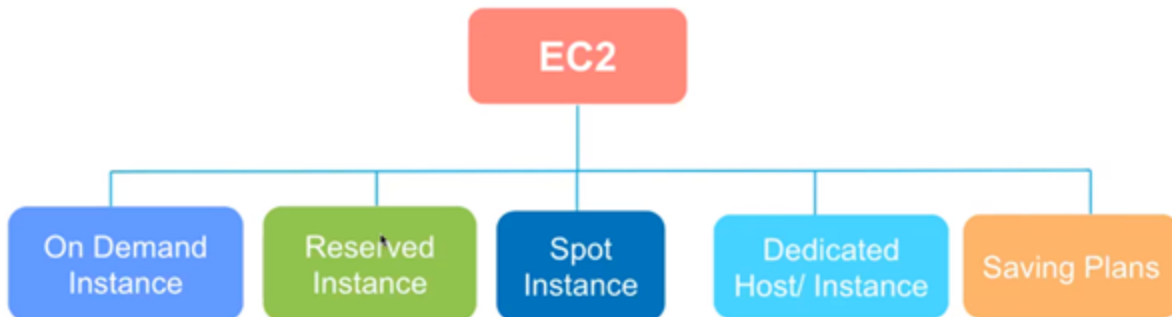




# EC2 Instances

## Pricing Model

### ► EC2 Instances Pricing Model of Instances



**On-Demand:** Geçici olarak açılıp kapatılan serverlardır. 1-2 saatlik kullanımlar için ideal

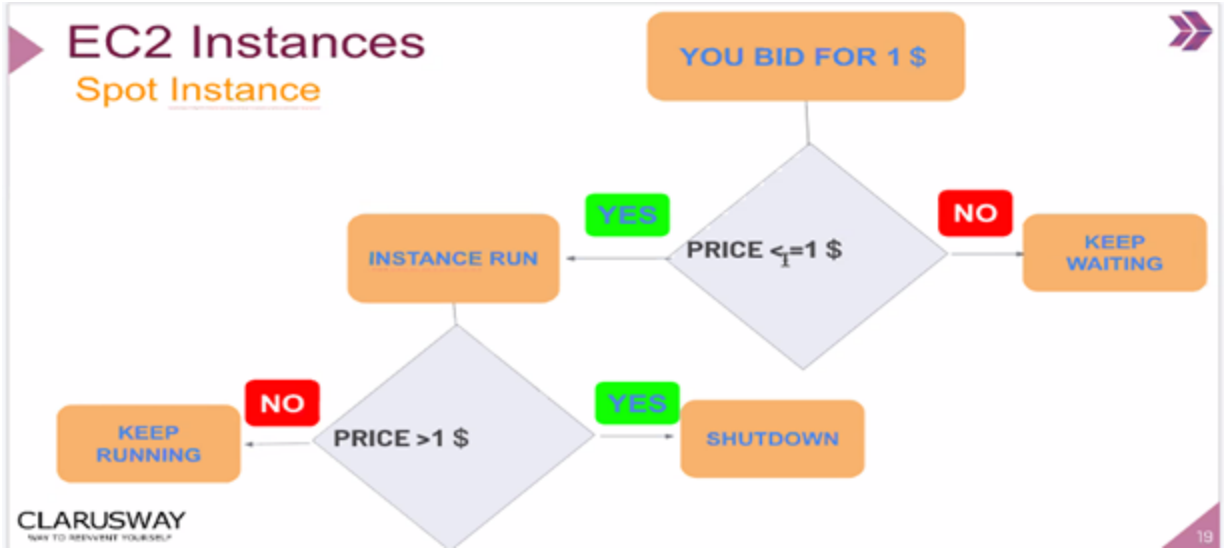
"spiky" dir yani kapatılınca bilgini kaybolması önemli değil. Dakika başına ücret ödenir.

**Reserved Instances (RI):** Toptan alımdır. 1 ya da 3 yıllık kiralanıyor. Örneğin 3 yıl boyunca fiyatını ödeyip server kiralyorsun. Hepsiburada vs gibi bir sitenin bunu kullanması mantıklı

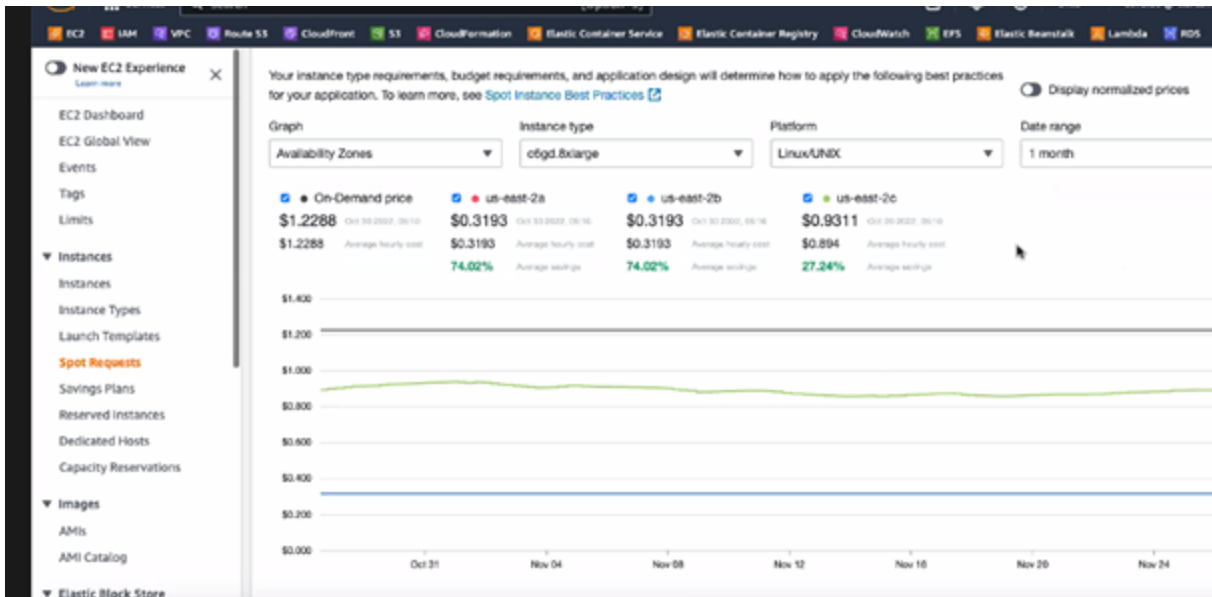
**Scheduled Reserved Instances:** Gün içinde belli saatler rezerve oluyor. Örneğin her gün saat 15-21 arası çalıştır gibi.

RI ile SRI kombine yapılabilir. Örneğin 5 tane RI 7/24 çalışsın, 5 tane de SRI mesai saatlerinde çalışsın gibi

**Spot Instance:** Borsa gibi çalışıyor. Bir fiyat belirlersin. Makinenin fiyatı 1,1\$ a gelirse makineyi çalıştır gibi. Bu ortada kalan o sırada kullanılmayan instancelar için geçerli. Eğer o instancenin fiyatı 1,5\$ olursa amazon bunları istediği zaman kapatıp senden alabilir.



Kapatılınca veri kaybının önemli olmadığı işlerde kullanırsın. Mining ya da herhangi bir test işlemi için kullanabiliriz ancak bir site ayağa kaldırmak için riskli bir yöntem.

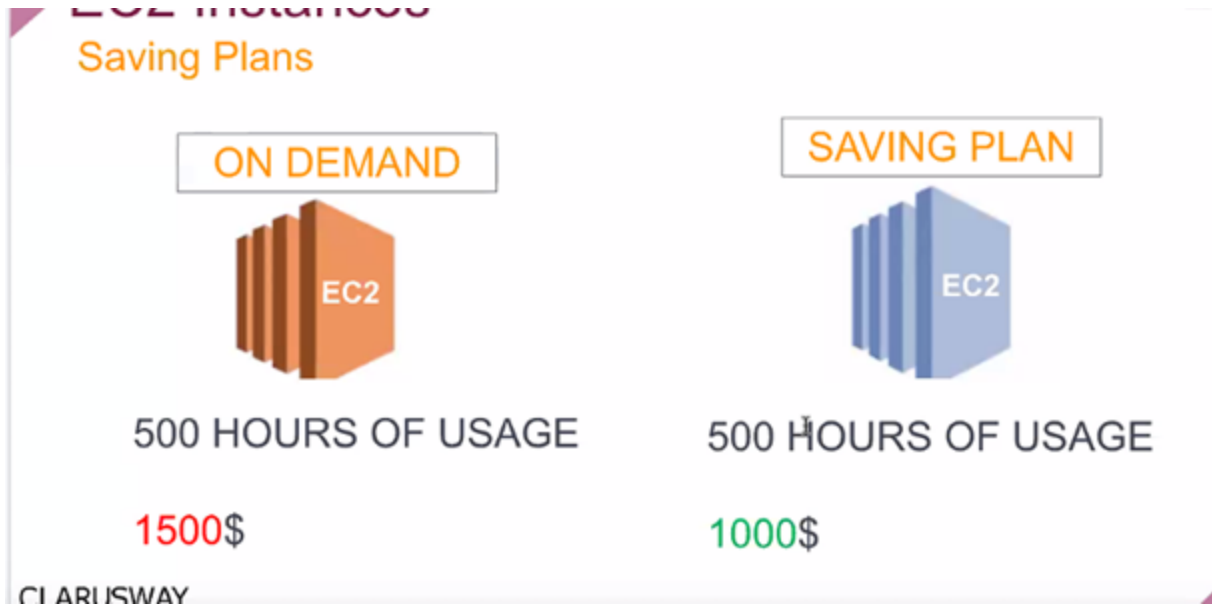


Yukarıda görüldüğü gibi On-Demand \$1.2 iken spotta ücret \$0.31 çıkıyor. Ancak amazon için öncelik On-demand tır. Yani kendi serverları yetersiz kalırsa senden spotu geri alıp kendi kullanır vs.

Spot instance da AWS'den kaynaklı bir hata olursa ve senin makineni kapatırsa ilk saat ücret almıyor. Ancak On-Demand da böyle bişey yok her türlü paraayı ödersimn. Ancak sadece ilk saat için geçerli


**Dedicated Host/Instances:** Belirli Lisanslı ürünleri kurmak için özel serverlar kiralaman gerek. Örneğin Oracle ... programı kurmak için. Reserved den farkı şu: Burada server ın belli. O server hep sana tahsisli. Kullansan da kullanmasan da. Ancak RI'da orada bir server ın var ama hangisi olduğu belli değil.

**Saving Plan:** Kontrollü kullanım. 500 saatlik kullanım hakkı alıyorsun. Kullandıkça hakkın azalıyor




Özetlersek;


## EC2 Instances Recap



**On Demand**


**Spot**





**Reserved**

**Saving Plan**



Purposing Model: Kullanım amacına göre server özellikleri değişir.

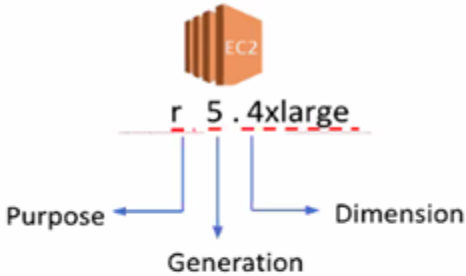
**General Purpose:** Ortalama bir server

**Compute Optimized:** High performance

**Memory Optimized:** Yüksek bellek kapasitesi

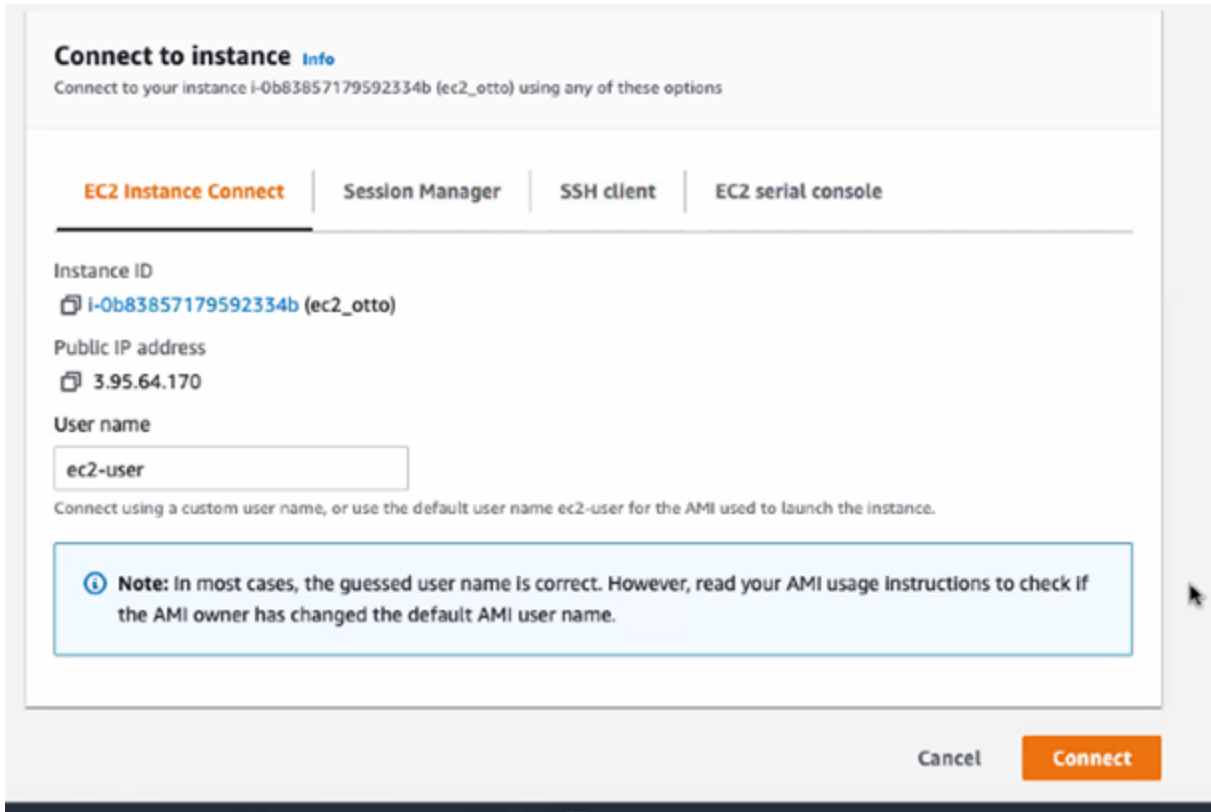
**Accelerated Computing:** Analiz vs için yüksek performans

## EC2 Instances Instance Coding



- **R** refers to its **purpose**. It means this EC2 is Memory Optimized instance.
- **5** refers to instance **generation**. For example, the last generation of the r-family is 5.
- **4xlarge** refers to **dimension** of instance. AWS has built servers of various sizes to suit every need in instance families. For example, the r5-family has 8 different sizes starting from **large** to **24xlarge**.
- Not all models have instances in every generation and size.

## EC2 Instance



**EC2 connect** Browser dan bağlanıyor ama çok sağlıklı değil.

**Session Manager** Uzaktan birçok instance a bağlanıp değişiklik yapmaya yarar. Key pair olmadan da bağlanılır.

**SSH client:** Ssh protokolü ile bağlanma

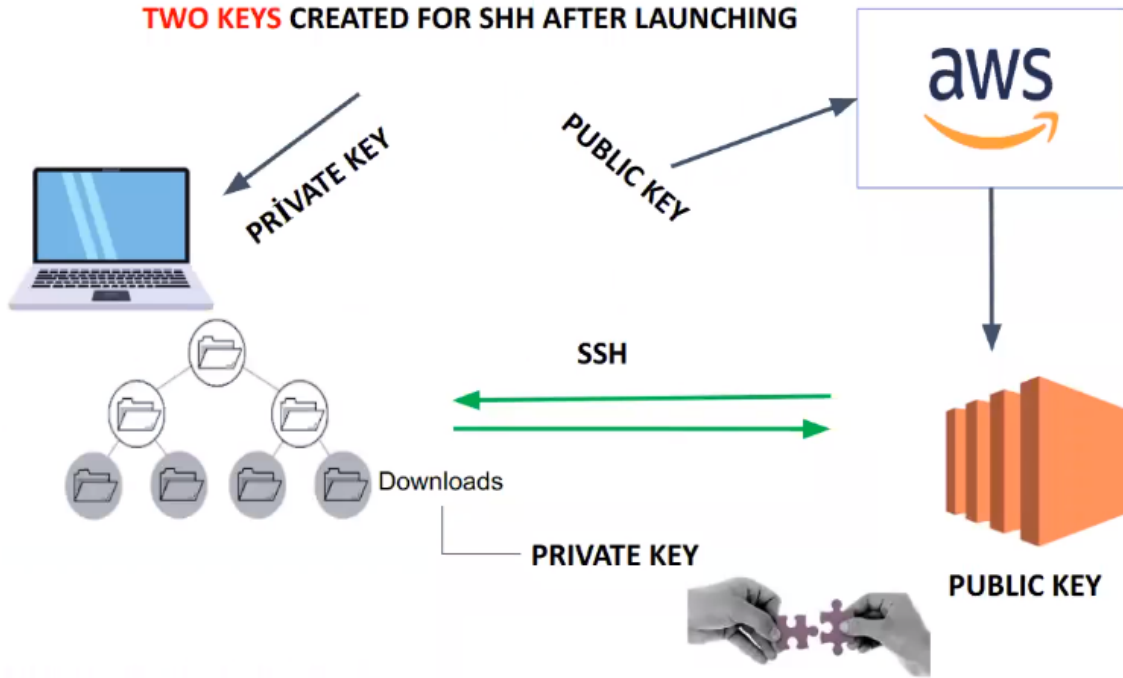
IAM = Identity & Access Management: Authentication ve Authorization işlerini yapar. Senin kullanıcı kimliğini tanıtır ve sana gereken yetkileri verir

Programmatic Access diye bişey var. Python Java vs ile yediğimiz kodu AWS in anlayacağı şekle çeviren bişey. Konsolu açmadan bu tarz kodlarla EC2 ayağa kaldırabiliriz veya konsola girmeden belli işlerimizi halledebiliriz. Bu kodları AWS ye uygunlaştıran program diyebiliriz. Bu şekilde terminalden yaptığımız işlemler AWS konsolundan yaptığımız işlemlerden daha kuvvetli. O yüzden burada root ile çalışmıyoruz. Kontrolsüz güç.

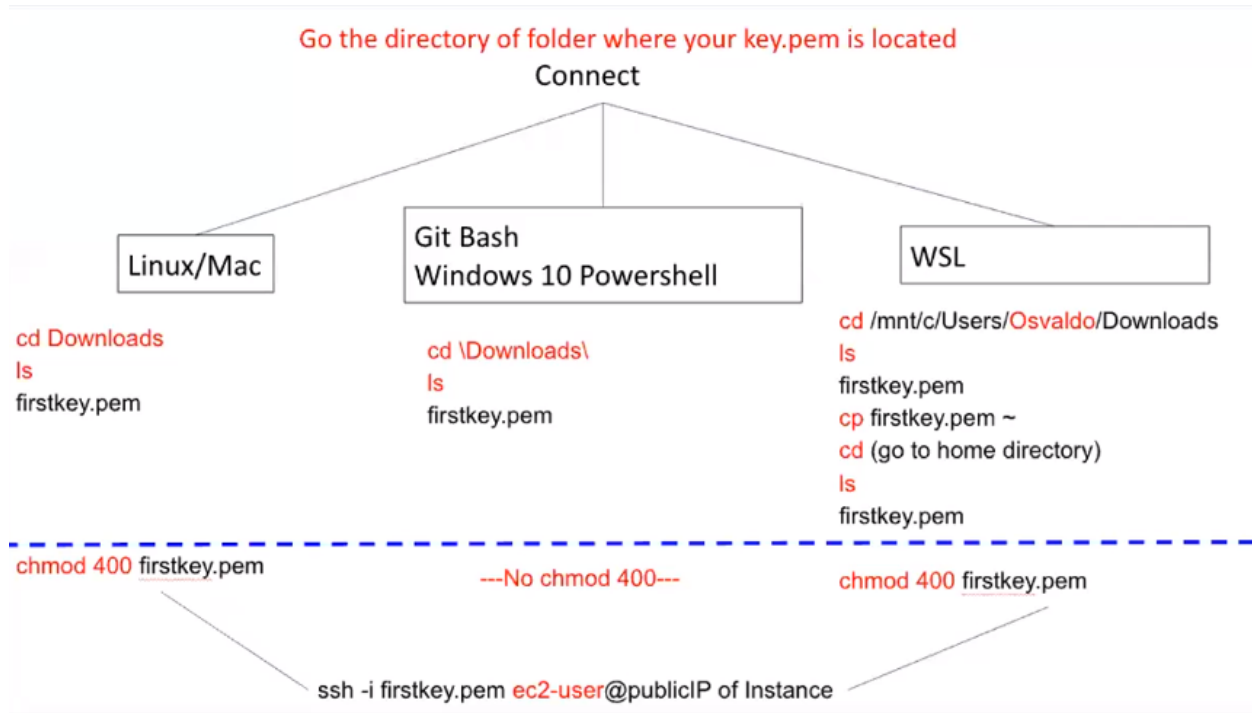
IAM Policies: JSON formatında yazılır. Denied üzerinden çalışır. Yani konsolda tüm yetkiler 0dır. İstediğin yetkiyi vermek lazım.

IAM Roles: Bir servisin diğeri bir servisle konuşması gerektiği zamanlarda kullanıyoruz. Örneğin bir EC2 bir database e bir şey yazmak istediğinde buna Role tanımlıyoruz.

SSH, secure bir bağlantıdır. SSH ile bağlanmak için bir pem keye ihtiyacımız var. Create instance menüsünden bunu oluşturabiliriz.



Key Pair, Public Key ve Private Key'den oluşur. Private keyi bilgisayarımıza indiririz. Her ayağa kaldırdığımız instance'ın içine de bu public key konulur. Daha sonra bağlanmak istediğimizde key çiftleri eşleştirilir, uyarsa bağlantı sağlanır.



Key pair oluşturdan sonra oluşan pem dosyasına sadece read yetkisi vermemiz gerekiyor. chmod 400 first-key.pem yazıyoruz. Bu linux ve mac için geçerli.

**Connect to instance** [Info](#)  
Connect to your instance i-07463ca4172b53cf9 (My-First-EC2) using any of these options

EC2 Instance Connect

Session Manager

**SSH client**

EC2 serial console

Instance ID  
i-07463ca4172b53cf9 (My-First-EC2)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is first-key.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.  
chmod 400 first-key.pem
4. Connect to your instance using its Public DNS:  
ec2-34-230-75-117.compute-1.amazonaws.com

Example:  
ssh -i "first-key.pem" ec2-user@ec2-34-230-75-117.compute-1.amazonaws.com

terminalde pem keyin bulunduğu klasöre gelerek aşağıdaki komutu yazmamız gerek

ssh -i "pem key" kullanıcı\_adı@Bağlanılacak\_bilgisayarın\_IP'si

```

  _ | _ | _ )
 _ | ( _ | /  Amazon Linux 2 AMI
__| \__|__|

https://aws.amazon.com/amazon-linux-2/
13 package(s) needed for security, out of 16 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-19-46 ~]$
```