AWS Services

Key Takeaways

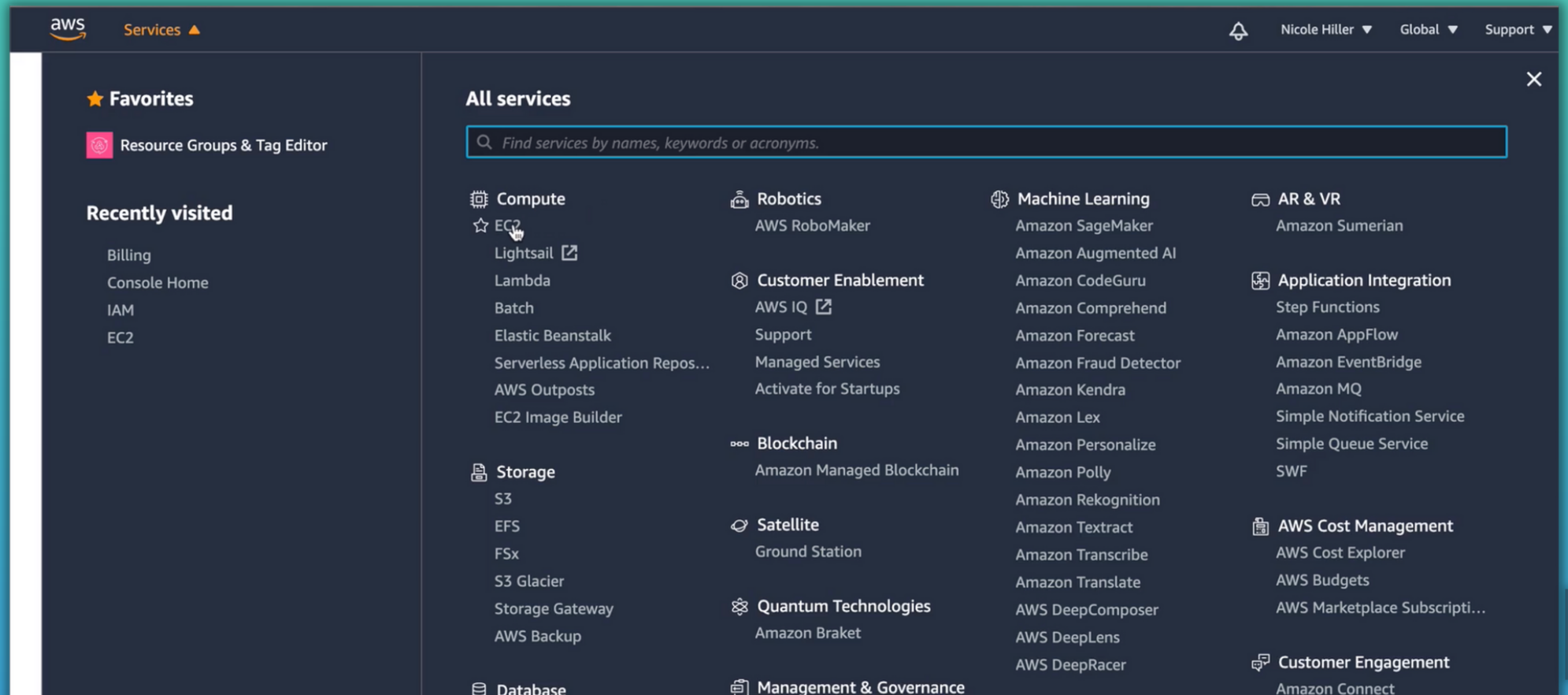
TECHWORLD WITH NANA

Introduction to AWS

TECHWORLD WITH NANA

AWS stands for **Amazon Web Services**

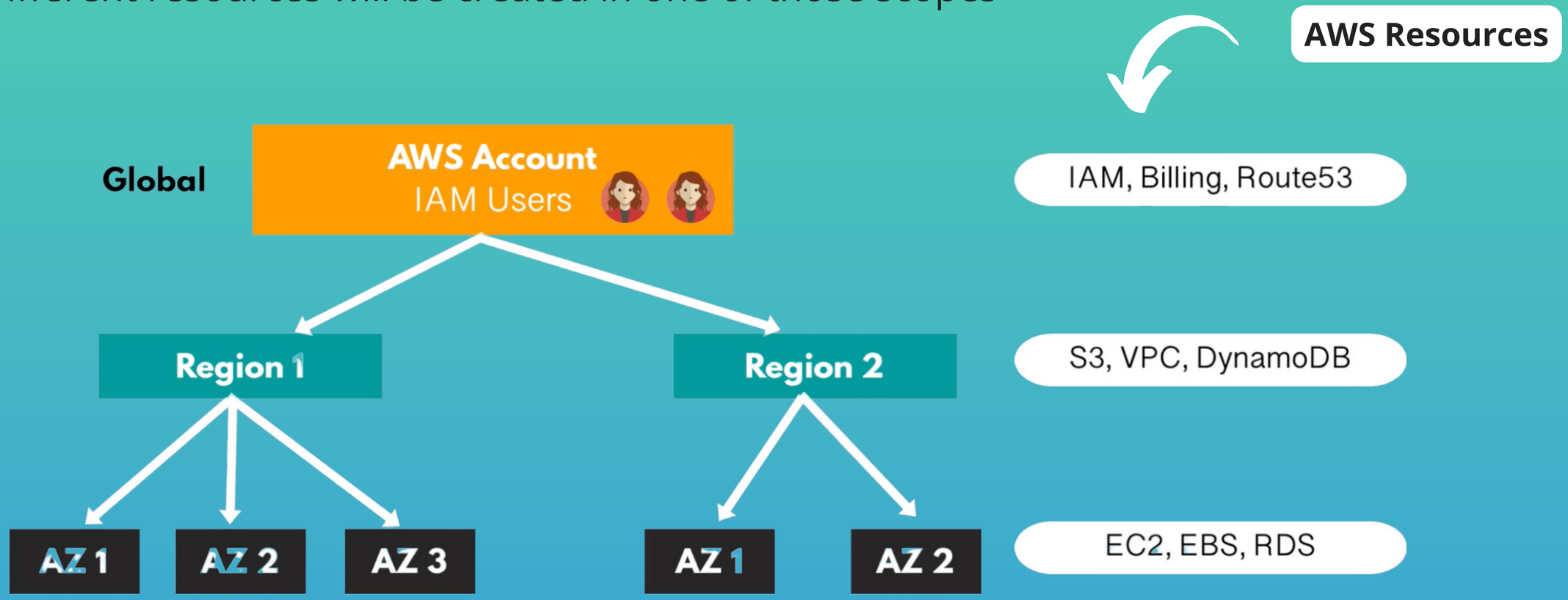
Currently most popular cloud platform, with a huge **collection of services**:



TECHWORLD WITH NANA

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

Add a payment method

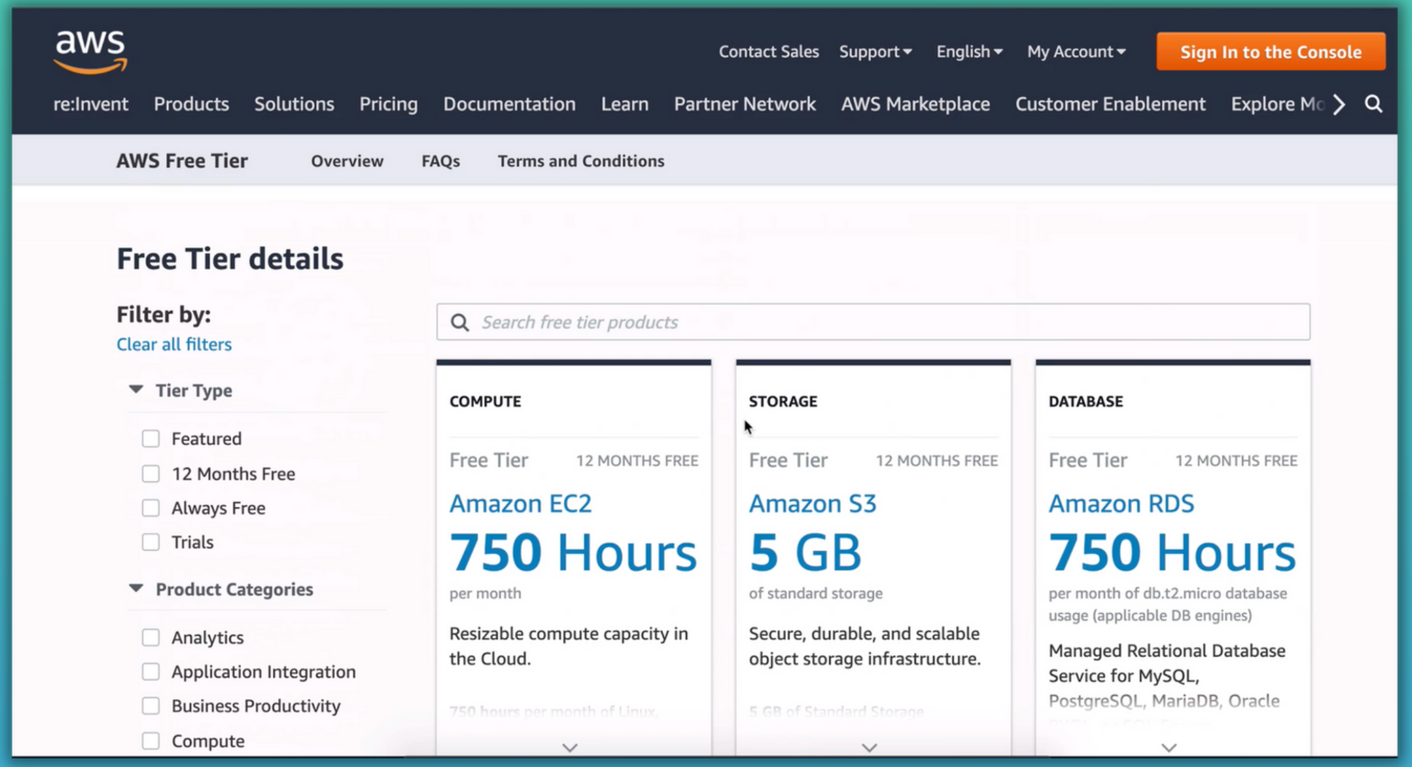
On the **Payment Information** page, enter the information about your payment method, and then choose **Verify and Add**.

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**



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

Delete Resources that you don't need anymore

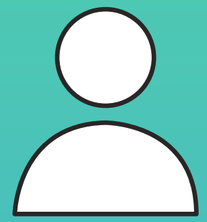
TECHWORLD WITH NANA

Core AWS Services

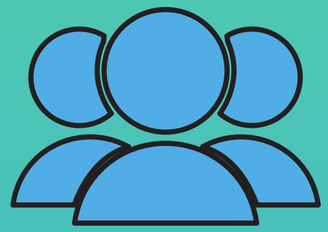
TECHWORLD
WITH NANA

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)



User

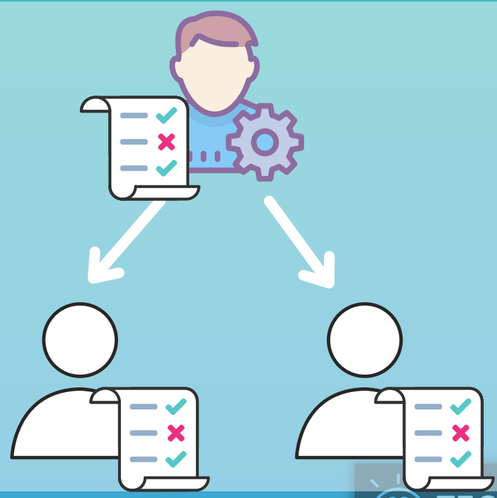


Group

Policy

- 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

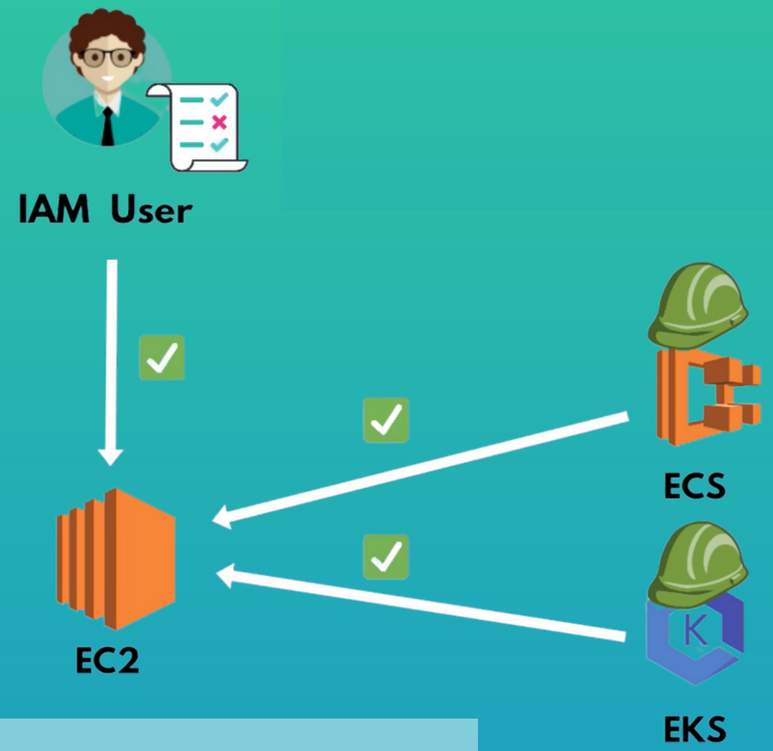
Groups

- 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**



How to:

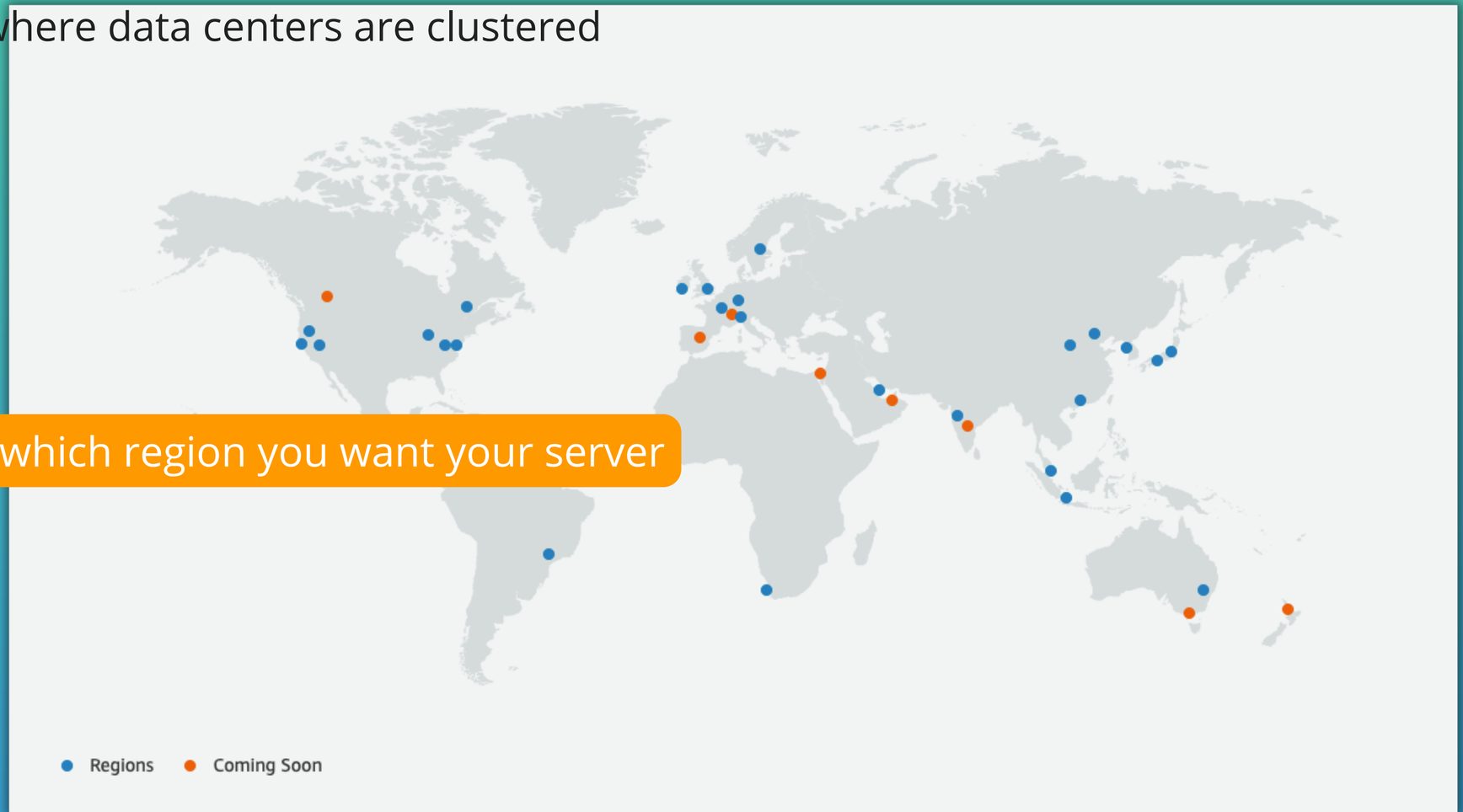
1. Create IAM Role
2. Assign Role AWS Service
3. Attach Policies to that Role

TECHWORLD WITH NANA

Regions & Availability Zones (AZ) - 1

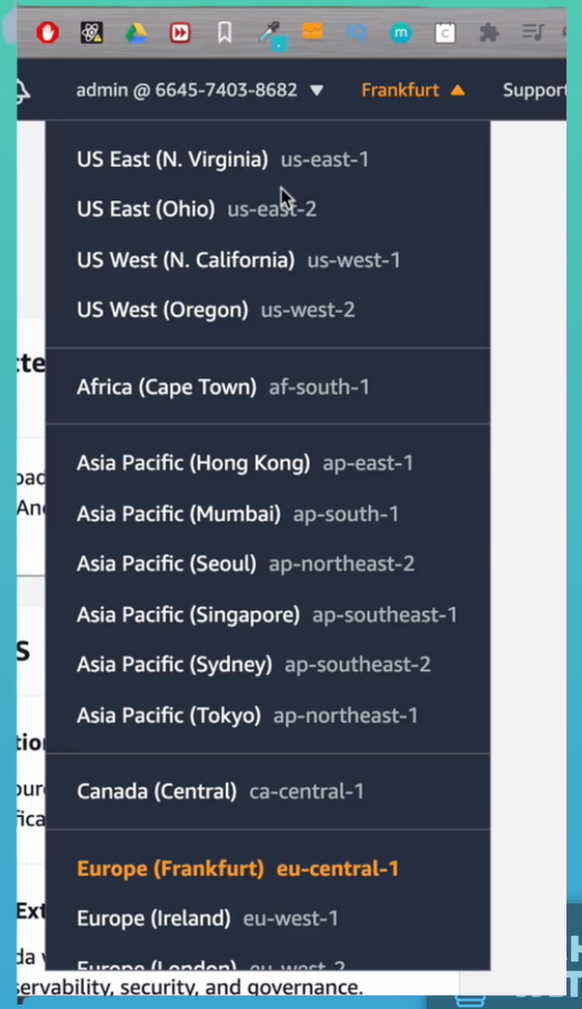
- Cloud providers have physical data centers
- Data centers all over the world
- **Region** = physical location, where data centers are clustered

You have to select, which region you want your server



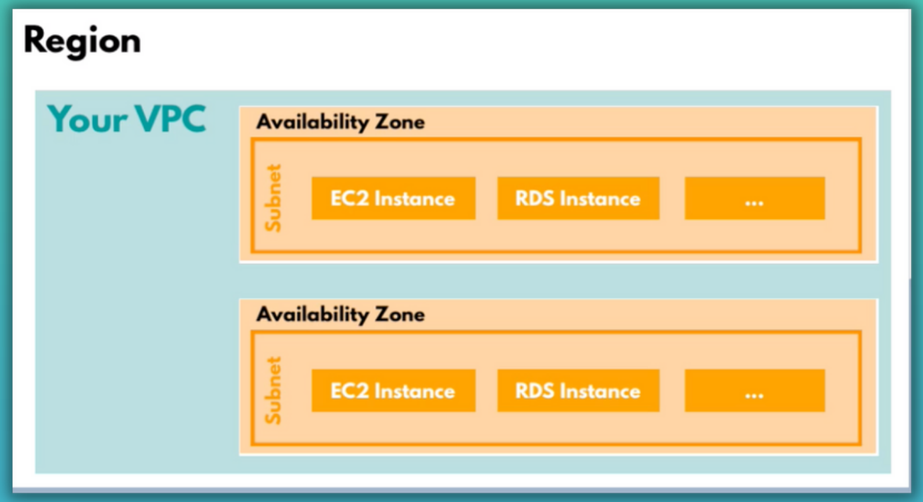
Regions & Availability Zones (AZ) - 2

- 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



- You have a VPC for each Region
- VPC **spans all the AZ (Subnet)** in that Region
- Multiple VPCs in different Regions

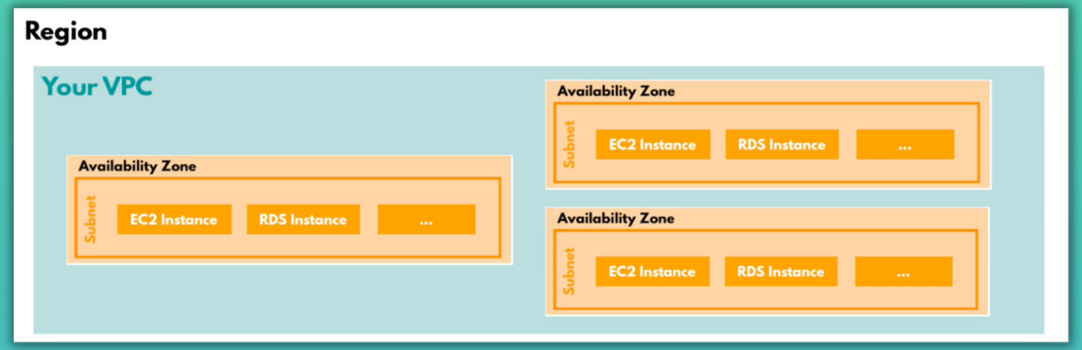
Your resources always have to run in a VPC!

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

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:

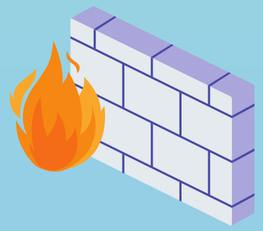


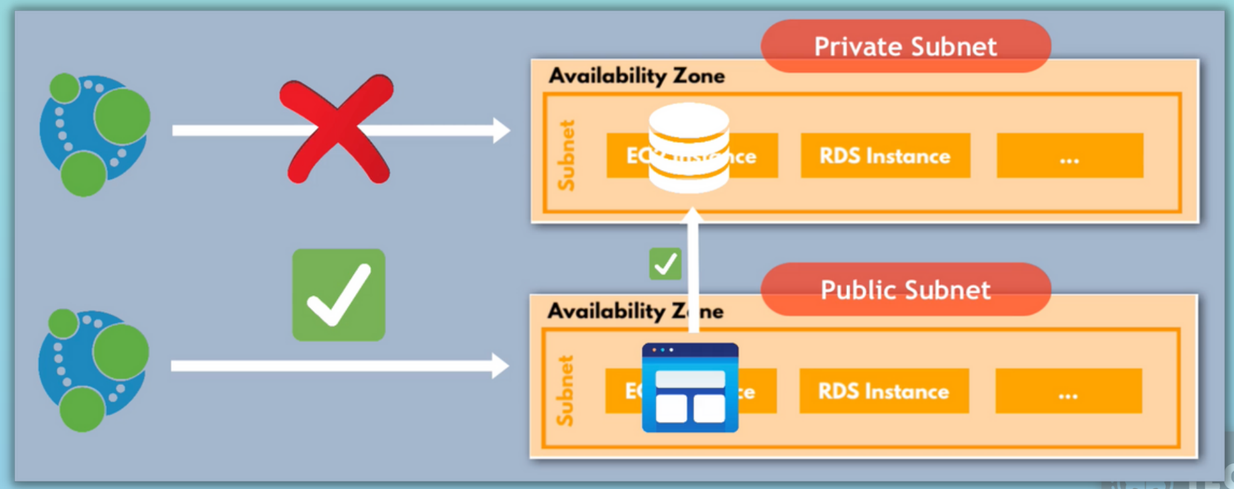
Private and **Public** Subnets

- **Based on firewall configuration**

  you can have a private and/or

  public subnet:

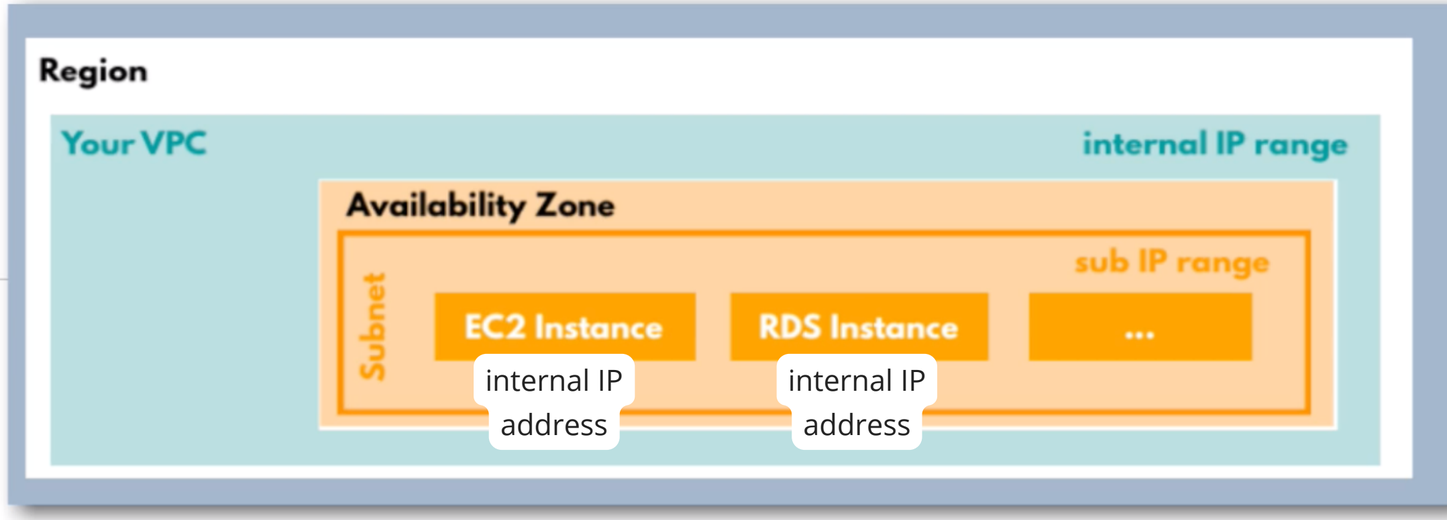




Subnet - 2

- 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**

Name	Subnet ID	State	VPC	IPv4 CIDR	Available IPv4
	subnet-64e53e29	available	vpc-0257076b	172.31.32.0/20	4091
	subnet-7744d20c	available	vpc-0257076b	172.31.16.0/20	4091
	subnet-d74639be	available	vpc-0257076b	172.31.0.0/20	4091

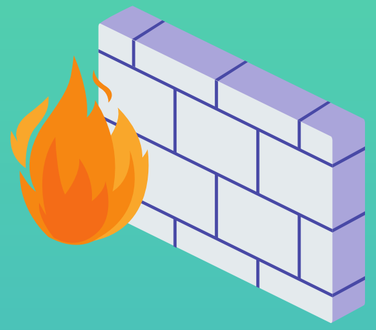


Internet Gateway

- 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

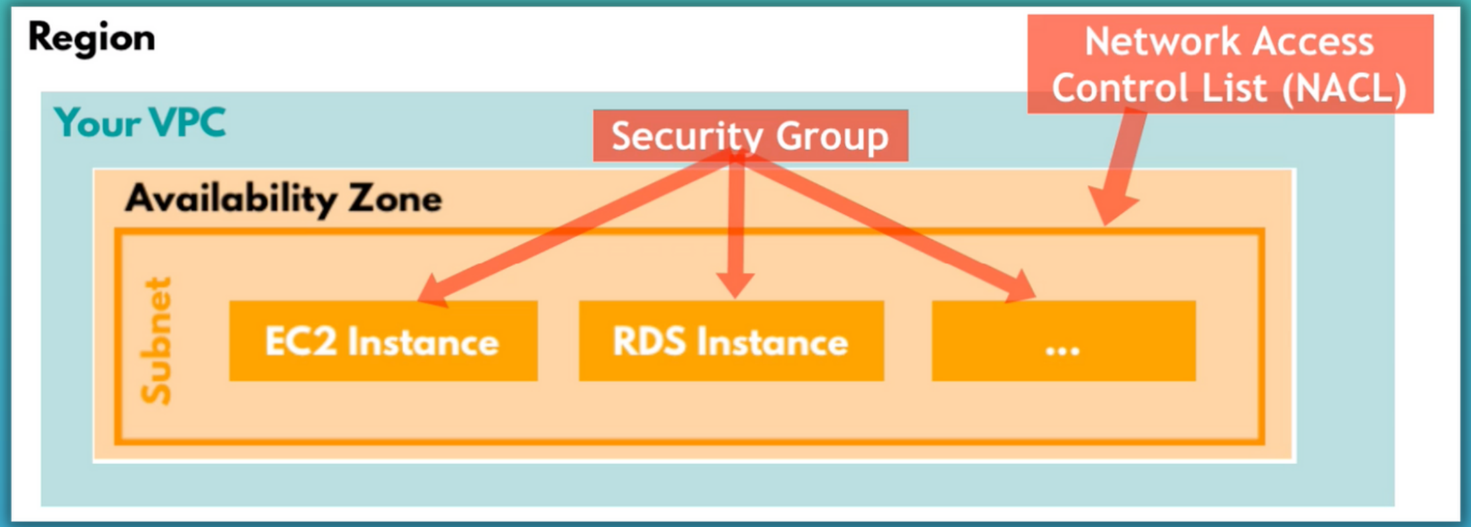


NACL

- Configure access on **subnet level**

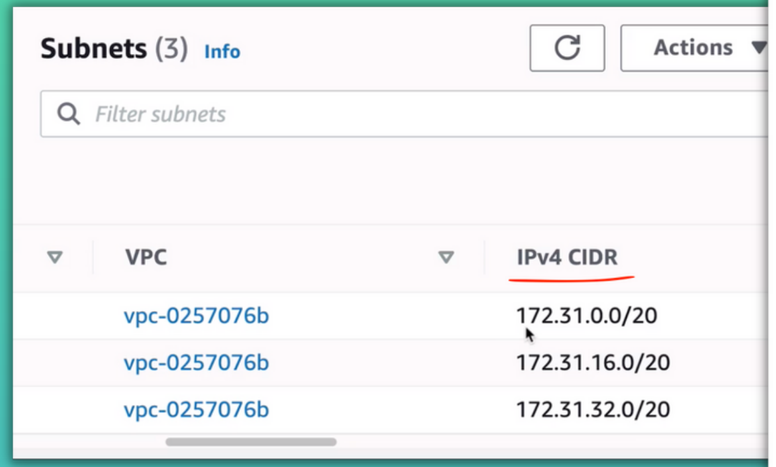
Security Group

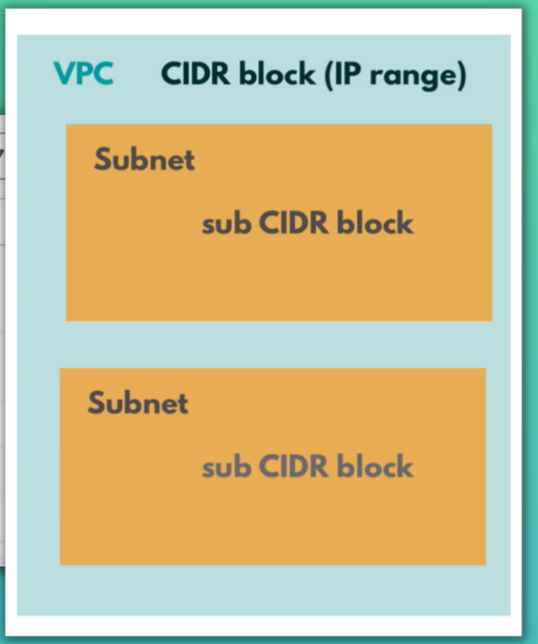
- Configure access on **instance level**



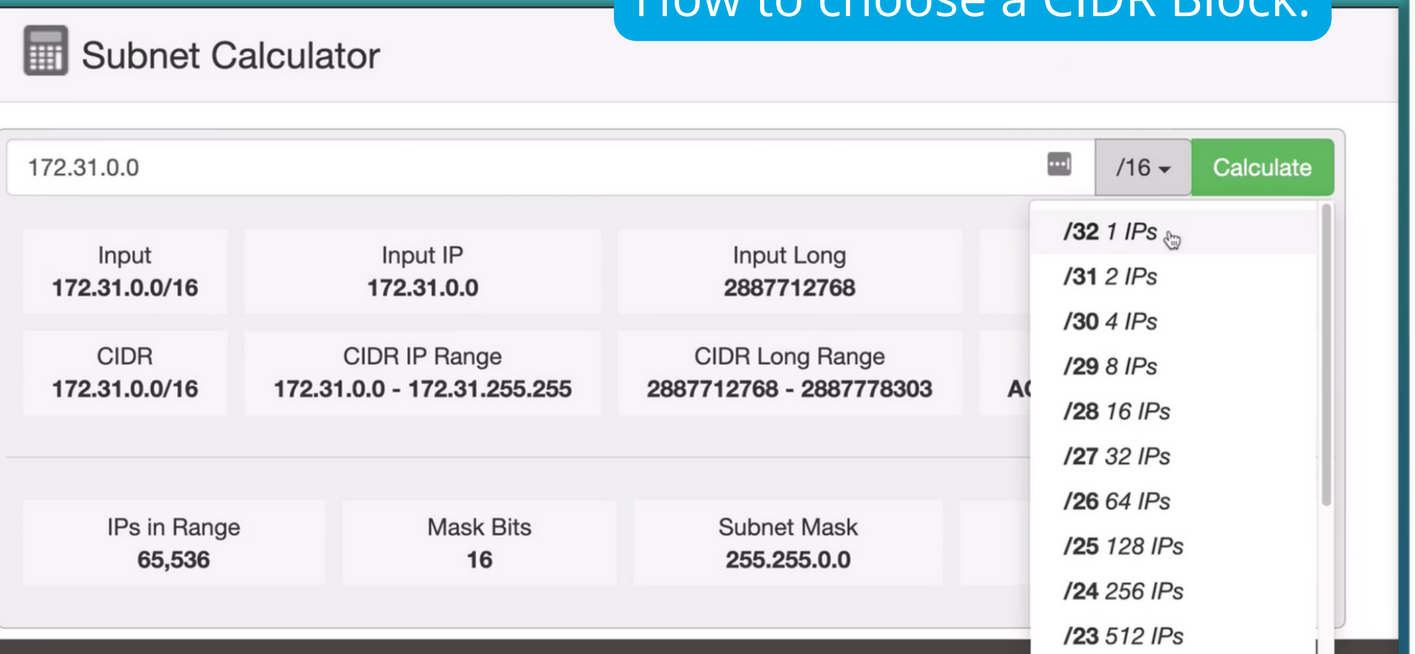
CIDR Block

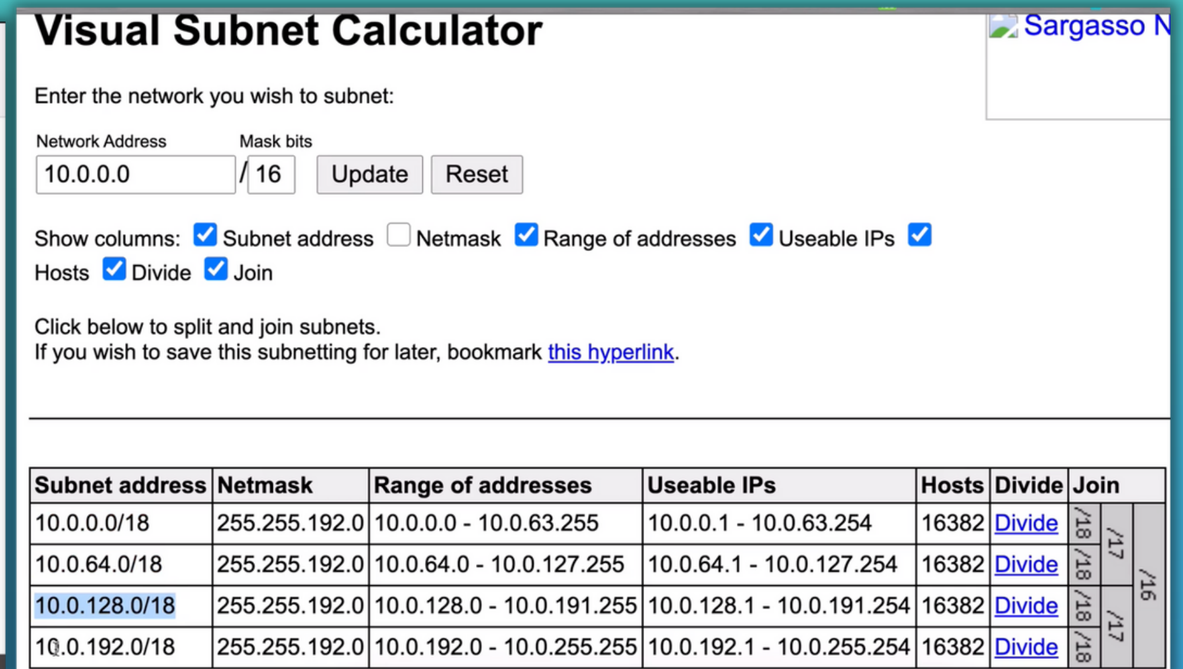
- When you create a subnet, you **specify the IPv4 CIDR block for the subnet**, which is a subset of the VPC CIDR block
- Defines a range of IP addresses
- CIDR = Classless Inter-Domain Routing





How to choose a CIDR Block:



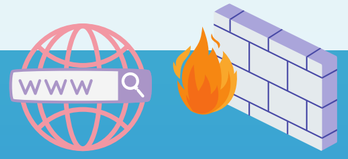


Elastic Compute Cloud (EC2)

- 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]$ s$h -i ~/.ssh/docker-server.pem ec2-user@15.236.202.138
```

TECHWORLD WITH NANA

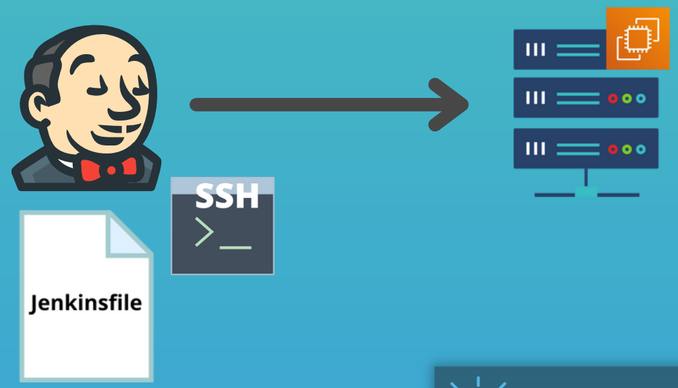
CD part of CI/CD - Deploy to EC2 from Jenkins

Deploy to EC2 Instance from Jenkins - 1

- CI Part
- CD Part

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

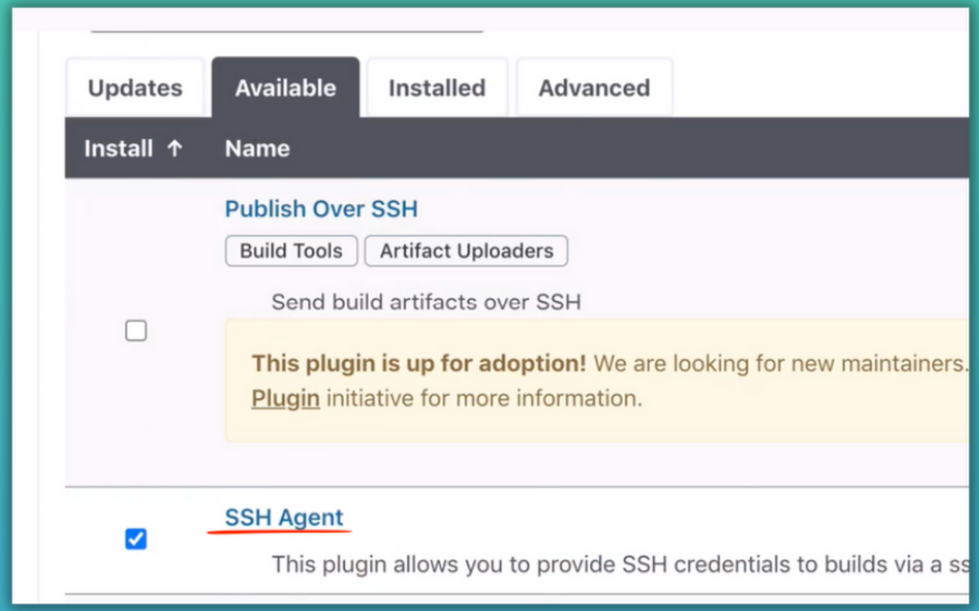


TECHWORLD WITH NANA

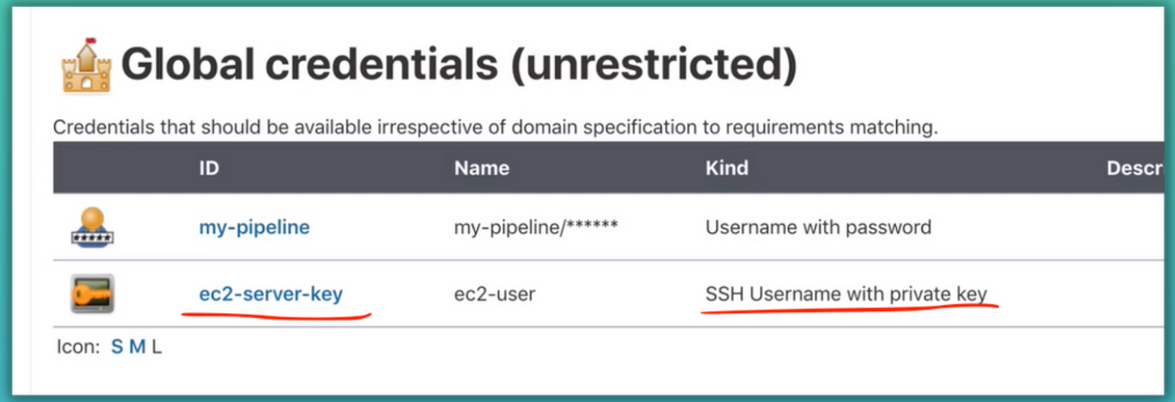
Deploy to EC2 Instance from Jenkins - 2

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

1) Install SSH Agent Plugin:



2) Configure credential



3) Use credential in "deploy" stage in Jenkinsfile

Jenkinsfile

```
age('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

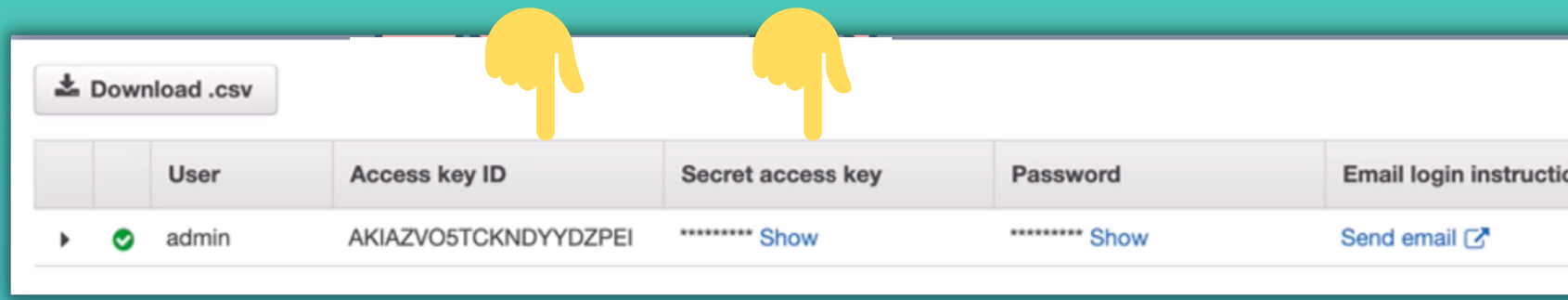
TECHWORLD
WITH NANA

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]: AKIAZVO5TCKNDYYDZPEI
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]$
```

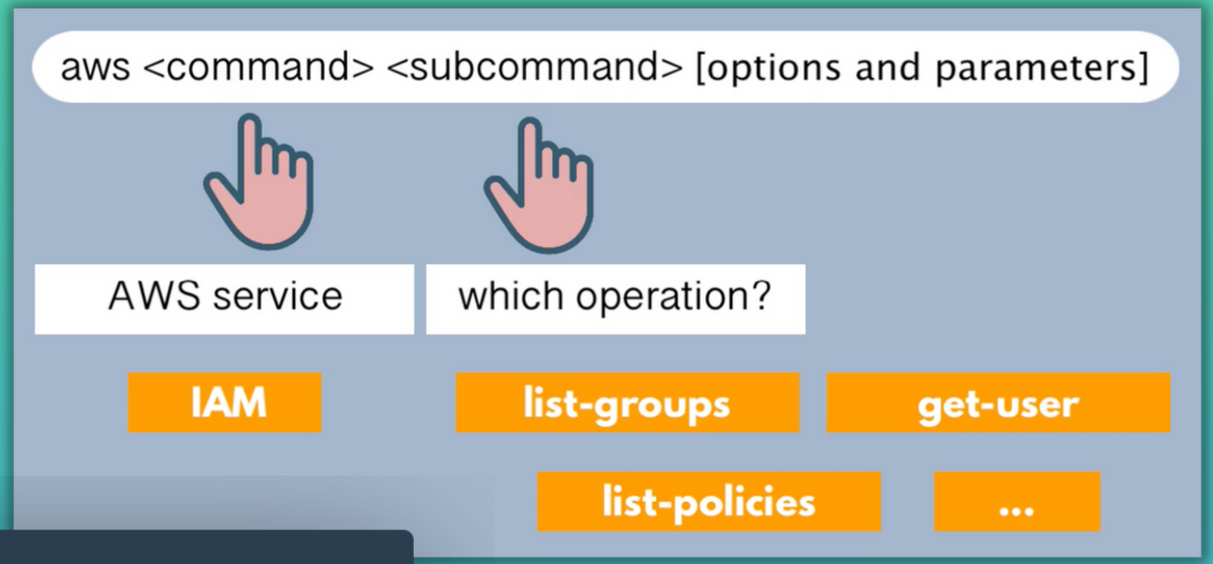
Configuration is stored in your home directory under /.aws

TECHWORLD WITH NANA

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

TECHWORLD WITH NANA

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

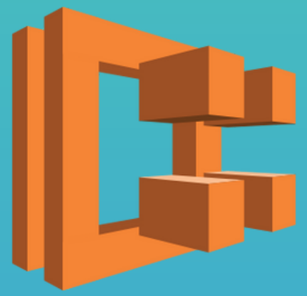
HashiCorp Terraform

TECHWORLD WITH NANA

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

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.