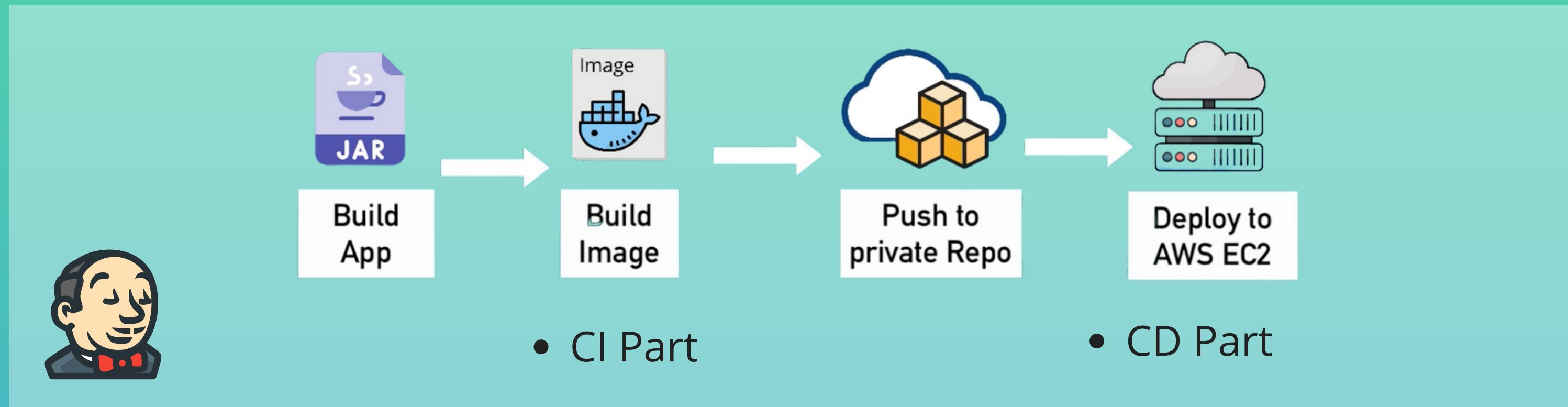
2) Connect to EC2 server via ssh using downloaded private ssh key:

```
nana@macbook /Users/nana  
% ssh -i .ssh/docker-server.pem ec2-user@18.184.54.160
```



CD part of CI/CD - Deploy to EC2 from Jenkins

Deploy to EC2 Instance from Jenkins - 1



Steps to deploy to EC2 instance

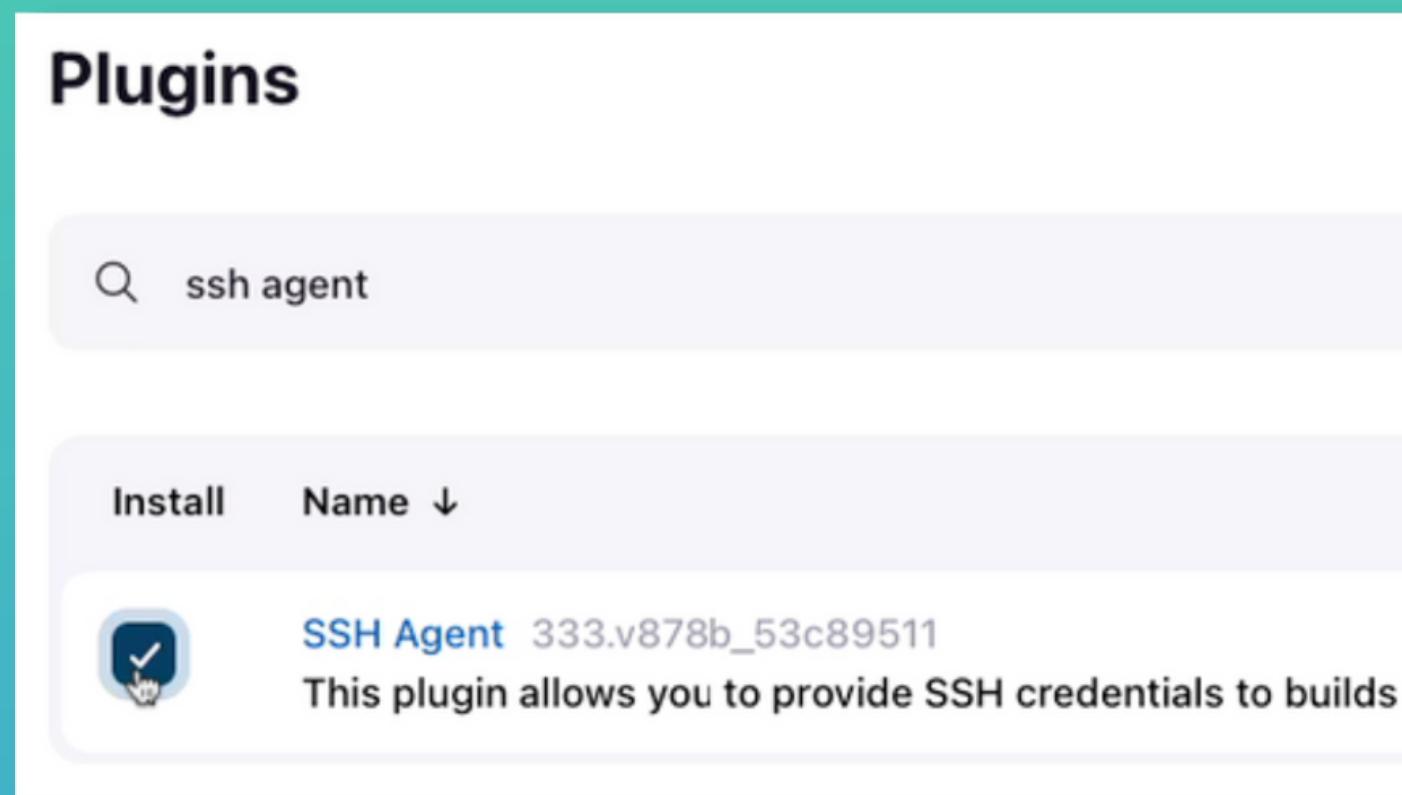
1. Connect to EC2 instance from Jenkins server via **ssh** (ssh agent)
2. Execute "*docker run*" on EC2 instance



Deploy to EC2 Instance from Jenkins - 2

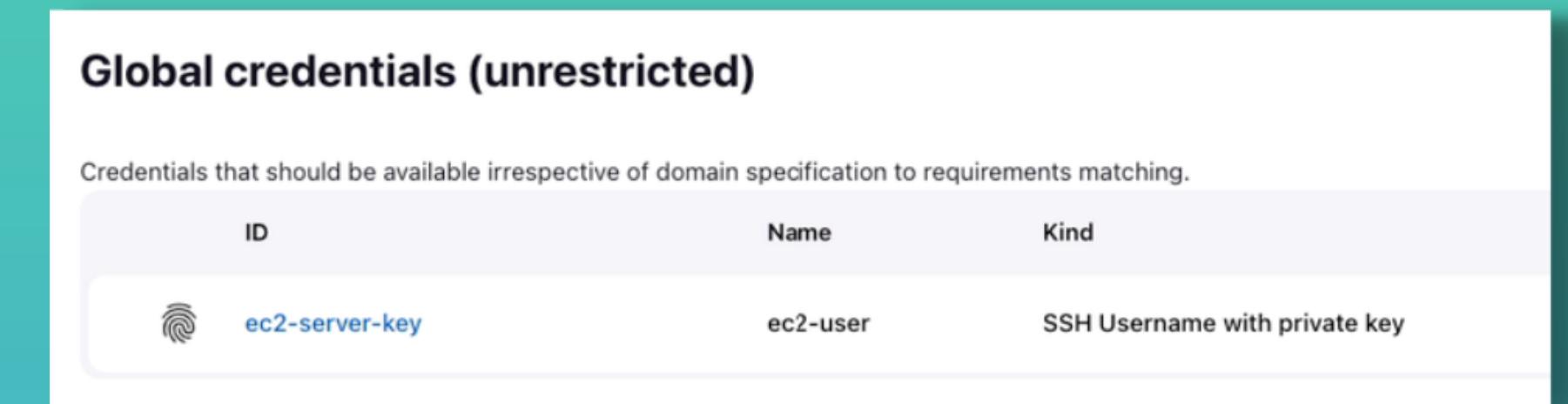
Steps to connect to EC2 instance from Jenkins server via ssh (ssh agent)

1) Install SSH Agent Plugin:



The screenshot shows the Jenkins Plugins page. A search bar at the top contains the text "ssh agent". Below the search bar, there are two buttons: "Install" and "Name ↓". A list of plugins is displayed, with the "SSH Agent" plugin highlighted by a blue border and a checkmark icon. The plugin's description below it reads: "This plugin allows you to provide SSH credentials to builds".

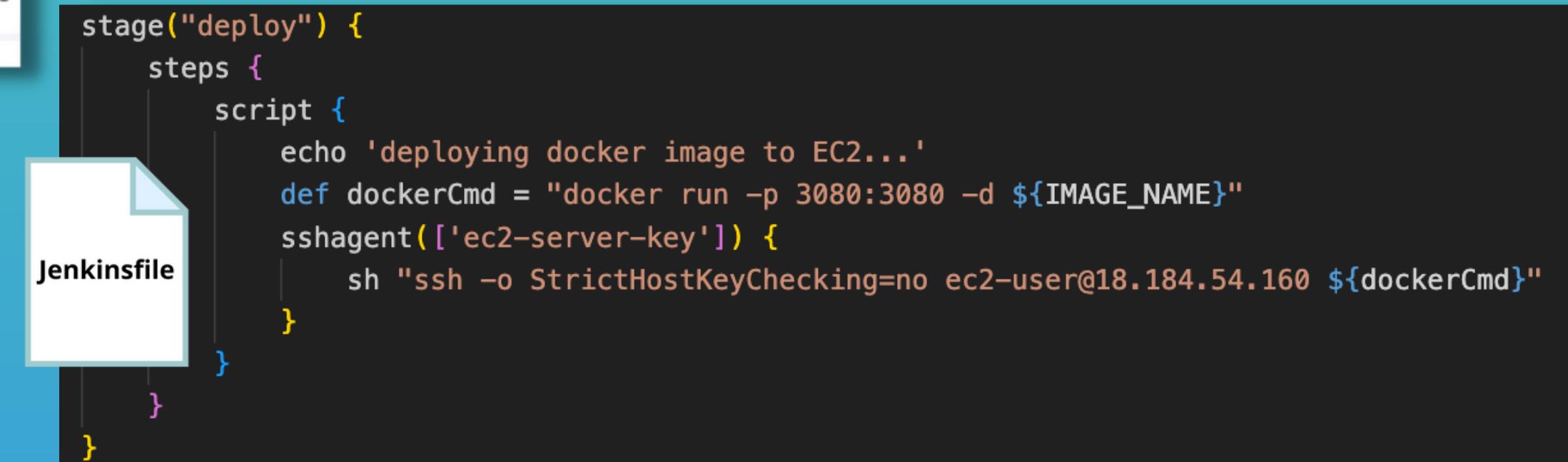
2) Configure credentials



The screenshot shows the "Global credentials (unrestricted)" configuration page in Jenkins. It displays a table with one row of data. The columns are labeled "ID", "Name", and "Kind". The data row contains the values "ec2-server-key", "ec2-user", and "SSH Username with private key".

ID	Name	Kind
ec2-server-key	ec2-user	SSH Username with private key

3) Use credential in "deploy" stage in Jenkinsfile



```
stage("deploy") {
    steps {
        script {
            echo 'deploying docker image to EC2...'
            def dockerCmd = "docker run -p 3080:3080 -d ${IMAGE_NAME}"
            sshagent(['ec2-server-key']) {
                sh "ssh -o StrictHostKeyChecking=no ec2-user@18.184.54.160 ${dockerCmd}"
            }
        }
    }
}
```

The screenshot shows a Jenkinsfile with a "deploy" stage. The stage contains a "script" block. Inside the script block, there is an "sshagent" step with the argument "['ec2-server-key']". This step is used to temporarily set the "ec2-user" credential for the duration of the "sh" command, which runs a Docker command on an EC2 instance.

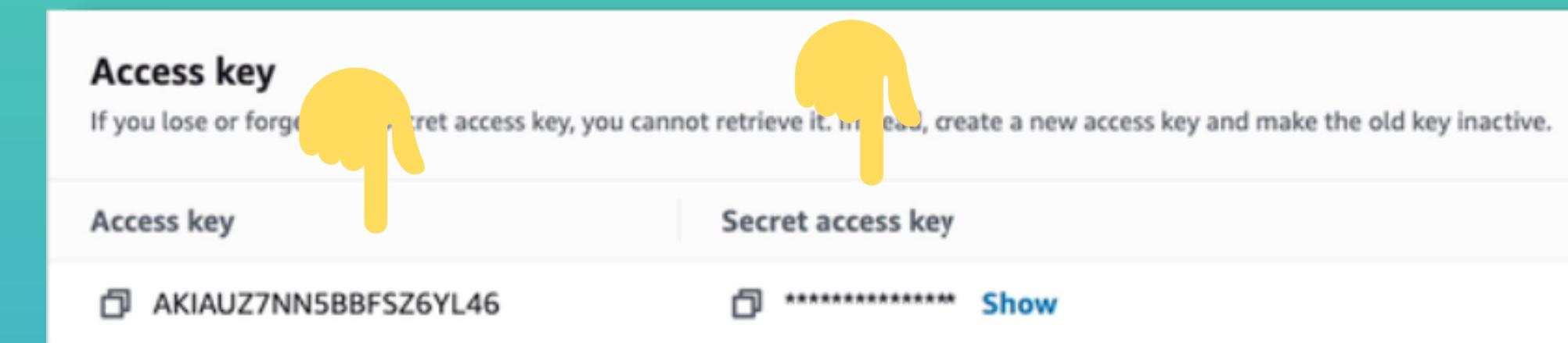
Introduction to AWS CLI

AWS Command Line Interface (CLI) - 1

- Instead of using the UI, we can use the AWS CLI to interact with our AWS account

UI Access through password

CLI Access through Access key ID and Secret Access Key



- For that you need to configure your AWS CLI:

```
nana@macbook /Users/nana
% aws configure
AWS Access Key ID [None]: AKIAUZ7NN5BBFSZ6YL46
AWS Secret Access Key [None]: /25ekyXYnz16yTMxC7C0uLWVjghuQvf9/YXIhkPM
Default region name [None]: eu-central-1
Default output format [None]: json
```

AWS Command Line Interface (CLI) - 2

Command Structure

- **aws** = the base call to the aws program
- **command** = the AWS service
- **subcommand** = specifies which operation to perform

```
$ aws ec2 run-instances \
  --image-id ami-12345678 \
  --count 1 \
  --instance-type t2.micro \
  --security-group-ids sg-187654321 \
  --subnet-id subnet-12345678 \
```



- Launch EC2 Instance via AWS CLI

AWS & Terraform - Preview



Before Infrastructure as Code

- Many commands to execute
- No overview
- Imperative commands telling how to do it



After Infrastructure as Code

- Create and manage resources with code
- Terraform is an IaC tool
- Declarative - describing the desired state

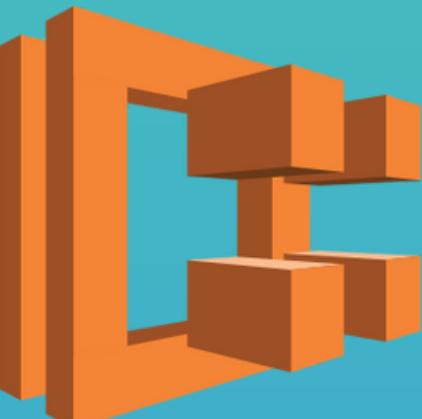


HashiCorp
Terraform

Container Services on AWS - Preview

- AWS provides **different services** to help you **deploy containerized workloads**

Elastic Container Registry (ECR) =
container registry to store, share
and deploy container images



Elastic Container Service (ECS) =
AWS proprietary container orchestration



Amazon EKS



Elastic Kubernetes Service (EKS) =
Amazon's Managed Kubernetes Service

Best Practices

- IAM best practices:

<https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html>

- VPC best practices:

<https://docs.aws.amazon.com/vpc/latest/userguide/vpc-security-best-practices.html>

- EC2 best practices:

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-best-practices.html>

- Keep your .pem file in the “standard” location in .ssh directory in your \$HOME. I.e. /Users/\$USER/.ssh/. You should protect this directory with permission 400
- You should not share these .pem files with your co-workers. Each user should generate their own SSH keypair and their public key should be deployed to each system they need access to. Private keys should be private to each user, generated by them.

