**A. DATA ENGINEER**

1. Veri Tipleri:

Structured data: Verilerin tablo halinde tutulmasidir. Baglanti kurmak kolaydir.

Semi structured data: JSON, XML, YAML, NoSQL gibi formatlari olur. Ornegin: kisi bazinda isim, soyisim ve sevdigi artisler olabilir. Artis sayilari ise farkli sayida olabilir. Patern izlerler.

Unstructured data: Herhangi bir tabloda yer almazlar. Incelemesi daha zordur. Text, sound, pictures veya video dosyalarinda olur. Verilerin cogu unstructured durumdadir.

1. SØL for Data Engineers:
   1. Create, Maintain and Update Table:

CREATE TABLE employees (

employee\_id VARCHAR(255),

first\_name VARCHAR(255),

last\_name VARCHAR(255),

role VARCHAR(255),

team VARCHAR(255),

full\_time BOOLEAN,

office VARCHAR(255)

):

b. Eger Data Scientist olusturulan tabloda sorgulama yaparsa:

SELECT first\_name, last\_name

FROM eemployees

WHERE role LIKE «%Data%»:

1. Database: Data Lakes and Data warehouse.
   1. Data lake:
      1. Stores all the row data,
      2. Cok buyuktur,
      3. Structured, semi-structured ve unstructured data bulunur,
      4. Uygun maliyetlidir,
      5. Difficult to analyze, (deep learning ile patern cikarilabilir.)
      6. Requires an up-to-date data catalog require,
      7. Used by data scientists for real time analytics, and engineers.
   2. Data warehouse:
      1. Stores specific data for spesific use
      2. Gorece daha kucuktur,
      3. Stores mainly structured data,
      4. More costly to manipulate,
      5. Optimized for data analysis,
      6. Used by data analytyts and businest analyts,
      7. Ad-hoc, real-only queries,

1. Data catalog for data lakes:
   1. It’s a source of truth that compensates for the lack of structure in a data lake.
   2. What is the source of this data?
   3. Where is the data used?
   4. Who is the owner of the data? (responsible for maintaining it)
   5. How often is this data updated?
   6. Good practice in terms of data governance,
   7. Ensures reproducibility,
   8. No catalog - - > data swamp (data catalog data lake’in cope donmesine engel olur),
   9. Good practice for any data storage solution,
      1. Reliability,
      2. Autonomy,
      3. Scalability,
      4. Speed.

1. Data processing:
   1. Converting raw data into meaningful information,
   2. Remove unwanted data, (verinin uzun donem incelemesi ile stabil oldugunu bulduktan sonra ihtiyac yoksa silinebilir)
   3. Optimize memory, process and network costs, (maliyetli oldugundan gereksiz kisimlar silinebilir)
   4. Convert data from one type to another, (Convert songs from .flac to .ogg)
   5. Organize data (Recognize data from the data lake to data warehouse)
   6. To fit into a schema/structure (employee table example)
   7. Increase productuvity (enable data scientists),

1. How Data Engineers process data:
   1. Data manipulation, cleaning, and tidying tasks: (rejecting corrupted files, deciding what happens with missing metadata)
      1. That can be automated,
      2. That will always need to be done,
   2. Store data

1. Scheduling:
   1. Can apply to any task listed in data processing
   2. It holds each piece and organize how they work together,
   3. Runs tasks in a spesific order and revolves all dependencies,
   4. It’s the glue of your system.
      1. Manual, time and sensor scheduling:
         1. Manually
         2. Automatically run at a specific time
         3. Automatically run at a specific condition
            1. Sensor scheduling,
2. Ingesting data: Batches and streams. Apache Airflow ve Luigi schedule icin kullanilir.
   1. Batches:
      1. Group records at intervals/siklik
      2. Often cheaper (employee table gibi acil olmayan data updatinin mesai disina schedule edildigi icin)
   2. Streams:
      1. Send individual records right away (herbir fatura anlik islenir)

1. Parallel Computing:
   1. Basis of modern data processing tools
   2. Necessary:
      1. Mainly because of memory
      2. Also for processing power
   3. How it works:
      1. Split tasks up into several smaller subtasks (when working with big data)
      2. Distribute these tasks over several computers
   4. Benefits and risks:
      1. Advantages:
         1. Extra processing power
         2. Reduced memory footprint
      2. Disadvantages:
         1. Moving data incurs a cost
         2. Communication time

1. Server isyerinde de olabilir cloud ortaminda da. Ayrica ayni anda birden fazla cloud ile calismaya multicloud denir.

**AWS**

1. **Uc ana storage sistemi vardir:**
   1. **Hard drives are block-based storage systems:** OS sees a volume and then the volume is partitioned and formatted. Ornegin 1000GB’lik volume varsa C: 800GB ve D: 200GB seklinde bolebilir, daha sonra file system (Windows icin Fat32 veya NTFS) secip format atilir. Daha sonra OS yuklenebilir.
   2. **NAS (Network-attached-storage):** NIC (network interface card) vasitasiyla network switch’e oradan da network attached storage server (NAS)’a ulasilir. Server onceden partition, format, file system islemleri tamamlanmistir. NAS shares filesystems over the network. Bilgisayariniz icin mapped network drive konumunda olur. OS de NAS’daki file system’i gorur.
   3. **Object Storage System:** Kullanici bir objeyi web browser’a yuklerse internetteki bir object storage container’a kaydedilir. Bunun icin http protokolu kullanilir. (http protocol is used with a REST API (e.g. GET, PUT, POST, SELECT, DELETE)). Web browserden herhangi bir text/resim dosyasi indirirken http GET request, upload ederken ise PUT veya POST request yapilir. Container’de herhangi bir hirarsi yoktur. File system olan hard drives veya NAS’ta bolum, ay, kisi … bazinda dosya sistemi kurulabilirken object storage system’de kurulamaz. Tum dosyalar ayni seviyededir.

1. 192.168.0.1: 192.168.0.1/24 = 255.255.255.0 subnet maski var demektir. Toplamda her sayi 8 binary formatta yazilir. 3x8 adedi network ID demektir.
   1. 192.168.0 > Network ID
   2. 1 > Host ID
   3. 255.255.255.0 > Subnet Mask

1. Container: OS’nin uzerine kurulurlar ve bunlarin uzerine ayri ayri farkli programlar kurulur. Her biri bagimsiz olark calisir. Cok hizli acilirlar ve kullanilacak program kadar ram kullanirlar.
   1. Dockers
   2. Kubernets

1. API (Application Programming Interfaces): Iki ayri programi konusmasini saglar.
   1. Restful API (Representational state transfer): http protocolu kullanir. Client calisirken, http protocolu kullanarak talimatlari gonderir. Daha sonra API backend services (web site, application, database…) ile kendi dilleriyle gelen talimatlari iletecektir. Bu servislerin herbirinin kendi dilleri vardir. Ancak API bu servisler ile nasil konusulacagini bilir. API’ye talimatlar gondererek cloud uzerinden farkli servislerle konusulabilir. Cloud computing API’s uzerine insa edilmistir.

**B. AWS CLOUD**

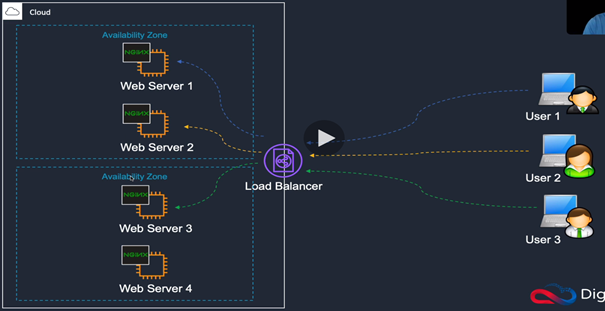
1. **Cloud Computing Service Models:**
   1. **Private Cloud**: Kendi serverlarinizi bulundurmanizdir. Hypervisor, OS, Application Framework, Data, Applications un herbirinin kontrolu bizdedir.
   2. **Infrastructure as a Service (IaaS)**: Sizin icin server gorevini gorup VMs ile OS ve applicationlarinizi calistirabilirsiniz. Hypervisor ve infrastructure bulunmaz. VMs’leriniz olur.
      1. Amazon Elastic Compute Cloud (EC2)
      2. Azure Virtual Machines
      3. Google Compute Engine
   3. **Platform as a Service (PaaS):** Burada OS ve Java framework (Java Runtime) AWS tarafindan calistirilir. Siz sadecxe Java WebApp kodunu ve datanizi kullanirsiniz. Tek dusunmeniz gereken dataniz ve application codunuz (developerlar icin)’dur.
      1. AWS Elastic Beanstalk
      2. Azure WebApps
      3. Compute App Engine
   4. **Software as a Service (SaaS):** Hicbirseyi kontrol etmek zorunda degilsiniz. Application size saglaniyor. Yalnizca Java WebApp (application code) yuklersiniz.
      1. Google Apps
      2. Salesforce.com
      3. Zoom

1. **Cloud Computing Deployment Models:**
   1. **Private Cloud:** Kendi serverlarinin bulunmasi ve data center’da applicationlarin yuklu olmasi. VMware, Microsoft, Redhat, Openstack. Kendi VMs’leriniz, Oss, software management olmali. Data Center’da VMs, Storage&Backup, Network&Firewall kurulup uzerine de Self Service Portal/Secice Catalog, Automation and Configuration Management, Billing and Reporting ve Multi-tenancy controller kurularak cloud olusturulur.
   2. **Public Cloud**: AWS, Azure, Google Cloud vb servislerin cloudunu kullanmaktir.
   3. **Hybrid Cloud:** Private ve Public Cloud’u ayni anda kullanmaktir.
   4. **Multicloud:** Ihtiyaca uygun olarak ayni anda hem private cloud hem de farkli servisleri cesitli amaclarla kullanmaktir.

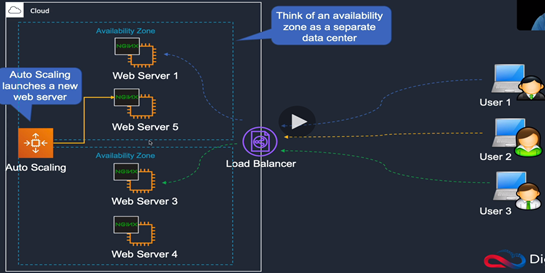
1. **Statefull and Stateless Services:** 
   1. **Stateless application**: F.ex: a news website (person check a news website). Login olmadan ve kendisiyle ilgili veri olmayan siteye girip sadece articles okursun. Ilerleyen saatlerde tekrar girdiginde de yine genel olarak ayni haberleri okur. Customized degildir. Web server birden fazla olabilir ve musteriyi musait olana gonderir cunki musterinin verisi yoktur. Her serverdan gazetesini okuyabilir.
   2. **Stateful application**: Login olarak Netflix’e girip bir film seyredilir. Netflix de musterinin neyi izledigini kaydeder. Burada kullanicinin bilgileri user’s session’a kaydedilir. Ne izledigini, behaviorlarini ve ne tarz programdan hoslandigini bilir. Bu bilgiler isiginda sana baska programlar tavsiye eder. Nasil isler:
      1. Database server, application server ve web server back end’de bulunur. Musteri bilgisayarindan browser araciligi ile alisveris yapar. Sectigi urunler cart’a eklenir ve cookies’e kaydedilir.
      2. Web serverda herhangi bir data depolanmayabilir bu islemle ilgili. Bu stateless olur. Veri sadece user’s computer’de saklanir.
      3. User urunu alacaksa Application server/layer talebi gerceklestirecek ve veriyi Database Server’a kaydedecektir. Ayrica login olundugu icin Application Server user’i ID’si uzerinden taniyacaktir. Ne aldigini bilecek ve cesitli tavsiyelerde bulunabilecektir.

1. **Scalability and Elasticity:** 
   1. **Scaling up:** Adding hardware resources to the server/VM. Daha fazla islemci/RAM vb ihtiyac halinde yapilir.
   2. **Scaling out**: Adding additional VM. Bir application’i birden fazla VM’de calistirmak icin yapilir. Gerekirse baska hardware uzerine de yeni VMs kurularak applicationun kullanim VMs’ini scale out etmis oluruz. Stateless services icin uygundur. Ornegin web applicationu calistirirken ve data baska bir yerde tutuluyorsa bircok VM demek ayni anda bircok musteriye hizmet demektir.

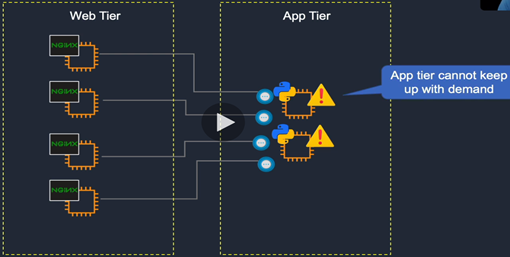
1. **Load Balancer**:
   1. Scaling out ile bircok VM calisir. Ornegin NGNX applicationu ayni anda bircok VM’de calisir. Ayni anda bircok client webten baglanabilir. You can keep scaling the number of web servers you have in order to accommodate (uygun hale getirmek) the load. Peki trafigi VM’lere nasil yonlendirecegiz.
   2. Normalde DNS name ve Domain name’imiz var ornegin: Amazon.com (197.2.34.1). Load balancer’imiz olur ve bir DNS name’i ve IP address’i bulunur.
   3. When traffic reaches the load balancer, it can then distribute it to different instances (VM) behind it. Client load balancer ile irtibata gecer. Load balancer ise talebi back end instances’e gonderir. Her bir client farkli bir instance/web server’a yonlendirilir.
   4. Trafigi bostaki web serverlara yonlendirir.
   5. Eger herhangi bir server cokerse client, load balancer tarafindan rerouted to another web server.

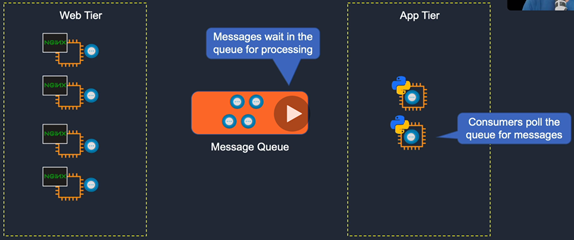


1. **Fault Tolerance:** 
   1. Serverda olusabilecek hardware arizalarina karsi harddisk, network card… vb sisteme yedek olarak eklenmesidir. Fault tolerance is protecting you from individual component failure.
   2. **Availibility zone**: Isolated data center demektir. Web serverleri ihtiva eden data serverlardir. Bu ornekte web server 1/2, bir availibility zone’da web server 3/4 ise baska bir availibility zoneda. Eger availability zone hata verir cokerse diger multiple availability centerlar calismaya devam edecektir. Bizim 4 Web Serverimiz iki Availibility Center’da bulunuyor. Bir Availibility Zone bir veya birkac data center’dan olusur.
   3. **Auto scaling**: web serverlerin calisma/hata verme durumlarini veya client yogunlugunu da takip ederek yeni web serverlar baslatir. Aizali instance yerine yeni nir instance baslatir.Yogunluk durumlarinda da gerekli miktar instances’i calistirir.

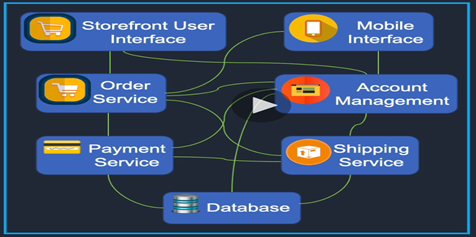


1. **Loose Coupling:**
   1. Tight Coupling, Loose Couplingin tersidir. Web tier’de web server ve applicationu var. App Tier’de what is going to process information bulunur. Clientlerden web servere gelen veriler App tier tarafindan islenir. Web service directly App tiere baglidir. Eger client verileri hizli akmaya baslar ve App tier verileri islemekte zorlanirsa bazi mesajlari alamayabilir veya kacirabilir. Ornegin musteri talepleri. Bunun cozumu icin Web tier ile App tier arasina Message Queue yerlestirilir. Bu durumda Web tierden mesajlar Message Queue’ye iletilir. Artik App Tier islem yaptikca mesajlari Message Queue’den ceker ve overload olmaz ve mesajlar kaybolmaz. Message Queue’nin kullanilmasi Loose Coupling’tir. Boylece Loose Coupling’te Web Tier ile Message Queue ve Message Queue ile de App Tier integrated olur. Message Queue tampon vazifesi gorur. App tier’de islem yapilirken Bir App tier cokerse mesage Message Queue’de yeniden belirir ve baska bir App tier’e gonderilir.

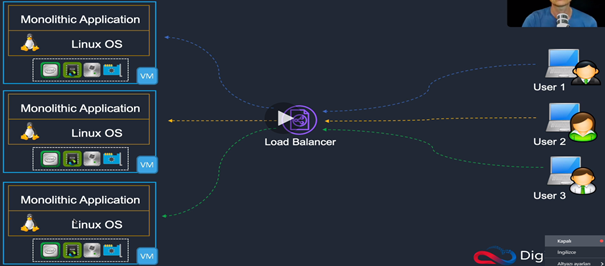




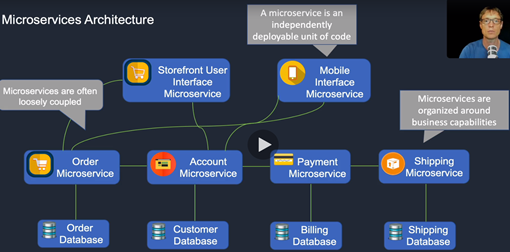
1. **Monolithic and and Microservices Architectures**:
   1. **Monolithic Application**: An application in blue box and is running o none server or multiple servers, but it has multiple components:
      1. Storefront User Interface that could be some kind of web service that your users can interact with and place orders.
      2. Mobile Interface for if they’re using their mobile clients to access your website.
      3. Order Service might be when they’ve actually added smtg to their cart and purchased it and it’s going to process that information for them.
      4. Account management is for costumers might log in to check their accounts, to update their details or see the progress or their orders.
      5. Payment Service is your payment gateway. So when they place an order, it has to talk to the payment service.
      6. Shipping Service is used for trasfer.
      7. Database is for storing lots of information.
      8. There are lots of different ways that these different components are talking to each other. And there could be more components. There is significant limitation for this model. In this model we combine what we call the user interface. So the component that the users actually see, they log into a website or they access a website or a mobile interface. Thr business logic. So a lot of the processing of orders, the account management (CRM layer) and that kind of thing. And then the data access layer. So the actual place where the data is stored about that costumer and about the orders they place and the statisitcs and information that we might record. So it’s all combined on a single platform.
      9. If you have to update any single components, so these could be very different pieces of codes, they could even have different development teams working on them and somebody might need to update the mobile interface. Now if you do that, you may well have to take down the whole application. Similarly, if an individual service fails, it can cause a problem for the entire application. Now, as i mentioned, you have multiple instances of your application,



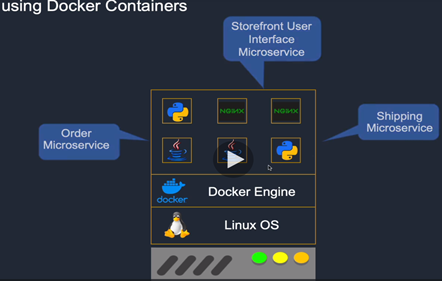
* + 1. But the application is all of those components of the application all running on each of your virtual instances. You might put a load balancer in front of it and you’ve got users connecting. So there is some protection against an individual virtual instance failing because you have multiple instances. But you have the same code running on all of these applications. So if you do some kind of updates or there is some kind of error occuring in them, it can take down the whole application.



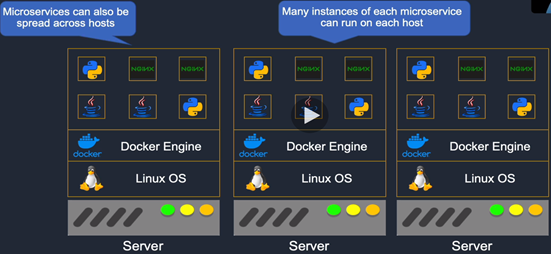
* + 1. So what’s another way of doing this? Rather than using a monolithic application, we can move to what we call a microservices architecture.
  1. **Microservices architecture**: We’re seperating component of our application. We will then often decouple them and connect them using APIs. We might put message passes in between so if we’ve got loose coupling.
     1. We may seperate the databases, so we have different databases for different components of the application to store information.
     2. Each of these is a seperate application that’s running and it can be deployed as what we call a micro service, and that’s an independently deployable unit of code.
     3. They’re often loosely coupled and they’re organized around business capabilities.



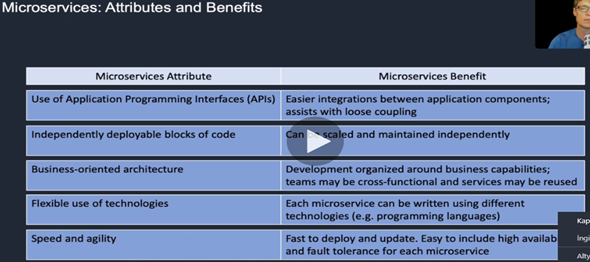
* + 1. How does it look like in Docker f.ex. Docker is used for hosting containers and containers are really useful for running your microservices applications. So, f.ex, each one of these containers could be a seperate microservice.



* + 1. And then you can scale that out to multiple containers across multiple Docker engines running on multiple physical hosts, which means you’ve got multiple instances of your application and you’ve seperated them from each other.
    2. So you could have multiple instances of each microservice on a single host or you can spread them across multiple hosts. So you have got lots of options. Containers are super fast to deploy, so if smtg fails, you can redeploy it much more quickly.
    3. They’re also low in terms of their usage of underlying resources.



* + 1. So let’s have a look at some of the attributes and benefits of microservices.
       1. In terms of loose coupling, it means you can integrate with things like a message pass using an API or a notification service to send information between your application components.
       2. Each microservice is a seperate block of code. Those can be scaled independently. So if one needs a lot more resources than some other component of your code, then it can scale without affecting the others or without causing any kind of constraints or issues on the other components.
       3. You can organize your development around your business capabilities.
       4. Each microsercice can be written using different technologies. So you might have different programming languages in use. Sometimes that can be difficult if you’re running on a single underlying operating system. With using microservices it becomes a lot easier.

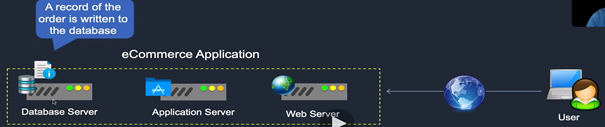


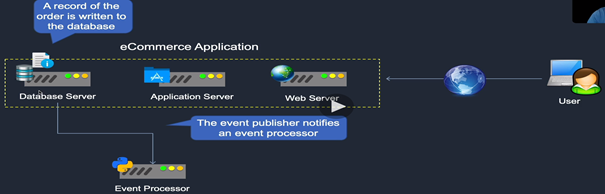
1. **Event-driven Architecture**:
   1. So, in this example, we have an event-driven architecture. So the user is shopping online and is going to place an order. The order goes through to an application server. The application server then processes that order, and it writes a record of the order to the database server. So, a record has been written Database Server.
   2. Now, what happens in an event-driven architecture is this event, this data being written to the database can cause something else to happen. So in this case, the event publisher, which in this example is the actual database server itself, sends a message to an event processor and says, "hey, I've just had a record written to my database. You might want to check it out."
   3. The event processor will then go to the database, find that record, and process it. And what does it do? Well, there's a myriad options. But in this particular case, maybe it just needs to process that information and save it in another database maybe for some kind of analytics or reporting that you're doing.
   4. So the point is, with an event driven architecture, your systems can respond to things that happen.
   5. An event-driven architecture is where when something happens, and it can be things like an order is placed, a database record is written, a file is uploaded to a storage system, then you want to do something when that happens. And so, it can automatically respond to those things happening.

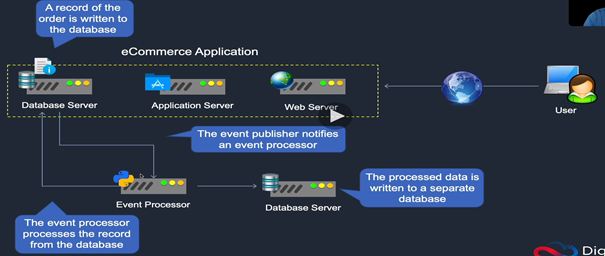






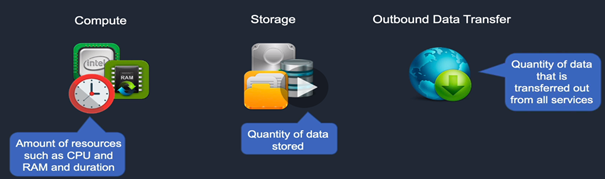






1. **AWS Global Infrastructure:** 
   1. **Region:** Farkli bolgelerde olan ve icerisinde en az 2 availibility zone bulunan bircok region vardir. That means, you can deploy your resources on AWS into a region that's closest to your users or wherever is that you need to run your own application or store your data. Bir Availibility zone’lar diger availibility zone’lardan bagimsiz ve ayri bir binadadir. Biri ariza yapsa digerini etkilemez.
   2. So each region is completely independent from other regions. Any kind of issues on one region are not going to affect any other region at all.
   3. At the moment, there are about 24 regions around the world and that does grow over time.
   4. Each region is also connected to each other via what's called the AWS Global Network. That is a very high bandwidth and low latency and very redundant network. So that means that there's a great network connection that doesn't use the Internet for sending traffic between regions. So, for example, you might have a database that's doing some kind of replication and that can be done across the global network at lower latencies or at least with more consistent performance than you would get if you were using the Internet.
   5. And there are some other bits as well called Direct Connect locations. We've got the Edge Locations and Regional Edge Caches, which are associated with the cloud front network. And there are 77 availability zones.

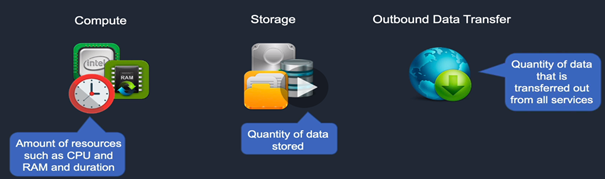
1. **AWS Pricing:**
   1. So there are three fundamentals of pricing on AWS.
   2. The first is you've got your compute.
   3. **Pay as you go:** So you get charged for the amount of resources you use and the amount of time that you use them. So, as an example, if you're running a virtual server instance on Amazon EC2, you pay for the amount of CPU and memory that you allocate, and the amount of hard drive space you allocate. And then you pay that rate for however long you're actually using that instance. And that's one of the main ways of paying.
      1. For storage, you're generally paying for the amount of data that you're storing. So, it's the actual quantity of data stored. In some cases, it's the actual amount of space you've allocated to store data on as well.
      2. The third fundamental of AWS pricing is data that is sent outbound. So let's say that you upload data to AWS. You might have a large volume of data and you upload that data to AWS. Maybe you put it in the Amazon S3 service, which is the object storage system on AWS. Now, you don't pay for uploading your data to AWS. There's no charge. That's inbound data. But when you then decide that you need to pull that data back down again, maybe it's backup data and you need to restore it. At that point you pay for the traffic that goes outbound. So, you pay for outbound data transfer.



* + 1. And there's a certain rate up to a certain quantity of data that you pay and then the rate might change a bit. So you've got to look at, you know, what the total aggregate is of data that you're sending outbound from all your services to be able to work out how much you're going to pay. So let's have a quick look on the AWS website to get a bit more detail on pricing. So this is the main AWS pricing page. If you just Google AWS pricing, this is what you're going to find. And I'll put the links by the way into the lessons so that you can actually access these materials directly as well.
    2. You're actually paying a rate for the amount of time that you use a resource.
  1. **Save when you reserve**: Another thing that you can do, though, is you can save money when you reserve. So in this case, it's no longer an on-demand or a pay as you go consumption model. It's more like you have a contract. So you say, well, I'm going to commit to running an Amazon EC2 instance for one year or three years. And because I contract to do so, I'm going to pay for that whole term. But I'll get a massive discount. And as you can see here, you can get up to 75% discount. So if you know that your workload, your application is going to run for a certain period of time, at least one year, for example, it might be better to actually reserve some capacity and get a big discount.
  2. **Pay less by using more:** So as you use more, for example, as you store more data, you'll find that you reach a threshold and then the rate of storing additional data gets lower. So you can achieve lower costs for certain services if you use more of them.
  3. **Spot instance**: Unused capacity on AWS is offered to you at a really low price. And so as long as you're workload's flexible, because when AWS need the capacity back again, they can take it and shut down your application.
  4. **Savings plan**: It is a kind of variation of that in which you can commit to a certain amount of usage, not specificly a certain amount of time. So you might have a certain amount of hours per day rather than just using it 24 hours a day for a year. So these are options where you can pay in a slightly different model to the consumption model.
  5. **Dedicated hosts**: There's other services where you might pay differently as well, such as dedicated hosts. So normally AWS is multi-tenant-cok kullanicili. So, your Amazon EC2 instances would be running on the same hardware as other customers.
  6. **Storage services**: One of them is Amazon S3. We can see that you pay for the amount of storage that you store. So, for the first 50 terabytes, you pay this rate. And then for the next 450, you pay a slightly lower rate.
  7. ***calculator.aws****: > create an estimate > bir service yaz veya sec > configure > Region sec > OS > Amount of resources (CPU, RAM, Memory) … seklinde size almak istediginiz service hakkinda tahmini fiyat verir.*

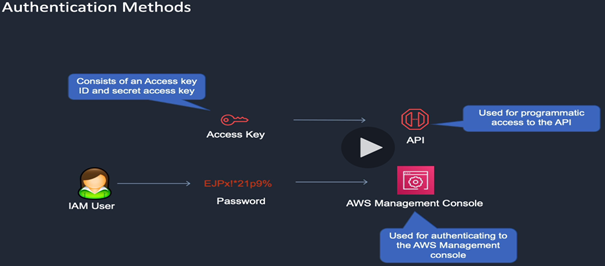
1. **AWS Free Tier account**:
   1. AWS Free Tier yazarak ulasilir. Burada bircok sinirsiz ve sinirli ucretsiz hizmetler var.
   2. **Billing alarm**: Hesap kurulumu sonrasi billing alarm kurulur. Billing alarm ile herhangi hizmette belirlediginiz siniri asarsaniz veya isiniz bitince herhangi bir hizmeti acik birakirsaniz size ikaz gonderiyor.
   3. *AWS Management Console > find’a «billing» yaz ve sec > Billing preferences > Receive Billing Alerts ve Receive Free Tier Usage Alerts (emai adresi girilir) > limitlere yaklasinca mesaj alirsiniz.*
   4. *Ustteki «aws» tiklayarak AWS Management Console ac > Services > Managemet&Governance > CloudWatch (performans takip tooludur, senin servicelerine izler ve kullanim miktarini gozlemler) > Sag ustten region seceriz (billing icin US East (N.Virginia) secilmesi zorunlu) > Sol taraftan «Alarms» > Create Alarm > Select metric > Billing > Total Estimated charge > USD > Select metric > Conditionstan sinirlari gireriz > next > create new topic > Create a new topic: Billing\_Notification > Email endpoint…:* [*byildiz2019@gmail.com*](mailto:byildiz2019@gmail.com) *> Create topic > View in SNS Console (sag tiklayip baska sayfada acilir) > Billing Notification > kendi emailinide acip onaylarsin. > Onceki sayfada next > Alarm name: Billing\_Alarm > next > ayar yapabilirsin burada > create alarm*

1. **AWS Identity and Access Management Service (IAM)**



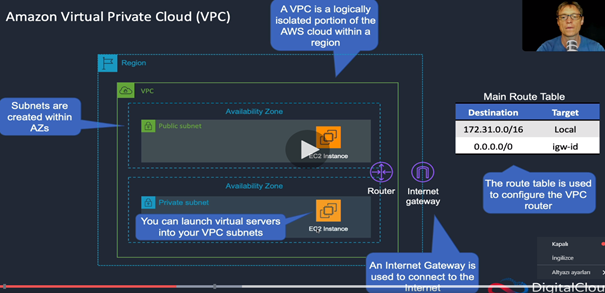
* 1. **IAM User**: Cloudu yonetecek/administrate calisanlara IAM User account acilir. IAM User bazinda hangi servisleri kullanacaginin yetkilendirmesi policy belirlenir.
  2. **IAM Group:** Kisi degilde ayni isi yapan grup bazinda yetkilendirme daha rahattir. Boylece ornegin developers grubu kurar ve permissions belirlersin ve hepsi ayni yetkiye sahip olur.
  3. **IAM Role**: Delegations icin kullanilir. For ex: you can applt a role to Amazin EC2 and that will give it permissions to do things with other services.

1. **Authentication Methods on AWS:**

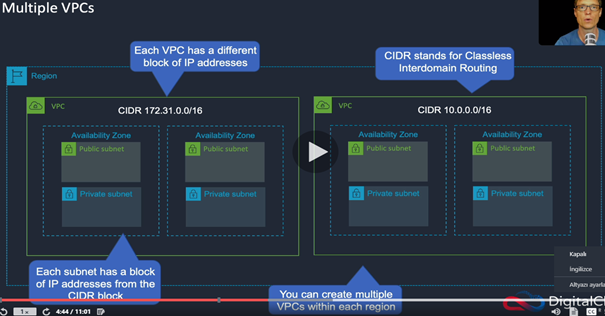


* 1. **Access Key:** We use access key for accessing AWS via the API.
  2. **IAM User:** AWS Management Console uzerinden root account password ile ulasilir. IAM User account olarak baglanilir.

1. **AWS Identity and Access Management Service (IAM)**:
   1. **IAM User ve Group olusturma:**
      1. *AWS Management Console > Services > Security, Identity, Compliance: IAM > Sign in linkini degistirmek icin «Customize» secilir. > isim girilir «ugur» > URL ismini rakam yerine isim olarak olusturduk. >*
         1. *Groups sol taraftan > Create New Group > Group name: Admins > Next step > Policyler secilir: AdministratorAccess (sectik) > Next Step > Create a group > Admins > Permissions >*
         2. Yukarda tanimlanan Group’a user eklemek icin*: «Users» soltaraftan > Add User > User name: umit > Access type: AWS Management Console access > Console password > Custom password: 1234 > Require password reset: deselect > next > set permissions boundary > Add user to group > Admins (sec) > next > review > create user > userin sign in icin kullanacagi lini kopyala > log out > kopyaladigin uzantiyi web servere yazip gir >*
2. **Amazon Virtual Private Cloud (VPC):** 
   1. **VPC**: VPC’ler Region icinde bulunur ve bir regionda en az 2 VPC bulunur. VPC’ler izole binalardadir VPC is a virtual space into which you launch your resources. VPC’ler Availibility Zonelardan olusur. Availibility Zonelar da 1 veya birkac data center’dan olusur. Verilerinizi yuklemek icin farkli data centerlara subnet olusturabilirsiniz. Kendi VPCslerinizi AWS’de olusturursunuz. Default VPC de bulunur.
   2. **Subnet**: You can then create subnets within your VPC and they're actually assigned to an availability zone. So you can create subnets that are sitting in different data centers and deploy your resources there.
   3. **VPC Router**: VPC icinde subnetler arasi veri iletisimi icin kullanilir. And the way that we can manipulate the router and configure it so that it sends data to the correct locations is by using what's called the **route table**. So we create route tables and assign them to our subnets. we have an address block and that address block is the master address block for which all our subnets will be created. And the target local means, the router knows that this is an internal address space and it will send the traffic between your subnets. Now, you might also need to send something to the outside world. So this is a special address. The 0.0.0.0/0 means everything else, which will mean other networks such as the Internet. So basically, if the destination address is within this range, the router will router internally, and if it's not, it will send it to the Internet gateway. So the Internet gateway is used therefore, to connect to the Internet. Now, you can launch your EC2 instances into your VPC subnets. So these instances here, which are virtual servers, can then run in subnets in different availability zones within the region. So they're in a private space and they're in different data centers and they have connectivity to each other. So high bandwidth, low latency connectivity because that's what's connecting the availability zones together.
   4. **The public subnet**: It means that the instances here will get a public IP address and can use an Internet gateway directly to access the Internet.
   5. **Private subnet**: they can't access the Internet gateway directly normally.

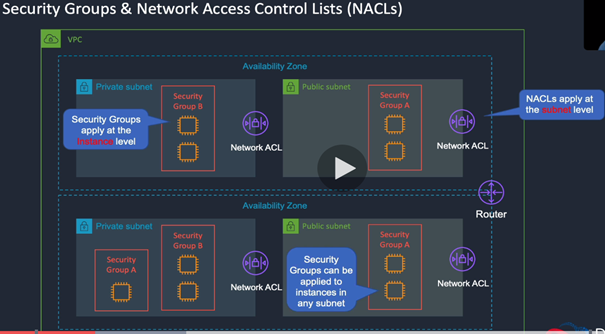


* 1. Farkli/ayni bolgelerde bircok VPCs’in olabilir.



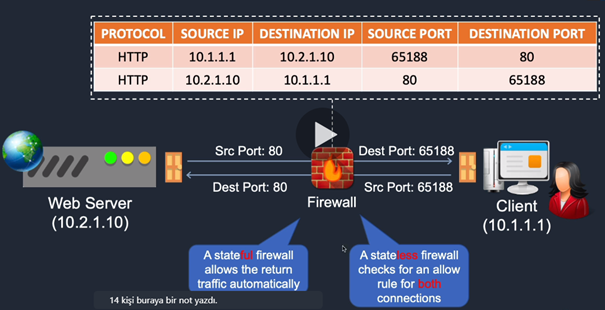
* 1. **IP Address**: Herbir VPC’ye tanimli IP addresi grubu tanimlidir ve VPC’lerin altindaki Subnetlere bu IP address blogundan IP address’i tanimlanir.
  2. *AWS management console > services > networking and content delivery > VPC* 
     1. we can see the resources (subnets, VPCs, …) by region.
     2. *Australia > Your VPCs (hazirda acik olan default VPC) > CIDR Blocks >*
  3. So auto-assign public IPv4 for address is set to, yes and that means that we're going to get an IPv4 public IP address. So an IP address that's available for use on the public Internet, which means we can connect directly to that IP address from the Internet and we can connect directly to the Internet from our instance. So a public subnet is one which has auto-assigned public IPv4 set to yes and it also needs to have an Internet gateway attached to it. And that's done through the route table.

1. **Security Groups and Network Access Control Lists (NACLs):**
   1. Her turlu firewallsu icerirler.

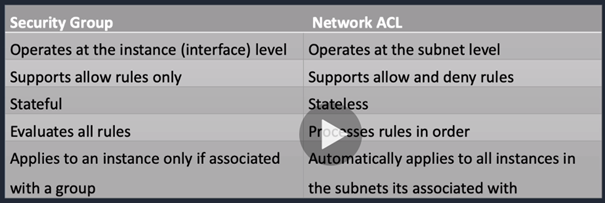


* 1. **Security Group**: Instance (yukarda chip olarak gosterilmekte) seviyesinde bir veya birkac instances’e uygulanir. Onemli nokta Security group olustururken icine koyacagimiz instances VPC bazinda kurulur ve farkli (public or private subnets) subnet ve availibility zone’lardan olabilir. Gelen giden trafigin kontrolunu yapar.
  2. **Network ACL**: Subnet seviyasinde uygulanir. Subnetten gelen-giden trafigi kisitlar. Ayni subnet icindeki iki instance arasindaki trafigi network ACL gormez.ama security group gorur. Ayni Security group icinde olsalar bile gorur.
  3. *AWS management console > services > networking and content delivery > VPC >* 
     1. *Network ACLs:* We can actually leave the default network ACL to control inbound and outbound traffic and then we're going to use security groups to apply further restrictions. Edit ile baska kurallar eklenebilir.
     2. *Security Groups:* So with a network ACL, you have a set of inbound and outbound rules, and network ACL can be applied to multiple subnets. For example, the first rule is 100. You could then add in another rule and at the end there is eight to nine, which means that if after processing all the rules and allow is not found, then the default deny will just basically drop the traffic. So if you want to create a rule, you can choose «*Edit inbound rules» > Add Rule > Rule: 200,Type: …*

1. **Stateful ve Stateless Firewalls**:
   1. **Stateful firewall**: Client makes a connection with web server. Serverden cliente cevap gelir. Stateful firewall will allow the return traffic automatically. So if the firewall has a rule that says i am going to allow this traffic inbound. Then it will send the outbound traffic without any further rules being required. Servere yalnizca 80 portundan (inbound rule) gelen mesajlari alir. Daha sonra 80 portundan gelen mesajlara outbound kisitlamasi uygulamadan her port numarasina cvp verilebilir. Web serverlar boyle calisir.
   2. **Stateless firewall**: It will check for and allow rule for both connections would not only need a rule that allows inbound traffic on port from this computer, it would also need a rule that allows outbound traffic on 65188 to this computer

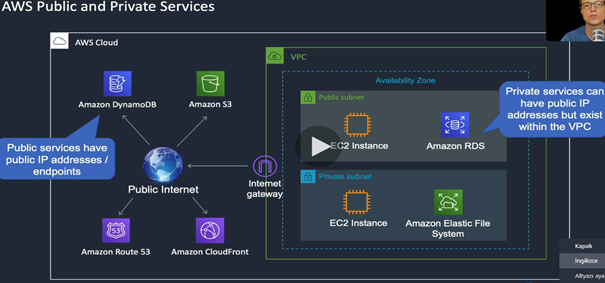


* 1. In the AWS world, the security group is a stateful firewall and a NACLs is a stateless.



* 1. Support allow rules only: by default, it will deny all traffic unless you create a rule to allow traffic. You don’t need to create a rule to say’ i’m going to deny traffic.

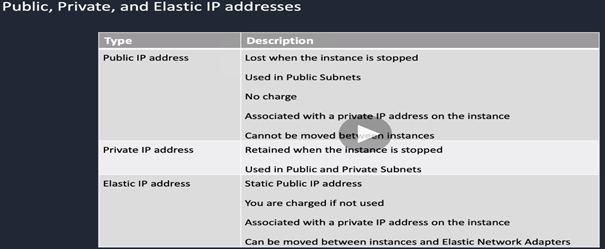
1. **AWS Public and Private Services:**
   1. **Private Service**: VPC icinde olusturulur. Private subnettekilerin private IP addresses olur ama public subnet icindelerse public IP addressleri olur. Belli seviyede guvenlik konusunda kontrolunuz olur.
   2. **Public sercices**: Sol taraftakiler ise public servislerdir ve public IP addressleri veya endpoints vardir. Internet uzerinden bu servislere baglanabilirsiniz.



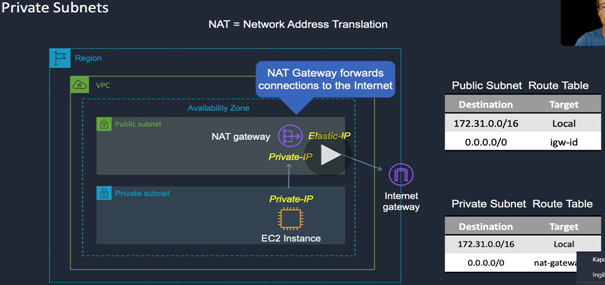
* 1. Eger private serviseteki EC2 Instance Amazon S3’e baglanmak isterse tipki public serverda oldugu gibi once internete baglanip oradan Amazon S3’e baglanir.

1. **AWC command line:**
   1. Aws cli install yazinca URL den bulunarak indirilir.
   2. Yukledikten sonra:
      1. aws –version kodu girilir ve versiyon gorulebilir.

1. Amazon Elastic Compute Cloud:
   1. EC2, instances’leri calistirir. Bir instance, AWS uzerinde calisan birtur Virtual Server veya Virtual Machine’dir. Ayrica instanceslerin launched edildigi Virtualaziation platformlari calistiran fiziksel serverler da vardir.



* 1. Tumu AWS tarafindan kontrol edilir IAAS. Herbir instance temelde bir virtual machine gibidir. Belli bir miktar CPU, RAM ,harddisk ve NIC bulundurur. Instance’lerin OS’leri ve applicationlari da olur. EC2’yi virtual server olarak dusun. VM’ye cok benzer. EC2 instanc’e IP address tanimlanir 3 farkli sekilde. Dogrudan internete baglanmak icin public IP addres cihaz durunca gider yeniden calisinca baska bir address alir, Kendi icinde konusmak icin surekli bir private IP address cihaz dursada devam eder. Ayrica bir de Elastic IP address vardir, public IP address’ten tek farki static olmasidir.
  2. Public subnet Elastic IP address ile internete baglanirken, private subnet’in yalnizca private IP address’i oldugu icin nete baglanmak icin NAT (network address translation) Gateway’a baglanir. NAT gateway public subnettedir.



1. **Launching an Amazon EC2 instance:**
   1. Amazon Machine Image (AMI) Icinde farkli versiyonlarda OS onceden yuklu olur. Hatta data bile onceden yuklu olarak gelir eger kendi AMI’ni olusturursan.
   2. Instance type secilir. t2.micro general purpose maksatlidir ve ucretsizdir.
   3. *AWS Management Console > Services > Compute > EC2 > Instances > Launch instances > (bircok OS alternatifi cikar) > My AMIs > AWS Marketplace > hazir AMIs var ornegin firewall yuklu > Quick Start > Amazon Linux 2 AMI (Amazonun kendi Linux’u) > Select > General purpose (free tier eligible) > Next: Configure Instance Details > Next: Add Storage > Next: Add Tags (Key: Dept, Value: Development) > Next: Configure Security Group (Security Group name: Web-Access) > Review and Launch > Launch > Create a new key pair, Key pair name: Oslo-KP, Download Key Pair, Allow > Launch instances > View launch log > View Instances >*

1. **Connecting to Amazon EC2 instances:**
   1. Region secilir (Windows Server Set up olur kendisi) > Launch Instance > Microsoft Windows Server 2019 Base - Select > t2 micro – next: configure Instance Details > Add Storage > Next: Configure Security Group > Security Group name: Web-Access, Description: Web-Access > Review and Launch > Launch > Create a new key pair, key pair name: Singapore-KP > Download Key Pair > Launch Instances >
   2. Sydney Regionu secilir > Instances > Connect > instance’e nasil baglanacaginizin uzantisini verir (Linux OS var). > 3. satirdaki komut cmd promt’a yazilir.
      1. >chmod 400 Sydney-KP.pem : yapistirilir.
      2. >Sonra 4. satirdaki ama alttaki satiri yapistiririz.
      3. Boylece SSH ile internet uzerinden instance ile baglanti kuruldu.
      4. >sudo yum update -y : tum dosya ve guncellemeleri yukler.
   3. Simdi de Region degistirip Singopore’a baglaniyoruz. > Instances > Connect > Get password > Choose file > Documents > AWS > Singapore-KP.pem > Decrypt Password > Public DNS Name kopyalanir >
      1. Kendi bilgisayarimizda RDP client (Microsoft Remote Desktop) acilir > PC name’e Public DNS name yapistirilir > Add > CT > Username: administrator, Password: Connect tusuna basinca Password olarak gosterilen sifreyi kopyalayip buraya yapistiririz. > continue > continue > Amazon EC2 uzerinden Windows Server’a baglandik > Ben Sydney,deyim ve instance Singapore’da.
   4. Windows’tan ulasmak istersek:
      1. PuTTY download edilir ve yuklenir > Menuden PuTTYgen acilir, PuTTY’de .ppk file calisir bizde ise .pem file var. Bunun donusturulmesi lazim
      2. Documents > AWS > Sydney-KP.pem RC > Open with > Other > TextEdit > Open > Icerigi kopyalariz > Notepad’i acariz > key pair icerigini buraya yapistiririz > save (Sydney-KP) desktop’a>
      3. PuTTYgen > load > Desktop > Sydney-KP > open > save private key (Sydney-KP.ppk) >
      4. PuTTY > AWS Management Console’dan Sydney Regionu secilir > Public DNS (IPV4) kopyalanir > PuTTY’deki Hostname’e basina ec2-user@ yazilarak yapistirilir > Soldan SSH tiklanir > Auth > Browse > Sydney-KP > Open > open > Sydney’deki Amazon Linux 2 AMI’ye Singapore’dan baglandik.

1. **Create a Website using a User Data:**

1. D

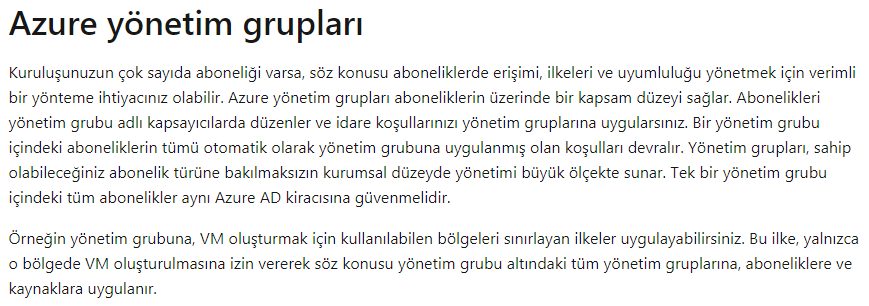
**C. MICROSOFT AZURE**

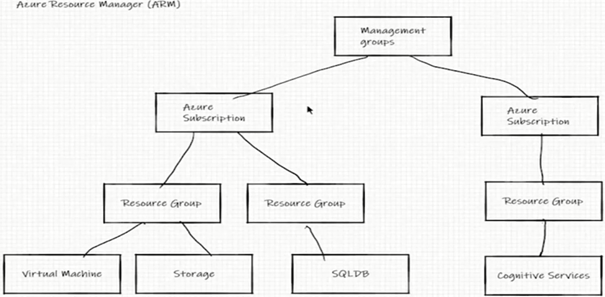
1. Kaynak:
   1. DONPER Danismanlik – Mehmet Kut’a her turlu soru linkedin ve twitterden sorulabilir.
   2. microsoft.com/learn
      1. Sign in ile uye olun > Browse all paths > “Azure Fundamentals” 9.48 saat bu eğitimin calisma pathi >
2. portal.azure.com adresi ile baglanilir.
3. **Azure Mimarisi (architectural components):**
   1. **Regions**: Halihazirsa 54 regions olup 140 bolgeye yayilmis durumdadir. Regionlar icinde de data centerlar bulunmaktadir. Region secimi nasil olmali:
      1. **Kullanicilarim nerede**: Hizmeti Norvecten yonetiyorsam bile kullanicilarim Ingiltere’de ise servisi Ingiltereden acmaliyim.
      2. **Region’da istedigim servis bulunuyor mu?** Her region ayni kabiliyete sahip degil.
      3. **Fiyat:** Ulke bazinda vergi avantajlari dolayisiyla region bazinda fiyatlar degiskendir.

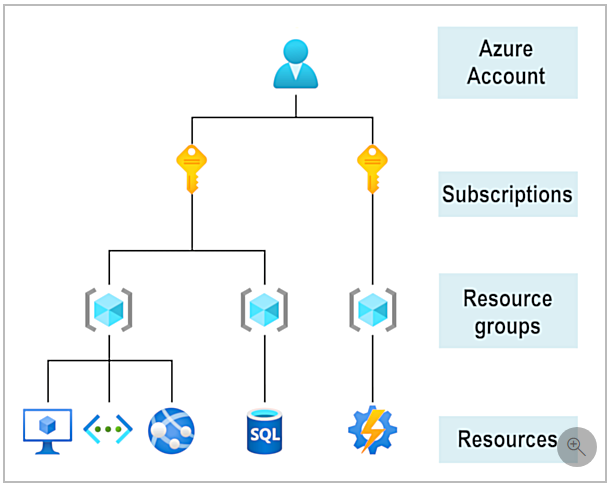


1. Services:
   1. AZ-900T0x Module 02: Core Azure services:

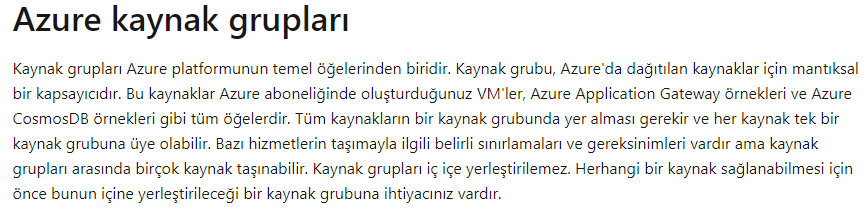
1. **Availibility sets:** 
   1. Applicationlarin maintenance ve failure durumlarinda Microsoft surekliligini saglar. Azure region icinde 3 adet data center olabiliyor. Farkli binalarda. Availibility zonelarin sogutma, yalitim, network,… ayridir.
   2. **Update domains (UD):** Scheduled maintenance, performance or security updates are sequenced through update domains. Ayni anda farkli serverlarin guncelleme almaz ve muhakkak bir kismi calisiyor durumda olur.
   3. **Fault domains (FD):** Provide a physical separation of workloads across different hardware in a datacenter.
2. **Resource groups (Kaynak Gruplari):**
   1. Once hesap (account) acilir buradan management groups ulasilir. Hesabin altinda istenildigi kadar abonelik (azure subscription) baslatilabilir. Herbir azure subscriptionun altina resource group olusturulabilir. Herbir resource grupun altina serviceler (VM, Storage, …) tanimlanir. Management group = Universite, subscription = muhendislik fakultesi seklinde degerlendirilebilir. Bu kaynak yonetimi yapisina Azure Resource Manager (ARM) denir. Faturalandirmalar subscription bazinda olur.
   2. Birden fazla subscription acilmasi, ornegin bir sirkette her bolume farkli bir abonelik acarak kendi kaynaklarini yonetmesine imkan vermek icin yapilir. Ilgili bolum kaynaklarin yoneticisi olmak isteyecektir.
   3. Herhangi Azure Subscription’da yetkiliyseniz baglisi resource grouplarda da yetkilisiniz ama resource group da yetkili iseniz sadece o resource groupu gorursunuz. Hatta kaynak bazinda yetkilendirilendirilirseniz ornegin yalnizca storage’i gorursunuz.
   4. Yetkiler uc cesittir: owner, contributer ve reader.



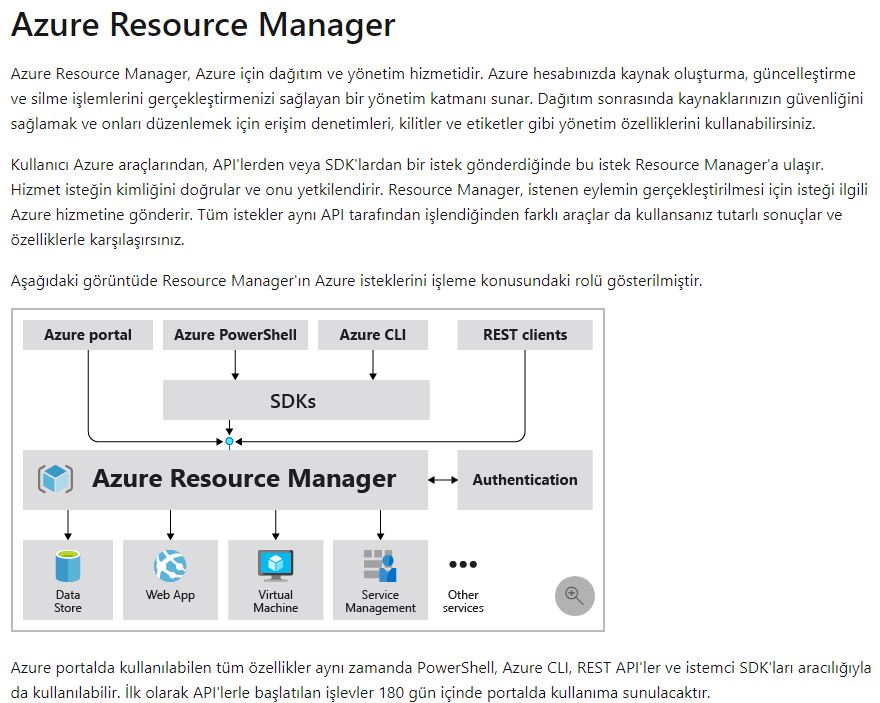




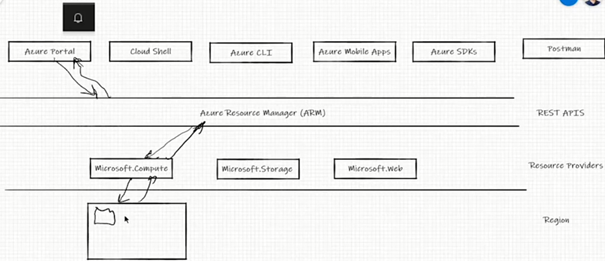
* 1. Containers for multiple resources that share the same life cycle. Ilk once region secilir. O kaynagi hangi kaynak gruba koymak istediginizi sececeksiniz.
  2. Aggregates resources into a single manageable unit.
  3. Every Azure resource must exist in one (and only one) resource group.
  4. Secure at the resource group (or resource) level – using role – based access control (RBAC).



1. **Azure Yonetimi:**
   1. Azure Portal, Cloud Shell, Azure CLI, Azure Mobile Apps, Azure SDKs ve Terraform vb ile Azure yonetilebilir. Bunlarin tamami client uygulamalaridir.
   2. **Azure Resource Manager (ARM):** Client uygulamalarindan ornegin azure portalden yapilan tum istekler ARM’a gider. Bu istekler ARM Template’e gider. ARM Template Jason formatindadir.

****

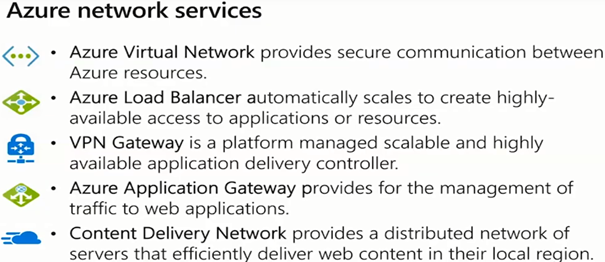
* 1. **ARM Template**: Ekranda client olarak girdigimiz veriler sonrasi create’a basinca portal vb client uygulamalari bizim icin ARM Template olusturur. Daha sonra bunlari ARM’a post ederler. Ornegin:
  2. ARM icinde bir sanal makine olusturma talebi olan bir Jason dosyasi var.
  3. **Resource Provider**: ARM altinda Resource Provider’lar bulunur. Sanal makineyi olusturmadan sorumlu resource provider: Microsoft.Compute’dur.
     1. Microsoft.Compute: Sanal makine olusturmayi saglayan Resource provider.
     2. Microsoft.Storage: Storage olusturmayi bilen resource provider
     3. Microsoft.Web: Web app olusturmayi bilen resouce’un adi… vb 100’lerce var.
     4. ARM daha sonra bu Jason formatindaki talebi Microsoft.Compute resource provider’a gonderir.
     5. Data Center: Daha sonra region secerek ilgili service’i hangi data center uzerinden saglayacagimizi belirleriz. Microsoft.Compute, West Europa Data Center’a diyorki bana bir D serisi makine ayarla. Data center’da icinde D serisi makinayi ayarliyor ve olusturuldu bilgisini Microsoft.Compute’a iletiyor.Resource provider da ARM’a islem tamam bilgisini iletir. ARM’da ilgili bilgiyi client’a iletir. Daha sonra Azure portal’da can isareti yanip sonerek bilgiyi verir.



* + 1. Bazen bazi resource providerlar her subscription icin register olmamis olabilir. Register etmek gerekiyor.
       1. *Azure portal > Subscription > Aktif Subscription > Resource providers > ilgili provider secilir > Register.*
    2. **Templates**: github.com/Azure/azure-quickstart-templates uzerinden bircok ARM template hazir var. Gorsel olarak da size gosterir ve deploy edebilirsiniz. Azure portal acilir ve sizden adet, yer vb bilgilerini girerek ilgili template’i calistirabilirsiniz.
    3. **Services:**
       1. Azure’daki tum serviceler Azure’in degildir. Azure portal’da service ismi aratinca sol tarafta Azure altinda gorulenler Azure’un, sag tarafta Marketplace’in altindakiler ise ARM Template sayesinde Goggleplay store gibi Azure da ARM Template uzerinden urununu sunanlarin urununu hizmete sunuyor. Ornegin Wordpress gibi.
       2. Azure Marketplace’e googledan ulasarak ilgili uygulamalarin ARM Template’lerini get it ile kendi azure’umuza kalkindirabiliriz.

1. **IoT Hub:** 
   1. Nesnelerin interneti icin kullanilir.
   2. Github.com/mehmetkut/az-900/blob/master/Walkthroughs/06-Implement%20the%20Azure%20IoT%20Hub.md ‘den Raspery pi online simulatorune tiklayip girilir. Burada nem ve sicaklik olcen bir cihaz var ve her mesaj gonderildiginde yanip sonen bir led var. Raspery pi’in uzerindeki Io portlari ile kontrol ettiriyoruz.

1. **Azure network services:**

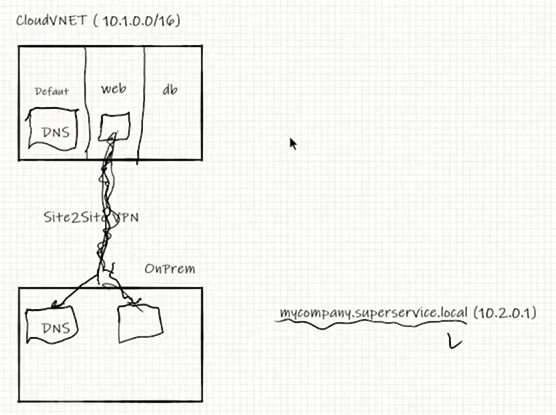


* 1. Azure Virtual network: Sanal makine olusturuyorsan mutlaka gerekli.
  2. Azure Load Balanser: Is yuklerini dagitmaya yarar.
  3. VPN Gateway: Hibrit mimaride primary ve azure data center’i birlikte kullanmaya yarar. Iki cihazi birbirine bagliyorsunuz.
  4. Azure Application Gateway: Uygulama gateway’i. Uygulamalari bu gateway’in arkasina alip disari acmadan public IP olmadan disari yayinlayabiliyorsunuz.
  5. Content Delivery Network: Buyuk dosyalarinizi dagitmak icin kullanilir,
  6. **Sanal Network olusturmak:**
     1. Create a resource > virtual network (arattik) > create > resource group: create new: kadirhasvnet-rg > Name: CloudVNet > next > next > next > create
  7. Sanal Networku calistirma:
     1. Home > CloudVNet
        1. *> Address space > 10.1.0.0/16 (buradaki adresin privat datacenter (server)’in adresi ile ayni olmamasi gerek)*
        2. > Connected devices > servisleri görebiliriz
        3. > Subnets > farkli subnetleri gorup network security group yapilandirarak izolasyon yapılır
        4. *> DDoS protection > Basic ucretsizdir, Standart ise ayni seviyede korusa da bilgilendirme vb ozellikleri nedeniyle ucretlidir,*
        5. > **Firewall** > Virtual networkun onunde kullanilabilir ayrica tek basina kendi sirket sistemi icin firewall olarak kullanabilirsiniz.
        6. > **DNS servers** >
        7. > **Peering** > Farkli subscriptionlar arasindaki Iki ayri VNet’i peer/eslestirmek edebiliyorsunuz. (ornegin iki ayri sirket). Nornalde ayni subscription’da olsaydilar subnet arasinda security group’lar ile iletisimi saglardik.
           1. *> Add > diger peeringin adini gir, subscription gir > … boylece uzaktaki subcription’lu VNet ile de erisebilimeye baslaniyor.*
        8. > **Service endpoints ve Private endpoints** > elimizdeki subscriptionda kullandigimiz kaynak/servisleri (PaaS’leri) diger subscription VNet’inde de kullanabiliriz.
           1. *> Add > Service > ilgili servisler yeni subscriptiona da atanir.*
           2. *Diger VNet’e ulasma: Azure Services > khas-rg-vnet-bastion (diger VNet’i sectik) > Virtual network/subnet: khas-rg-vnet/AzureBastionSubnet*

*> (Connected devices’ta diger subscription’un IP’sini gorebiliriz.)*

*> Diagram > tum topolojiyi goruruz.*

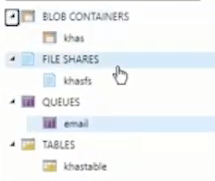
* 1. Virtual Network (VNET):



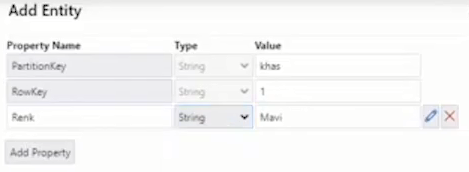
* + 1. icinde ise subnetlerimiz var.
    2. Subnetler arasinda ise network security group’lar ile iletisim var.
    3. Asagida ise OnPrem kendi private veri merkezimiz var.
    4. VPN Gateway ile VNET ve OnPrim arasinda baglanti kurduk. Site2Site VPN. Boylece Cloud VNET ile OnPrem haberlesmeye basladi.
       1. VNET IP addressi ile OnPrem icindeki IP addressleri cakismamali.
       2. VNET icinde cesitli subnetler var (Default, Web, Database).
       3. Web subnette bir web serwer sunucumuz var ve OnPremdeki bir servise/kaynaga Site2Site tunelden erisiyor.
       4. Ilgili kaynak mycompany.superservice.local (10.2.0.1) ise ilgili servisin IP’sini OnPrem’deki DNS buluyor. Daha sonra Site2Site tunelden geri gidip tekrar ogrendigi IP ile tunelden OnPrem’e girip kaynaga ulasir.
       5. Azure portalin altinda acilan CloudVNet’in altindaki DS servers servisi veya ozelligi sayesinde
          1. CloudVNet > DNS servers > custom > 10.5.0.1 (seklinde rastgele yapti tanimlarsiniz)
          2. Onerilen ise OnPrem’de DNS’iniz varsa onun readonly bir kopyasini VNet’e replike edin. Her iki taraftaki DNS, custom DNS olarak verilse de tunelden gidis donusu azaltmis oluruz.
       6. Peering

1. **Yetkilendirme:**
   1. **Subscription bazinda yetkilendirme yapmak icin:**
      1. *Subscriptions > Subscription name altindan biri secilir > Access control (IAM) > Role assignments (yetkili kimseler cikar) > Add > Role: owner, Assign access to: Azure AD user group or service principal (servisler), yetkilendirilecek kisinin (Pamir) mail adresi > save > Pamir’e davet linki gitti, Pamir kabul edince kendisi de owner olarak tanimlanmis olacak*
      2. Yetkilendirmeye subscription bazinda yaptigimiz icin ilgili subscriptionun altindaki tum kaynak gruplarinda yetkili olur.
         1. *Azure services > Resource Groups (kaynak gruplari) > khas-vg (bir kaynak sectik) > ilgili kaynak altindaki tum servisler acildi (bunlarin da owneri)*
   2. **Kaynak Grubu bazinda yetkilendirme (Azure RBAC):**
      1. Azure Services > Resource Groups > khas-vg (bir kaynak) > Access control (IAM) > Role assignments > Add > Roll assignments > Role: contributer, you are a guest...: [umitceylan9311@gmail.com](mailto:umitceylan9311@gmail.com) > save > yetki verildi.
   3. **Servis bazinda yetkilendirme yapmak:**
      1. Azure Services > Resource Groups > khas-vg (bir kaynak) > vmkhas (bir servis) > Access control (IAM) > Role assignments > Add > Roll assignments > Role: contributer, you are a guest...: [umitceylan9311@gmail.com](mailto:umitceylan9311@gmail.com) > save > yetki verildi.
   4. **Yetki cesitleri:**
      1. Owner: tam yetki, kaynagi silip, calistirip, ayarlari degistirebilir. Baskalarina da yetki verebilirsiniz.
      2. Contributer: Owner rolu gibi yetkileri vardir. Baskasina yetki veremezsiniz.
      3. Reader: sadece ozelliklerini gorebilirsiniz.

1. **Kisi bazinda islem takibi (Activity log):**
   1. Azure services > Recent sources: ymkhas (sanal makine) > Activity log > Timespan: last 24 hours sectik > Yapilan operasyonlari ve kimin yaptigini gosterir.
2. **Tags (Etiketleme):** 
   1. Azure services > Resources: khas-rg >Tags > Name: Owner, Value: Mehmet Kut; Name: Env, Value: Production; Name: Admin, Value: Pamir Erdem > Save > 3 adet tag olusturduk
   2. Azure Services > arama cubugu “Tags” > taglar cikar ve Owner: Mehmet Kut olan kaynaklari gosterir. > vmkhas (bir kaynak sectik) >
3. **Storage (Storage account):**
   1. **Storage olusturma:** *Azure sevices > create a resource > Storage account – blob file table > Resource group: khas-rg, Storage account name: khasmk11, Location: West Europe, Replication: (kritik olmayan log datasi sakliyorsaniz: LRS, Onemli veriler icin: GRS, ) > next > Connectivity method: Public endpoint (all network-internetten ulasilabilir) > next > Security transfer ..: Enabled, Large file shares: Disabled > Data protection: Enabled (yanlislikla storage’tan dosyalari geri getirebiliyoruz belirlenen sure icinde, ve ancak admin gorebilir o dosyayi), Data Lake Storage Gen2: Disabled > next: > Taglamadik > next > create > go to resource > storage hesabina girilir > khasmk11*
   2. Account kind: StorageV2... secildigi icin 4 adet farkli ozellikli harddisk verdi, ama Blob secseydik sadece container verecekti.
   3. Sonradan Replication ayari degistirilebilir.

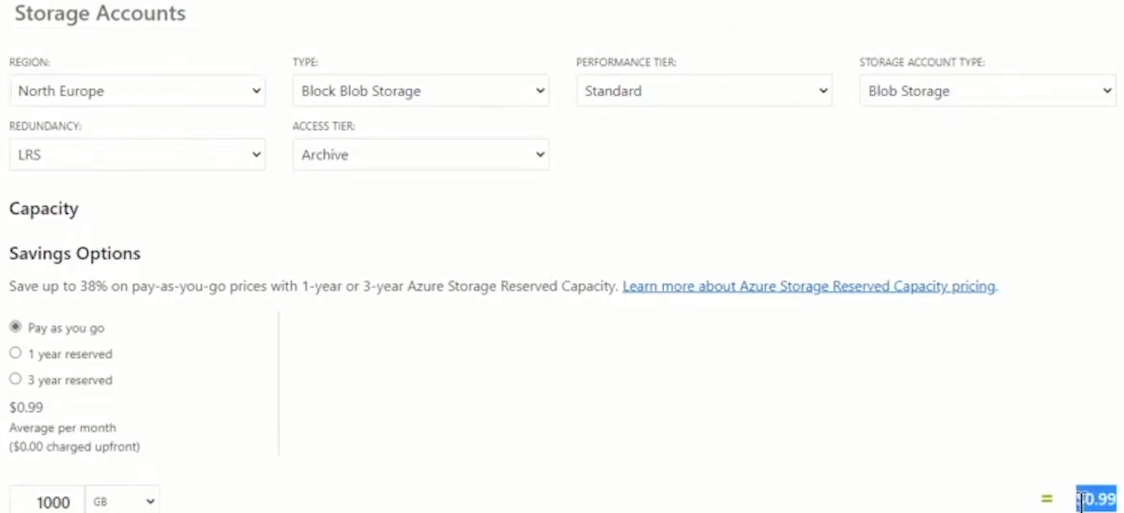


* 1. **CONTAINER’E dosya kopyalama (portalden):**
     1. **Container olusturma**: Azure services > Resources: khasmk11 > Container > + Container (conteiner olusturulur) > Name: khas, Public access level: Private (yalniz solda Shared access signature’dakiler), Blob (adresi bilen herkes erisebilir) secildi, Container (adresi bilen herkes erisebilir ve o containerdeki diger dosyalari de kesfedebilir), > create >
     2. **Container’e dosya yukleme:** … > khas (container) > icinde dosya yok > upload > Files (sagdaki dosya isareti) > bilgisayardan bir dosya secip yükleriz
     3. **Container’deki dosyaya erisim**: Yukledigimiz dosyanin uzerine tiklayinca URL: … gorulur. URL’yi kopyalayip arama cubuguna yapistirinca dosyaya ulasilir.
     4. Container’in Access level degistirme: … > khas > Change access level > Private > Ok
     5. Private Access Container’e erisim: … > khas > yuklu dosya > Generate SAS > Start:, Expiry: ile girilebilecek sureyi, Allowed IP adresses: ile yalnizca belirli kisilere izin verilebilir. > Generate SAS token and URL > Blob SAS URL kopyalanir > Nette bu URL’yi yapistirinca belirli sure icin dosyaya ulasilabilir.
     6. Ic ice container acilamaz. Ic ice folder acilabilir.
  2. **Containere dosya kopyalama (Azure Storage Explorer):**
     1. Nette: “azure storage explorer” yaz ve gir > Azure Storage Explorer > Download now > kur > (kullanici adi, sifre gir veya … Storage account > khasmk > Access keys > key1 > Connection string) > Connection string’i portal uzerinden gidip kopyalar ve ve buraya yapistirirsaniz artik ulasilabilir.
     2. Azure Storage Explorer bir uygulamadir. Dosyalarin suratle Azure Storage’e gonderilmesi ve download edilmesi icin kisa bir yoldur.
  3. **Containere dosya kopyalama (Storage Explorer (preview)):**
     1. Azure portal … > Storage account > khasmk > Storage Explorer (preview > Blob Containers > khas >
  4. **FILE SHARES:** 
     1. SMB protokolunde file share imkani tanir.
     2. **File share olusturma:** Azure portal Storage account > khasmk > File share > + File share > Name: khasfs, Quota: 50GB > create > file share oluştu
     3. **Olusturulan File share bilgisayara kurma**: Azure portal Storage account > khasmk > File share > khasfs > connect > Windows > Drive letter: K (sectik) > alt pencerede uretilen powershell script’ini kopyala
        1. Desktop > arama “powershell” > Windows PowerShell ISE > PowerShell komutunu yapistir > ileri tusuna bas > This PC’nin altinda Network locations: khasfs olusur. (ayni fileshare’e tum ekip arkadaslari attach olup buradan dosya paylasimi yapilabilir. )
        2. Istedigimiz dosyayi buraya surukleyerek paylasabiliriz.
     4. **Kapasiteyi arttirma:** Azure portal Storage account > khasmk > File share > khasfs > Edit Quota > 70Gb > ok
  5. **Tables:** 
     1. Azure portal Storage account > khasmk > Storage Explorer (preview) > Tables (oku) > create table > Table name: khastable > khastable > +Add > PartitionKey: khas, RowKey: 1 > Add Proerty > Renk: Mavi > Insert



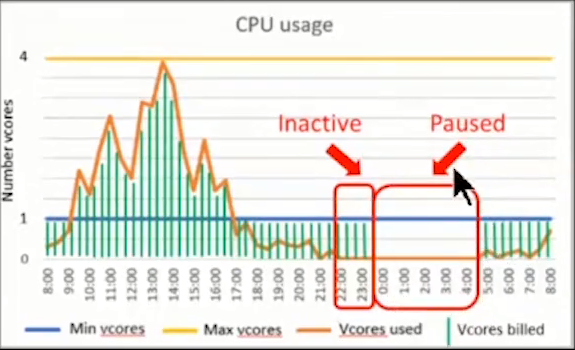
* 1. **QEUES:**
     1. Azure portal Storage account > khasmk > Storage Explorer (preview) > Queues > name: email > ok > email > +Add Message (mesaj eklemek icin) > Message text: test… > ok >
  2. Data transfer:
  3. Azure portal > … Storage > khasmk > Data transfer > Estimated data size…: 100Gb, App network bandwith: 100Mbps (benim bant genisligim), Transfer freq: Once (bir kerede) > alternatif online ve offline transfer usullerini gösterir.

1. **Ucretlendirme:**
   1. **Burasi önemli**
   2. Nette “azure pricing calculator” > azure.microsoft…calculator > kendi sayfamiza girdik > Products
      1. **Storage Accounts** (eklemistik) > Daha once seçtiklerinizi ve aylik ucreti gösterir. > Export (tahmini ucreti cikti verir)
      2. **Virtual Machines** > Daha once seçtiklerinizi ve aylik ucreti gösterir.
         1. OS (Windows): Azure Hybrid Benefit (ilgili kurumlarin kendi lisans anlasmalari varsa secilir), > Export (tahmini ucreti cikti verir)
   3. Burada alternatif mimarileri deneyerek farkli fiyatlandirmalari görebilirsiniz.
   4. Region en ucuz: North Europe, Access Tier: Archive (cok sik erismeyeceksek), Performance tier: Standard, Redundancy: LRS



* 1. Nette “azure pricing calculator” > azure.microsoft…calculator > kendi sayfamiza girdik > Example Scenarios > Bir senaryo sec > Add to estimate
     1. Ihtiyacimiz olan mimarinin ornegini bulup fiyatini görebiliriz.

1. **Mimari Planlama Nasil Yapilmali:**
   1. **Buradan ihtiyaca uygun ornek servislerin kurgusunu bulma:**
      1. Nette “azure architecture center” > docs… > Example Workloads (secilebilir) >
   2. **Lokal serverda kurunca cikacak maliyeti hesaplama:**
      1. Nette “azure total cost of ownership” > azure.mic… > Total Cost of Ownership Calculator > +Add server workload > Servers: SQLDB (sunucu), CPU, OS, RAM … vb gireriz. Diğerlerini de gireriz > next > elektrik kw/ saat birim fiyatlarini da yazarak, IT calisan giderleri, yazilim/donanim giderleri, datacenter giderleri girince siz ebir rapor cikarir ve azure ile kiyaslamasini verir.
2. **Azure SQLDB:** 
   1. **SQLDB oluşturma:**
      1. *Azure portal > Create a resource > SQL Database > Resource group: khas-rg, Database name: khasdb, Server::Create new (halihazirda yok) > Server name: khssvr, Server admin login: sysadmin, Password: … , Location: West Europe > ok > Want to use SQL elastic pool: yes, Configure database > Basic/Standard… seçilir > Save Money: Already have a SQL Server license?: Yes (sirkette zaten lisansi varsa seçilir.), Serverless (ücretlendirme Provisioned’teki gibi saatlik değil saniye bazli olur. Ornegin gece database calismadigi için ucret odemezsiniz. Belli bir sure kullanilmayinca otomatik olarak ucretlendirmeye kapanir yeniden kullanilmaya baslanincaya dek), Hyperskale (gecikme zamanini ornegin yaz donemi, secim donemi vb kritik günler için hizini kisaltabilirsiniz.) > Apply > next > Connectivity method: public (disardan ulasilabilir) > Allow Azure servics.. to Access this server: yes, Add current client IP address: Yes > next > use existing data: sample, Enable advanced data security: > next > next > create > Go to resource > veri tabani acildi*
      2. **SQLDB veri tabanina baglanma:**
      3. Sonradan dagisiklikler, kapasite arttirimlari yapılabilir. PaaS olarak calisir.



* 1. **SQLDB’ye bağlanma**: Farkli usulleri vardir.
     1. Azure Data Studio: *Nette “Azure data studio” > docs.microsoft… > Azure data studio > Install > Windows >*
     2. Azure portal:
        1. SQLDB baglanti kurma: *Azure portal > … > khasdb(khassvr/khasdb) > Query editör (preview) > Login: sysadmin, Password: … > firewall hatasi cikti VPN acik (IP addresi orda yaziyor) olduğu için onu da tanıtmak gerek > Set server firewall > Rule name: VPN, Start IP: VPN’in IP’si, End IP: VPN’in IP’si > Save > Ok > kurduk*
        2. SQLDB’ye baglanma*: Azure portal > > … > khasdb(khassvr/khasdb) > Query editör (preview) > Login: sysadmin, Password: … > Tables*
  2. **Configure**:
     1. Konfigurasyon/kapasite değişiklikleri yapılabilir.
  3. **Geo-Replication:** 
     1. Farkli lokasyonlar seçip > Save ile datami replike edebilirim.
  4. Connection strings:
     1. Developerlar için connection string alabilirsiniz.
  5. **Advance data security:**
     1. Kisisel bilgileri koruma:
        1. Azure portal > … khasdb > Advance data security > Data Discovery &Classification > Classification > Table’deki basliklarin gizli olmasi konusunda uyarir ve ilgili tum confidentialleri seçip save edersiniz ve korur.
     2. **Acik tarama:**
        1. Azure portal > … khasdb > Advance data security > Vulnerability Assesment > Scan (aciklari tarar) > problemleri gosterir
     3. Advanced Threat Protection: