IT457 – Cloud Computing

Course Information

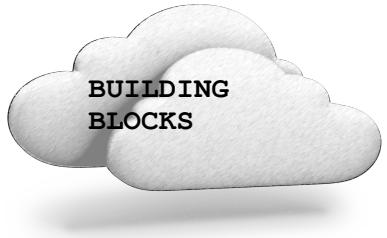
- Grading Scheme

  - Mid-sem exam – 40%
  - End-sem exam – 40%
  - Labs – 20%

  *subject to online exam system working well

Course Information

- Course Plan
  - Cloud Computing – Developer perspective
  - Cloud Computing – Architect perspective
  - Cloud Computing – IT perspective
- Cloud Service Providers
  - Azure
- References
  - Online material (like here, here and here).
  - No single reference book



Cloud Computing

- Definition

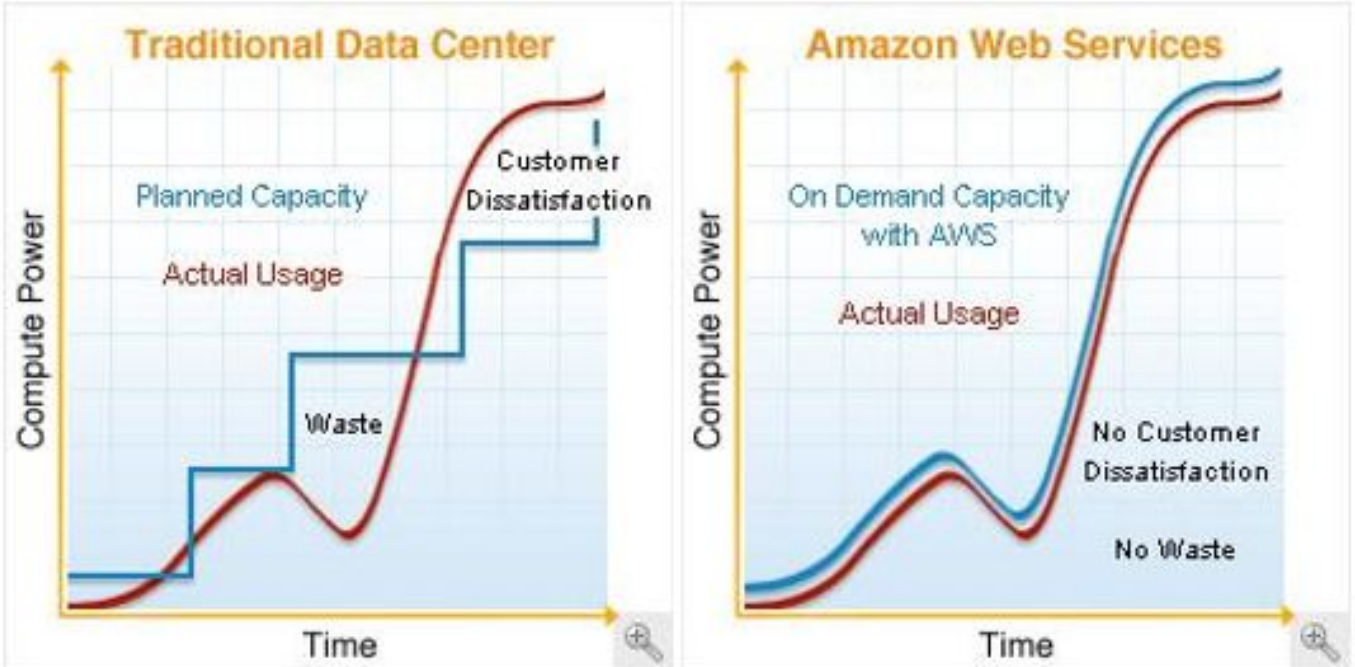
"Cloud computing is a specialized form of distributed computing that introduces utilization models for remotely provisioning scalable and measured resources" and platform services

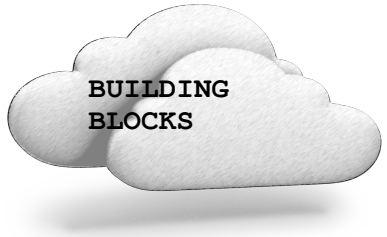
- Business Drivers (i.e. moti

Capacity Planning

Cost Reduction

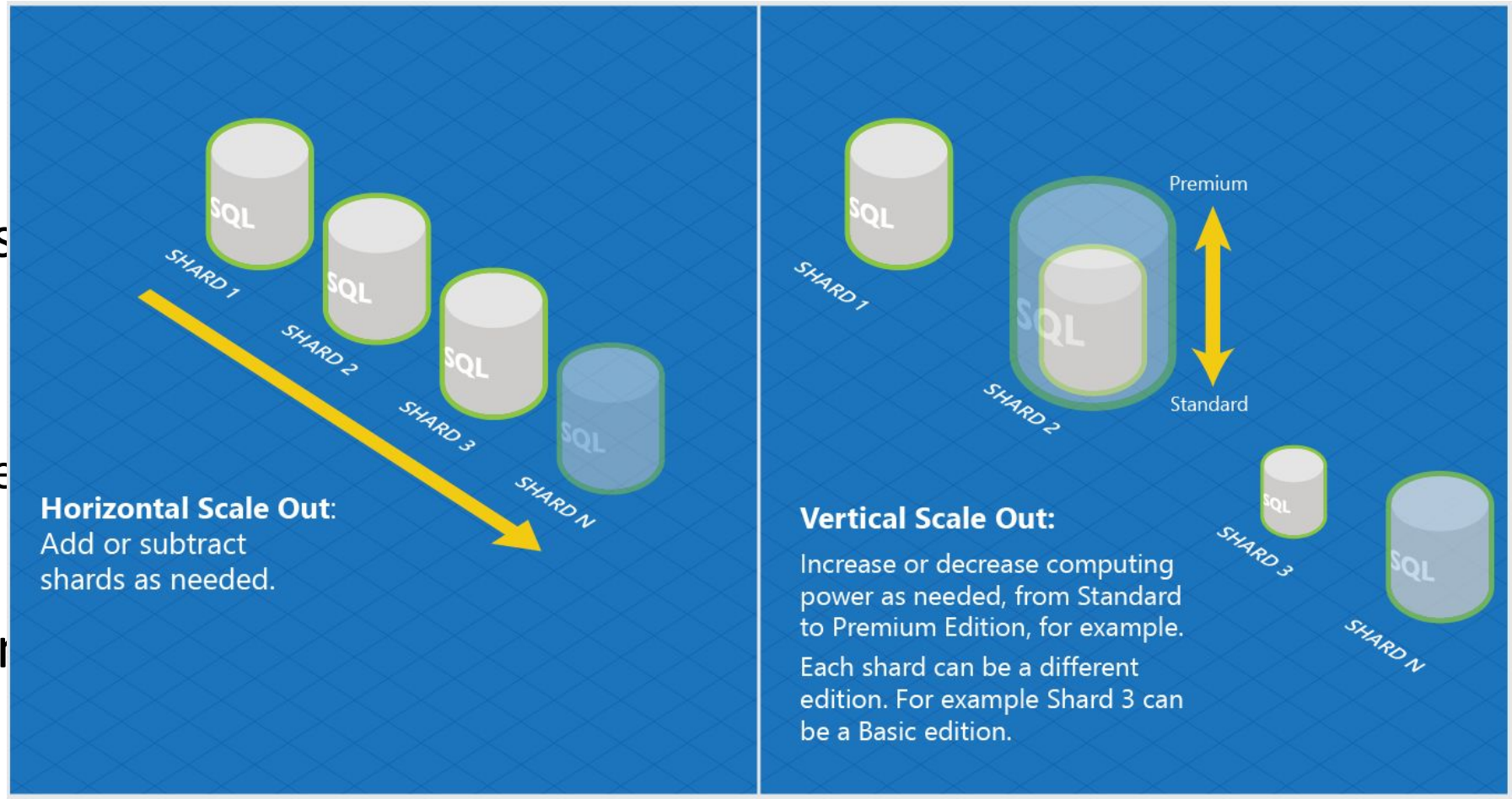
Organizational Agility

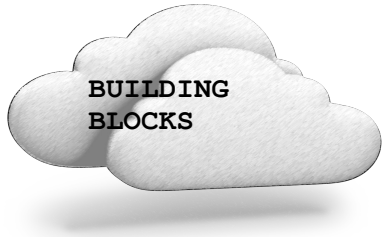




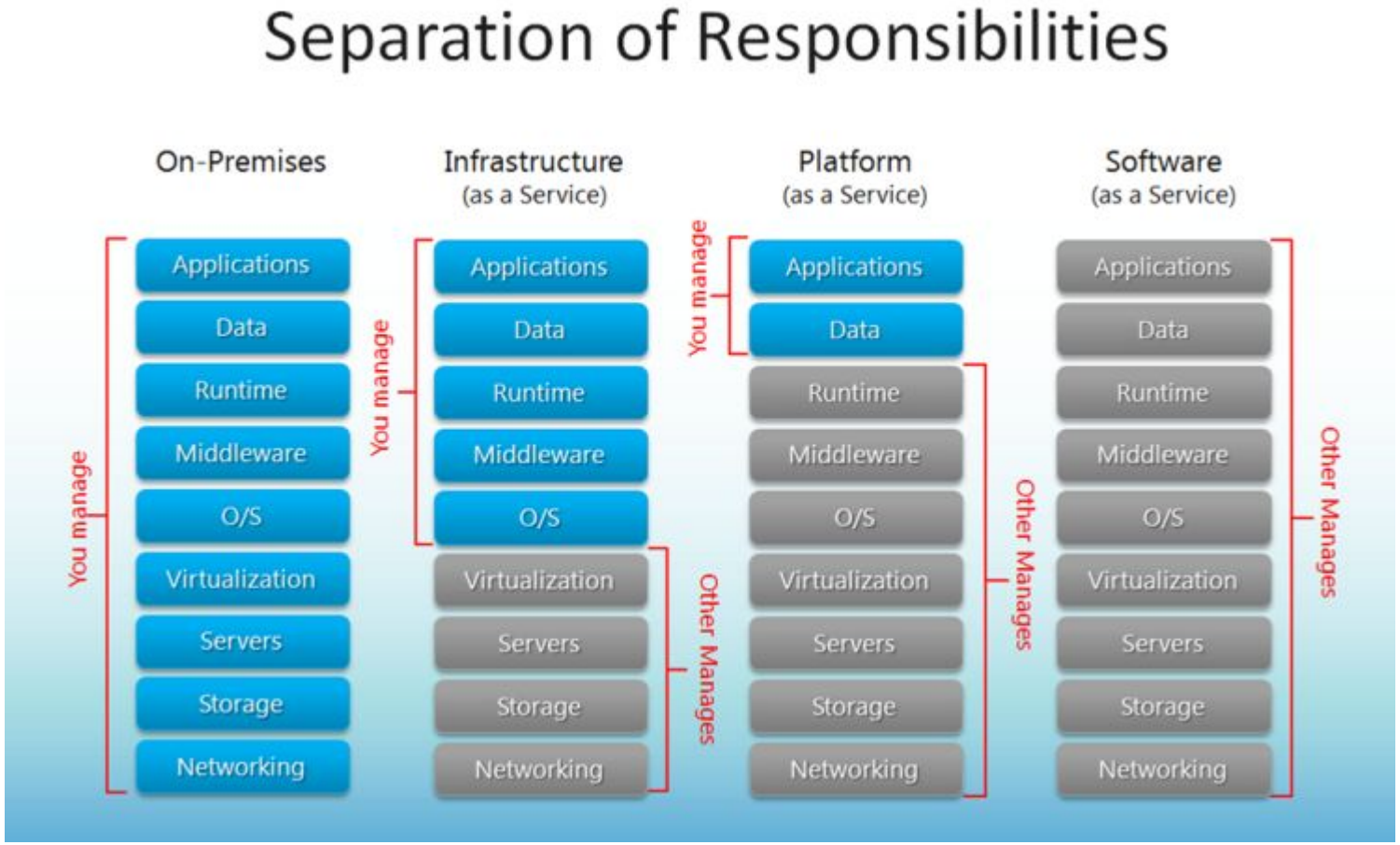
Basic Concepts & Terminologies

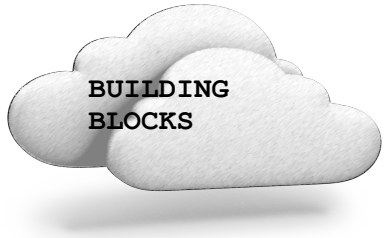
- Cloud (same as internet?)
- IT Resource
- On-Premise
- Cloud Consumers
- Scaling
  - Vertical (Scale U
  - Horizontal (Scale
- Cloud Service
- Cloud Service Co





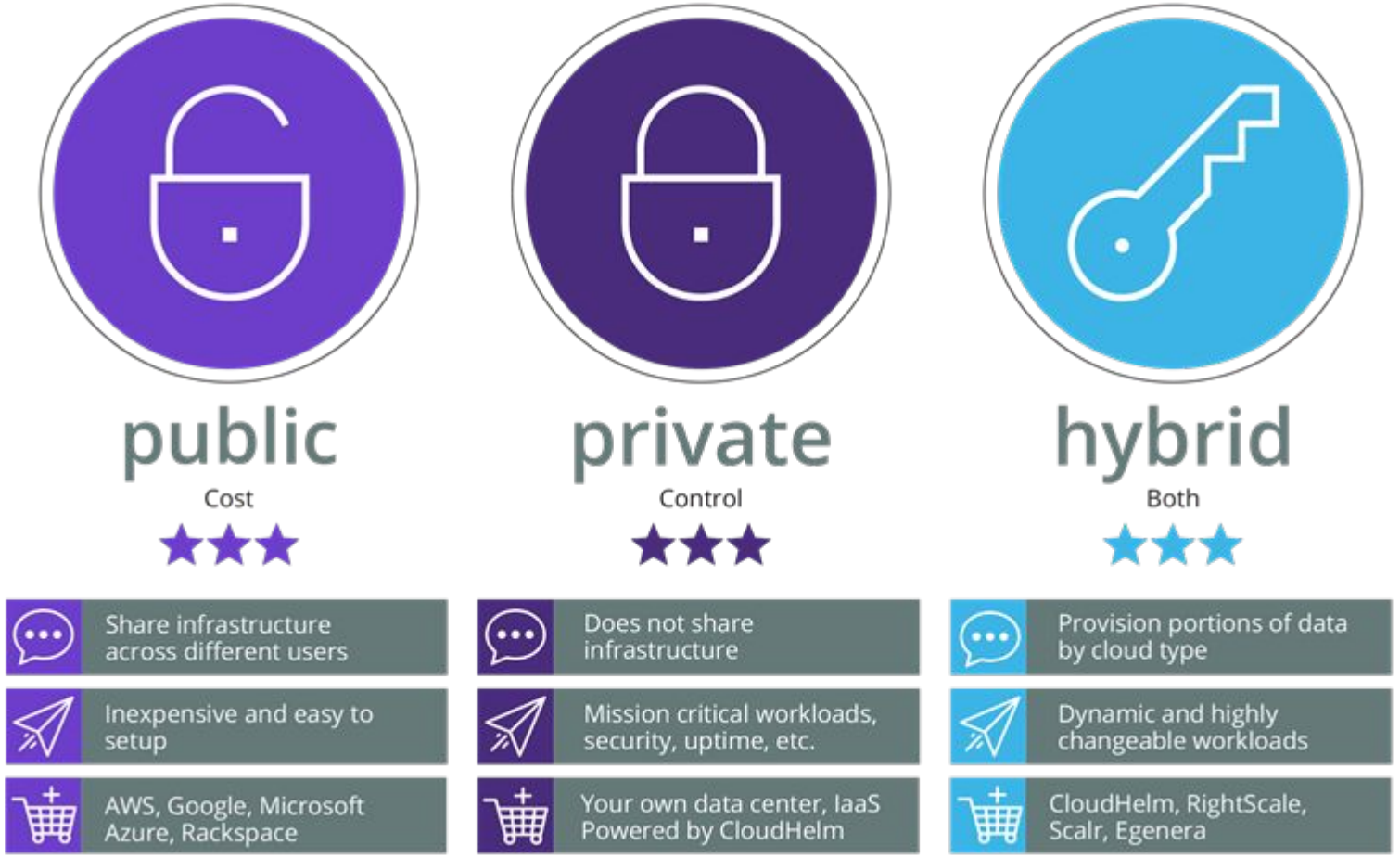
Cloud Delivery Models





Cloud Deployment Models

- Public Cloud
- Private Cloud
- Hybrid Cloud
- Community Cloud



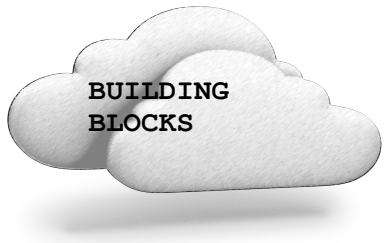
Cloud vs Traditional

Traditional (on-premises) Systems

- Monolithic, centralised
- Design for predictable scalability
- Relational database
- Strong consistency
- Serial and synchronised processing
- Design to avoid failures (MTBF)
- Occasional big updates
- Manual management
- Snowflake servers

Cloud based Systems

- Deconstructed, decentralised
- Design for elastic scale
- Polyglot persistence (mix of storage technologies)
- Eventual consistency
- Parallel and asynchronous processing
- Design for failure (MTTR)
- Frequent small updates
- Automated self-management
- Immutable infrastructure



Cloud Provider – AWS

Amazon Web Services

AWS World

AWS Account

AWS Company Account

AWS Account (Master)

AWS Account

AWS Account

AWS Account

AWS Account

AWS Account

AWS Account

AWS Organizations

OU: Admins

SCP: Admin

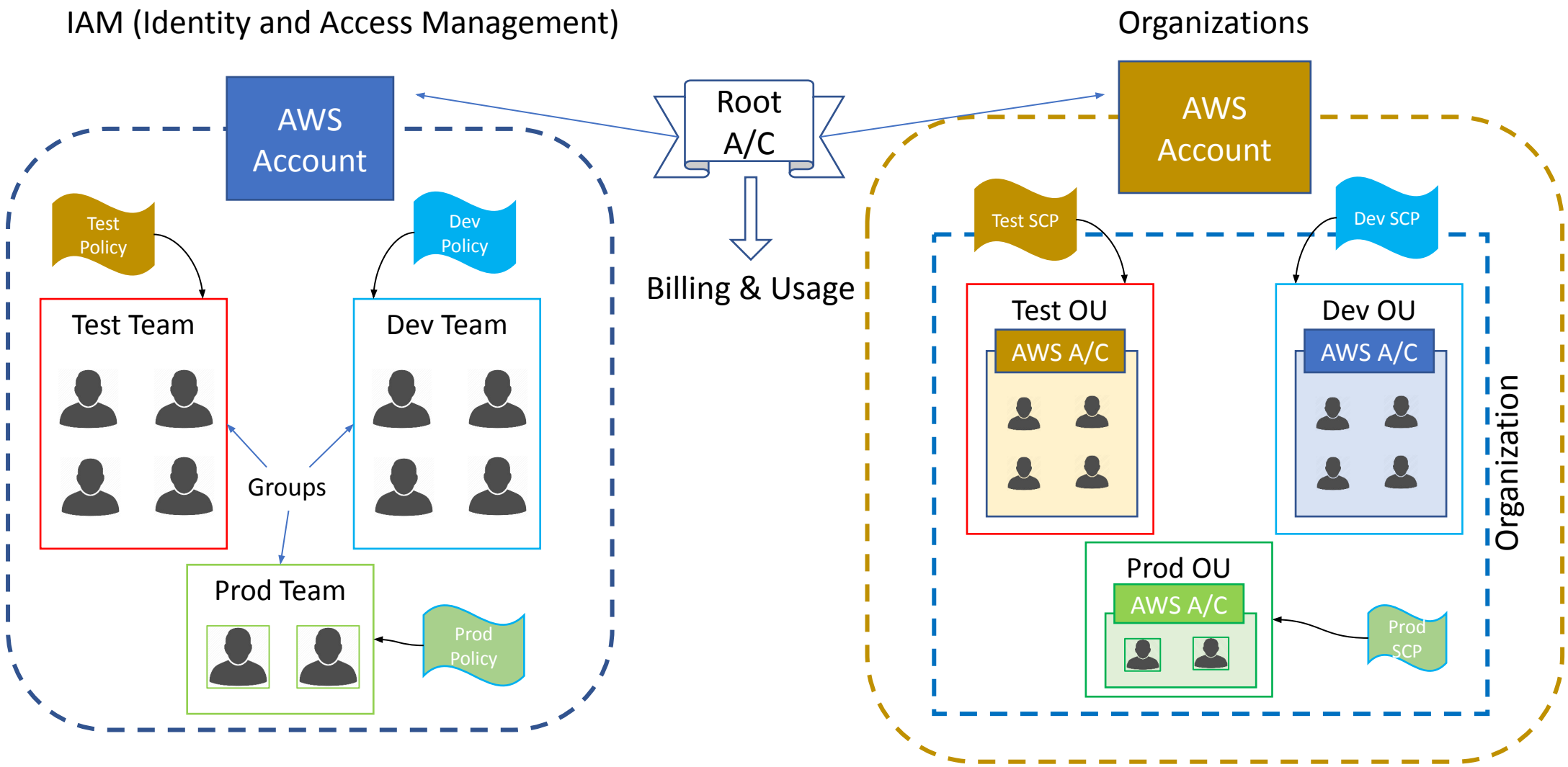
OU: Dev

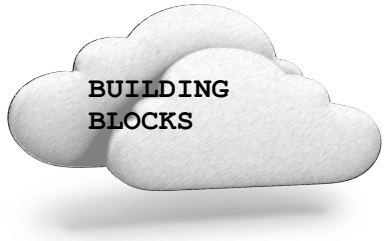
OU: Test

SCP: DevTest

BUILDING BLOCKS

AWS – IAM vs Organizations





Cloud Provider – Azure

Azure World

<id>@Hotmail.com; <id>@outlook.com; <id>@live.com;
Used for accessing various Microsoft services.

Microsoft Account

Organizational Account

1:1

Account Administrator

Azure Account

<id>@<org name>.onmicrosoft.com => Gives a unique domain for accessing Azure services for the org.

Subscriptions

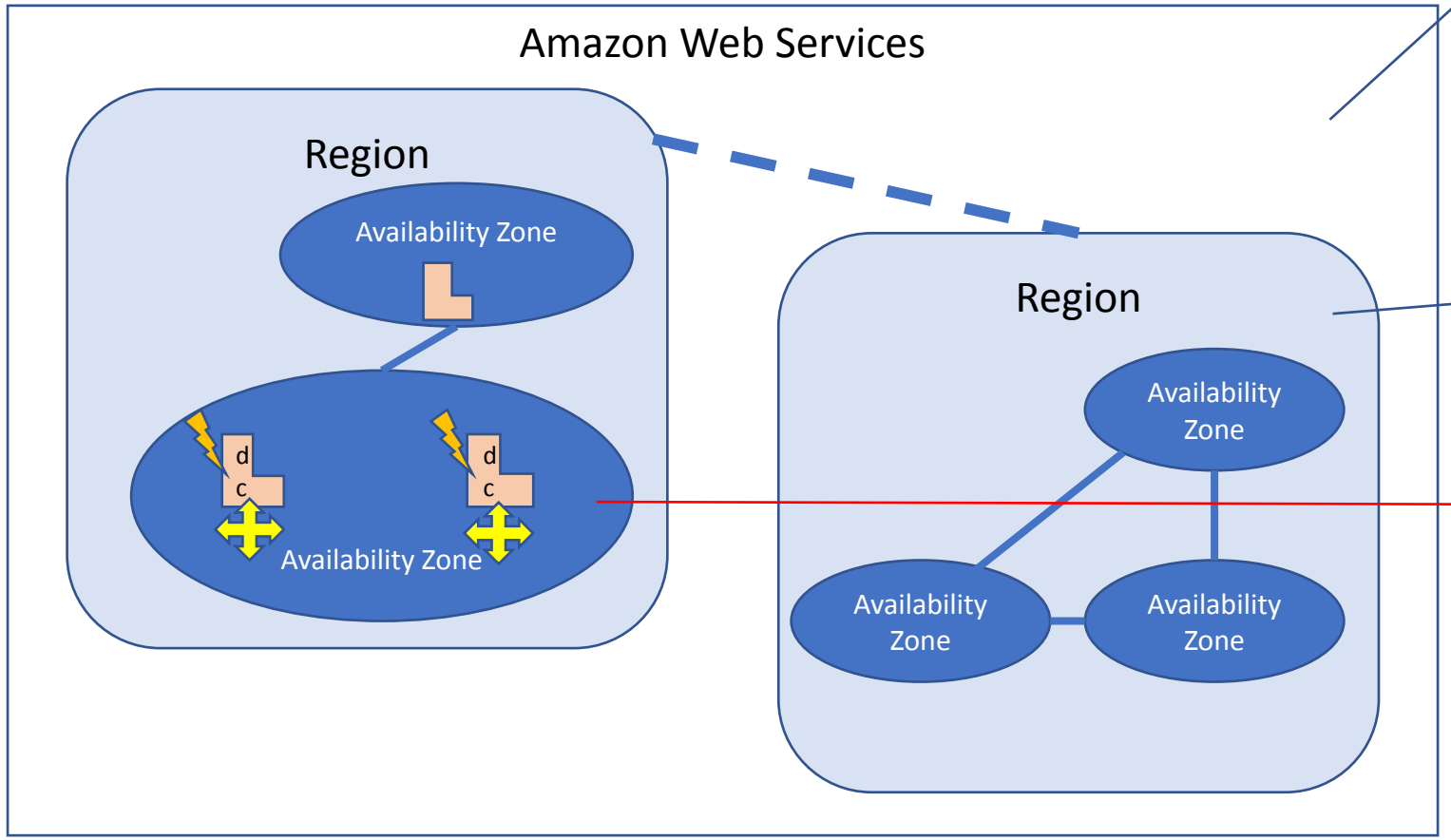
Service Administrator 1:1 Development 1:10 Co-Administrator

Service Administrator 1:1 Staging 1:10 Co-Administrator

Service Administrator 1:1 Production 1:10 Co-Administrator

BUILDING BLOCKS

Data Centres Technology - AWS



Global scope (eg. AWS Account)

Regional scope (eg. EC2 Instance ID)

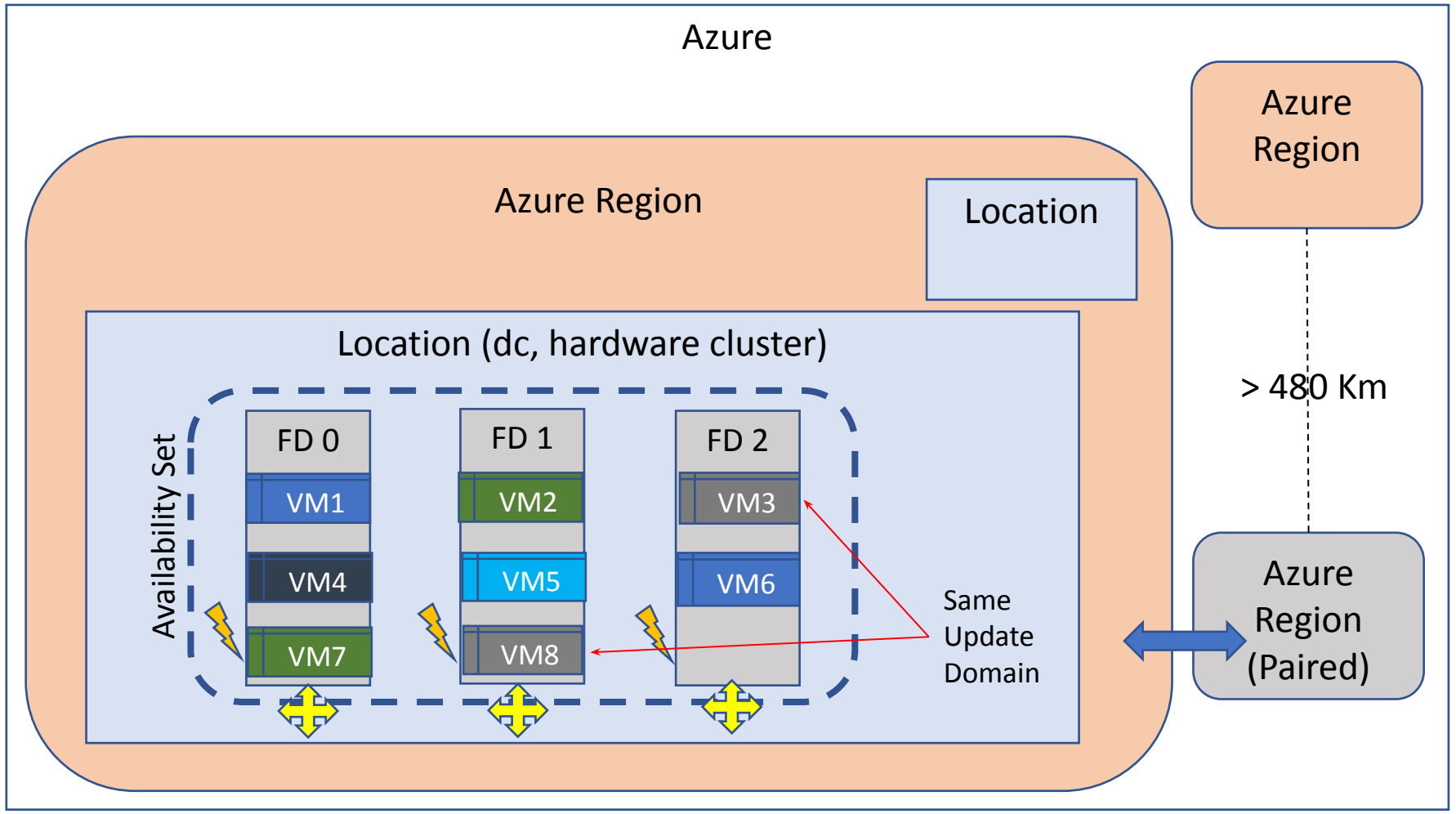
- Regions are completely isolated from other regions
- Greatest possible fault tolerance & stability
- Will not replicate resource across regions automatically
- Charge for data transfer across regions

with fast, private FO networking.

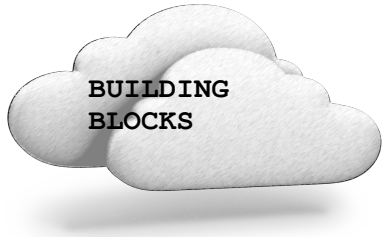
- Distribute instances across AZs for continuity of application.

BUILDING BLOCKS

Data Centre Technology - Azure



- Storage data is triplicated in region and it's paired region
- Availability set is created by user. SLA of 99.95% holds only if two or more VMs are put in an AvSet.
- Fault Domains – up to 3 per AvSet.
- Update Domains – up to 5 (20 via RM) per AvSet.



Quiz time!

Q. The SQL database instance in cloud is throttling. I move it from Standard tier to Premium tier. This move is an example of Horizontal or Vertical scaling?

A. Vertical scaling.

Q. I need to create my resume for placement interviews. I run MS Word online through Office 365 subscription to create my resume. This is an example of IaaS/PaaS/SaaS?

A. SaaS.

Q. I provisioned (created) an VM instance in my AWS cloud for installing a web server. This is an example of IaaS/PaaS/SaaS?

A. IaaS

Q. My company has multiple AWS accounts for running its business in AWS cloud. So many accounts are causing access and tracking confusions. My boss just asked me to look for a solution in AWS. What is that solution?

A. Use AWS organizations to create OUs & assign SCPs and make one AWS account as master account for that organization. The billing then will happen against that master AWS account.

Q. In Azure, an Azure subscription can have multiple Azure accounts. True or false?

A. False. An Azure account can have multiple subscriptions but not vice versa.

Q. In Azure, I create my Azure account and then created a VM under it. All good. True or False?

A. False. Without first creating a subscription, no resources can be created under an Azure account.

Exercise

For a web application, following are the resources that you are creating in Azure: Two servers, S1 & S2, for serving HTTP requests, two data base servers Db1 and Db2 for the databases and B1 & B2 servers for

processing back-end/business jobs for the service. How will you distribute these machines in availability sets, fault and upgrade domains to make a system resilience to failure?

Resource Management in Azure

Azure Resource Manager is the deployment and management service for Azure.

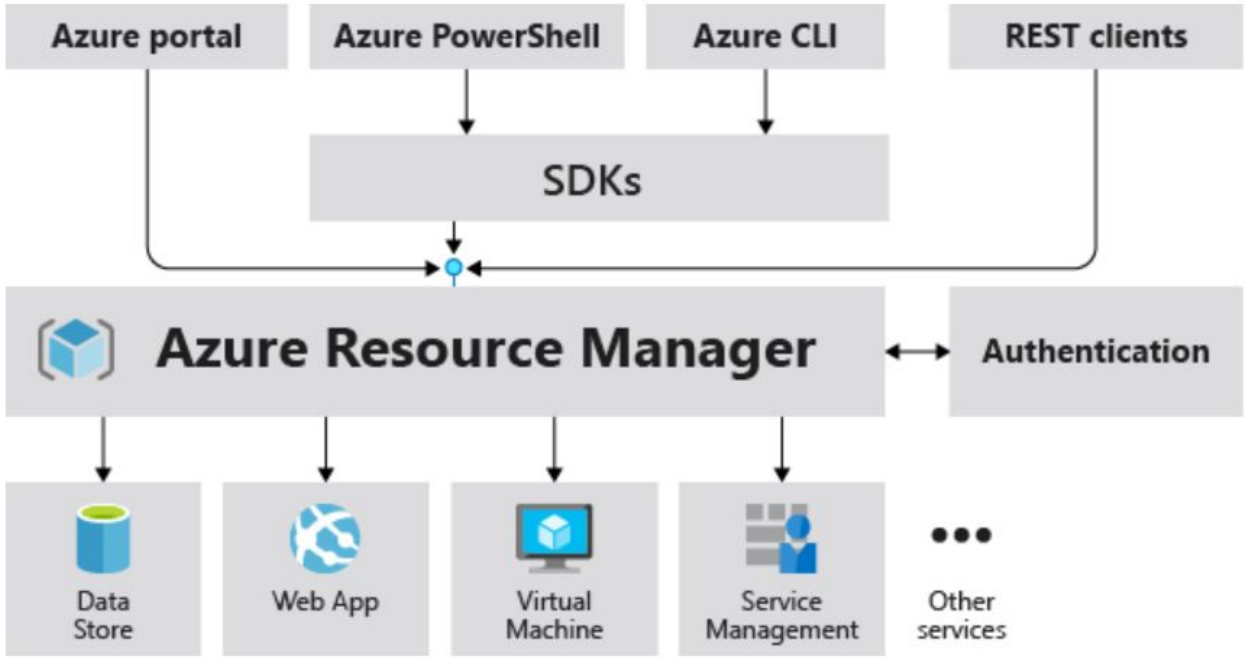
It provides a management layer that enables you to create, update, and delete resources in your Azure account.

You use management features, like access control, locks, and tags, to secure and organize your resources after deployment.

Consistent management layer

When a user sends a request from any of the Azure tools, APIs, or SDKs, Resource Manager receives the request.

When a user sends a request from any of the Azure tools, APIs, or SDKs, Resource Manager receives the request.



Terminology

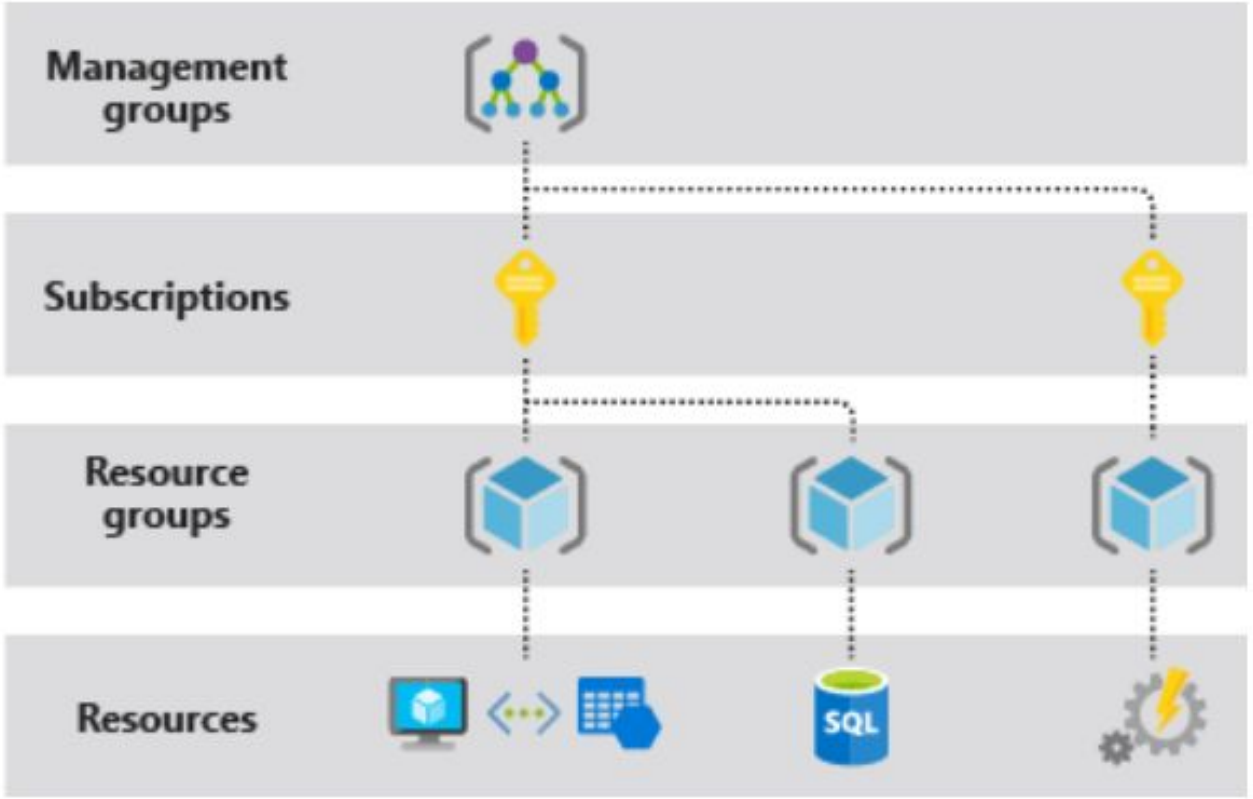
- **resource** – A manageable item that is available through Azure. Virtual machines, storage accounts, web apps, databases, and virtual networks are examples of resources. Resource groups, subscriptions, management groups, and tags are also examples of resources.
- **resource group** – A container that holds related resources for an Azure solution. The resource group includes those resources that you want to manage as a group.
- **resource provider** – A service that supplies Azure resources. For example, a common resource provider is Microsoft.Compute, which supplies the virtual machine resource. Microsoft.Storage is another common resource provider.
- **Resource Manager template** – A JavaScript Object Notation (JSON) file that defines one or more resources to deploy to a resource group, subscription, management group, or tenant. The template can be used to deploy the resources consistently and repeatedly.

Understand scope

Azure provides four levels of scope: management groups, subscriptions, resource groups, and resources.

You apply management settings at any of these levels of scope

Lower levels inherit settings from higher levels.



Resource groups

- All the resources in your resource group should share the same lifecycle. You deploy, update, and delete them together. If one resource, such as a server, needs to exist on a different deployment cycle it should be in another resource group.
- Each resource can exist in only one resource group.
- The resources in a resource group can be located in different regions than the resource group.
- When creating a resource group, you need to provide a location for that resource group.
- A resource group can be used to scope access control for administrative actions.
- A resource can connect to resources in other resource groups. This scenario is common when the two resources are related but don't share the same lifecycle. For example, you can have a web app that connects to a database in a different resource group.
- When you delete a resource group, all resources in the resource group are also deleted.
- Some resources can exist outside of a resource group. These resources are deployed to the subscription, management group, or tenant. Only specific resource types are supported at these scopes.

AZURE AUTOMATION USING AZ CLI

az cli

- The CLI is a tool designed to get you working quickly and efficiently with Azure services, with an emphasis on automation
- Works on Windows, Mac OS, Linux
- Can also be run in Azure portal as cloud shell
- Sign In
  - az login

Common Commands

- Commands in the CLI are organized as *commands* of *groups*.
  - Each group represents an Azure service, and commands operate on that service.

Resource type	Azure CLI command group
Resource group	az group
Virtual machines	az vm
Storage accounts	az storage account
Key Vault	az keyvault
Web applications	az webapp
SQL databases	az sql server
CosmosDB	az cosmosdb

Globally available arguments

--**help** prints CLI reference information about commands and their arguments and lists available subgroups and commands.

--**output** changes the output format. The available output formats are json, jsonc (colorized JSON), tsv (Tab-Separated Values), table (human-readable ASCII tables), and yaml. By default the CLI outputs json. To learn more about the available output formats, see Output formats for Azure CLI.

--**query** uses the JMESPath query language to filter the output returned from Azure services. To learn more about queries, see Query command results with Azure CLI and the JMESPath tutorial.

--**verbose** prints information about resources created in Azure during an operation, and other useful information.

--**debug** prints even more information about CLI operations, used for debugging purposes. If you find a bug, provide output generated with the --debug flag on when submitting a bug report.

Creating a virtual machine using az cli

- az login
- In Azure, all resources are allocated in a resource management group.
  - Resource groups provide logical groupings of resources that make them easier to work with as a collection.
- Create a resource group
  - az group create –name mscitrg –location centralindia
- Create a vm under resource group
  - az vm create --resource-group mscitrg --name testvm1 --image ubuntults --admin-user lavneetsingh --generate-ssh-keys --verbose --output json

- az vm show --name testvm1 --resource-group mscitrg
- az group delete --name mscitrg --no-wait
- az group wait --name mscitrg --deleted

Infrastructure as Code

- With the move to the cloud, many teams have adopted agile development methods. These teams iterate quickly
- They need to repeatedly deploy their solutions to the cloud, and know their infrastructure is in a reliable state
- As infrastructure has become part of the iterative process, the division between operations and development has disappeared
- Teams need to manage infrastructure and application code through a unified process (**DevOps**)

- To meet these challenges, you can automate deployments and use the practice of infrastructure as code.

- In code, you define the infrastructure that needs to be deployed. The infrastructure code becomes part of your project.

- Just like application code, you store the infrastructure code in a source repository and version it.

- Any one on your team can run the code and deploy similar environments.

ARM Templates – Infrastructure as Code in Azure

- The template is a JavaScript Object Notation (JSON) file that defines the infrastructure and configuration for your project.
- In the template, you specify the resources to deploy and the properties for those resources.
- Hello World template:

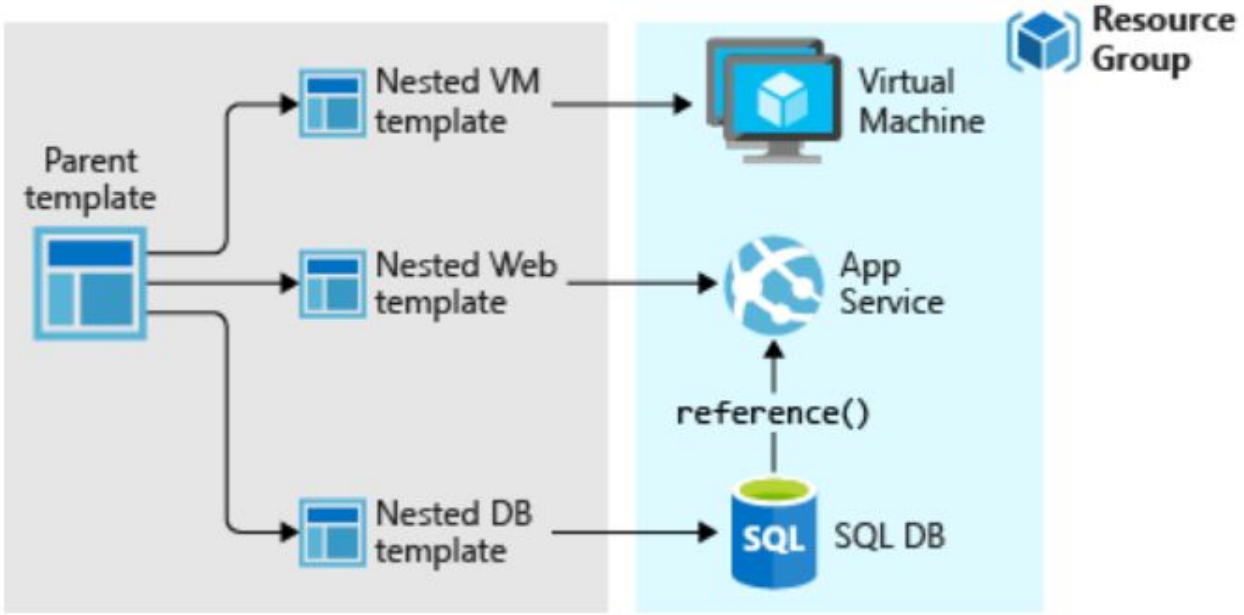
```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTemplate.json#",
    "contentVersion": "1.0.0.0",
    "resources": []
}
```

Template files

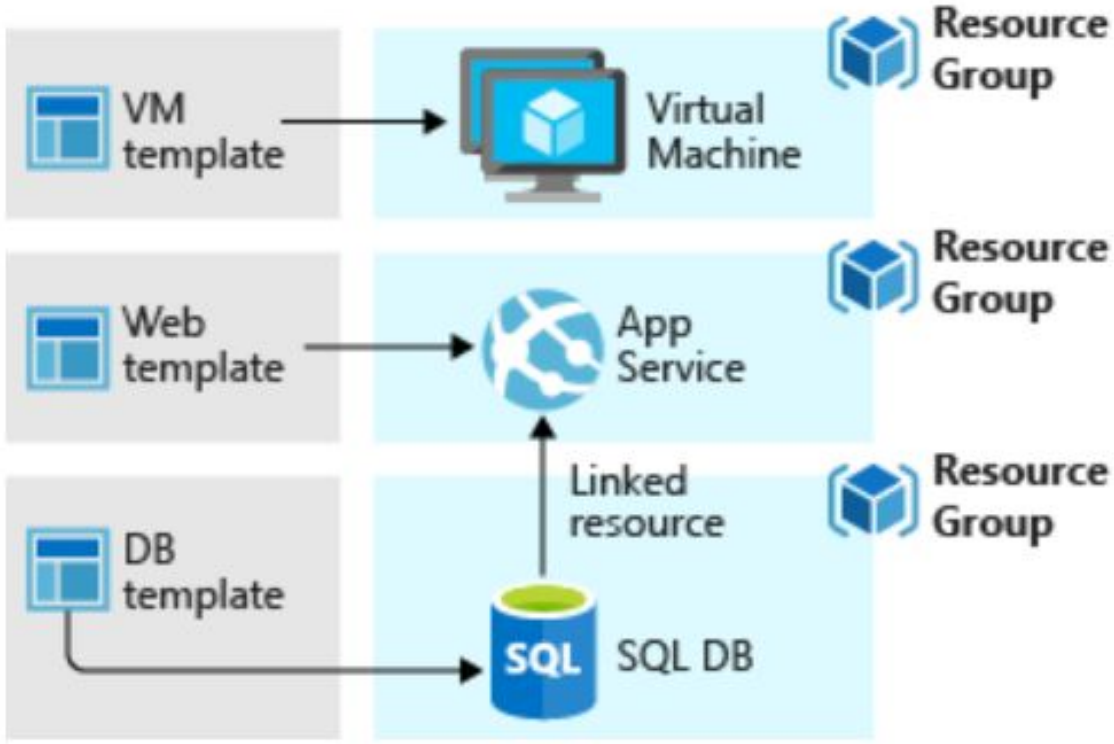
- Within your template, you can write template expressions that extend the capabilities of JSON. These expressions make use of the functions provided by Resource Manager.
- The template has the following sections:
  - **Parameters** - Provide values during deployment that allow the same template to be used with different environments.
  - **Variables** - Define values that are reused in your templates. They can be constructed from parameter values.
  - **User-defined function**s - Create customized functions that simplify your template.
  - **Resources** - Specify the resources to deploy.
  - **Outputs** - Return values from the deployed resources.

Template Design

- You don't have to define your entire infrastructure in a single template.
- Often, it makes sense to divide your deployment requirements into a set of targeted, purpose-specific templates
- To deploy a particular solution, you create a master template that links all the required templates.



- If you envision your tiers having separate lifecycles, you can deploy your three tiers to separate resource groups.
- The resources can still be linked to resources in other resource groups

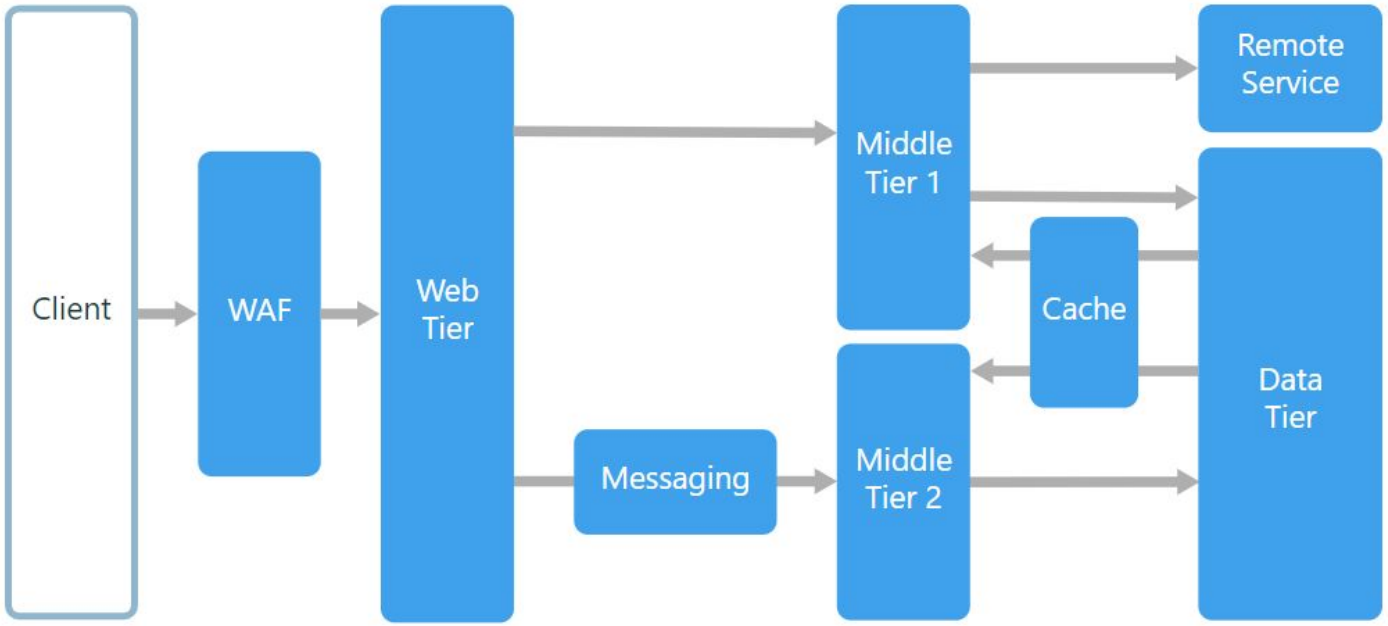


Cloud Architectures

N-Tier	• Traditional layered • Higher layer calls lower ones, not vice-versa • Can be a liability – not easy to update individual components • Suitable for app migration • Generally IaaS
Web-Queue-Worker	• Purely PaaS • Web FE + Worker BE • FE <-> BE communication through async msg Q • Suitable for simple domains
Microservices	• Many small independent services • Services are loosely coupled • Communicate through API contracts
Event-driven Architecture	• Uses pub-sub model • Pubs & Subs are independent • Suitable for apps that ingest large volume of data • Suitable when subsystems need to perform different actions on same data
Big Data Big Compute	• Specialized architecture that fit certain profiles • Divides large datasets into chunks, parallel process and analysis

N-tier Architecture Style

- An N-tier architecture divides an application into **logical layers** and **physical tiers**.
- Layers are a way to separate responsibilities and manage dependencies.
- Tiers are physically separated, running on separate machines.

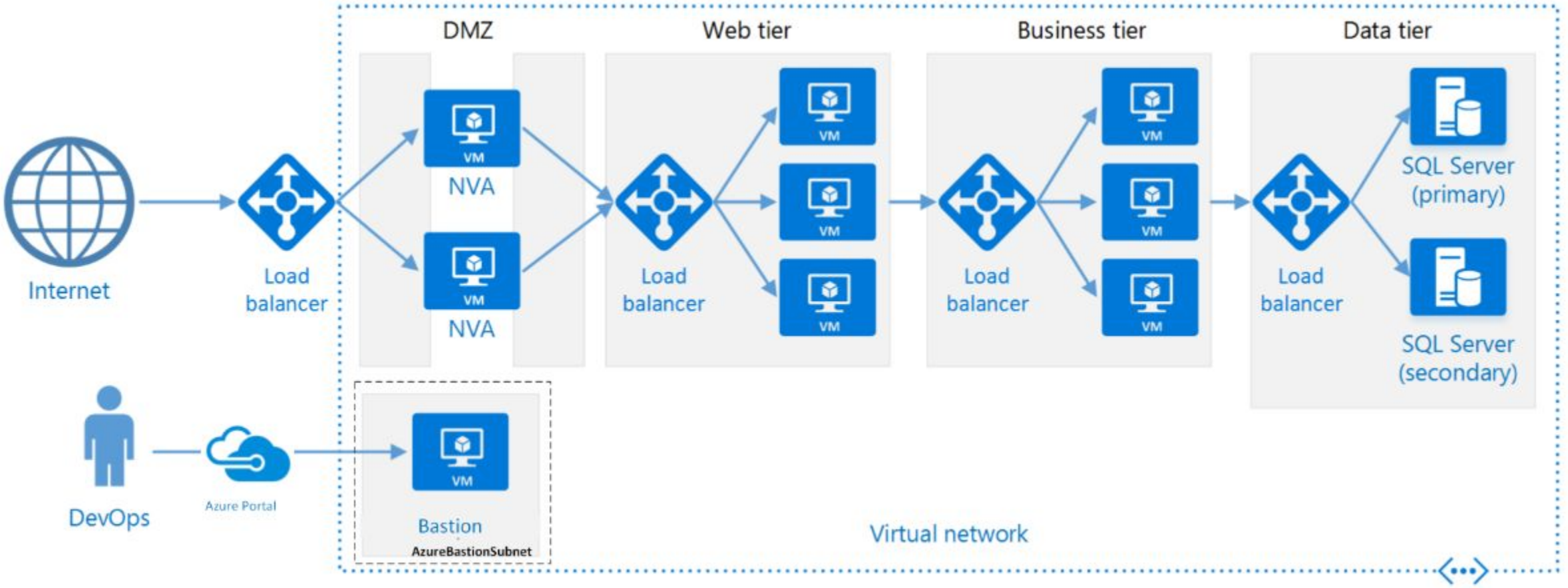


When to use this architecture

- Typically implemented as infrastructure-as-service (IaaS) applications, with each tier running on a separate set of VMs.
- Consider an N-tier architecture for:
  - Simple web applications.
  - Migrating an on-premises application to Azure with minimal refactoring.
  - Unified development of on-premises and cloud applications.

- **Best practices**
- Use autoscaling to handle changes in load. See Autoscaling best practices.
- Use asynchronous messaging to decouple tiers.
- Cache semi static data. See Caching best practices.
- Configure the database tier for high availability, using a solution such as SQL Server Always On availability groups.
- Place a web application firewall (WAF) between the front end and the Internet.
- Place each tier in its own subnet, and use subnets as a security boundary.
- Restrict access to the data tier, by allowing requests only from the middle tier(s).

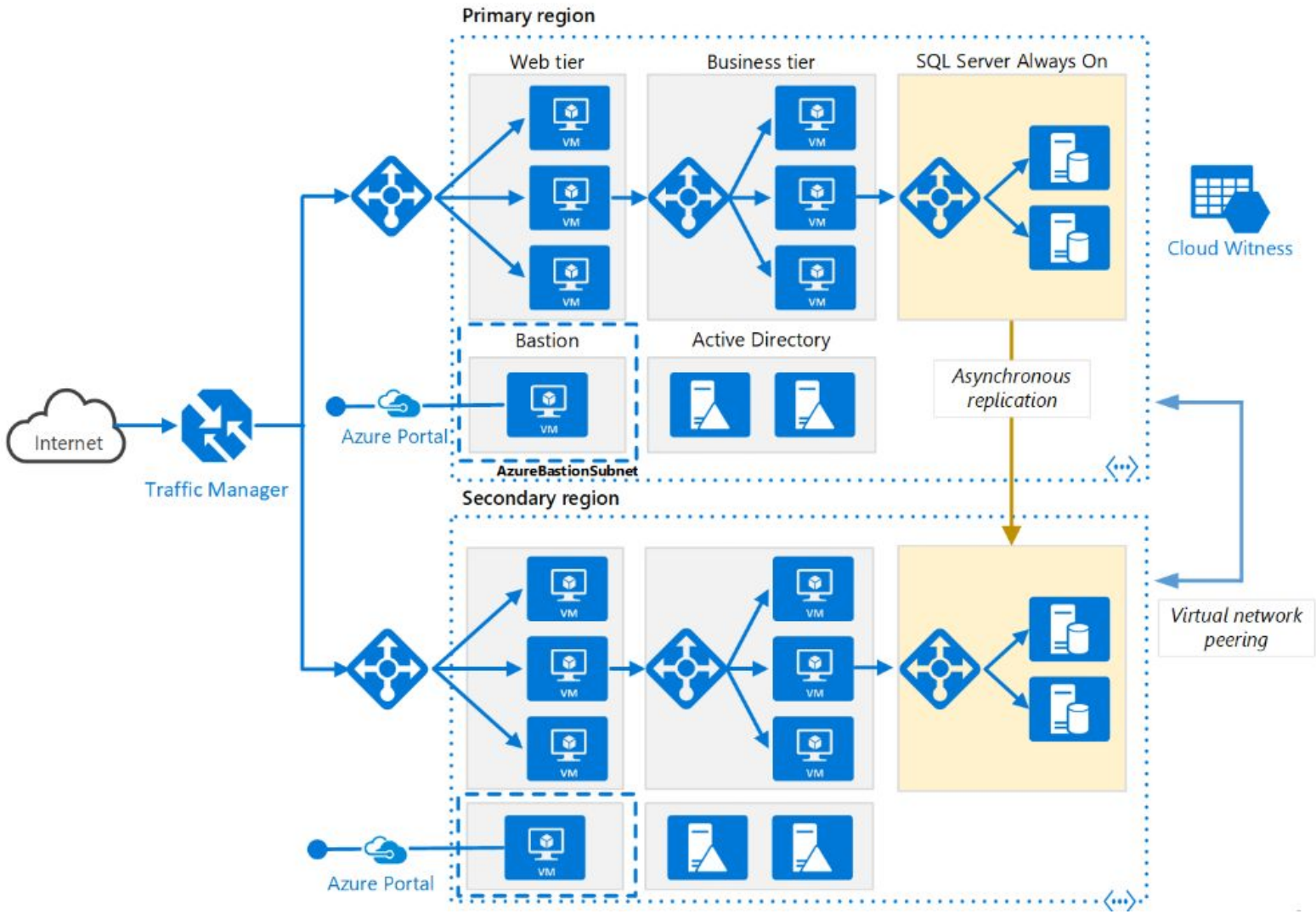
N-tier architecture on Virtual Machines



- Each tier consists of two or more VMs, placed in an availability set or virtual machine scale set.
- Multiple VMs provide resiliency in case one VM fails.
- Load balancers are used to distribute requests across the VMs in a tier.
- A tier can be scaled horizontally by adding more VMs to the pool.
- Each tier is also placed inside its own subnet, meaning their internal IP addresses fall within the same address range.
- That makes it easy to apply network security group rules and route tables to individual tiers.
- The web and business tiers are stateless. Any VM can handle any request for that tier. The data tier should consist of a replicated database.
- Network security groups restrict access to each tier. For example, the database tier only allows access from the business tier.

Role of Application Gateway

- Web traffic load balancer enables us to manage traffic to our web applications
- Operates at application level (OSI level 7 - Application).
  - Traditional load balancers operate at transport layer (OSI level 4 – TCP & UDP) and route traffic based on IP address and port.
  - Application Gateway can route traffic based on URL
- SSL termination
  - supports terminate encryption at gateway, so data travels un-encrypted to the backend servers. Saves overhead.
- Web application firewall – protection against common exploits
- Multiple site hosting, Redirection, Session affinity etc.



N-tier architecture – Multi-region deployment

Role of Traffic Manager in Multi-region deployment

- Traffic Manager allows users to control traffic distribution across various application endpoints.
- The Traffic Manager distributes the traffic using the configured routing method
  - Priority, Weighted, Performance, Geographic
- Constant monitoring of the endpoint health.
- Automatic failover in case the endpoints fail.
- Functions at the DNS level. It uses the DNS to route clients to precise service endpoints, according to the traffic-routing rules and methods.
- Traffic Manager operation modes:
  - Active | Passive – hot failover
  - Active | Passive – cold failover
  - Active | Active – load balanced