



Key Takeaways

AWS Services

Introduction to AWS

AWS stands for Amazon Web Services

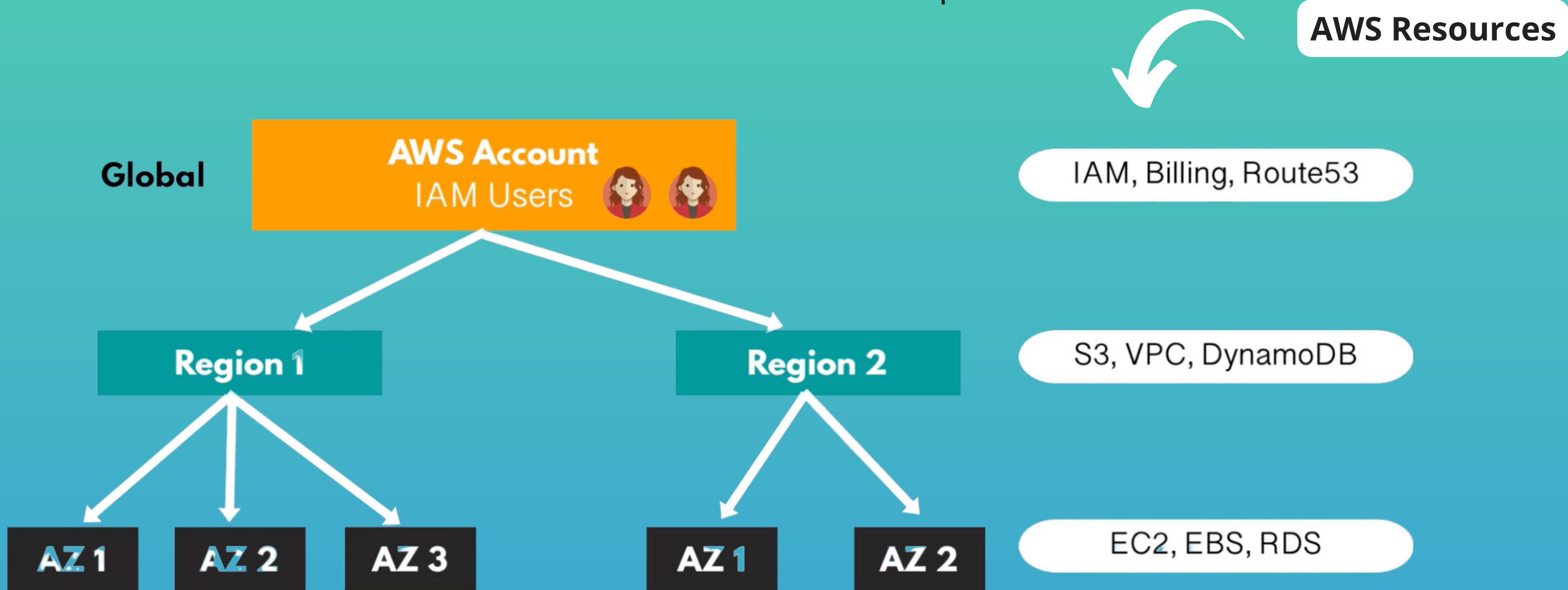
Currently most popular cloud platform, with a huge collection of services:



The screenshot shows the AWS Management Console homepage. The top navigation bar includes the AWS logo, a "Services" dropdown, a notification bell, user profile "Nicole Hiller", "Global" dropdown, and "Support" dropdown. On the left, there's a sidebar with "Favorites" (Resource Groups & Tag Editor), "Recently visited" (Billing, Console Home, IAM, EC2), and a "Database" section. The main content area is titled "All services" with a search bar. It lists services under various categories: Compute (EC2, Lightsail, Lambda, Batch, Elastic Beanstalk, Serverless Application Repositories, AWS Outposts, EC2 Image Builder), Storage (S3, EFS, FSx, S3 Glacier, Storage Gateway, AWS Backup), Database, Robotics (AWS RoboMaker), Customer Enablement (AWS IQ, Support), Blockchain (Amazon Managed Blockchain), Satellite (Ground Station), Quantum Technologies (Amazon Braket), Management & Governance, Machine Learning (Amazon SageMaker, Amazon Augmented AI, Amazon CodeGuru, Amazon Comprehend, Amazon Forecast, Managed Services, Activate for Startups), AR & VR (Amazon Sumerian), Application Integration (Step Functions, Amazon AppFlow, Amazon EventBridge, Amazon MQ, Simple Notification Service, Simple Queue Service, SWF), AWS Cost Management (AWS Cost Explorer, AWS Budgets, AWS Marketplace Subscriptions), and Customer Engagement (Amazon Connect). A "Management & Governance" section is partially visible at the bottom. The footer features the "TECHWORLD WITH NANA" logo with a lightbulb icon.

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

Create your account

1. Open the [Amazon Web Services 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. Enter your account information, and then choose **Continue**. Be sure that you enter your account information correctly, especially your email address. If you enter your email address incorrectly, you can't access your account.

4. Choose **Personal or Professional**.

Note: Personal accounts and professional accounts have the same features and functions.

5. Enter your company or personal information.

Important: For professional AWS accounts, it's a best practice to enter the company phone number rather than a personal cell phone. Configuring a [root account](#) with an individual [email address or a personal phone number](#) can make your account insecure.

6. Read and accept the [AWS Customer Agreement](#).

Note: Be sure that you read and [understand the terms of the AWS Customer Agreement](#).

7. Choose **Create Account and Continue**.

Add a payment method

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

Note: If you want to use a different billing address for your AWS billing information, choose **Use a new address** before you choose **Verify and Add**.

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 for Contact Sales, Support, English, My Account, Sign In to the Console, re:Invent, Products, Solutions, Pricing, Documentation, Learn, Partner Network, AWS Marketplace, Customer Enablement, and Explore More. Below the navigation is a search bar labeled "Search free tier products". The main content area is titled "Free Tier details" and features three columns: "COMPUTE" (Amazon EC2, 750 Hours per month), "STORAGE" (Amazon S3, 5 GB of standard storage), and "DATABASE" (Amazon RDS, 750 Hours per month of db.t2.micro database usage). Each service has a brief description and a "Learn more" link at the bottom.



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

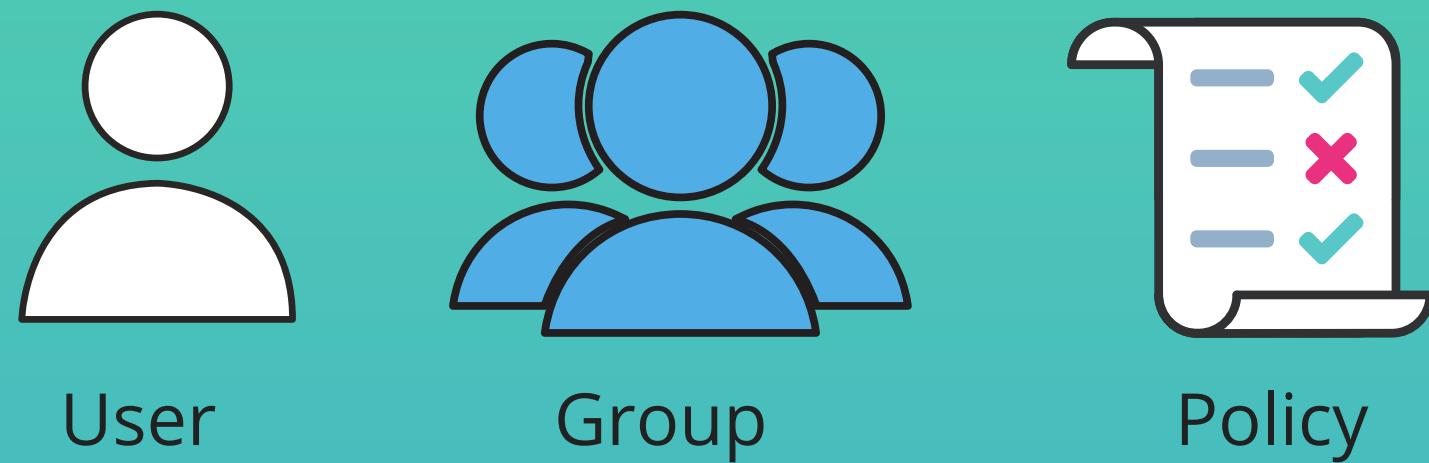


Delete Resources that you don't need anymore

Core AWS Services

Identity and Access Management (IAM) - 1

- With 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

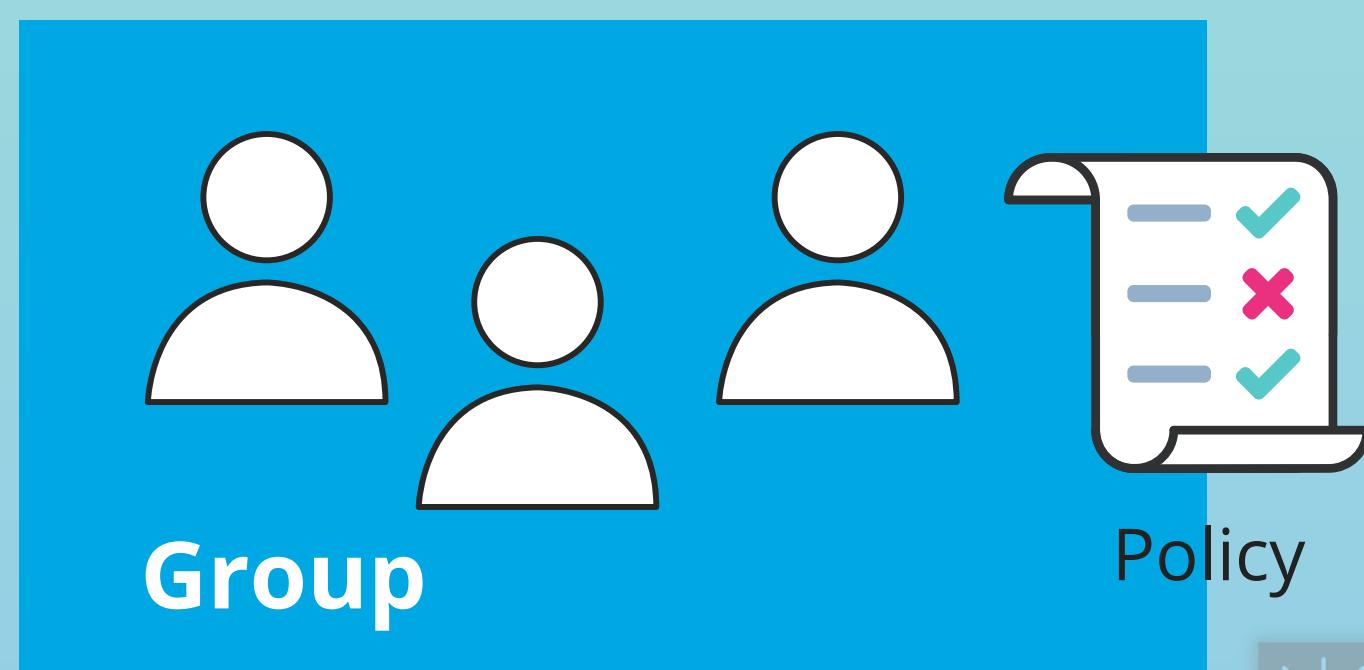
1. Human Users

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

- 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
- Also Policies cannot be assigned to AWS services directly
- So role is used to **grant AWS services access to other AWS services**



IAM User



EC2



ECS



EKS

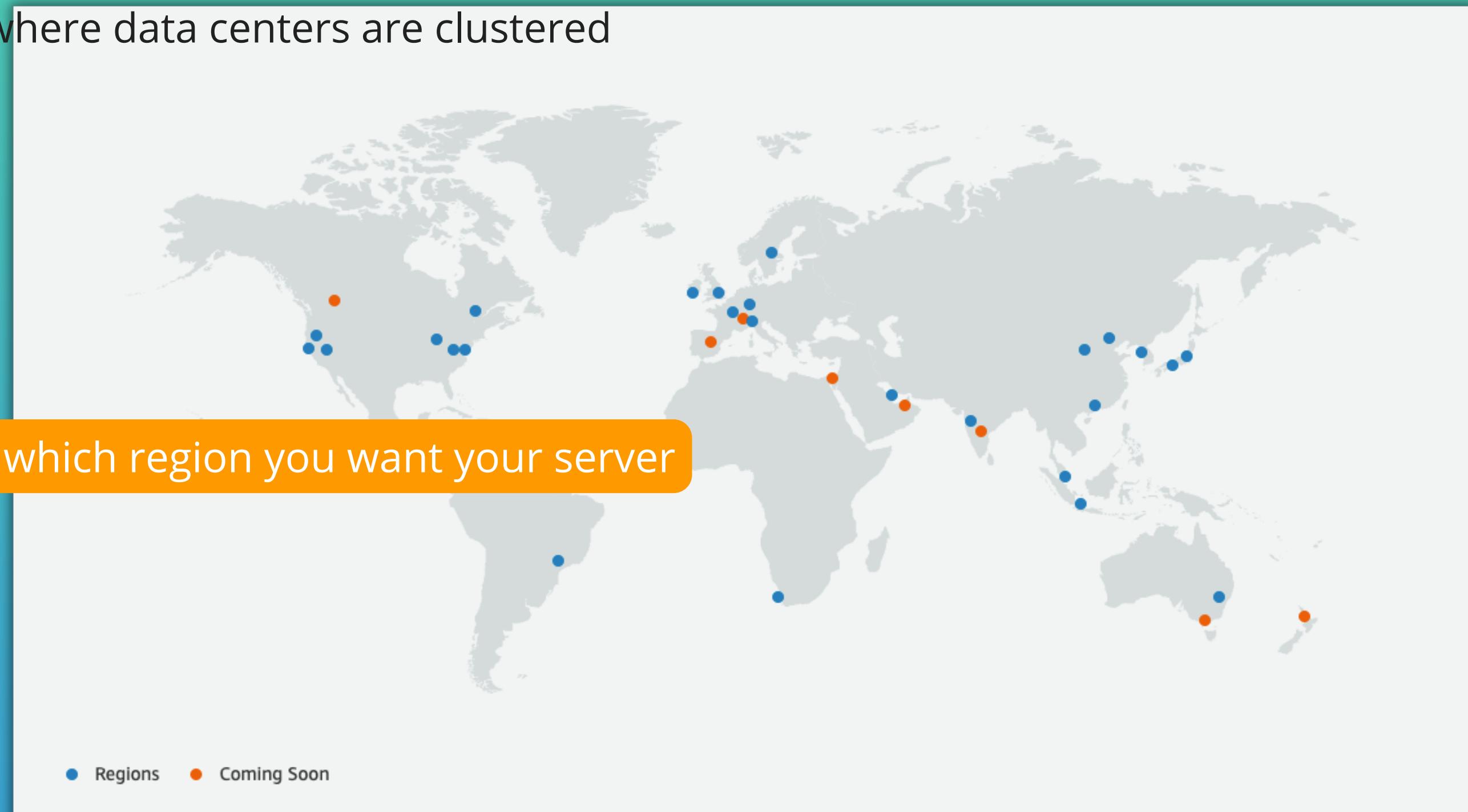
How to:

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

Regions & Availability Zones (AZ) - 1

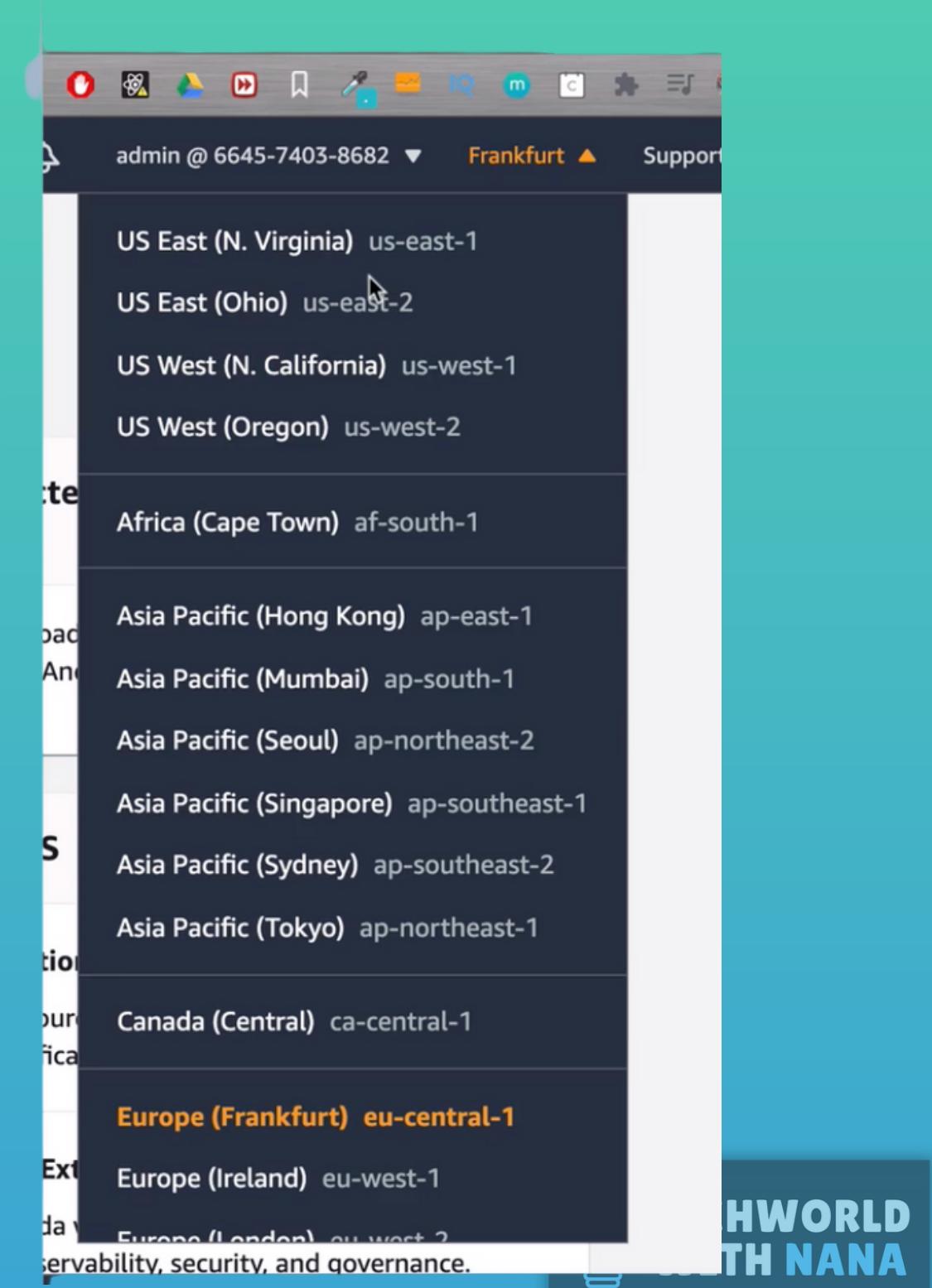
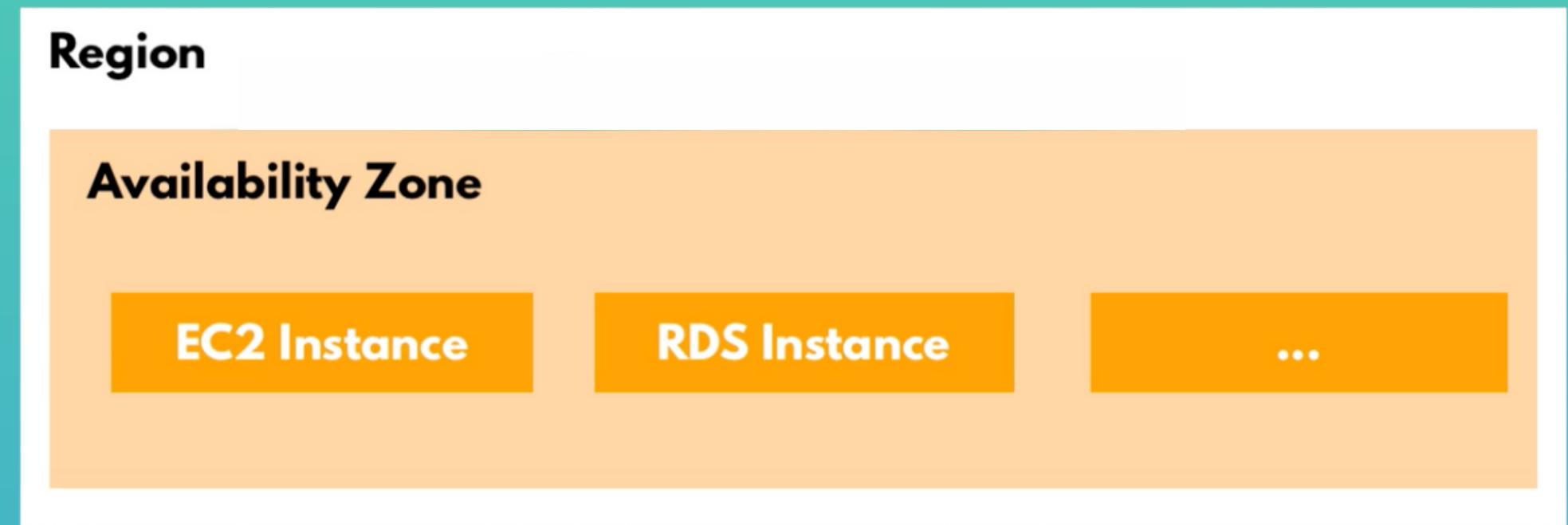
- Cloud providers have physical data centers
- Data centers all over the world
- **Region** = physical location, where data centers are clustered

You have to select, which region you want your server



Regions & Availability Zones (AZ) - 2

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



Virtual Private Cloud (VPC)

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



Region



- You have a VPC for each Region

- VPC **spans all the AZ (Subnet)** in that Region

- Multiple VPCs in different Regions

- VPC is like a **virtual representation of network infrastructure**: Server setup, network configuration

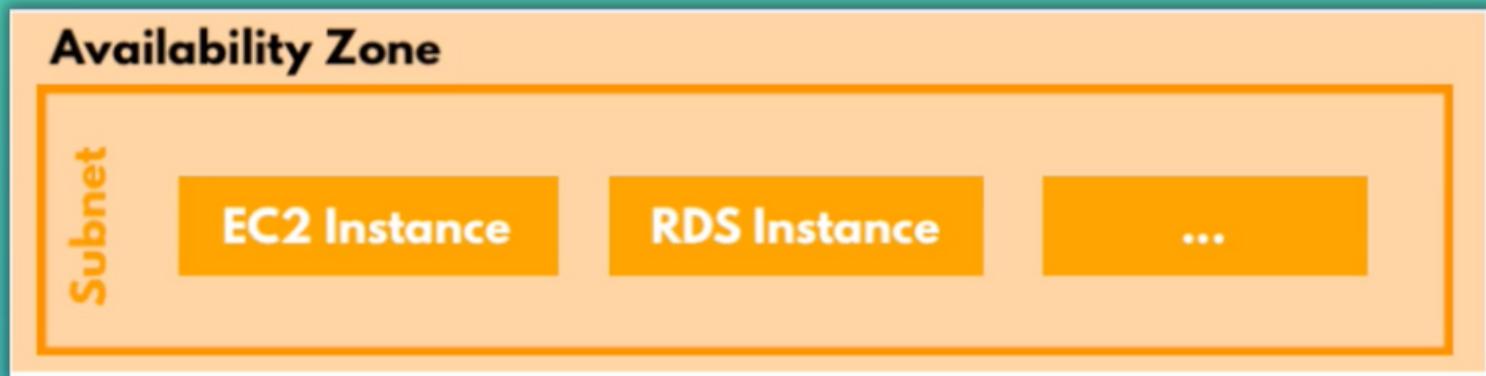
moved to cloud



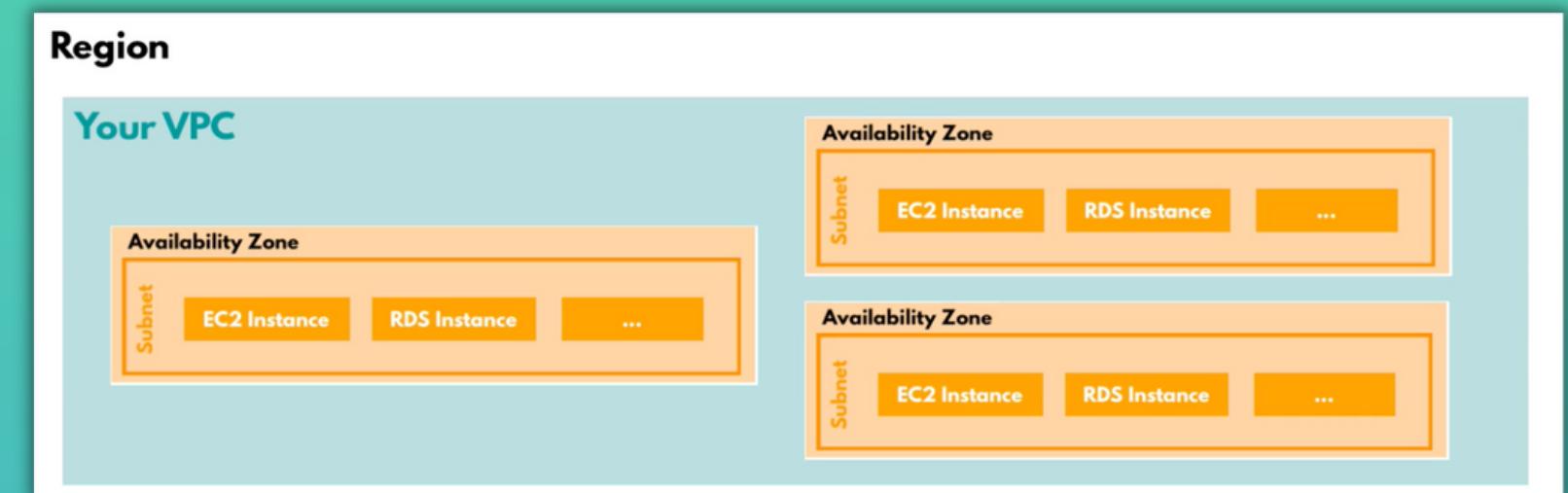
Your resources always have to run in a VPC!

Subnet - 1

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



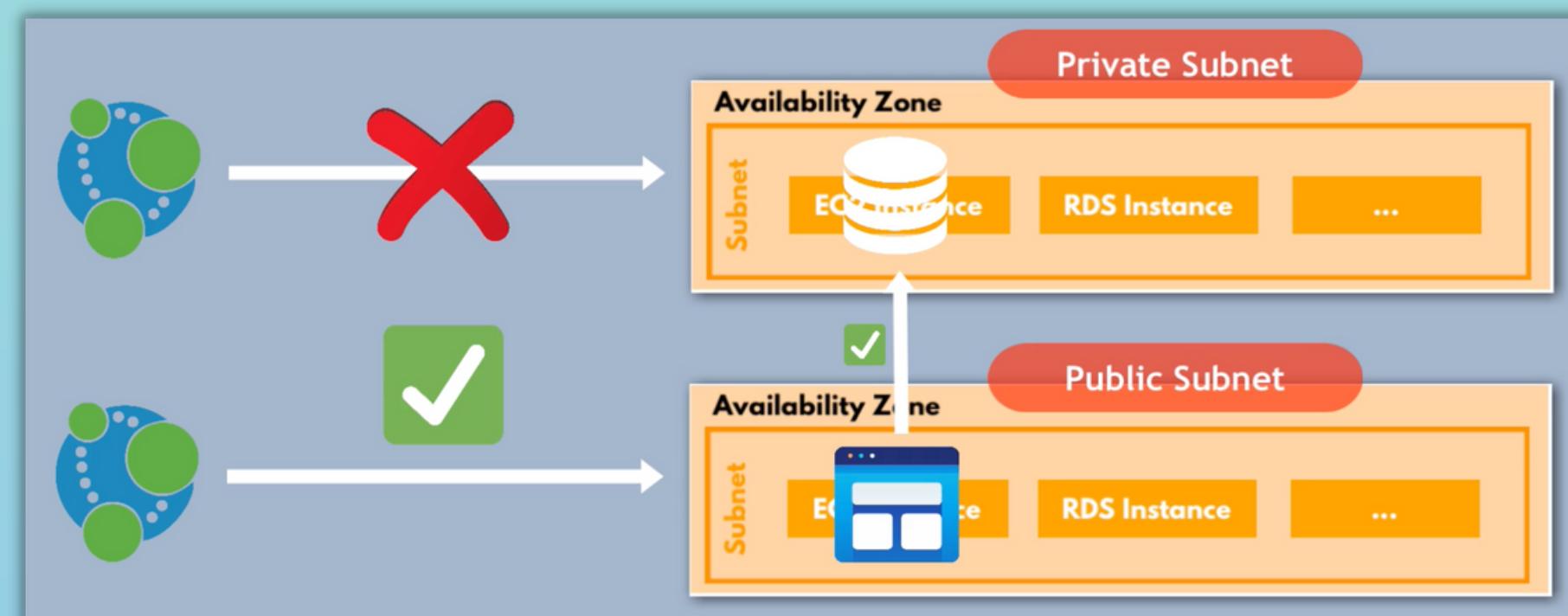
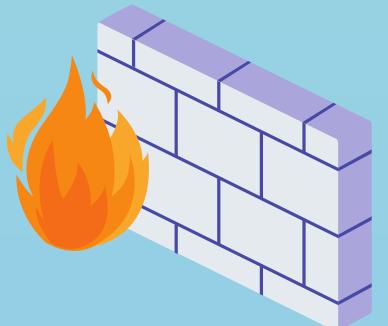
- You have a subnet for each Availability Zone:



Private and Public Subnets

- **Based on firewall configuration**

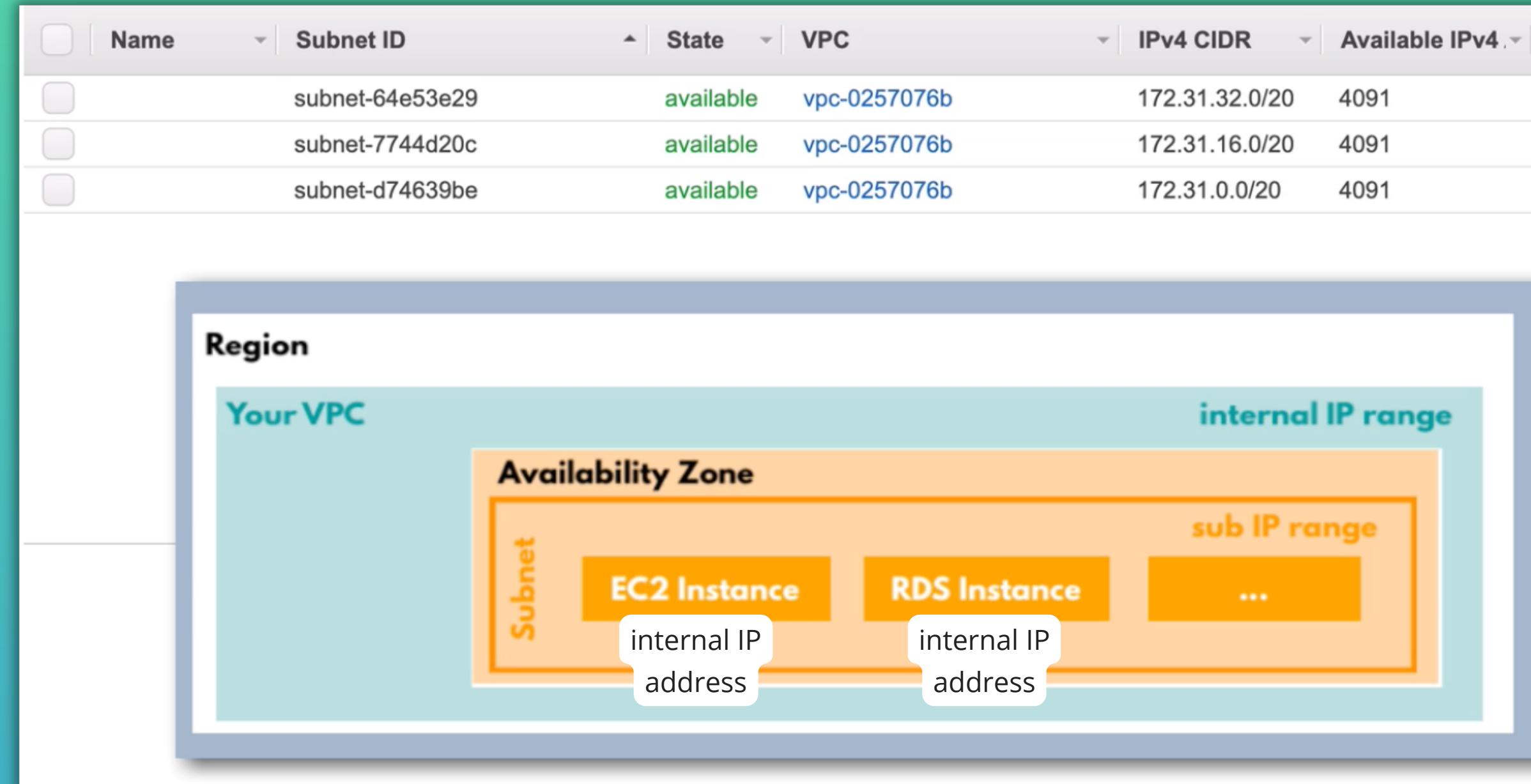
you can have a private and/or public subnet:



Subnet - 2

- A subnet has a default range of **internal IP addresses**
- When you create a new resource like EC2 instance then an IP address is assigned within this subnet's IP range
- For **communication inside the VPC**

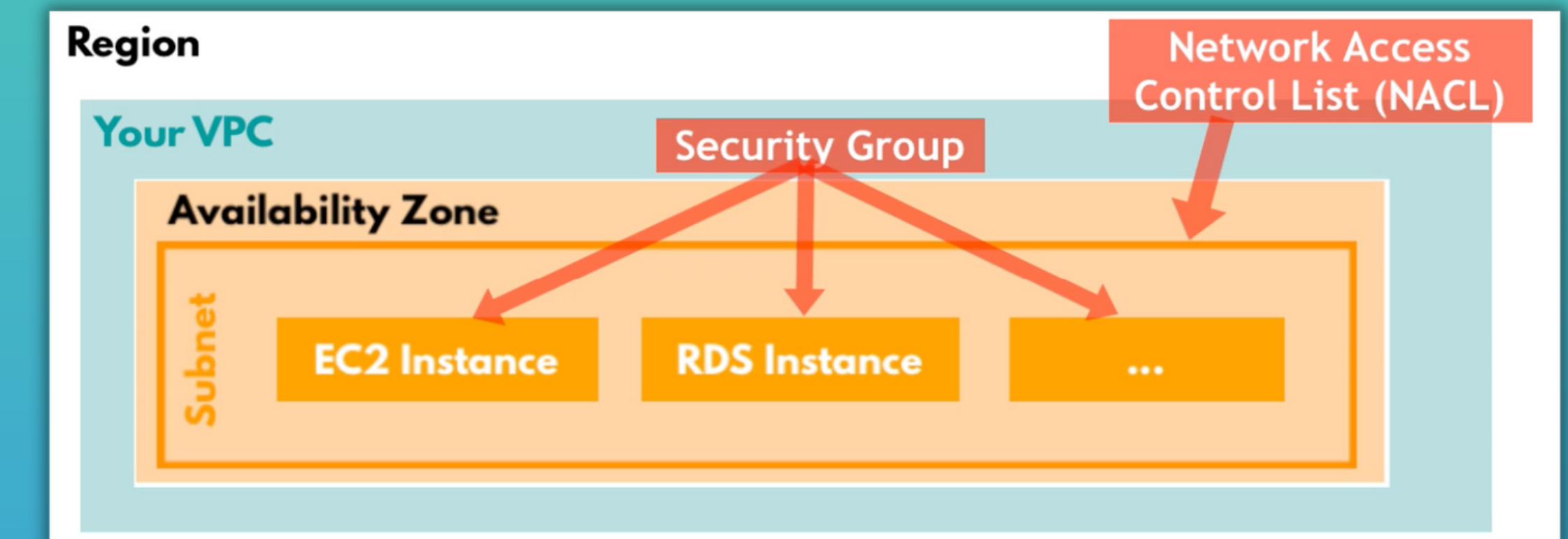
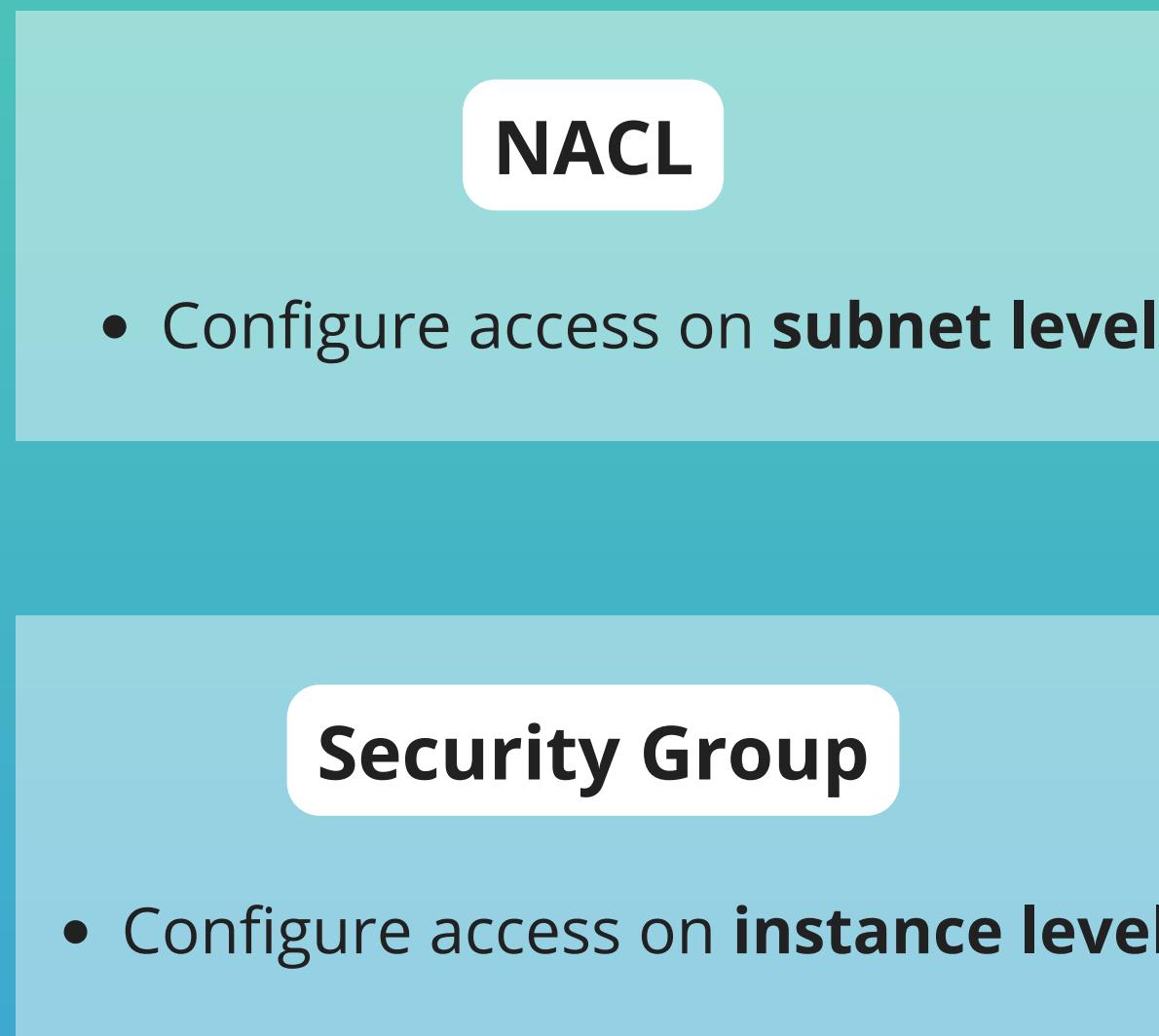
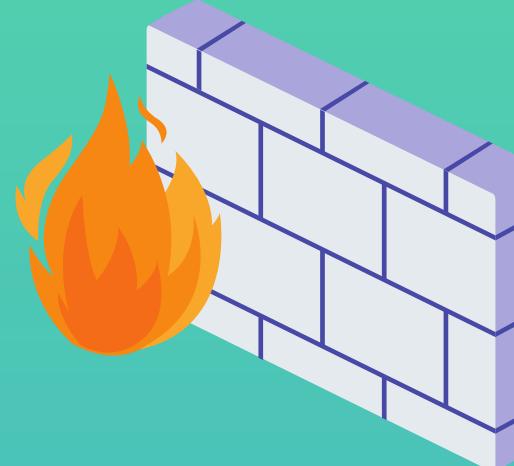
Internet Gateway



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

Subnets (3) Info

VPC **IPv4 CIDR**

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**

How to choose a CIDR Block:

Subnet Calculator

Network Address: 172.31.0.0 /16 Calculate

Input: 172.31.0.0/16	Input IP: 172.31.0.0	Input Long: 2887712768	/32 1 IPs
CIDR: 172.31.0.0/16	CIDR IP Range: 172.31.0.0 - 172.31.255.255	CIDR Long Range: 2887712768 - 2887778303	/31 2 IPs
IPs in Range: 65,536	Mask Bits: 16	Subnet Mask: 255.255.0.0	/30 4 IPs
			/29 8 IPs
			/28 16 IPs
			/27 32 IPs
			/26 64 IPs
			/25 128 IPs
			/24 256 IPs
			/23 512 IPs

Visual Subnet Calculator

Enter the network you wish to subnet:

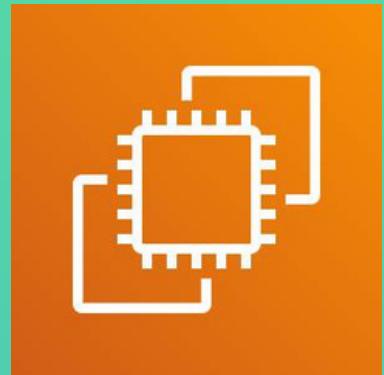
Network Address: 10.0.0.0 Mask bits: /16 Update Reset

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	Netmask	Range of addresses	Useable IPs	Hosts	Divide	Join
10.0.0.0/18	255.255.192.0	10.0.0.0 - 10.0.63.255	10.0.0.1 - 10.0.63.254	16382	Divide	/18
10.0.64.0/18	255.255.192.0	10.0.64.0 - 10.0.127.255	10.0.64.1 - 10.0.127.254	16382	Divide	/18
10.0.128.0/18	255.255.192.0	10.0.128.0 - 10.0.191.255	10.0.128.1 - 10.0.191.254	16382	Divide	/18
10.0.192.0/18	255.255.192.0	10.0.192.0 - 10.0.255.255	10.0.192.1 - 10.0.255.254	16382	Divide	/18

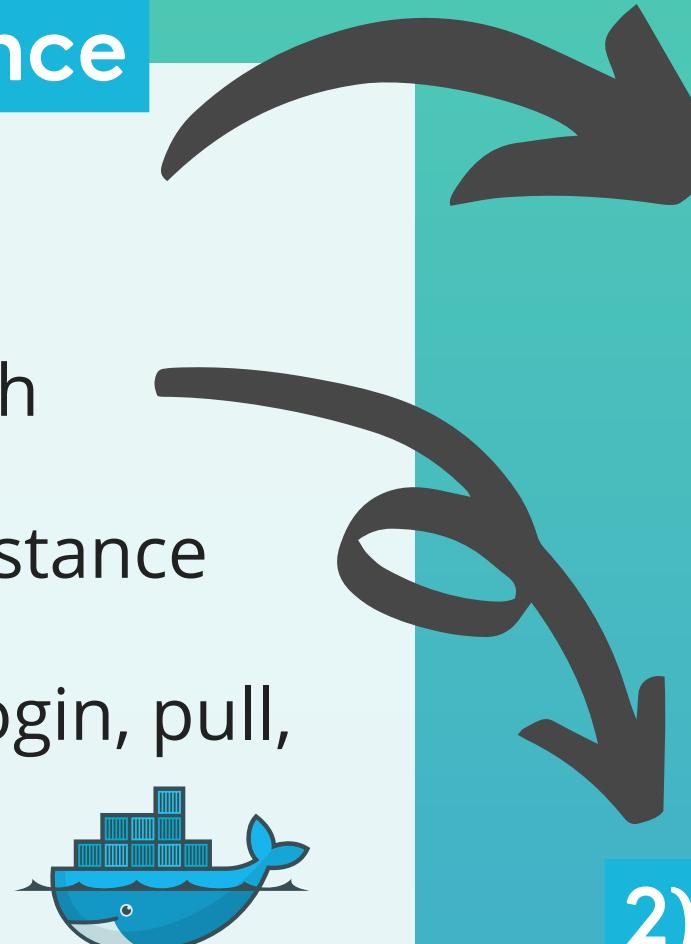
Elastic Compute Cloud (EC2)



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

Steps to deploy a web application on 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 Firewall to access application externally from the browser



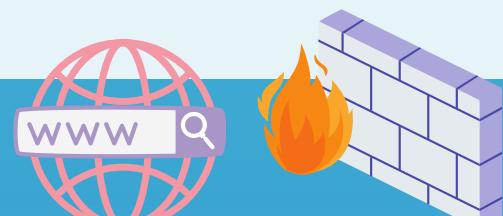
1) Steps to launch EC2 instance

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



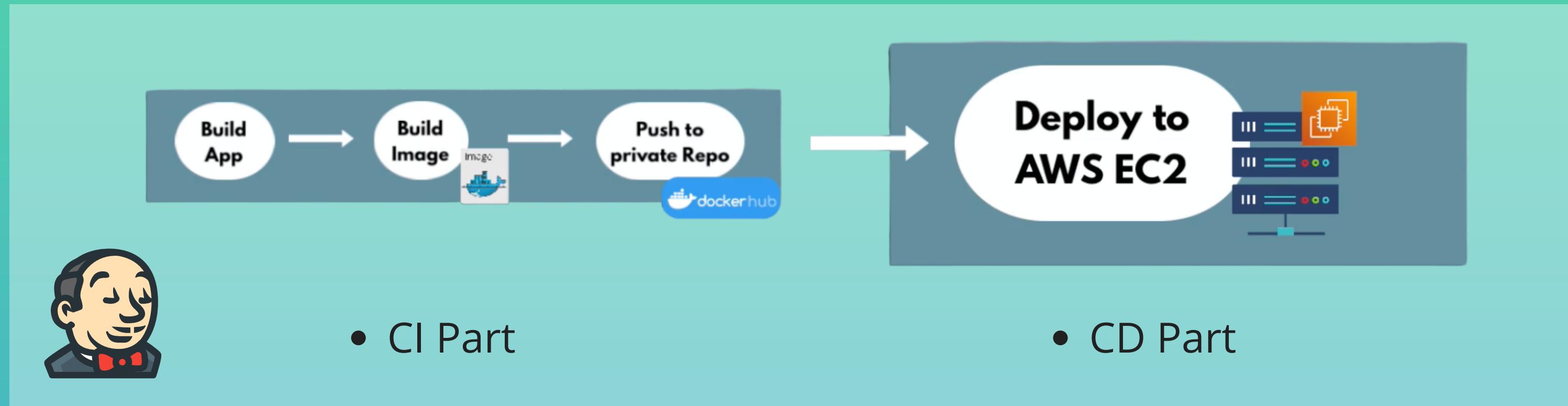
2) Connect to EC2 server via ssh using downloaded private ssh key:

```
[ \W]$ ssh -i ~/.ssh/docker-server.pem ec2-user@15.236.202.138 ]
```



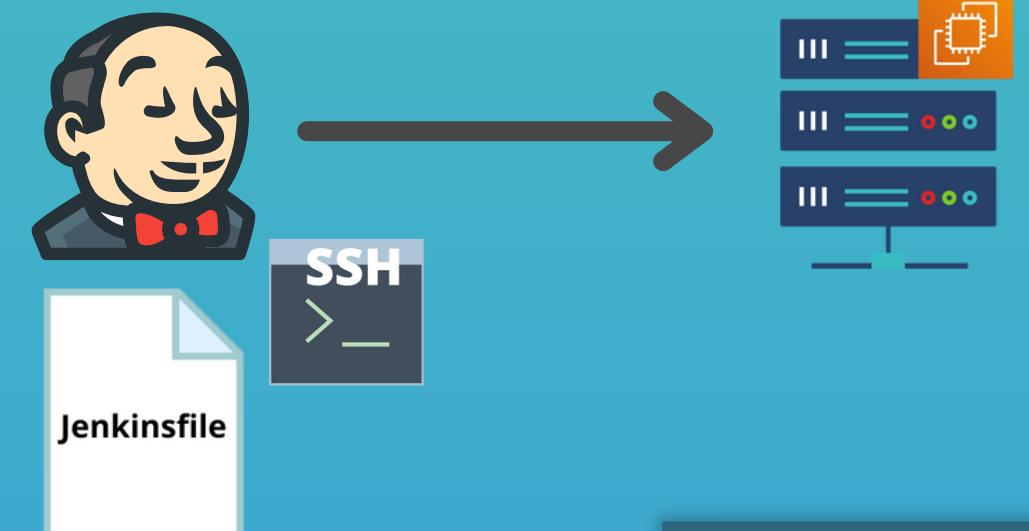
CD part of CI/CD - Deploy to EC2 from Jenkins

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 plugin manager interface. At the top, there are tabs for 'Updates', 'Available' (which is selected), 'Installed', and 'Advanced'. Below these are sections for 'Install' and 'Name'. Under 'Name', there are sections for 'Publish Over SSH' (with 'Build Tools' and 'Artifact Uploaders' options) and 'Send build artifacts over SSH'. A note below states: 'This plugin is up for adoption! We are looking for new maintainers. [Plugin](#) initiative for more information.' At the bottom, there is a section for 'SSH Agent' with a checked checkbox and a descriptive text: 'This plugin allows you to provide SSH credentials to builds via a s...'. There is also a 'Jenkinsfile' icon.

2) Configure credential

The screenshot shows the 'Global credentials (unrestricted)' configuration page. It lists two credentials: 'my-pipeline' (Kind: Username with password) and 'ec2-server-key' (Kind: SSH Username with private key). The 'ec2-server-key' row is highlighted with a red underline. Below the table, there is a note: 'Icon: S M L'.

ID	Name	Kind	Descr
my-pipeline	my-pipeline/*****	Username with password	
ec2-server-key	ec2-user	SSH Username with private key	

3) Use credential in "deploy" stage in Jenkinsfile

The screenshot shows a portion of a Jenkinsfile. It includes a 'stage' block for 'deploy' with a 'script' block containing a 'def' block for 'dockerCmd' and an 'sshagent' block for 'ec2-server-key'. The 'sshagent' block contains a 'sh' command to run an SSH command on an EC2 instance.

```
stage('deploy') {
    steps {
        script {
            def dockerCmd = 'docker run -p 3080:3080 -d nanajanashia/demo-app:1.0'
            sshagent(['ec2-server-key']) {
                sh "ssh -o StrictHostKeyChecking=no ec2-user@35.180.251.121 ${dockerCmd}"
            }
        }
    }
}
```

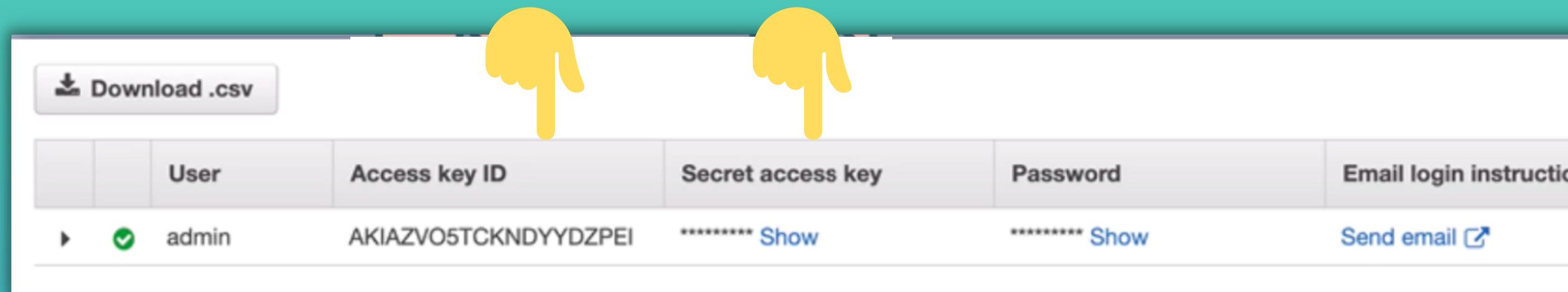
Introduction to AWS CLI

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:

```
[ \W]$ aws configure
AWS Access Key ID [None]: AKIAZV05TCKNDYYDZPEI
AWS Secret Access Key [None]: dJRnI+tGBxT/sHZZT5hxadJ1C6V9Th7Z32qAhHUK
Default region name [None]: eu-west-3
Default output format [None]: json
[ \W]$ ls ~/.aws
config      credentials
[ \W]$
```

Configuration is stored in your home directory under /.aws

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

- Launch EC2 Instance via AWS CLI

aws <command> <subcommand> [options and parameters]



AWS service



which operation?

IAM

list-groups

get-user

list-policies

...

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

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 a IaC tool
- Declarative - describing the desired state

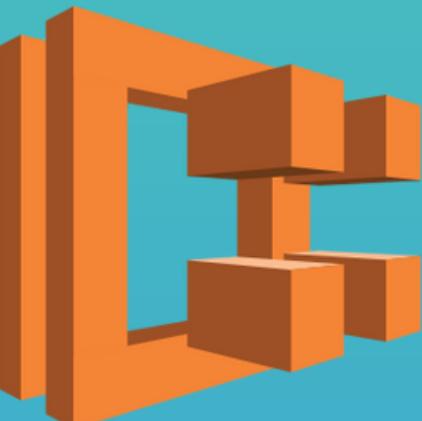
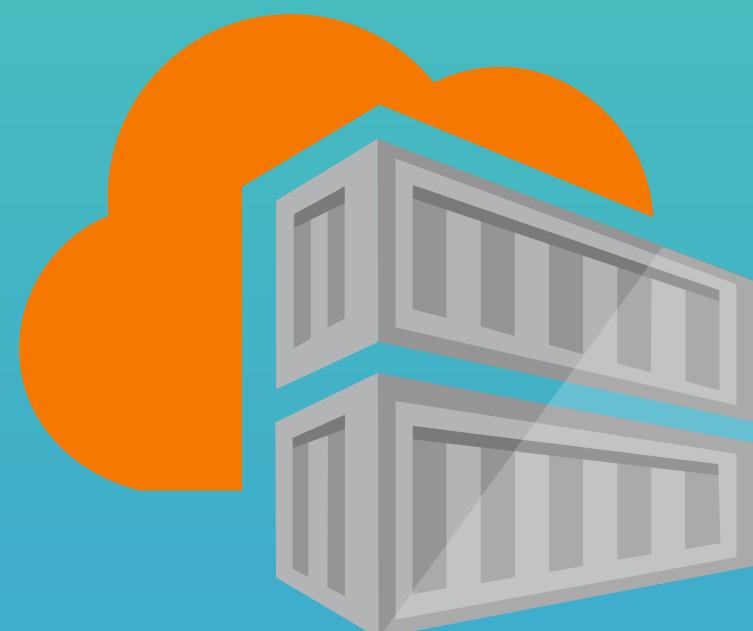


HashiCorp
Terraform

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.

