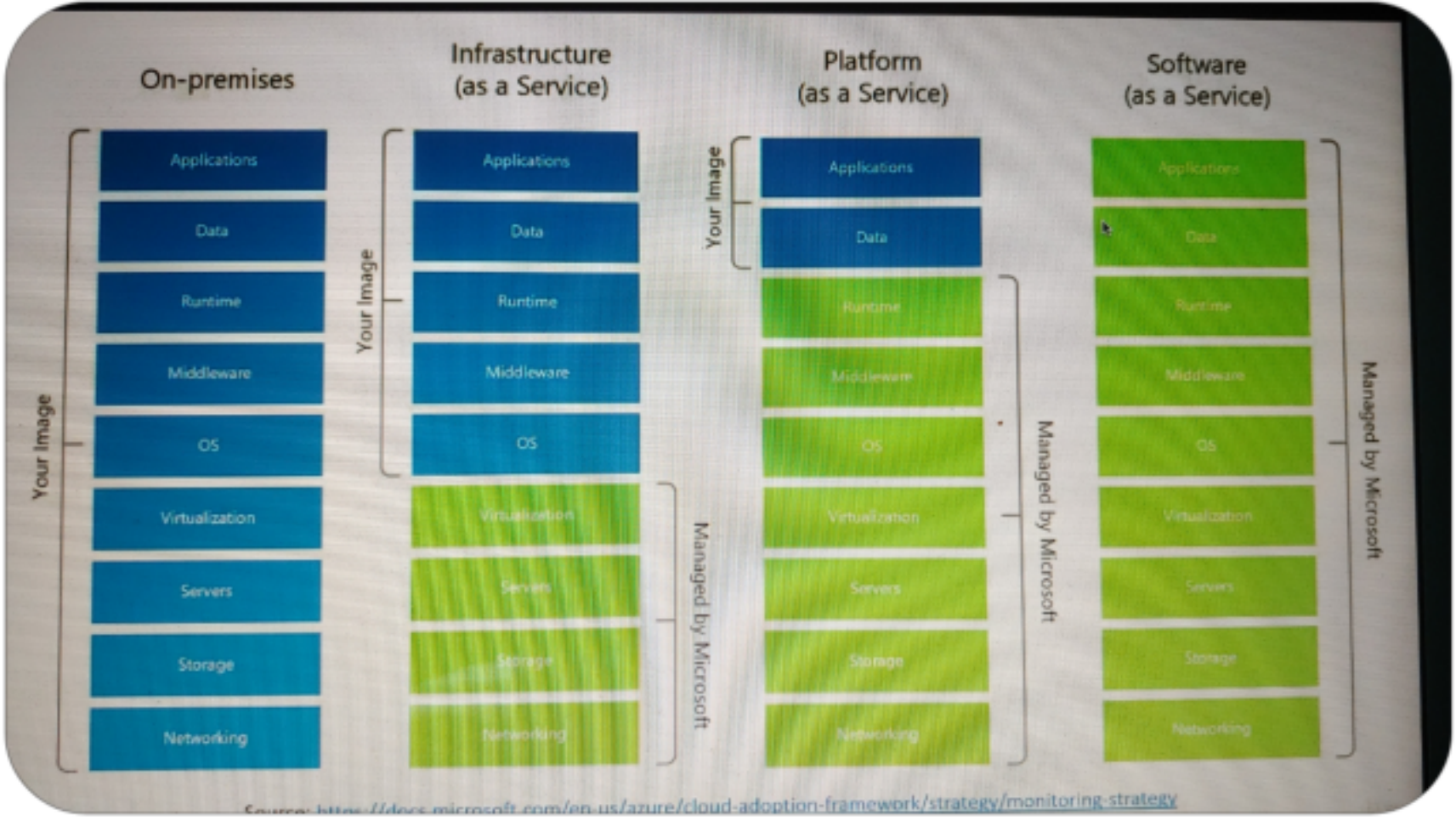
Microsoft Azure

Introduction to cloud

- **What is a cloud:** cloud provider setup huge data centers in particular locations. Then fill it with VM servers, networking, storage/ databases and other cloud services like iot, ai, kubernetes etc. And finally they make it publicly available
- If we need to use this cloud, we can setup a server in a minute, deploy our applications use and pay for the usage, shut it down of not needed. Cloud providers automatically maintains, patch updates, secures and monitors it.
- **Characteristics of the cloud:** on demand self service, broad network access, resource pooling,

rapid elasticity,measured service

- **Types of cloud services: Iaas**- Cloud provides only compute, networking and storage. ex: virtual machines. **PaaS**- Cloud provides platform for running apps including compute, network, storage, runtime environment, scaling, redundancy, security, update, maintenance etc. client just bring code to run. ex: web applications. Here client has no access to underlying virtual machines. **Saas**- Software running completely in the cloud. User does not need to install anything. User just uses these softwares. provider of the software takes care of updates patches redundancy scalability etc. Ex: Office 365, salesforce etc.



Source: https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/strategy/monitoring-strategy

- Additional service types are FaaS,DBaaS,IOTaaS,AIaaS etc
- **Types of cloud: Public**- Available for all clients and users. Clients have no access to underlying infrastructure. Ex: aws, azure, google and IBM cloud. **Private**- A cloud setup in an organization's premises and accessible only in the organization's network. Ex: vmware cloud, red hat openshift, azure stack. **Hybrid**: A cloud setup in an organization's premises but also connected to public cloud. Sensitive data are

kept in organization's premises and public data in the public cloud. Ex: Azure Arc, AWS Outposts.

==========

Introduction to Azure

- **Regions**: Microsoft built a lot of data centers for Azure. Each data centers location is callee Region. There are nearly 60 Azure regions. Some of the regions may not provide all the cloud services.
- **Zones:** Some of the regions have more than one physical data center. Great for availability in case one data center fails. Each of these physical data center is called zone. When there are more than one data center in a region, the region is said to have

Availability zones

- **Paired Regions:** Some regions have designated pair region for increased availability. When a full region fails the other one can fill its place.
- **Cloud service:** Everything that can be done in the cloud is called Cloud Service. I.e. Creating VMs, building databases, set up networks, use AI algorithms etc

---

First look at Azure

- **Azure Portal and resources:** Its a user interface to create and manage any Azure resources.
- **Subscription:** Azure subscription is a logical container that contains various resources you provision in the cloud like VMs, DBs, networks etc. Subscription can be attached to various

accounts. Ex: single subscription can be used by multiple employee accounts of an organization.

- **Account:** Azure account is an identity(I.e you) with access to the resources which are in the subscriptions. Account can be attached to multiple subscription. Ex: an cloud architect account can be attached to multiple client's subscription.
- New Account and Subscription are created and then link this account to a subscription or vice versa.
- **Resource Group**: Its a container for multiple resources. Create a resource group with a valid name. Every resource group must be linked to a subscription and must be created in a particular region. Later we can search for this resource group by its name from

search bar present in the portal. We can also click on the resource and go to its resource page where we can see various other options and use it. Also, we can delete a resource group if its not needed

- **Azure CLI or Power Shell**(Cloud Shell): provides another option to manage resources in Azure apart from Azure Portal. We can download this cloud shell to local machine and interact with azure from local machine without portal
- We can create a new resource group through cloud shell by following command: 'az group create -l westus -n my-rg'

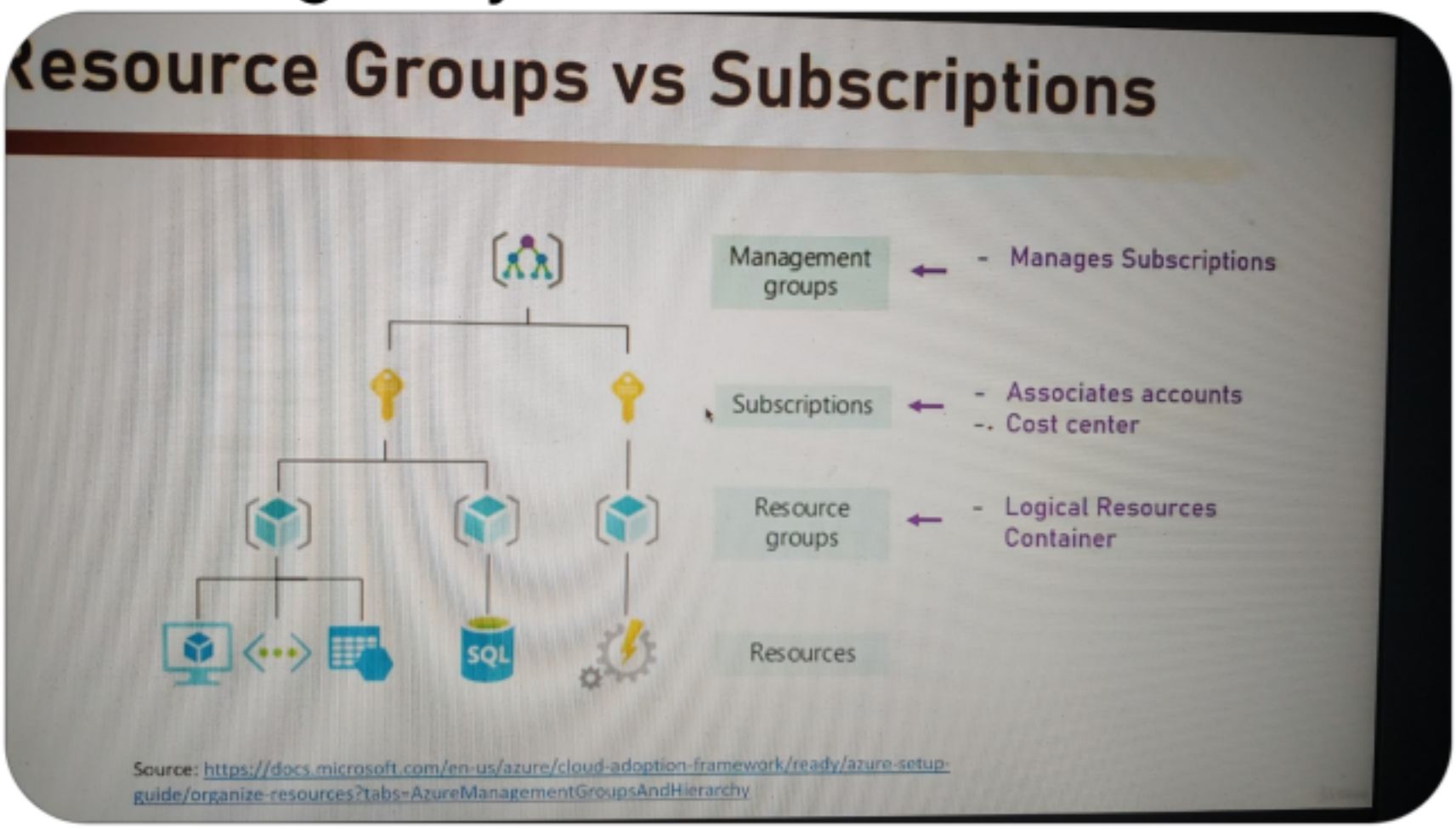
Azure basic concepts

- **Selection of a region**
- **Resource groups:** Its a logical container for resources. Used for

grouping resources by a logic boundary. Ex: Resource groups for a development, test and production resources. Another example is by separating resources by teams. Common suffix for resource group is rg/RG. Almost all resources in Azure is places in a resource group.

- **Resource group vs Subscription:** subscription is a container for account and cost centers.

Resource group is a container for logically related resources.



- **SLA:** Service level agreement on

the uptime % of a cloud service. Like 95%, 99%, 99.99(52m 35s) of downtime per year.

- **Storage Account:** Used to store almost anything in Azure.
- **Cost:** Almost everything in the cloud costs money. Few pricing models are: Per resource(i.e VM), Per consumption(I.e Function Apps), Reservations for lets say 3 years to get discounts. Always check resource cost before provisioning. Use azure price calculator tool to find out the overall price of the selected resources. Prize will vary depending on the resource type and it's configuration selected.
- **Setting Budget:** Go to budget section inside Cost management in azure portal. By default since azure provides 200$ of free resource we can set budget

maximum limit as 200$, so we will be get notified when we reach that budget limit.

- **Architects and the cloud:** Software architects designing a regular non cloud system need to know non-functional requirements, technology stack, components architecture, communication pattern. Along with the above list if an architect developing cloud based system he must have infrastructure knowledge, security and hands on
- **Azure Account & Azure App Service:** A VS Code Plugins - While working with visual studio install a plugin called Azure Account which will helps to connect to azure cloud and install Azure App service which helps in managing Azure App service through VS code.

==========

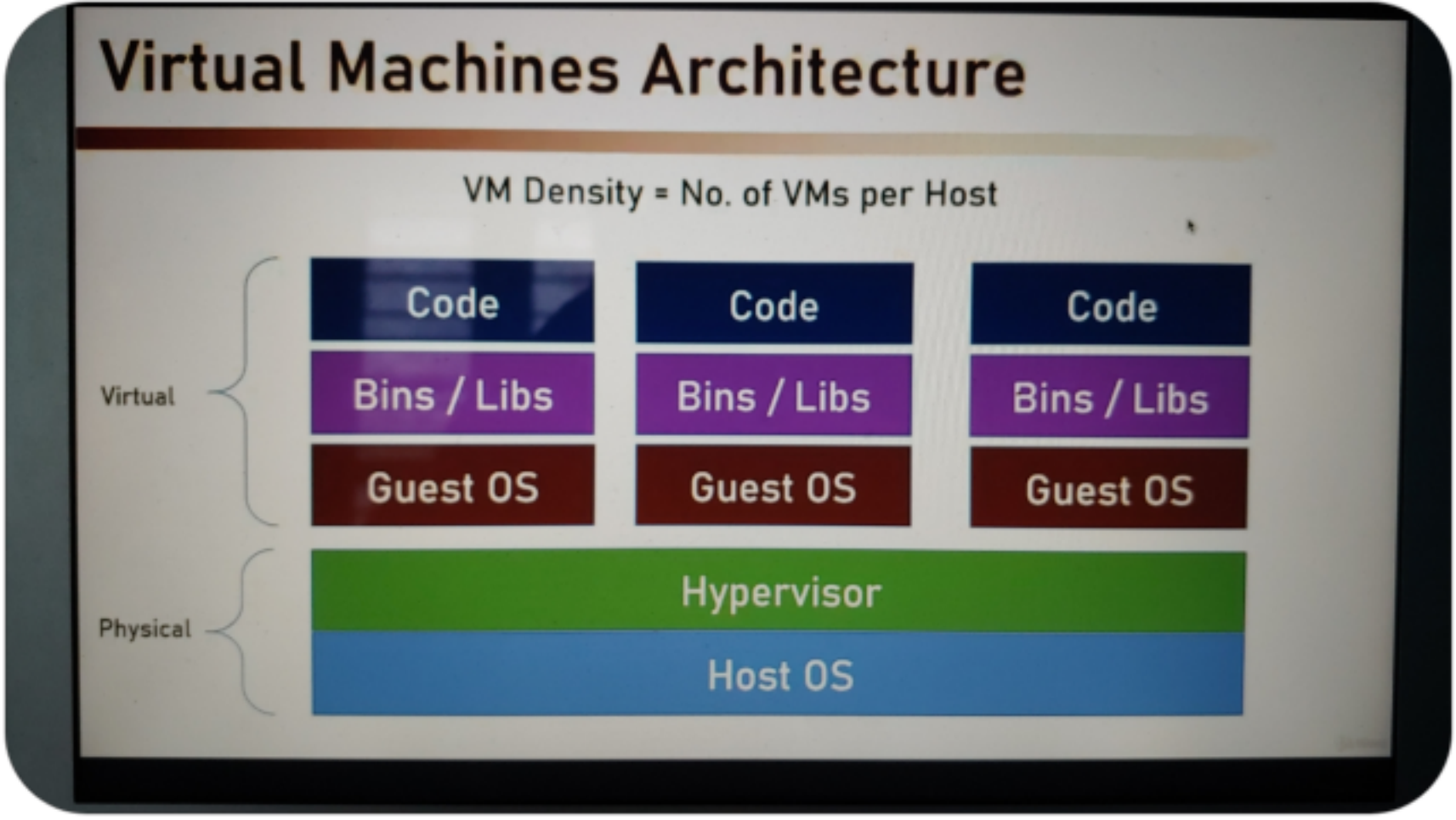
Azure Compute

- **Compute:** Its a set of cloud services for hosting and running your applications. It allows to upload your code and then running it. It offers various levels of control and flexibility.
- **Types of compute services:** Virtual machines, App Services, AKS, Azure functions

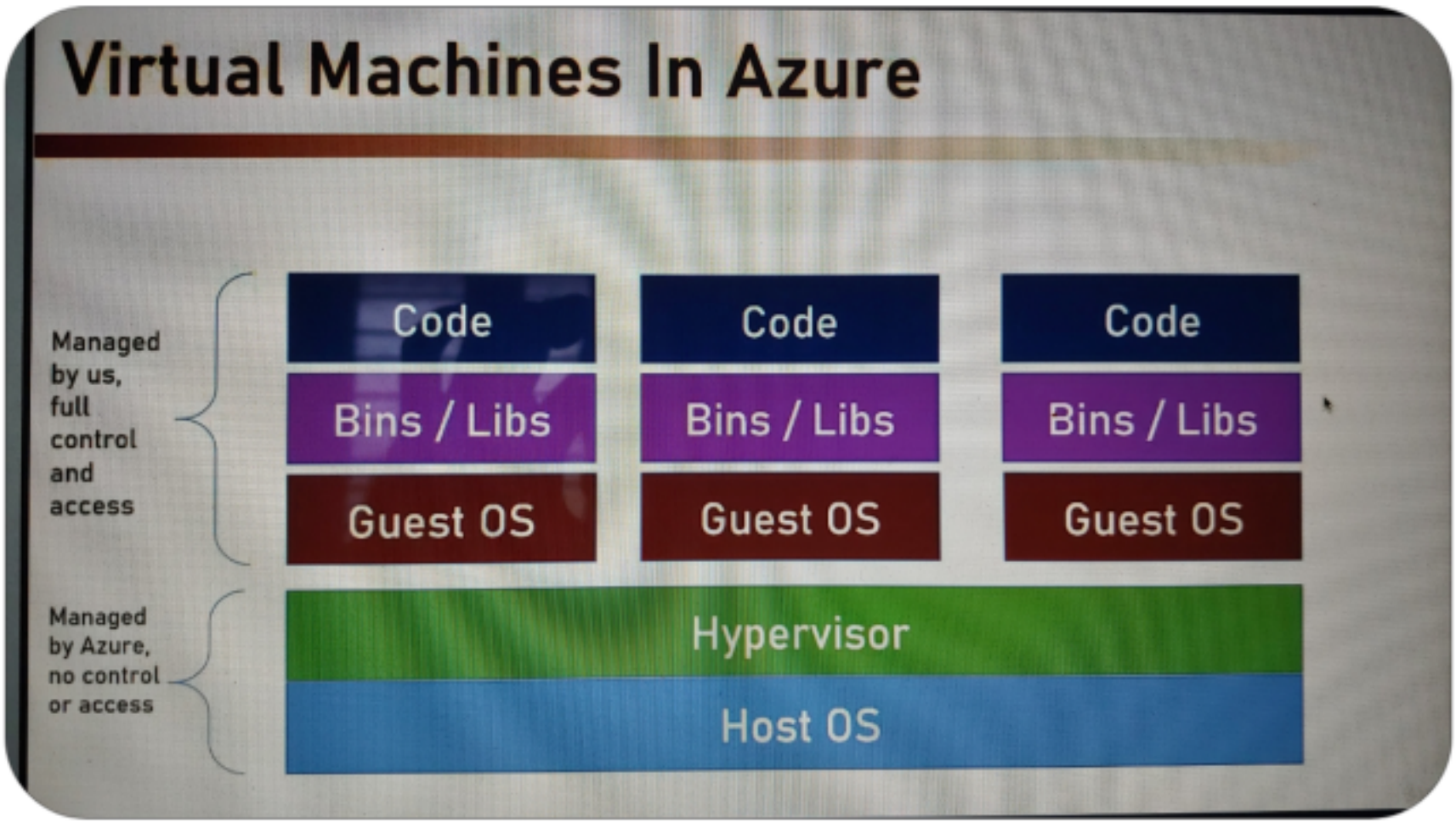
---

Virtual Machines

- Its a virtual server running on a physical real server. Allows creating new servers extremely quick.
- **Virtual Machines Architecture:**



Virtual Machines in Azure:



Steps for create/delete VM in Azure:

- Open the azure portal, select create virtual machines option and then select the region/ location, select the base

image(OS+Pre-Installed software), Select the size(CPU & Memory). Also, select the right subscription, resource group(create a new one if does not exist),public IP address and provide a name to this VM.

- Once azure creates a VM, It creates multiple resources like Virtual Machine, Virtual Network, Network interface, Network security group, public IP address and Disk.
- To delete this VM, go to the resource group and delete the entire resource group so it will delete all the resources which are created in it.

The real cost of VM

- Includes VM, Disk, Public IP, Storage Account(where image of the VM will be stored)

Reducing the cost of VM

- Auto shutdown: while creating VM select the option auto shutdown which can be used for dev or testing VM
- Reserved instances: Allow upfront payment with substantial discount which can be used for production environment
- Spot instances: Machines that run on unused capacity in Azure which can be evicted any moment when needed by azure. Great for non critical non continuous tasks
- Disk optimization: Make sure to select the right disk for your machine. Note that disk type effects SLA.
- Select right size for your machine, select linux over windows,check price in nearby regions

Availability of a VM

- SLA for VM depends on single or

multiple instances of VM in the same Availability set or two or more Availability zones. Also, it depends on the type of disk selected.

- **Availability Concepts in Azure:** **Fault domain** - similar to rack in a traditional system. If there is a problem with the power or networking in the rack all servers in it shut down. You want to make sure your servers are spread across more than one fault domain(rack). **Update domain** - if all your servers are in the same update domain they will reboot at the same time during maintenance. **Availability Set** - A collection of fault domains and update domains your VMs will be spread across. So azure will create 1 instance of VM in one set of Fault domain and Update

domain. And another instance of VM will be created in a separate fault and update domain. Hence always deploy identical VMs into the same Availability set which ensures they won't shut down simultaneously when a single fault domain shits down or an update domain reboots.

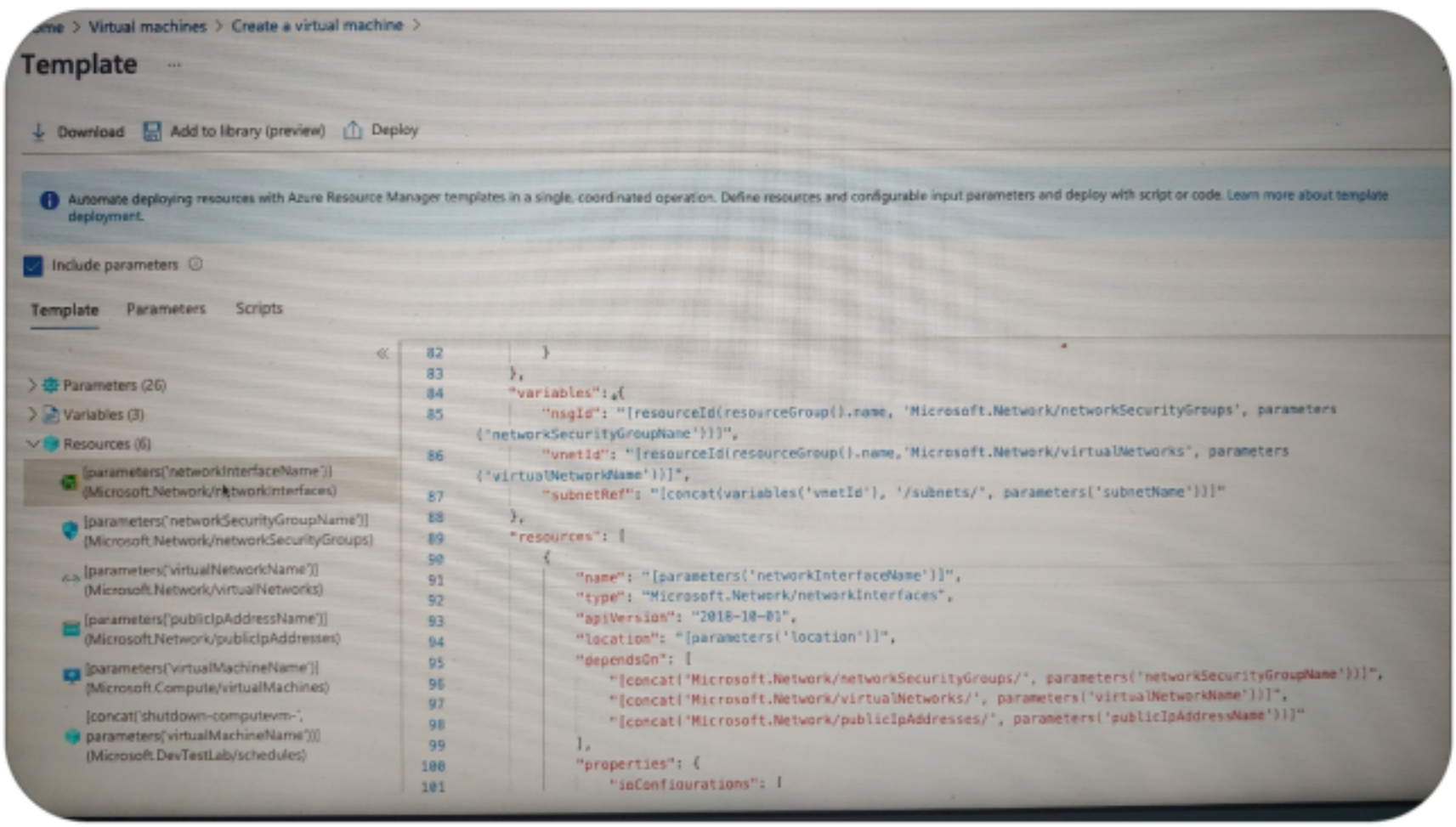
Availability zone - its a physically separate zone within an azure region. Now multiple instances of same VM will spread across multiple Availability zones of a region. Each zones functions as a fault and update domain. Availability zones provides better SLA. Hence, deploy identical VMs into separate Availability zones in the same region. Availability Set and Availability zones are free. We only pay for additional VMs.

ARM Template:

- ○ Azure resource manager template. Its a JSON file representing a VM and describing the resources to be created. Its used by Azure in almost all deployments. It can be exported modified uploaded and deployed. It can be exported from portal while creating a nee VM or can be created from scratch.
- ○ ARM template is a declarative way of deploying resources against the imperative approach.
- ○ Declarative process: It describes the end result. Can deploy multiple resources at once. Can be integrated in CI/CD processes. Can be source controlled.
- ○ Imperative approach: Sends instruction to run. Does not represent end result. Its error prone. Cant be verified. Cant be source controlled as its not a file.

Suited for quick and dirty operations.

- ARM templates uses declarative approach where as Azure CLI, PowerShell uses imperative approach(although they can uses ARM template too)
- ARM template basically contains list of parameters, variables and resources.



- We can download this ARM template which basically contains 2 json files. One is template.json which has list of resource details with parameter

names and another one is parameter.json which contains values of those parameters.

- Edit these json files with updated required values. For example change osDiskType from PremiumSSD to StandardSSD. And then copy these files into newly created folder 'templates' in azure CLI storage accounts and then deploy these files through azure cloud CLI through azure deployment create command. Open azure bash and cd to clouddrive/templates and then type the command: 'az deployment group create --resource-group optimized-vm-rg --template-files template.json --parameters parameters.json'

Virtual machine scale set(VMSS):

- A group of separate VMs sharing the same image. Can be scaled

out or in manually or automatically according to predefined conditions.Great for handling unpredictable load.

- ○ Once set up, the machines should not be modified by changing files, installing apps etc. This is because new machines created by the scale set will be always based on the original image. So, all the changed files, installed apps will be lost. Always changes must be done on the image instead of running VMs
- ○ For web apps, a load balancer should be put in front of the scale set. Load balancer exposes single address to users and it routes all incoming requests to Virtual Machine Scale Set which contains actual multiple instances of VMs created from same image.

- Scale Set resources are free. But you pay for the VMs deployed in it

Using Scale Set

- Virtual machine scale set can be created from azure portal create virtual machine scale set option. By default 2 instances of the VM will be created.
- There are 2 methods for scaling in VMSS. Manual and custom. In custom scale set we add a policy or rule set to indicate when should the scale out or in should be triggered. For ex: Add a new instance of VM when average CPU is greater than 80% and its been more than 80% for 10 minutes. And scale set can increases such VM counts until it hit the maximum configured limit. **Cool down** period is the time interval for which scale set will wait for the next scaling

operation.

- **Scale Out** is the process of adding new VM by scale set and **Scale In** is the process of removing existing VM by Scale Set. We can define separate rule for these two activities. For example we can define a rule for scale in such that scale set will remove an existing instance of VM when average CPU is greater than 30% and its been more than 30% for 10 minutes.
- **Scale-In policy**: There are multiple option to scale in like NewestVM- Balance across availability zones, then delete the newest created VM. OldestVM-Balance across the Availability zones and then delete the oldest VM etc
- In order to create this scale in or scale out rules in scale set we

must need to register a special resource provider called '**microsoft-insights**' into our subscription.

Azure instance metadata services

- ○ Its a REST API accessible from within a VM. It provides a lot of info about the VM like storage, networking, scheduled events etc
- ○ With Scale set VM get notification about the upcoming eviction of that VM. And if we are listening to the metadata services of that VM, we can know this VM is going to be removed in next few minutes so that we can run any final action before this VM evicted. This metadata can be pulled every 1 min to get enough time to close things up.
- ○ We can connect to VM from azure portal. Go to VM connect link and click on 'Download KDP

file' which will take us inside remote windows machine. Here we can install postman and call those metadata rest service APIs

Virtual machine tips and tricks

- Want to backup your VM so that it can be restored in case of failure? Check the Backup page in the portal for a VM and define the frequency of the Backup and the retention period.
- You can define the DNS name for the VM so that it will be accessible not just using its IP. This can be done by clicking the DNS Name: configure link in the overview page of a VM in portal.

Azure architecture diagrams

- When designing architecture for Azure apps its a good idea to use Azure symbols in the diagram. We can download all those azure symbols/icons from the azure

website to local system.

Practical

- **Catalog App**: Create one VM through Azure portal. Select the OS as Windows. Login to VM through KDP file. Then Install the runtime environment either .net/ java jdk. Install the required App server like tomcat. Deploy the application to App server. Access the app through public IP of VM.
- **Weather App:** Create another VM in Azure through portal. Select the OS as Ubuntu. Login to VM through KDP file and Install the nodejs server. Install git and clone the weather app. Then run npm install and start commands to run the app. Access the app through public IP of VM.

App Services

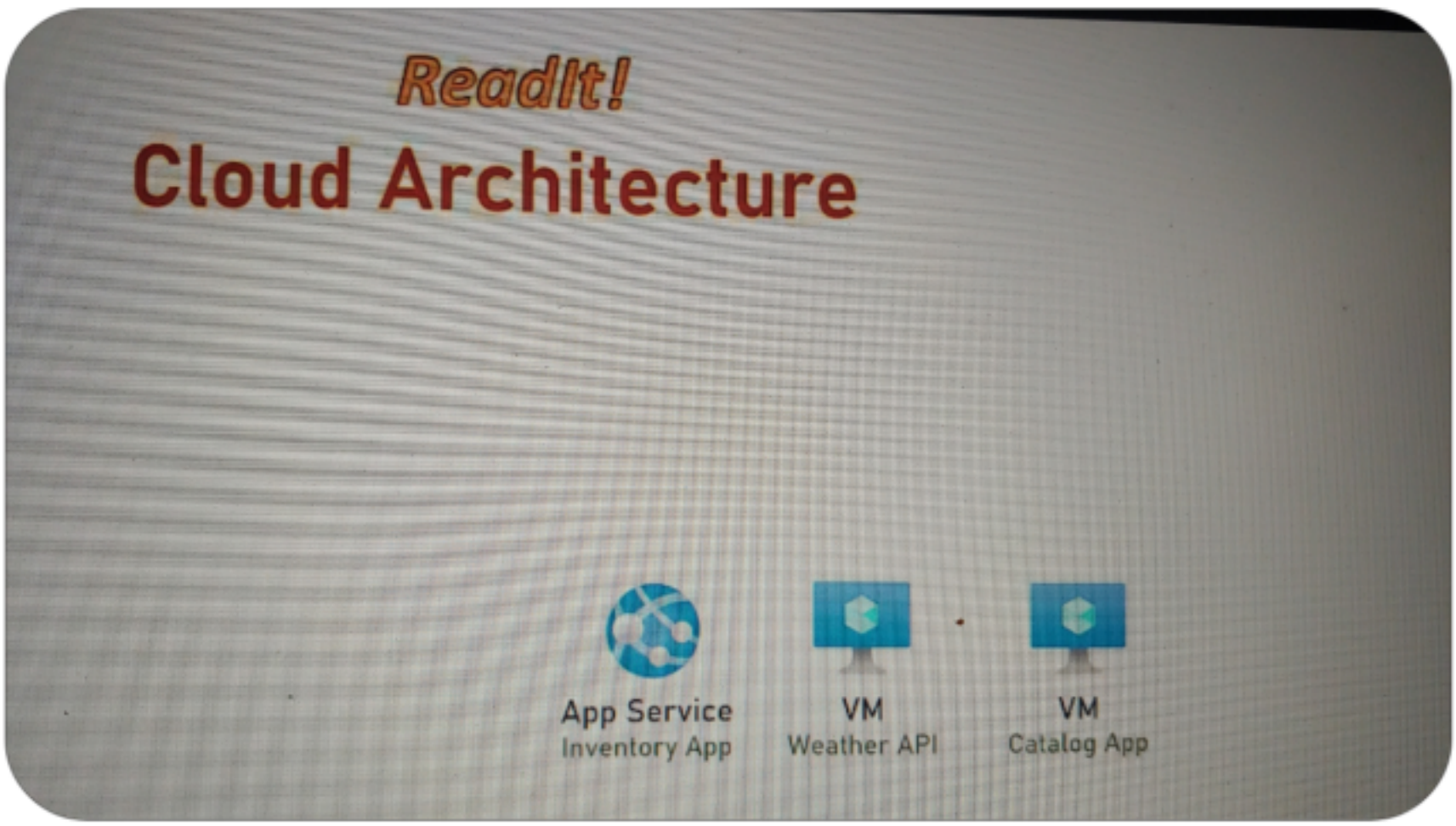
- ○ Its a fully managed web hosting for web services. You just publish your code and it just runs. You don't need to install any VM and maintain it. It is managed, scaled, patched and updated automatically by azure and you don't need access infrastructure
- ○ It can be integrated with many source control and DevOps engines. For ex we can push a new version of the app to Github and then App Services will pull this latest version of the app and runs it. We can also publish containers which contains the application, then App services can run such containers also.
- ○ App services can support Apps of type Web apps, Web API, Web jobs(batch processes)
- ○ App services are extremely easy to deploy. First develop your app

then create web app which can be done from the IFR and finally publish your code.

- ○ App services can be auto scaled to support spikes in load. With VM you pay only when the VM is on. But, with App Service the only way to stop paying for it is to completely delete it.
- ○ App services provides different app service tiers to choose from, where each tier has specific functionality and cost
- ○ Practical: (**Inventory App**) Create a new App service in azure through portal. While creating select publish option as either code or docker container, and select the right runtime stack(.net, java). Now in the IDE add the extension 'Azure App Service'. This extension provides an option 'Deploy the web App" in

the IDE. We can click that and publish the application directly to the created App Services in Azure by selecting the right App service name. Finally we can access this app through domain name/ url of this app service. We can also config scale in or out.

- **Current Architecture:**



Azure Kubernetes services(AKS)

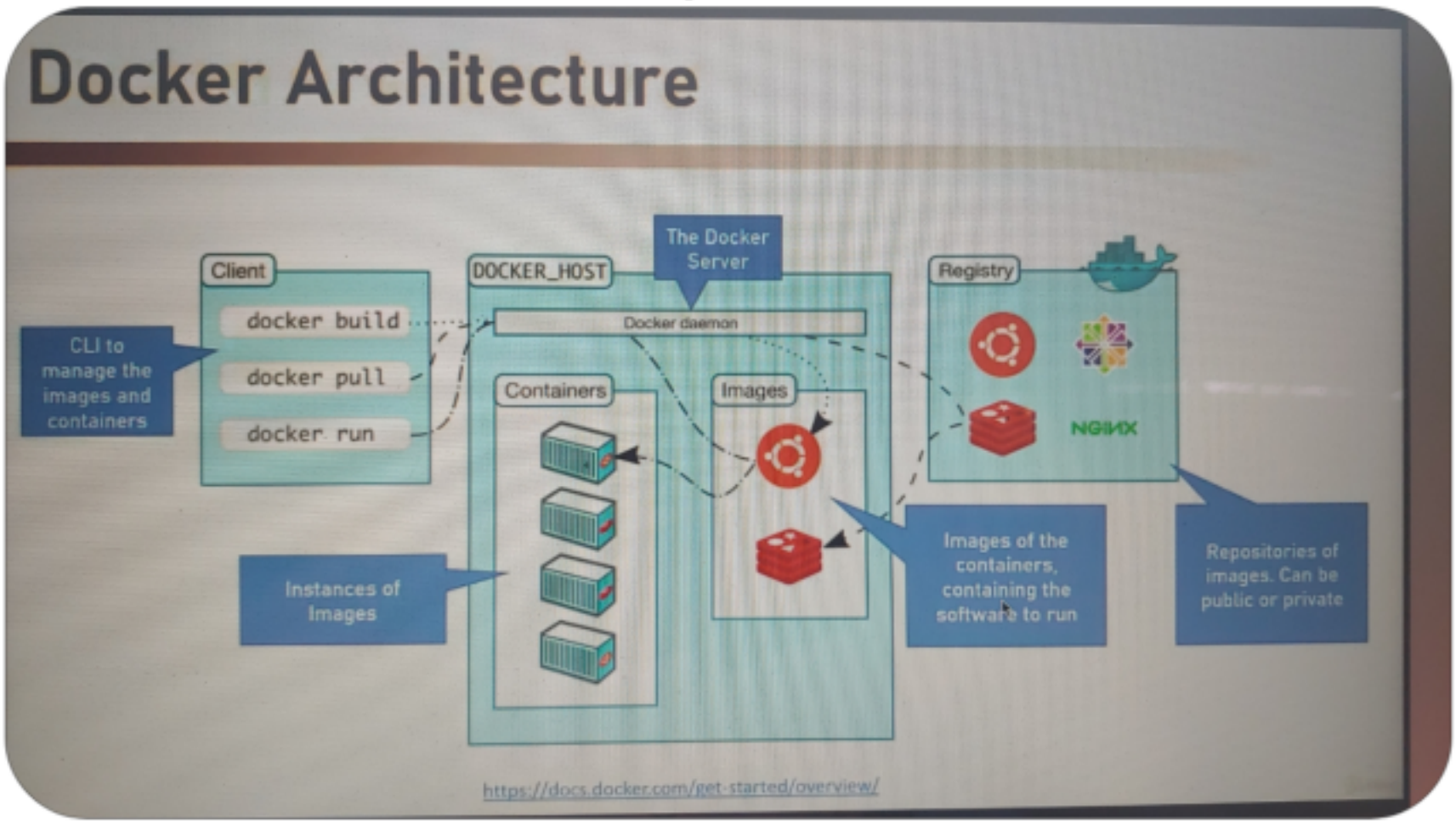
- Its a managed kubernetes on Azure. It allows deploying containers and managing them

using kubernetes on azure. No extra payment for the k8s. Pay only on the instances/VMs used.

- **Containers**: Before containers there was a problem then code works in developers machine but not in the production. Containers are mainly helped to solve such issue. Containers are the thin packaging model. It packages software, its dependencies, runtime environment and configuration files. It can be copied between machines. Container uses its underlying OS.
- **Why Containers**: Predictability - The same package is deployed from the dev machine to the test to production. Performance - containers goes up in seconds vs minutes in VM. Since containers uses host OS it no need to boot OS again. Density - One server

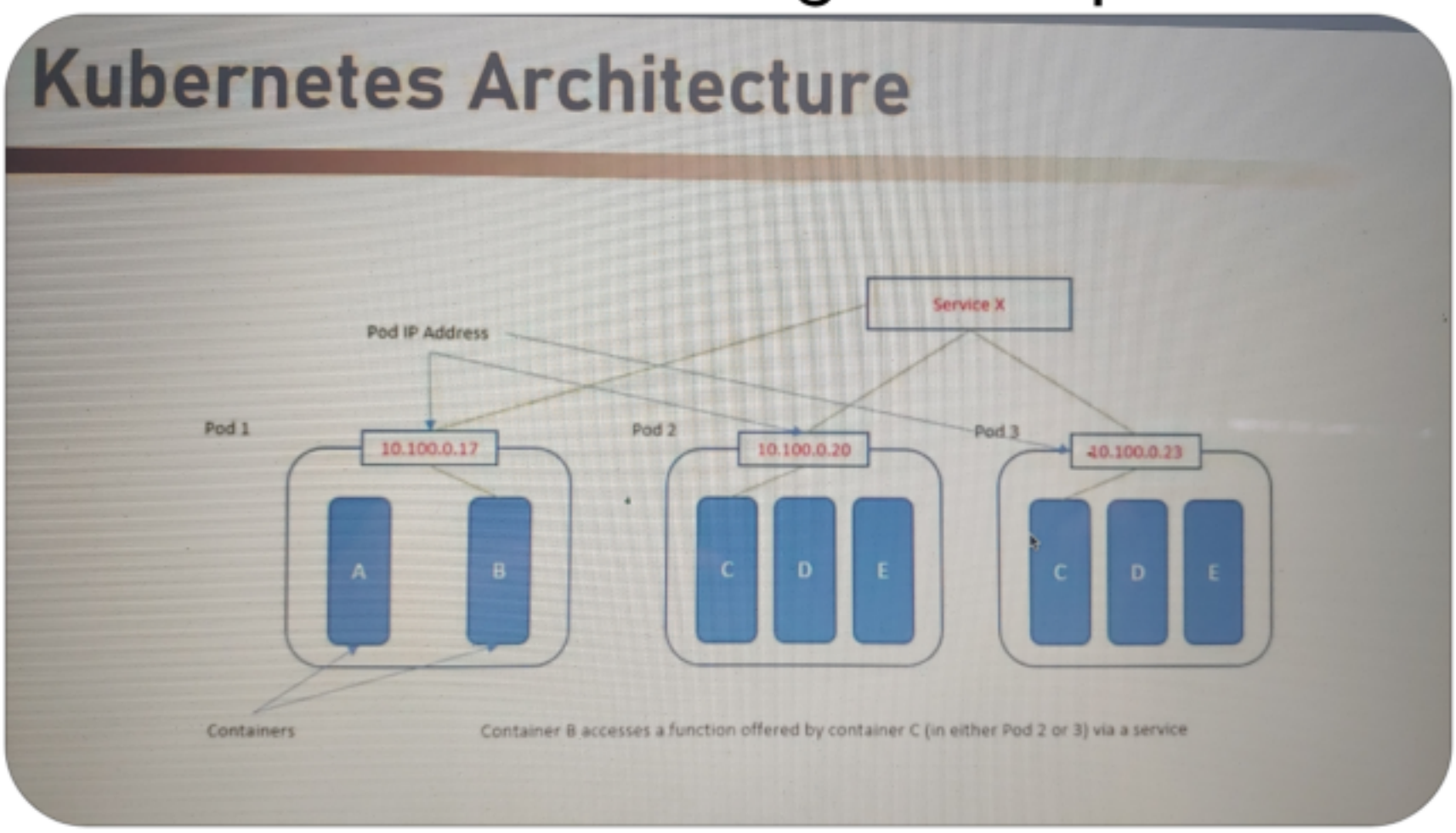
can run thousands of containers vs dozens of VM.

- Why not containers: Isolation - Containers share the same OS so isolation is lighter than VM.
- **Docker**: Is the most popular container implementation.



- **Dockerfile**: Contains instruction for building custom images.
- Docker is supported by all major OS and major cloud providers. Also, major cloud providers has their own docker registry like Amazon ECR, Azure ACR.

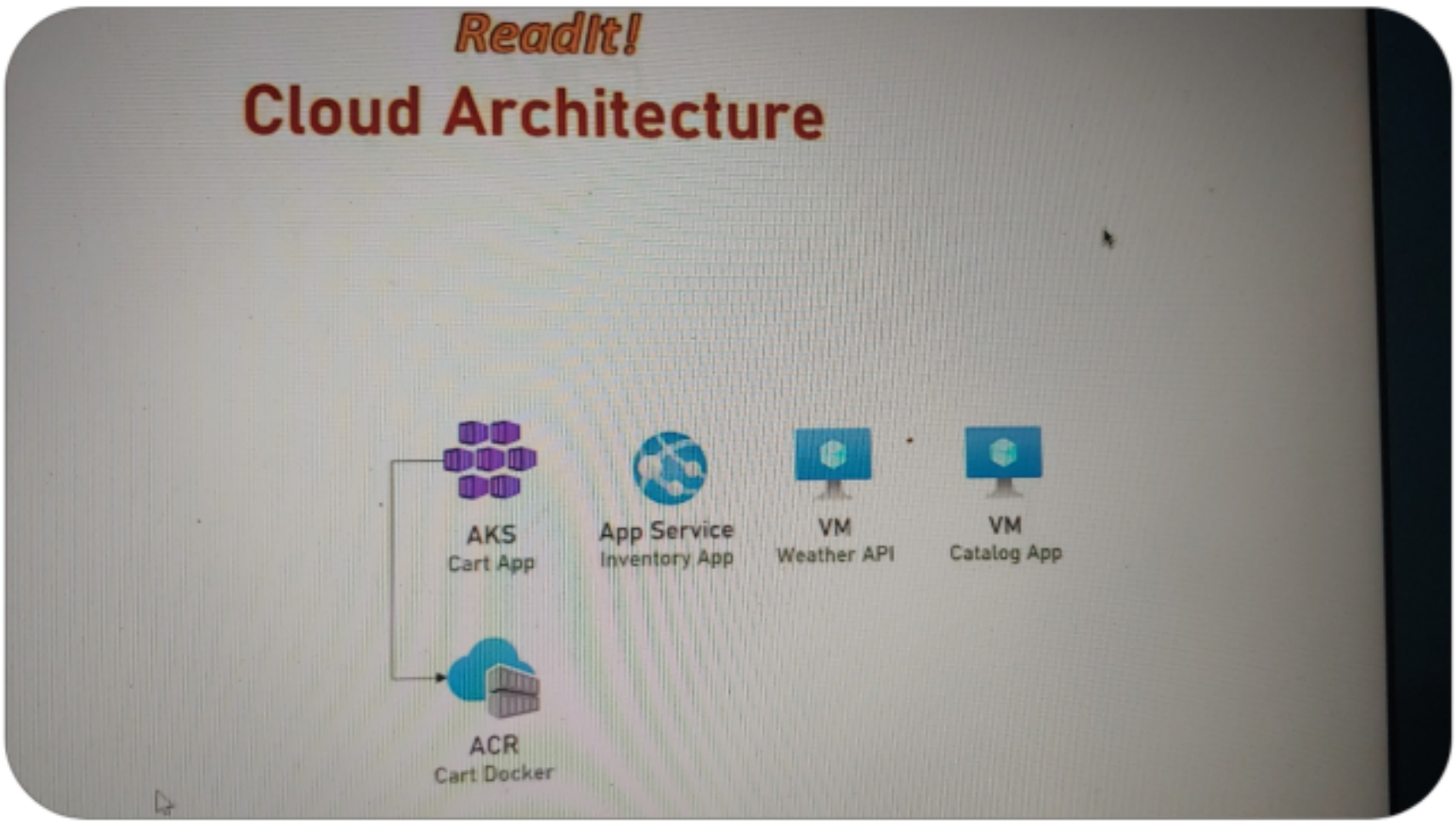
- **Containers management**: what happens when there are thousands of containers deployed in a cloud. How can we manage deployment, configuration, scalability, monitoring, routing, high availability.
- **Kubernetes**: Most popular container management platform.



- **Practical: (Shopping Cart App with Cart docker ACR)** Create a java/.net project, include Dockerfile in the project or use any third party plugin, Build an

docker image from the Dockerfile, Run the docker image to create docker container and test the application. Then push the created docker image to remote docker registry like Azure Container Registry(ACR) from VS Code using docker extension. Install the Azure CLI & AKS CLI in local machine. Through Azure CLI connect to azure cloud and run a command to create AKS cluster with single node. Create K8S resources yml file in the project. Then through AKS CLI apply these K8S resource files to cluster. In these yml files provide the image details of the application which was pushed to ACR previously. Now through K8S services external IP we can access the application which is running in AKS pods.

Current Architecture



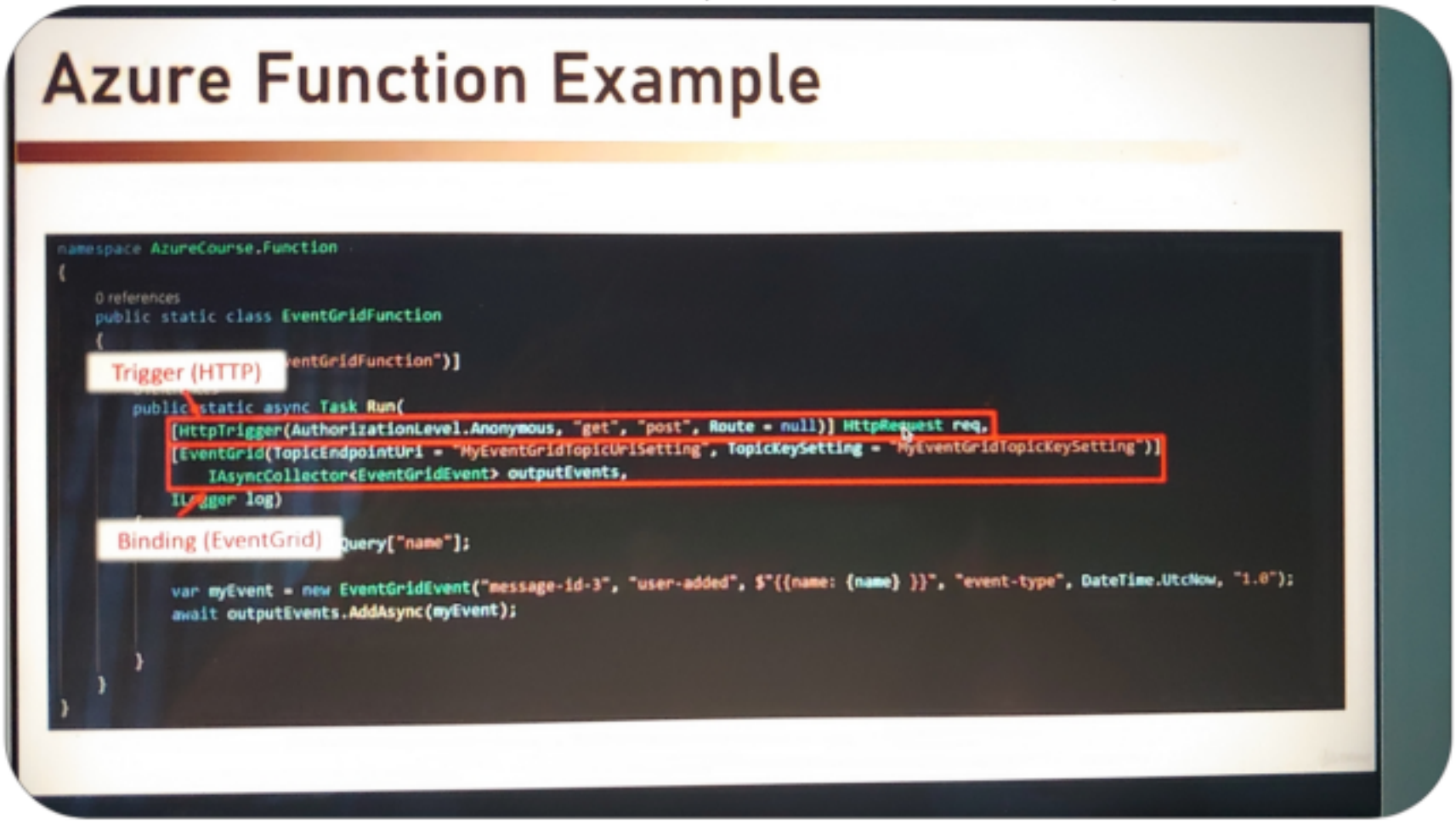
Azure Functions

- Its small, focused functions running as a result of an event. Great for event driven systems. Azure functions are automatically managed by azure to start, stop and to auto scale. These functions has flexible pricing plans. Also these functions are Serverless.
- **Serverless**: its a cloud resource that is managed completely by

the cloud. Users do not need to think about VMs, CPU, Memory.

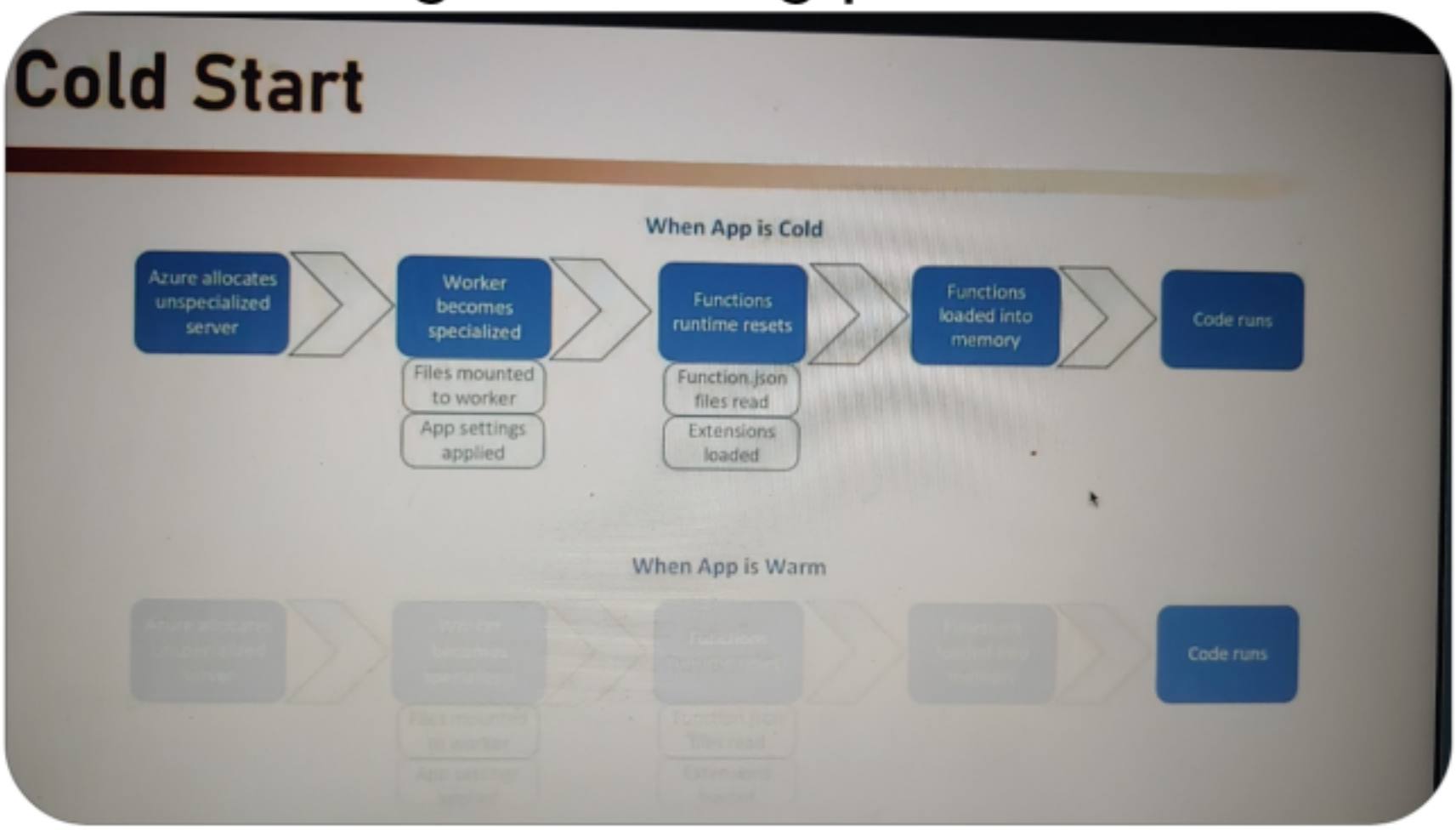
- Usually azure functions are consists of triggers and bindings
- **Triggers**: Its an event that made the azure functions to run. Azure provides such a list of events for which we can listen. There are many types of triggers which can trigger a function.
- **Bindings**: Declarative connection to other resources. Using bindings you can setup a connection between an azure function and another resource in the cloud so that you can read or write value to this resources whether as input, output or both. These bindings are provided as a parameter to the function.
- Trigger and Binding example: A function that run every 5 minutes (Timer Trigger) and calculate the

sum of a column in DB. If its above 115, send an event in EventGrid(Binding). Another example when a message arrives in the Orders Queue (Queue Trigger) save it in Cosmos DB(Binding) for future handling. Last example Receive HTTP request(HTTP Trigger) with 4 numbers, and return the smallest one of them (No Binding).



- Azure function support almost all languages.
- **Cold start:** Azure functions are completely managed by Azure.

After sometime of inactivity azure might take down the function's host. The next activation of the function will take time. So this will take 2-3 seconds before the code runs. A problem mainly for HTTP-Triggered functions. How to avoid cold start? By selecting the right hosting plan.



≡ ○ Azure functions hosting plan: Consumption plan - Pay only for what you actually use (function execution time). Premium plan - Pay for pre-warmed instances. No cold start and no memory limit in

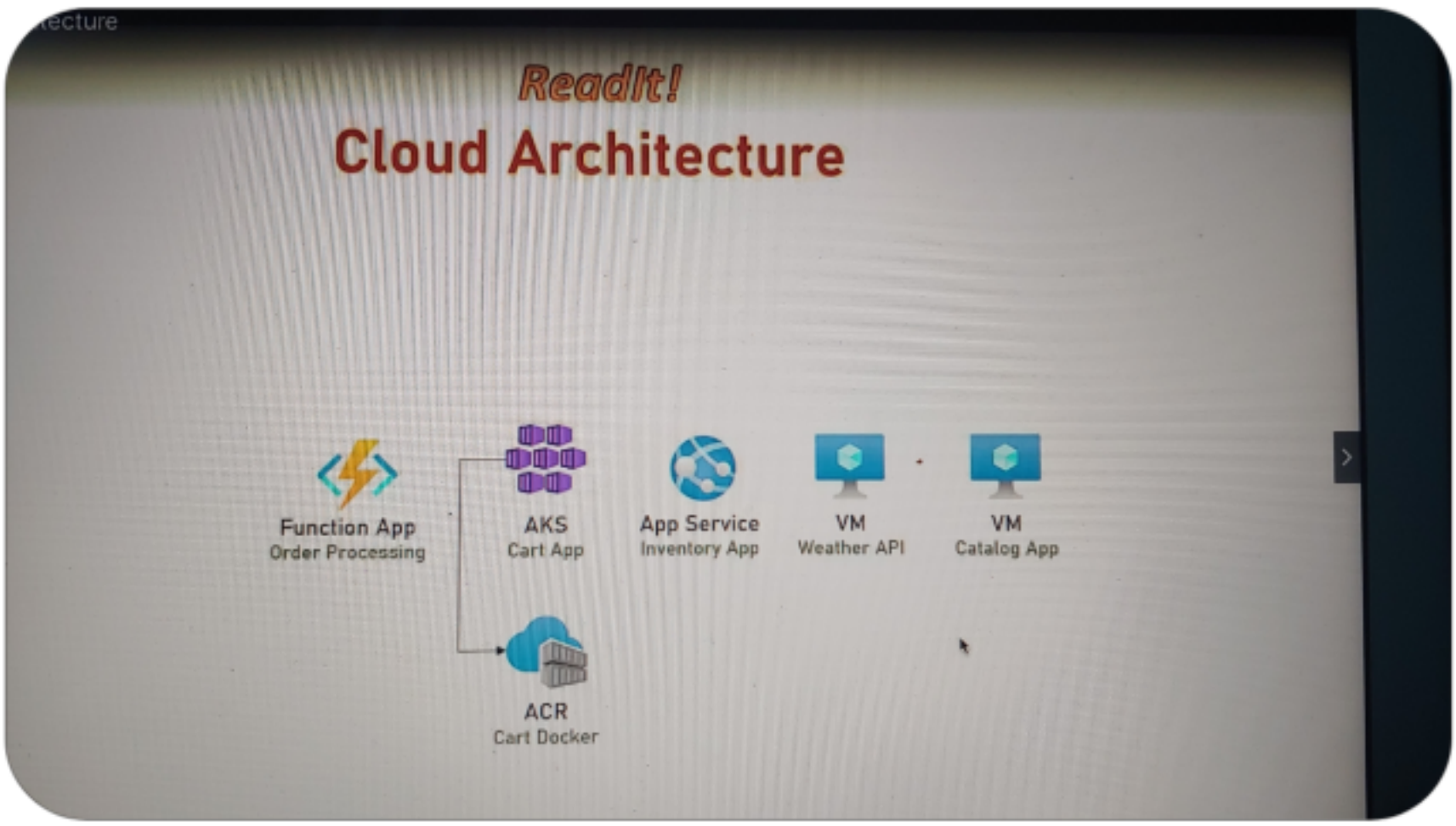
this plan. Dedicated plan - The functions run on an existing App Service. Great of server is under utilized. No additional cost. No auto scale in this plan.

- **Durable functions**: Its a state full functions that interact with external resources and keep track of flow. For example: Function chaining - call various functions sequentially and apply the output of each function to next one.

- **Practical: (Order Processing Function)** Local Testing - Download and install Azure Functions Core Tool to local machine. Write the Azure functions using corresponding language specific azure function library. Install a 'azure function' extension in IDE. Run the function and test it. If its a Http Trigger function then call the http rest api

from postman by providing the json value. Running functions on Azure - Open cloud shell in portal and type the command 'az storage account create --name readitfunctionstorage --location westeurope --resource-group readit-app-rg --sku Standard_LRS' and enter. Now create functionapp by command 'az functionapp creare --name readitfunctionapp --storage-account readitfunctionstorage --consumption-plan-location westeurope --resource-group readit-app-rg --functions-version 3'. Now upload our functions to this Function App. Using Azure function extension deploy it. Once its deployed copy the function url and test it from local postman.

- **Current Architecture:**



- **Other compute options**: Logic apps, ACI-Azure Container Instance, App Service Container-Deploy docker to App Service.

==========

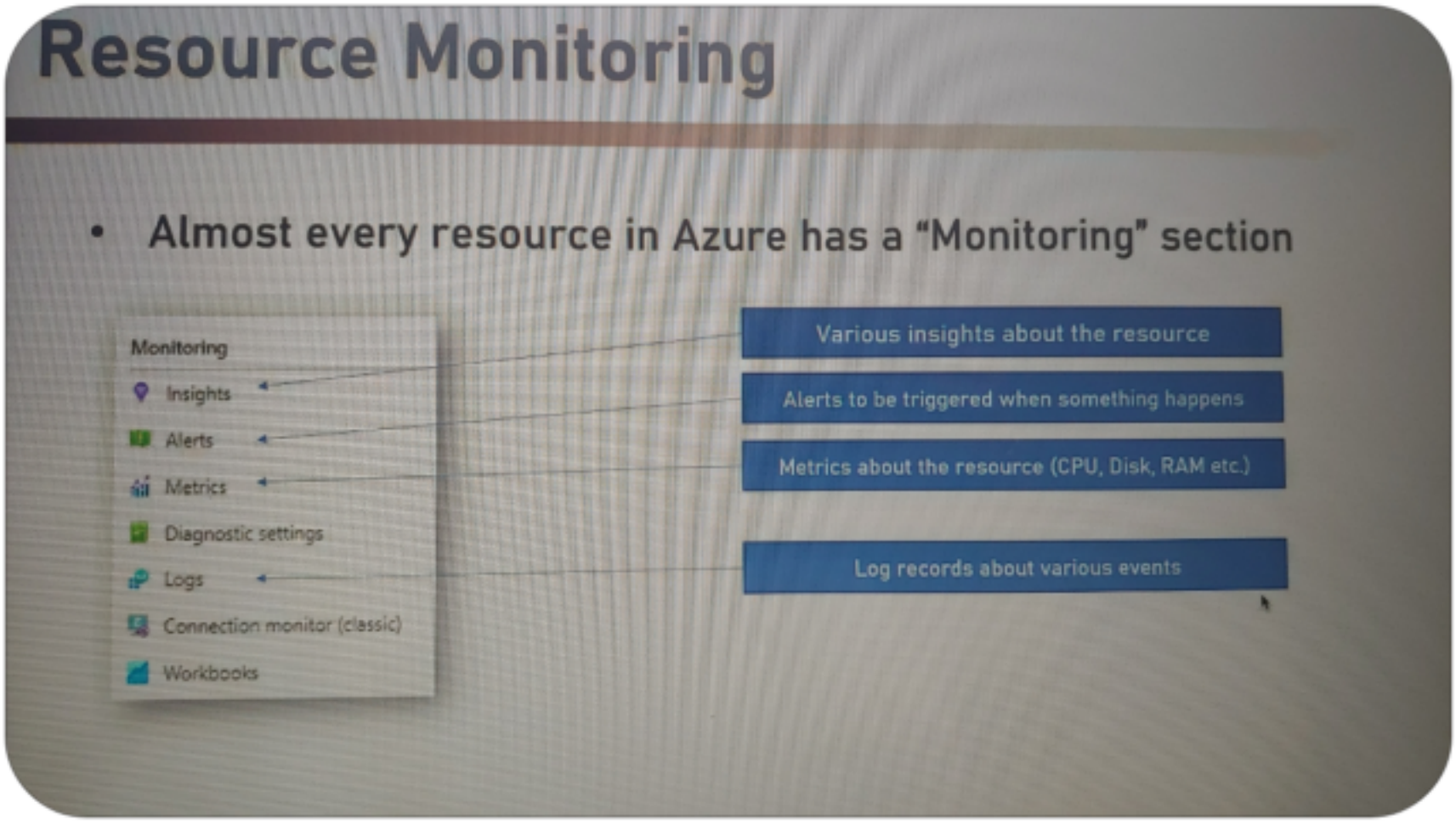
<u>Monitoring in Azure</u>
(Continued from other Doc)

- A working App is not enough. We must know its status, how it performs, and when it has problems. And this is done via

Monitoring.

- Azure offers a lot of built in monitoring mechanisms. It also provides a centralized monitoring hub where all monitoring data is streamed and can be queried or used for triggers. Monitoring in Azure is extremely cost effective.
- **Types of data for Monitoring:** Monitoring is based on two types of data one is Metrics and other one is Logs.
- **Metrics:** These are the numeric values describing resource's aspect at a specific point in time. Examples: CPU, Disk, Response time etc
- **Logs**: These are the events that occurred in the system. These can be textual & numeric. For example System Started, Error Occurred.
- **Resource Monitoring:** Almost all

resource in Azure has a 'Monitoring' section.



Metrics in Portal

- Now go to Portal and Select any VM resource we created. Under the Monitoring section we can see Metrics option where we can visualize different kind of metrics data in graph and also configure them differently. If we go to any API Gateway resource, we will be having some other unique metric like back end response time

which we can analyze further. Also, we can apply filter and group to select a specific back end pool and see response time. Also, we can apply splitting to see different back end pool metrics in one graph.

Azure Dashboard

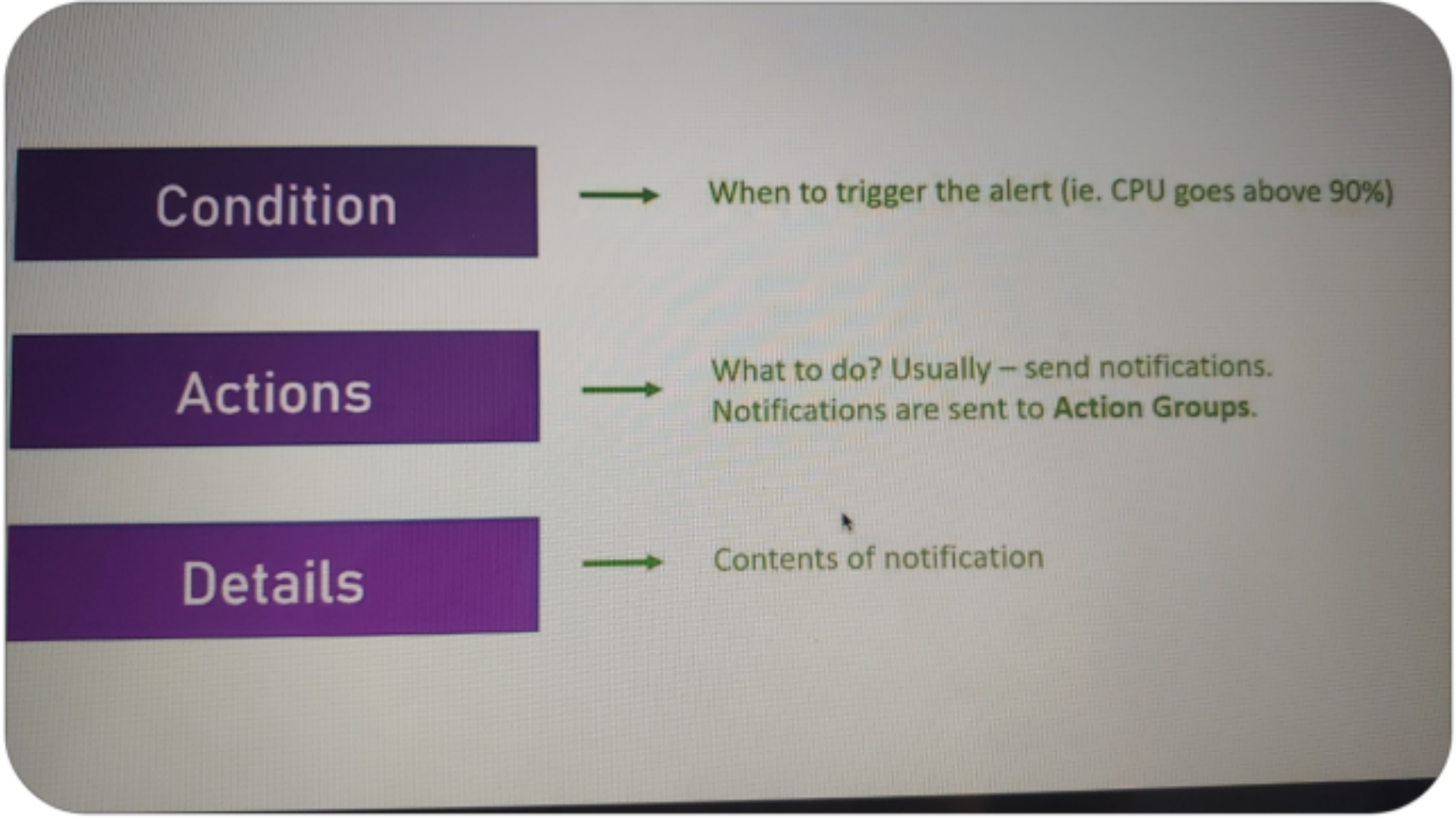
- Metrics are not saved per resource. So whatever customized graph we prepare in metrics it will not be saved. But, Metrics are saved in dashboard. So we can click the button 'Pin to dashboard' in the portal and save it with a name given to it.
- Now in the main menu, go to dashboard option where we can see these saved graphs. Also, we can customize these dashboard by adding various widgets. There

are plenty of widgets to show our list resource groups, list of resource, list of users in AD etc

---

Alerts

- With Alerts we can get real time notifications about some critical events which are triggered in our resources. For example we can get an alert when VM's CPU goes above 90% or more than 20% of requests fails in your web apps...
- **Alerts Components:** In alert definition we need to define 3 components.



- Alerts pricing are depends on per metrics measured or per notification type. It will be available for free up to some limit

Alerts in Action

- Go to a VM's monitoring section in Portal. Select Metrics option and then select 'Percentage CPU' metric from drop down. Now click on 'New alert rule' link. Here we can define a alert definition for this selected metric.
- Select the scope of this alert I.e VM name. Next write the specific Condition for which this alert

needs to be triggered. Since we selected CPU percentage metric this condition can be like whenever the avg % of CPU is greater than 90.

- Next define what action we need to trigger. For example we can send a notification to Action Group by creating a new Action Group. Action group is the collection of destination where the alert needs to be send. In this action group define what kind of notification we need to send and the email address to which this notification need to send.
- In this action group we can also define any action which we need to trigger along with sending notification email. For example we can run a Azure function.
- Finally add Alert rule details by providing alert rule name,

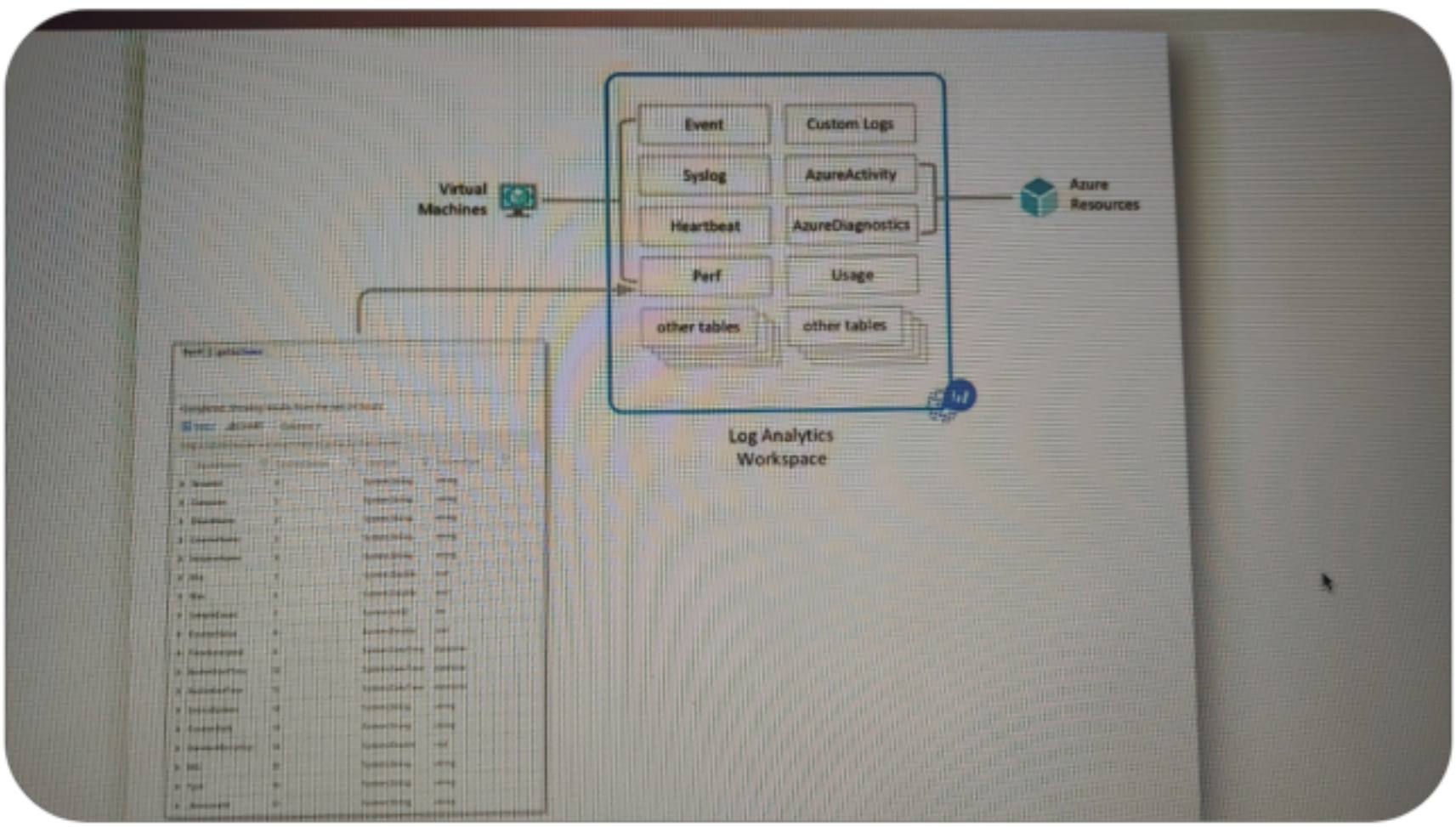
description, alert rule resource group and its severity level.

- Once Alert is defined, we can go to Alert option in the VM monitoring section, where we can see all those defined alerts.

Log Analytics Workspace

- Almost every Azure resource generate logs. This logs might contains data about performance, events, errors etc. These logs records need central repository to be stored and viewed. This is log analytics workspace.
- **Log Analytic Workspace:** A central location for storing, organizing and analyzing logs. It aggregates logs from all connected, monitored resources. It uses specialized query language to query logs.

- It will located in a designated region. Better be the same region of the resources to save costs



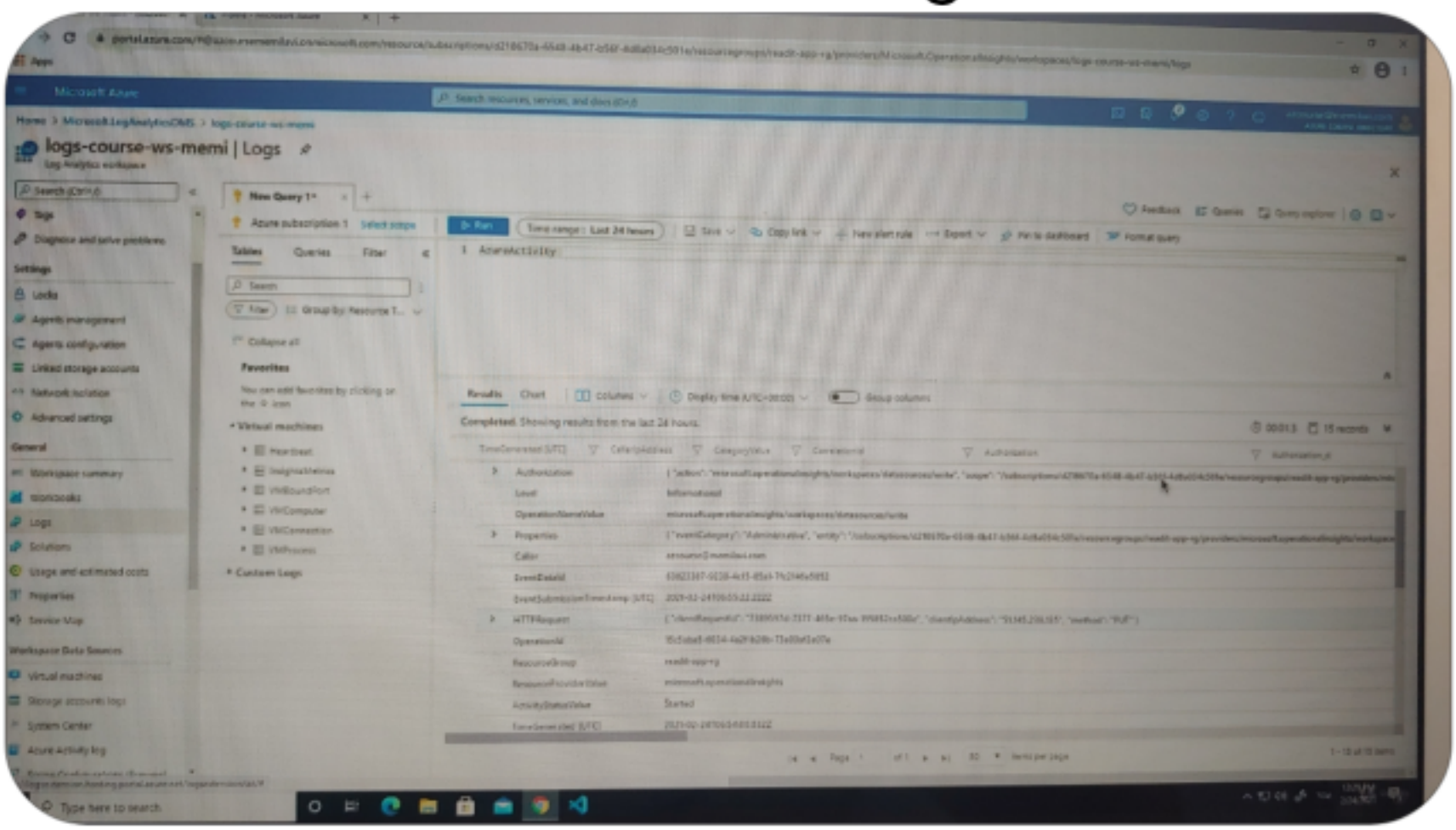
Log Analytic Workspace in Action

- Go to Log Analytics Workspace in Azure Portal. Create a new one by selecting subscription, resource group and region. Also provide a valid name.
- Now go to any VM's Monitoring section and click logs. Here click on enable option and select the previously created Log Analytics workspace to connect to it.

- Now go back to previously created Workspace menu and select the Virtual Machines option under the 'Workspace Data Source', where it list all the running VMs. It will show those VMs which are connected to it in green colour.
- Go to Azure Activity Log option from side menu, which contains the various log details of the Azure resource events like when it was created, when it was shut down, when it was modified etc. Every resource has this Activity Log where we can see various events which are generated.
- In order to stream the above Activity log of the selected VM to workspace, connect the Azure subscription where our resource is running to this workspace. Once it shows connected, then

we can now view and query logs from activity logs.

- Basically we have connected two types of resources. We connected our Catalog VM to this workspace and we connected the activity logs of all our subscriptions to this workspace too.
- Now under the General menu of Workspace, select logs and select the parent subscription under scope so that we can query logs from all the subscriptions.
- Now we can run various query and visualize the logs.



- ○ We can also configure which type of log events should be send to workspace in each resource's Activity Log setting. For example send logs of types Administrative, Security and Alert only.
- ○ We can also collect event logs and performance counters of the VMs into Analytics workspace.
- ○ Similar to alerts we can also pin log to dashboard, so that the query and its results will be always available for us.

Insights, Azure Monitor & Resource Tag

- ○ Its a collection of metrics, statistics and insights about the Resource. It is specific to resource type and is generated automatically.
- ○ Code based services like App Services, apps on VM can

integrate Application Insights into the code and gain a lot of data about app usage, performance etc

Insights in Action

- Go to storage account which we created earlier and under monitoring click insights. Insights offers us a lot of information about the availability of the storage account, the transactions tue latency, the performance, the capacity etc. Same way we can go to any running VM and see its insights.
- **Azure Monitor:** Its a central location for all the Monitoring aspects of Azure's resources. It provides access to metrics, logs, insights and more.
- **Resource Tags:** Sometime we want more information about the resource. For example, we might

want to know to which environment that these resources belong, to the test or to the production environment. Also, we might want to know to which app it belongs to. For this we have resource tags. Tags help organize resources using name-value pairs

- For example 'Environment: Test' this tells the resource belongs to Test environment. 'App: readit' this tells resource belongs to readit application. 'Group: A Team' this tells resource belong to A team group.
- Tag can be created during the resource creation or after that. Useful for many scenarios like Resource Querying(show all resource of the team A), Cost analysis(how much did the test environment cost last month)
- Go to any resource, select Tag

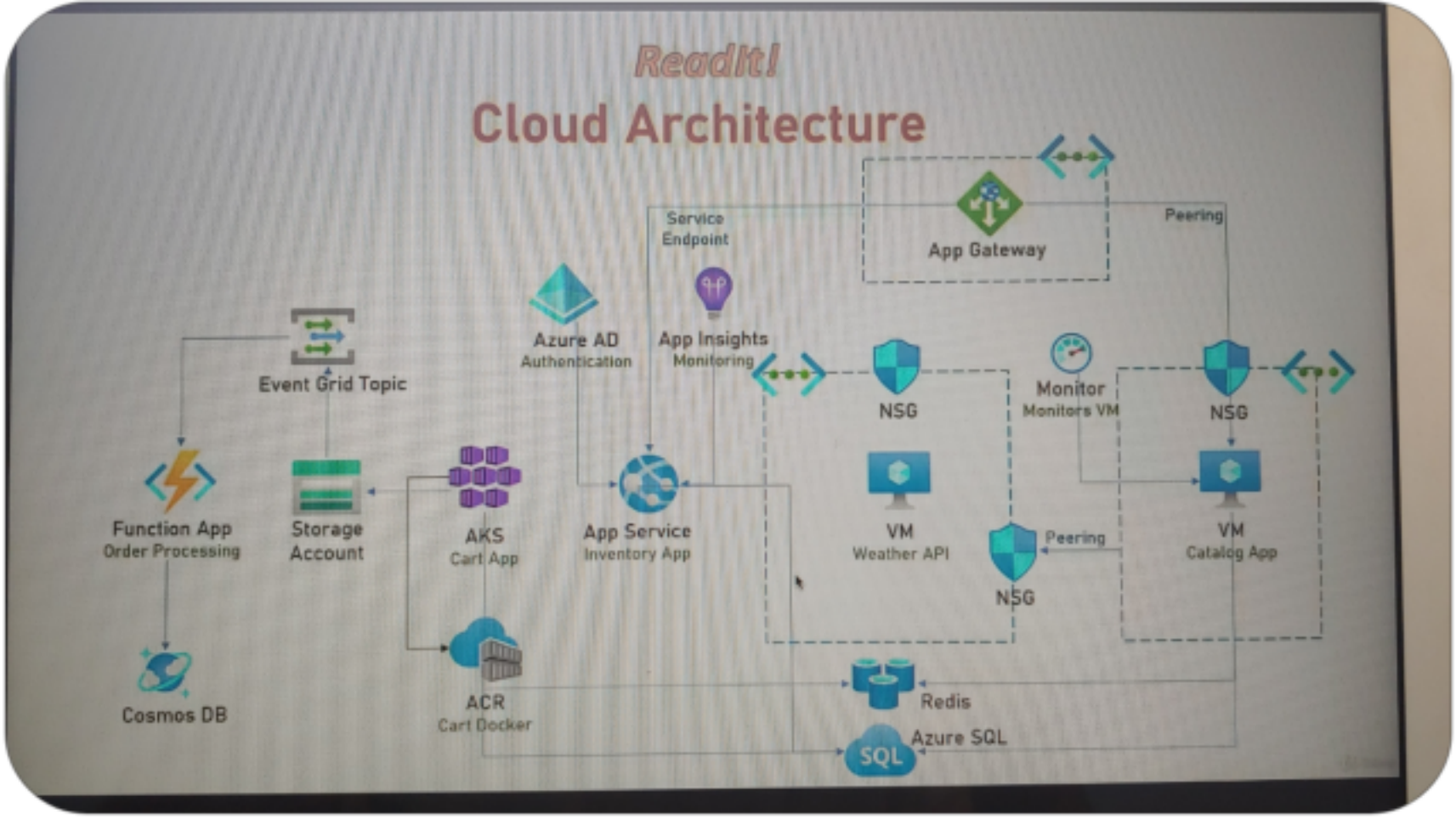
option and define name value pairs.

- Also, we can use Resource Graph Explorer tool in Portal

Azure Monitor and Application Insights in Action

- Go to portal and select Monitor option. There we don't need to create the monitor. It is already created for us. Here we have alerts, Metrics and logs which are taken form other resources.
- Also, we can see our own deployed application specific insights data also. We can define availability test for our application here, so it will call the application URL for every 5 minutes.

Current Architecture



- Here we have added App Insights connected to the Inventory App Service. Using this App Insights we can monitor the availability of the App Service and get notified when there are any availability problems with this service.

==========

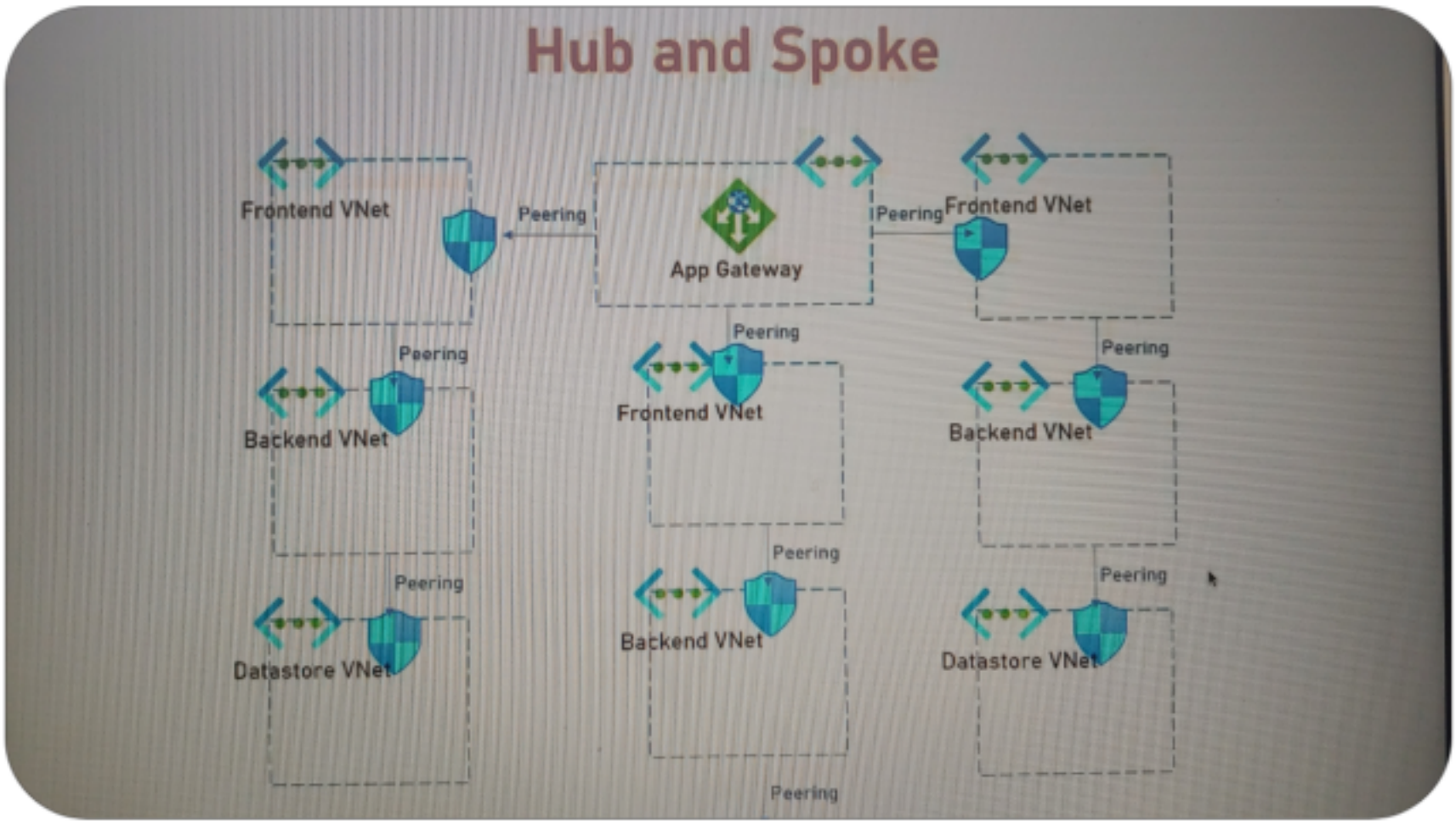
Security in Azure

- **VM Security Best Practices:**
  Restrict access to the VM as much as possible. Make sure only

the required ports are open to the internet(22/1389/443/80), whether it is the SH/RDP/TLS/ regular http ports.

- ○ In addition limit access to specific IP addresses when possible, which is more relevant to SSH and RDP that often should be accessed by specific user well authorized to log into the machine. Prefer using Bastion for accessing the VM, so no need for open ports.
- ○ If the VM is not public facing - place it in a VNet that's not connected directly to internet.
- ○ **Networking Security Best Practices:** VNet that contains private resources only - should not be exposed to the internet. Always use NSG to restrict access to Subnets or VM.
- ○ Use Service Endpoint / Private

Endpoint to restrict access to resources that should be connected to the VNet. Always use Hub and Spoke security model.



- **Database security best practices:** Use encryption at rest and transit with your database. This encryption will be there by default Connect DB to relevant VNet using Service Endpoint / Private Endpoint. Also, access DB from App using Managed Identities and do not store connection string as plain text.

Use DB's Firewall rule to restrict external access, so that only specific IP address can access DB.

- **App Service Security Best Practices:** Don't expose App services directly to internet, always use API Gateway. Connect to API Gateway's VNet using Service/Private Endpoint. Use Azure AD for authentication, enforce with MFA. Use Managed Identities to access other resources when possible that is true when connecting it to DB.

KeyVault

- Many apps have secrets that need to be kept safely. For example connection string to DB, Keys for encryption/decryption, Certificates, API keys etc

- These secrets/keys/certificates are usually kept in configuration files, databases etc which means they are not really secure. KeyVault or Vault solves this problem.
- **KeyVault:** Safely stores secrets of various types. It has very restricted access - needs Azure AD authentication in order to access it. Its easily manageable and accessed via Rest API.

KeyVault in Action

- Integrating Catalog App with KeyVault: Go to Portal, search for 'Key vaults' and create a new one. Select a unique Key Vault name.
- Click 'Access policy' setting and select the 'Azure Virtual Machine' option. This will allow VM to access this Key Vault. Create a Private Endpoint between this Key Vault and the VNet where

Catalog App is running.

- Now change the Catalog App code, to connect to Key Vault, so that all the secrets values will be read from Key Vault, instead of directly mentioning it in the configuration file.
- Finally, open Key Vault in Portal and add the required secret key value pairs. Make the required Firewall setting changes to access Key Vault for example adding IP address of the machine to allow access.
- Now, when we start the Catalog App in VM, it can not able to connect to Key Vault to get the secret values, because Key Vault needs the identity of this user to provide the access. So we should use managed identities to solve this. For that go to catalog VM, click on identity and enable it,

which will create a managed identity to this VM.

- ○ Now go to Key Vault, click on Access Policies and add a new policy, by selecting catalog VM's managed identity as a principal field value. This will allow the Catalog VM to access Key Vault.
- ○ Now Catalog VM can connect to Key Vault and fetch the list of catalogs from DB.

Security Center

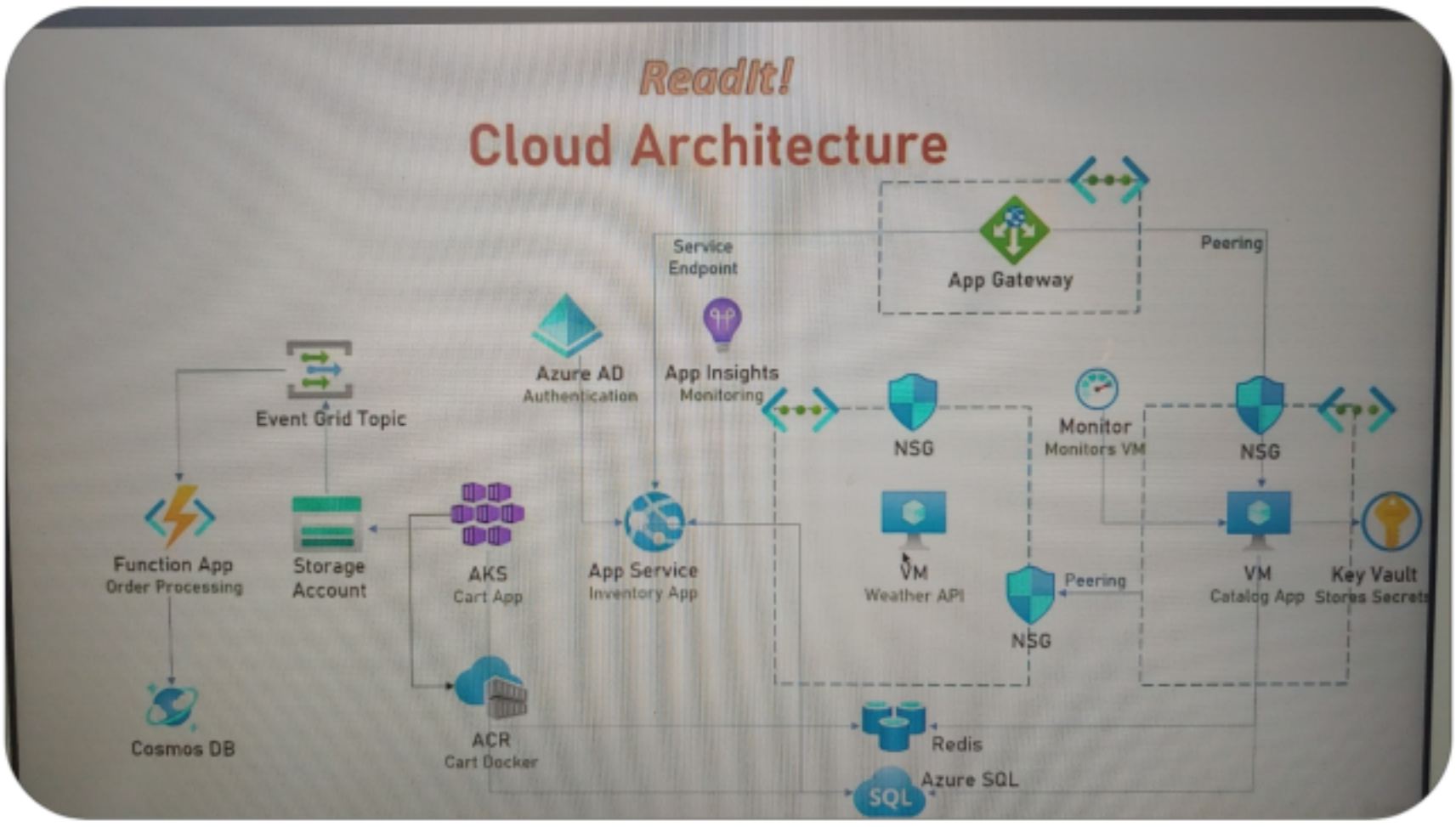
- ○ Its a central location for monitoring and alerting security related issues in Azure. It displays a summarized list of problems found in the subscription's resources. In some cases it provides a single click fix

Security Center in Action

- ○ Go to Security Center option in

portal, where we can see many security related improvements or recommendations for each of the resources.

Current Architecture



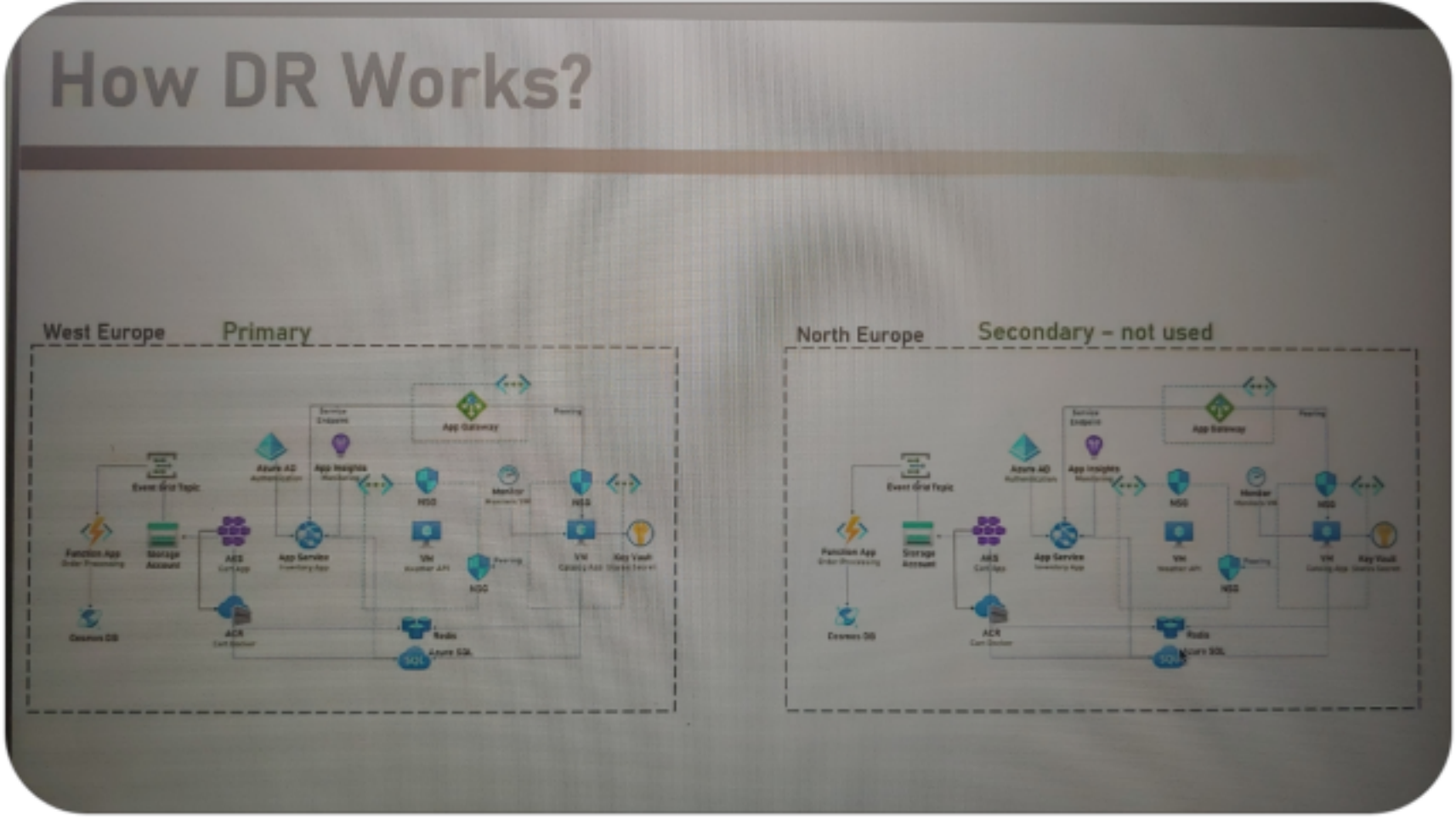
- The only change we made is addition of Key Vault. This Key Vault is used by Catalog App installed on the VM, so instead of storing connection string of DB in plain text in a configuration file, the app queries the Key Vault for this value using Managed Identity

and thus making it much more secure.

==========

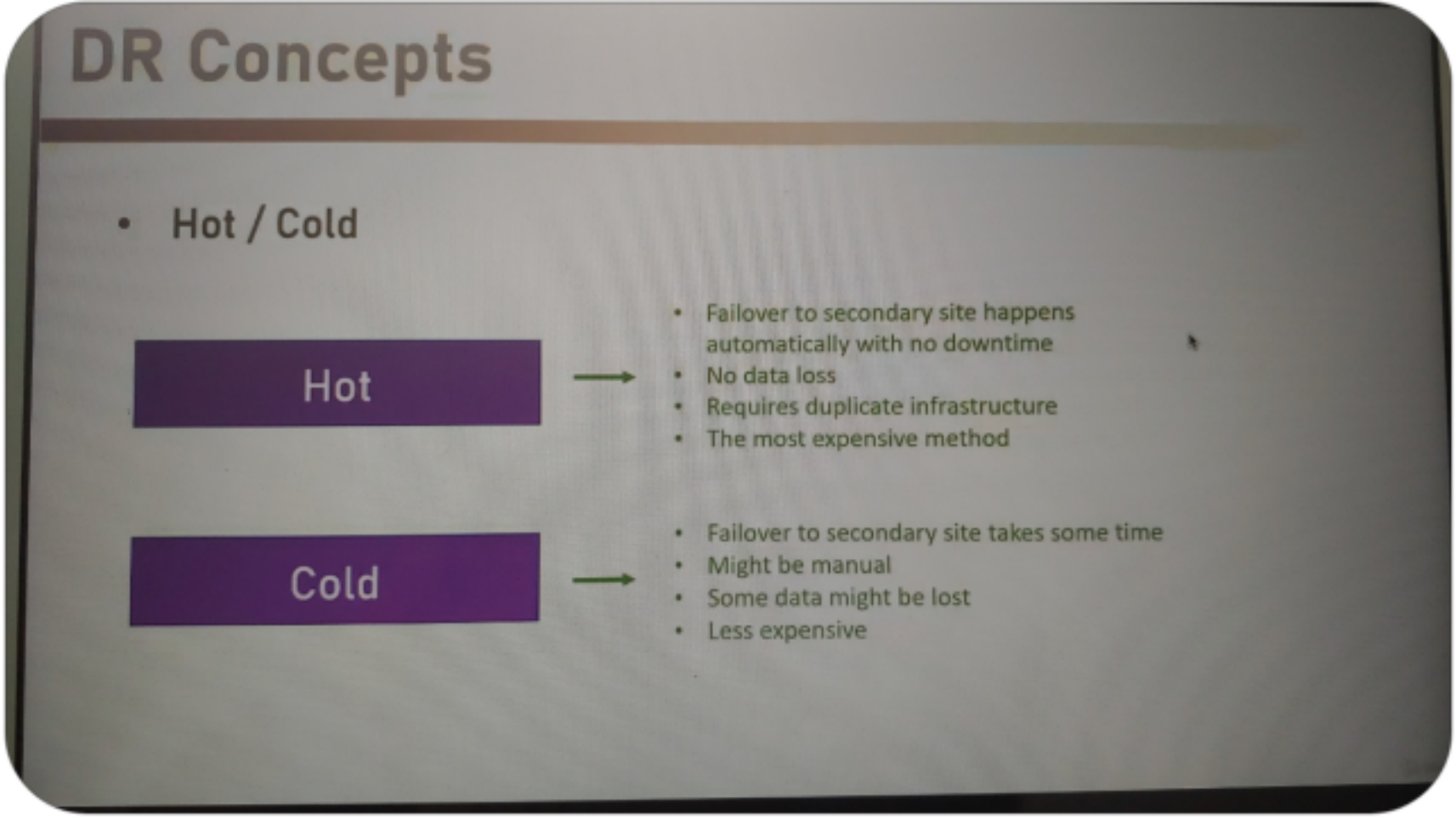
DR in Azure

- It is Disaster Recovery. Basically its a plan to recover from a complete shutdown of a region, usually as a result of disaster like earthquake, flood etc. Some apps require it some don't.
- **How DR works:** In order to setup DR, we need to do the following: Select a DR site, meaning a secondary region that will function as our primary in case of a disaster. And after selecting this site, we need to configure it to be ready for activation when necessary.

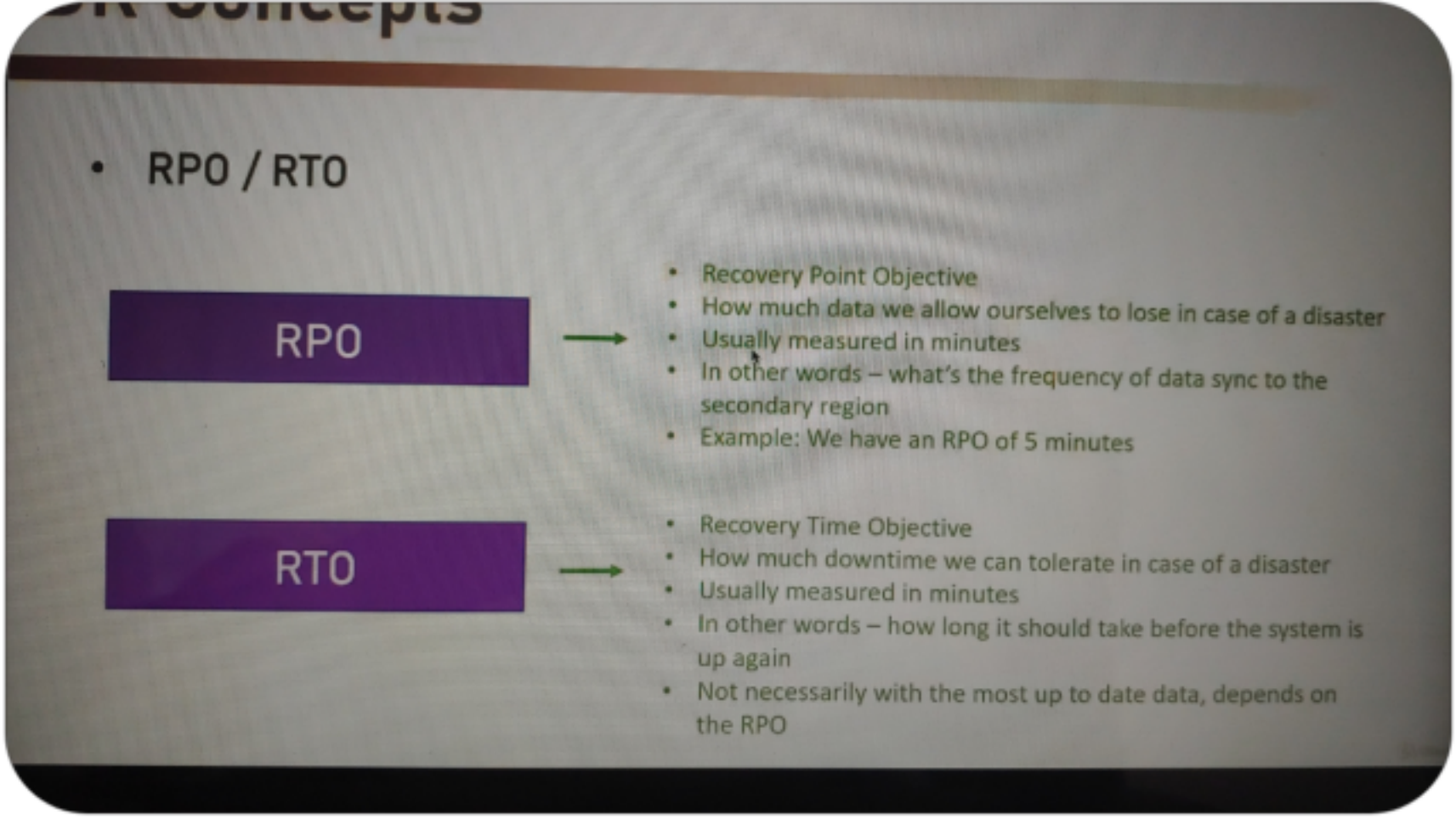


- Now when West Europe region is gone, then North Europe region becomes our primary region and from this moment on, this is the region that user will be using.

DR Concepts

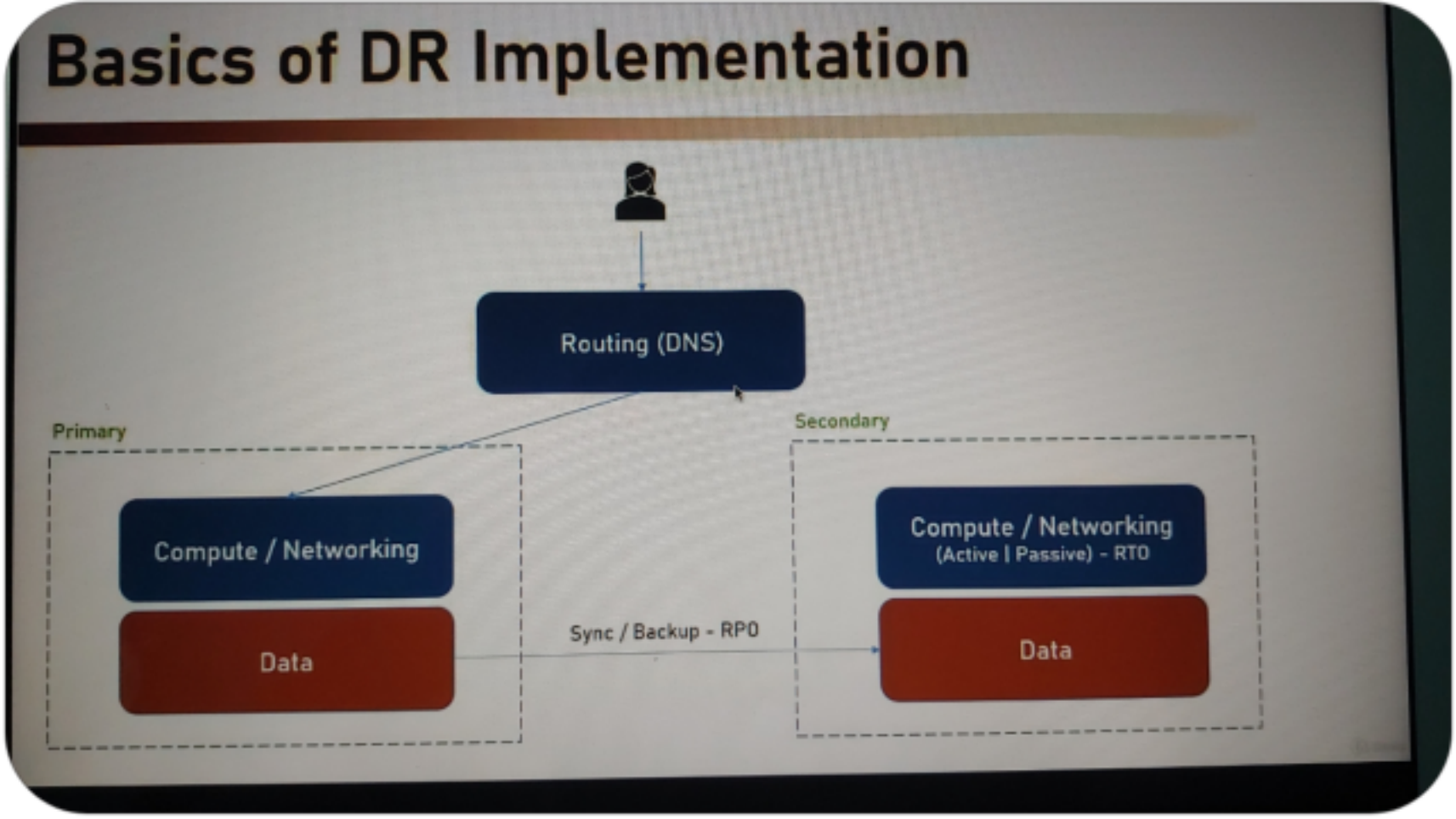


- How do we decide Hot or Cold: It depends on the system requirements. For example global e-commerce website, serving millions of customers - probably hot. An HR app for the organization - definitely cold.



- With RPO we are talking about data, and with RTO we are talking about the App itself.
- How do we decide RPO or RTO: It depends on the system requirements. For example a massive reporting system will probably go for low RPO, but can compromise on the RTO. Data in the reporting system must be critical. On the other hand a global chat app will focus on RTO.

Basics of DR Implementation



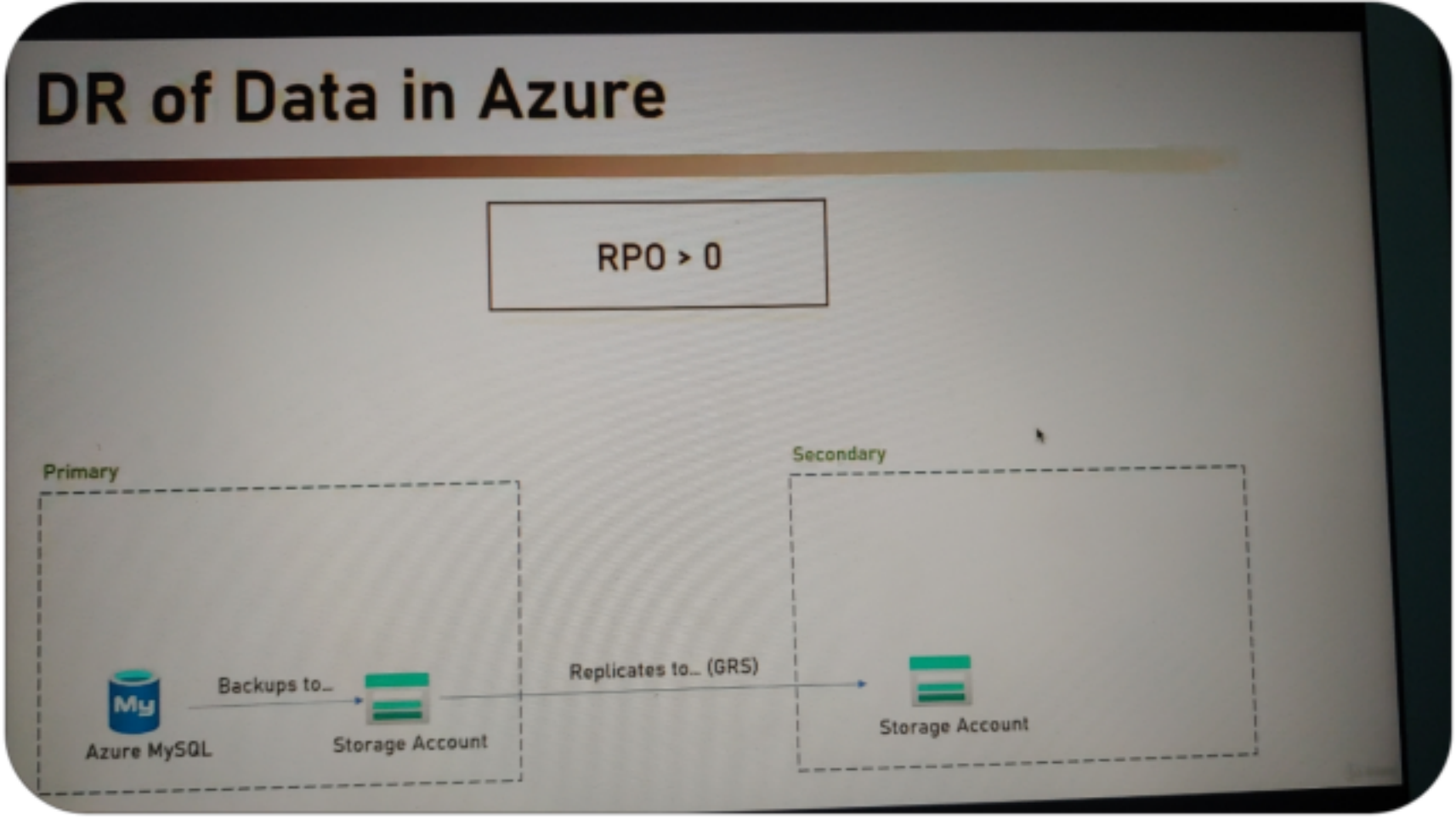
- In primary and secondary regions we have data and compute/ networking resources on top of this data. Note that with the compute and networking element they might be active or passive and that depends on RTO. If we need very low RTO we might keep them running all the time in secondary region also, even though we are going to pay for it, because it must take a very short time to make the secondary region operational. On the other hand, if the RTO is higher then we

can keep them passive or not working and will turn them on only when needed.

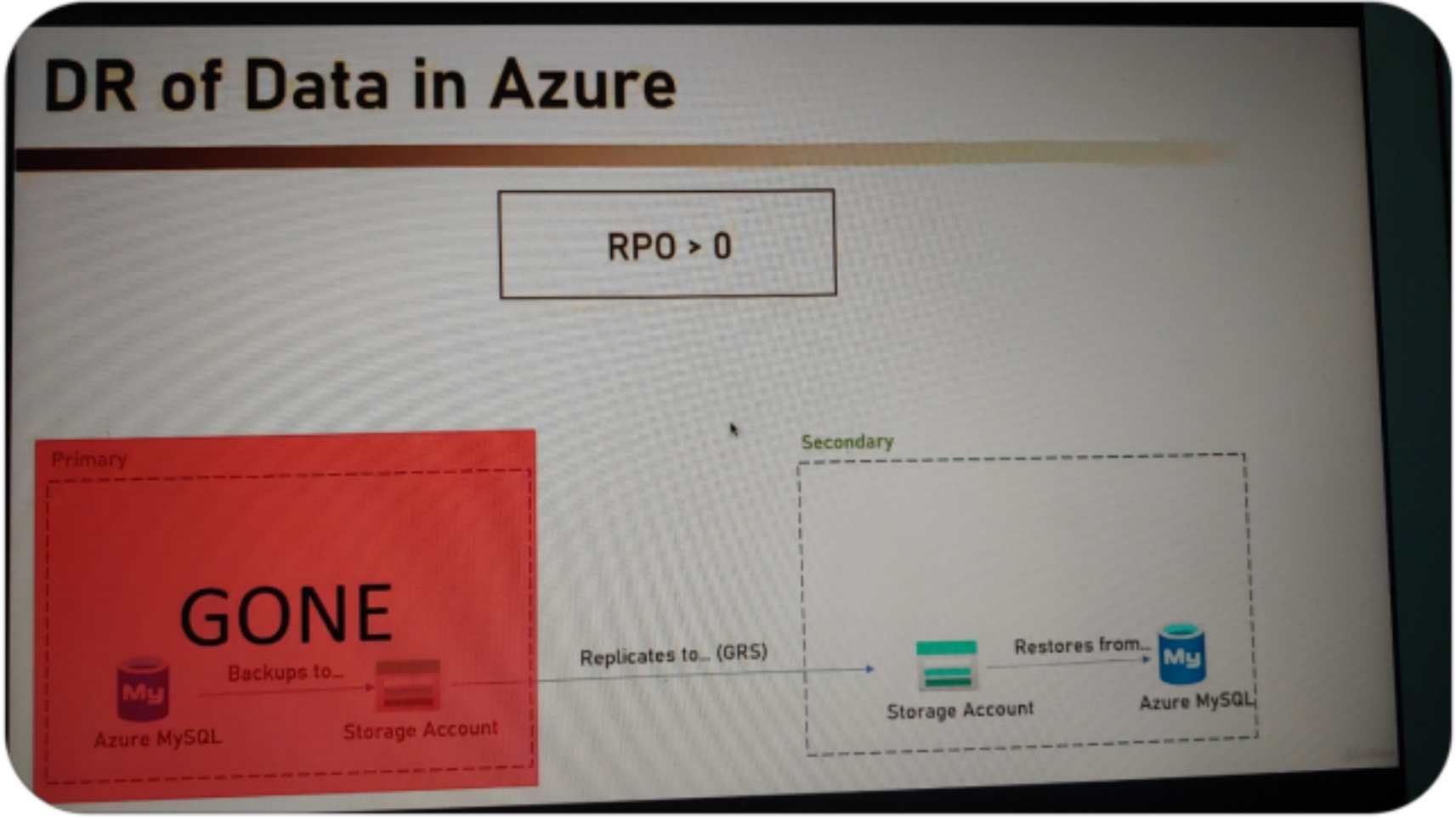
- ○ Since we have two databases one in primary and one os secondary, we need to define some kind of synchronization or backup between these databases. And the exact type synchronization or backup will depend on RPO. The lower the RPO, the higher the chances we will use synchronization and the higher the RPO, the higher the chance we will use backup.
- ○ And on top of everything we have Routing(DNS), which routes request from the end users to the app. Normally Router will route the request to primary region.
- ○ Now DR happens and the primary region no more active. Now we need to restore the data to

secondary region of not synced in real time. Next activate the compute resources in secondary region if it's not active. And finally modify routing which can be automatic or manual so that all request will route to secondary region.

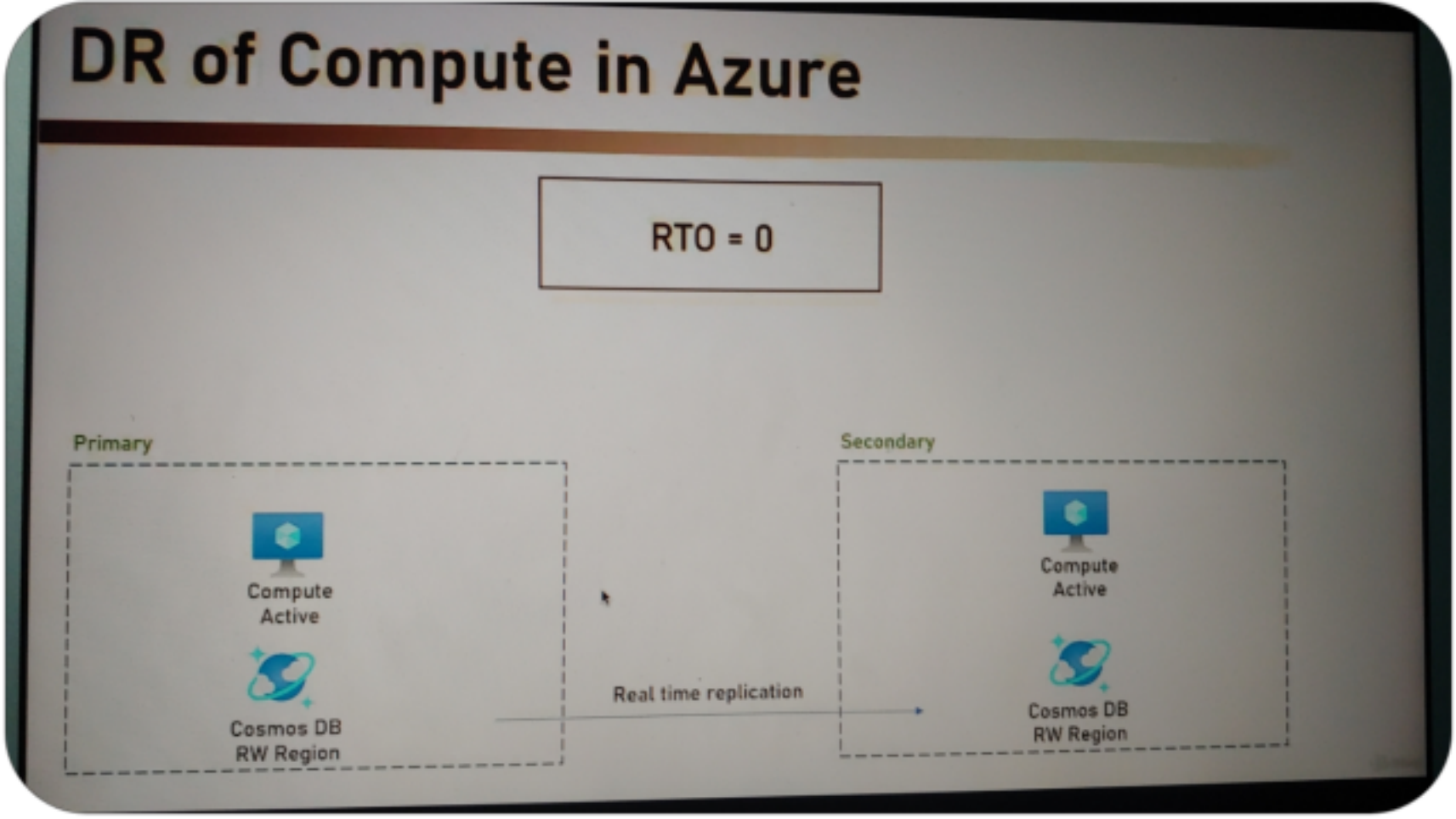
- **DR of data in Azure:** Main question when designing the DR of data is, what is the RPO? Or how much data loss do we tolerate.
- If RPO = 0(no data loss in case of disaster), then we need database that always syncs with the secondary region.



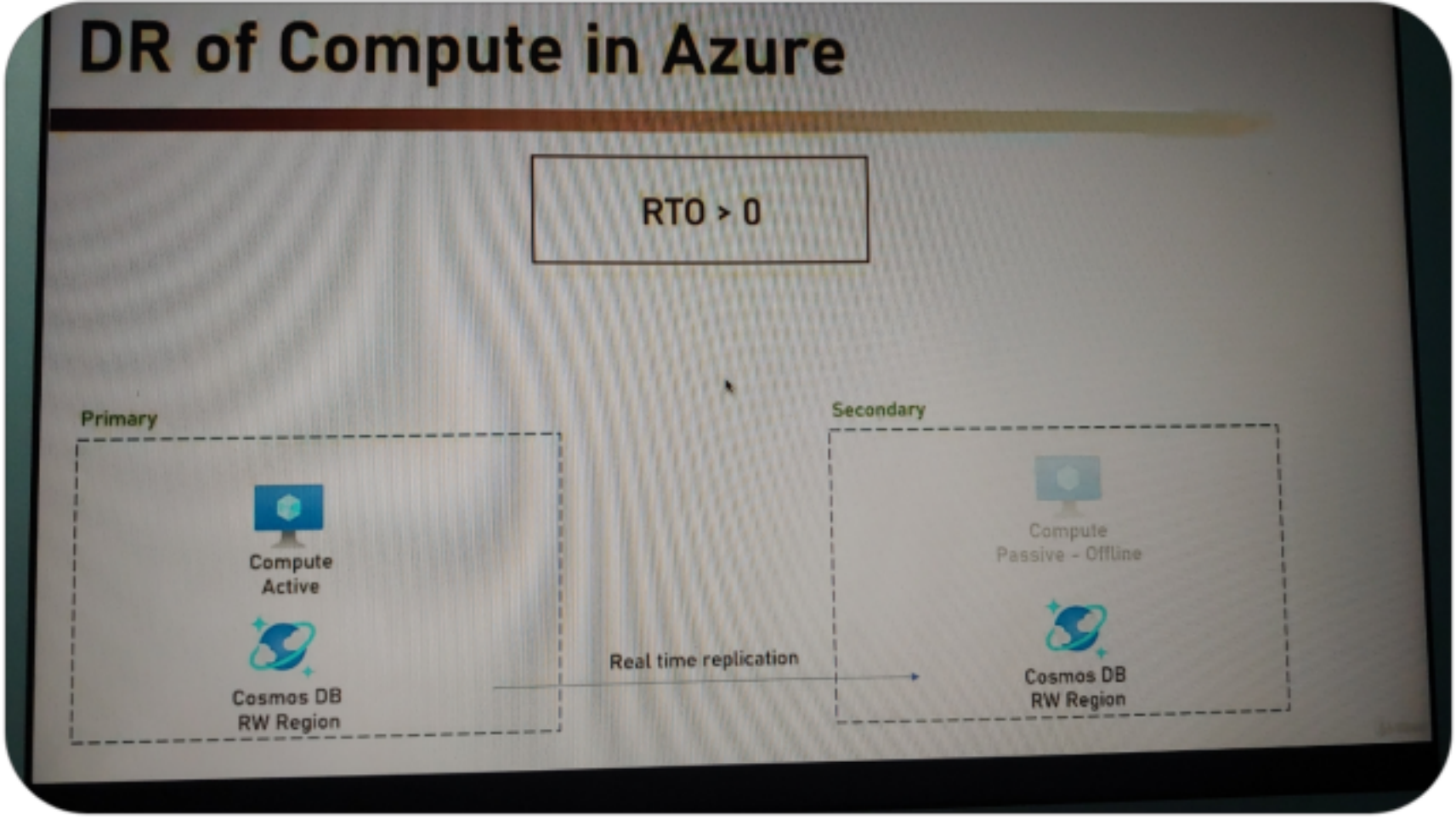
- Currently 3 such databases in Azure which provides continuous replications between regions, so no data losses occurs. Those are Azure sql (with Geo-Replication & Failover Group) Cosmos DB (with multi-region account) and Azure Storage (with GRS redundancy).
- If RPO > 0 (Some data can be lost), then ensure DB's backup frequency is compliant with the RPO. And the backup storage should be GRS, means it can be replicated in secondary region.



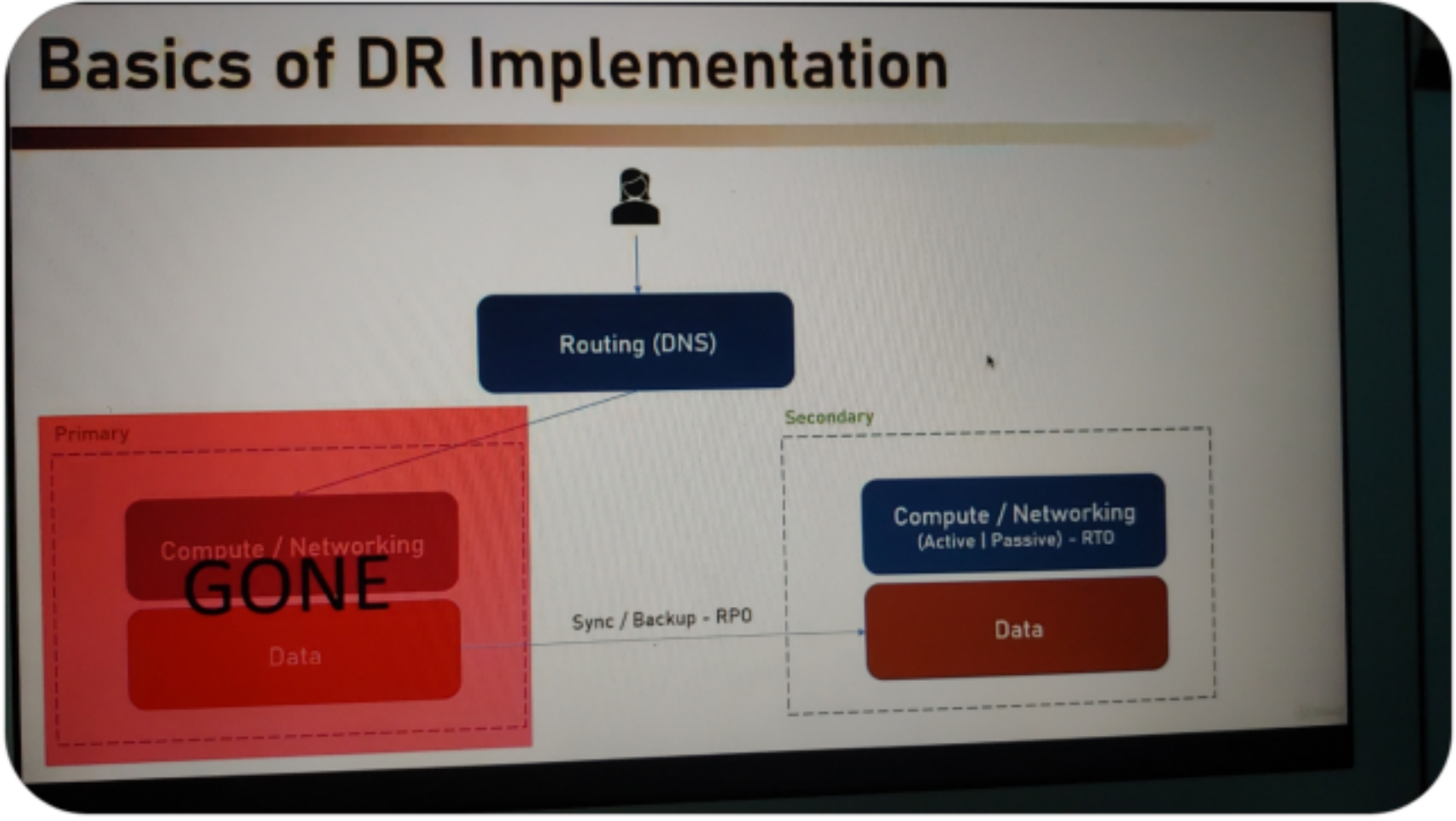
- Note that, the second example is much cheaper, no secondary active database is needed when primary is active.
- **DR of Compute in Azure:** Main question when designing the DR of compute is, what is the RTO or how much downtime can we tolerate.
- If RTO=0 (no downtime in case of disaster), then compute in secondary region should always be up and running



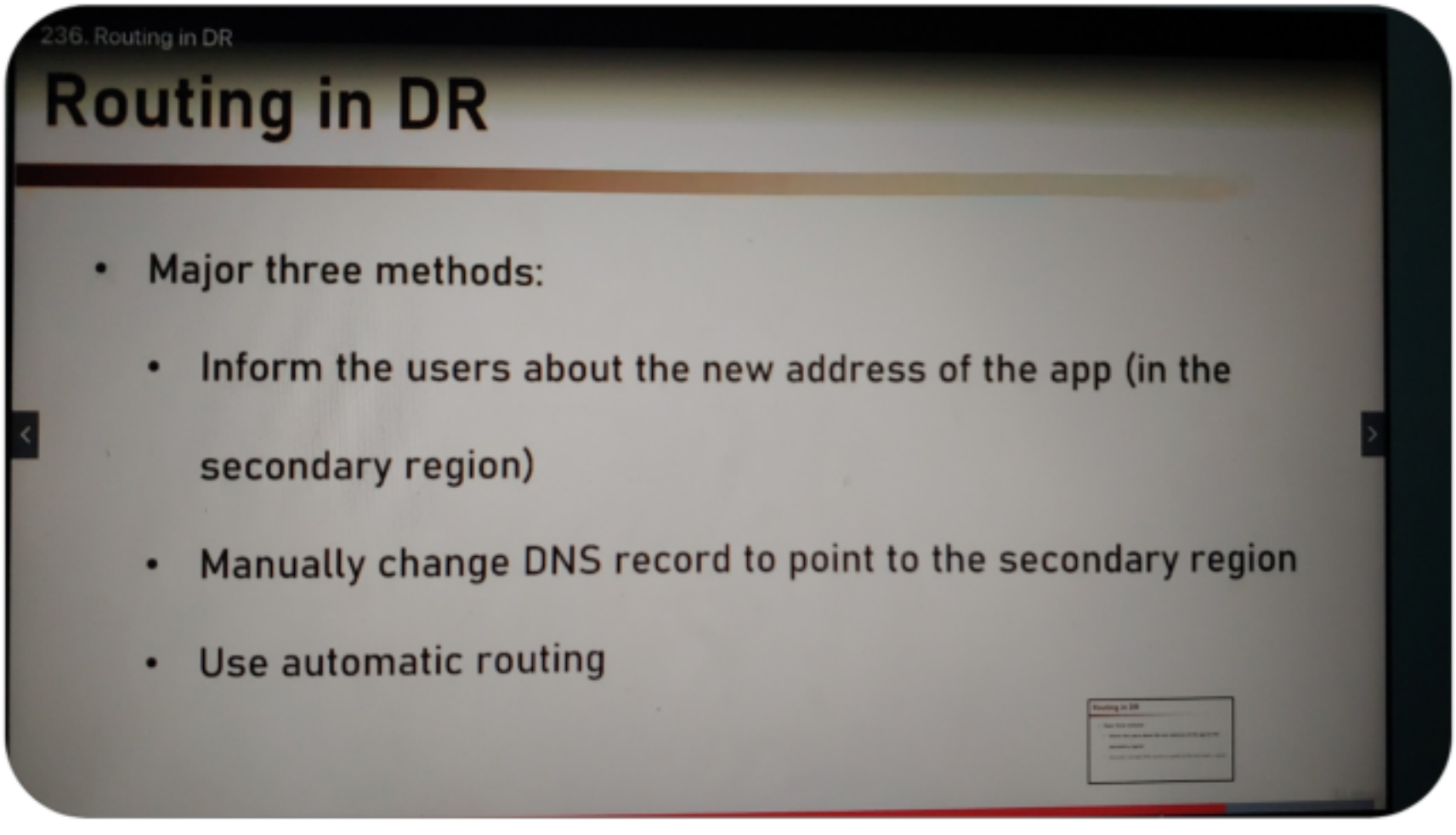
- If RTO>0 (some downtime is tolerated), then either we have a non-active(passive) compute on standby in secondary region or create the compute when disaster occurs in secondary region.



- When the primary region gone, compute resources in secondary region will become active and this ofcourse will take some time to become active.
- Note that, the second example is much cheaper, no secondary active component is needed when primary is active.
- **DR of Routing:** During DR users should be routed to secondary region.



- Once the compute become active in secondary region, request should be routed to secondary region.
- Methods for routing:

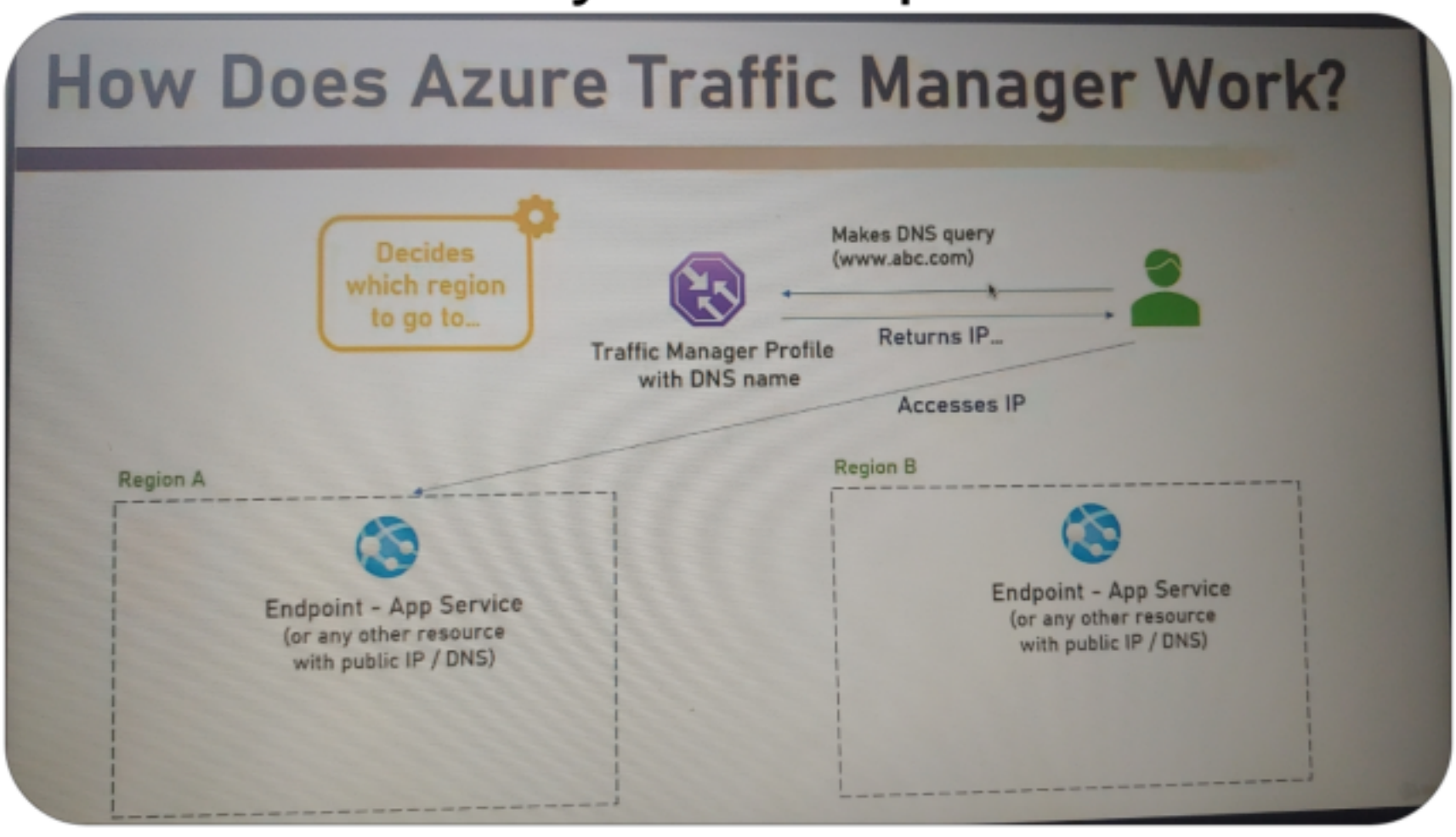


- Azure has two automatic routing

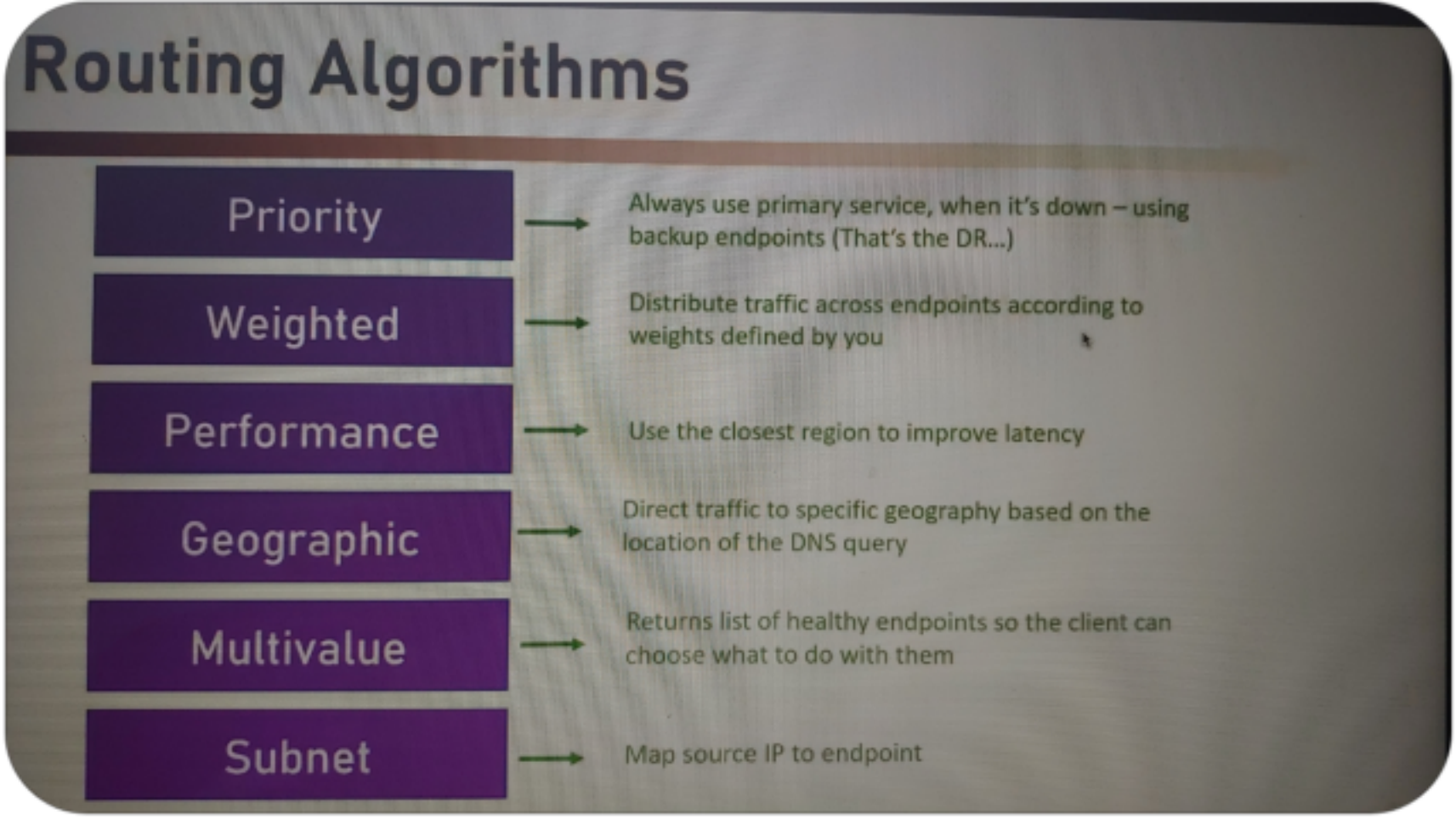
services. Traffic Manager and Front Door.

Azure Traffic Manager

- Its a DNS based traffic load balancer. It enables traffic distribution across global Azure regions. It provides high availability and responsiveness.



- **Routing Algorithm:** 6 types



- The priority algorithm provides us the DR capability.

Traffic manager in Action

- Go to portal and select Traffic Manager Profiles and create a mew one. Provide a unique name for this traffic manager. Select the routing method as 'Priority'.
- Once it is created go to its home page select configuration option. 2 of the main settings are Probing interval(means interval between the health checks, the traffic manager does to the regions. That means traffic manager is

going to check the health of the primary region every 30s and if it fails, then it will direct the traffic to the secondary region) & Tolerated number of failures (how many failures should the traffic manager receive before announcing their primary region as failed)

- We can place traffic manager in front of our apps like the Inventory App Service which is running at port 80. Go to Endpoints option in Traffic Manager and define the end point that the traffic manager is going to send the request. In this endpoint configuration, provide the details of the inventory App Service to which it needs to connect & send traffic.
- Now copy the DNS name of the traffic manager and open it in a

browser, which will indeed opens the inventory app service UI.

- ≡ ○ Now create similar inventory app service in some other region. Now add another new Endpoint in traffic manager, by providing details of this new app service which we created in secondary region. Select the priority as 2.
- ≡ ○ Finally, stop the primary inventory app service. And access the traffic manager URL in browser which will send the request to secondary region which has become active now.

Azure Front Door

- ≡ ○ Its another routing service apart from traffic manager in Azure. Its a global entry point for web apps. It has multiple routing methods.
- ≡ ○ It is similar to Application

Gateway but in global scale.

- Front Door Features: URL path based routing, session affinity, SSL offloading, URL Rewrites etc

Azure Front Door in Action

- Go to portal and open Front Doors option from search bar. Create a new Front Door. Provide configuration details for below 3 things. 1. front end or the address where the end user will go to 2. The back end pool where our apps are running similar to API gateway 3. Routing rules which will connect the front end to back end pool.
- Front end - provide a unique host name which will be accessed via browser. Back end pool - select the primary inventory app service which we created along with its port. Also, add another back end pool by selecting inventory app

service which is running in secondary region. And mention its priority as 2. Routing rules - select the front end and back end pool which we just created.

- Now copy the Front Door's front end host URL and access it in browser, which will show the inventory app home page.
- Now stop the inventory app service in primary region and access the front door URL, which will redirect the traffic to secondary region inventory app.
- Also, we can add Access Restriction rules to these app services, so that only traffics from front doors will be allowed.

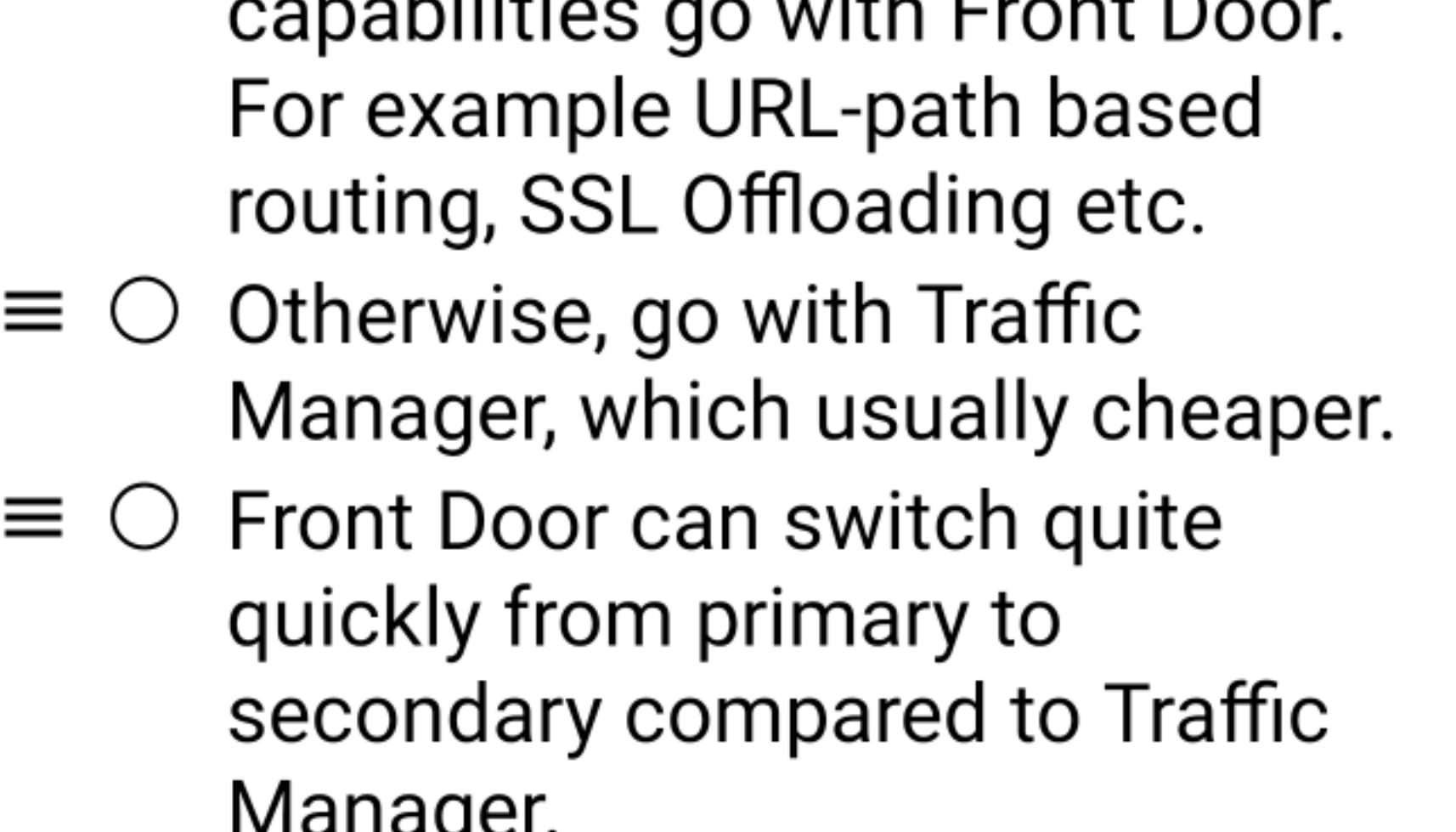
Traffic Manager Vs Front Door

- Which one to choose. Generally - if you need HTTP related

capabilities go with Front Door. For example URL-path based routing, SSL Offloading etc.

- ○ Otherwise, go with Traffic Manager, which usually cheaper.
- ○ Front Door can switch quite quickly from primary to secondary compared to Traffic Manager.

Current Architecture



- ○ There is a major change here. We have a secondary region along with primary region, which

currently contains only a single app service, because we didn't want to duplicate our environment.

- So we have Traffic Manager and Front Door at the top, both connected to inventory app service and to the secondary inventory app service.
- So we have added DR capabilities to our cloud architecture of the website.

==========

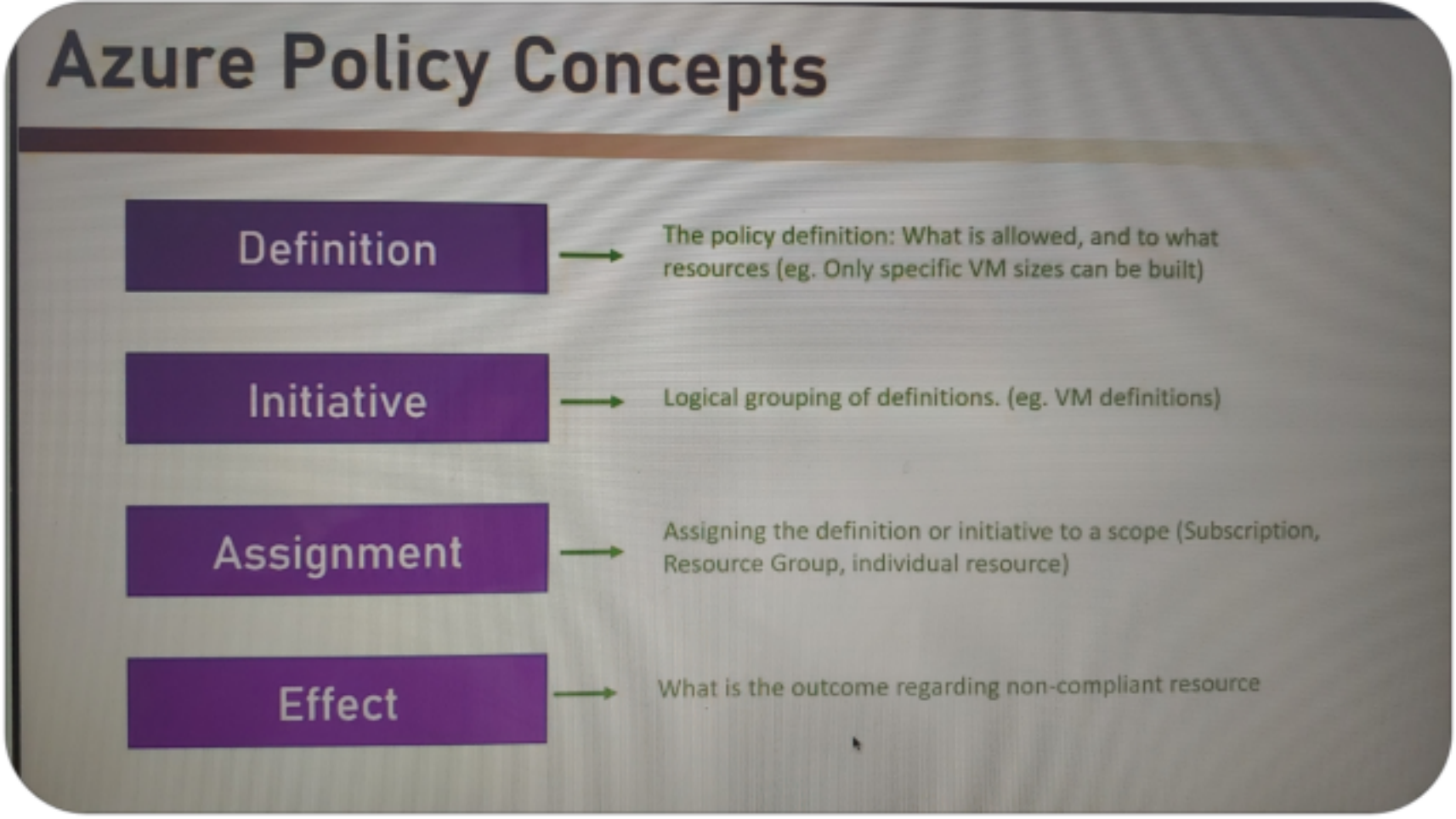
- **Cost Management Tool in Azure:** Its the central hub For all cost aspects. Displays reports with sophisticated filters, makes predictions, enables alerts and can even manage costs of resources outside Azure for example AWS or Google Cloud.

Azure Policy

- Enforcing organizational standards and compliance at scale is not simple. This is quite important in the cloud. There are many standards and restrictions that an organization would want to employ on it's resources. For example not everyone will be able to create any kind of results.
- For this Role based access helps a little, but not enough. It focuses more on the permissions and authorization aspects and not on the resource itself.
- So lets see some examples we would like to have with Azure. All VMs should be built on a specific region. Only specific types of VM are allowed to build. Tags must be specified on all resources in

the resource group. App services should only be accessible over HTTPS.

- **Azure Policy:** It is the mechanism for defining such restrictions. It allows to define policies, assigning policies to scopes (Subscription, Resource Group, Specific Resource), and it is free.
- **Azure Policy Evaluation:** Policies are evaluated when a resource is created/updated/deleted in a scope or when a new policy is assigned to a scope or when a policy is updated.
- **Azure Policy Outcome:** As a result of non-compliant resource
  - resource change might be denied, resource change might be logged, resource is altered before or after the change.
- **Azure Policy Concepts**



Azure Policy in Action

- Go to portal and search for Policy, which will show details of all compliant and non-compliant resources which we created.

Architecting Apps For Cloud

Architecting Apps for Azure

- Architecting for the cloud is different than classic Software Architecture
- Two main differences:
  - Use existing services
  - Consider cost

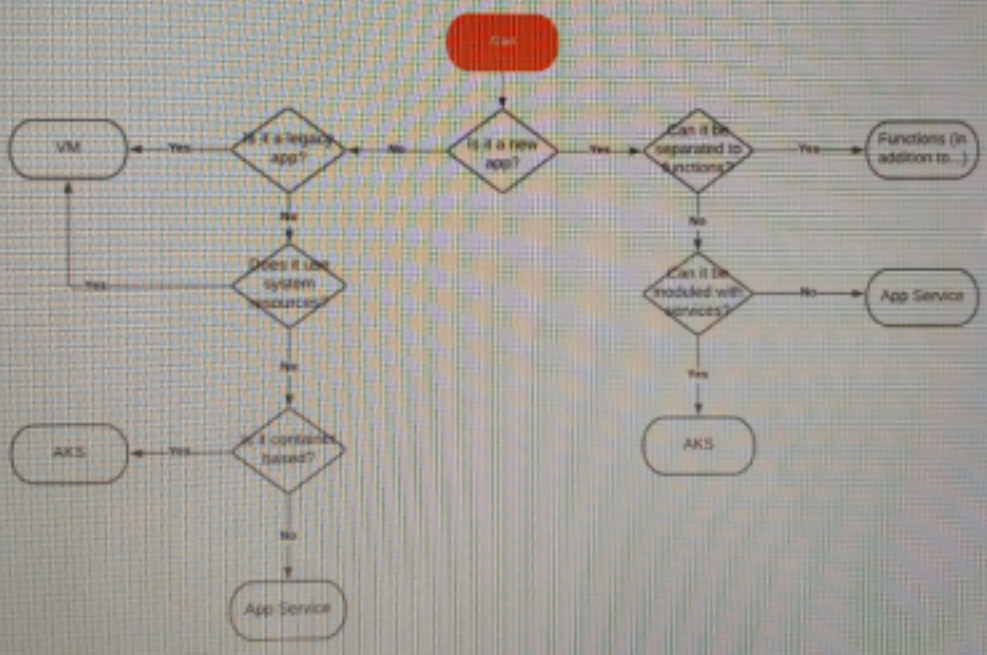
Use Existing Services

- Azure contains hundreds of services
- Whenever possible – use them
- Usually:
  - Managed
  - Reliable and Scalable
  - Cost effective

Consider Cost

- Cloud Architecture is cost oriented
- Always factor in the cost of the cloud service
- You'll sometimes go for limited service due to cost reasons
  - eg. Storage Queue vs Service Bus

Choosing Compute Platform



Choosing Data Platform

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

Choosing Messaging Platform

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

Implementing Security

- Extremely important in the cloud
- Use the best practices discussed in the Security section
- Mainly:
  - Restrict access to VMs and App Services
  - Use NSG
  - Use encryption in data stores
  - Use strong authentication

Logging and Monitoring

- Azure offers various logging and monitoring tools
- Utilize alerts to notify on any exceptional situation
- Create dashboards to visualize the system state
- Use Application Insight to gain insights into your app

Azure Architecture Center

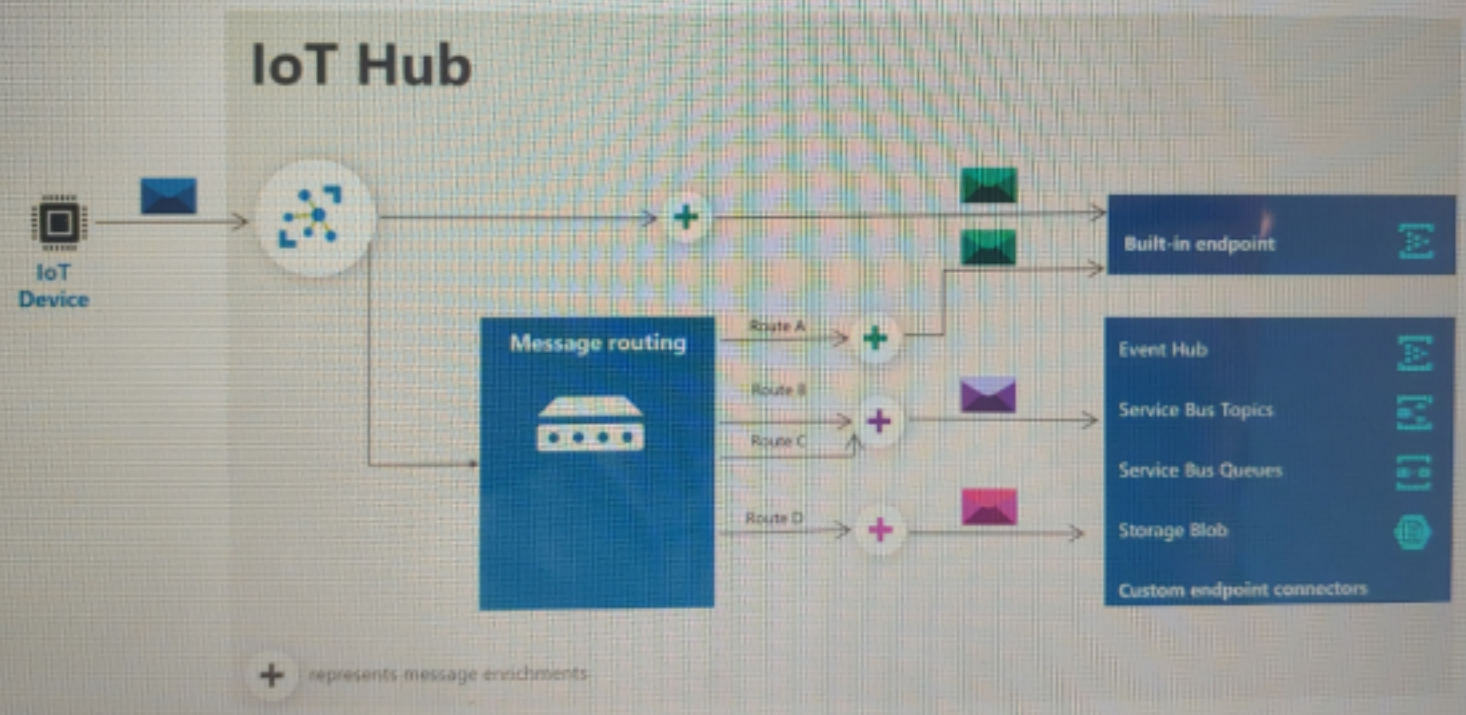
- Central hub for all-things Azure architecture
- How-tos, documents, design guidelines, case studies
- Fresh content, updated regularly

Advanced Services

IOT Hub

- Central message hub for IoT devices
  - IoT = Internet of Things (cameras, sensors, connected cars etc.)
- Handles security, reliability, enrichment, routing and more
- Integrates with other Azure services
- Handles millions of devices easily

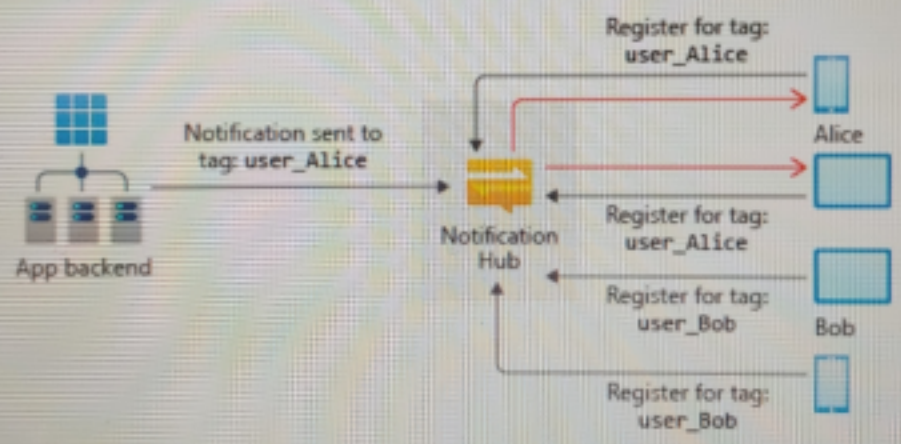
IOT Hub



Notification Hub

- Central hub for sending push notifications to any platform
- Supports iOS, Android, Windows, Kindle and Baidu
- Can be used by any backend, including on-premise one
- Supports million of mobile devices

Notification Hub



Cognitive Services

- Set of APIs running AI (Artificial Intelligence) algorithms
- Does not require machine learning knowledge
- Enables: Anomaly detection, Language Understanding, Translator, Speech to Text, Computer Vision and lots and lots more...
- Extremely easy to use

Summary

Microsoft Azure

From Zero to Hero – The Complete Guide

Architecture Summary

When designing cloud architectures for Microsoft Azure, use this summary that will help you design the system select the right resources for the right task.

Area	Resources / Notes
Compute	Virtual Machines (VMs) – When a full control is needed, or dev platform is not supported on any other cloud service (e.g. C++). Can be auto scaled using Scale Set. App Service – PaaS for web apps. Use when system is a web app running on modern platform. Not requiring any setup or configuration. Great integration with authentication providers and Application Gateway. Azure Kubernetes Services (AKS) – If your apps already run on Kubernetes or Docker containers – use it. If you plan to deploy your app using containers – use it. Azure Functions – Use for focused, lightweight actions (calculations, conversions, validation etc.). Extremely efficient, beware of cold starts.
Networking	VNet – Virtual Network. By default inaccessible to other VNets. Design your app around VNets using the Hob-and-Spoke model. Subnet – Logical segment inside a VNet. By default accessible from other subnets in the VNet. Network Security Group (NSG) – Filters traffic based on 5 tuples. ALWAYS set up NSG in front of every subnet. Load Balancer – Balances load across various resources based on various algorithms. Works on Layer 4. Use ONLY for internal resources. Application Gateway – Used to expose web endpoint to the outside world. Load balancer, WAF (optional), works on Layer 7. Use in front of App Services.
Data	Relational Databases – Azure SQL (the only resource with 99.995% SLA), Azure MySQL, Azure Postgres NoSQL Database – Cosmos DB (the only resource with 99.999% SLA) Object Store – Blob Storage

Messaging	Storage Queue – Dead simple queueing, no additional cost, max msg size – 64KB Event Grid – Messaging system for Event Driven Architectures, great integration with other services, max msg size – 1MB Service Bus – Advanced queueing solution with advanced features, durable, max msg size – 256KB Event Hubs – Big data streaming, designed for heavy load, Kafka-based. Max msg size – 1MB
Authentication	Azure Active Directory – Identity and Access Manager (IAM), extremely robust, supports MFA, conditional access and more. Great integration with App Services. Azure AD B2C – Simplifies integration with apps for identity scenarios (log in, log out, sign up, etc)
Monitoring	Set up Alerts to get notifications when something goes wrong Use Metrics to see system's status Read Logs to find out what the system did Put important information on Dashboards to get wholistic view Use Tags to categorize the resources

Security	- Close unnecessary open ports of a VM - Use NSG - Use authentication - Encrypt data at rest and at transit - Use KeyVault to securely store secrets - Use Security Center regularly
DR	- Decide between hot and cold DR - Remember – hot DR is much more difficult to design and expensive to implement - Use Traffic Manager or Front Door for automatic routing

I Hope you enjoyed the course, and that it made you a Microsoft Azure expert. Stay tuned to more courses on Azure, which will make you an even better Azure Architect!

For any question or comment contact me at:

memi@memilavi.com

Thanks,

Memi