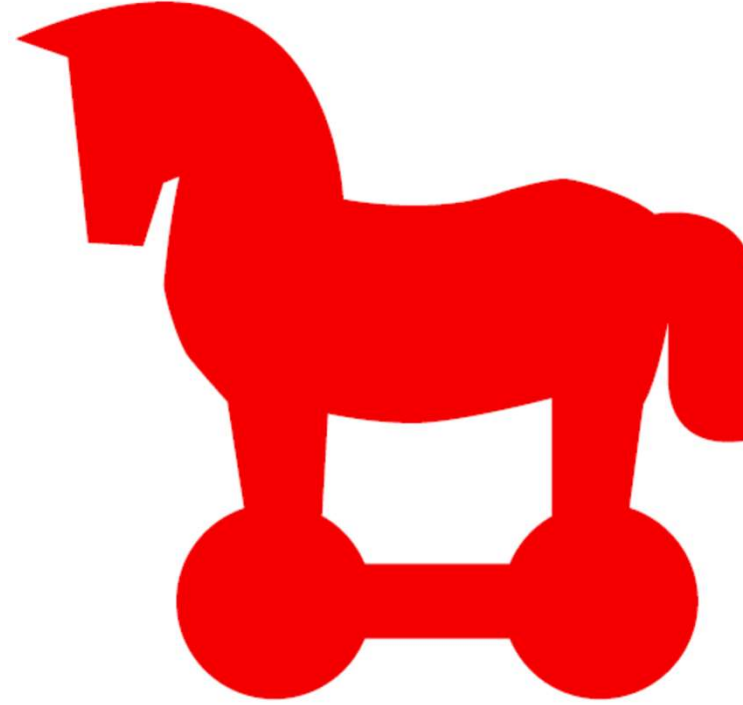




**GANTEK**  
**INTEGRATING FUTURE**



**GANTEK**  
**ACADEMY**

**40 YILLIK TECRÜBE İLE**

# AWS (Amazon Web Services)

- Amazon Web Service (AWS) firmalara, IT altyapısını bulut bilişim de denilen web hizmetleri adıyla sunmaya başladı.
- AWS bölgeler (region) olarak organize olmuş erişilebilirlik alanları olarak organize olmuştur. 2023 Mayıs itibariyle 31 coğrafi bölgede 99 erişilebilirlik alanı vardır.



# AWS (Amazon Web Services)

- Amazon Web Service (AWS) erişilebilirlik alanı (availability zone), sunucuların ve ağ alt yapısının kurulu olduğu bir veri merkezi anlamına gelir.
- Yerel alanlar (local zones), bir kısım müşterinin buluta erişim süresini 10ms'nin altına indirmek için, kendi İnternet bağlantısı olan ve erişilebilirlik alanları içinde bulunan ayrı veri merkezleridir.

## Kuzey Amerika

### ABD Batı (Oregon) Bölgesi

Erişilebilirlik Alanları: 4

Açılış: 2011

7 Local Zone

Açılış: 2019

### ABD Doğu (Kuzey Virginia) Bölgesi

Erişilebilirlik Alanları: 6

Açılış: 2006

Local Zone'lar: 10

Açılış: 2020

### ABD Batı (Kuzey Kaliforniya) Bölgesi

Erişilebilirlik Alanları: 3\*

Açılış: 2009

### ABD Doğu (Ohio) Bölgesi

Erişilebilirlik Alanları: 3

Açılış: 2016

### Kanada (Orta) Bölgesi\*\*

Erişilebilirlik Alanları: 3

Açılış: 2016

### GovCloud (ABD Batı) Bölgesi

Erişilebilirlik Alanları: 3

Açılış: 2011

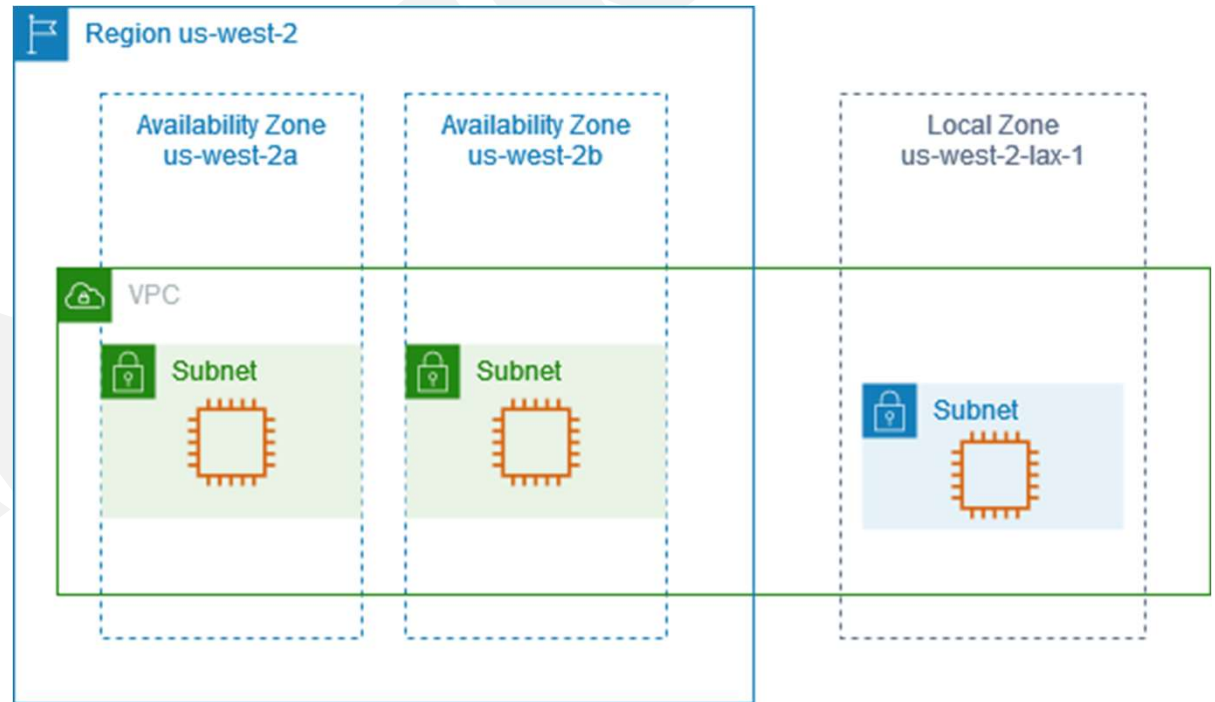
### GovCloud (ABD Doğu) Bölgesi

Erişilebilirlik Alanları: 3

Açılış: 2018

# AWS (Amazon Web Services)

- **us-west-2** bölgesinde iki tane erişilebilirlik alanı olsun. Bunların isimleri **us-west-2a** ve **us-west-2b**.
- Bunların içinde bulunan bilgisayar, ağ cihazları, vb sistemlerin bir kısmı VPC (Virtual Private Cloud) içine alınıp, hızlı İnternet bağlantılarıyla Los Angeles'e bağlanabiliyor.
- Bu şekilde Los Angeles'teki müşteriler, buluta daha hızlı bağlanmış oluyorlar.

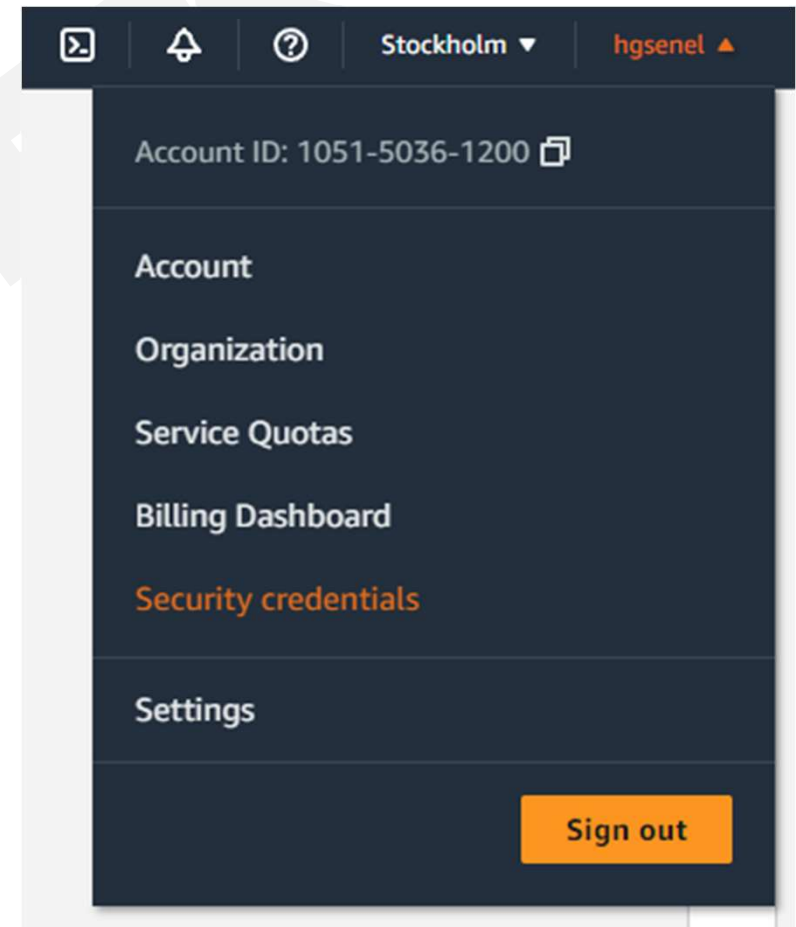


# AWS'nin sunduğu temel hizmetler

- **EC2**: sanal makine servisi
- **S3**: nesne depolama servisi
- **Aurora**: yüksek performanslı ilişkisel veritabanı motoru
- **DynamoDB**: Yönetilebilen NoSQL veritabanı
- **RDS**: Bulutta ilişkisel veritabanı kurma, işletme ve ölçekleme
- **Lambda**: Sunucusuz kod çalıştırma hizmeti
- **VPC**: izole bulut kaynakları oluşturma
- **Lightsail**: Sanal özel sunucular başlatma ve yönetme
- **SageMaker**: makine öğrenmesi modelleri oluşturma, eğitme.

# Güvenlik şifrelerin oluşturulması

- AWS'de yönetim konsolunda, kullanıcı adınızın bulunduğu sağ üst köşeden «Security credentials» bölümüne tıklayarak, IAM (Identity and Access Management) bölümüne girin.
- Sol tarafta «Users» öğesine tıklayın ve «Add users» butonuna basarak «terraform\_user» adı altında yeni bir kullanıcı oluşturun.



# Güvenlik şifrelerin oluşturulması

- Oluşturulacak kullanıcıya **terraform\_user** ismini verelim.
- «Attach policies directly» opsiyonunu seçerek «Administrator Access» yetkisini verelim. Alt kısımda «next» butonuna basarak devam edelim.
- Ardından «Create user» diyerek kullanıcıyı oluşturalım.

### Specify user details

#### User details

User name

The user name can have up to 64 characters. Valid characters: A-Z, a-z, 0-9, and + = , @ \_ - (hyphen)

☐ Provide user access to the AWS Management Console - *optional*  
If you're providing console access to a person, it's a [best practice](#) to manage their access in IAM Identity Center.

[If you are creating programmatic access through access keys or service-specific credentials for AWS CodeCommit or Amazon Keyspaces, you can generate them after you create this IAM user. Learn more](#)

Cancel **Next**

### Permissions options

☐ Add user to group  
Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.

☐ Copy permissions  
Copy all group memberships, attached managed policies, and inline policies from an existing user.

☒ Attach policies directly  
Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

#### Permissions policies (Selected 1/1102)

Choose one or more policies to attach to your new user.

All types

Policy name	Type	Attached entities
<input type="checkbox"/> AccessAnalyzerServiceRolePolicy	AWS managed	0
<input checked="" type="checkbox"/> AdministratorAccess	AWS managed - job function	0
<input type="checkbox"/> AdministratorAccess-Amplify	AWS managed	0



# Güvenlik şifrelerin oluşturulması

- «**Create user**» diyerek, kullanıcıyı oluşturalım.

## Review and create

Review your choices. After you create the user, you can view and download the autogenerated password, if enabled.

### User details

User name terraform_user	Console password type None	Require password reset No
-----------------------------	-------------------------------	------------------------------

### Permissions summary

< 1 >

Name	Type	Used as
<a href="#">AdministratorAccess</a>	AWS managed - job function	Permissions policy

### Tags - optional

Tags are key-value pairs you can add to AWS resources to help identify, organize, or search for resources. Choose any tags you want to associate with this user.

No tags associated with the resource.

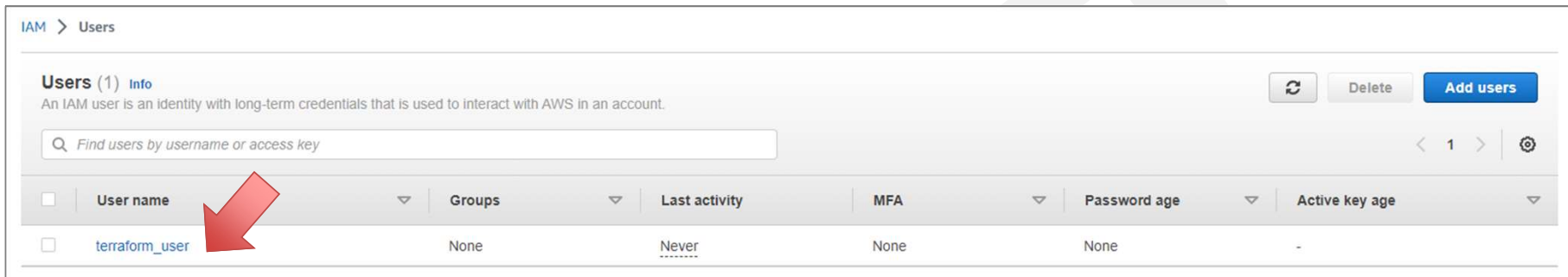
Add new tag

You can add up to 50 more tags.

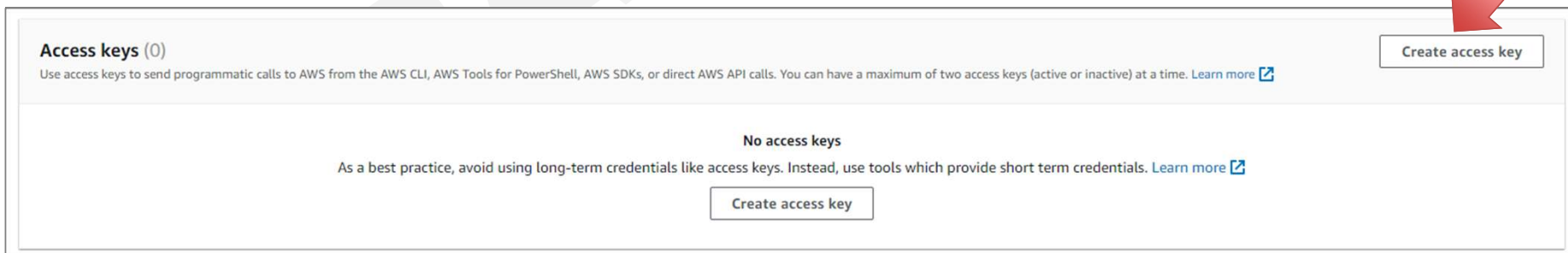
CancelPreviousCreate user

# Güvenlik şifrelerin oluşturulması

- Oluşturulan kullanıcının üzerine tıklayın.



- «Security credentials» tabına tıklayarak, alt kısımda yer alan «create access key» butonuna basınız.



# Güvenlik şifrelerin oluşturulması

- Erişim şifresinin ne amaçla kullanılacağını belirtilmesini istemektedir.

### Access key best practices & alternatives

Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and alternatives.

☒ **Command Line Interface (CLI)**  
You plan to use this access key to enable the AWS CLI to access your AWS account.


☐ **Local code**  
You plan to use this access key to enable application code in a local development environment to access your AWS account.

☐ **Application running on an AWS compute service**  
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

☐ **Third-party service**  
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

☐ **Application running outside AWS**  
You plan to use this access key to enable an application running on an on-premises host, or to use a local AWS client or third-party AWS plugin.

☐ **Other**  
Your use case is not listed here.

 **Alternatives recommended**

- Use [AWS CloudShell](#), a browser-based CLI, to run commands. [Learn more](#)
- Use the [AWS CLI V2](#) and enable authentication through a user in IAM Identity Center. [Learn more](#)

☐ I understand the above recommendation and want to proceed to create an access key.

Cancel Next

# Güvenlik şifrelerin oluşturulması

- Erişim şifresinin ne amaçla kullanılacağını tanıtan bir etiket istemektedir. Bu opsiyoneldir.

## Set description tag - *optional*

The description for this access key will be attached to this user as a tag and shown alongside the access key.

### Description tag value

Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.

terraform için kullanilacak anahtardir

Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: \_ : / = + - @

Cancel

Previous

Create access key

# Güvenlik şifrelerin oluşturulması

- Erişim anahtarı için «**Access key**» bir de «**Secret access key**» oluşturur. Bunlar, görülebilir veya CSV olarak indirilebilir. Bunlar AWS komut satırı programında kullanılacaktır.

Retrieve access keys

**Access key**  
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key	Secret access key
AKIAR WRUF	***** <a href="#">Show</a>

**Access key best practices**

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

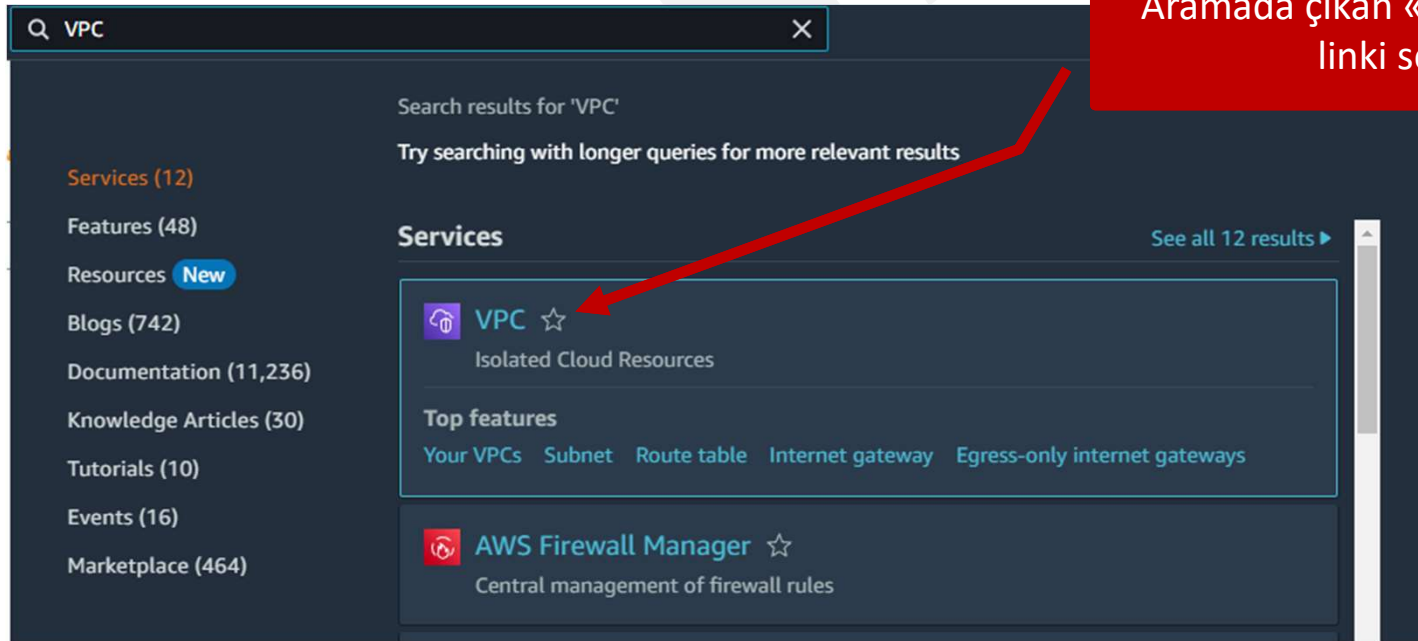
For more details about managing access keys, see the [Best practices for managing AWS access keys](#).

[Download .csv file](#) [Done](#)

Erişim ve gizli erişim anahtarlarını sadece oluşturma anında görebilirsiniz. Bu nedenle «download .csv» diyerek dosya olarak indirin.

# AWS ile ilgili kavramlar

- AWS VPC (Virtual Private Cloud): AWS hesabınıza bağlanmış bir sanal ağ yapısıdır. Diğer sanal ağlardan izole durumdadır.
- VPC'niz içinde IP aralığı belirleyebilir, alt ağlar, geçitler ve güvenlik grupları oluşturabilirsiniz.
- VPC-ID'mizi, AWS yönetim konsolunda VPC kelimesini aratarak ulaşabileceğimiz «VPC Dashboard»tan öğrenebiliriz.



Aramada çıkan «VPC» başlıklı ilk linki seçiniz

# AWS ile ilgili kavramlar

- Görüleceği üzere, AWS hesabımızı oluşturduğumuz zaman bize her bölgede çalışacak şekilde VPC, Alt ağlar, vb yapıları da oluşturulmaktadır.
- Her VPC'nin bir ID'si vardır. VPC içinde bir sanal makine oluşturmak veya firewall kurabilmek için öncelikle bu VPC-ID'yi kenara not etmelisiniz.

The screenshot displays the AWS VPC dashboard. On the left, a sidebar lists navigation options: VPC dashboard, EC2 Global View, Filter by VPC, Virtual private cloud (expanded), Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways, DHCP option sets, Elastic IPs, Managed prefix lists, Endpoints, Endpoint services, NAT gateways, Peering connections, Security (expanded), Network ACLs, Security groups, DNS firewall (expanded), Rule groups, and Domain lists. The main content area shows 'Resources by Region' for the Europe region. It lists various VPC resources with their counts: VPCs (1), Subnets (3), Route Tables (1), Internet Gateways (1), Egress-only Internet Gateways (0), DHCP option sets (1), Elastic IPs (0), Endpoints (0), NAT Gateways (0), VPC Peering Connections (0), Network ACLs (1), Security Groups (1), Customer Gateways (0), Virtual Private Gateways (0), Site-to-Site VPN Connections (0), and Running Instances (0). Two red arrows point to the 'VPCs' and 'Subnets' entries.

Resource	Europe
VPCs	1
Subnets	3
Route Tables	1
Internet Gateways	1
Egress-only Internet Gateways	0
DHCP option sets	1
Elastic IPs	0
Endpoints	0
NAT Gateways	0
VPC Peering Connections	0
Network ACLs	1
Security Groups	1
Customer Gateways	0
Virtual Private Gateways	0
Site-to-Site VPN Connections	0
Running Instances	0

# AWS ile ilgili kavramlar

- AWS hesabımızla oluşturulan VPC'nin ID'sini görebiliriz.

Your VPCs (1) [Info](#)

<input type="checkbox"/>	Name	VPC ID	State	IPv4 CIDR	IPv6 CIDR
<input type="checkbox"/>	-	vpc-0f47241087dd1b70	Available	172.31.0.0/16	-

İşlem yapabilmek için bu ID'yi kenara kaydetmemiz gerekiyor.

- Aynı şekilde, «VPC Dashboard»da «Subnets» bölümüne giderek, kullanacağımız alt ağın ID'sini de almak zorundayız.

Subnets (3) [Info](#)

<input type="checkbox"/>	Name	Subnet ID	State	VPC	IPv4 CIDR
<input type="checkbox"/>	-	subnet-0c831761907703f	Available	vpc-0f47241087dd1b70	172.31.16.0/20
<input type="checkbox"/>	-	subnet-073e4bae581a346	Available	vpc-0f47241087dd1b70	172.31.32.0/20
<input type="checkbox"/>	-	subnet-05edde2781ebce8	Available	vpc-0f47241087dd1b70	172.31.0.0/20

Bu ID'lerden birini kenara kaydetmemiz gerekiyor.



# «AWS CLI» kurulumu

- GCP CLI'yı Ubuntu makinenizden kullanabilmek için ilk önce «**sudo apt install curl**» diyerek «**curl**» paketini yüklemeniz, ardından da «**curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "awscliv2.zip"**» CLI paketini indirebilirsiniz.

```
$ sudo apt install curl
[sudo] password for adminpc:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following package was automatically installed and is no longer required:
  gir1.2-goa-1.0
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
  curl
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 161 kB of archives.
After this operation, 413 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu focal-updates/main amd64 curl amd64 7.68.0-1ubuntu2.18 [161 kB]
Fetched 161 kB in 1s (128 kB/s)
Selecting previously unselected package curl.
(Reading database ... 179364 files and directories currently installed.)
Preparing to unpack .../curl_7.68.0-1ubuntu2.18_amd64.deb ...
Unpacking curl (7.68.0-1ubuntu2.18) ...
Setting up curl (7.68.0-1ubuntu2.18) ...
Processing triggers for man-db (2.9.1-1) ...
$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 55.2M  100 55.2M    0     0 5046k      0  0:00:11  0:00:11 --:--:-- 4580k
```

# «AWS CLI» kurulumu

- İndirilen **zip** dosyasını «**unzip awscliv2.zip**» komutuyla açmamız gerekiyor.
- Ardından da «**sudo ./aws/install**» diyerek programın kurulumunu başlatmalıyız.

```
$ sudo ./aws/install
You can now run: /usr/local/bin/aws --version
$ aws --version
aws-cli/2.12.1 Python/3.11.3 Linux/5.15.0-75-generic exe/x86_64.ubuntu.20 prompt/off
```

- AWS CLI'nin kurulumu için en kolay yöntem, «**aws configure**» komutudur. Oluşturulan AWS anahtarı burada kullanılabilir.

```
$ aws configure
AWS Access Key ID [*****htar]: AKIARQ11TFLTYIAGWTYRF
AWS Secret Access Key [*****NHD2]: fhEnR2AYqzQU9R6a4OBkm5qQHze1VwJyut6znUT
Default region name [us-west-2]: us-west-2
Default output format [None]:
```

«json», «xml», «table», «yaml»,  
«yaml-stream», vb

# «AWS CLI» kullanımı

- «aws» komut satırı programının kullanımı için «aws help» komutu kullanılabilir.

```
AWS () AWS ()

NAME
    aws -

DESCRIPTION
    The AWS Command Line Interface is a unified tool to manage your AWS
    services.

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

    Use aws command help for information on a specific command. Use aws
    help topics to view a list of available help topics. The synopsis for
    each command shows its parameters and their usage. Optional parameters
    are shown in square brackets.
```

- Belirli bir AWS özelliği (örneğin EC2) için «aws ec2 help» denilerek alınabilir.
- Alt özelliklerle ilgili yardım sayfalarına örneğin «aws ec2 describe-instances help» komutuyla ulaşılabilir.
- Bir AWS nesnesi için «aws ec2 describe-instances --instance-ids i-5203422c» komutu kullanılabilir. JSON formatında çıktı üretecektir.

# «AWS CLI» kullanımı

- «**aws configure**» komutundan sonra, güvenlik bilgileri ve erişim anahtarları girilir ve AWS işlemleri için hazır duruma gelinir. Burada AWS kullanıcılarını listelemek için «**aws iam list-users**» komutunu kullanıyoruz.

```
$ aws configure
AWS Access Key ID [None]: AKIACK77RTASKKLPDBYZ
AWS Secret Access Key [None]: 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
Default region name [None]: us-west-2
Default output format [None]: json
```

```
$ aws iam list-users
{
  "Users": [
    {
      "Path": "/",
      "UserName": "terraform_user",
      "UserId": "AIDARQ63TFRTY6YFCKRE",
      "Arn": "arn:aws:iam::103244461400:user/terraform_user",
      "CreateDate": "2023-06-18T23:20:26+00:00"
    }
  ]
}
```

Hesabınız hangi bölgedeyse, bunu yazmalısınız. Yoksa, yaptığınız komut satırı işlemlerinde **InvalidVpcID.NotFound** hatasını alırsınız.

## EC2

- AWS'nin popüler özelliklerinden biridir. AWS'nin temelini oluşturmaktadır.
- EC2'nin öğrenilmesi, AWS'nin nasıl çalıştığını anlamak için faydalıdır.
- EC2 şu hizmetleri sağlar
  - Sanal makinelerin kiralanması (EC2)
  - Sanal sürücülerde verilerin depolanması (EBS)
  - Makineler arasında yükün paylaşılması (ELB)
  - Otomatik ölçekleme kullanarak hizmetlerin ölçeklenmesi (ASG)

## EC2 anahtar çifti

- EC2 (Elastic Compute Cloud) için gerekli anahtar çiftlerini **aws** komut satırı programı üzerinden oluşturabilirsiniz. Bu anahtarlar EC2'de oluşturulan sanal makinelere erişim için kullanılmaktadır.
- «**aws ec2 create-key-pair --key-name AnahtarCifti --query 'KeyMaterial' --output text > AnahtarCiftim.pem**» komutu, açık ve gizli anahtar çifti oluşturur ve **gizli anahtar AnahtarCiftim.pem** isimli dosyaya kaydedilir.
- Bu anahtar AWS'de tutulmamaktadır ve bu nedenle kullanıcı tarafından saklanmalıdır.
- Anahtar elde edildikten sonra «**chmod 400 AnahtarCiftim.pem**» komutuyla, dosyanın sistemdeki diğer kullanıcılar tarafından görülmesi engellenmelidir.

```
$ aws ec2 create-key-pair --key-name AnahtarCifti --query 'KeyMaterial' --output text  
> AnahtarCiftim.pem  
$ chmod 400 AnahtarCiftim.pem
```

## EC2 anahtar çifti

- EC2 (Elastic Compute Cloud) için gerekli anahtar çiftleriyle ilgili kullanılan komutlar şunlardır:
  - `$ aws ec2 create-key-pair`
  - `$ aws ec2 delete-key-pair`
  - `$ aws ec2 describe-key-pair`
    - Bu komut elimizdeki gizli anahtarın «**fingerprint**»ini gösterir ve elimizdeki anahtarın AWS tarafında geçerli olup olmadığı görülebilir.

```
$ $ aws ec2 create-key-pair --key-name MyKeyPair --query 'KeyMaterial' --output text > MyKeyPair.pem
{
  "KeyPairs": [
    {
      "KeyPairId": "key-04509d6ade711a7d1",
      "KeyFingerprint": "b1:41:54:3d:fc:56:f7:b2:80:44:23:41:9a:4e:b3:6e:e1:87:0e:b4",
      "KeyName": "MyKeyPair",
      "KeyType": "rsa",
      "Tags": [],
      "CreateTime": "2023-04-19T07:34:12.924000+00:00"
    }
  ]
}
```

# EC2 Security Group

- «**Security Group**», sanal «**firewall**» olarak çalışan bir sistemdir.
- EC2’de sanal makineler oluşturulmadan önce güvenlik grubu oluşturulmalıdır.
- EC2 sanal makinelere giren ve çıkan ağ trafiği için kurallar belirlenebilir.
- Kurallar IP ve portlar bazında oluşturulabilir.
- Grup kuralları her zaman izin veren tarzdadır ve «**deny**» şeklinde bir kural belirlenmez.
- Örneğin, Elastic Load Balancer’dan (ELB) web sunucuların bulunduğu alt ağa giren trafiğe izin verilir. SG, ELB’yi izin verilen kaynak olarak belirler.
- EC2’de oluşturulan sanal makinelere birden fazla SG tanımlanabilir çünkü bir sanal makineye en fazla 5 adet sanal NIC tanımlanabilmektedir. Ayrıca alt ağlarda yer alan her bir sanal makine farklı SG’lere atanabilir.
- **Güvenlik grubu oluşturabilmek için VPC’nin ID’si ve en az bir Alt Ağ (subnet) ID’sine gerek duyulur. Bu ayrıntılar AWS yönetim konsolundan elde edilebilir.**



# EC2 Security Group

- EC2 SG için temel komut «**aws ec2 create-security-group**» komutudur. Belirli bir VPC için güvenlik grubu oluşturmak için aşağıdaki komut verilebilir.

```
$ aws ec2 create-security-group --group-name benim-gg --description "guvenlik-grubu" --vpc-id vpc-1a2b3c4d
{
  "GroupId": "sg-903004f8"
}
$ aws ec2 describe-security-groups --group-ids sg-903004f8
{
  "SecurityGroups": [
    {
      "IpPermissionsEgress": [
        {
          "IpProtocol": "-1",
          "IpRanges": [
            {
              "CidrIp": "0.0.0.0/0"
            }
          ]
        }
      ],
      "UserIdGroupPairs": []
    }
  ],
  "Description": "guvenlik-grubu",
  "IpPermissions": [],
  "GroupName": "benim-gg",
  "VpcId": "vpc-1a2b3c4d",
  "OwnerId": "123456789012",
  "GroupId": "sg-903004f8"
}
```

# EC2 Security Group

- Belirli bir IP'ye izin verilmesi «**authorize-security-group-ingress**» opsiyonuyla gerçekleştirilir.

```
$ aws ec2 authorize-security-group-ingress --group-id sg-903004f8 --protocol tcp --port 3389 --cidr x.x.x.x/x
```

- EC VM'lere SSH izninin verilmesi için

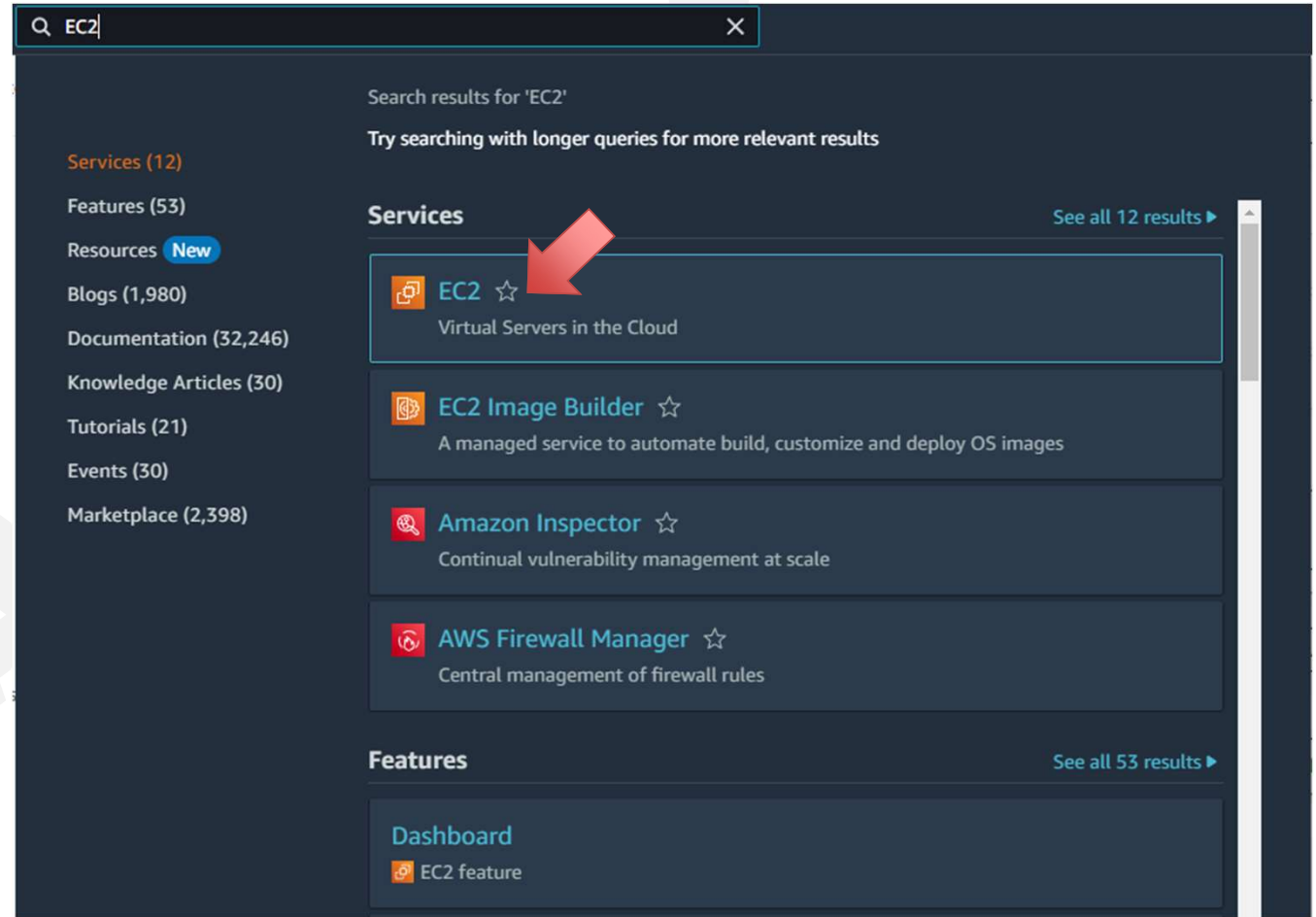
```
$ aws ec2 authorize-security-group-ingress --group-id sg-903004f8 --protocol tcp --port 22 --cidr x.x.x.x/x
```

- SG'deki kuralların görülmesi için

```
$ aws ec2 describe-security-groups --group-ids sg-903004f8
{
  "SecurityGroups": [
    {
      "IpPermissionsEgress": [
        {
          "IpProtocol": "-1",
          "IpRanges": [
            {
              "CidrIp": "0.0.0.0/0"
            }
          ]
        },
        .....
      ]
    }
  ]
}
```

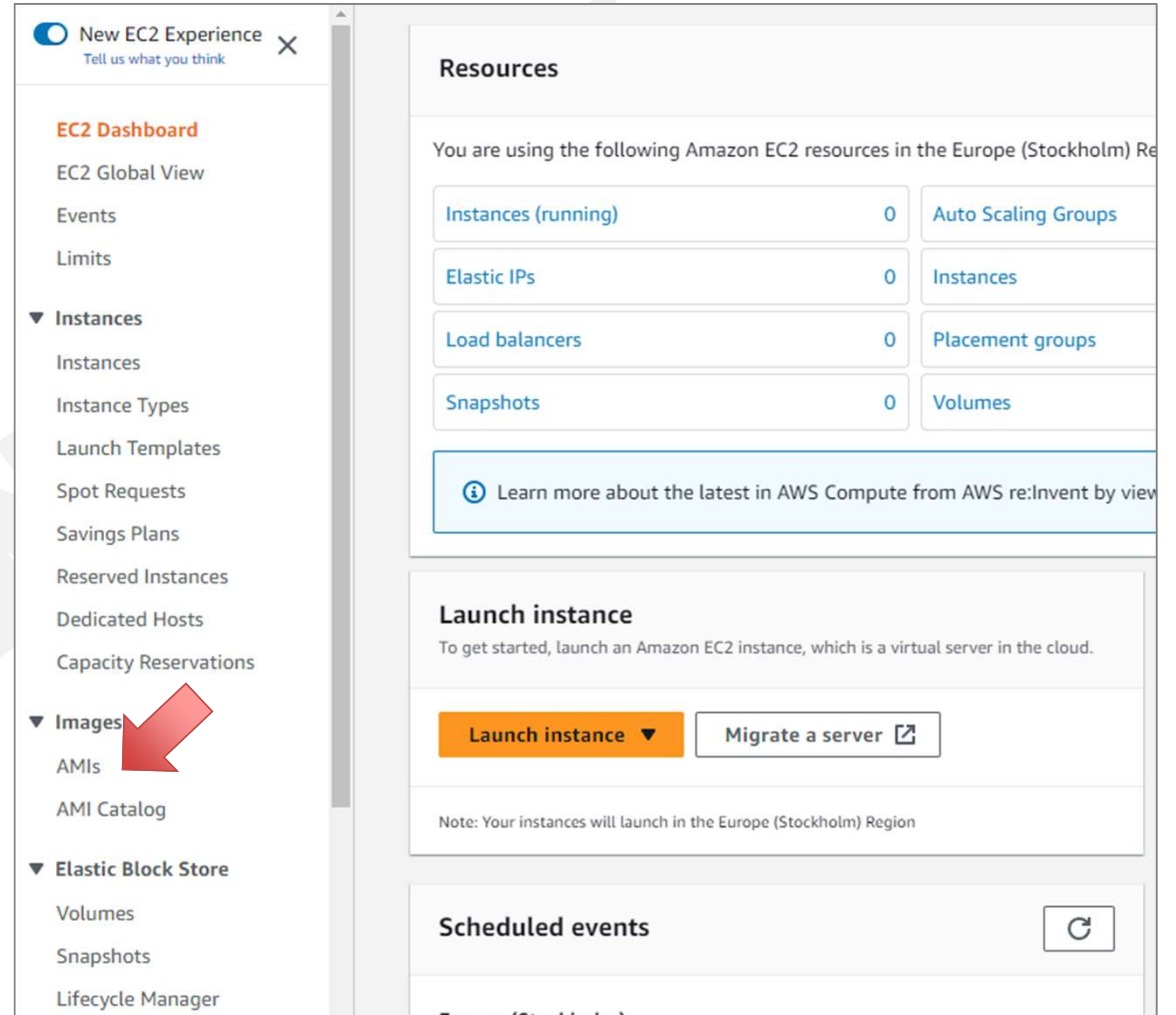
## EC2 imaj ID

- Sanal makine oluşturmak için hangi işletim sistemi imajını kullanmamız gerektiğini, **aws** komutuna bildirmemiz gereklidir.
- İmaj kataloğuna erişim için arama bölümünde «**EC2**» yazılarak, alt kısımda «**EC2**» opsiyonu seçilmelidir.



## EC2 imaj ID

- EC2 Dashboard'un altında «**Images**» opsiyonunun altındaki «**AMIs**» yazan kısma tıklandığında daha önce kullandığınız imajlar görebilirsiniz. Ancak daha önce kendiniz imaj oluşturmadıysanız bu bölümün boş olduğunu görebilirsiniz.
- Böyle bir durumda «**AMI catalog**» yazan opsiyonu seçmelisiniz.



## EC2 imajın seçilmesi

- «**AMI catalog**» altında çok sayıda imajın listelendiği görülebilir.
- Bazı imajlar, Amazon'un 5 yıllık **LTS** (Long Term Support) desteğiyle birlikte gelmektedir.
- Windows imajlar da bulunmaktadır.

The screenshot displays the AWS AMI Catalog interface. At the top, there is a search bar with the placeholder text "Search for an AMI by entering a search term e.g. 'Windows'". Below the search bar, there are four tabs: "Quickstart AMIs (47) Commonly used AMIs", "My AMIs (0) Created by me", "AWS Marketplace AMIs (7858) AWS & trusted third-party AMIs", and "Community AMIs (500) Published by anyone". The "Quickstart AMIs" tab is selected. On the left side, there are filters for "Free tier only Info", "OS category" (All Linux/Unix, All Windows), and "Architecture" (64-bit (Arm), 32-bit (x86), 64-bit (x86), 64-bit (Mac)). The main content area shows three AMI entries. The first entry is "Amazon Linux 2023" (ami-04e4606740c9c9381) with a red arrow pointing to the "Select" button. The second entry is "Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type" (ami-0cf14aecd598bc8f). The third entry is "macOS Ventura 13.4" (ami-0c7f3a4316669ef18). Each entry includes a description, platform, root device type, virtualization, and ENA enabled status.

# EC2 imajın seçilmesi

- Arama bölümüne Ubuntu yazalım ve çıkan imajlara bakalım ve bir tanesini seçelim

Search results for 'ubuntu' (9 filtered, 9 unfiltered):

**Refine results**

Clear all filters

OS category

Architecture

**ubuntu** <sup>®</sup>

Free tier eligible

Verified provider

**Ubuntu Server 22.04 LTS (HVM), SSD Volume Type**

ami-0989fb15ce71ba39e (64-bit (x86)) / ami-0ebb6753c095cb52a (64-bit (Arm))

Ubuntu Server 22.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Platform: ubuntu Root device type: ebs Virtualization: hvm ENA enabled: Yes

**Select**

64-bit (x86)

64-bit (Arm)

**Ubuntu Server 20.04 LTS (HVM), SSD Volume Type**

ami-08766f81ab52792ce (64-bit (x86)) / ami-0ff124a3d7381bfec (64-bit (Arm))

Ubuntu Server 20.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Platform: ubuntu Root device type: ebs Virtualization: hvm ENA enabled: Yes

**Select**

64-bit (x86)

64-bit (Arm)

**Ubuntu Pro - Ubuntu Server Pro 22.04 LTS (HVM), SSD Volume Type**

ami-02fd0c9fa584f3ec9 (64-bit (x86)) / ami-0a5e68f5aabce2242 (64-bit (Arm))

Ubuntu Pro - Ubuntu Server Pro 22.04 LTS (HVM), EBS General Purpose (SSD) Volume Type. Support available from Canonical (<http://www.ubuntu.com/cloud/services>).

Platform: ubuntu Root device type: ebs Virtualization: hvm ENA enabled: Yes

**Select**

64-bit (x86)

64-bit (Arm)

**HVM:** Hardware virtual machine  
**PV:** Paravirtual Machine



## EC2 imajın seçilmesi

- Seçtikten sonra «**Create Template with AMI**» veya «**Launch Instance with AMI**» denilerek, Web arayüzü üzerinden işlemler yapılabilir.
- Ancak bu aşamada sadece AMI ID'yi elde etmek için buradayız. Yeni bir VM'i «**aws**» CLI ile oluşturacağız.

AMIs

Selected AMI: (ami-0989fb15ce71ba39e)

Create Template with AMI Launch Instance with AMI

Q ubuntu

Quickstart AMIs (9) Commonly used AMIs

My AMIs (0) Created by me

AWS Marketplace AMIs (1962) AWS & trusted third-party AMIs

Community AMIs (500) Published by anyone

Refine results

Clear all filters

Free tier only Info

OS category

All Linux/Unix

All Windows

ubuntu (9 filtered, 9 unfiltered)

ubuntu

Free tier eligible

Verified provider

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type

ami-0989fb15ce71ba39e (64-bit (x86))

ami-0ebb6753c095cb52a (64-bit (Arm))

Select

64-bit (x86)

64-bit (Arm)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type. Support available from Canonical (http://www.ubuntu.com/cloud/services). Platform: ubuntu Root device type: ebs Virtualization: hvm ENA enabled: Yes

AMI ID olarak bu değeri kullanacağız:  
**ami-0989fb15ce71ba39e**

# Sanal makine tipi «**Instance Type**»

- Oluşturulacak sanal makinenin hangi işletim sisteminde çalışacağını bulduk. Ancak, bu makineyi ne kadar bellek ve ne kadar sanal CPU ile çalıştıracağımızı belirlememiz gerekli. Bu tipler, «**EC2 Dashboard**» altında «**Instance**» ve «**Instance Types**» öğeleri üzerinden incelenebilir.
- AWS bize farklı makine şablonları sunmaktadır. Bunlara «**instance type**» adı verilmektedir.
- Örneğin, «**t3.nano**» tipi dediğimiz zaman 2 sanal CPU'lu, X86\_64 mimarisinde 0.5GiB RAM'ı olan bir makine tarif ediyoruz. Bu tipe en fazla 5 Gbps'lık ağ bant genişliği sağlanmaktadır.
- Diğer yandan, daha büyük makineler de (daha pahalıları da) mevcuttur. Örneğin, «**c5n.metal**» türü makine 72 sanal CPU'lu ve 192GB RAM'a sahiptir. 100Gbps'lık ağ bant genişliği vardır. Ancak Linux çalıştığında saat başına \$4176 ödenmelidir.



# Sanal makine tipi «Instance Type»

- Makine tipleri şu şekilde görülebilir.

Instance types (275+)							
<input type="text" value="Find resources by attribute or tag"/>							
<div>&lt; 1 2 3 4 5 6 ... &gt; ⚙</div>							
<input type="checkbox"/>	Instance type ▾	vCPUs ▾	Architecture ▾	Memory (GiB) ▾	Storage (GB) ▾	Storage type ▾	Network performance
<input type="checkbox"/>	t3.nano	2	x86_64	0.5	-	-	Up to 5 Gigabit
<input type="checkbox"/>	t3.micro	2	x86_64	1	-	-	Up to 5 Gigabit
<input type="checkbox"/>	t3.small	2	x86_64	2	-	-	Up to 5 Gigabit
<input type="checkbox"/>	t3.medium	2	x86_64	4	-	-	Up to 5 Gigabit
<input type="checkbox"/>	t3.large	2	x86_64	8	-	-	Up to 5 Gigabit
<input type="checkbox"/>	t3.xlarge	4	x86_64	16	-	-	Up to 5 Gigabit
<input type="checkbox"/>	t3.2xlarge	8	x86_64	32	-	-	Up to 5 Gigabit
<input type="checkbox"/>	c5.large	2	x86_64	4	-	-	Up to 10 Gigabit
<input type="checkbox"/>	c5.xlarge	4	x86_64	8	-	-	Up to 10 Gigabit
<input type="checkbox"/>	c5.2xlarge	8	x86_64	16	-	-	Up to 10 Gigabit

# Güvenlik Grubu (Security Group) oluşturma

- Sanal makine oluşturmadan önce güvenlik grubu oluşturmaliyiz.

```
$ aws ec2 create-security-group --group-name demo-gg --description "AWS deneme" --vpc-id "vpc-0f47241087ddccb70"
```

```
An error occurred (InvalidVpcID.NotFound) when calling the CreateSecurityGroup operation: The vpc ID 'vpc-0f47241087ddccb70' does not exist
```

```
$ aws configure
```

```
AWS Access Key ID [*****GPYY]:
```

```
AWS Secret Access Key [*****s3yh]:
```

```
Default region name [us-west-2]: eu-north-1
```

```
Default output format [json]:
```

```
$ aws ec2 create-security-group --group-name demo-gg --description "AWS deneme" --tag-specifications 'ResourceType=security-group,Tags=[{Key=Name,Value=demo-gg}]' --vpc-id "vpc-0f47241087ddccb70"
```

```
{
  "GroupId": "sg-06ead57486493777b",
  "Tags": [
    {
      "Key": "Name",
      "Value": "demo-gg"
    }
  ]
}
```

Neden bu hatayı aldık. VPC-ID'miz hesabımızın olduğu «eu-north-1» alanında tanımlı. «aws configure» ile bunu değiştirmeliyiz.

Yeni güvenlik grubumuzu oluşturduk.

# Sanal Makine oluşturma

- Sanal makine oluşturmadan önce güvenlik grubu oluşturmaliyiz.

```
$ aws ec2 run-instances --image-id ami-0989fb15ce71ba39e --count 1 --instance-type t3.micro --key-name MyKeyPair --security-group-ids sg-06eda57486493ea8b --subnet-id subnet-0c831761907703f8d
```

```
{
  "Groups": [],
  "Instances": [
    {
      "AmiLaunchIndex": 0,
      "ImageId": "ami-0989fb15ce71ba39e",
      "InstanceId": "i-04b05bd9fa285682a",
      "InstanceType": "t3.micro",
      "KeyName": "MyKeyPair",
      "LaunchTime": "2023-04-19T14:03:52+00:00",
      "Monitoring": {
        "State": "disabled"
      },
      "Placement": {
        "AvailabilityZone": "eu-north-1a",
        "GroupName": "",
        "Tenancy": "default"
      },
      "PrivateDnsName": "ip-172-31-31-7.eu-north-1.compute.internal",
      "PrivateIpAddress": "172.31.31.7",
      "ProductCodes": [],
      "PublicDnsName": "",
      "State": {
```

Anahtarı daha önce oluşturmuştuk.

# Güvenlik grubunda kural değişimi

- Güvenlik grubuna 80 ve 22 numaralı portlara erişim imkanı sağlayalım.

```
$ aws ec2 authorize-security-group-ingress --group-id "sg-06eda57486493ea8b" --tag-specifications  
'ResourceType=security-group-rule,Tags=[{Key=Name,Value=demo-gg}]' --ip-permissions  
"IpProtocol=tcp,FromPort=22,ToPort=22,IpRanges=[{CidrIp=0.0.0.0/0},{CidrIp=10.0.0.0/24}]" --ip-permissions  
"IpProtocol=tcp,FromPort=80,ToPort=80,IpRanges=[{CidrIp=0.0.0.0/0},{CidrIp=10.0.0.0/24}]"  
{  
  "Return": true,  
  "SecurityGroupRules": [  
    {  
      "SecurityGroupRuleId": "sgr-01f40bf972d44297a",  
      "GroupId": "sg-06eda57486493ea8b",  
      "GroupOwnerId": "105150361200",  
      "IsEgress": false,  
      "IpProtocol": "tcp",  
      "FromPort": 80,  
      "ToPort": 80,  
      "CidrIpv4": "0.0.0.0/0",  
      "Tags": [  
        {  
          "Key": "Name",  
          "Value": "demo-gg"  
        }  
      ]  
    },  
    {  
      "SecurityGroupRuleId": "sgr-0c9ca023cd6a92193",  
      "GroupId": "sg-06eda57486493ea8b",  
      "GroupOwnerId": "105150361200",  
      "IsEgress": false,  
      "IpProtocol": "tcp",  
      "FromPort": 22,  
      "ToPort": 22,  
      "CidrIpv4": "0.0.0.0/0",  
      "Tags": [  
        {  
          "Key": "Name",  
          "Value": "demo-gg"  
        }  
      ]  
    }  
  ]  
}
```

22 numaralı kural aşağıda!



# Güvenlik grubunda kural değişimi

- Güvenlik grubuna 80 ve 22 numaralı portlara erişim imkanı sağlayalım.

```
$ aws ec2 run-instances --image-id ami-0989fb15ce71ba39e --count 1 --instance-type t3.micro --  
key-name MyKeyPair --security-group-ids sg-06eda57486493ea8b --subnet-id subnet-  
0c831761907703f8d --associate-public-ip-address --query 'Instances[0].InstanceId'  
"i-06ce5b5c6509c0759"  
$ aws ec2 describe-instances --instance-ids "i-06ce5b5c6509c0759" --query  
"Reservations[0].Instances[0].PublicIpAddress" --output text  
16.16.211.87
```

Bu «instance»ın IP numarasını  
bulalım

# AWS EC2 VM OLUŞTURMA ÖRNEĞİ

<https://cloudbytes.dev/aws-academy/how-to-create-an-aws-ec2-instance-using-aws-cli>

## Gerekli hazırlıklar

- Bir sanal makine oluşturmak için, ilk olarak AMI (imaj) ID'sini, «**Instance Type**»ı, belirlemek gereklidir. Bunlar için Bash değişkenler oluşturmak iyi fikirdir.

```
$ INSTANCE_TYPE=t3.micro  
$ AMI_ID=ami-0989fb15ce71ba39e
```

- Anahtar çifti oluşturarak, bu anahtar çiftinin gizli olanını dosya sistemine kaydedeceğiz ve ardından erişim modunu değiştireceğiz.

```
$ aws ec2 create-key-pair \  
  --key-name my-key-pair \  
  --query 'KeyMaterial' \  
  --output text > my-key-pair.pem  
$ chmod 400 my-key-pair.pem
```

Gizli anahtarımız içinde bulunduğumuz dizinde, **my-key-pair.pem** dosyasında yer alıyor.

## Gerekli hazırlıklar

- Güvenlik grubunu oluşturacağız ve 22, 80 ve 443 numaralı portları erişime açacağız.

```
$ SECURITY_GROUP=$(aws ec2 create-security-group \
  --group-name "my-web-sg" \
  --description "Web security group" \
  --query 'GroupId' \
  --output text) && \
echo "Security group created with id $SECURITY_GROUP"
$ aws ec2 authorize-security-group-ingress \
  --group-id $SECURITY_GROUP \
  --protocol tcp \
  --port 22 \
  --cidr 0.0.0.0/0
$ aws ec2 authorize-security-group-ingress \
  --group-id $SECURITY_GROUP \
  --protocol tcp \
  --port 80 \
  --cidr 0.0.0.0/0
```

\$SECURITY\_GROUP isimli değişken, güvenlik grubunun ID'sini içerecek.



## Gerekli hazırlıklar

- 443 numaralı portu açalım

```
$ aws ec2 authorize-security-group-ingress \  
  --group-id $SECURITY_GROUP \  
  --protocol tcp \  
  --port 443 \  
  --cidr 0.0.0.0/0
```

- Subnet'i bulmamız gerekiyor. Subnetlerimiz, VPC'miz oluşturulduğunda daha önce oluşturulmuştur. Birden fazla olabilir ama birincisini alacağız ve **SUBNET\_ID** isimli Bash değişkenine koyacağız ve ekrana yazdıracağız.

```
SUBNET_ID=$(aws ec2 describe-subnets \  
  --filters "Name=availability-zone,Values=eu-north-1a" \  
  --query "Subnets[0].SubnetId" --output text) && \  
echo "Subnet ID for eu-north-1a: $SUBNET_ID"
```

İlk subnet

# EC2 Sanal makineyi oluşturalım

- Şu ana kadar oluşturduğumuz değişkenler **AMI\_ID**, **INSTANCE\_TYPE**, **SECURITY\_GROUP** ve **SUBNET\_ID** oluşturma sırasında kullanılacak.

```
$ INSTANCE_ID=$(aws ec2 run-instances \
  --image-id $AMI_ID \
  --count 1 \
  --instance-type $INSTANCE_TYPE \
  --key-name my-key-pair \
  --security-group-ids $SECURITY_GROUP \
  --subnet-id $SUBNET_ID \
  --associate-public-ip-address \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=my-ec2-instance}]' \
  --block-device-mappings '[{"DeviceName":"/dev/xvda","Ebs":{"VolumeSize":20,"VolumeType":"gp2"}}]' \
  --query 'Instances[0].InstanceId' \
  --output text) && \
echo "Instance launched with id $INSTANCE_ID"
```

- Oluşturulan makinenin IP numarasını bulabilmek a ilk subnet t kullanılabilir.

```
INSTANCE_IP=$(aws ec2 describe-instances \
  --instance-ids $INSTANCE_ID \
  --query "Reservations[0].Instances[0].PublicIpAddress" --output text) && \
echo "EC2 instance myServer1 IP: $INSTANCE_IP"
```

# EC2 Sanal makineyi oluşturalım

- Şu ana kadar oluşturduğumuz değişkenler **AMI\_ID**, **INSTANCE\_TYPE**, **SECURITY\_GROUP** ve **SUBNET\_ID** oluşturma sırasında kullanılacak.

```
$ INSTANCE_ID=$(aws ec2 run-instances \
  --image-id $AMI_ID \
  --count 1 \
  --instance-type $INSTANCE_TYPE \
  --key-name my-key-pair \
  --security-group-ids $SECURITY_GROUP \
  --subnet-id $SUBNET_ID \
  --associate-public-ip-address \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=my-ec2-instance}]' \
  --query 'Instances[0].InstanceId' \
  --output text) && echo "Instance launched with id $INSTANCE_ID"
Instance launched with id i-05630781a104a8e41
```


Oluşturulan  
makinenin ID'si

- Oluşturulan makinenin IP numarasını bulabilmek aşağıdaki komut kullanılabilir.

```
$ INSTANCE_IP=$(aws ec2 describe-instances \
  --instance-ids $INSTANCE_ID \
  --query "Reservations[0].Instances[0].PublicIpAddress" --output text) && \
  echo "EC2 instance myServer1 IP: $INSTANCE_IP"
EC2 instance myServer1 IP: 16.170.146.11
```

# Sanal makineye SSH üzerinden erişim

- Sanal makineye SSH ile oluşturduğumuz gizli anahtar üzerinden erişebiliriz.



```
$ ssh -i my-key-pair.pem ubuntu@$INSTANCE_IP
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.19.0-1025-aws x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

System information as of Mon Jun 19 16:50:34 UTC 2023

System load:  0.0166015625      Processes:            103
Usage of /:   20.8% of 7.57GB   Users logged in:     0
Memory usage: 23%              IPv4 address for ens5: 172.31.19.249
Swap usage:   0%

Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Mon Jun 19 16:49:08 2023 from 13.48.4.202
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-19-249:~$
```

Ubuntu imajı  
kullandığımızdan,  
kullanıcı ismi olarak  
Ubuntu vereceğiz.

## Ne öğrendik?

- AWS'yi komut satırından kullanmanın ne kadar zor olduğunu gördük.
- Sanal bir makine başlatmak için AWS'de bir dizi işlem yapmamız gerektiğini ve bu adımlardan birini bile atladığımızda makinenin oluşturulamayacağını gördük.
- AWS'nin yapısı da daha önce gördüğümüz Azure'un mimarisinden değişik olduğu için iş yapış sırası Azure'dan oldukça farklı.
- AWS'yle kıyaslandığında Azure'u kullanmak daha kolay olabilir.
- Terraform işimizi çözer mi?

**TERRAFORM**

# Terraform nasıl kurulur?

- Ubuntu'da kontrol makinesinde kurulum için aşağıdaki komutların girilmesi yeterlidir.

```
$ sudo apt update
```

```
$ sudo apt upgrade
```

```
$ sudo apt install software-properties-common gnupg2  
curl
```

- Terraform için kullanılacak GPG anahtarını indirip kuracağız (ev dizinizde veya yazma yetkisi olan bir dizinin içinde olmalısınız)

```
$ curl https://apt.releases.hashicorp.com/gpg | gpg --  
dearmor > ./hashicorp.gpg
```

```
$ sudo install -o root -g root -m 644 ./hashicorp.gpg  
/etc/apt/trusted.gpg.d/
```

- Hashicorp'un paket deposunun linkini APT'e ekliyoruz.

```
$ sudo apt-add-repository "deb [arch=$(dpkg --print-  
architecture)] https://apt.releases.hashicorp.com  
$(lsb_release -cs) main"
```

# Terraform'un yüklenmesi

- Terraform'u «**sudo apt install terraform**» ile yüklüyoruz ve «**terraform --version**» komutuyla sürümüne bakıyoruz.

```
$ sudo apt install terraform
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  gir1.2-goa-1.0 linux-headers-5.15.0-67-generic linux-hwe-5.15-headers-5.15.0-67
  linux-image-5.15.0-67-generic linux-modules-5.15.0-67-generic
  linux-modules-extra-5.15.0-67-generic
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  terraform
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 21.5 MB of archives.
After this operation, 64.5 MB of additional disk space will be used.
Get:1 https://apt.releases.hashicorp.com focal/main amd64 terraform amd64 1.4.6-1 [21.5 MB]
Fetched 21.5 MB in 4s (5,101 kB/s)
Selecting previously unselected package terraform.
(Reading database ... 279485 files and directories currently installed.)
Preparing to unpack .../terraform_1.4.6-1_amd64.deb ...
Unpacking terraform (1.4.6-1) ...
Setting up terraform (1.4.6-1) ...
$ terraform --version
Terraform v1.4.6
on linux_amd64
```



# Kurumsal kullanıcılar: Terraform nasıl kurulur?

- Kurumsal kullanıcılar, genellikle, sistemlerindeki yazılım kurulum süreçlerini kontrol altında tutmaktadır. Bu nedenle kurumsal yapılarda Terraform'un kurulabilmesi için «**sudo apt install**» komutu kullanılamayabilir.
- Böyle bir durumda, Terraform'un çalıştırılabilir kodu, ZIP dosyası olarak indirilerek, «**/usr/bin/**» dizinine kopyalanarak gerekli izinlerle çalıştırılabilir hale getirilebilir.
- Ancak, aşağıdaki komutların çalışabilmesi için «**wget**» ve «**unzip**» paketlerinin kurulu olması gereklidir. Bunlar «**sudo apt install wget unzip**» komutuyla yüklenebilir veya daha önce bir şekilde yüklü olması gereklidir.
- **TERRAFORM\_VERSION** isimli bir çevre değişkenine, «**inline**» bir Bash betikle Terraform'un en son sürümünü eşitleyeceğiz.

```
$ TERRAFORM_VERSION=$(curl -s  
https://api.github.com/repos/hashicorp/terraform/releases/latest | grep tag_name |  
cut -d: -f2 | tr -d \"\, \v | awk '{ $1=$1 };1')  
$ echo $TERRAFORM_VERSION  
1.6.2
```

Kuracağımız son sürüm numarası 1.6.2  
(İstenirse, betik yerine web sitesine giderek son sürümün  
numarası bulunup elle de set edilebilir)

# Kurumsal kullanıcılar: Terraform nasıl kurulur?

- Terraform'un en son sürümünü «**wget**» komutuyla yazma yetkisine sahip olduğumuz dizine (örneğin ev dizinimize) kaydediyoruz.

```
$ wget
"https://releases.hashicorp.com/terraform/${TERRAFORM_VERSION}/terraform_${TERRAFORM_VERSION}
_linux_amd64.zip"
--2023-10-23 22:20:26--
https://releases.hashicorp.com/terraform/1.6.2/terraform_1.6.2_linux_amd64.zip
Resolving releases.hashicorp.com (releases.hashicorp.com)... 18.244.87.80, 18.244.87.55,
18.244.87.13, ...
Connecting to releases.hashicorp.com (releases.hashicorp.com)|18.244.87.80|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 24738688 (24M) [application/zip]
Saving to: `terraform_1.6.2_linux_amd64.zip'

terraform_1.6.2_lin 100%[=====>] 23,59M 5,13MB/s in 4,5s

2023-10-23 22:20:31 (5,26 MB/s) - `terraform_1.6.2_linux_amd64.zip' saved [24738688/24738688]
```

# Kurumsal kullanıcılar: Terraform nasıl kurulur?

- Ardından, «**unzip**» komutuyla Terraform dosyasını açıyoruz ve **/usr/local/bin** dizinine «**sudo**» ile aktarıyoruz. Program dosyasına çalıştırma izni de vermemiz gerekli.

```
$ unzip terraform_1.6.2_linux_amd64.zip
Archive:  terraform_1.6.2_linux_amd64.zip
  inflating: terraform
$ sudo mv terraform /usr/local/bin
$ sudo chmod 755 /usr/local/bin/terraform
$ which terraform
/usr/local/bin/terraform
$ file /usr/local/bin/terraform
/usr/local/bin/terraform: ELF 64-bit LSB executable, x86-64, version 1
(SYSV), statically linked, Go BuildID=UG2gu9an-
ojtIj8yko1K/crRbiafo7Zpt9xcmh5Ps/57zggVki4nYocMCXY7Rd/93LLkxR1-62LmsvFIY3G,
stripped
$ terraform version
Terraform v1.6.2
on linux_amd64
```

# Terraform

- Basit bir Terraform kodu oluşturalım ve Terraform ile EC2'de sanal bir makine oluşturmaya çalışalım:

```
$ cat > main.tf
# Configure the AWS provider
provider "aws" {
  region = "eu-north-1"
}

# Create an EC2 instance
resource "aws_instance" "example" {
  ami          = "ami-0989fb15ce71ba39e"
  instance_type = "t3.micro"
}
$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding latest version of hashicorp/aws...
- Installing hashicorp/aws v5.22.0...
- Installed hashicorp/aws v5.22.0 (signed by HashiCorp)

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!
$ terraform validate
Success! The configuration is valid.
```

# Terraform

- «**terraform apply**» komutu verdiğimizde oluşturacağı makinenin bütün özelliklerini gösterecek ve «**yes**» yanıtını bekleyecek:

```
.....  
+ user_data_replace_on_change      = false  
+ vpc_security_group_ids           = (known after apply)  
}
```

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?  
Terraform will perform the actions described above.  
Only 'yes' will be accepted to approve.

Enter a value: yes

aws\_instance.example: Creating...  
aws\_instance.example: Still creating... [10s elapsed]  
aws\_instance.example: Creation complete after 14s [id=i-04891555dc29302e9]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

«**yes**» girilmesini  
bekleyecek»

Instances (5) Info									
Find instance by attribute or tag (case-sensitive)									
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Pub	
<input type="checkbox"/>	-	i-04b05bd9fa285682a	Terminated	t3.micro	-	No alarms	eu-north-1a	-	
<input type="checkbox"/>	-	i-04891555dc29302e9	Terminated	t3.micro	-	No alarms	eu-north-1a	-	
<input type="checkbox"/>	-	i-025a0f5bbe08abf5b	Running	t3.micro	Initializing	No alarms	eu-north-1a	ec2-	
<input type="checkbox"/>	-	i-06ce5b5c6509c0759	Terminated	t3.micro	-	No alarms	eu-north-1a	-	
<input type="checkbox"/>	my-ec2-instance	i-05630781a104a8e41	Terminated	t3.micro	-	No alarms	eu-north-1a	-	

<https://github.com/alfonsof/terraform-aws-examples/blob/master/code/01-hello-world/main.tf>

# Terraform

- «**terraform destroy**» komutu verdiğimizde oluşturduğumuz makinenin ortadan kaldırılmasını sağlayacak (yes yazılmasını isteyecek):

```
$ terraform destroy
aws_instance.example2: Refreshing state... [id=i-025a0f5bbe08abf5b]

Terraform used the selected providers to generate the following execution plan.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

# aws_instance.example2 will be destroyed
- resource "aws_instance" "example2" {
  - ami                                = "ami-0989fb15ce71ba39e" -> null
  - arn                                = "arn:aws:ec2:eu-north-1:105150361200:instance/i-025a0f5bbe08abf5b" -> null
  - associate_public_ip_address       = true -> null
  - availability_zone                  = "eu-north-1a" -> null
  - cpu_core_count                     = 1 -> null
  - cpu_threads_per_core               = 2 -> null
  - disable_api_stop                   = false -> null
  - disable_api_termination            = false -> null
  - ebs_optimized                      = false -> null
  - get_password_data                  = false -> null
  - hibernation                        = false -> null
  - id                                 = "i-025a0f5bbe08abf5b" -> null
  - instance_initiated_shutdown_behavior = "stop" -> null
  - instance_state                     = "running" -> null
  - instance_type                      = "t3.micro" -> null
  - ipv6_address_count                 = 0 -> null
  - ipv6_addresses                     = [] -> null
  - monitoring                         = false -> null
  .....
}
```

<https://github.com/alfonsof/terraform-aws-examples/blob/master/code/01-hello-world/main.tf>

# Terraform

- «**terraform destroy**» komutu verdiğimizde oluşturduğumuz makinenin ortadan kaldırılmasını sağlayacak (yes yazılmasını isteyecek):

```
.....  
- volume_size      = 8 -> null  
- volume_type      = "gp2" -> null  
}  
}
```

Plan: 0 to add, 0 to change, 1 to destroy.

Do you really want to destroy all resources?

Terraform will destroy all your managed infrastructure, as shown above.

There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

```
aws_instance.example2: Destroying... [id=i-025a0f5bbe08abf5b]  
aws_instance.example2: Still destroying... [id=i-025a0f5bbe08abf5b, 10s elapsed]  
aws_instance.example2: Still destroying... [id=i-025a0f5bbe08abf5b, 20s elapsed]  
aws_instance.example2: Still destroying... [id=i-025a0f5bbe08abf5b, 30s elapsed]  
aws_instance.example2: Destruction complete after 30s
```

Destroy complete! Resources: 1 destroyed.

<input type="checkbox"/>	Name ▾	Instance ID	Instance state ▾	Instance type ▾	Status check	Alarm status	Availability Zone ▾	Pub
<input type="checkbox"/>	-	<a href="#">i-04b05bd9fa285682a</a>	⊖ Terminated ⓘ	t3.micro	-	No alarms +	eu-north-1a	-
<input type="checkbox"/>	-	<a href="#">i-04891555dc29302e9</a>	⊖ Terminated ⓘ	t3.micro	-	No alarms +	eu-north-1a	-
<input type="checkbox"/>	-	<a href="#">i-025a0f5bbe08abf5b</a>	⊖ Terminated ⓘ	t3.micro	-	No alarms +	eu-north-1a	-
<input type="checkbox"/>	-	<a href="#">i-06ce5b5c6509c0759</a>	⊖ Terminated ⓘ	t3.micro	-	No alarms +	eu-north-1a	-
<input type="checkbox"/>	my-ec2-instance	<a href="#">i-05630781a104a8e41</a>	⊖ Terminated ⓘ	t3.micro	-	No alarms +	eu-north-1a	-

<https://github.com/alfonsof/terraform-aws-examples/blob/master/code/01-hello-world/main.tf>

## Ne öğrendik?

- Terraform, komut satırında önceden yapmamız gereken bir çok işi kendisi tamamlamakta ve o makineyi ayağa kaldırmaktadır.
- 7 satır kod HCL kodu yazarak bütün çalışmayı tamamladık ve sanal bir makineyi ayağa kaldırdık.
- **SORU:** Bu kadar basitse neden Terraform kullanılmıyor?
- **SORU:** Daha bir çok parametre değerini girmemiz gerekmiyor mu?
- **SORU:** Bu parametrelerin isimlerini nereden öğreneceğiz?



# Terraform

- Başka bir Terraform kodu oluşturalım ve Terraform ile EC2'de **tag** kullanarak sanal bir makine oluşturmaya çalışalım:

```
$ cat > main.tf
# Configure the AWS provider
provider "aws" {
  region = "eu-north-1"
}

# Create an EC2 instance
resource "aws_instance" "example3" {
  ami          = "ami-0989fb15ce71ba39e"
  instance_type = "t3.micro"

  tags = {
    Name = "terraform-example3"
  }
}

$ terraform apply
```

<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type
<input type="checkbox"/>	-	i-04b05bd9fa285682a	Terminated	t3.micro
<input type="checkbox"/>	-	i-04891555dc29302e9	Terminated	t3.micro
<input type="checkbox"/>	-	i-025a0f5bbe08abf5b	Terminated	t3.micro
<input checked="" type="checkbox"/>	terraform-example3	i-08e2414916d417824	Running	t3.micro
<input type="checkbox"/>	-	i-06ce5b5c6509c0759	Terminated	t3.micro
<input type="checkbox"/>	my-ec2-instance	i-05630781a104a8e41	Terminated	t3.micro

Terraform used the selected providers to generate the following execution plan.

Resource actions are indicated with the following symbols:

+ create

Terraform will perform the following actions:

```
# aws_instance.example3 will be created
+ resource "aws_instance" "example3" {
  + ami          = "ami-0989fb15ce71ba39e"
  + arn          = (known after apply)
```

# AWS kaynakları için etiketler

- AWS, oluşturulan kaynaklara etiket eklenmesini, kaynakların izlenmesi ve erişim kontrollerinin daha iyi sağlanması açısından uygun bir strateji olarak tanımlamaktadır.

```
provider "aws" {  
  profile = "default"  
  region  = "us-north-1"  
  
  default_tags {  
    tags = {  
      Environment      = "Test"  
      Service          = "Example"  
      HashiCorp-Learn = "aws-default-tags"  
    }  
  }  
}
```

# Terraform'un Çıkış ve Giriş özellikleri

- Terraform, bazı değişkenleri dosyadan okuyabilir ve çıktıları da dosyaya yazabilir.
- Aşağıdaki iki dosyayı dizinimizde oluşturuyoruz.

## – output.tf

```
# Output variable: Public IP address
output "public_ip" {
  value = "${aws_instance.example4.public_ip}"
}
```

## – vars.tf

```
# Input variable: server port
variable "server_port" {
  description = "The port the server will use for HTTP requests"
  default = "8080"
}
```

# Terraform'un Çıkış ve Giriş özellikleri

- En son **main.tf** dosyasını oluşturuyoruz:

```
# Configure the AWS provider
provider "aws" {
  region = "eu-north-1"
}

# Create a Security Group for an EC2 instance
resource "aws_security_group" "instance" {
  name = "terraform-example-instance"

  ingress {
    from_port = "${var.server_port}"
    to_port   = "${var.server_port}"
    protocol  = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }
}

# Create an EC2 instance
resource "aws_instance" "example4" {
  ami           = "ami-0989fb15ce71ba39e"
  instance_type = "t3.micro"
  vpc_security_group_ids = ["${aws_security_group.instance.id}"]

  user_data = <<-EOF
  #!/bin/bash
  echo "Hello, World" > index.html
  nohup busybox httpd -f -p "${var.server_port}" &
  EOF

  tags = {
    Name = "terraform-example4"
  }
}
```

Oluşturulacak makinede 8080  
numaralı portu açıyoruz

# Terraform'un Çıkış ve Giriş özellikleri

- «**terraform apply**» komutunu verdiğimiz zaman, EC2 makineyi oluşturacak ve çıktı olarak makinenin IP'sini verecektir.

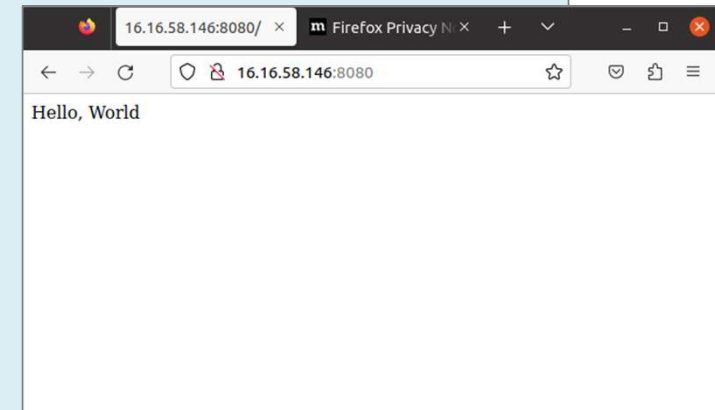
```
$ terraform apply

# aws_instance.example4 will be created
+ resource "aws_instance" "example4" {
  + ami                  = "ami-0989fb15ce71ba39e"
  + arn                  = (known after apply)
  + associate_public_ip_address = (known after apply)
  + availability_zone     = (known after apply)
  + cpu_core_count        = (known after apply)
  + cpu_threads_per_core  = (known after apply)
  + .....
Enter a value: yes

aws_security_group.instance: Creating...
aws_security_group.instance: Creation complete after 3s [id=sg-0737674148442f742]
aws_instance.example4: Creating...

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:
public_ip = "16.16.58.146"
```



Oluşturulan makinenin  
IP'sini gösterecek.

## Deneme

- «**main.tf**» dosyasında yeni bir «**ingress**» alanı eklemek ve SSH portunu da açsak acaba sisteme giriş yapabilir miyiz?
- SSH ile girmeye çalışsak, maalesef giremeyeceğiz. Çünkü açık ve gizli anahtarlarımızı belirtmedik ve sunucuya açık anahtarın yüklenmesini sağlayamadık.
- Terraform'un bir anahtar çifti oluşturmasını sağlayarak, açık olanın sanal makineye aktarılmasını sağlayacağız.
- Yoksa şöyle bir mesaj alırız:

```
$ ssh -i my-key-pair.pem ubuntu@13.53.112.153
The authenticity of host '13.53.112.153 (13.53.112.153)' can't be established.
ECDSA key fingerprint is SHA256:SqisSdJXy81yon4Lb+kMhbUiR4P79My9uryPJLvcIek.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '13.53.112.153' (ECDSA) to the list of known hosts.
ubuntu@13.53.112.153: Permission denied (publickey).
```

# Örnek: döngüler

- Terraform'u üç adet IAM kullanıcısı oluşturmak için kullanacağız.

main.tf

```
# Configure the AWS provider
provider "aws" {
  region = "eu-north-1"
}

# Create an IAM user
resource "aws_iam_user" "example5" {
  count = "${length(var.user_names)}"
  name = "${element(var.user_names, count.index)}"
}

# Data source: IAM policy document
data "aws_iam_policy_document" "ec2_read_only" {
  statement {
    effect = "Allow"
    actions = ["ec2:Describe*"]
    resources = ["*"]
  }
}

# Create an IAM policy
resource "aws_iam_policy" "ec2_read_only" {
  name = "ec2-read-only"
  policy = "${data.aws_iam_policy_document.ec2_read_only.json}"
}

# Create an IAM user policy attachment
resource "aws_iam_user_policy_attachment" "ec2_access" {
  count = "${length(var.user_names)}"
  user = "${element(aws_iam_user.example5.*.name, count.index)}"
  policy_arn = "${aws_iam_policy.ec2_read_only.arn}"
}
```

outputs.tf

```
# Output variable: ARN
output "neo_arn" {
  value = ["${aws_iam_user.example5.*.arn}"]
}
```

vars.tf

```
# Input variable: user names
variable "user_names" {
  description = "Create IAM users with these names"
  type        = list(string)
  default     = ["neo", "trinity", "morpheus"]
}
```

Üç tane yeni kullanıcı  
oluşturduk.

```
$ terraform apply
.....
Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

neo_arn = [
  "arn:aws:iam::105150361200:user/neo",
  "arn:aws:iam::105150361200:user/trinity",
  "arn:aws:iam::105150361200:user/morpheus",
],
]
```

# Terraform'da değişkenler

- «string», «boolean», «list» ve «map» cinsinden değişkenler oluşturulabilmektedir.

```
# Input variable: user names
variable "isim" {
  description = "Kullanici ismi"
  type        = string
  default     = "hakan"
}
```

```
variable "uygulama" {
  description = "kurulacak programlar"
  type        = list(string)
  default     = ["wget", "curl", "openssh-server"]
}
```

`vars.uygulama[0]` olarak erişilebilir.

```
variable "imajlar" {
  description = "imajlar ve karsiliklari"
  type        = map
  default     = {
    "istanbul" = "ami-0989fb15ce71ba39e"
    "ankara"   = "ami-09abs857342423acb"
  }
}
```

`vars.imajlar["istanbul"]` olarak erişilebilir.



# Terraform'da değişkenler

- «**terraform**»u çalıştırırken değişkenler de opsiyon olarak işe eklenebilir.
- Örneğin, «**image\_id**» diye bir değişken kullanılıyorsa, şöyle set edilebilir:  

```
$ terraform apply -var="image_id=ami-0989fb15ce71ba39e"
```
- Liste olarak gönderilecek bir değişken şöyle tanımlanır:  

```
$ terraform apply -var='apps={"wget","curl"}'
```
- İstenirse, değişkenlerin tanımlandığı dosyanın ismi verilebilir:  

```
$ terraform apply -varfile="degisken.tf"
```
- BASH kabukta TF\_VAR\_image\_id denilerek değişken değeri verilir.  

```
$ export TF_VAR_image_id="ami-0989fb15ce71ba39e"
```

## Terraform'da Veri Kaynakları

- Terraform'da kaynaklar (Resources) bazı işlerin yapılabilmesi için kullanılan özelliklerdir.
- Bir de veri kaynağı olarak görülen yapılar da bulunur. Terraform tarafından yönetilmeyen ama dinamik veri sağlayan veri kaynakları yaygın olarak kullanılırlar.
- Değişkenler statik bilgi sağlarken veri kaynakları uyarlanabilir ve filtrelenebilen veri öbekleri sunarlar.
- Dış veri kaynakları JSON formatında veri sunarlar ve bu veriler içinde istenen değer «resources» bölümünde kullanılabilir.

# Terraform Örnekler

- Önceki örneklerimizde imaj ID'sini kendimiz tespit ettik ve kenara not ettikten sonra TF dosyalarına yazarak kullandık.
- Ancak, bu imajlar devamlı güncellenmekte ve sürümleri değişmektedir.
- Ne yapabiliriz?
- Yapabileceğimiz iş şu: AWS'deki «**AMI Catalog**» içinde arama yapalım ve isminde Ubuntu geçen, HVM (Hardware Virtual Machine) olan ve SSD bulunan imajlardan en yenisini tespit edip onu kullanabilir miyiz?
- «**aws\_ami**» veri sağlayıcısını kullanarak, belirli arama kriterlerine göre istediğimiz imajın ID'sini öğrenip kullanabiliriz.

# İmaj ID'sinin otomatik olarak bulunması

- «**main.tf**» dosyasını şu şekilde kaydedelim.

```
data "aws_ami" "ubuntu" {
  most_recent = true

  filter {
    name      = "name"
    values    = ["ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*"]
  }

  filter {
    name      = "virtualization-type"
    values    = ["hvm"]
  }

  owners = ["099720109477"] # Canonical
}

resource "aws_instance" "web" {
  ami          = data.aws_ami.ubuntu.id
  instance_type = "t3.micro"

  tags = {
    Name = "HelloWorld"
  }
}
```

Arama kriterleri

# Azure'da sanal makinenin özellikleri

- AzureRM'da çalışan bir sanal makinenin ismi ve kaynak grubu verilerek, ID'sinin çıktı olarak ekrana yazılması sağlanabilir.

```
data "azurerm_virtual_machine" "example" {  
  name                = "myVM"  
  resource_group_name = "myResourceGroup"  
}  
  
output "virtual_machine_id" {  
  value = data.azurerm_virtual_machine.example.id  
}  
  
# name - Specifies the name of the Virtual Machine.  
# resource_group_name - Specifies the name of the resource group the Virtual  
Machine is located in.
```

Arama kriterlerinde  
makinenin ismi kaynak  
grubunun ismi verilir  
ve veri değerine erişilir.