Microsoft Azure (Continued)

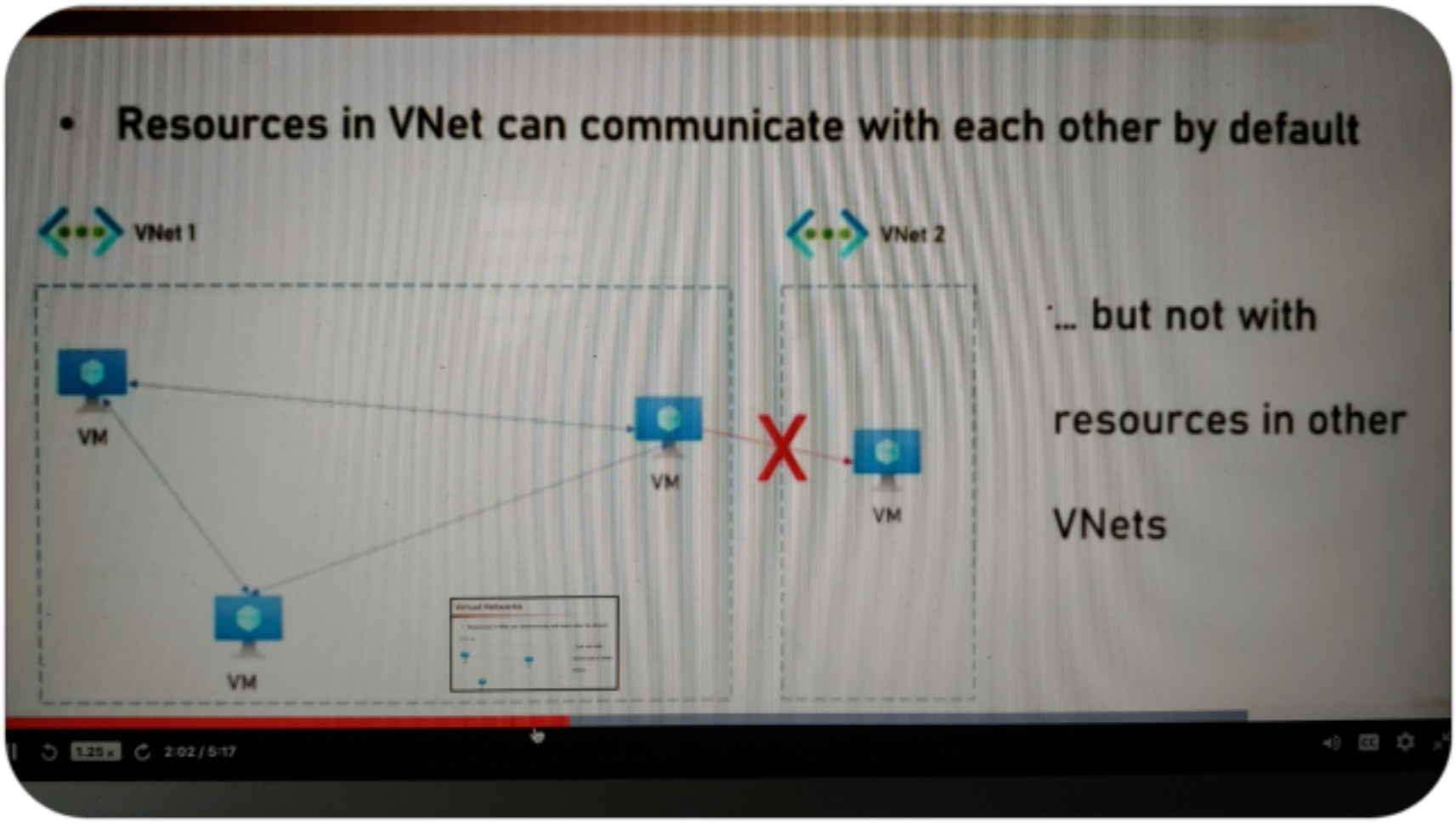
Azure Networking

- **Networking in Azure:** Networking is the foundation for designing cloud security. Never leave a VM open so that it can be accessible from the internet or can be RDPed from anywhere. There are 2 main threats if we open a VM - Brute force attack on port 3389(RDP) and No line of defense in front of the VM web server. Networking knowledge is what makes an amazing cloud architect.
- There are mainly 4 networking related cloud services: VNets, SubNets, Load Balancer, Application Gateway.

Virtual Networks (VNets)

- Its a network where we can deploy many of cloud resources like VMs, App Services, DBs etc
- "Virtual" as in "based on physical network and logically separated from other virtual networks". So when we create a new virtual network, we don't really create a physical network with cables, switches and routers. Instead, we use an existing networking resources that are part of Azure and inside the azure network we create a logically separated network that we are going to use.
- Resources in a single VNet can communicate with each other by default. For ex if we have a single VNet where we defined 3 VMs all of them can communicate between each other by default. But they can not communicate

with resources(for ex VMs) in other VNets.

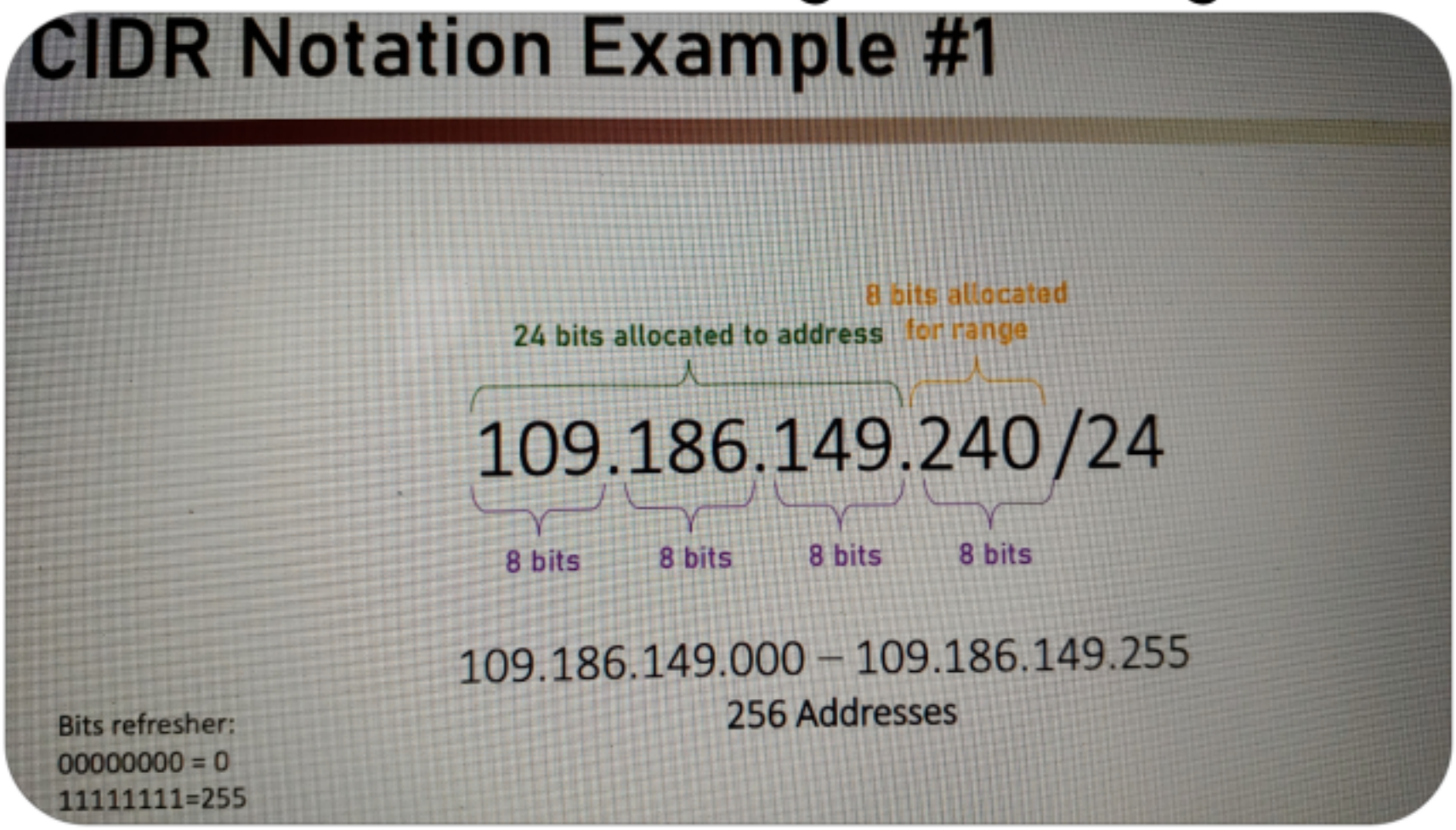


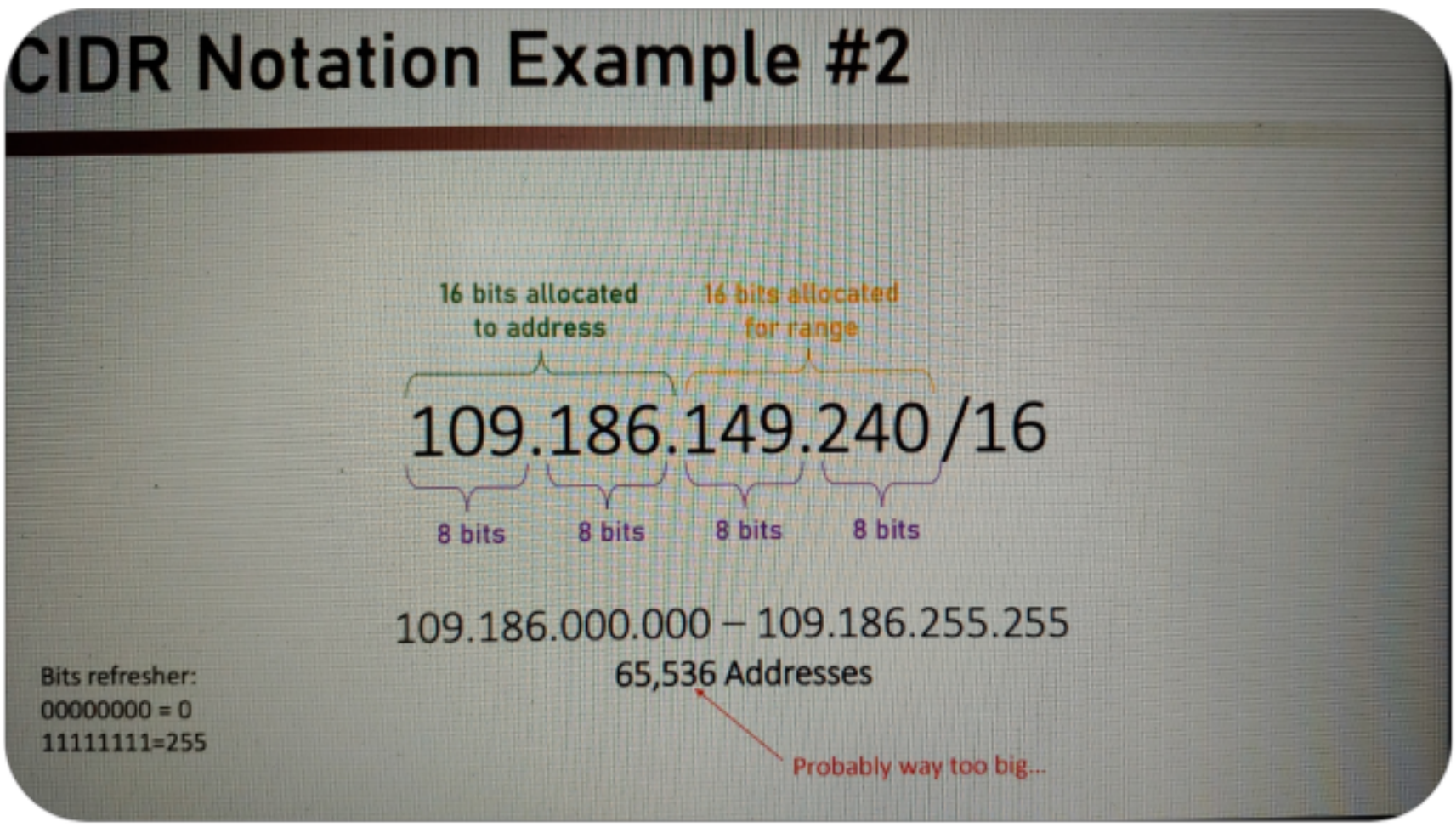
- Think VNets as organization's private network. In AWS its called VPC - Virtual Private Cloud. That means other organization's VNets can not communicate with your organization's VNets.
- VNet Pricing: VNets are free but its limited to 50 per subscription across all regions.
- **Characteristics of VNets**: Its scoped to a single region, cannot span multiple regions. Scoped to a single Subscription. Can be

connected via Peering. VNets are segmented using Subnets and protected using NSG.

- **Security and VNet**: The most important thing to think about when designing network is how to limit access to the resources in the VNet so that risk is minimized
- **Address of VNet**: Each VNet has its own address range or IP range. By default this range spans 65536 addresses. This number can be customized. All network devices must be in this address range. This addresses are expressed using CIDR notation.
- **CIDR Notation**(Classless Inter-Domain Routing): Its a method for representing an IP range or address range. It composed of an address in the range and a number between 0

and 32. The number indicates the number of bits that are allocated to the address. The smaller the number - the larger the range.

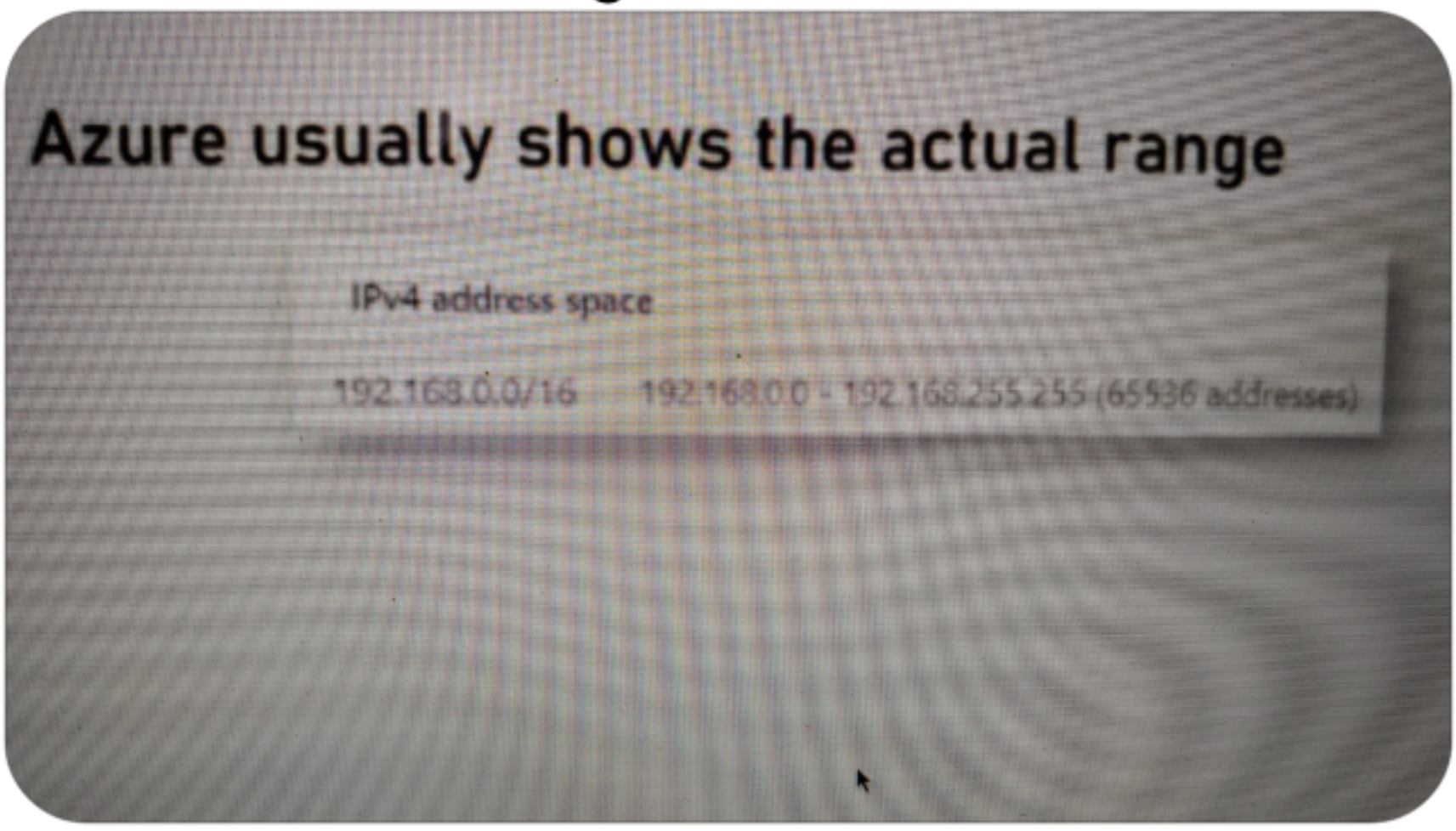




≡ ○ So as you can see the lower the

number - larger the adsress range

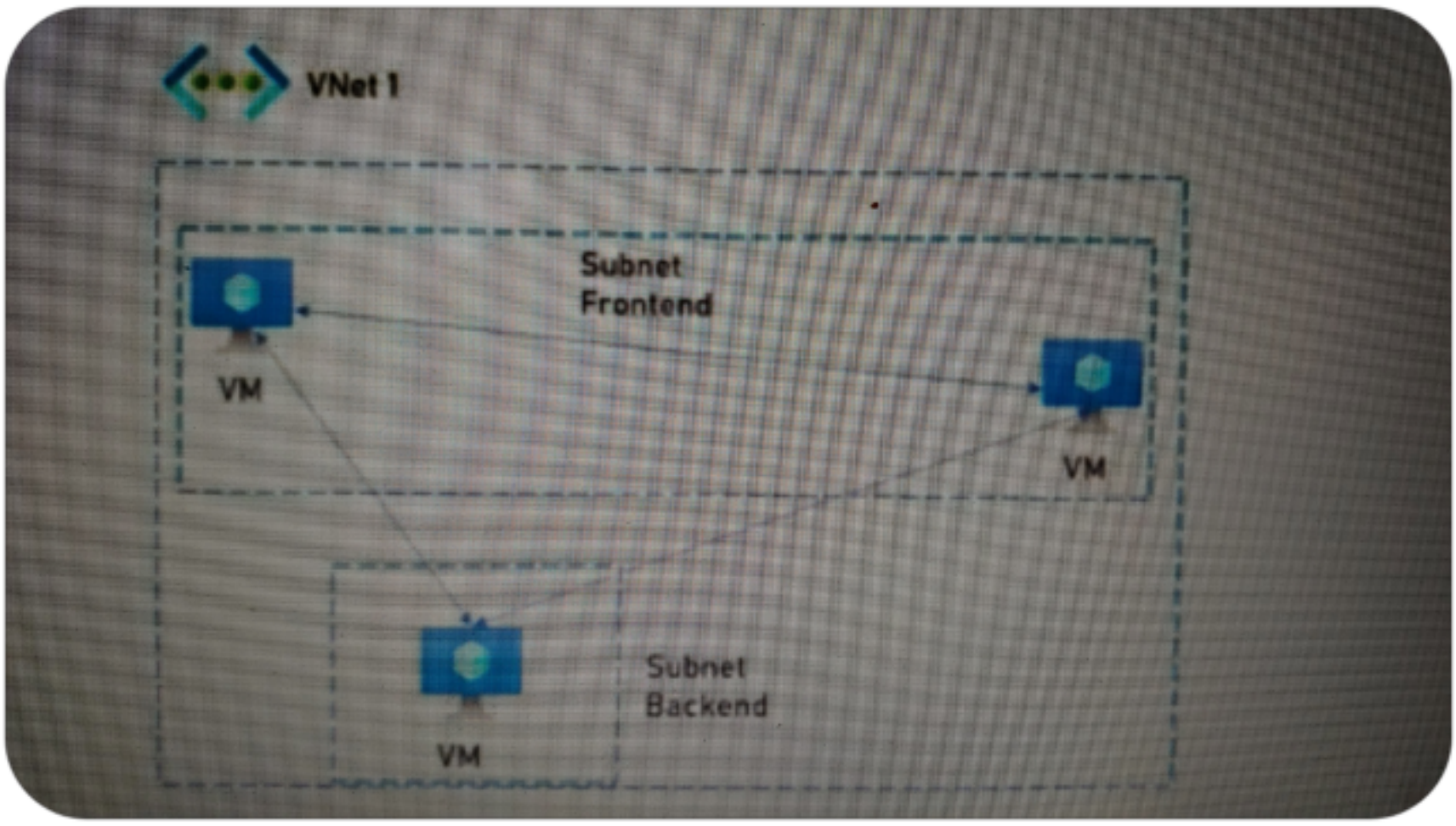
- The good news is we don't have to remember these numbers. A lot of CIDR calculator available.
- Also, Azure usually shows the actual range



SubNets

- It's a logical segment in the VNet. It shares a subset of temhe VNet's IP range. It is used as a logical group of resources in the VNet.

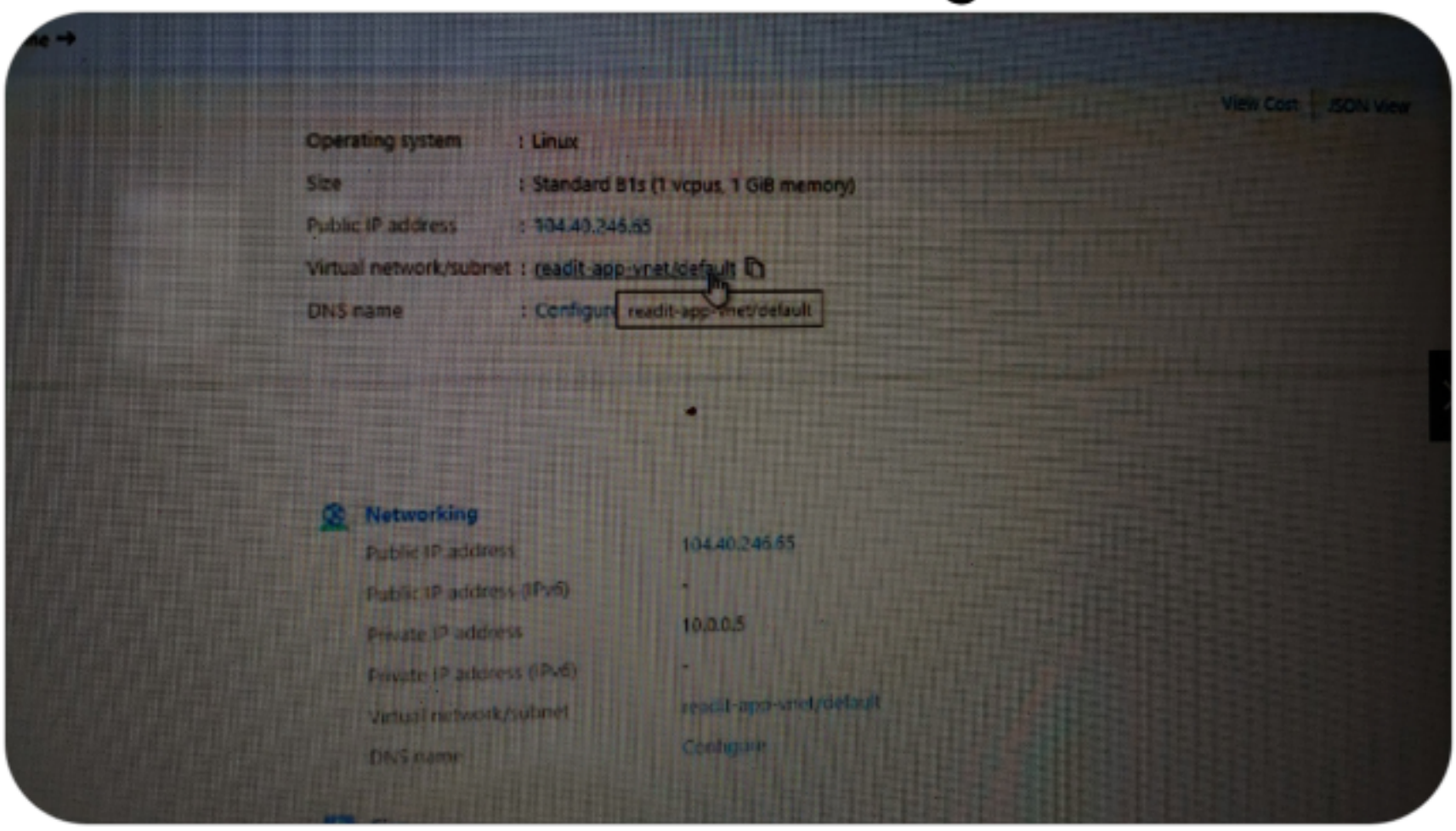
- SubNets are must. Resources must be placed in a Subnet, cannot be placed directly in the VNet. Resources in a subnet can talk to resources in other SubNets in the same VNets.



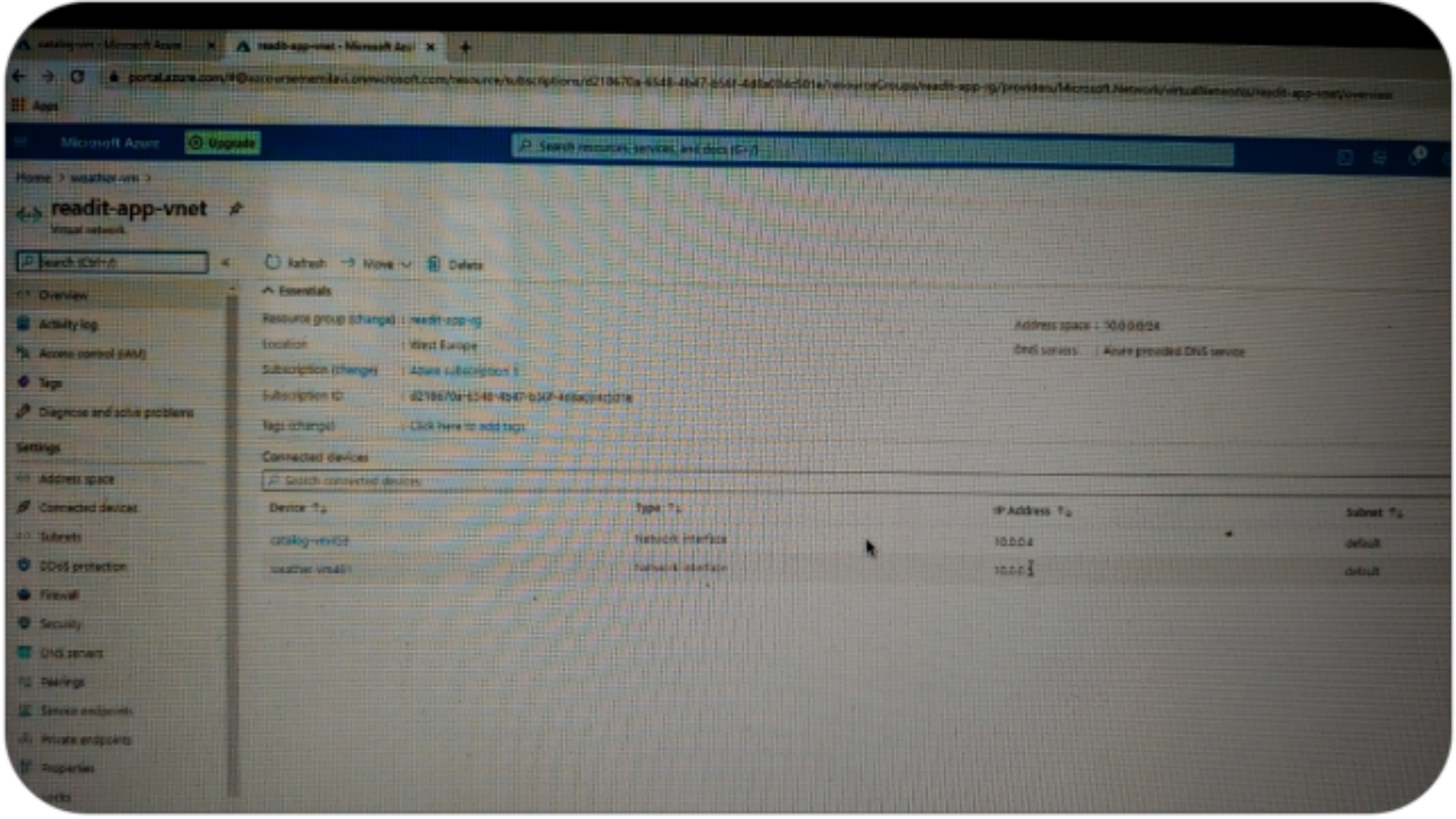
- **Address of Subnet**: Each Subnet gets a share of the parent VNet's IP range. Never use the full range of the VNet in a Subnet.
- Subnet Pricing: Free and limit of 3000 Subnets per VNet.

Practical (VNet and SubNets)

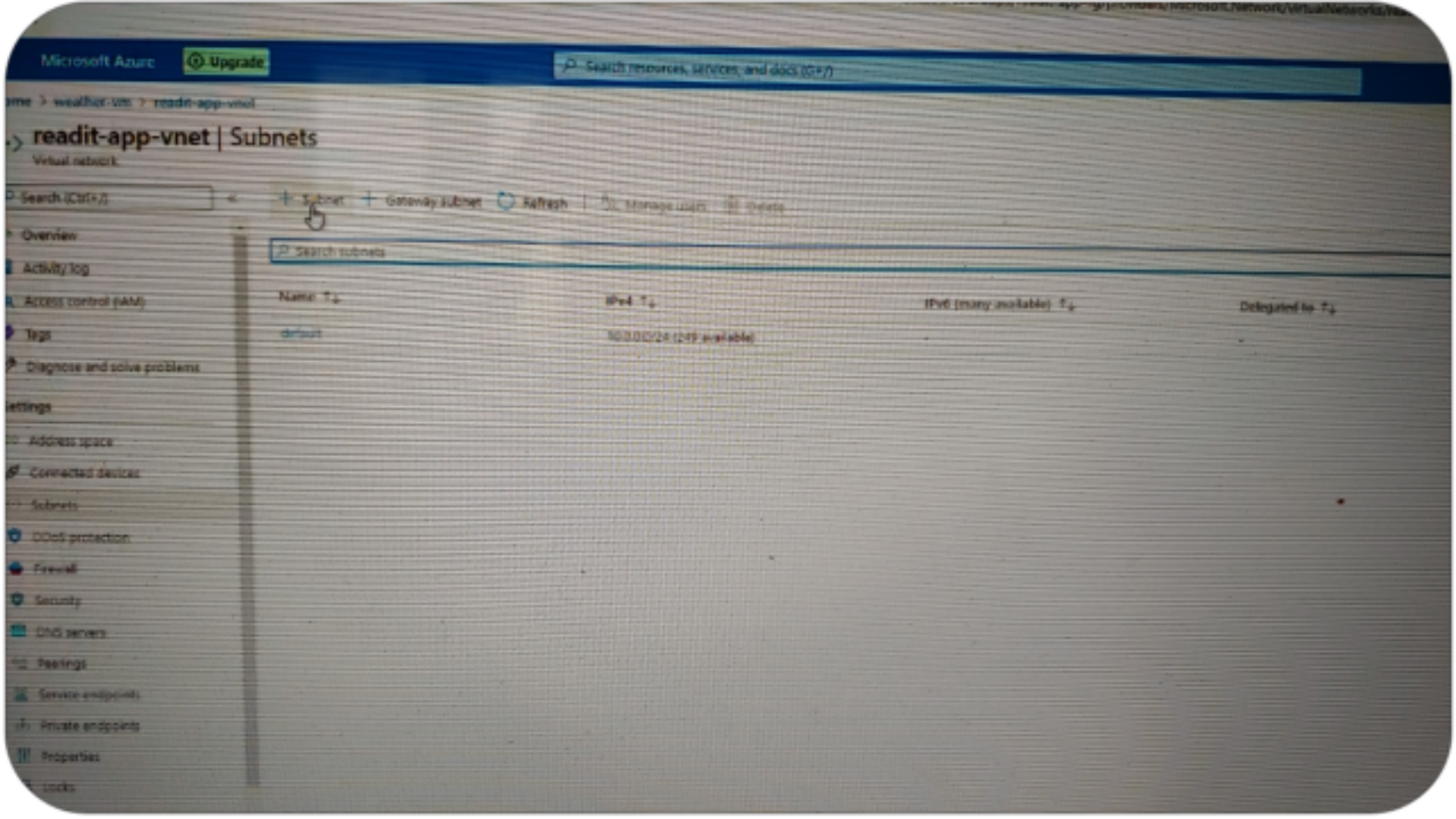
- When we create a new Virtual machine in Azure, we have to create a new Virtual Network first and assign it to VM. When we create a new VNet, a new Subnet called 'default' also get created



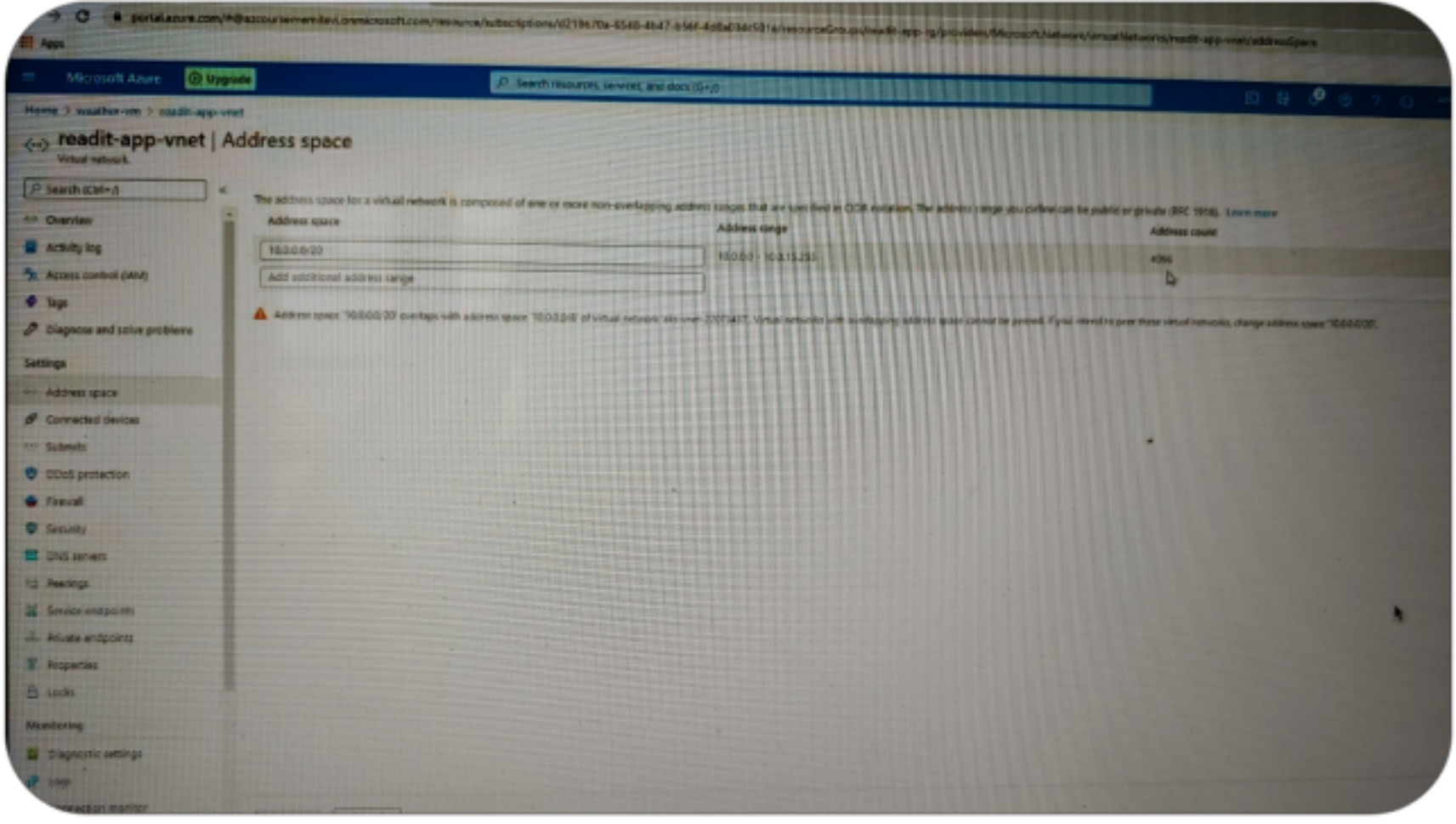
- VNet has a address space like 10.0.0.0/24 which means we can have up to 256 network devices, out of which currently 2 Network interfaces are created one for each of the created VM. Network interface connects VM to VNet.



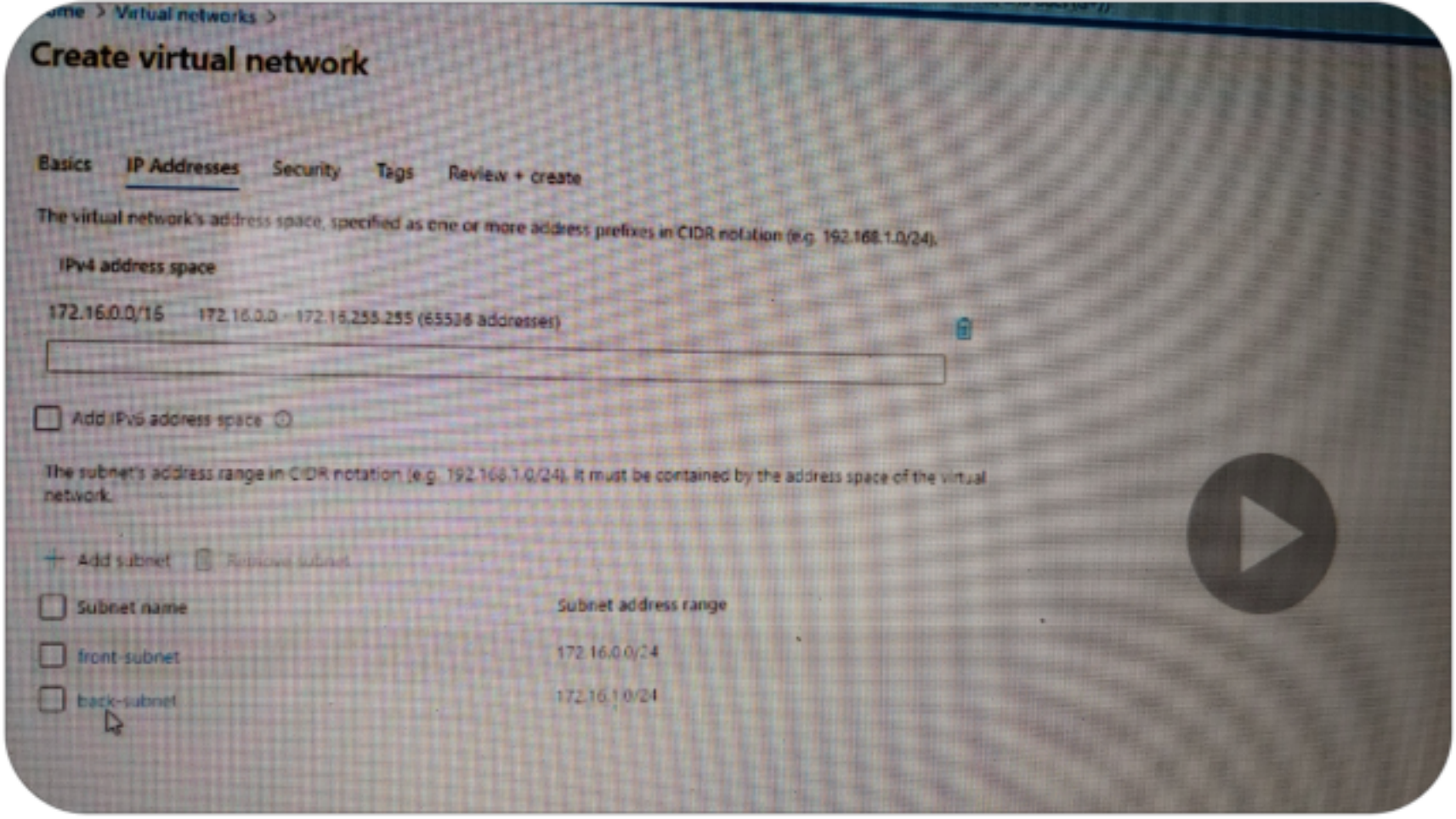
- By default a subnet named 'default' is created for the newly created VNet 'readit-app-vnet'. And this subnet took all the address space of its VNet as its address range is '10.0.0.0/24'. So now we can not create any additional subnet in this VNet.



≡ ○ Now if we want to create a new subnet, then go to the Address space of the VNet and increase its address space from '10.0.0.0/24' to '10.0.0.0/20'. Now the address counts will increase from 256 to 4096.



- Note: When azure creates a Subnet, it reserves 5 out of the available addresses of Subnet.
- **Create VNet in Azure with multiple Subnets**: Select the right region and subscription. Then we can change the Subnet name from 'default' to any valid name like 'front-subnet', or add any new multiple SubNets. Note that each Subnet will have its own address range shared from VNet.



≡ ○ Now we can create 2 new VMs by selecting the same VNet 'my-vnet'. For the first VM 'front-vm' select the Subnet as 'front-subnet' and for the second VM 'back-vm' select the Subnet as 'back-subnet'. Now login to 'front-vm' VM through RDP file. Copy the private IP address of the 'back-vm' VM and connect to it through Remote Desktop Connection option. This will show we can communicate from one subnet to another subnet within the same VNet.

Network Security Group (NSG)

- Its a gatekeeper for Subnets or a VM, which defines who can connect in and out of Subnet/VM. In other way you can think of a mini-firewall for Subnets.
- Basically it provides required security at the VM/Subnet level so that we can restrict/allow any action on VM or services which are running in VM.
- We can attach a NSG at a Subnet level or at a VM level. Whenever we create a subnet/VM, we should also attach a NSG to it. It is free of cost.
- **How NSG works:** We define following Security Rules, based on which the NSG either allows or denies the connection.

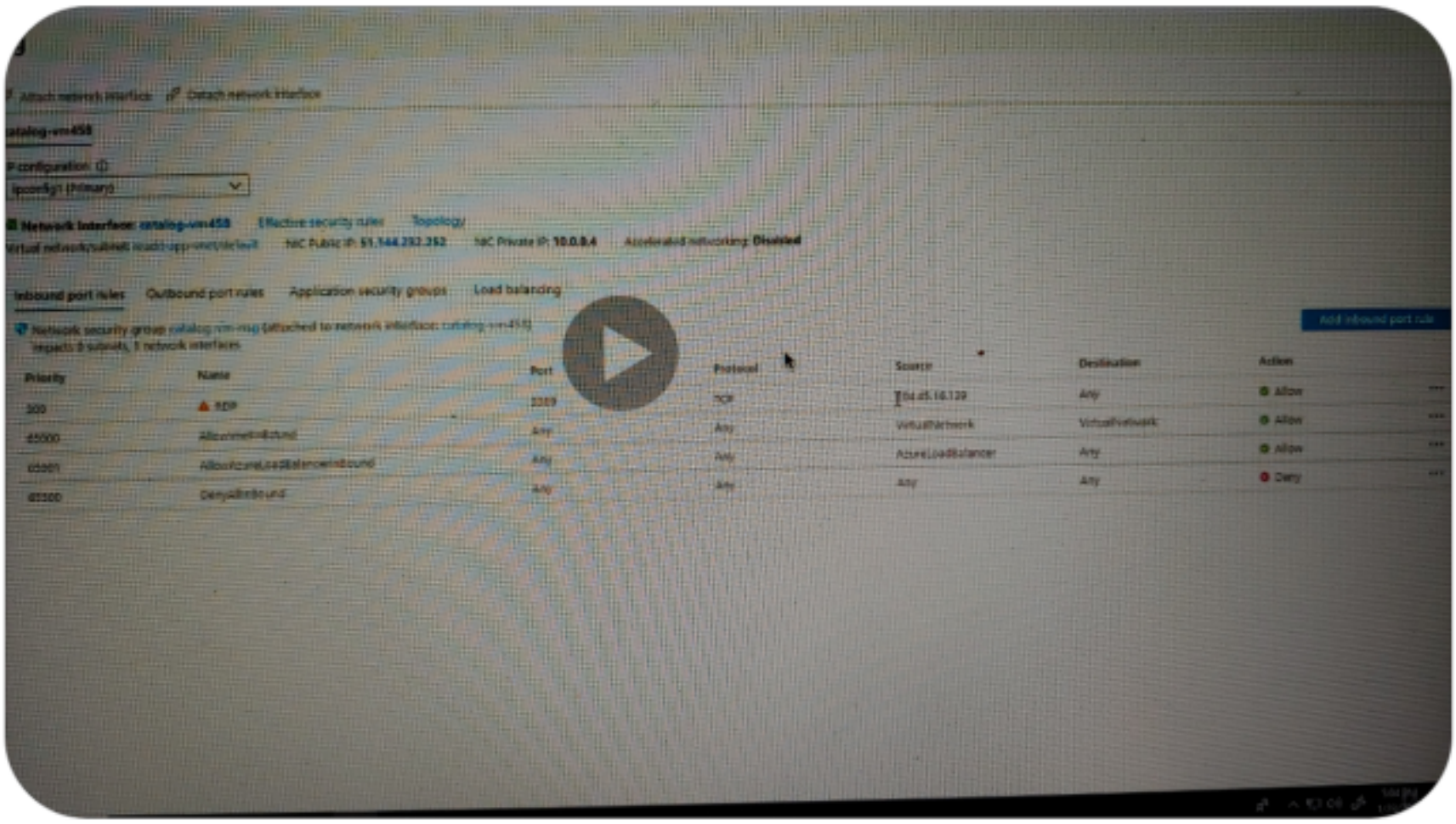
- Looks at 5 tuples:
  - Source (=Where did the connection come from)
  - Source Port (=The port the source is using)
  - Destination (=Where does the connection request goes)
  - Destination Port (=To which port does it want to connect)
  - Protocol (=TCP, UDP, Both)

- Each rule is assigned a number and lower the number - the higher the priority of the rule.
- **NSG and VMs:** When we create a new VM, NSG also automatically created and attached to that VM's Network Interface with some set of predefined Security Rules.
- By default NSG allows Opening RDP(on Windows) or SSH (on linux) port to anyone. So this **must** be handled after creation.

NSG in Action

- By default when we create a new

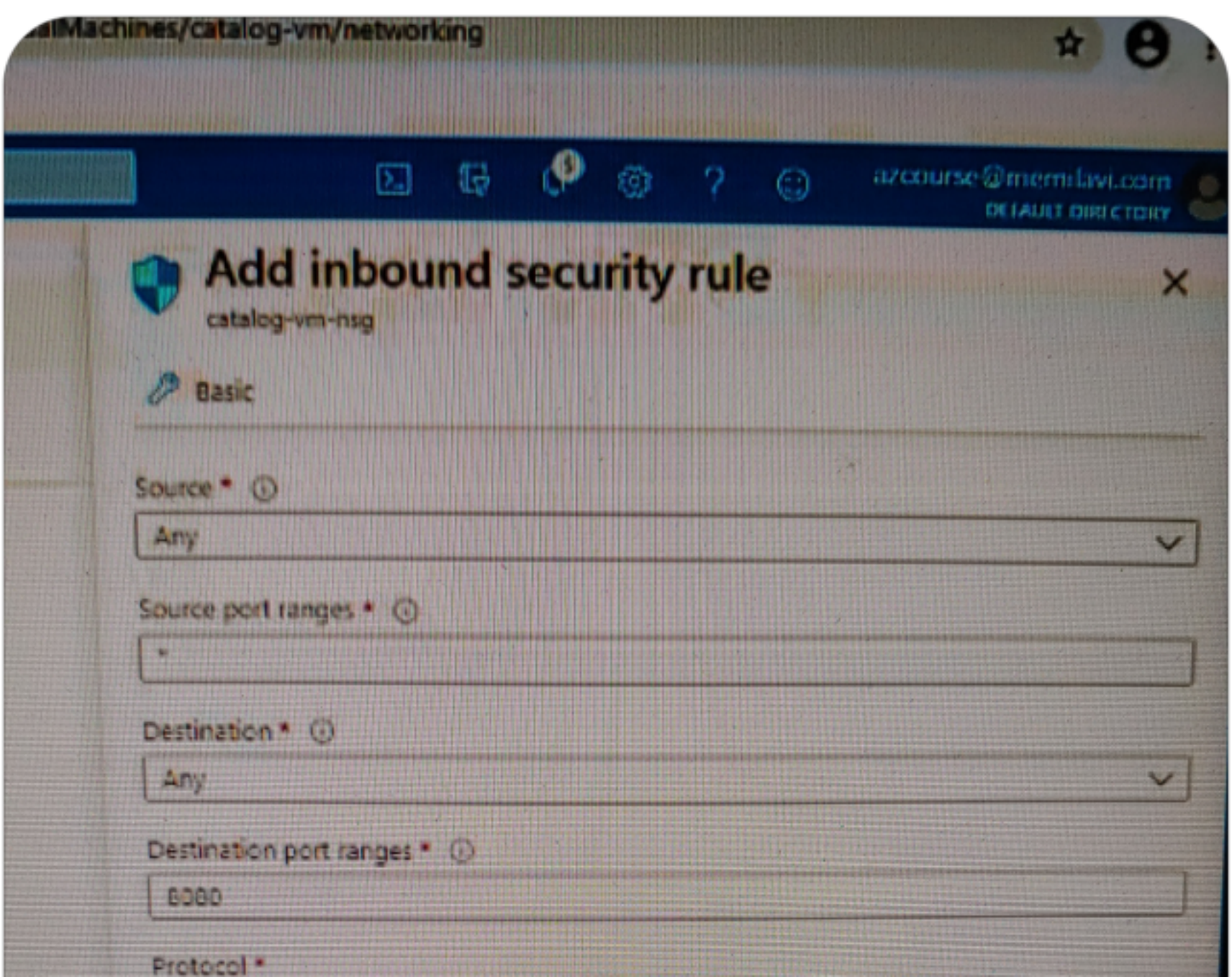
VM, NSG will be created and attached to VM's Network Interface with set of default Inbound port rules and Outbound port rules.

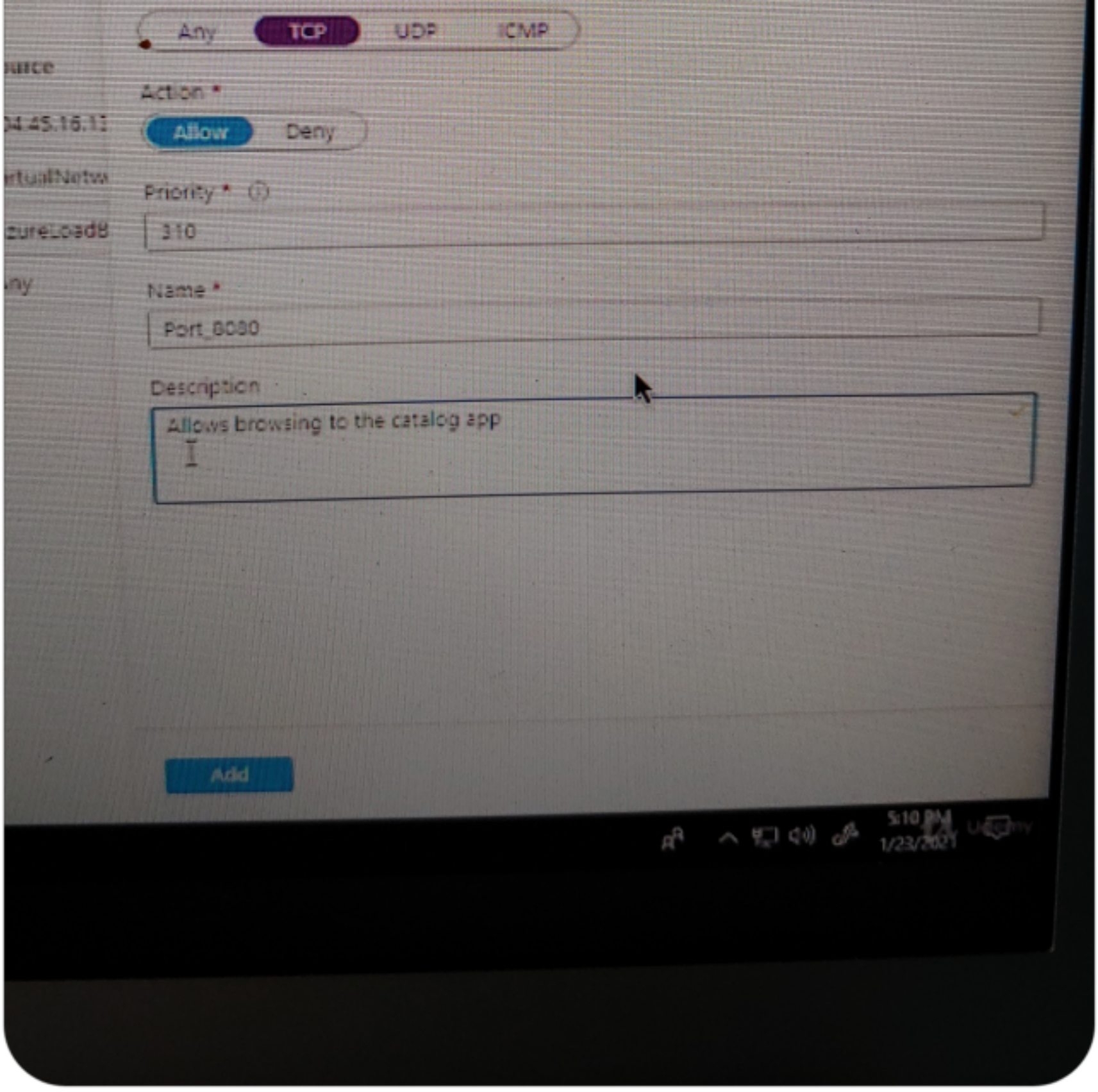


- Each of these rules will have a name & defines a specific action and this action can be configured to either allow or deny. Each rules will also have a priority number, port, protocol, source and destination details which can be configured. For example, we can restrict the RDP rule and configure it to allow only for a

specific host machine IP address, instead of allowing anyone to access the VM through RDP file.

≡ ○ Another example for configuring inbound security rule is to allow accessing of a web application which is running in VM inside port 8080. Now go to the NSG rules defined for this VM, and add a new rule with destination port as 8080, protocol as TCP and action as Allow.



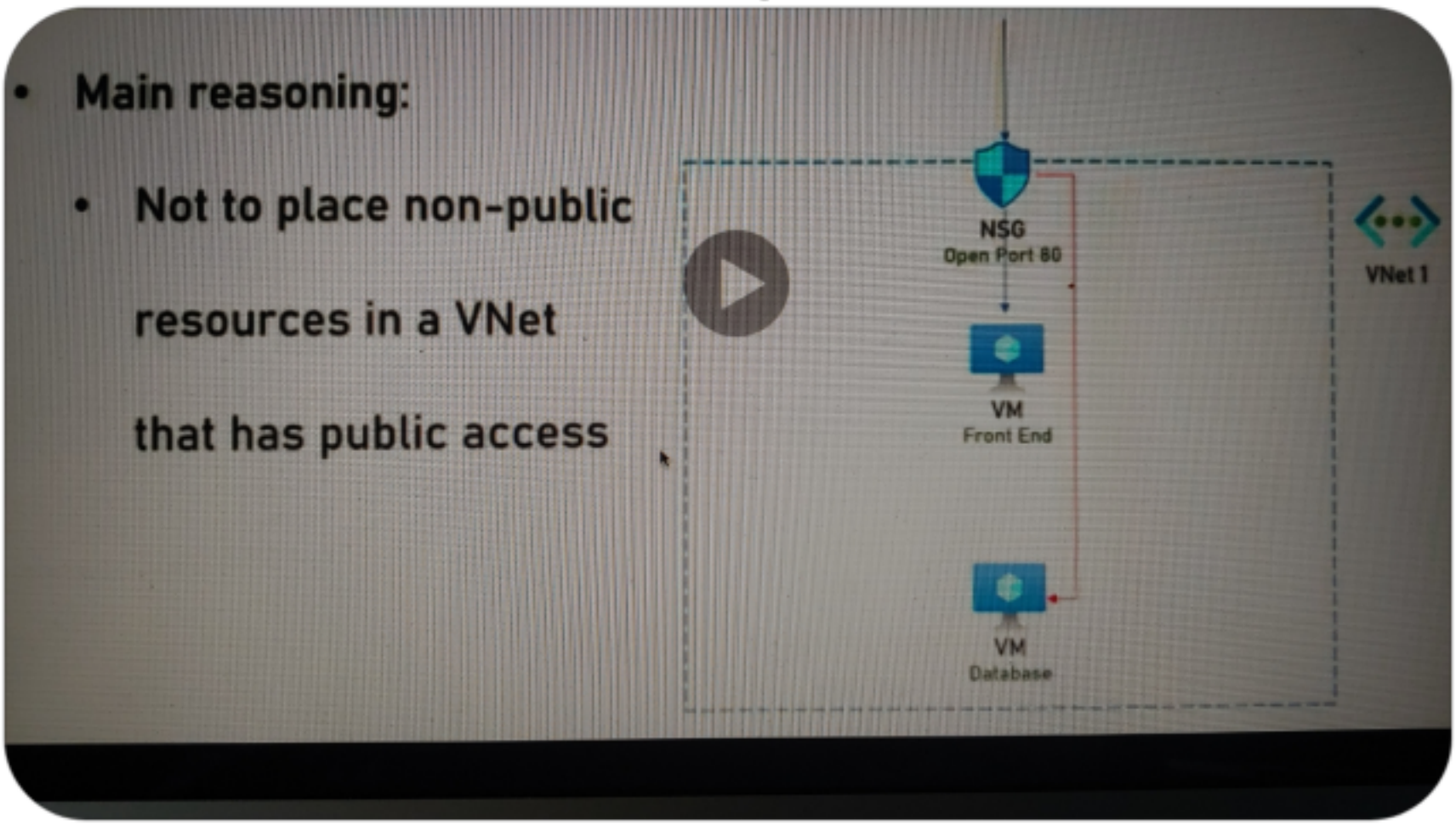


Network Peering

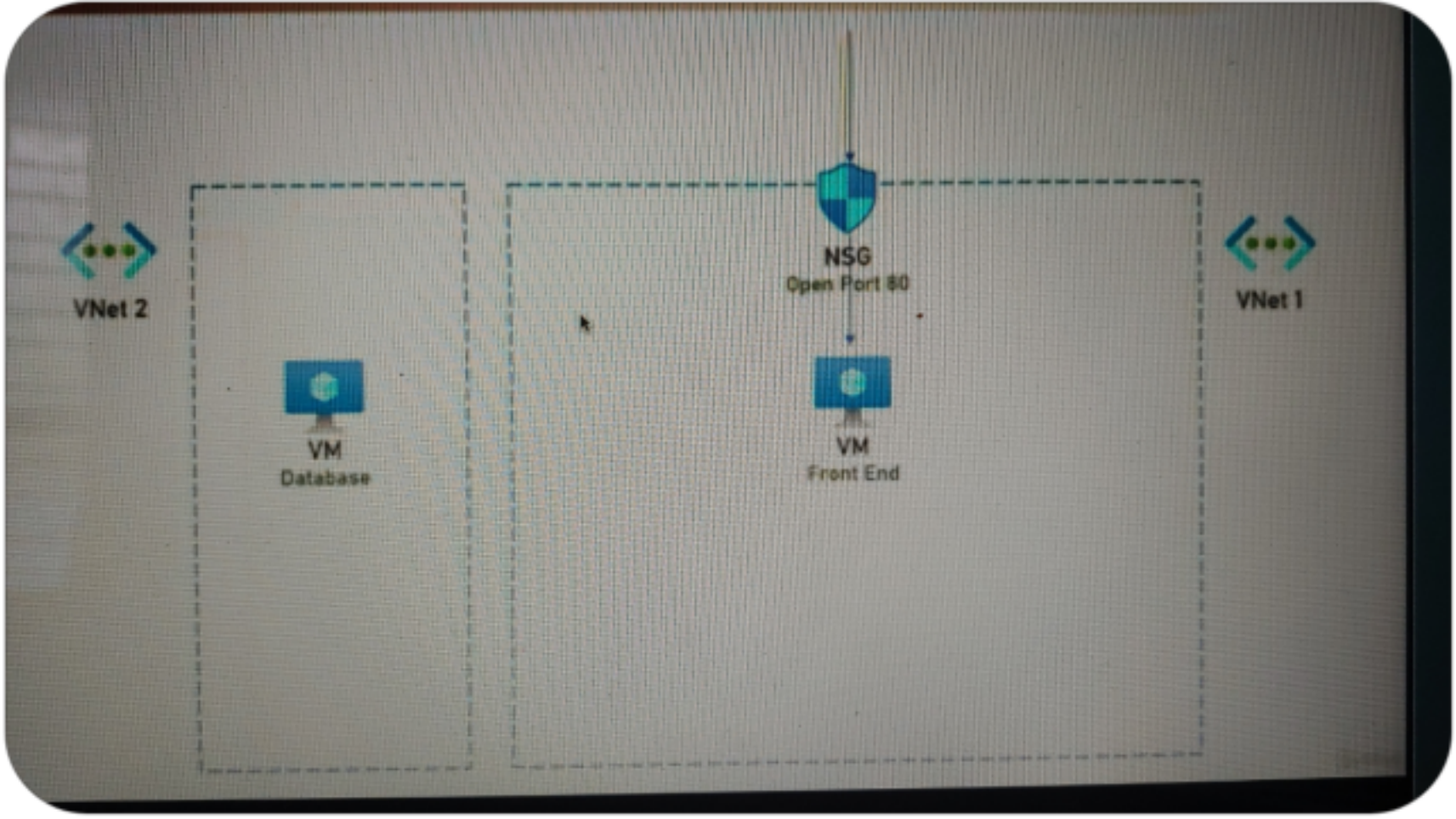
- Sometimes, to increase the security, we want to place some resources in a completely different VNet not in a same VNet . Placing in 2 different

Subnets within a same VNet will not be enough.

- For example Separate systems, System layers(like front-end back-end db), Sensitive dbs
- **Main reasoning for N/W Peering**: User might find a way to get access to non-public resources

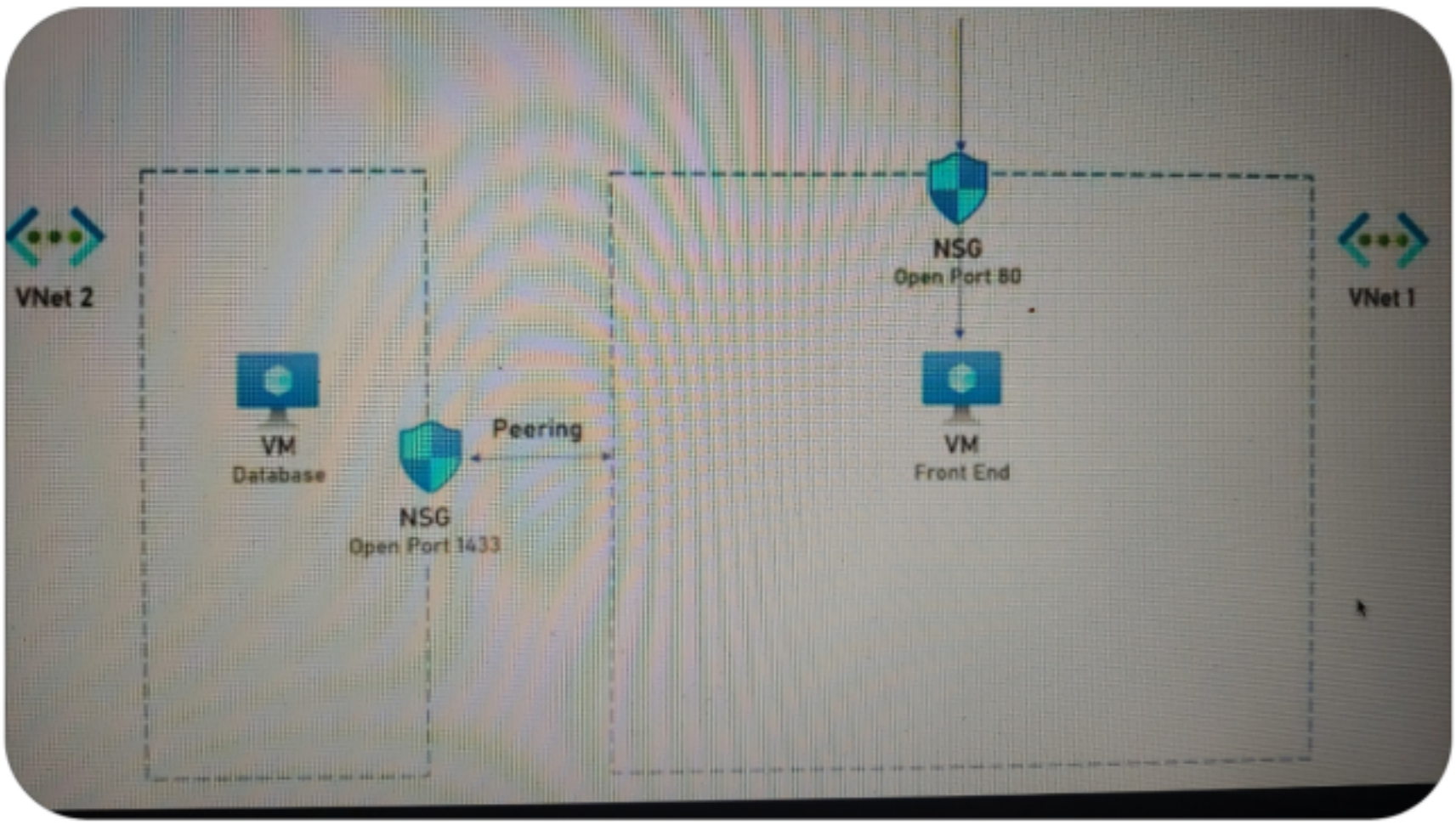


- Solution is to place sensitive resources into a VNet which does not have public access.



- But, this has a problem that resources in VNet can not communicate with resources in other VNets. So the VM in VNet1 in the above diagram can not communicate with VM in VNet2.
- **Network Peering**: The solution is to use Network Peering, that allows two VNets to communicate each other. From users point of view its a single VNet. Make sure address spaces of 2 VNets are not overlapped. Use NSG for protection while peering.

- By using peering we can connect multiple VNets across multiple regions. Network Peering is not for free. With Peering our architecture looks like this:

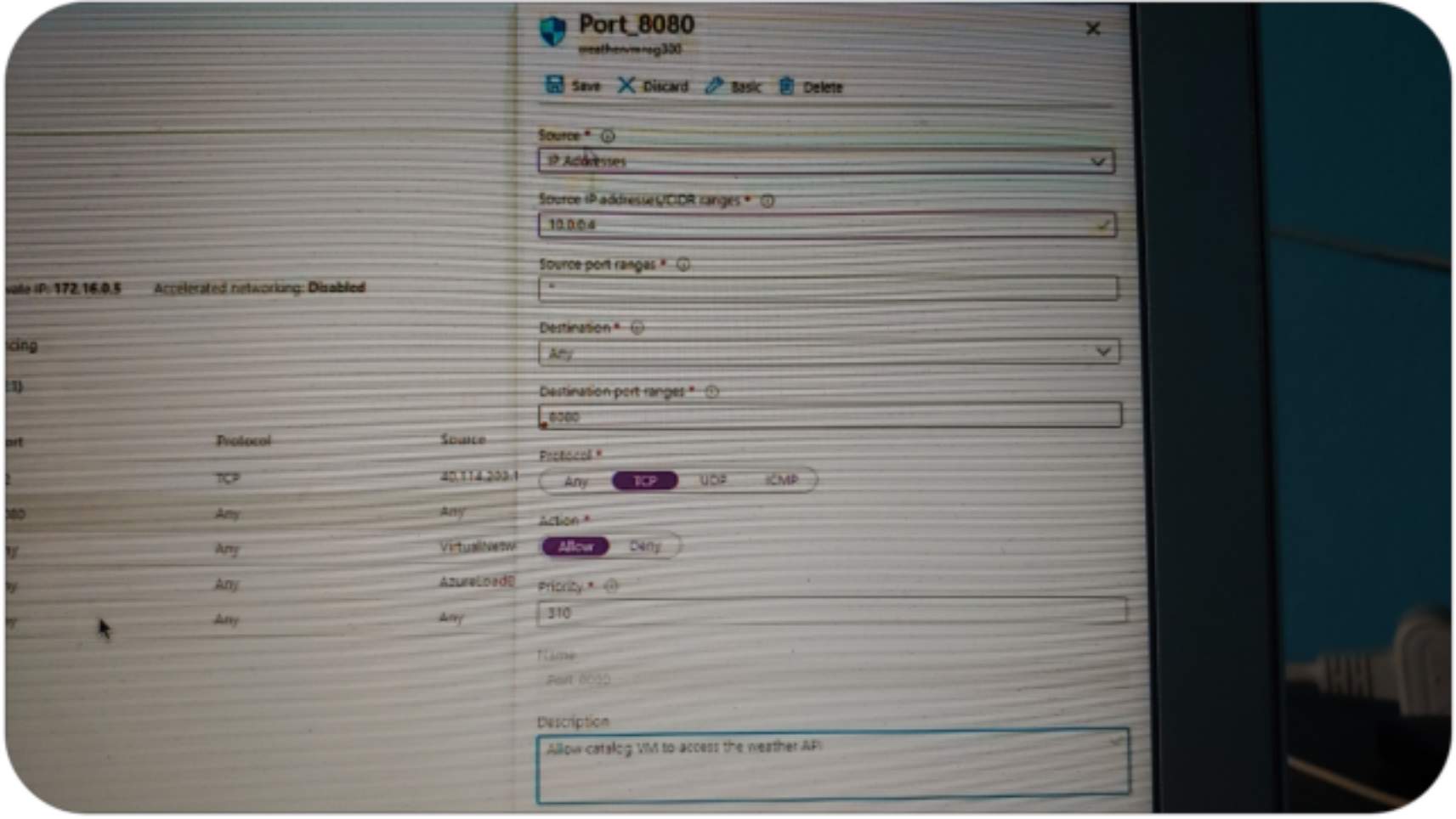


Network Peering In Action:

- Create 2 VNets with single Subnet. Create a single VM in each of these VNets, and install some service in it. Now both of these services can not communicate between each other.
- Now go to first VNet, select the Peering option and click on Add.

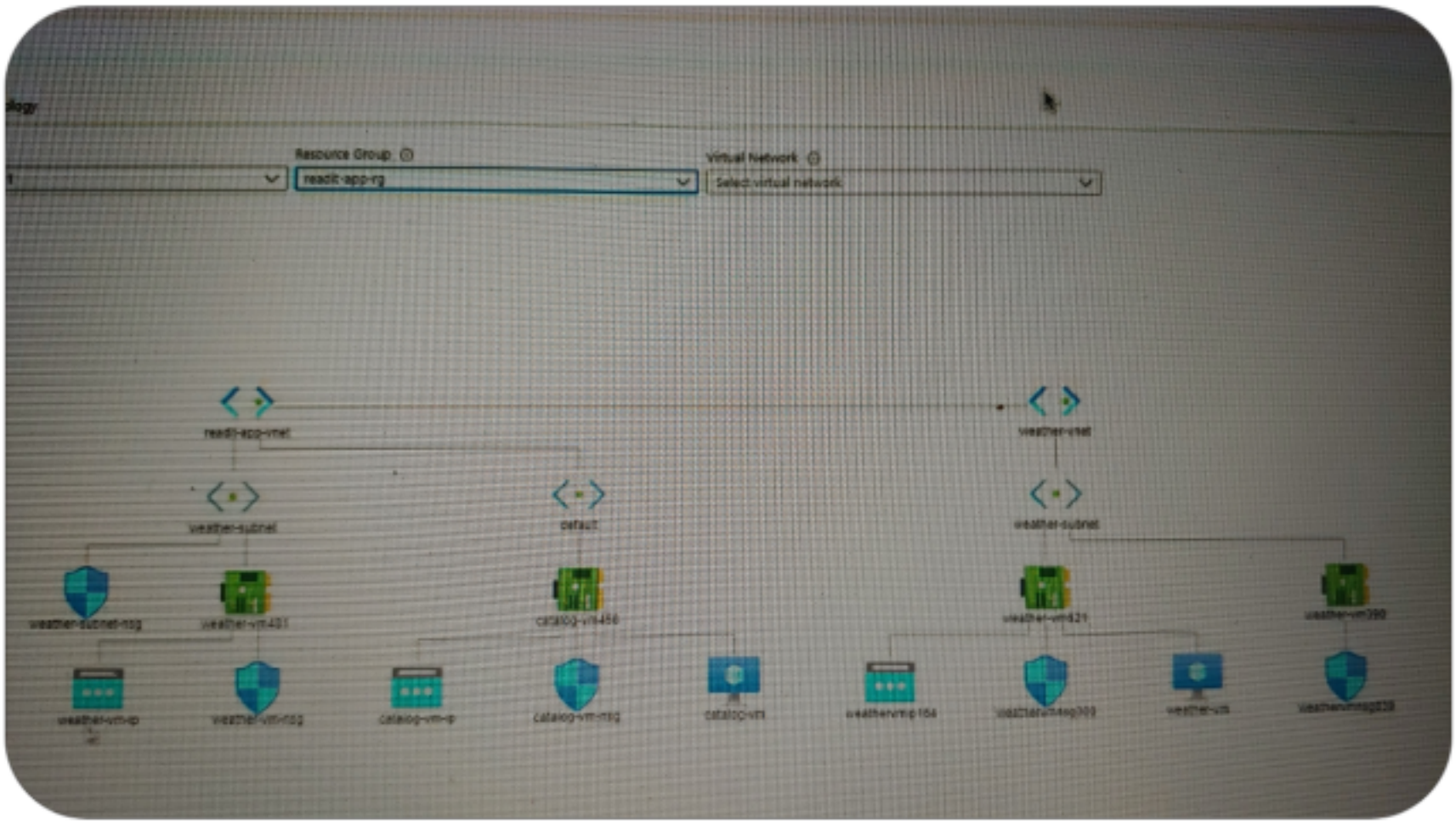
Enter the details like Peering name from current VNet to other VNet and also from other VNet to current VNet. This will create 2 Network Peering.

- ○ Finally, we can add NSG rule in the Subnet/VM of second VNet, which defines that, an application which is running in the VM of second VNet on the port 8080 is accessible only from the private IP address of the VM which is running in the first VNet and add this private network value to the Source IP address field of the rule.

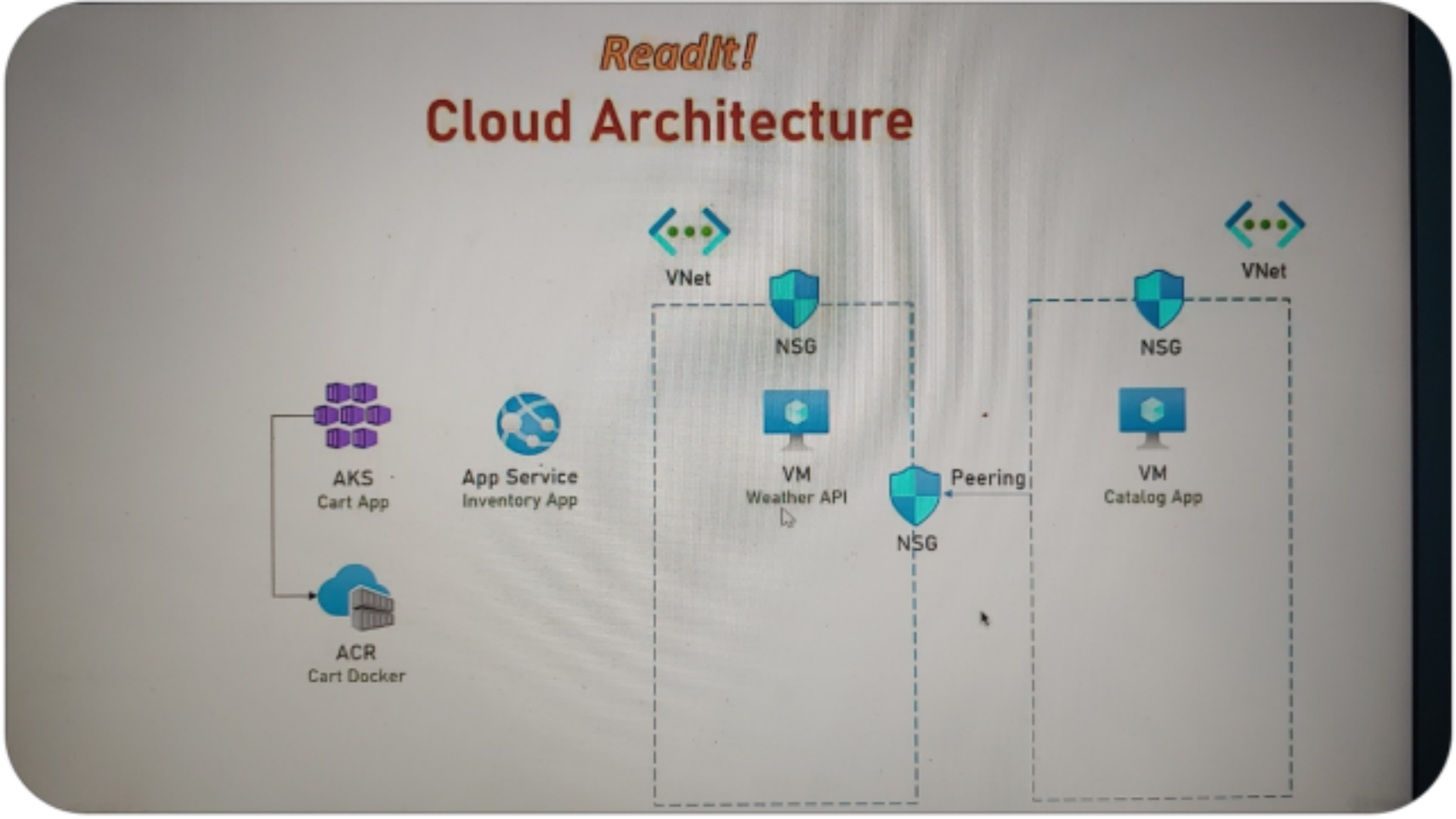


- So what we have now is, we have 2 VM(one is running catalog app and another one is running weather app), each in a separate VNet and these VNets are connected using Network peering and we have working NSG infront of the Second VNet that protects the VM and allows only traffic from a specific IP address to access this VM.
- **Network Topology**: after adding lot of VNets, Subnets and resources we will get confused. For that we have a option in

portal called Network Watcher which has an option called Topology. In this topology click on the resource group which contains the network you want to see. This will shows current structure of entire networks.



Current Architecture



- Each of these VMs are placed inside its own VNet with an NSG and a Network Peering that is defined between those VNets. This makes our architecture much more robust and secure.

Secure VM Access

- The services which are running in a VM can expose a public API which are considered as Attack Surface since hackers can use these and enter VM and harm the entire system in any way. Also,

another possible Attack Surface for the hackers is allowing users to make RDP/SSH access to VM.

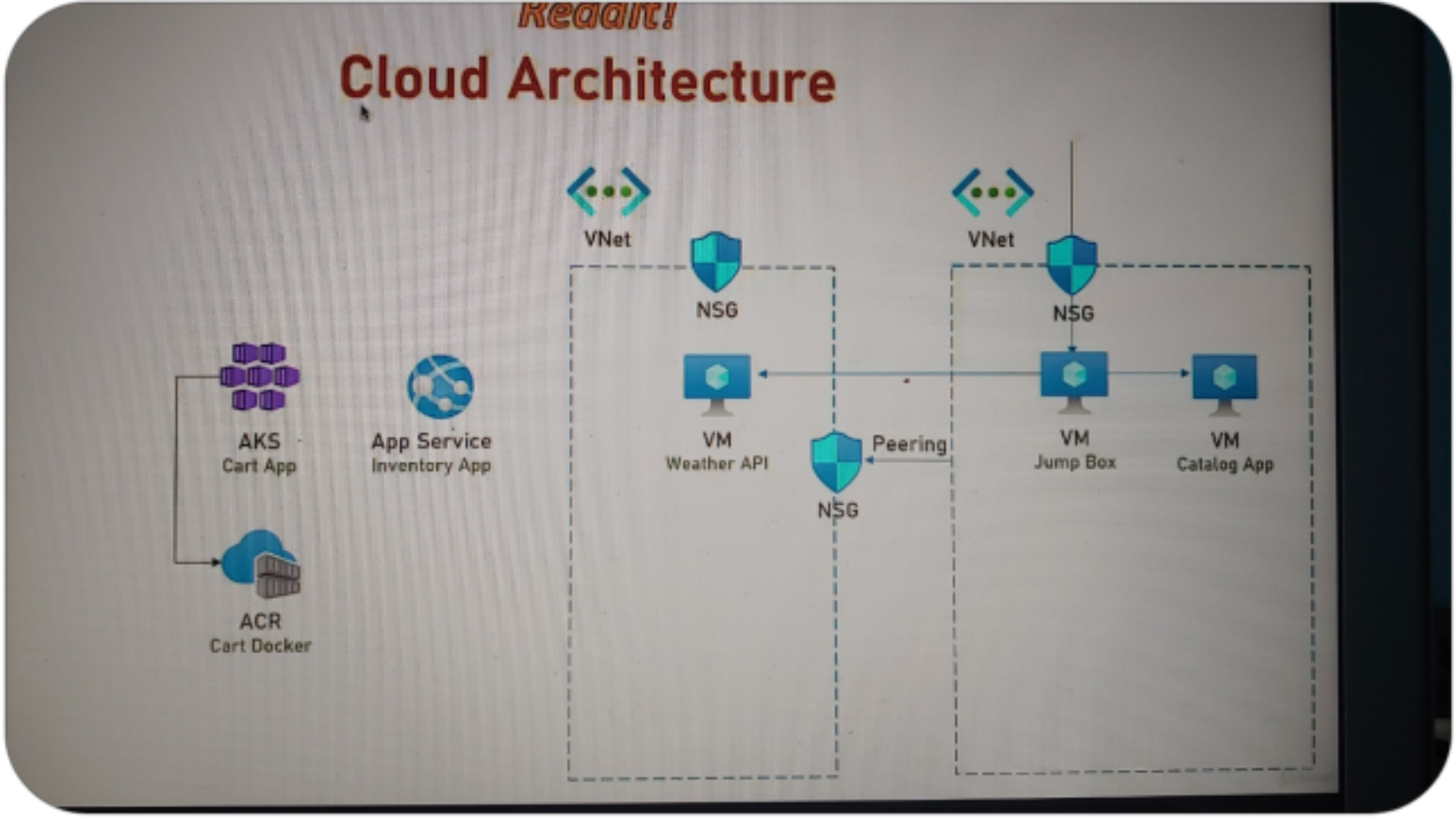
- We can minimize the risk causes by exposing the public IPs to internet so that APIs will be called or connect to VM either by using RDP or SSH.

RDP/SSH connect:

- The risks causes by connecting to VM through RDP or SSH can be reduced by following some techniques like JIT Access, VPN, Jump Box, Bastion.
- **Just In Time Access:** Opens the port for access on demand, and automatically closes it. Rest of the time - its closed. It can be configured from the VM's page in the Azure Portal.
- **VPN**: Its a secure tunnel to the VNet and can be configured so that no one else can connect to

the VNet. This requires VPN software and license which are not part of Azure, and hence its quiet complex and expensive.

- **Jump Box:** Place another VM in the VNet and allow access only to this newly created VM in the VNet. If we need access to one of the other VM in the VNet connect to this newly created VM and from that VM connect to the relevant VM. So at a time we are exposing port of only one newly created VM, other VMs ports are secured. So Jump Box is the one which is exposed to internet and rest of the VMs are connected through this Jump Box. Cost of this is for the additional VM.



- **Bastion**: Its a web based connection to the VM. No open port is required for this. Its simple and very secured. And it costs more money (140$/month).
- We can create a new Bastion service from the Portal directly. While creating we must need to select the VNet where this Bastion service will be installed.
- Now go to VM page in Portal and choose Bastion option instead of RDP from connect option and provide VM's username and password to connect. This will

open the VM's page in our web page from where we can access the entire VM.

- Note that Bastion will work even though we disable RDP and SSH ports through NSG settings.

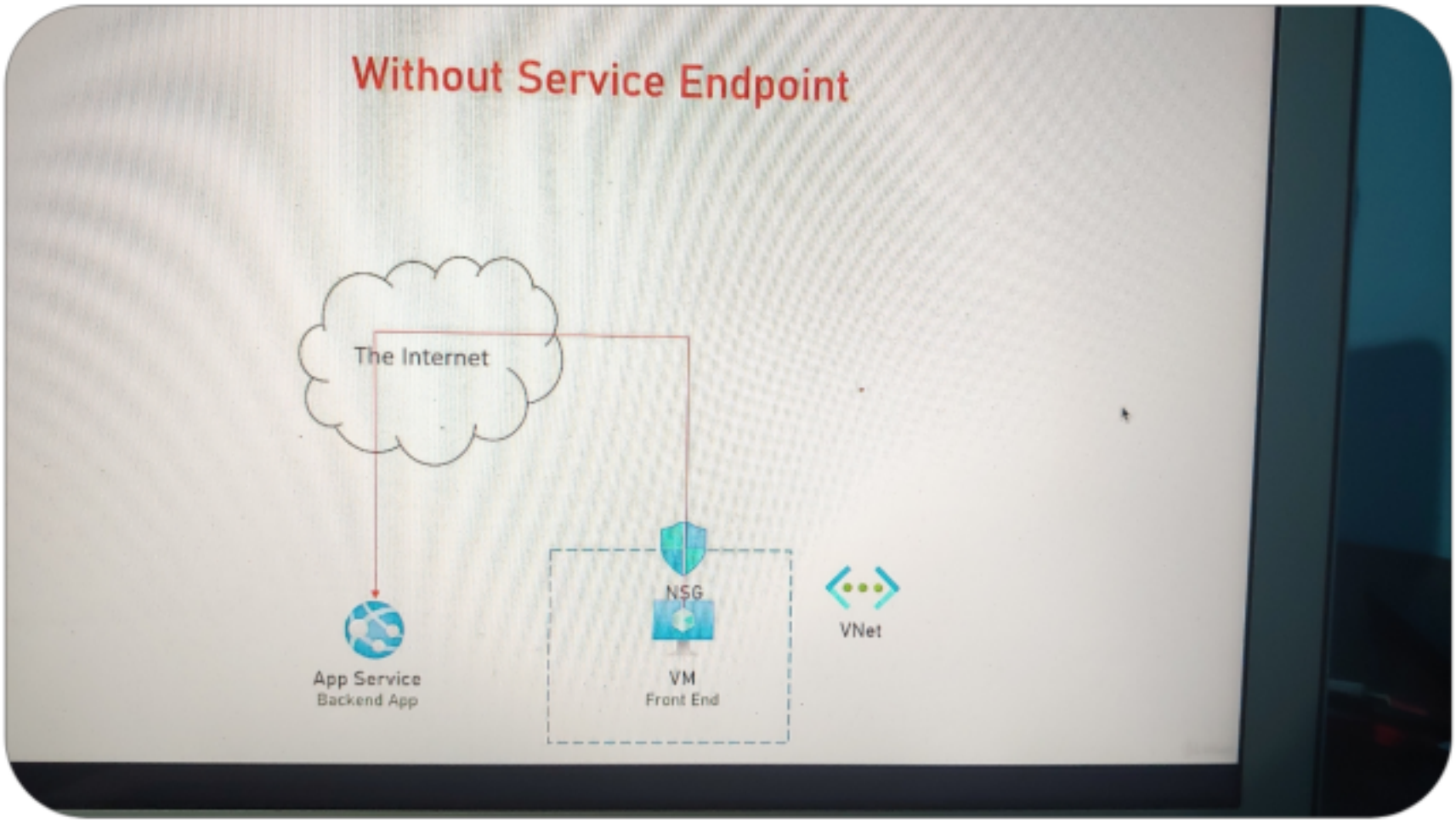
Service Endpoint

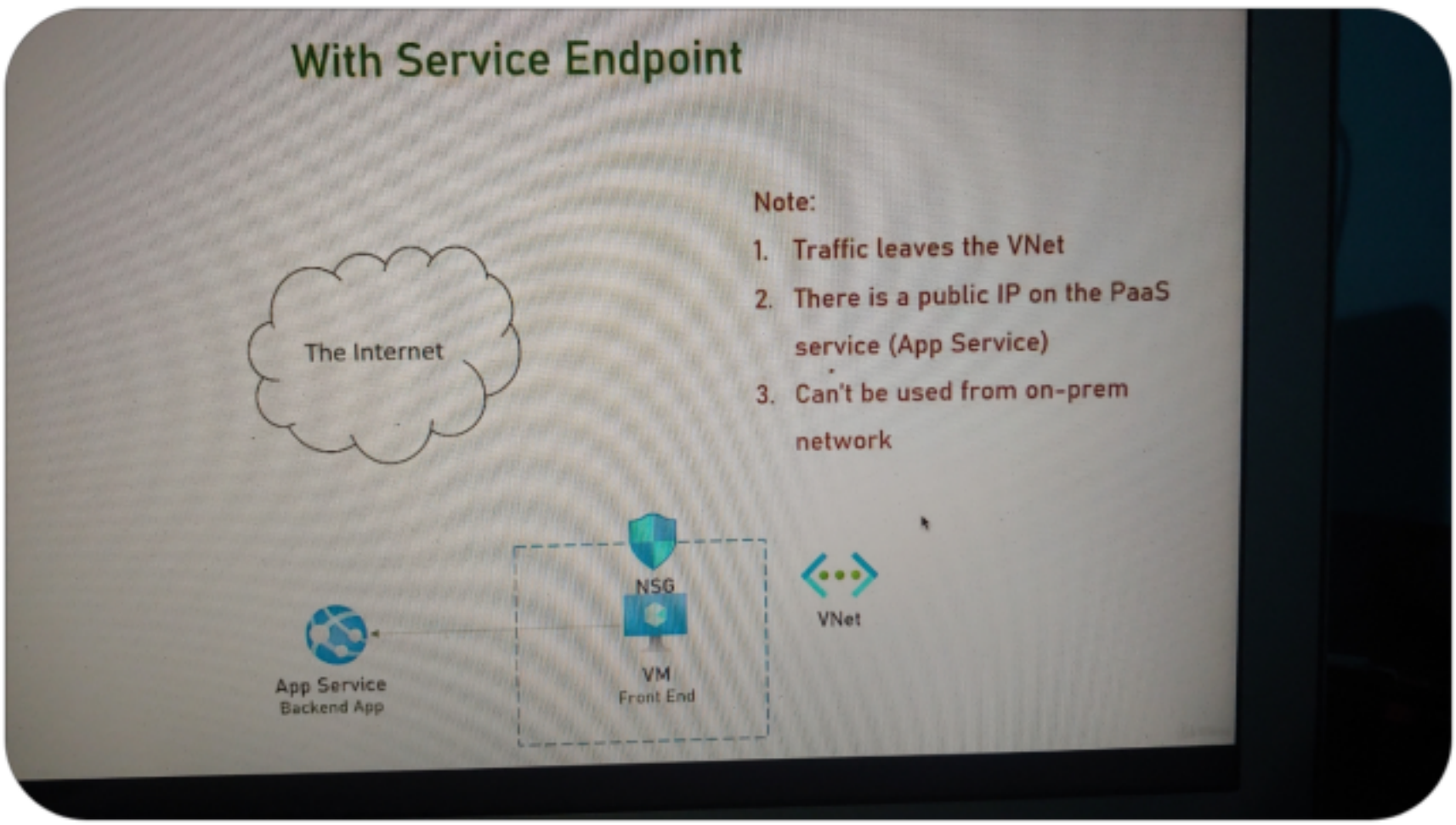
- A lot of managed services like Azure SQL Server, App Services, Storage, VM etc exposes public IP to internet. Sometimes these resources are accessed only from resources in the cloud. For example databases in the cloud are accesses only by some back end service in the cloud.
- **Service Endpoint**: IPs exposed by these managed services might pose a security risk and which can be solved by Service Endpoint. It creates a route from the VNet, where the resource connecting to the managed

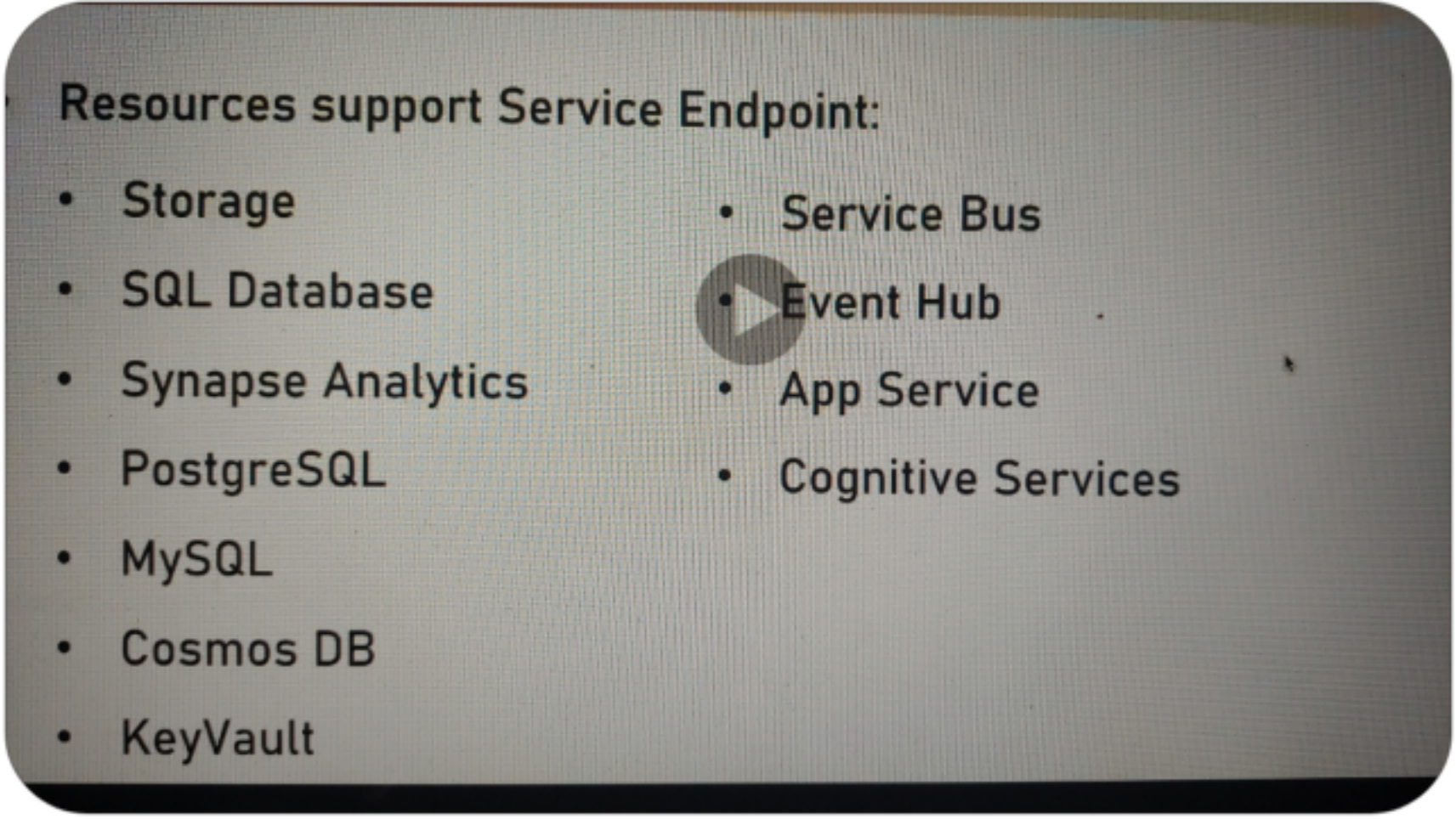
service exists to the managed service itself. That means the traffic never leaves the Azure backbone, although the resource still has a public IP exposes to the internet and the access from internet can blocked

- Basically what it means is, now we have a direct connection from a calling resource for example VM to the called managed service for example database. These are basically for free.
- **How it's done:** Enable Service Endpoint on the Subnet from which you want to access the resource for ex VM, and on the managed service for example database, set the Subnet as the source of traffic. After that we will be having a service endpoint connecting the VM and database directly without going through

internet.


Without Service Endpoint
The Internet
NSG
VNet
App Service
Backend App
VM
Front End

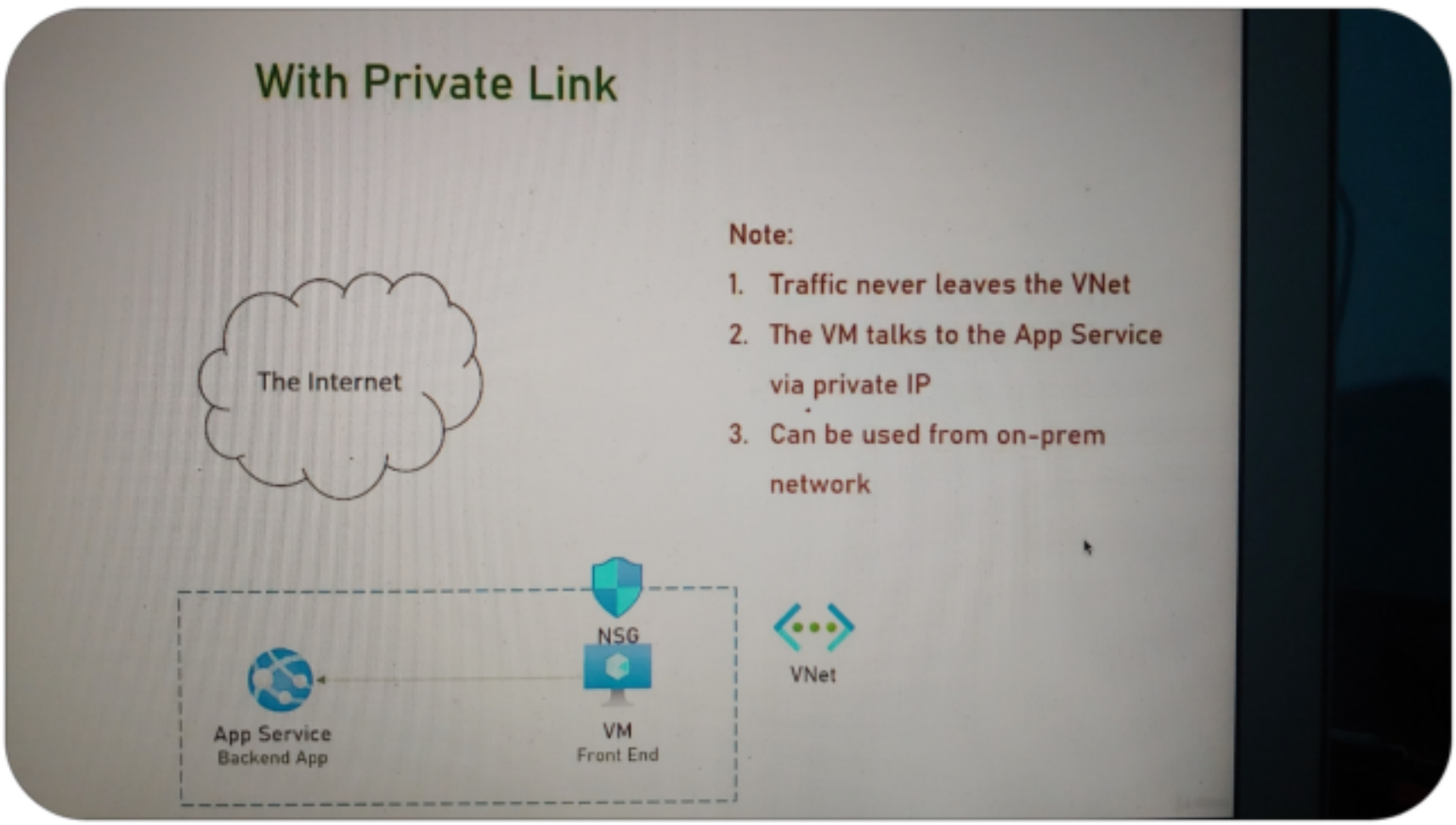

With Service Endpoint
Note:
1. Traffic leaves the VNet
2. There is a public IP on the PaaS service (App Service)
3. Can't be used from on-prem network
The Internet
NSG
VNet
App Service
Backend App
VM
Front End



Private Link

- It provides a newer and better solution for the same problem for which we used Service Endpoint previously.
- It extends the managed service into the VNet so traffic never leaves the VNet and it stays private all the time. That means access from the internet can be safely blocked on these managed services. But its not for free.
- **How its done:** Configure the managed services to connect to the VNet where the VM which

makes the call to these managed services, is located and then we basically create a private link between VM and Managed service like database. Then we need to configure private FNS which will handle routing between resources.



Private Link

- Resources support Private Link:
  - Storage
  - SQL Database
  - Synapse Analytics
  - PostgreSQL
  - MySQL
  - Cosmos DB
  - KeyVault
  - Redis
  - AKS
  - Search
  - ACR
  - App Configuration
  - Backup
  - Service Bus
  - Event Hub
  - Monitor
  - Relay
  - Event Grid
  - App Service
  - Machine Learning
  - Automation
  - IOT Hub
  - SignalR
  - Batch

Service Endpoint vs Private Link:

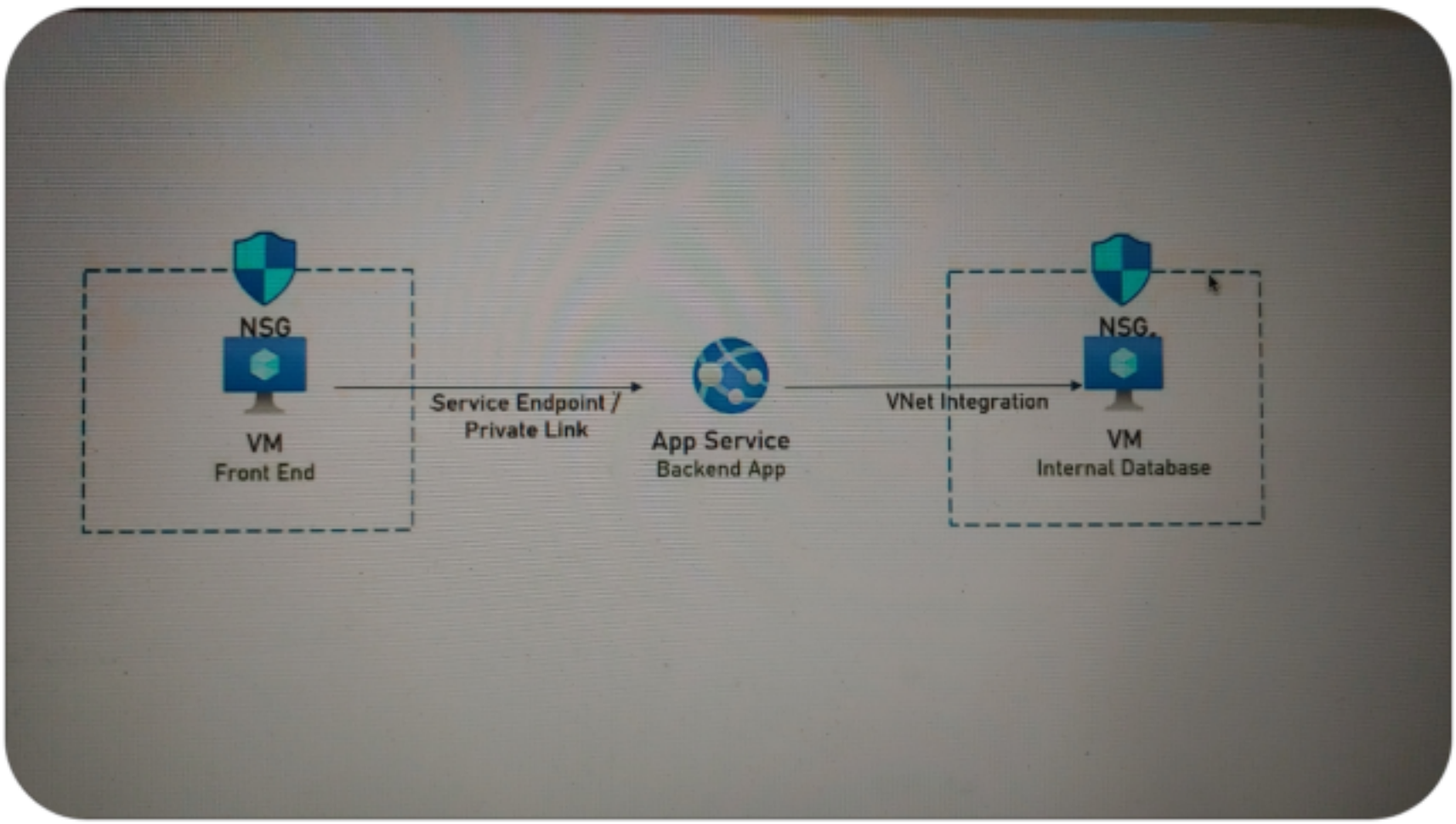
Service Endpoint vs Private Link

	Service Endpoint	Private Link
Security	Connects via Public IP	Connects via Private IP
Simplicity	Very simple	More complex
Price	Free	Not free
Supported services	Limited list	Large list, probably will get larger
On-Prem connectivity	Quite complex	Supported

App Service VNet Integration

- Allows access from App Services to resources within VNet, so that these resources should not be exposes on the internet.

- Useful when App Services needs to connect to an VNet and access to a VM which is running in it with some internal resources.
- Supports same region VNets. For VNets in other regions - a gateway is required.
- **SE/PL vs VNet Integration**: SE/PL manages upstream connection from VNet to managed services where as VNet integration manages downstream connection from managed services to VNet.

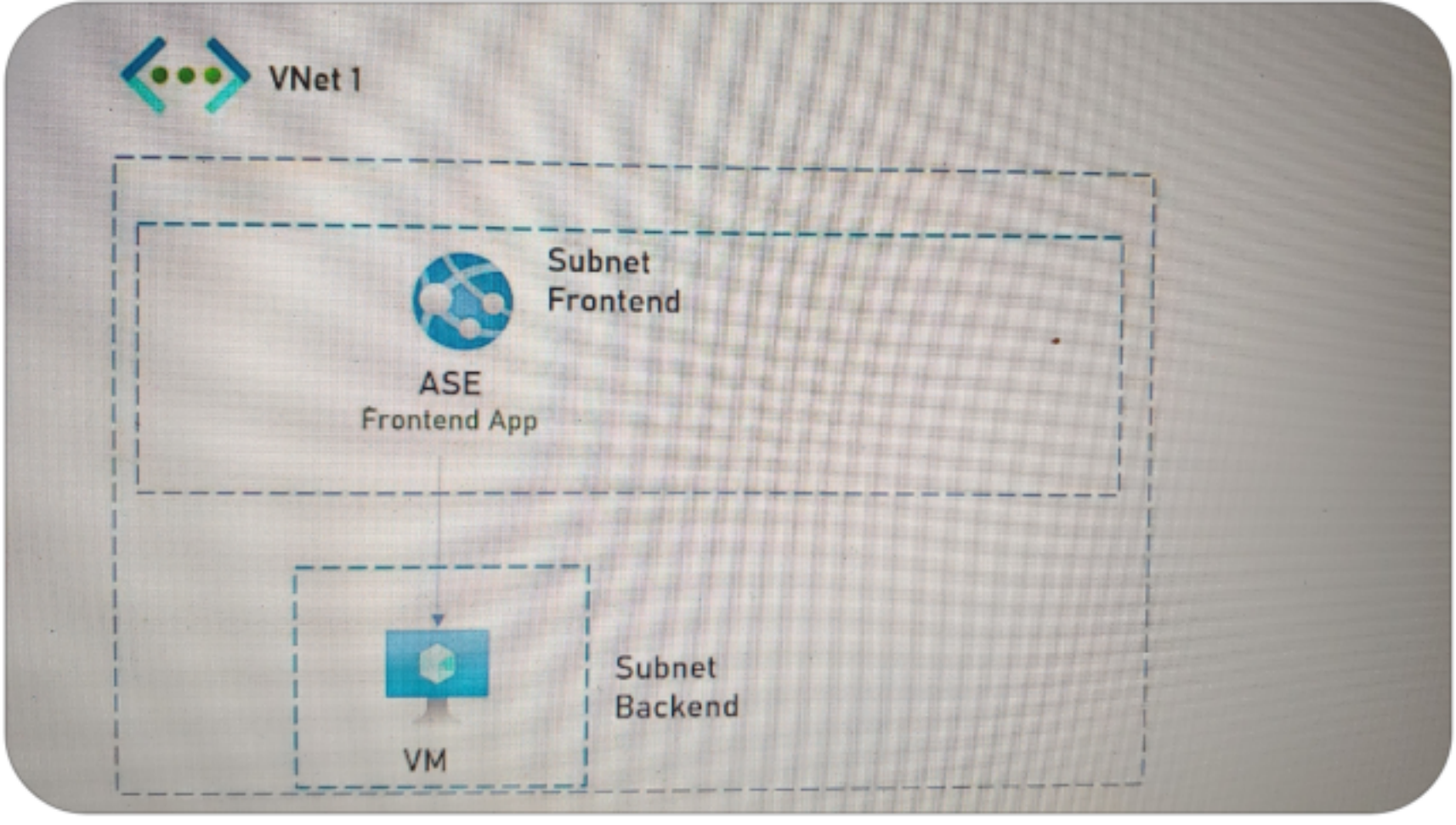


App Service Access Restrictions

- Access Restrictions are similar to NSG, but for App Services where as NSG are applied to VM/SubNet
- It restrict traffic to App Services. By default all inbound traffic are allowed on port 80 and 443 as App Services by default hosts web services. Using Access Restrictions, inbound traffic is restricted to the allowed IPs/ VNets/Service Tag.
- Service Tag are the general name for the Azure Managed services like Load Balancer, API Gateway, Azure Monitor etc.
- Main use cases: Backend App Service that should be accessed from front end App Service/VM only. App service that sits behind Application Gateway/Load Balancer and should not be accessible directly.

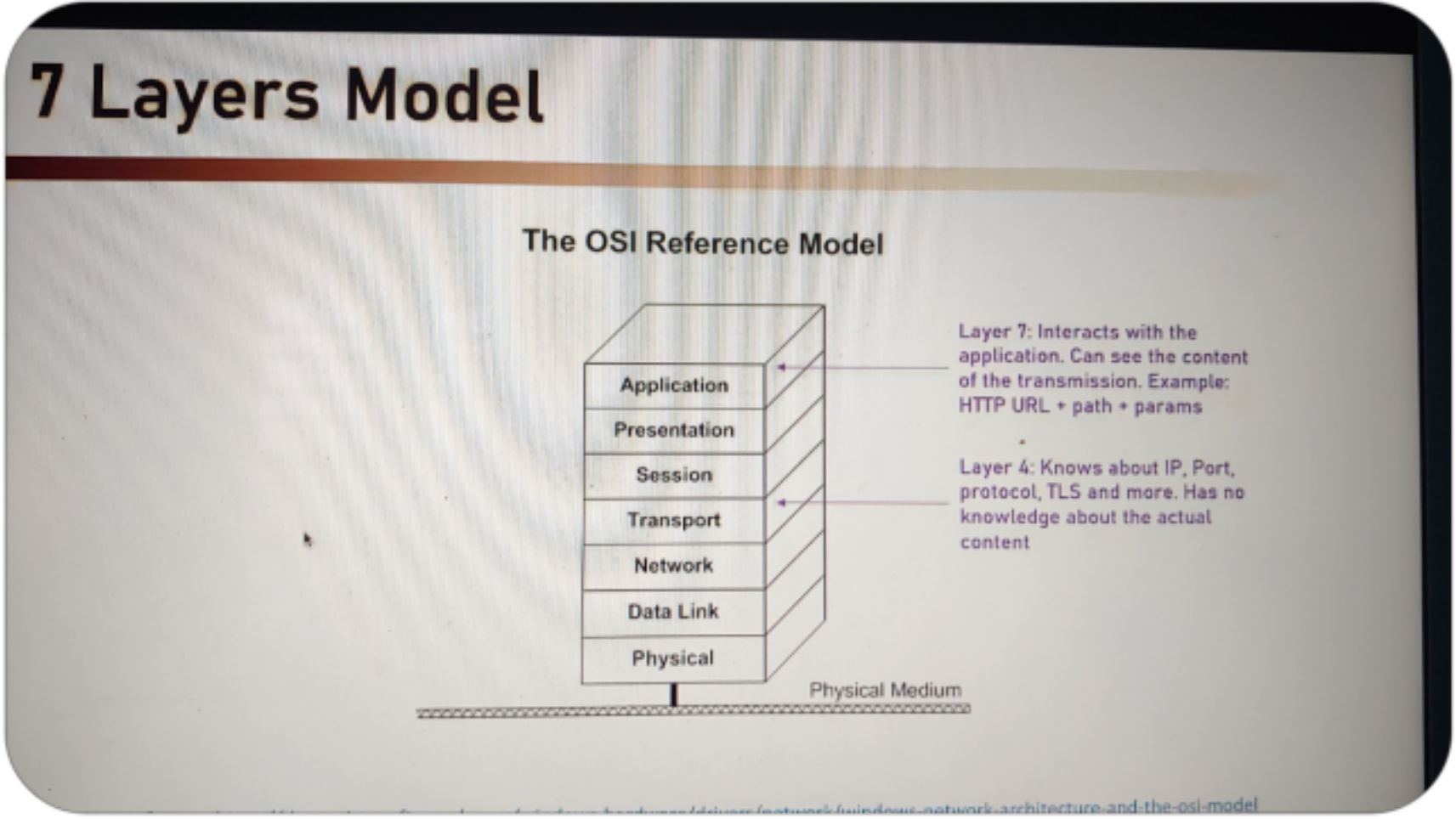
App Service Environment(ASE)

- ○ Its a special type of app service deployed directly to a dedicated VNet and dedicated hardware. App services can not be deployed inside a VNet, as it needs to be accesses from outside. But we can restrict the traffic through Access Restrictions, but still we don't have full protection of a Virtual Network. ASE is expensive
- ○ Major use cases: Elevated security - complete isolation. Very high scale requirements.
- ○ By using ASE, we can place App Services inside VNets similar to any other resources and take all the advantages of VNets.

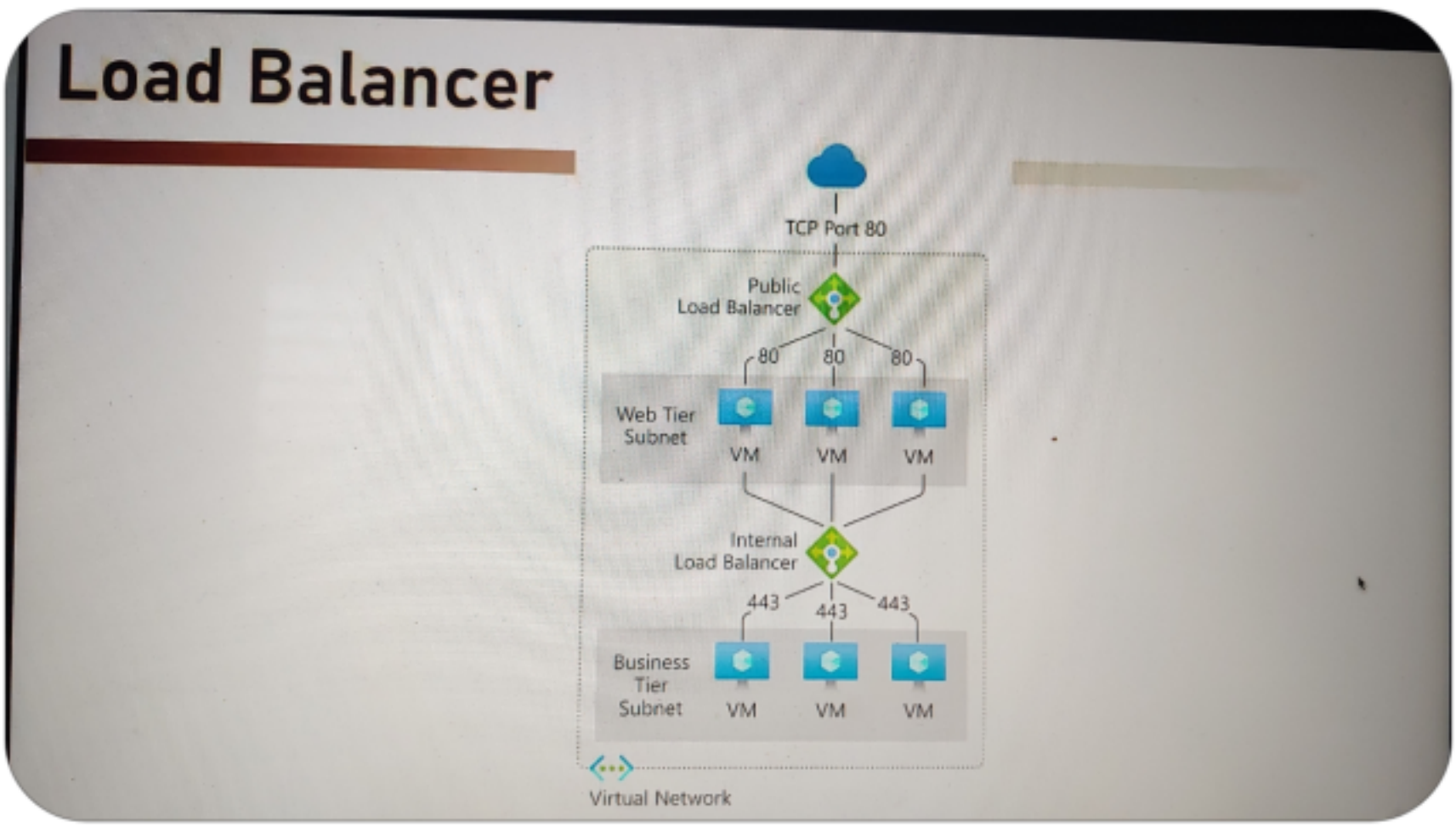


Load Balancer

- Its an Azure service that distributes load and checks health of underlying VMs. So when a VM is not healthy - no traffic is directed to it. It can work with VMs or Scale Set. It can also be public or private.
- Load balancer operators at layer 4 of the OSI model.

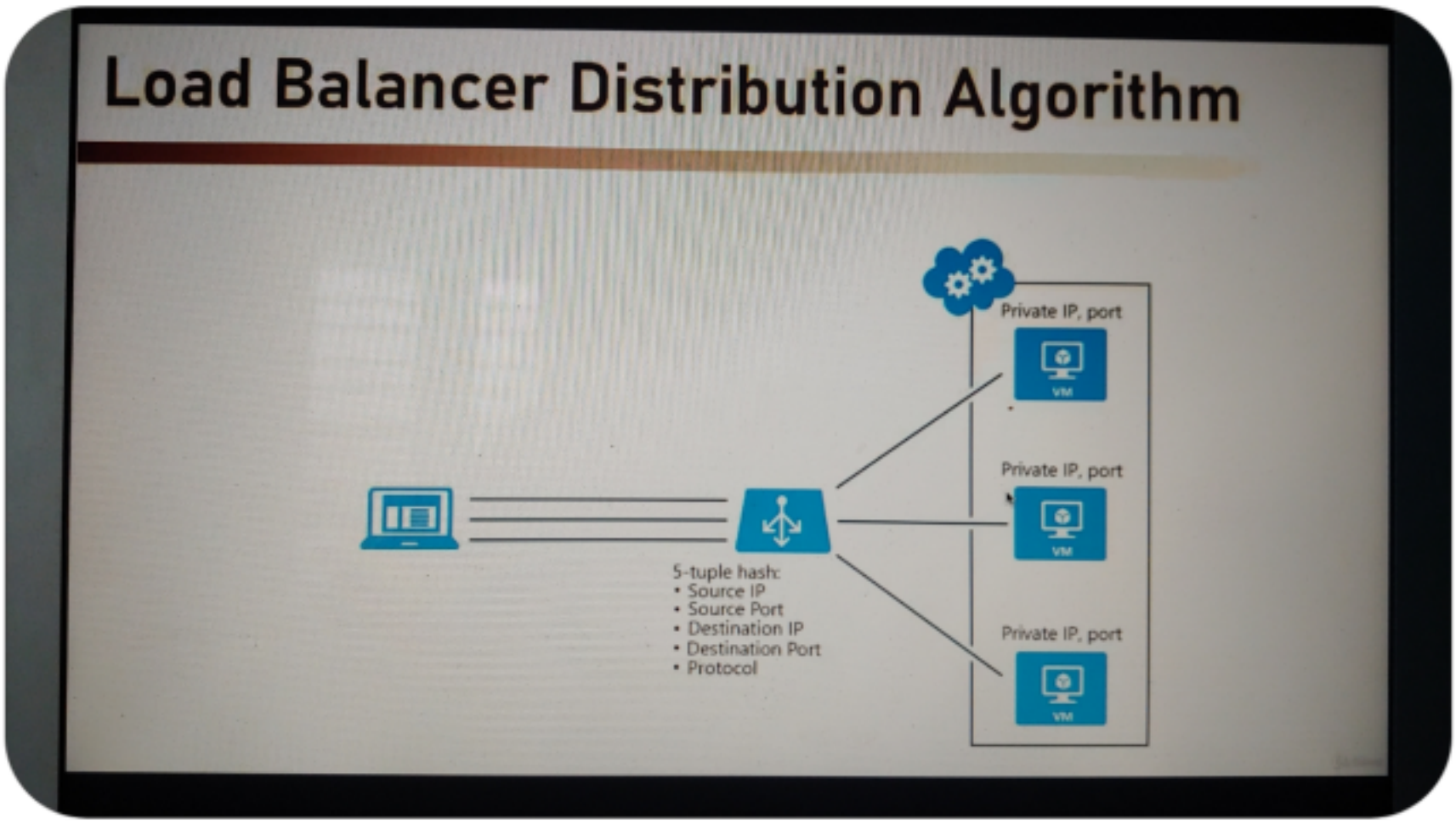


- Typical deployment of Load Balancer looks like this:



- **Load Balancer Distribution Algorithm:** It based on 5 tuples which are Source IP, Source Port, Destination IP, Destination Port

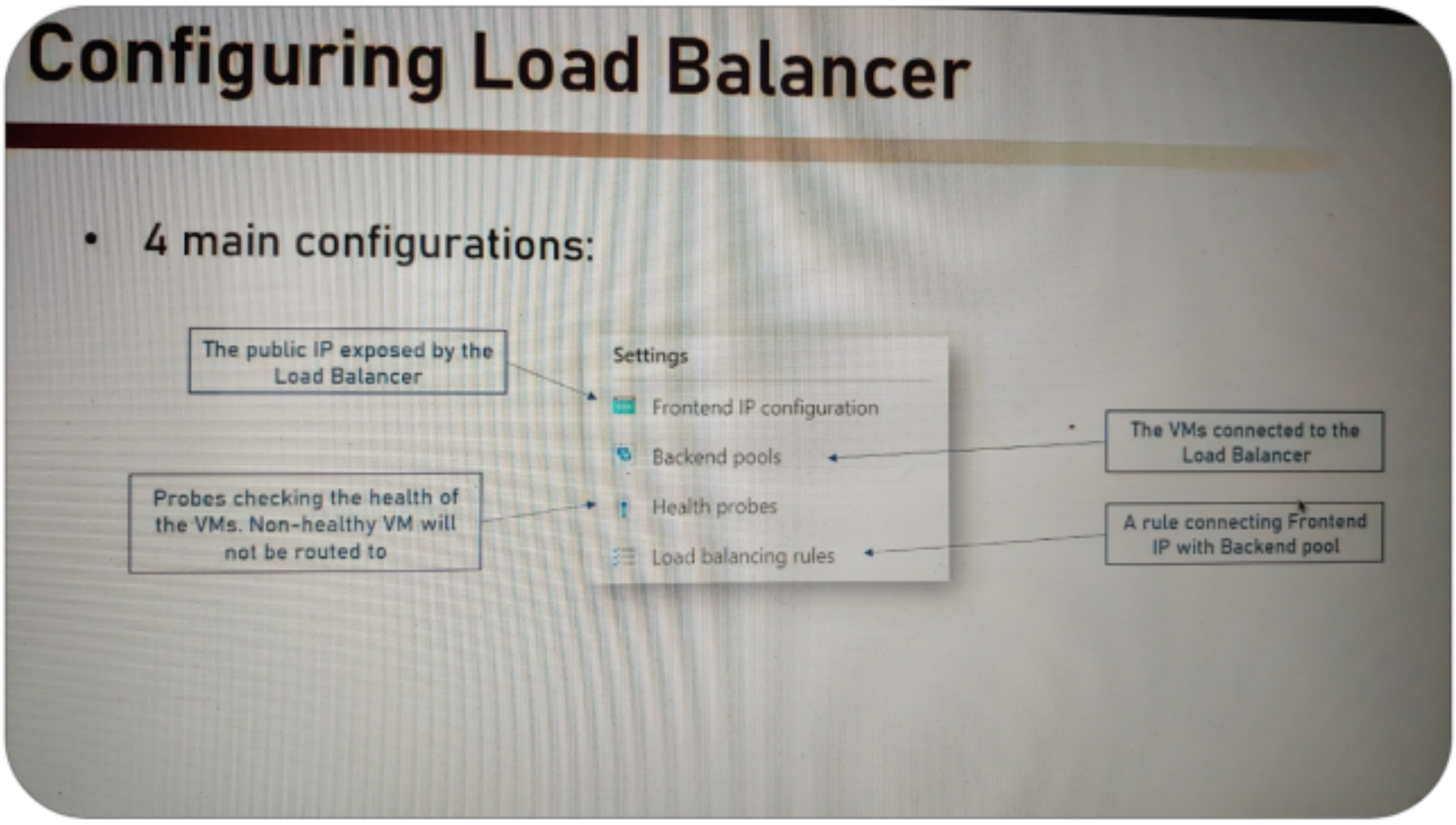
and Protocol Type. These tuples are same as the one for the NSG. These tuples are used to determine how to route the incoming traffic to destination.

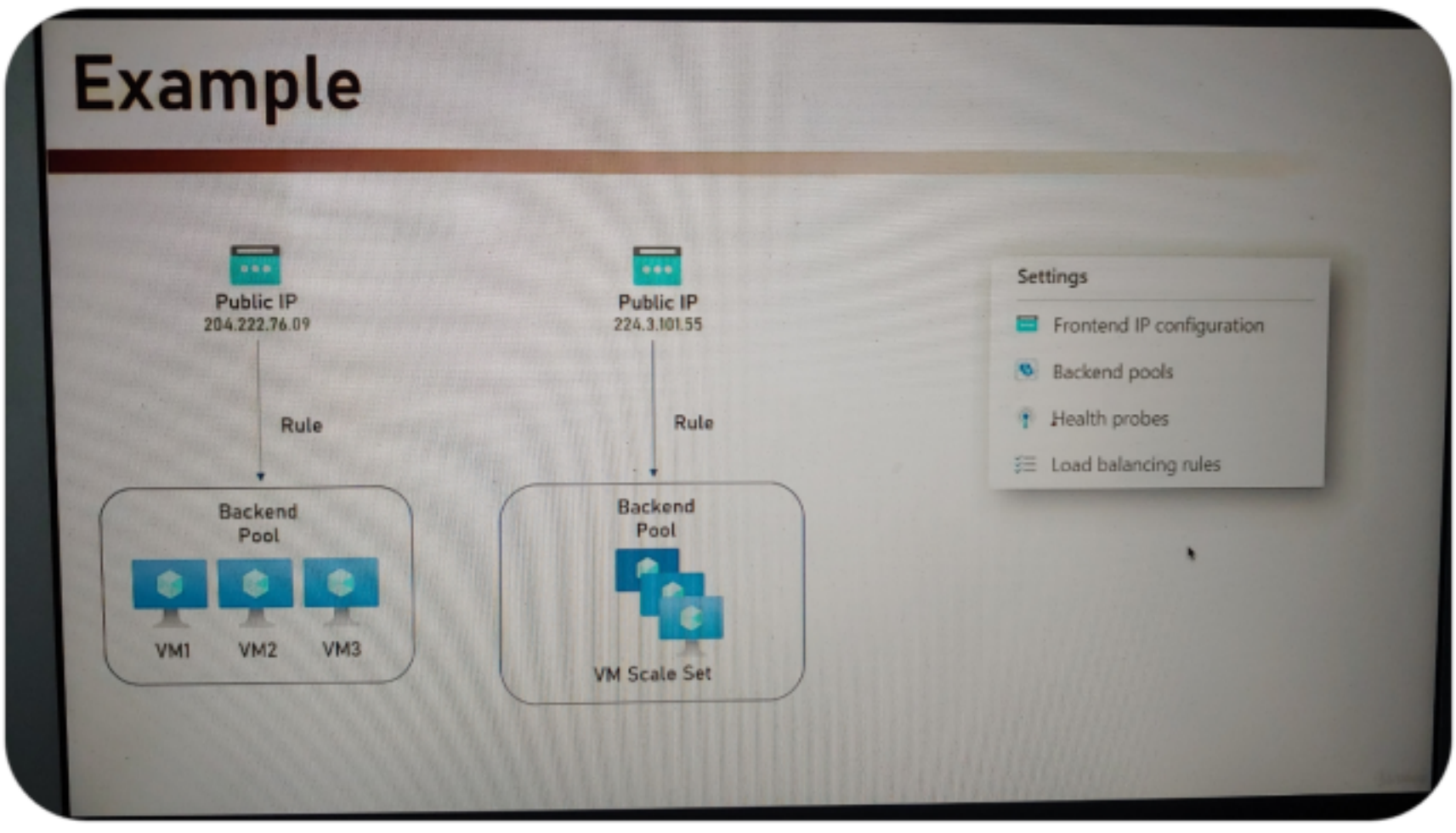


Load Balancer Types:

Basic	Standard
No redundancy	Redundant
Open by default	Secure by default
Up to 300 instances	Up to 1000 instances
No SLA	99.99% SLA
Free	Not Free

Configuring Load Balancer:

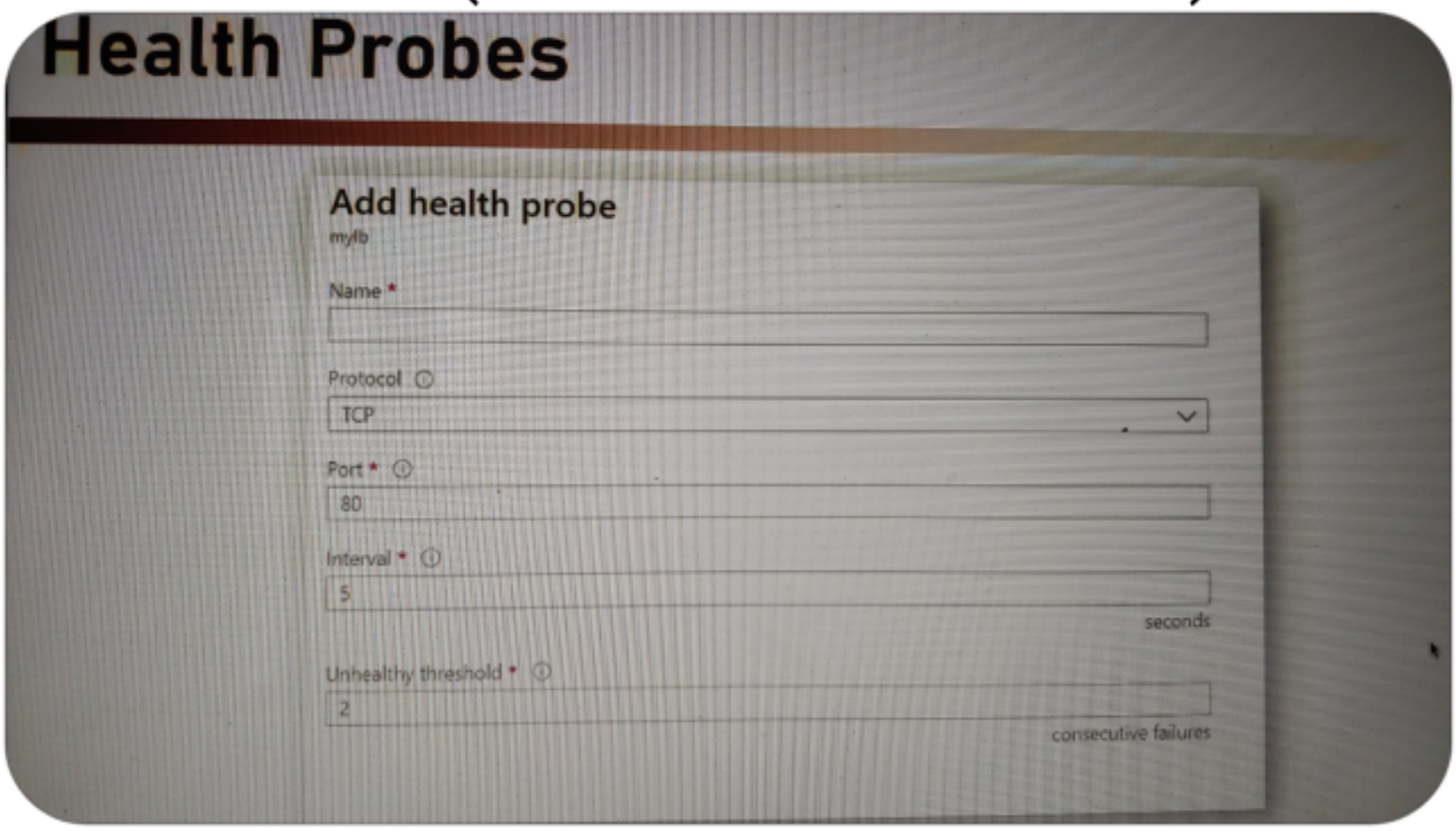




- **Health Probe:** It checks the health of the VM. A non-healthy VM will be marked as Down and will not be routed to. It run in

intervals usually a few seconds. It can run on TCP, HTTP, HTTPS etc

- Health probes run on the VM's host. No network traffic outside the host. Health probes are allowed by default in NSG
- Configurable unhealthy threshold - how many times a check should fail for the VM to be marked as Down (default is 2 in a row)



- **When to use Load Balancer:** Great for internal resources. Do not use for external resources, especially on Web Apps or Web API, since it can't handle http. It

does not route based on the path. Also, it has no protection. For this we have application gateway, which is basically a load balancer with improved http and web capabilities.

Application Gateway

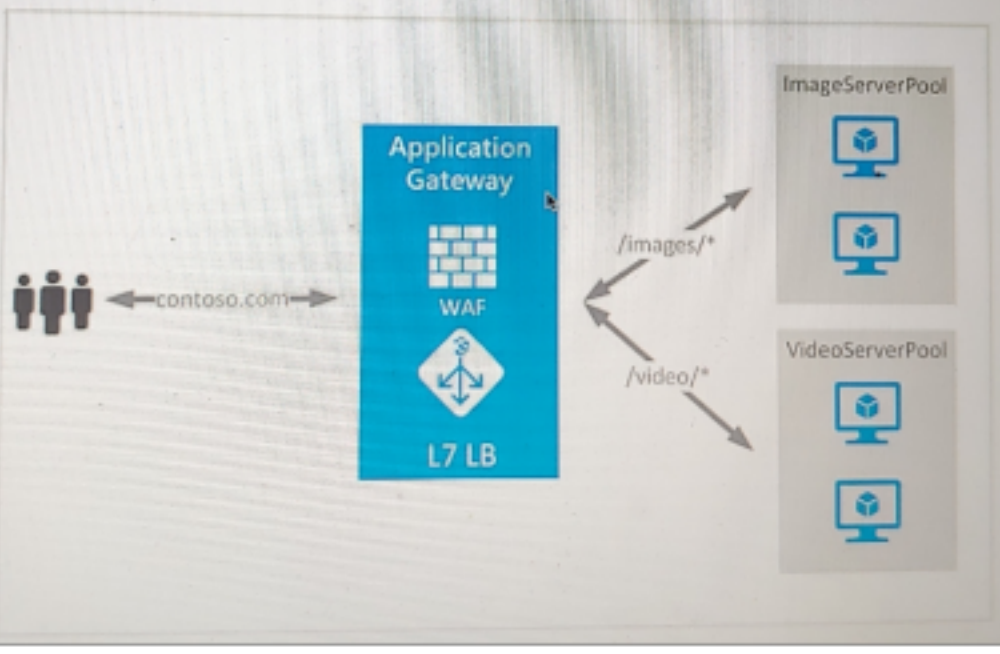
- ○ Its a load balancer with improved http and web capabilities. It can function ad the external endpoint if the web app. It works with VMs, VM Scale Sets, App Services and K8s.
- ○ It is similar to the Load Balancer with additional features like:

- Similar to the Load Balancer...
- With additional features:
  - SSL Termination
  - Autoscaling
  - Zone redundancy
  - Session affinity
  - URL based routing
  - WebSocket and HTTP/2 support
  - Custom error pages
  - Header & URL rewrite
  - WAF
  - And more...

≡ ○ In contrast to the Load Balancer, the application gateway operates at layer 7 of the OSI model.

Application Gateway

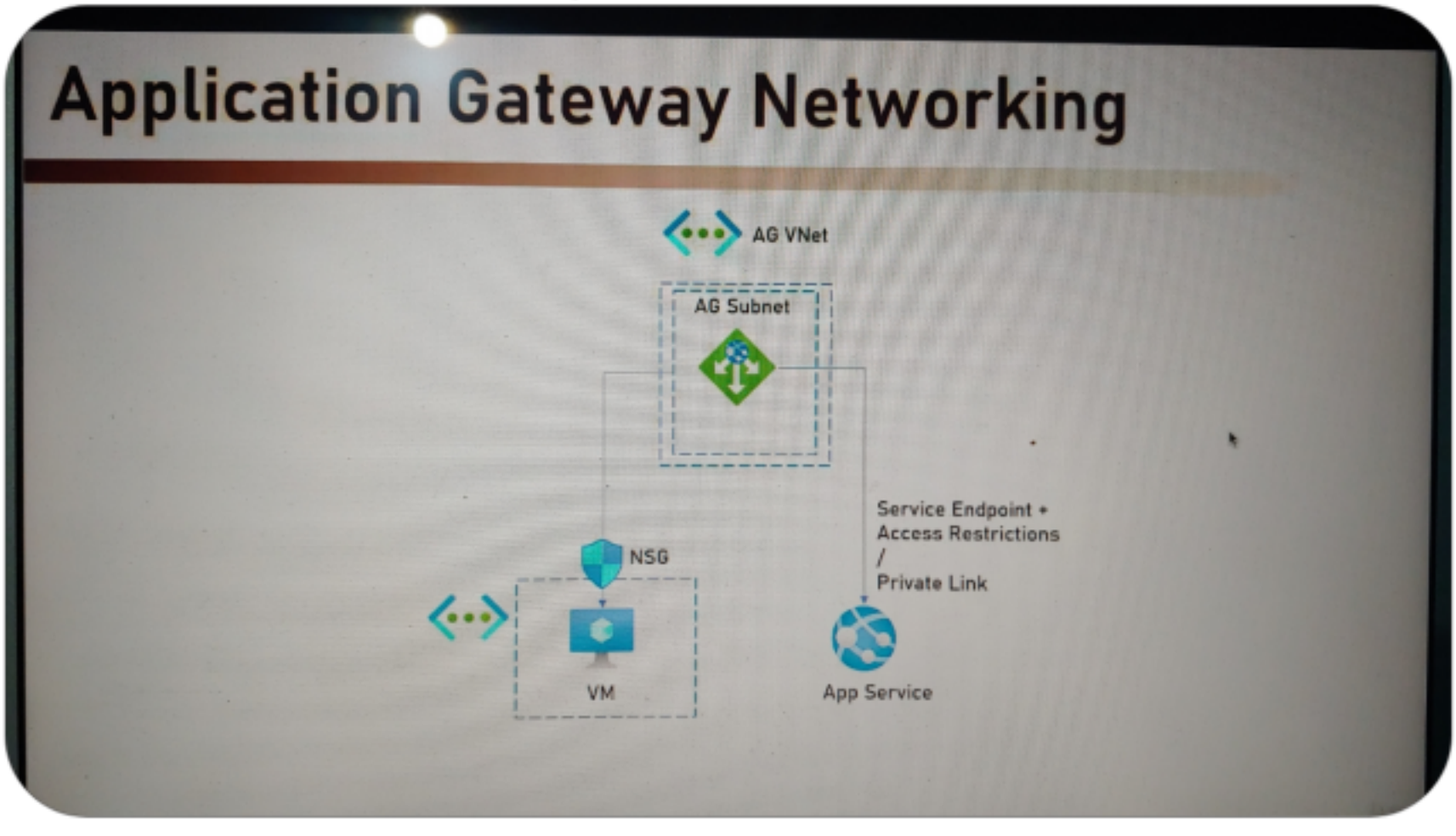
- Operates at layer 7 of the OSI model



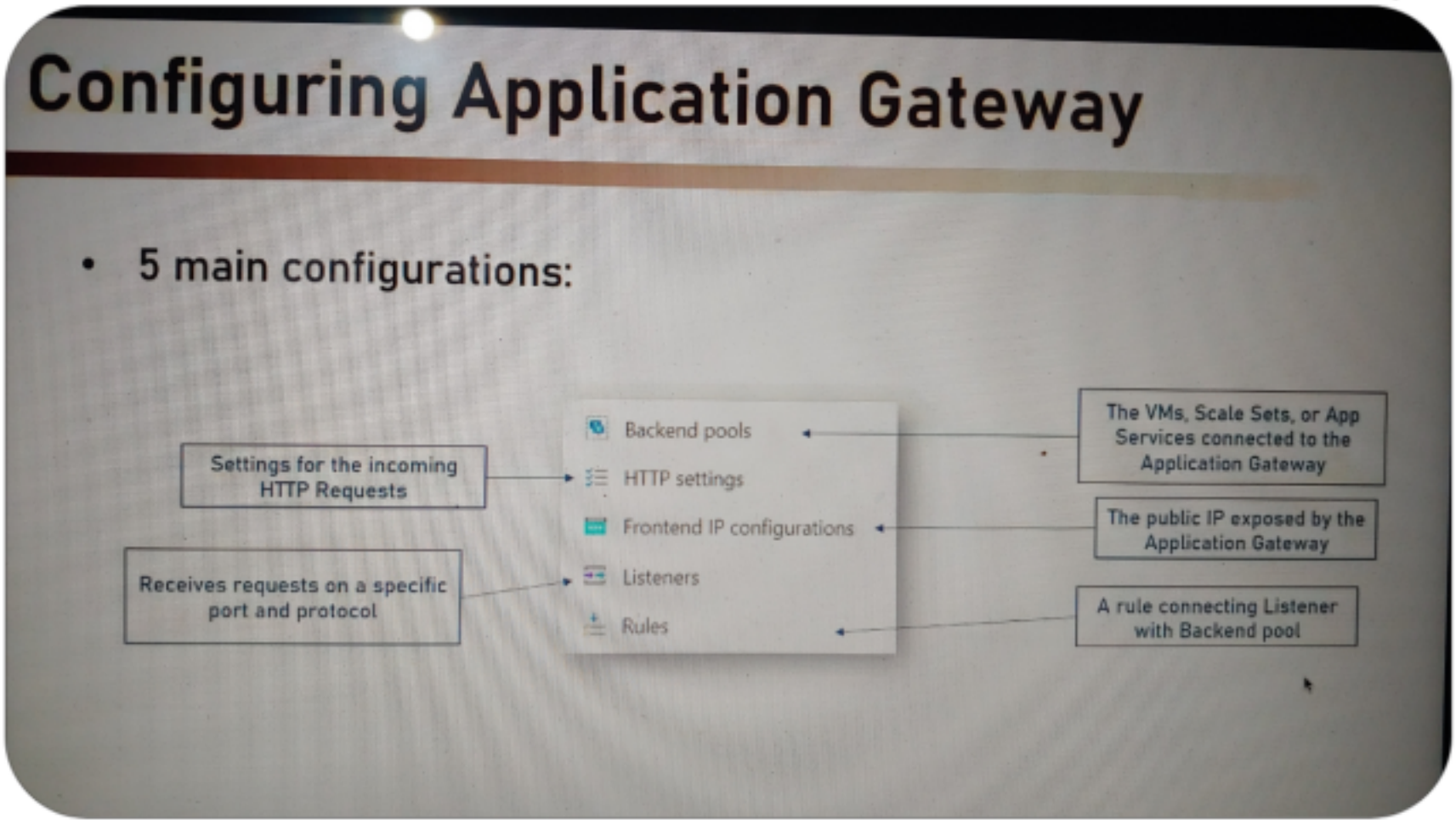
≡ ○ **Web Application Firewall: WAF** protects web apps against the common attacks such as

Cross-site scripting, SQL injection etc. It works in detection or prevention mode.

- **Application Gateway Networking:** Application Gateway is placed in its own Subnet often in its own VNet. For this we must make sure backend resources are accessible from the Application Gateway subnet and not accessible from anywhere else. Because if we bypass the AG and go straight to the VM or App Services, then there is no point of having AG.
- **AG Deployment:** It look like this:



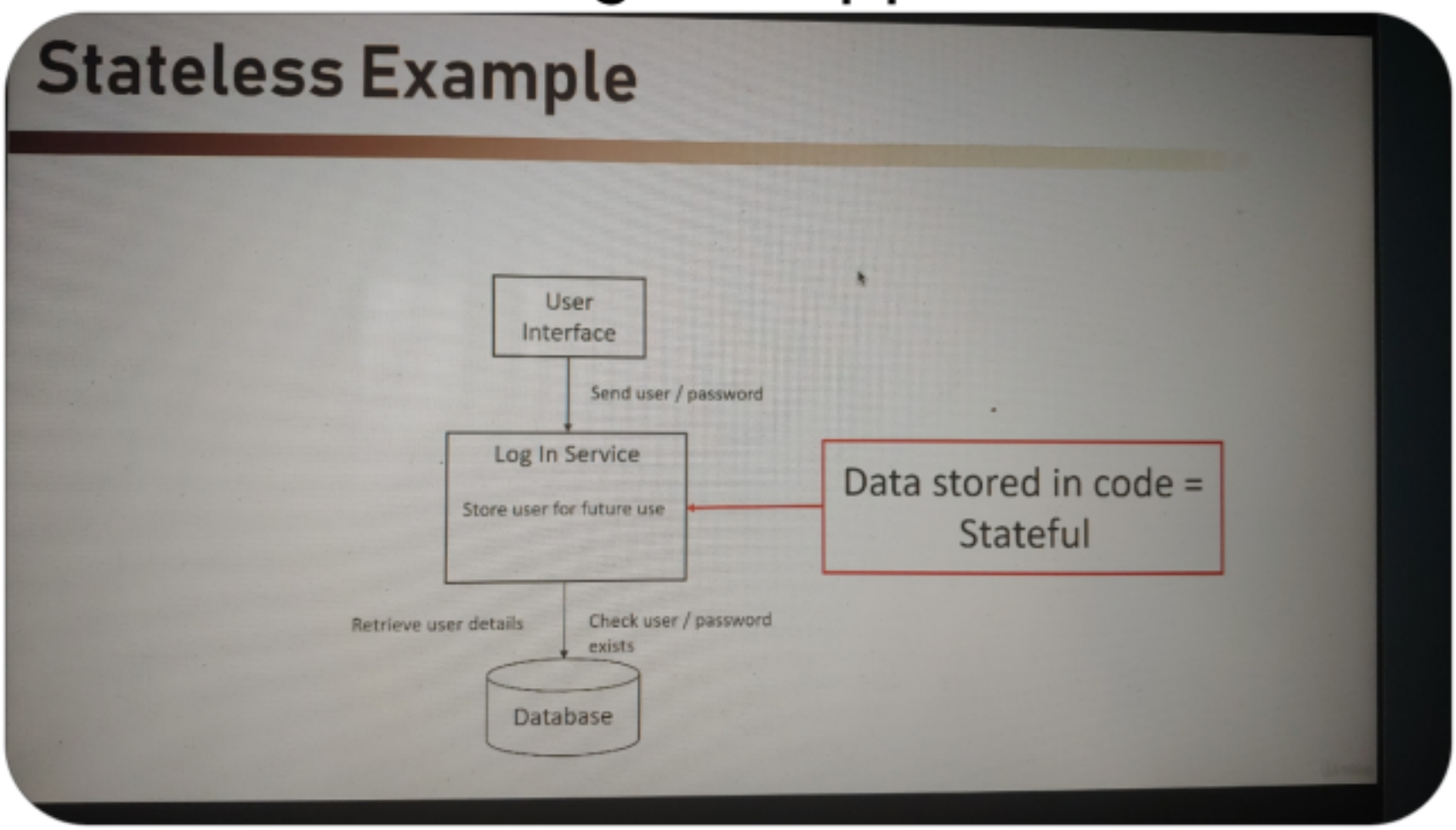
- **Configuring Application Gateway:**



- **Affinity:** It makes sure user will always be directed to the same instance(VM/App Service) it begin with. So if user is routed to VM1 when they made a first request, then if Affinity is enabled the subsequent request will also be routed to the same VM1 even though load in VM1 is high already. That's why affinity feature should be avoided when possible. It usually required in Stateful app. It usually sign of bad design. Always try to design stateless

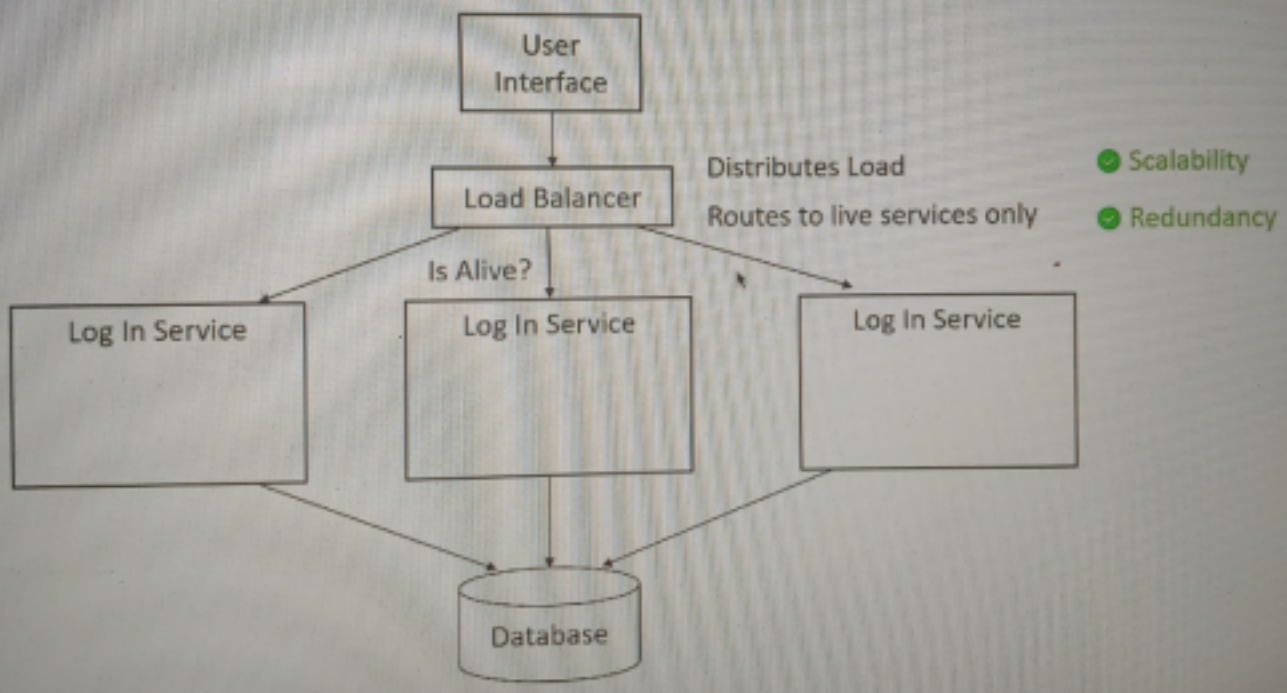
app which can avoid affinity.

- **Stateless Architecture:** In stateless architecture the applications state is stored in only two places - the data store and the user interface. No state is stored in applications code. State are nothing but applications data.

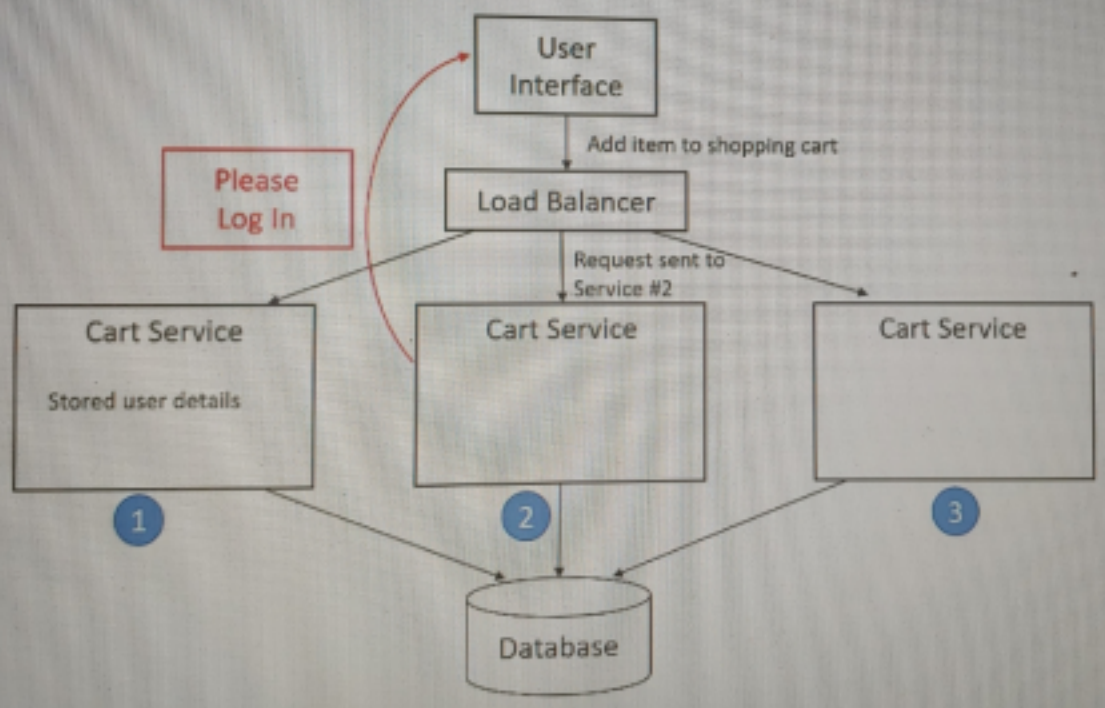


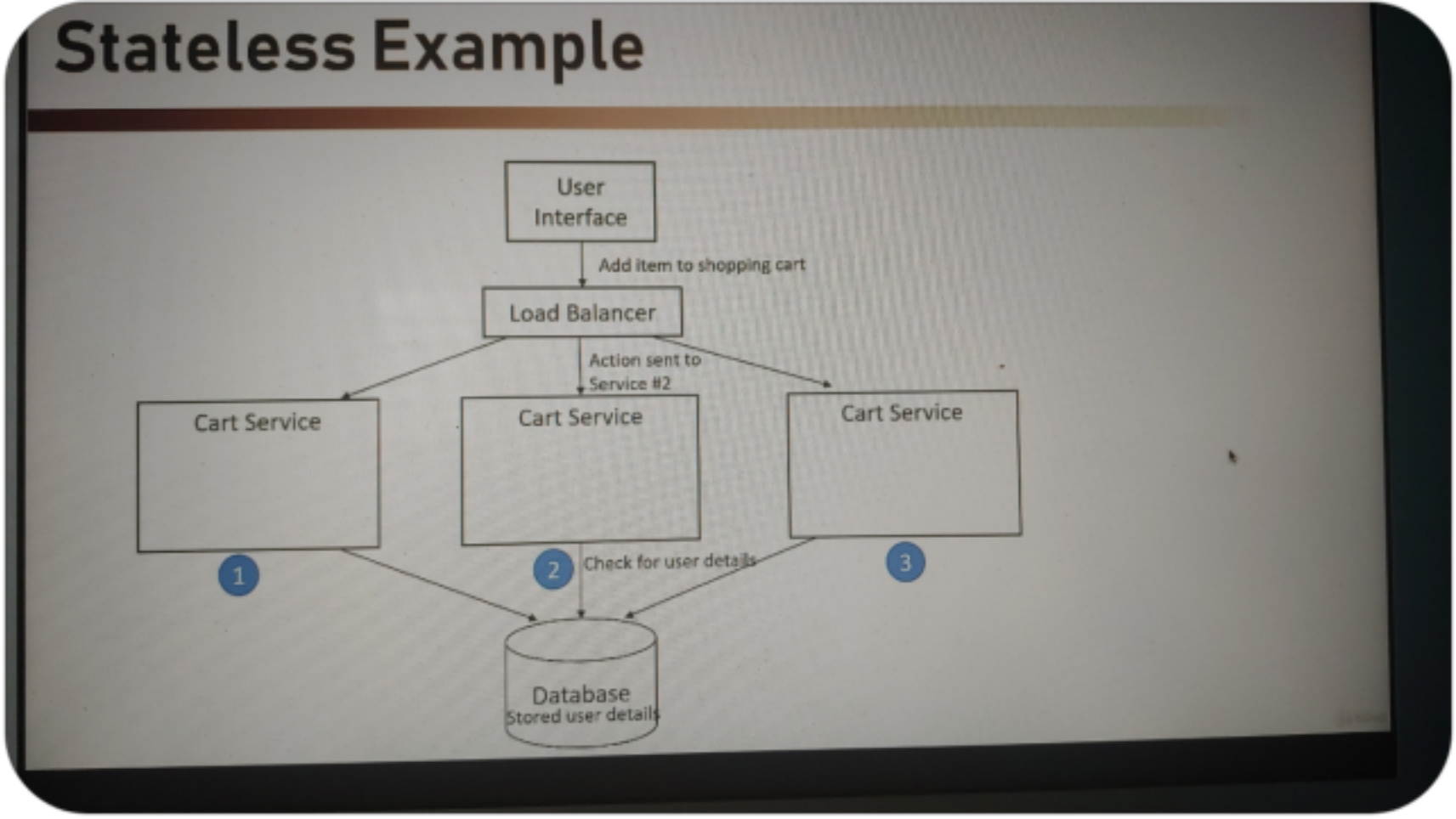
- **Scalable & Redundant Architecture**: Scalability helps in scale up or scale down of the system whenever needed and Redundancy allows the system to function properly when resource is not working.

Scalable & Redundant Architecture

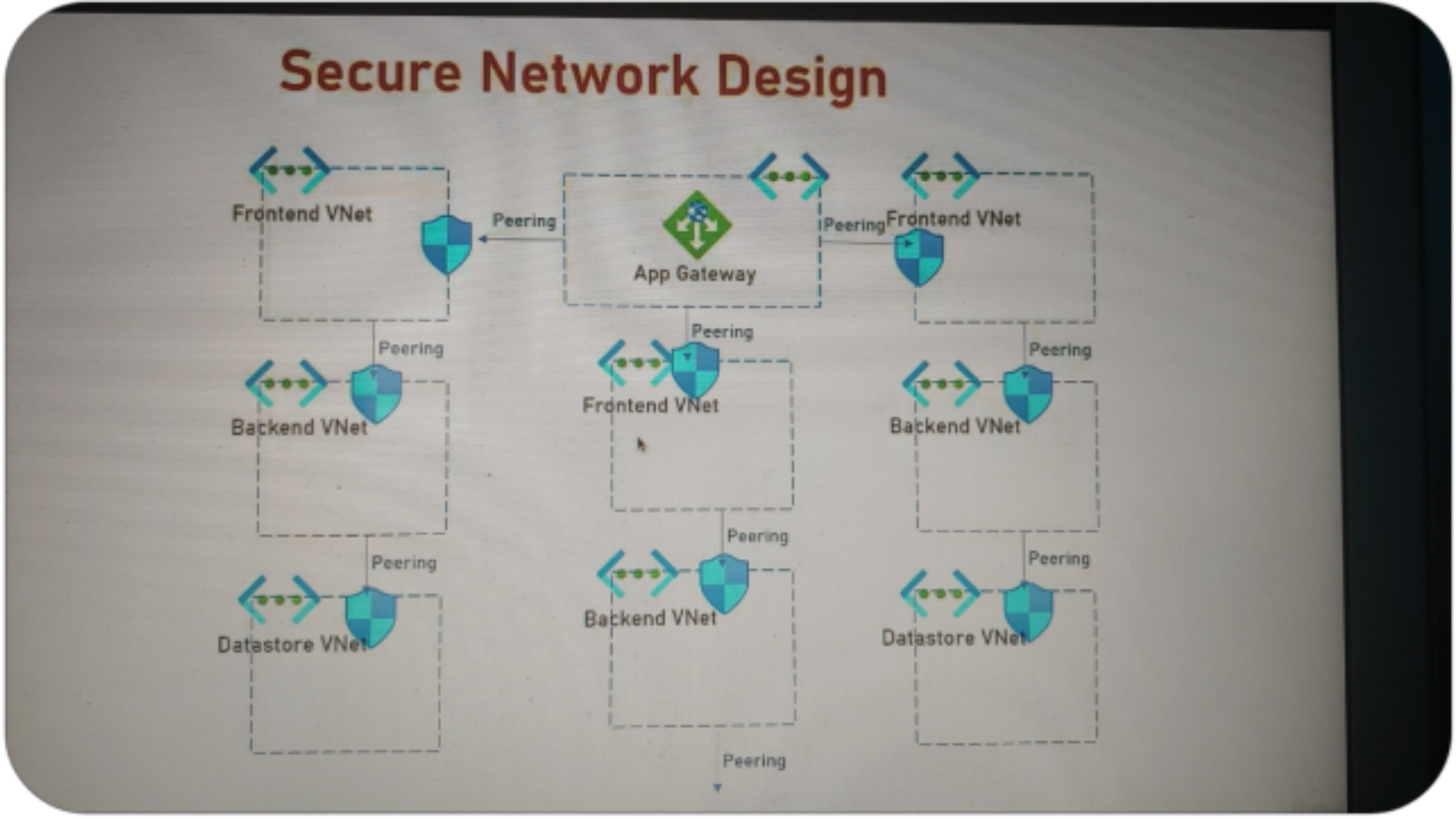


Stateful Example





- **Secure Network Design:** The following diagram shows secure network design where AG is placed in its own VNet and other Azure resources or services are placed in their own VNets. Finally there will be Network peering along with NSG between multiple VNets. This network design pattern is called **Hub and Spoke**.



Application Gateway In Action

- Create a new application gateway through portal by placing it within a newly created VNet and Subnet. Create AG with below configurations:
- Frontends: create a new Frontend with static Public IP Address which will be exposed to users.
- Backends: Backends are the resources to which AG will route the traffic. Backends can contains VM, VMSS and App Services. Add multiple backend

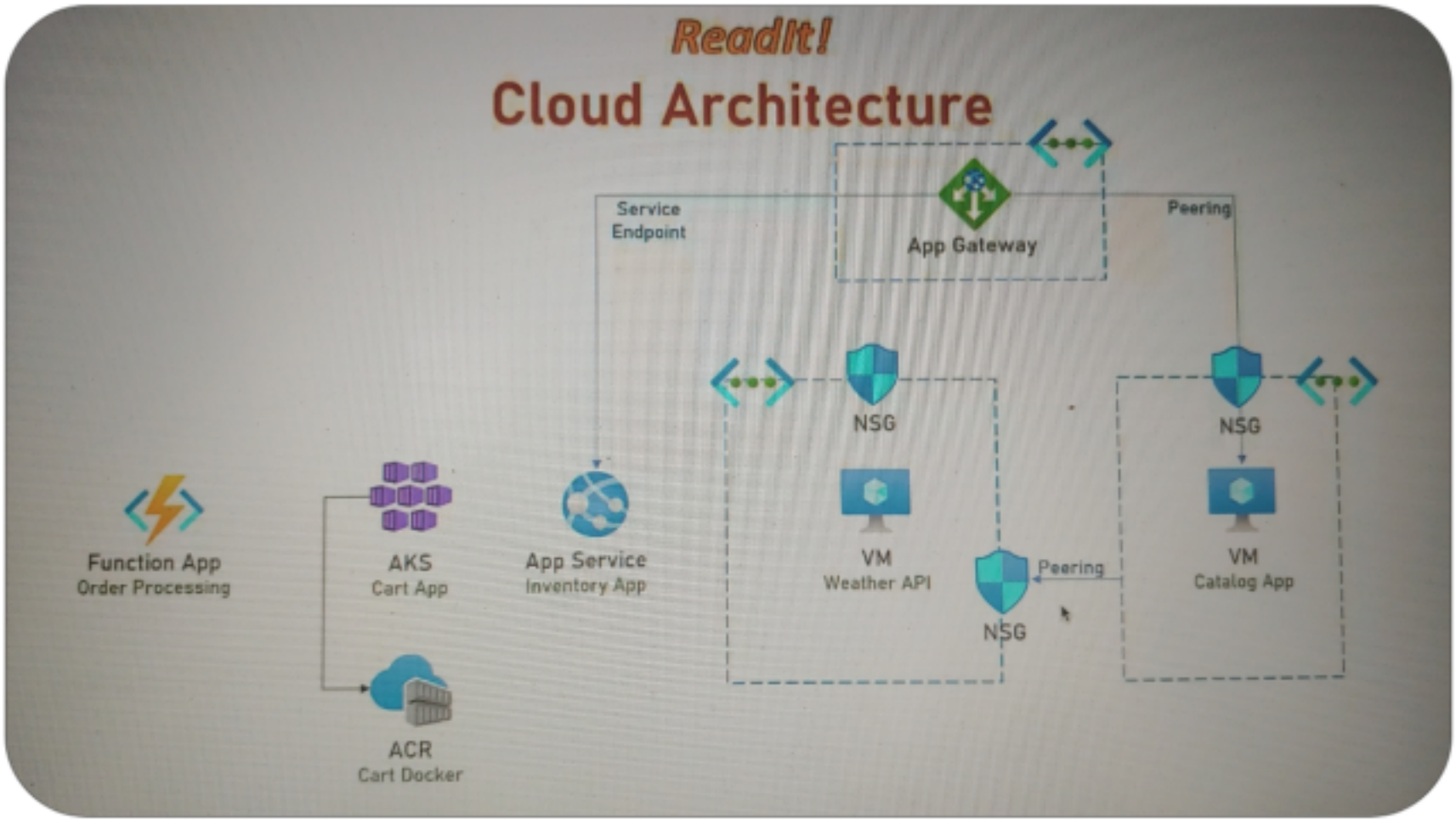
pool for each of the applications we have.

- ≡ ○ Configuration: Here we define the routing rule, which tells how a front end IP connect to a backend pools. Define routing rule for each of the defined backend pool along with incoming and targeted port numbers.
- ≡ ○ Finally access each of the application through the public IP address of the API gateway and incoming port of the application.
- ≡ ○ Note: Define a service endpoints between the subnet where the Application Gateway is defined and the App service so that those App service will receive the traffic only from this subnet. Also, configure the Network Peering between the VNet where API Gateway is running and with other VNet where application is

running, so that those applications can be accessed only through the API Gateway.

- **Application Gateway and AKS:** AG had No built in integration with AKS.
- AKS has kind of gateway services
- There is Application Gateway Ingress Controller(AGIC) that does this. But it is in preview mode and not recommended
- Better to use 3rd party products.
- **Application Gateway and Functions:** Functions Apps basically App Services. So they can be protected by Application Gateway the same way App Services are.
- We can configure them in Backend pool and the configure their Access Restrictions.

Current Architecture



- New element App Gateway is added which connects to App Service using Service Endpoint and connects to Catalog App/VM using a Peering, because they reside in different VNet. Note also the use of the NSG in front of the Catalog App which is used to block any direct traffic to this VM/App so that only traffic from App Gateway VNet is allowed to App

==========

Data in Azure

- Azure provides many data solutions as cloud services such as relational databases, NoSQL databases, object stores etc
- These databases are fully managed services and these can be part of Azure App or completely independent. These services are better than unmanaged data services.
- **Major database features**: what to look for when selecting a database? Security - we mean mainly network isolation and encryption. Backup - we want to look up for Backup types and retention Period. Availability - we look for SLA, Replication and DR.
- **Database on VM**: We can also setup Azure VM with required database software. In fact there

are ready made VMs in the marketplace with pre-installed database.

- Pros of Database on VM: Full flexibility, Full control.
- Cons of Database on VM: You have to take care of everything like SLA, updates, Availability, Back and Restore, Security etc.

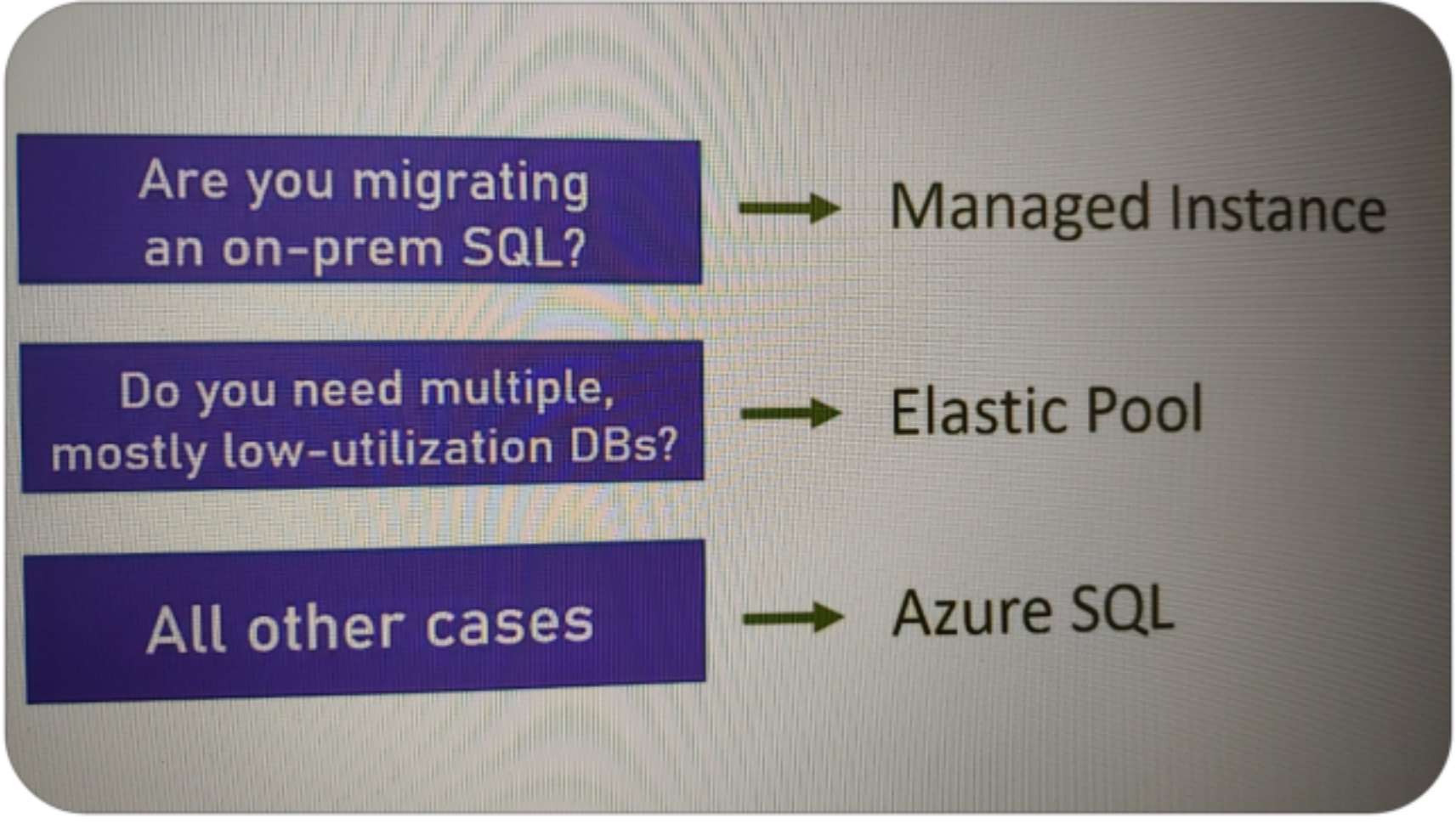
Azure SQL

- Its a managed SQL Server on Azure. It provides built in security, backup, availability etc
- There are 3 Azure SQL flavor which are Azure SQL database, Elastic Pool & Managed Instance.
- **Azure SQL Database**: Its a managed SQL server on azure which is a single database on a single server. It provides security

like IP firewall rules, Service Endpoint, secure communication between your application and SQL.

- Backup: It provides variety of backup like Full(once in a week), Differential (every 12-24 hrs), Transaction Log(every 5-10 min)
- Retention Period: It provides variety of retention period like Regular Backup (7-35 days), long term backup (up to 10 years).
- Availability: Backup is stored in a geo-redundant storage. It can offer Active geo-replication which provides fully active-active deployment of the database so if a region is shutdown, the second region has up to date database which we can use immediately. It provides SLA of 99.9 - 99.995%. Its the relational database with highest SLA in Azure.

- Compute tiers: Azure provides 2 option to choose SQL. One is Provisioned(Pay for allocated resources regardless of actual use) and another one is Serverless(Pay for actual use - vCore + RAM / second).
- **Elastic Pool:** It is based on Azure SQL, but the main difference is, it allows storing multiple databases on single server. Its very cost effective and we need to purchase the compute resources which we need, not the database.
- **Managed Instance**: It is closer to the on-premise SQL Server. We can deploy these to VNet, which cant be done with the other SQL server flavour. It does not provide active geo-replication. No auto scaling & tuning support by default, we have to take care of it.
- Which Azure SQL to Choose?



Azure SQL in Action

- ○ Create a new SQL database in Azure Portal by providing a valid name for this database and creating a new SQL Server with valid username and password. Select the elastic pool option no.
- ○ Add a new extension 'SQL Server' in VS Code. This extension provides 2 option. 1 is to create various connection and other one is to execute queries. Copy the connection strings depending on the language you are using and create a new connection in VS

Code with it.

- By default Azure SQL does not allow any user to connect to it. If we want to access it from specific IP then we need to configure it in Azure SQL firewall setting's add client IP option.
- Next we can connect multiple applications which are running in a VM to this Azure SQL. In order to protect this communication between the VM and the Azure SQL, we can create a new service endpoint in the VM for the sql.

SQL vs NoSQL

SQL

SQL Database

- Stores data in tables
- Tables have concrete set of columns

Column Name	Type	Nullable?
OrderId	Numeric	No
OrderDate	DateTime	No
CustomerId	Numeric	No
DeliveryAddress	String	No

SQL Database - Relationships

Column Name	Type	Nullable?
OrderId	Numeric	No
OrderDate	DateTime	No
CustomerId	Numeric	No
DeliveryAddress	String	No

Column Name	Type	Nullable?
OrderItemId	Numeric	No
OrderId	Numeric	No
ItemName	String	No
Quantity	Numeric	No

SQL Database - Transactions

- Atomic set of actions
- ACID:
  - Atomicity
  - Consistency
  - Isolation
  - Durability

SQL Database - Querying

- Using SQL
- Structured Query Language
- Very mature

```
Select OrderID, OrderDate, CustomerId, DeliveryAddress
From Orders
Where OrderDate >'01/01/2018'
```

NoSQL

- One of the main issue of SQL databases are performance and size. Since relational databases maintains schema, for each

record and enforced transaction, the performance will degrade as the database is getting larger and larger. This is the problem which NoSQL trying to solve.

NoSQL

- Emphasis on scale and performance
- Schema-less
- Data usually stored in JSON format

NoSQL - Transactions

- Eventual Consistency
- Data can be temporarily inconsistent

NoSQL - Querying

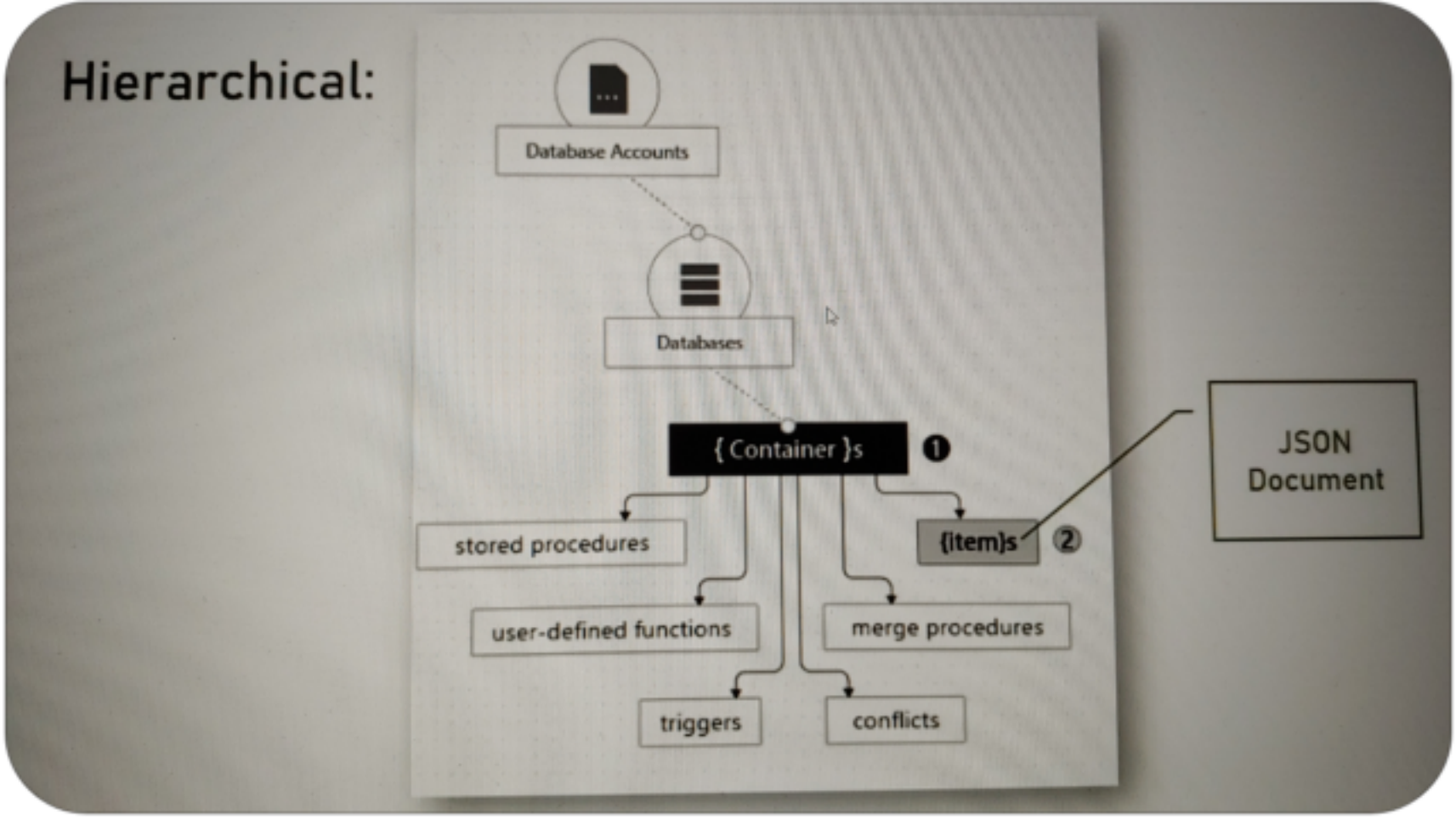
- No standard
- Can be frustrating...

Data Store - Summary

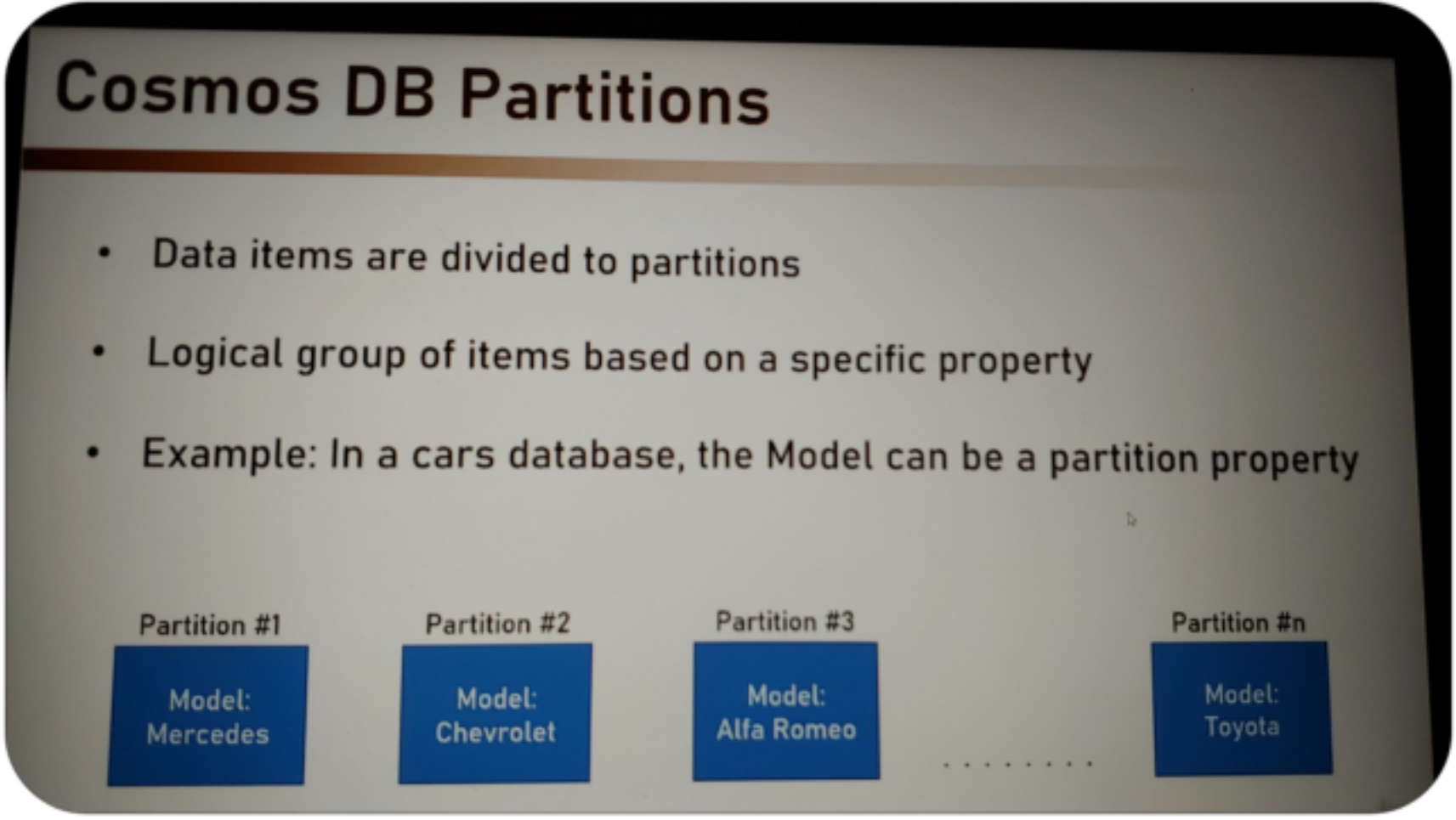
SQL Databases	NoSQL Databases
Not Huge	Huge
Structured Data	Un- or Semi- Structured

Azure Cosmos DB

- ○ Its a fully managed NoSQL database. It provides amazing high performance. Its globally distributed means we can replicate the data in Cosmos DB across multiple regions.
- ○ Fully automatic management - updates, scaling, fixes etc.
- ○ It also provides multiple APIs like SQL, Mongo, Azure Table, Cassandra etc. So while creating your Cosmos DB in Azure, we can choose how we wants to use this database and accordingly we can select either SQL or Mongo APIs.
- ○ Its a hierarchical database.



- **Availability**: Can be distributed across many regions. API automatically picks the closest one region. It provides very high SLA of 99.999%.
- **Backup:** Provides full backup every 1-24 hours(default is 4). Retention period is 20-30 days.
- **Security:** Can configure IP firewall rules, Service Endpoints, Private Endpoint, Secure communication, Azure AD authentication.
- **Cosmos DB Partitions**:



- Partitions are the basic scale unit in Cosmos DB, that means we adds partitions and removes. We need to make sure while creating the databases items are divided as evenly as possible. So its extremely important to select the right partition property. Once we set a partition property we can not modify it later.
- **Cosmos DB Consistency Level:**

- Traditionally:
  - Relational DB – Strong consistency: Call returns only after successful commit in all replicas (High availability)
  - NoSQL DB – Eventual consistency: Call returns immediately, commit in replicas happens later (Low latency)

- Cosmos DB offer five consistency levels:
  - Strong (<= As in regular relational DB)
  - Bounded Staleness
  - Session
  - Consistent Prefix
  - Eventual (<= As in regular NoSQL DB)

≡ ○ basic question with consistency is, if region X updates an item and region Y reads the item, which version will it get?

≡ ○ **Strong** - Region Y will get the last

version of the item updated in region X. It is used for critical data and high read performance. **Bounded Staleness** - Region Y will lag behind X by K versions or T time. Used for high write performance and when order is important. **Session** - In a client session it provides Strong consistency. But for other clients it provides Consistent Prefix. **Consistent Prefix** - Keeps the order of the versions but no guarantee of the lag size of the versions. Used when write operation is top priority and there are less read operation. **Eventual** - No guarantee on the order of the versions and its lag size. Used for count of Re-Tweets, Likes etc.

- Consistency level will be configured at the account level

for example as Strong consistency. And it can be relaxed on the request level where we can request for Bounded staleness level.

- ≡ ○ There are no databases in the industry which can offer these kind of various consistency levels

Cosmos DB in Action

- ≡ ○ Create a new Cosmos DB in portal. Select the Account name, API as SQL or Mongo or anything, enable geo redundancy and availability zones if needed. Provide the backup policy and retention period details.
- ≡ ○ Configure the right VNet to access this Cosmos DB through firewall settings, instead of having public access to it.
- ≡ ○ So till now we created Cosmos DB Account, underneath it create Cosmos DB database with a

name by clicking on New Database option. Now create new Container inside this database. We should provide a container name like 'orders' and a partition key like '/priority', so based on the priority field of an order partitions will be created.

- Now start adding items or data to this container 'orders' in the form of json value. Similarly we can query, update, delete any data.

Azure MySQL / PostgreSQL

- Its managed MySQL/PostgreSQL on Azure. Works like any other MySQL/PostgreSQL database using the same tools. Great compatibility with on-prem MySQL/PostgreSQL database. It provides built in security, backups, availability and more

similar to other databases.

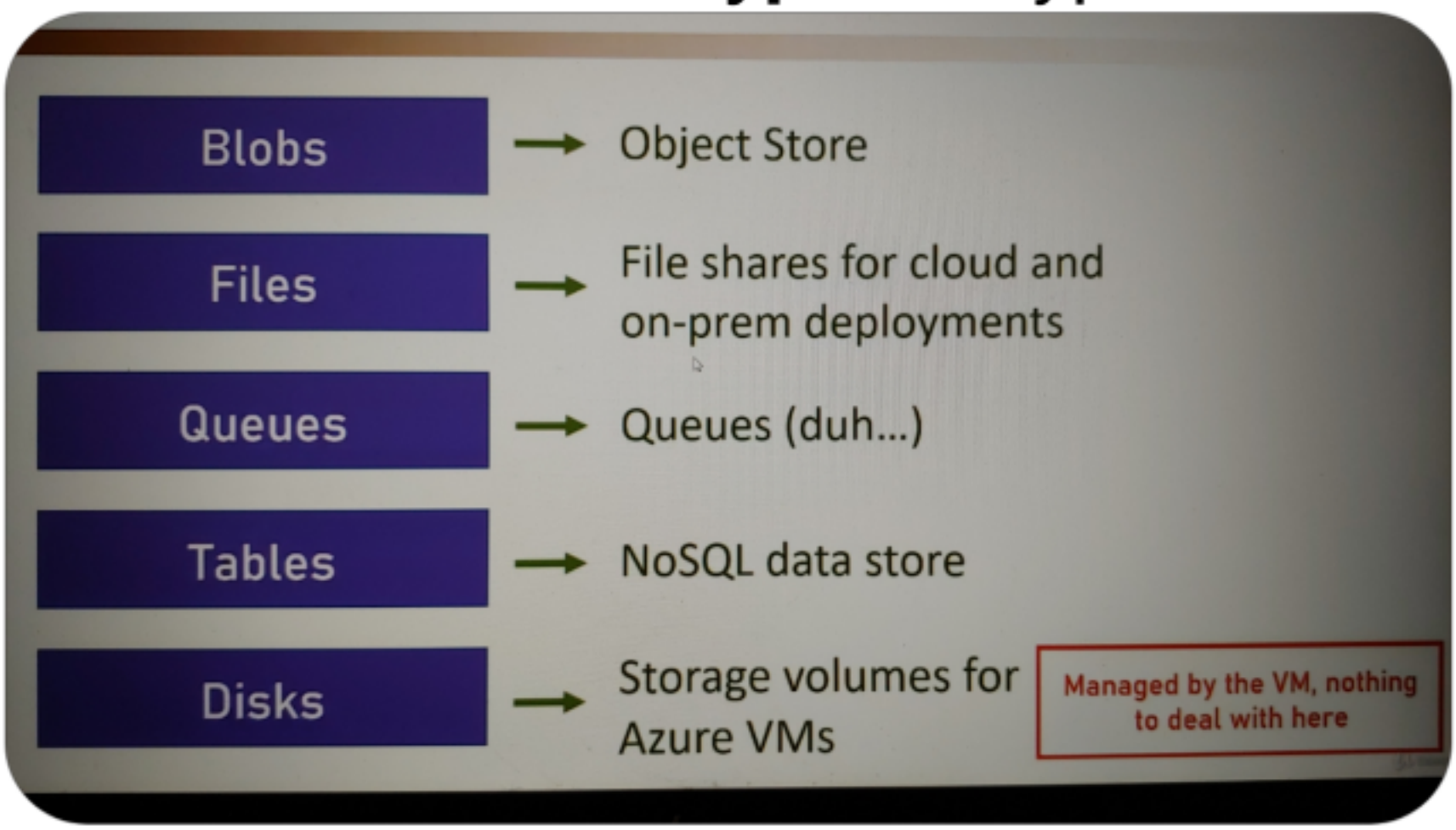
- We can create these databases through Azure Portal and connect to it from any applications.

Azure Storage

- Its a specialized database which can store Objects such as files, documents, images, videos etc. We will not use this object store to store relational data(sql) or json data(NoSql).
- Azure Storage is massively scalable. It can also accessible via http or https. To help you access Azure Storage, there are many client libraries for almost every language. Its highly durable and available.
- **Azure Storage Accounts**: An Azure Storage Account contains all of your Azure Storage data

objects. Basically Storage Accounts will have one or more containers, where we can save any Azure Store Data Objects.

- **Azure Store Types:** 5 types

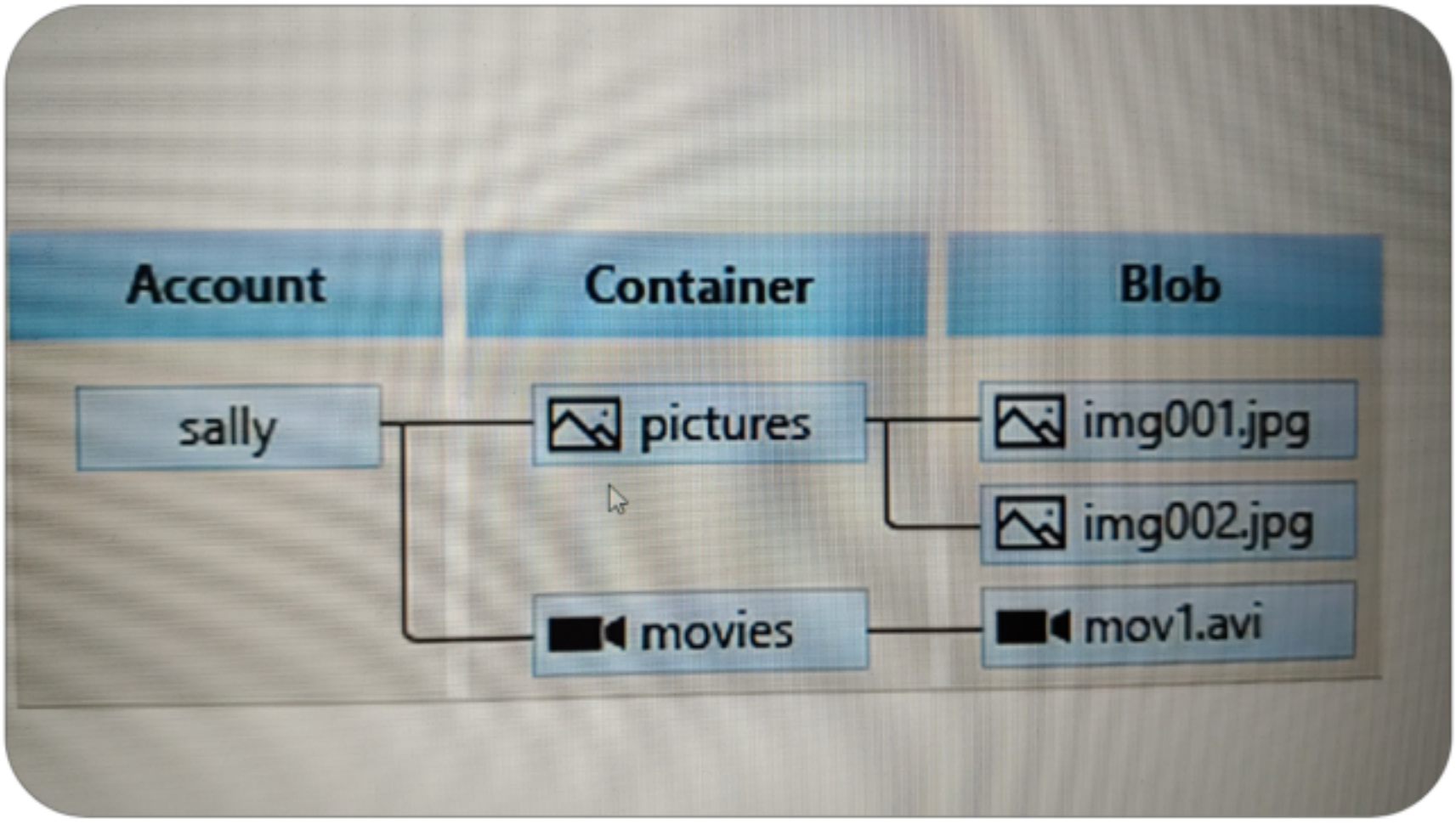


- Tables are similar to Cosmos DB, but less optimized for performance and availability. But it is cheaper than Cosmos DB.
- Files are related more to compute quite low level.
- **Blobs Storage:** Stands for Binary Large Objects. It is great for files, documents, videos, images, large texts etc. It is extremely cost

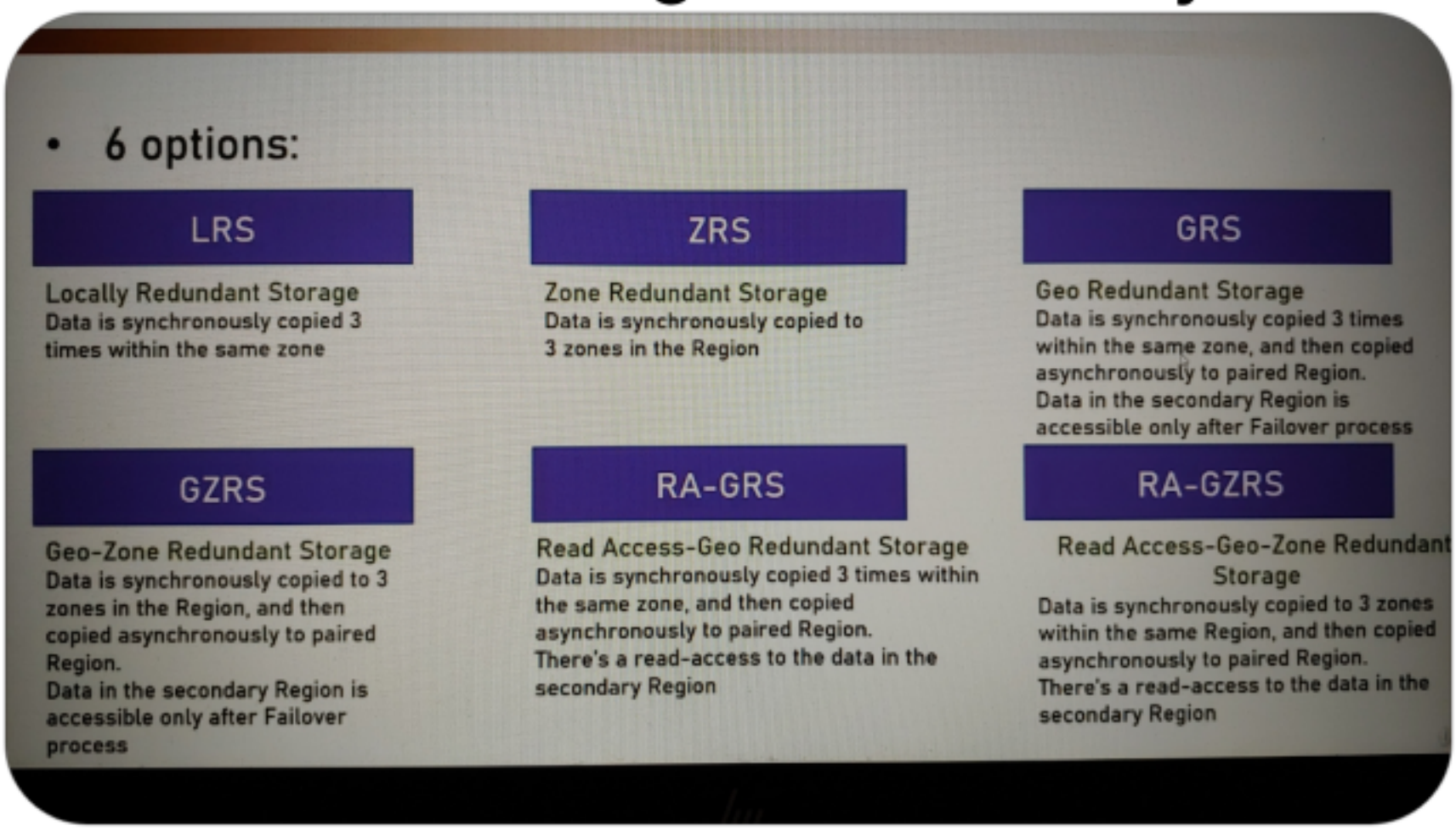
effective and cheapest data store of all. It can massively scalable. It has great availability options. It is extremely easy to use.

- ≡ ○ It is usually used in conjunction with SQL/NoSQL database, so that databases stores the actual data of the system, and the Blobs storage stores related files.
- ≡ ○ **Security**: Similar to other databases it provides IP firewall rules, Service & Private Endpoints, Secure Communication etc
- ≡ ○ **Blobs Storage Structure:** At the top level it has Azure Storage Account, below it we create containers which are logical groups of blobs, and in the container we upload the blobs. This is similar to Cosmos DB where we have containers, inside which we store JSON Documents Here we don't deal with Json

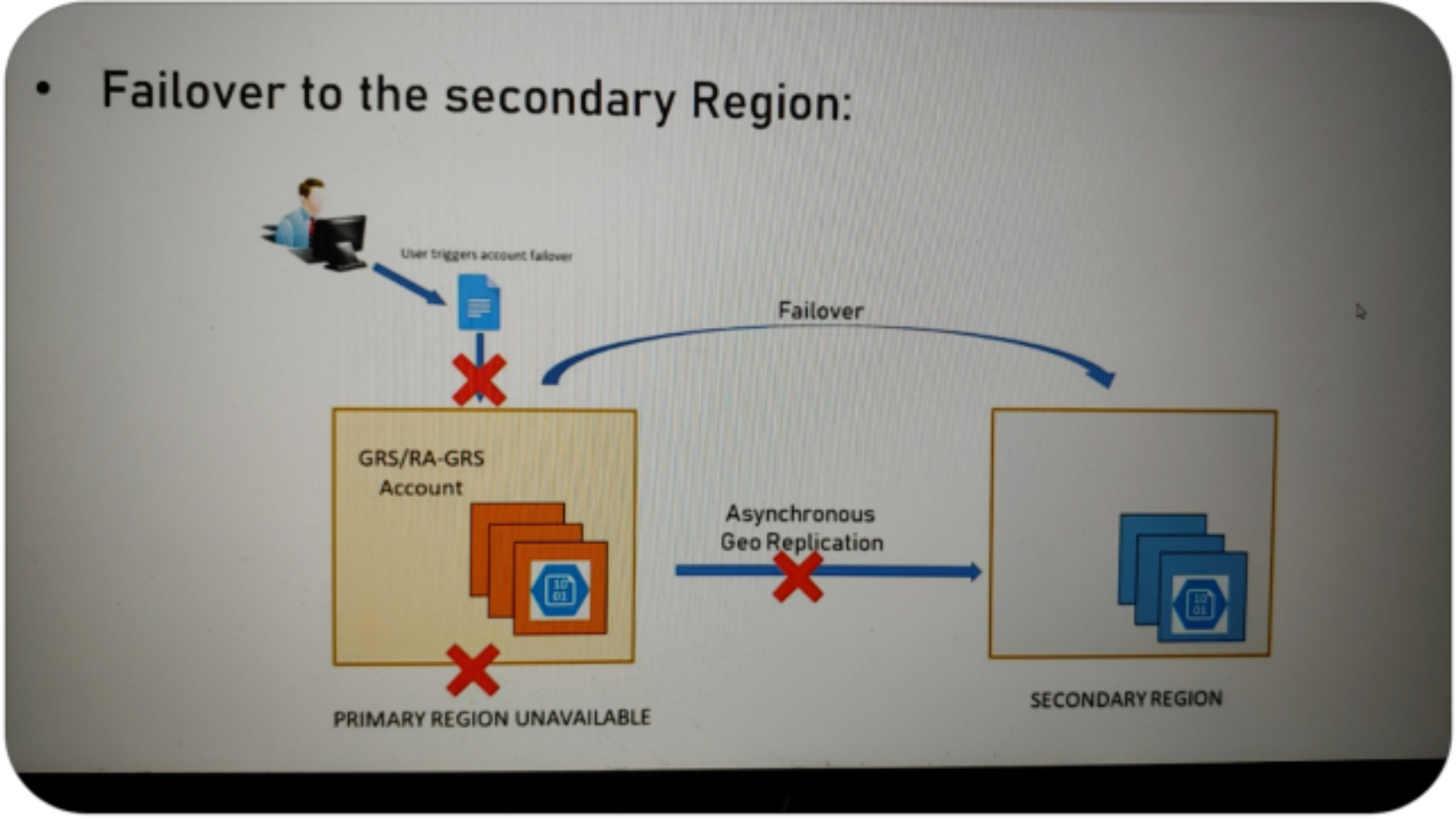
documents but with actual files.



≡ ○ Blobs Storage Redundancy:

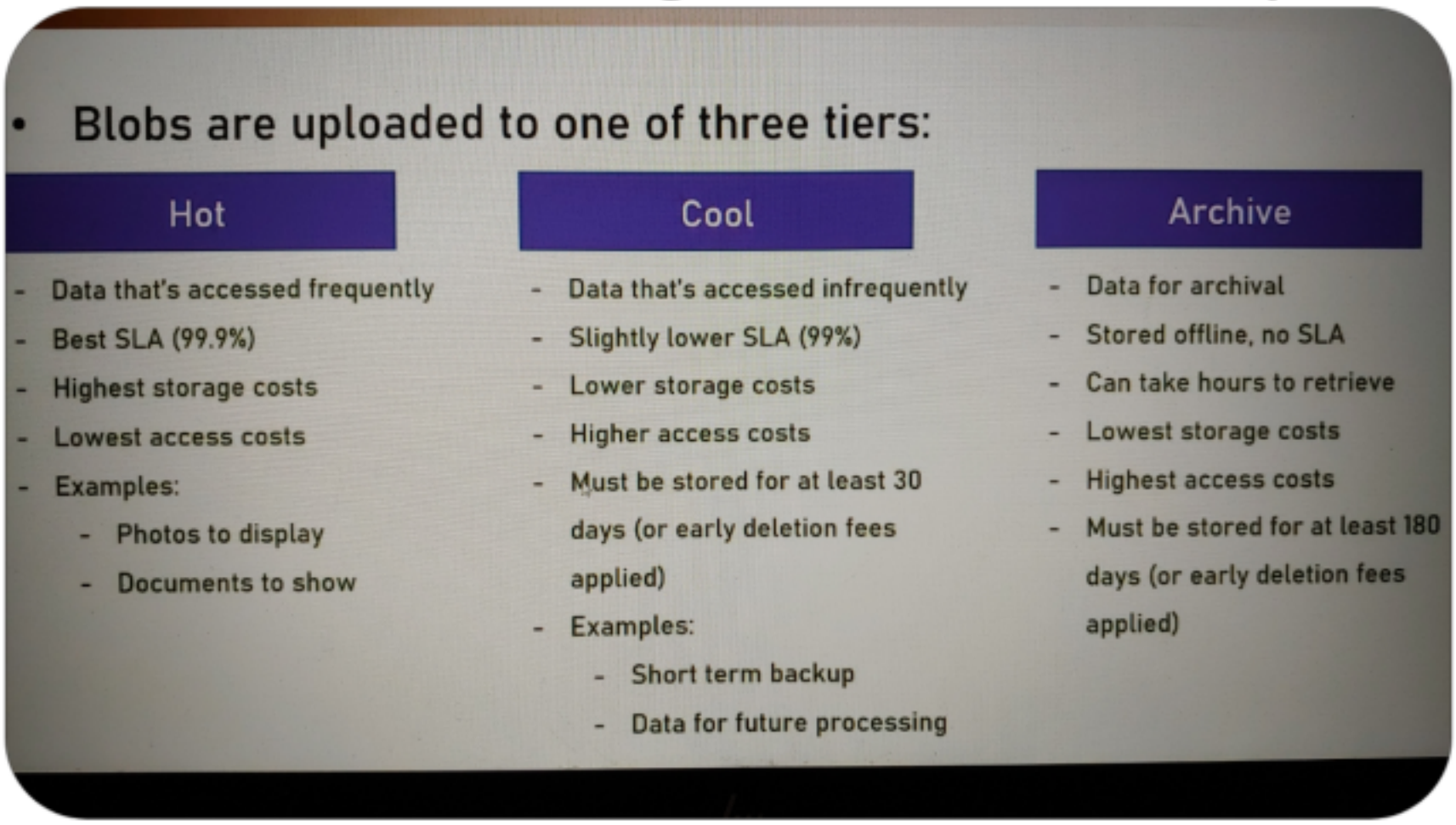


≡ ○ Blobs Storage Failover:



- If we add any data to primary region, these data will be copied to secondary region through asynchronous geo replication. If the primary region fails now, we can not have replication as the entire region is failed. So tue failover is initiated to the secondary region. That means the subsequent request to the storage account will be directed automatically to the secondary region after the failover complete.
- Failover can be initiated through Portal, Azure CLI or PowerShell.

- **Blobs Storage Tiers:** 3 Tier option



Blobs Storage in Action:

- Create a Storage Account in Portal. Select the right redundancy/replication type(ex RA-GRS). Select the recovery option if needed like turn on soft delete for blobs.
- Now Storage Account is created. Next add a container in it with a proper name and right access level(either private, container or blob).
- Once container is created add

any blobs to it by browsing a file from system and upload. While uploading a blob, also select the right access tier either hot, cool or archive.

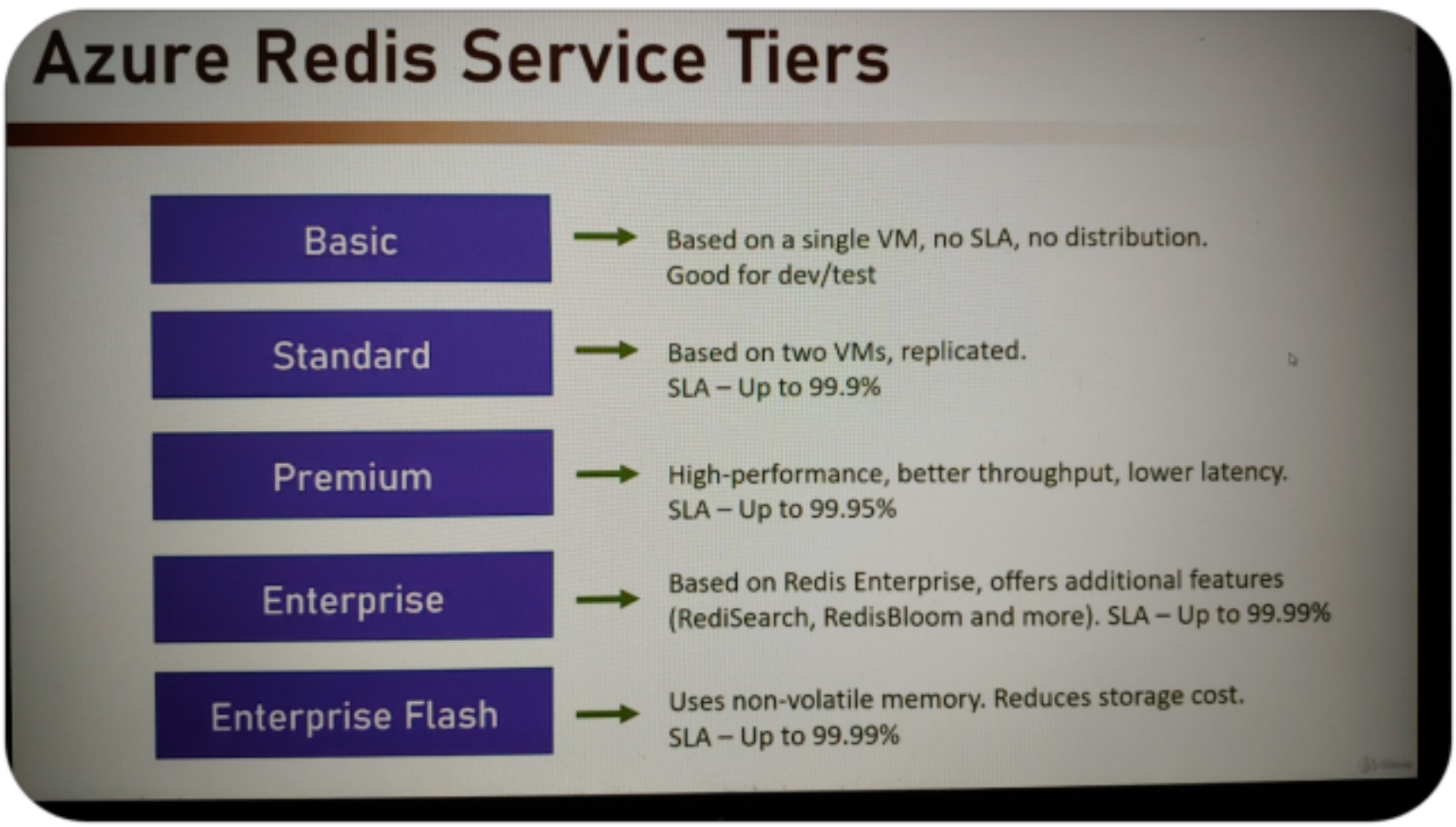
- ≡ ○ Once this image is uploaded we can access it through a URL, if we didn't set the private access before.
- ≡ ○ If we set private access then we can configure either Azure AD, Access Keys or Shared Access Signature(SAS) to access such private blobs.

Azure Redis

- ≡ ○ Its a managed Redis database on Azure. Redis provides lightning fast in-memory distributed cache, where the data will not be stored on the disk. Its the default Distributed Cache in the industry.

Its great for short lived, frequently accessed data such as shopping cart, stock queues etc.

- **Redis Service Tires:**



Redis in Action

- Create a new Azure Cache for Redis in Portal. Select a unique DNS name of the Redis. Also select the right cache type.
- Once its created we can use its Host name to connect to it from any application. Also, we need to copy primary and primary connection string access keys of this Redis from portal and use it

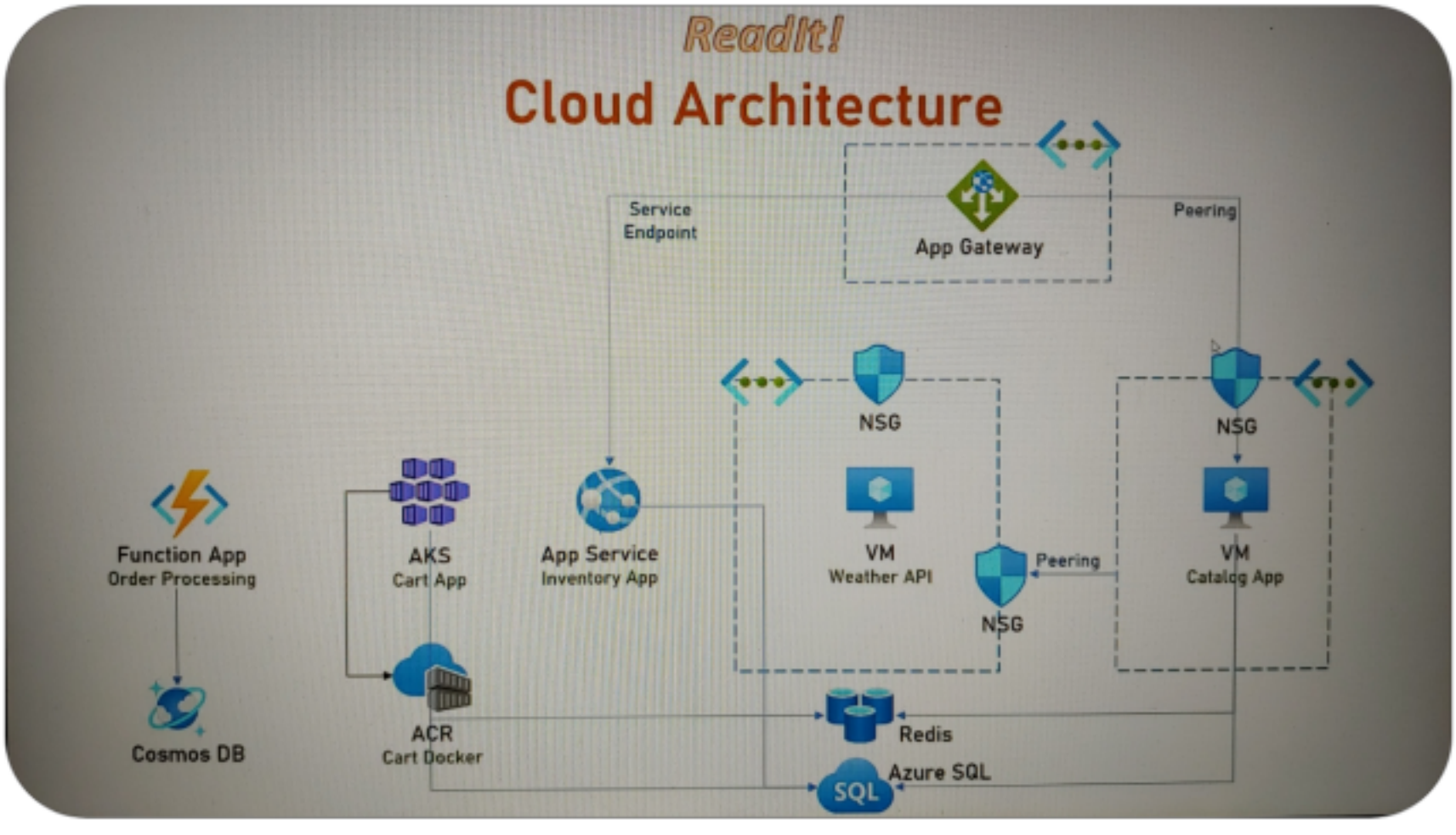
in the application, so that application can connect to Redis on Azure.

- For example, We can have one Catalog online application, where we save a shopping cart details of an user selection to Redis and then retrieve the total number of items available in the cart from Redis and display it in UI.
- Later, we can have another application called Shopping Cart, which we will fetch all the details of an user cart from Redis and display it in the UI.

How to select Data Store Solution

Data Type	Used For...	Examples	Options in Azure
Relational	Structured data	Items in store, demographic data	Azure SQL, MySQL, PostgreSQL
NoSQL	Semi-structured data	Reviews, Log records, when flexibility is required	Cosmos DB (with SQL, Mongo, Azure Table API)
Graph	Data representing relationships	Family tree	Cosmos DB (with Gremlin API)
Blob	Files, videos, docs	Items' photos	Azure Blob Storage

Current Architecture



- Changes here are the addition of DBs like Azure SQL, Redis and Cosmos DB.

- Inventory App which is connected to Azure SQL will update the remaining count of the books.
- Catalog App which is connected to Azure SQL will load all the available books and shows a button to add each book to shopping cart if the available count is more than zero. It also connected to Redis, to get the data about the Shopping Cart.
- Shopping Cart App which is connected to Azure SQL to fetch the details of each books which are added to cart and it also connected to Redis to get the Cart details.
- Finally Function App connected to Cosmos DB, which will store the placed order details. Order can be placed from Shopping Cart App.
- So till now we have implemented

applications, secured networks and many databases.

Are we done?

- Nope ☺
  - The Orders Function is publicly available and synchronous
  - The inventory page is open for everyone
  - We don't really know how the app is functioning
  - The website is not redundant – what happens if the whole region goes down?

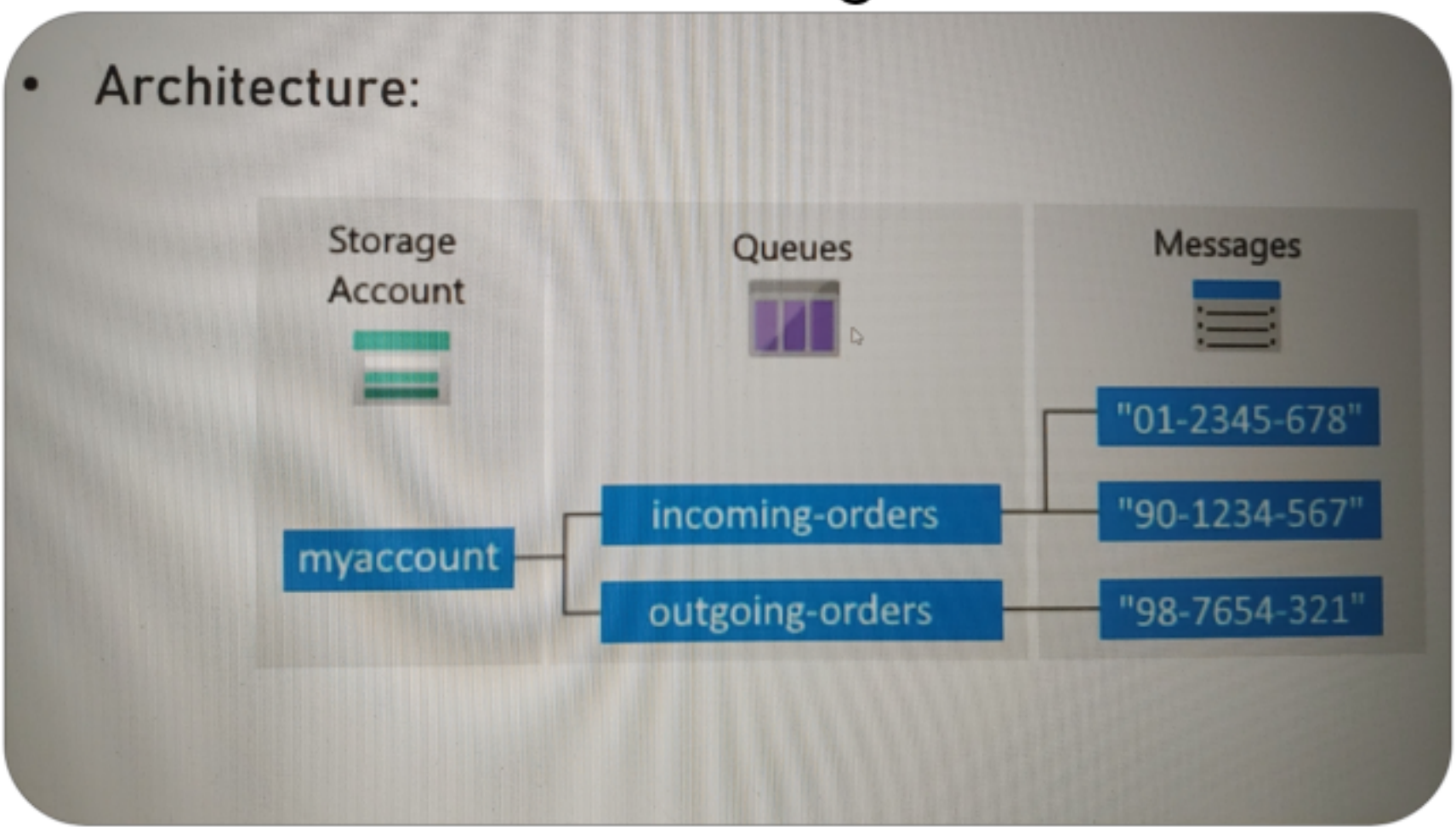
Messaging in Azure

- Messaging is extremely important aspect of Software Architecture especially between services. Messaging services must be able to handle load, throughput and great latency. Its a core part of every microservice architecture.

≡ ○ Azure has 4 fully managed messaging services which are Storage Queue, Event Grid, Service Bus, and Event Hubs.

Storage Queue

≡ ○ Its part of Azure Storage Account. Its the simplest queue implementation. It basically has 3 simple components. Create queue, Send message and Receive message. No special pricing for the queue, as its included in Storage Account.



- For the development, we need to use Client libraries of the corresponding languages.

Storage Queue in Action

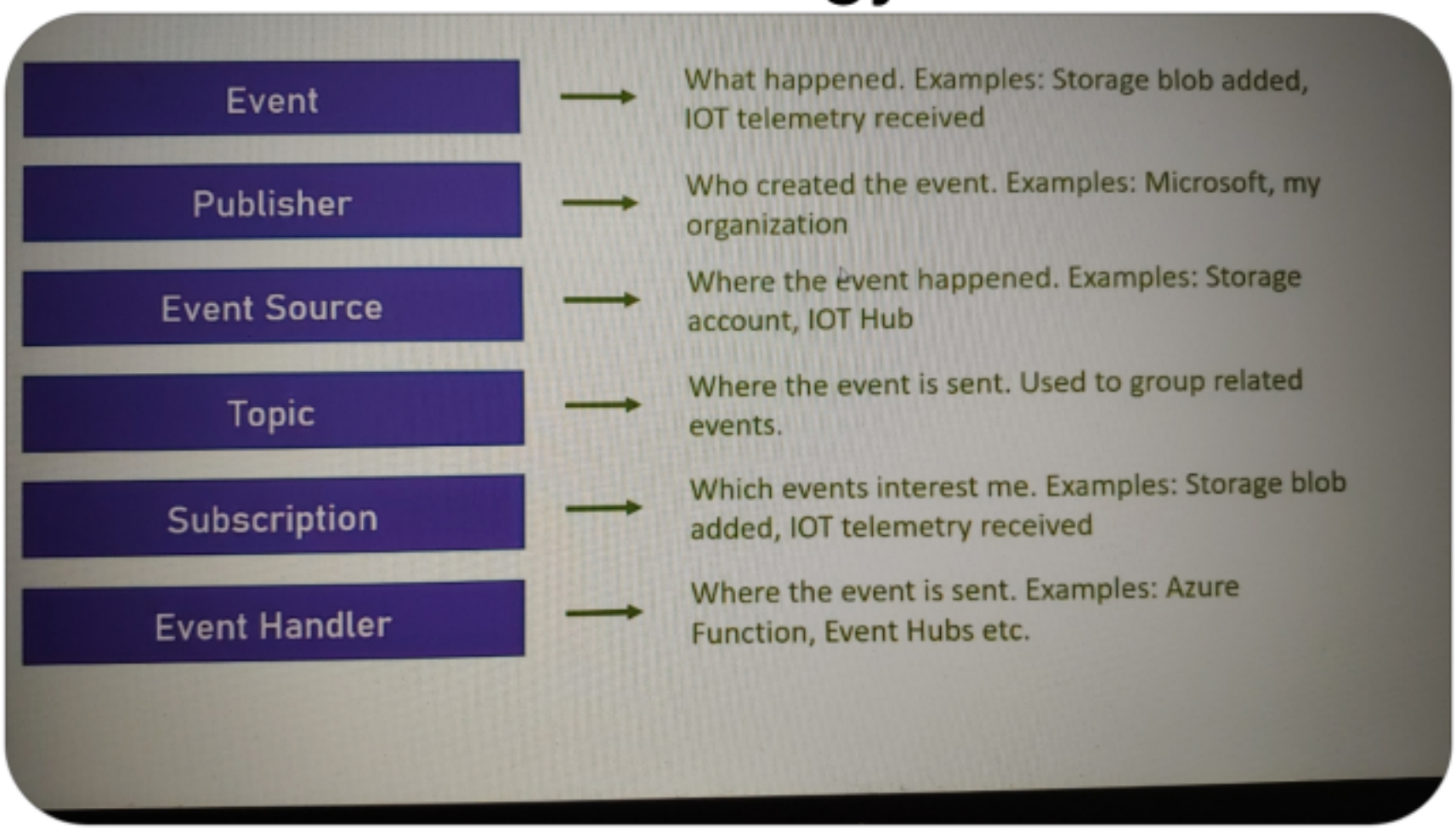
- Create a storage account first and then create a queue inside of it with a valid name. Then we can add a message which will send the typed message to this queue.
- Now create some sample application which uses the Storage Queue Client libraries and connect to the queue which we created and reads the data. We also need to provide the storage account keys. Once the data is read, it will be deleted automatically from the queue.

Event Grid

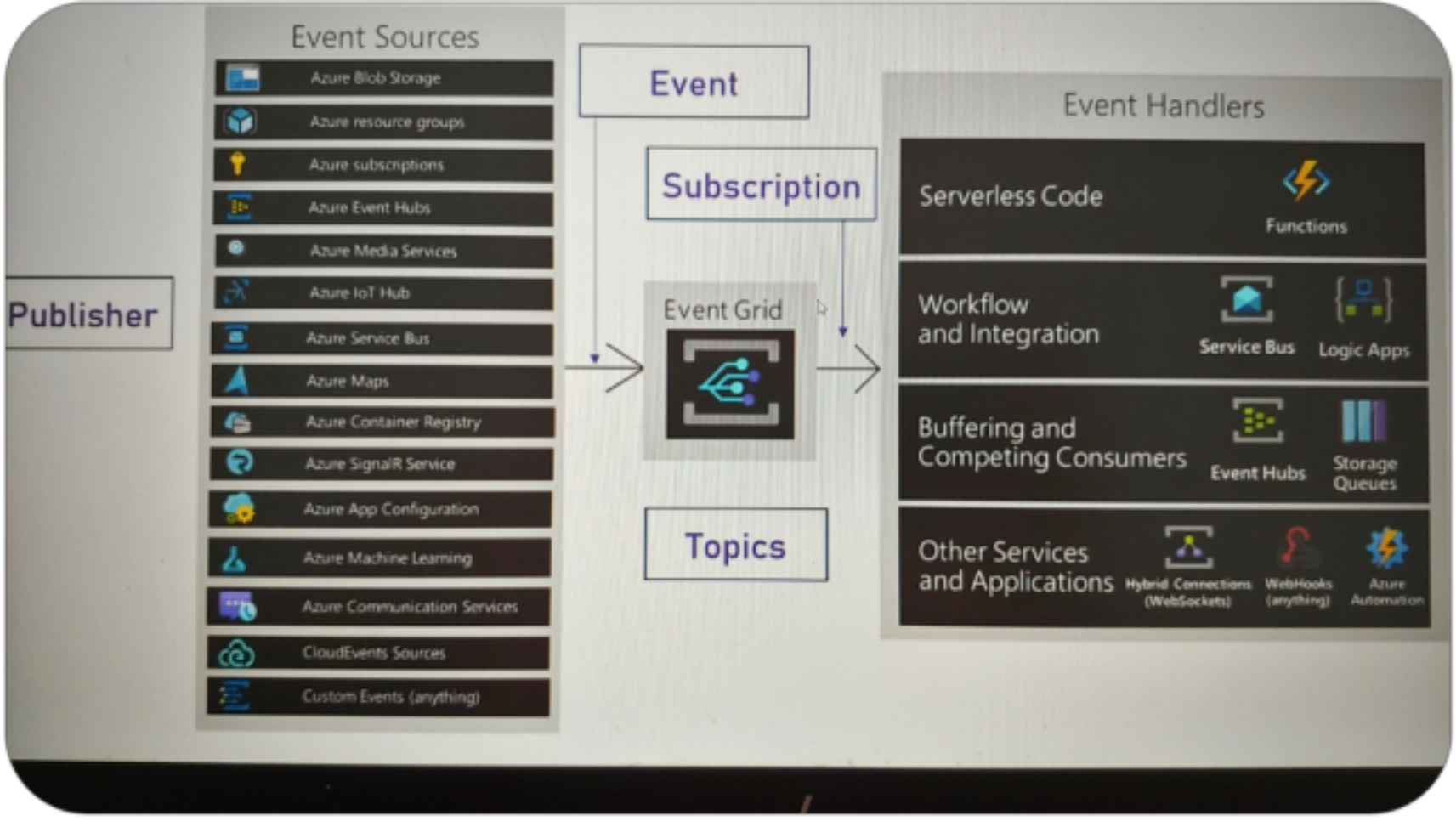
- It allows building event based architecture, that means it

publishes events to interested parties. It does not contains any queue so the order of the events can not guaranteed.

- It has strong integration with many Azure services. Extremely cost effective and one of the simplest pricing scheme in Azure. No tiers to select & HA is built in.
- **Basic Terminology of Event Grid**:



- **Architecture of Event Grid**:



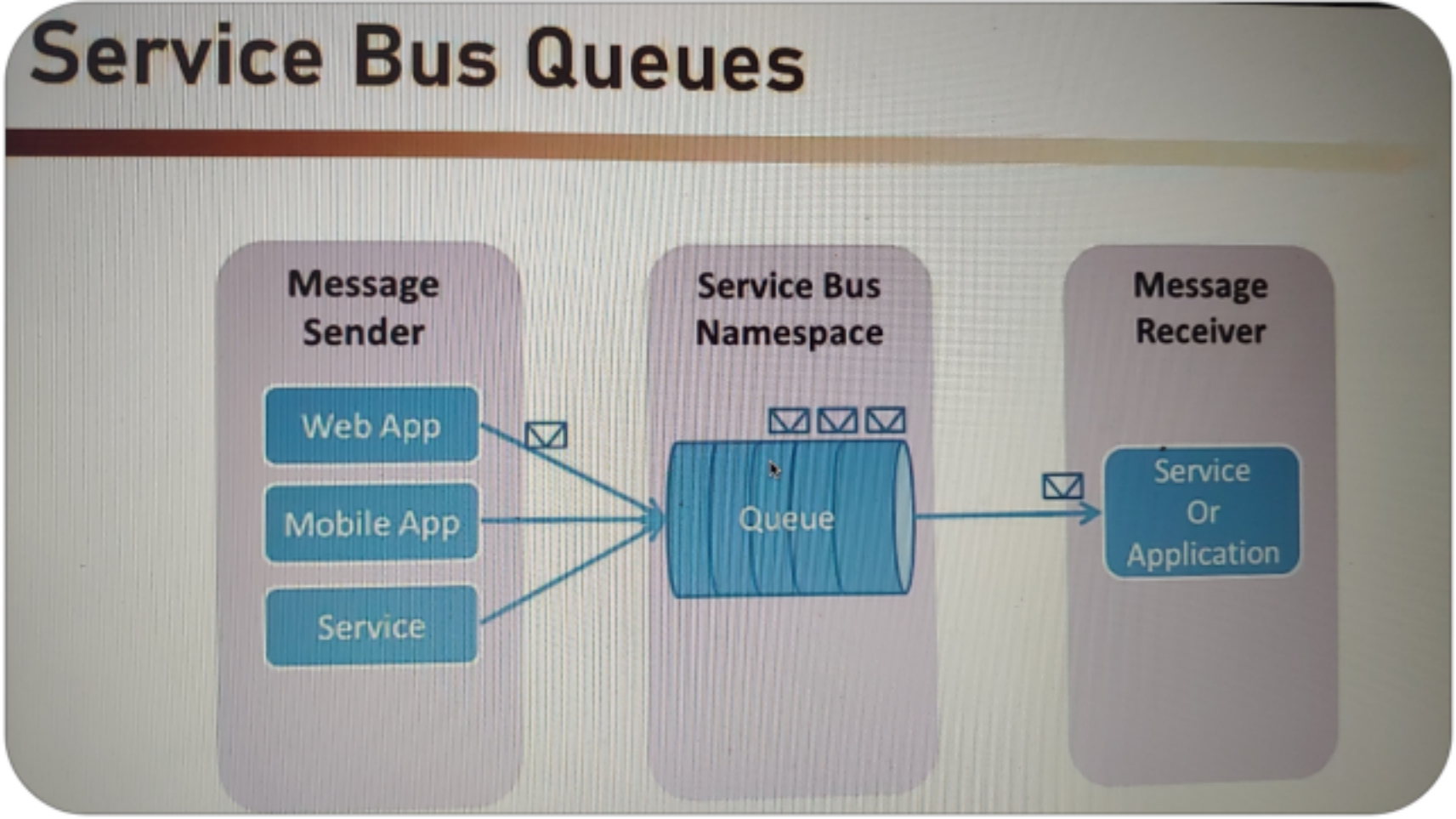
Event Grid in Action

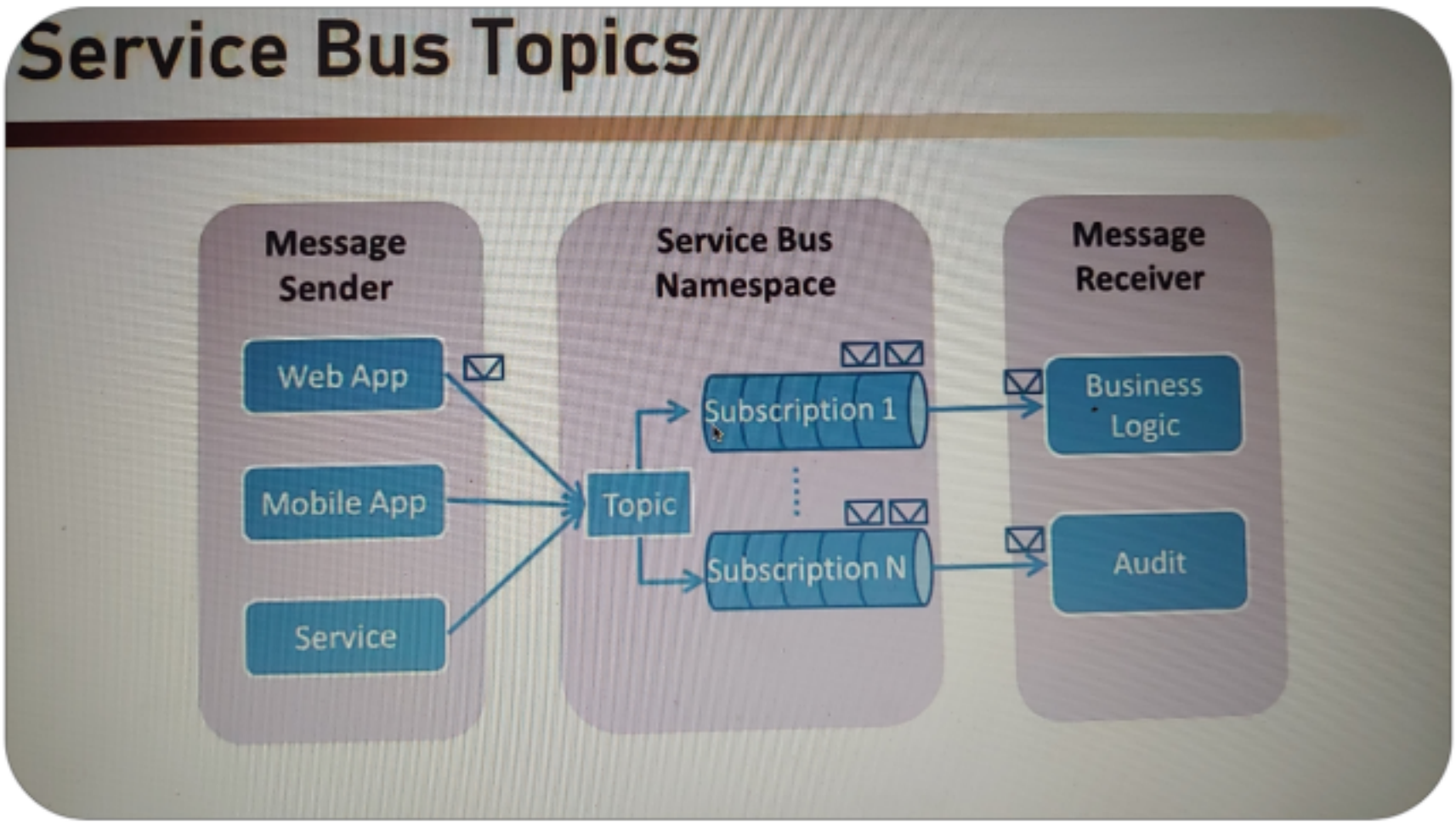
- Create a new 'Event Grid System Topics' in Portal. Select the specific Topic Type as per the source of the events. For example Storage Accounts Blob. And provide a name to this topic.
- Then in the created Event Grid select and subscribe a specific listener for this event.
- Use cases for example is, when a new order is generated and placed in a storage account blob, then an event 'Blob Created' will be generated automatically and

which will be published to the topic of this Event Grid and we can write a custom Azure function which can subscribe and listen to this Event Grid events and handles them by saving orders to Cosmos DB. In the function we can configure it to receive the traffic only from event grid.

Service Bus

- Fully managed, full blown message queueing service. Its extremely durable so messages will lightly to loose. Supports point to point (queue) like Storage Queue and pub/sub (topic) like Event Grid scenarios. It is also compatible with AMQP and JMS protocol.

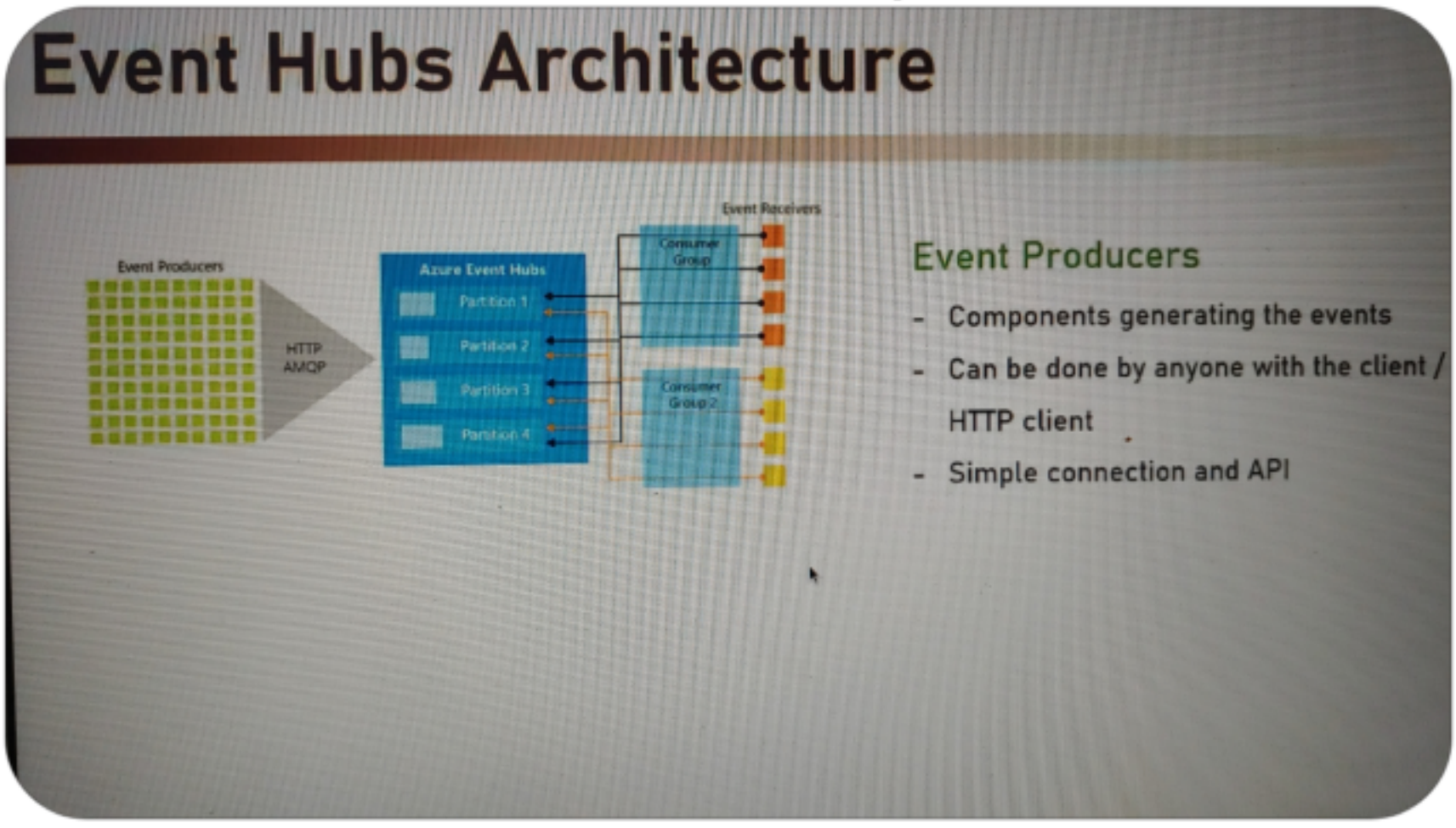




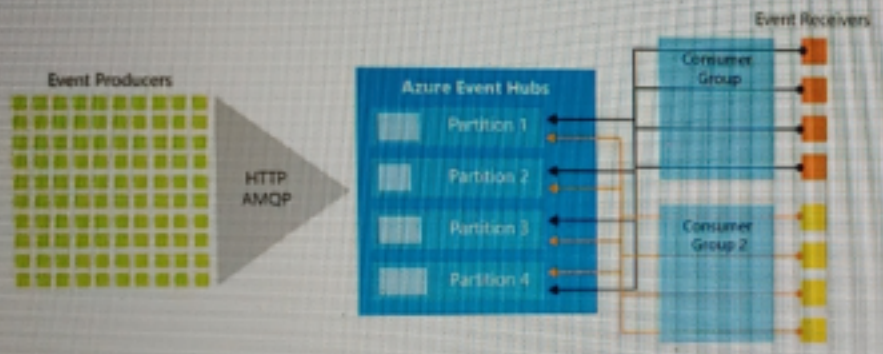
- It also provides advanced feature like Message Sessions, Dead-letter topic, Scheduled delivery, Transactions, duplicate detection etc.

Event Hubs

- Its a Big Data streaming platform and Event ingestion service. Its basically a managed Kafka implementation. It can receive millions of events per second.



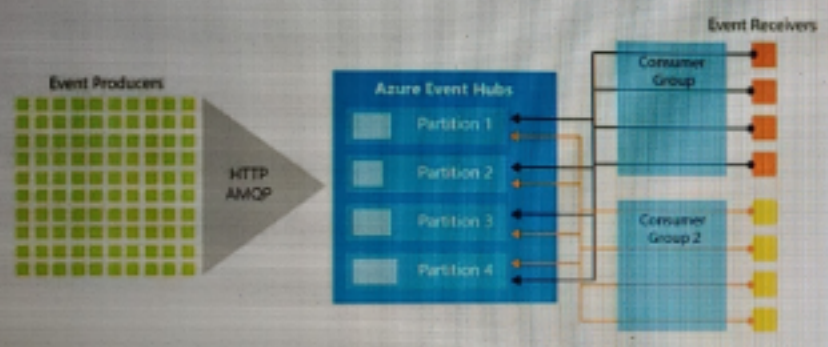
Event Hubs Architecture



Partition

- Single event stream
- Can think of it as a single queue
- Guarantees order
- Limited availability
- Better to spread messages across partitions to improve availability
- ...but then order is not guaranteed

Event Hubs Architecture



Consumer Group

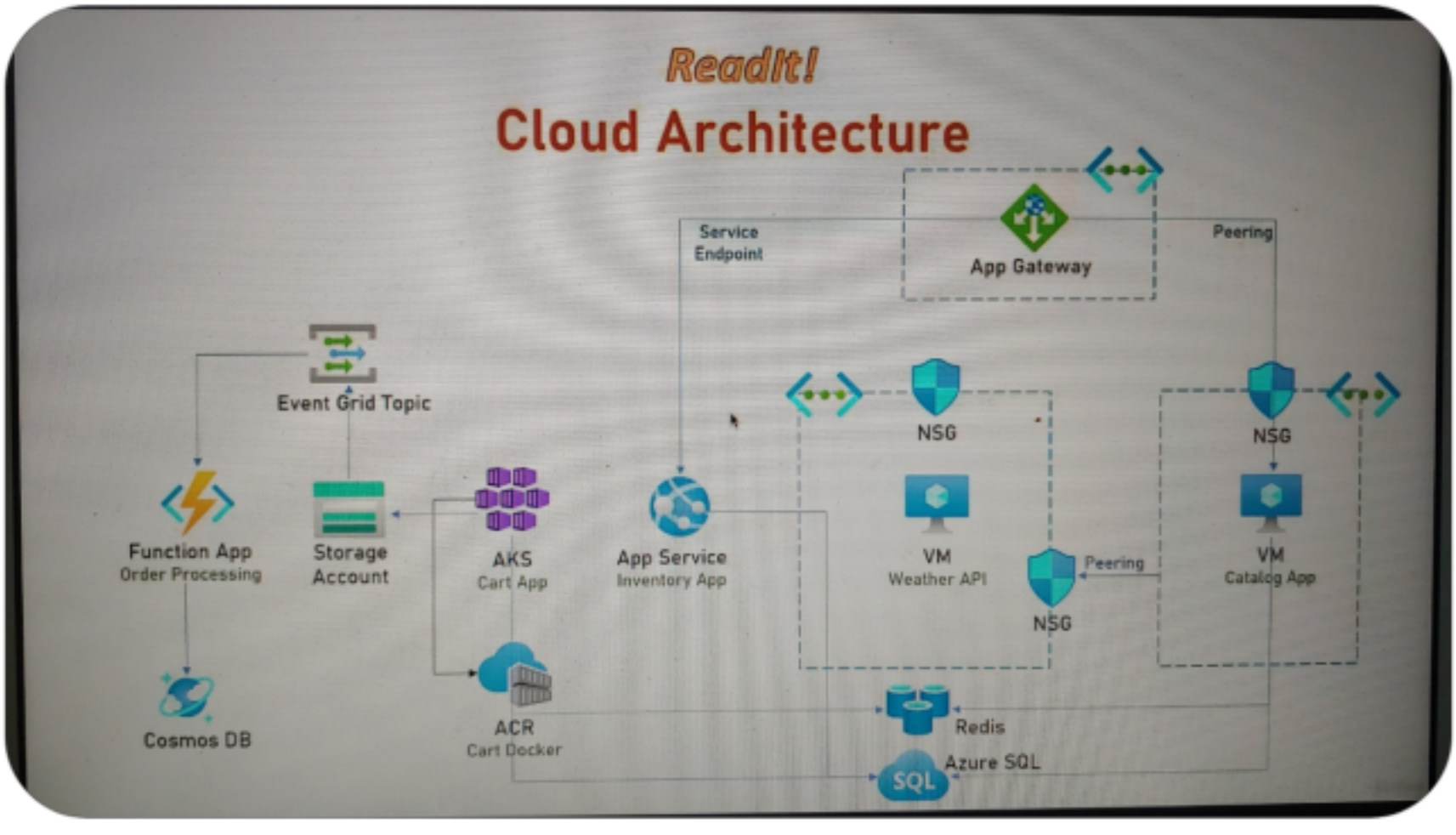
- Logical group of receivers, belong to the same application
- Example:
  - Receivers for processing telemetry
    - Consumer Group
  - Receivers for storing the telemetry
    - Consumer Group
- Event receiving is done via AMQP protocol

Selecting Messaging Solution

Selecting Messaging Solution

Service	Used For...	Guarantees Order	Max Msg Size	And also...
Storage Queue	Dead simple queueing	Yes	64KB	Extremely simple, no additional cost
Event Grid	Event driven architectures	No	1MB	Great integration with other services
Service Bus	Advanced queueing solutions	Yes	256KB	Advanced messaging features, durable
Event Hubs	Big data streaming	Yes	1MB	Low latency, designed for heavy load

Current Architecture



- Major change here is addition of Event Grid Topic which connects the Storage Account and the

Function App. That means this Function App no longer triggered through a http request, but by Event Grid trigger which makes the function more secure and asynchronous. Cart application will store the placed order into this Storage Account, which finally will be saved in Cosmos DB by Order Function App.

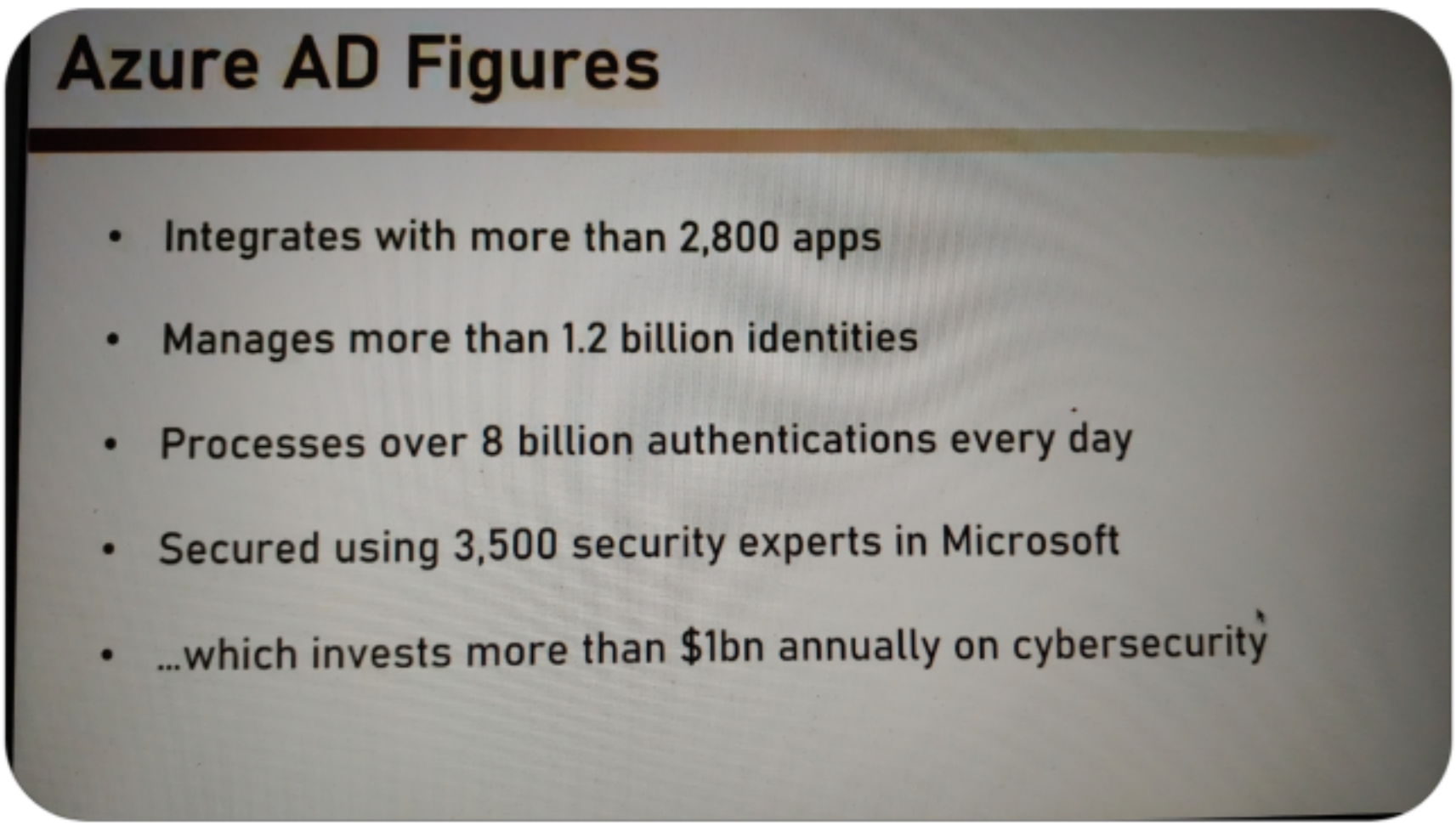
==========

Azure Active Directory (AD)

- Its a central identity and access management cloud service. Its the largest identity and access management service in the world. It is used to manage access to thousands of apps, including the Azure Portal and their resources including our own Apps. Basically

it helps in having Authentication and Authorization for the Apps.

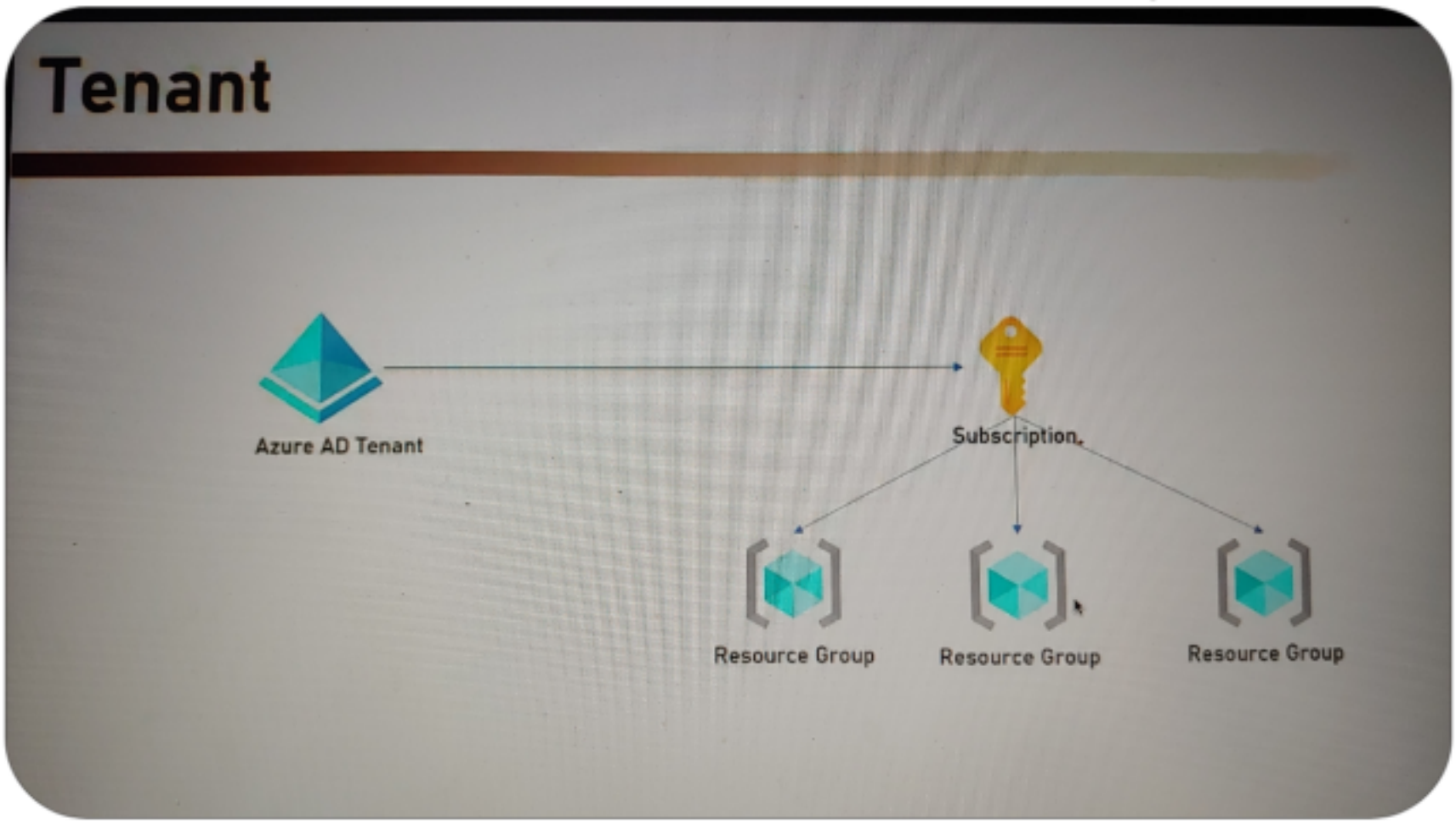
- ○ Its not an Azure resource. It is actually an external resource that can connect to many Apps and one of them is Azure.
- ○ One of the support provided by Azure AD is to control the access to Azure resources by setting up **users, groups and roles**.
- ○ We can use Azure AD to add authentication to out own apps. It can also done via Azure AD B2C.



- ○ **Tenants**: A specific instance of Azure AD containing users and

groups. It is also called directory. A tenant can be assigned or connect to multiple subscriptions.

- ○ Tenants are not part of the subscription hierarchy. It exists beside the subscription. For new subscriptions, a nee tenant is also created automatically.



- ○ We no need to create Azure Active Directory tenant in the portal as its not an Azure resource. When we create a subscription, a tenant will create automatically and attaches to

this subscription.

- Default name of the Azure AD tenant which is created for us is 'Default Directory'. We can change this name to any name in portal.
- If we go to the users tab under this directory, we can see list of all users who are part of this directory, usually it will be you.
- The Assigned Role tab contains list of roles which defines what we allowed to do in this directory. By default we will get Global Administrator role by which we can manage all aspects of Azure AD and Microsoft services that uses Azure AD identities.

- **Users and Groups:** These two are the main three objects managed by Azure AD. The third one is Role Through these, Azure AD

manages and stores the users that are part of the tenant. Also, it groups the users in Groups. For example IT Admins group, Developer groups etc which will holds the users which will belongs to these logical groups. Groups allows defining roles to it instead of each user.

- So each user will be put into some groups and multiple roles are assigned to these groups.

Users and Groups in Action

- Go to portal, go to the Azure Active Directory. Select the users option and create a new user under this Azure AD with name and password. By default this user is not linked to any groups and it has a AD User role.
- Now login to Portal with this user credentials and update password.

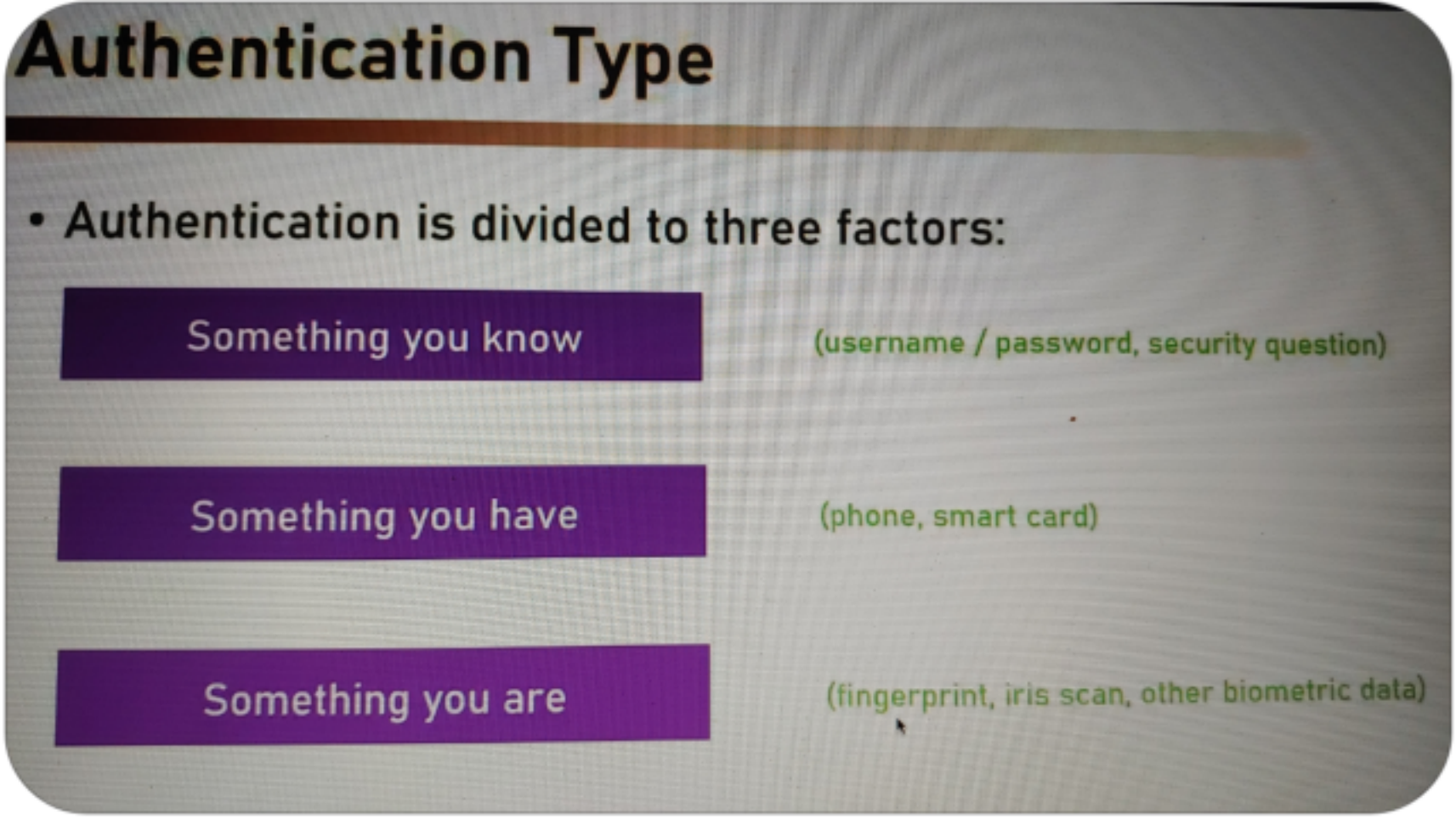
- After logged in, notice that we can not create any resources because this user don't have any subscription. We didn't connect this user to our subscription.
- We can also create a new Guest User who are not part of this directory and send an invitation to them. Once the user accepts this invite they can also become part of our Azure Active Directory. But they will not be able to create any resources because their user is not attached to subscription.
- Now select the Groups option under the AD. Click on New Group and create a new one with name for ex 'Azure Architect' and add members to this group by selecting the previously created users. So we have one new group with 2 users in it.

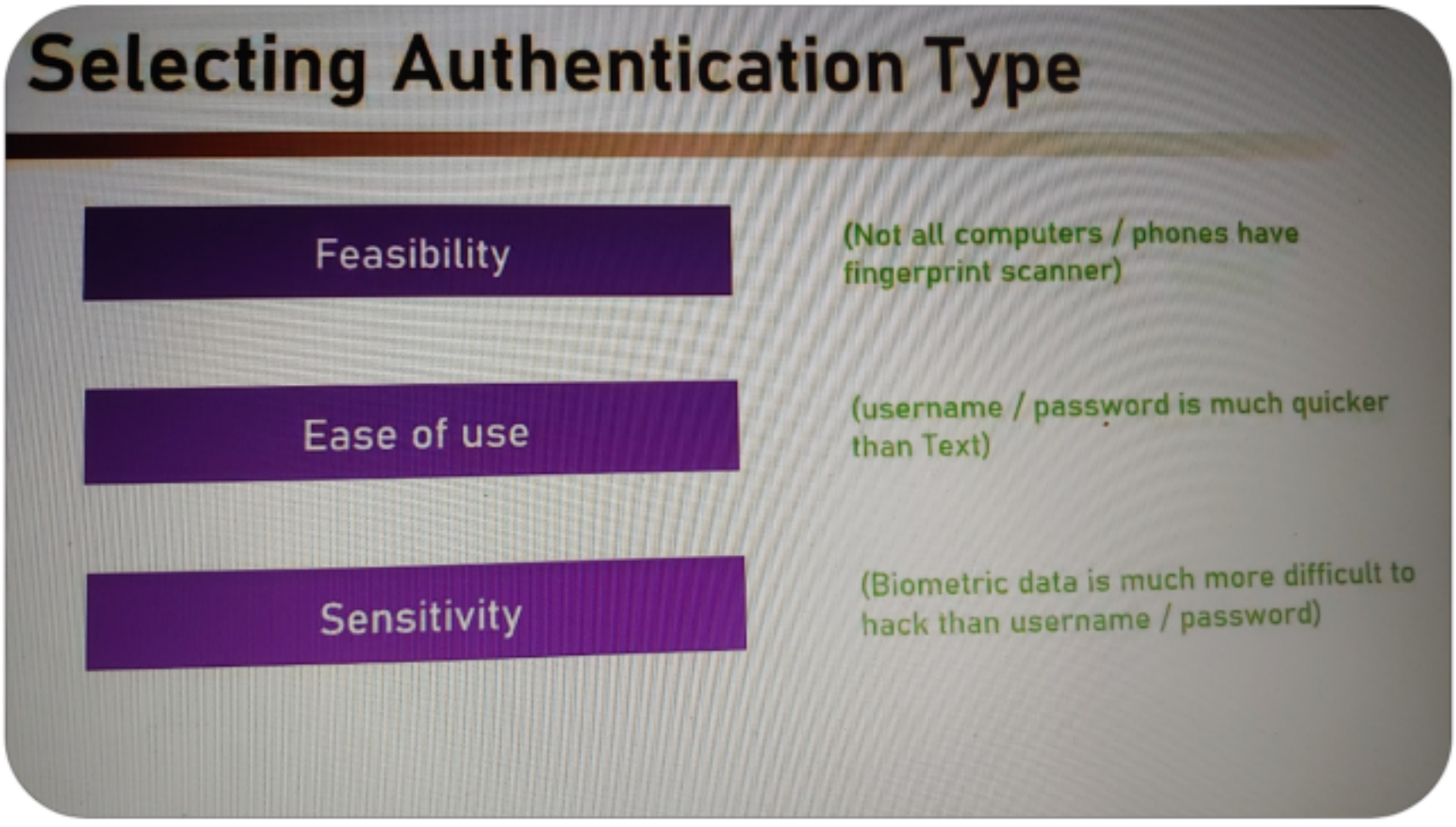
- **Azure AD Licenses:** Azure AD licenses have great effect on the functionality & price of Azure AD.

	Free	Premium 1	Premium 2
Max Objects	500,000	Unlimited	Unlimited
Users & Groups	X	X	X
MFA	X (All or nothing)	X (With Conditional Access)	X (With Conditional Access)
Dynamic Groups		X	X
Conditional Access		X	X
Risk Detention			X
Risk based Conditional Access			X
Privileged Identity Management (PIM)			X
Price	Free	6$ user / month	9$ user / month

Multi Factor Authentication (MFA)

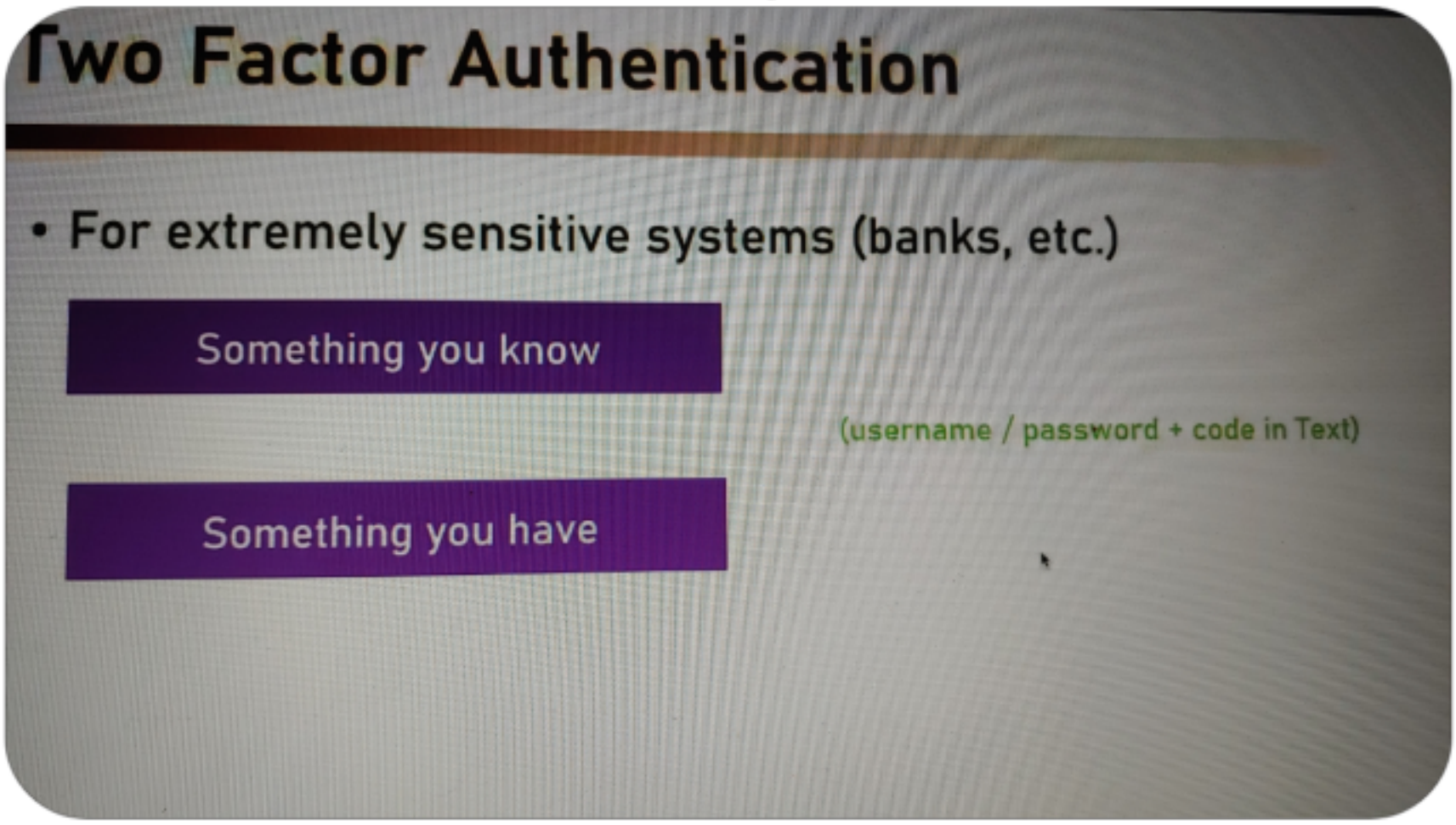
- **Authentication Type:**





≡ ○ **Two Factor Authentication:** Combination of 2 or more Authentication types for extremely sensitive systems. For example username/password +

code in text to phone.



- When selecting an Authentication Engine make sure it supports two factor/multi factor authentication
- In addition all these supports should come put of the box. Should not required development.
- **Azure AD Security Defaults:** Using Security Defaults we can increases protection of the organization in the Free tier. MFA will provides protection in Premium 1 and 2 tiers. It adds free configured security settings such as requiring all users to use

MFA, block legacy authentication.

- We can enable security defaults from the properties section of the Active Directory home page. Once its enabled, and if we try to login to the portal with username and password the Multi Factored Authentication kicks off so it will send a code to the email address and we need to enter that code for the successful login.

Role Based Access Control (RBAC)

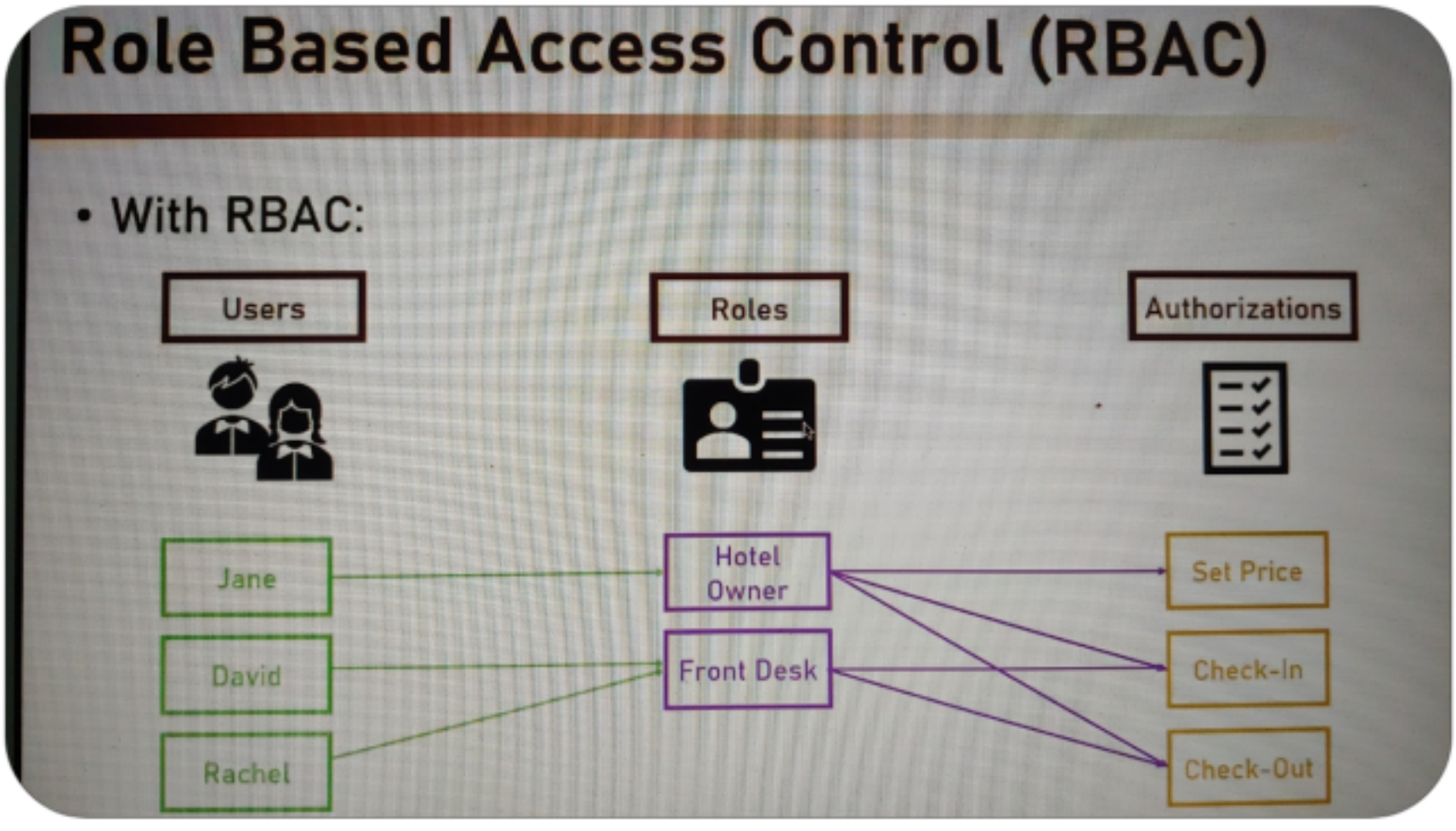
- In the past authorization was defined per user or user group. For examples David is allowed to add items to the inventory or John is not allowed to read data of other doctors. This process is very granular and extremely hard to maintain.
- For example what happens of

David leaves the organization? System Admin has to go through all the authorization definition of David which can be dozens and cancel each one of them. Another example is what happens when we add new category in the store and we need to allow all the employees to add items to it. It means we need to go through all users in the system and grant each and every one of them the right to add items to category

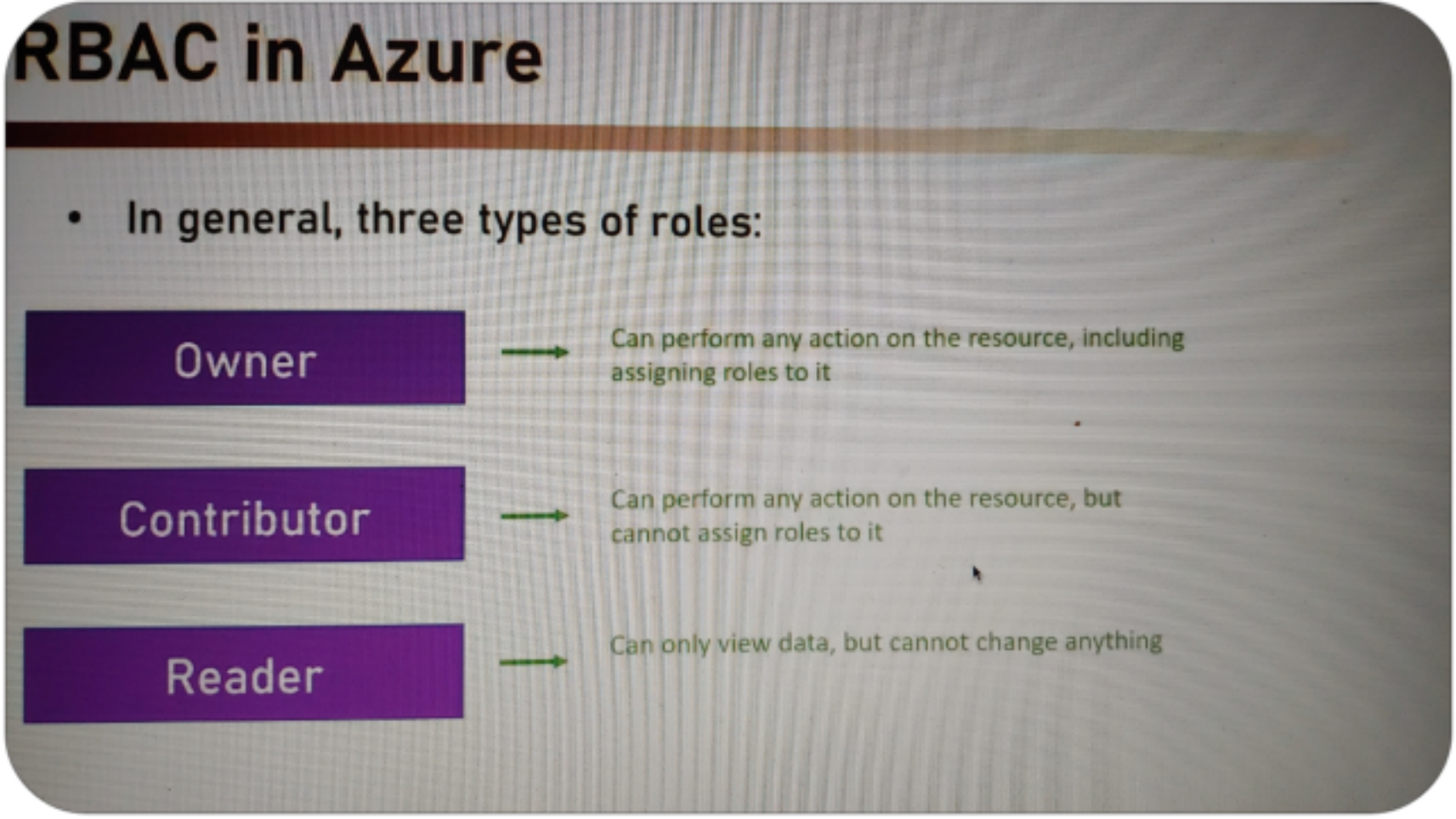
- To solve the above problem RBAC is invented.
- **RBAC:** With RBAC we have three elements which together defines Authorization. Those are Users, Roles and Authorizations/ Privileges. Once we define users, possible roles of a system, and the privileges which each role can contains, we associate/link roles

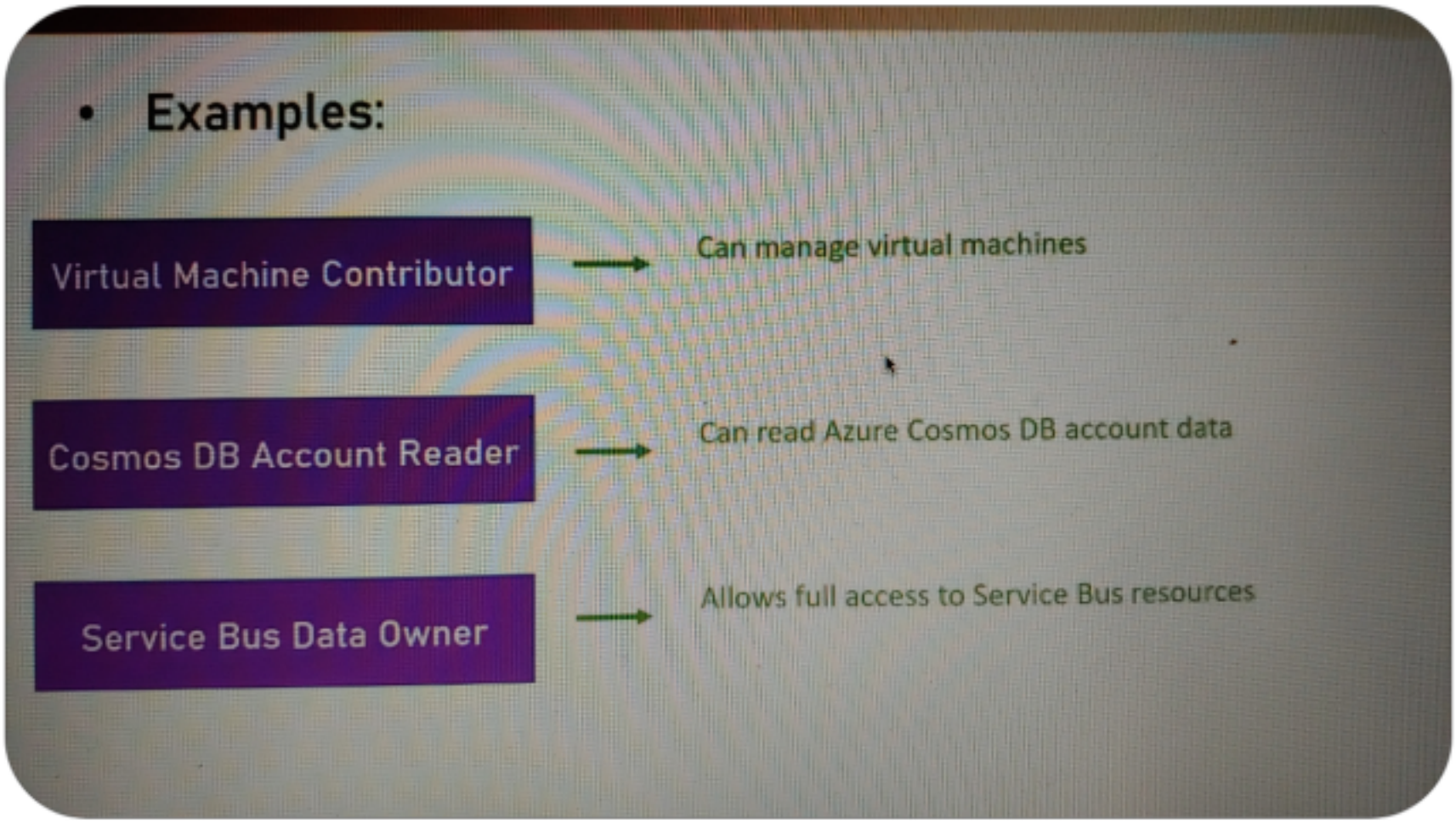
with Authorizations/Privileges. Next we associate each user with the corresponding role.

- Now we have a much simpler system to maintain. The privileges of the roles are more or less constants and are not changed frequently and the association of users to the roles are very simple and are usually one to one association. Now we can easily add new user and attach them to single or multiple role and remove a user and their corresponding association.



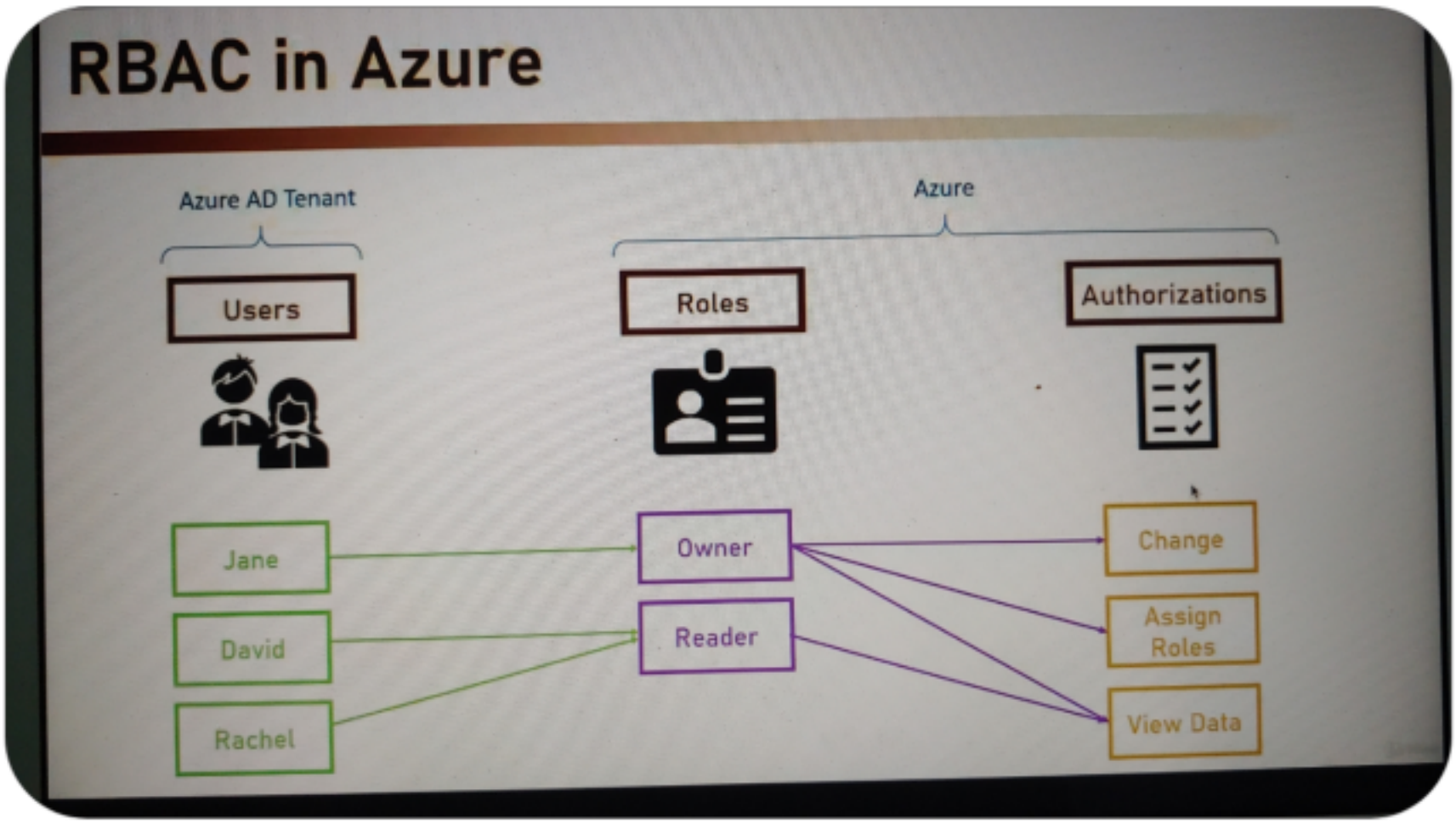
- **RBAC in Azure:** In order to perform any operation, or access any data in Azure you have to have the appropriate Role. For example if you want to create resource groups, Access data in SQL, See metrics of App Service etc you have to have the right role else you will get an empty portal.
- By default when we create a new Subscription, you will automatically granted the highest role available in Azure i.e Owner. That's why we can access the Portal even though we don't define any roles. But for any other users you will have to define roles in Azure and associate it to user.
- **Types of Roles in Azure:**





≡ ○ **How Users, Roles and Privileges looks in Azure:** Users are defined in Azure AD Tenant. Roles and Privileges are defined in Azure itself. So basically in order to

assign Roles in Azure, we assign Users and Groups from the Azure AD Tenant into Roles in Azure.



- Its always better to assign roles to groups and not to individual users so that its easy to maintain.
- **User -> Group -> Role -> Privilege**

Azure Roles in Action

- Every resource in Azure has a Access Control option, which basically allows us to assign Users and Groups and Roles to resources.
- Now go to a Resource Group and

under its Access Control option, we can assign the role 'Contributor' to the previously created 'Azure Architects' Group. So that each member/user which we added into the 'Azure Architects' group is going to be a Contributor to this resource group

- ○ Role Contributor allows to do whatever user wants with the resources except add or modify Role Assignments. Also, this group will not be able to do anything with other resources.
- ○ Now if we login with any user who are part of the Azure Architects group, we can able to see only this Resource Group and all the resources under this and we can add or remove any resource under it.
- ○ So, this way we can use the role assignments to grant

permissions to users or groups to perform various actions on Azure resources.

- ≡ ○ Also, by this way we can grant a permission to a particular resource group to any external user so that they will maintain all the resources of it.

Managed Identities

- ≡ ○ Its the ability to assign Azure AD identity to Azure resources. That means now this resource can connect to other Azure resources using this identity. So, with managed identities there is no meed to handle credentials like usernames, passwords etc.
- ≡ ○ Types of Managed Identities:

- Two types of Managed Identities:
  - System assigned – Managed by Azure, tied to the resource's lifecycle (when the resource is deleted – so is the identity)
  - User assigned – Managed by the user. Can be assigned to multiple resources, not tied to any lifecycle

- Resources that can be assigned Managed Identity are App service, VM, Event Grid, Functions etc
- Resources that can be authorized using Managed Identities are SQL, Event Hubs, Service Hubs, Storage, Key Vault etc

Managed Identities in Action

- Go to the App Service in Portal which we created. Select the identity option and enable the System assigned Managed Identity. Now Managed Identity was created and it is registered with Azure AD. Now in the App

Service code base, configure the authentication using AD. Also, we configure App Service to use the AD interactive authentication provider as the authentication provider for the SQL.

- Now go to SQL server which we created earlier, where under the overview page, we can see the Active Directory admin setting is not configured. This configuration allows Resources with managed Identities to connect to this SQL database. In this config we can define the new user which we created earlier as the Admin of this SQL Server.
- So now when we try to perform actions on this SQL server using AD, it will use this user as the Admin for these operations. So, login to SQL DB as this user and run commands that will allow the

managed identity of App Service to connect & perform various actions against this database. Commands are used to create a App Service user and then assign read, write & change DB structure permission to it.

- Now go to App Service configuration where connection string with username & password of SQL is configured. Instead of this now database should be connected using the managed identity of the App Service.
- Now we can access the App Service through configured API Gateway URL, which will connect to SQL and fetch the data.

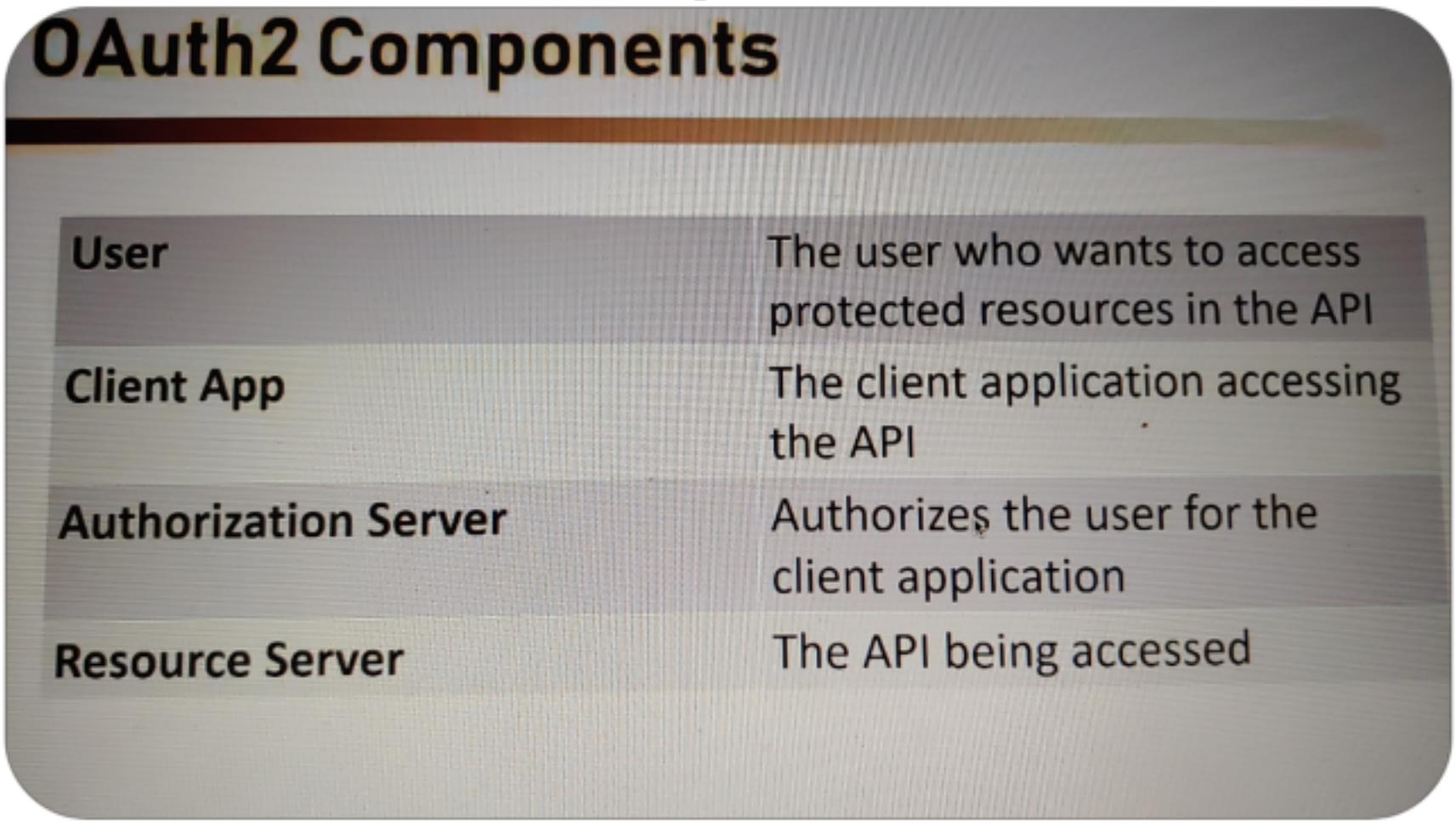
OAuth2 & JWT

- Azure AD can be used as

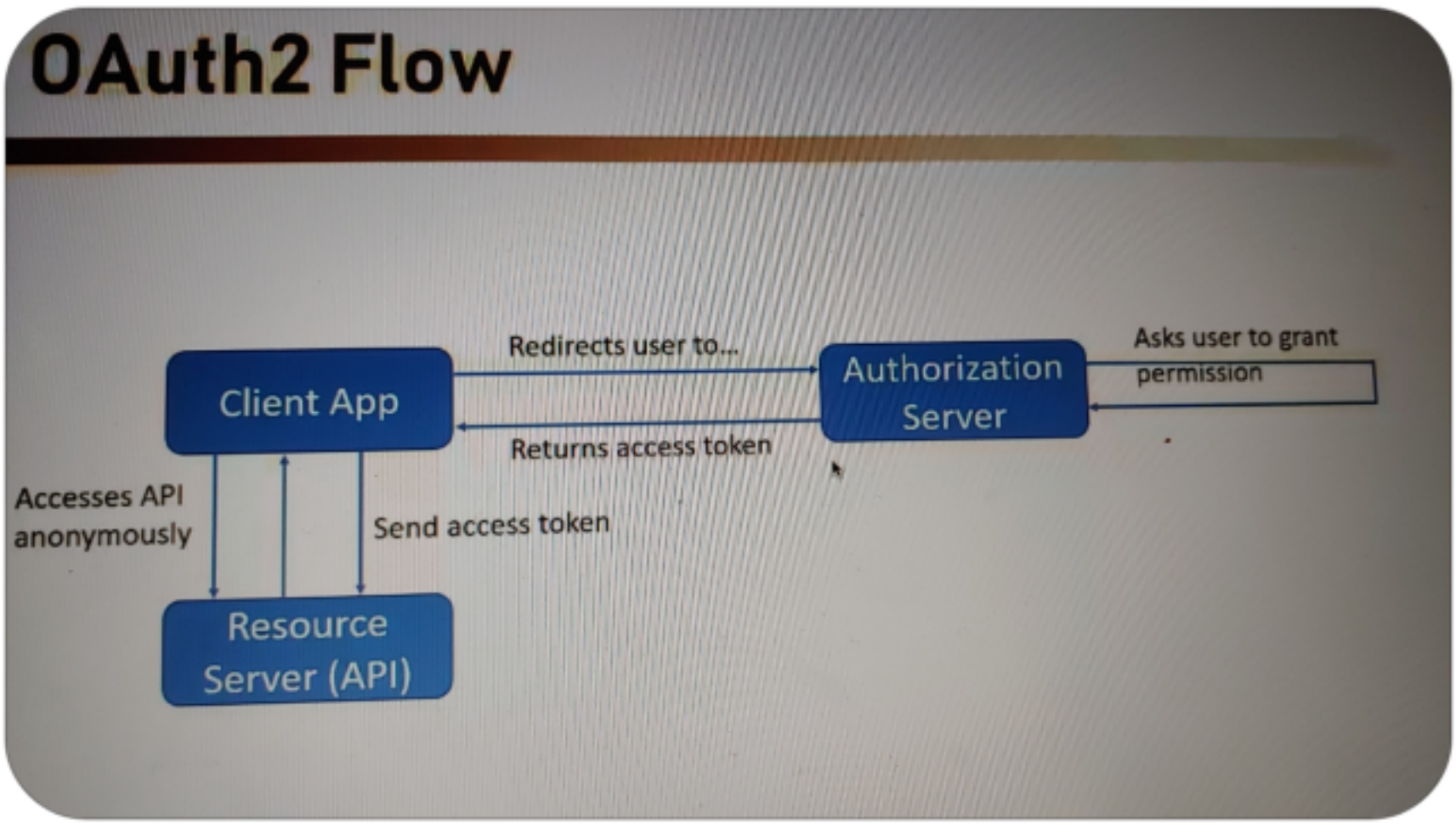
authentication engine not only to authenticate in Azure Portal but also on other apps including our own applications.

- ○ The Inventory App what we have should not be accessed by anyone. It should be accessible only for the stores employees.
- ○ The process of adding Azure AD authentication to another app: 1. Register our App in Azure AD 2. Add code to use Azure AD as authentication engine in our own app. In case of application which are of type App Service we can easily do it in Portal directly without code changes 3. Using OAuth and JWT implement authentication
- ○ **OAuth2:** Its a standard protocol for Authentication and Authorization. Its widely used mainly in Web Apps.

≡ ○ OAuth2 Components



≡ ○ OAuth2 Flow:



≡ ○ **App Registration:** In order to work the above flow, the Authorization Server should be familiar with Resource Server. For that, the

Resource server must register with Authorization Server. This is called App Registration. In this process Resource server mainly provides its name and the URL address to which the Authorization server should return the user details. Once its registered, Resource Server will get a unique Client ID and Secret which it needs to pass in every request.

- **JWT:** JSON Web Token. It contains the data the Resource server needs in order to authenticate the user. This token is returned by OAuth2 Authorization Server and it contains everything the Resource server needs so that it will know exactly who the user is and what is he allowed to do.
- **JWT Format**: it has 3 sections.

Header - contains type of token (JWT) and signing algorithm.
Payload - It contains the actual data of the user in any format.
Signature - so the JWT is signed electronically so that it cant be tampered with during the transfer from Authorization server to resource server. All the 3 parts are encoded in base64 and are concatenated with a .(dot)

Configuring Azure AD & Inventory App Service for Authentication:

- Go to Azure AD in Portal. Under AD go to App registration setting. Here we are going to add registration for the inventory app service, so that any access to this app will require authentication.
- Click new registration and provide name of the inventory app and a valid redirect URL and create it.
- Now go to Authentication setting,

and Select both the check box for Access tokens and ID tokens, as web apps needs this kind of grant for implementing full flow of authentication. Next go to Certificates & secrets setting and add a new Client Secrets for the Inventory App.

- Now configure Authentication and Authorization in Inventory App Service. Go to inventory app service home page and click Authentication & Authorization setting and enable App Service Authentication. Select &configure the specific Authentication provider like Azure Active Directory, Facebook, Google etc.
- In this example select Azure Active Directory option and provide client & tenant id &secret.
- Finally, make any required changes in the Inventory App

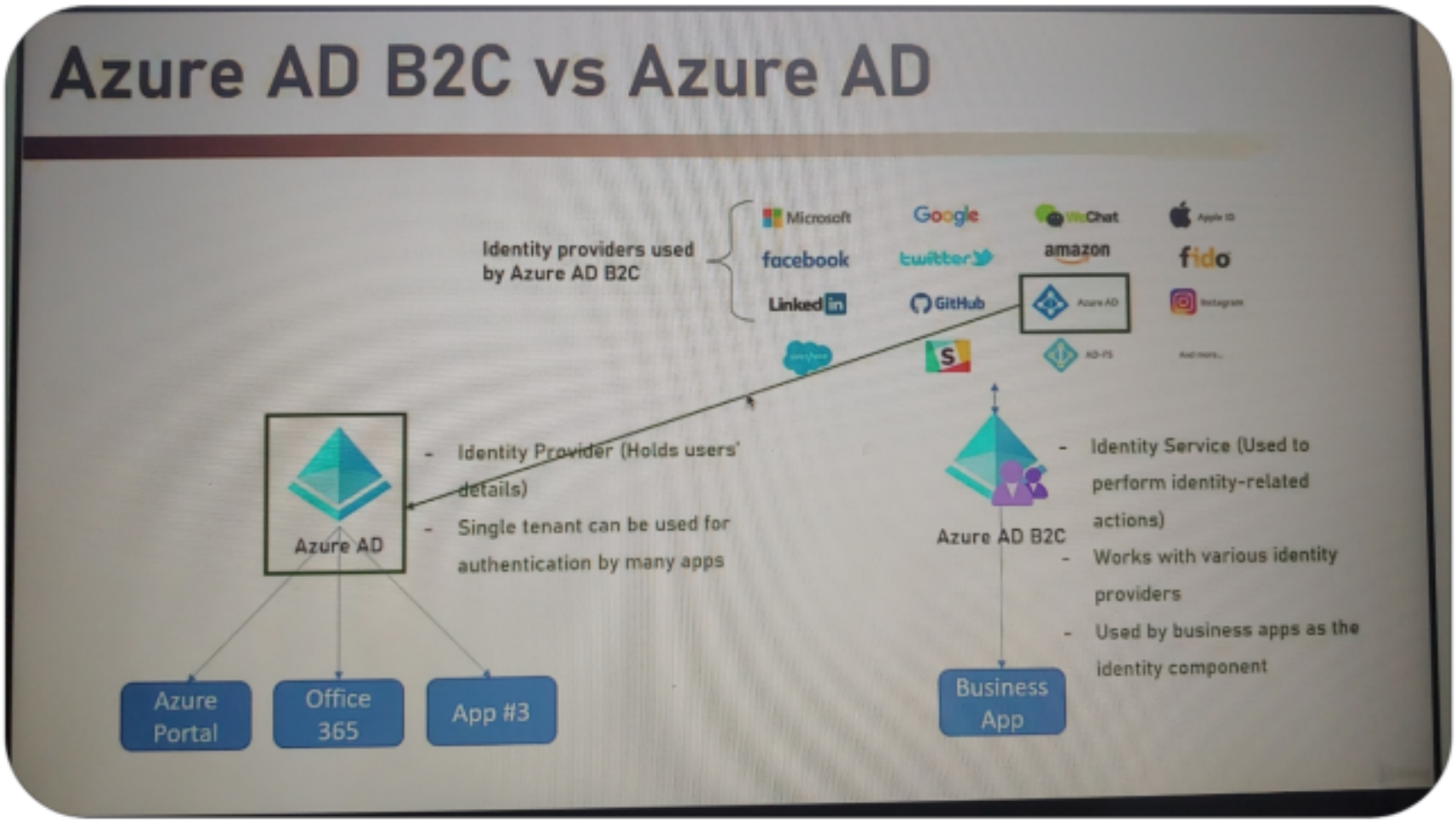
service code. Now when we try to login to App Service Azure will ask to provide valid username and password and upon successful login it opens the app.

Azure AD B2C

- ○ It is an identity service which can use various identity providers as the source of the user details and provides various identity service related functionalities.
- ○ Azure AD B2C allows to work with various identity providers where as Azure AD is one of such identity provider. It also provides various user flows and customization to this user flows.
- ○ Identity services provided by Azure AD B2C are Sign Up, Sign In, Log Out, Reset Password etc. We can implement all of these

only through some configurations no code changes are needed.

- **Azure AD vs Azure AD B2C:**



- **Authentication Features:** Various authentication features provided by B2C are MFA, Conditional Access, Audit Log, Custom Policies, Custom Pages etc

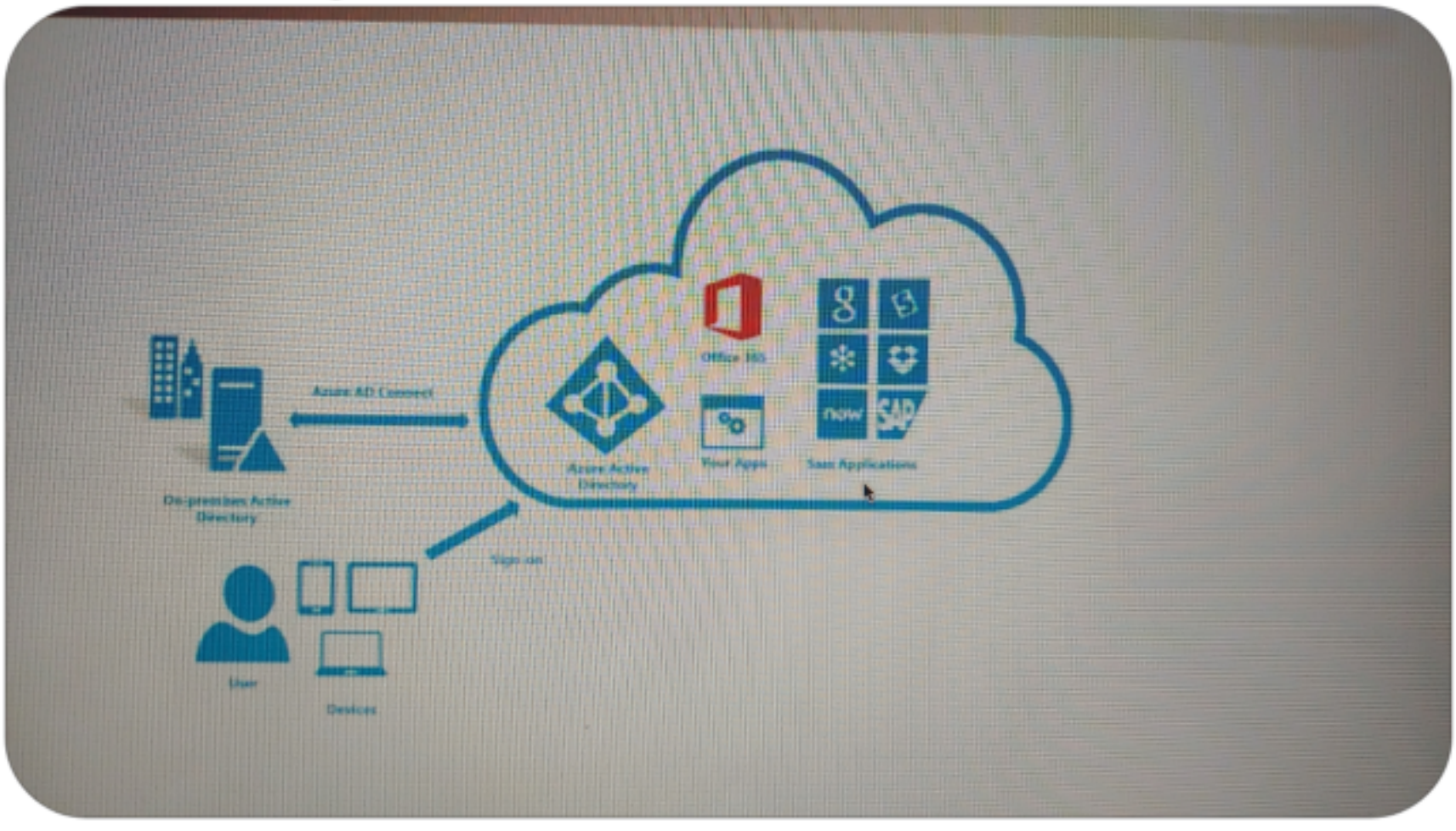
---

Syncing Azure AD with On Prem

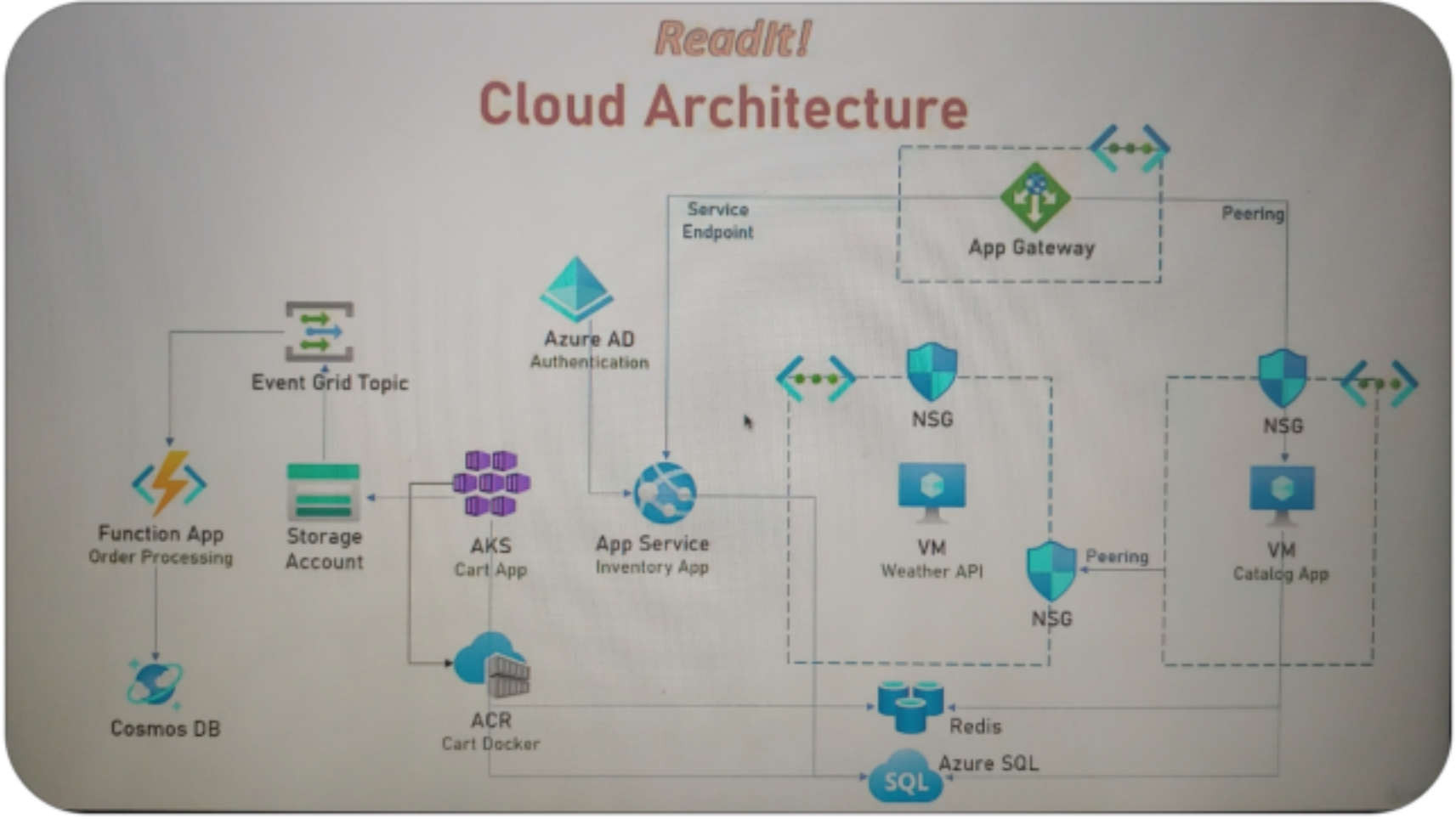
- Many organizations want yo sync their on Orem Active Directory with Azure AD. This is useful when the organization has apps

on Prem and in cloud and wants to have a single user base so that users wont have to log in twice, once in the cloud and one on Prem. And the solution for this challenge is AD Connect.

- **AD Connect:** Its basically an Agent connecting on active directory with Azure Active Directory. And this way the users data is synched between the two engines.



Current Architecture



- Here we have added the authentication using Azure AD, that is connected to the inventory app service. This addition protects the inventory app for unauthorized access by users who are not the store's employees.

==========