# AZ300-something Notes

**Azure Policy:** eg. only deploy to West EU in Resource Group 1

Export Azure diagnostic log data to:

* Azure Storage
* Event Hub: subscribe to event data funneled into the hub
* Azure Monitor logs: Kusto Query Language (KQL) to query

**Logs** are **blob** objects,  
**Metrics** (measured values, counts representing time series 🡪 they form the baseline) as **tables**.

Configure log collection for public IP address resource:  
- Monitor  
- Settings, Diagnostic settings  
- Add diagnostic setting: logs/metrics are resource specific (eg DDoS logs for IP)  
(we could have done this via Resource Manager API, Azure CLI or PowerShell)

**Alerts:**  
- Azure Monitor  
- Alerts  
 New Alert Rule (we can select from multiple Azure subscriptions)  
 - **Trigger:** can be either **Metrics** or **Activity Log** (listen to particular events)  
 Actions, Add: send email, sms, or trigger Function or, Logic App, send webhook, start Azure  
 Azure Automation runbook or integrate with our IT Service Management platform

We can even create a KQL query and use it in an Alert as a rule

**Action groups:** take action automatically when an alert is fired  
Generally, best practice is to create Action Groups first, then reuse them later on in alert definitions  
- Azure Monitor  
- Manage Actions, Add Action Group (specify subscription and resource group)

**Log Analytics:** collect Activity logs from multiple subscriptions (KQL)

**Log Analytics workspace:** data warehouse, to which we can link Azure resources, they send there their data

Unused Resources: big cost saving opportunity  
**Azure Advisor** -> Cost recommendations

Price (historic data) 🡪 Cost Management + Billing  
Price (forecast) 🡪 Azure Monitor + Usage and Estimated Costs

**Storage account:** name must be globally unique (each service is bound default to public Azure Resource Manager API endpopint like http://account-name.blob.core.windows.net

* General purpose v2 (blob, table [NoSql], queue, file)
* General purpose v1 (backwards compatibility only, has no tiers nor zones)
* Blob storage (backwards, only stores VM virtual hard disks – nowadays put them in managed disk storage instead)

Hot tier: discount on transaction cost (data frequently accessed)  
Cool tier: discount on data storage (data not frequently accessed)  
Archive: only access data on special occasions (we must rehydrate blobs first which have cost)

**Locally Redundant Storage (LRS):** 3 copies of Storage Account, in single datacenter  
**Zone Red. Stor.:** 3 copies, same region, but different datacenters  
**Geo Red. Stor.:** 3 copies in home region, 3 copies in second (paired) region [these are close to each other]  
**Read access geo red. Stor.:** Same as geo redundant, but we can access the 2ndary region, it has a unique URL <account-name>.secondary<service>.windows.net

**Access keys:** 2, 512 bit interchangeable access key for Storage Accounts (can be in Key Vault)  
SAS: **Shared Access Key**:  
- Protocol (HTTPS)  
- Address (fully qualified path of storage acc)  
- Permissions (CRUD)  
- Time interval

**VM: availability set** can be set, but only during creation (wanna change? Redeploy VM)  
**Availability set** is a logical container, ensuring that VM instances reside on different hardware hosts, separate racks  
Not regional, only local level (planned, unplanned maintenance)

**Availability zone:** VMs in same region, but different datacenter (not available for every region)  
AVAILABILITY ZONES WITH AVAILABILITY SETS CANNOT BE COMBINED

Want different region for VMs? Use Traffic Manager  
Single instance SLA: use high speed premium storage for VM, then don’t have to bother with these, as it’s SLA is 99.9%

Resizing VMs restart them.

**VM monitoring:**  
1. Azure Diagnostics: deploy VMDiagnosticSettings agent to VM, collect OS metrics, logs  
2. Log Analytics: deploy Log Analytics agent, connect the VM to Log Analytics Workspace

Hard to scale IaaS (consider PaaS)  
**Scale sets:** scale a group of identical VMs

Low priority VMs: can be reclaimed by MS  
 - Workload should rely on external storage, and be stateless (which node does the work? Nvm)  
 - Jump box VM: manage scale from it, its placed on the scale set’s virtual network

**Use VHD** (virtual hard disk) **to create VM:**1.AzCopy, Azure Storage Explorer to upload onpremises vhd to blob  
2. Azure portal to capture a generalized VM image, store it in managed disks library  
**In the ARM template, use osDiskVhdUri**

**Azure Disk Encryption:** whole-disk encryption  
 Windows VMs: BitLocker Drive Encryption 🡪 KeyVault necessary (same region as VM)  
 Linux VMs: DM-Crypt library

**VNets:** security boundaries for Azure resources (if one VM runs in VNet1, it cannot have routing to VM in VNet2)

* Connect VNets:
  + VNet peering
  + VNet-to-VNet virtual private network (VPN)

**VNet peering:** cheap, no need for VPN (no end-to-end encryption though, but it does go through an Azure network backbone)  
VNets can be in different subscriptions  
If Vnet1 is connected to Vnet2, and Vnet2 is connected to Vnet3, then Vnet1 cant communicate with Vnet3 🡪 VNet peerings are intransitive. This means that, by default, two spoke VNets are unable to communicate through a hub (transit) virtual network

**Name resolution with VNet peering can be tricky:  
- private Azure DNS zone:** nonroutable DNS zone, attach to both VNets  
**- DNS servers:** deploy DNS serve VM in both VNet, config each to forward name resolution requests to the other VNet’s name server

**Todo: look into UDRs and NVA** (network virtual appliance)

It’s not enough to setup a Vnet VPN, 🡪 NGS (network sec. group) blocking incoming/outgoing traffic, as well as VM firewalls, etc

**Network Watcher:** test connectivity between vnets in Azure.  
in Azure, IP addresses are associated with virtual network interface cards, not virtual machines

Azure-provided name resolution enables you to resolve DNS hostnames for all resources within a single VNet.

Which Azure resources you can supply custom DNS server IP addresses:  
- Virtual Network (vnet)  
- Virtual Network Interface

**Service Endpoints:** enable us to secure Azure resources to particular vnets (Azure Storage, SQL db, KeyVault, etc)

more than one Azure subscription can trust a single Azure AD tenant  
a single subscription can be associated with only one Azure AD instance at a time

**Azure AD Connect**: sync on premises AD users identities to Azure AD tenant  
No need for ExpressRoute or site-to-site VPN (all communication over TCP443)  
Free Microsoft utility to support SSO between on-premises AD and Azure AD  
For this, we’d need custom domain for Azure AD tenant (not .onmicrosoft.com)

* Install Azure AD Connect on on premise domain member server
* **Password Hash Synchronization:** each on premise AD entity has a representation in the Azure AD (Azure Connect keeps passwords in synch, maintains password hashes for both)  
  vs  
  **Pass-Through Authentication:** no password hases kept Azure AD (but this requires deployment of agent software on premises)  
  vs  
  **Federation With AD FS:** Azure AD Connect automated deployment of AD Federation Service farm to support token based SSO.  
  vs  
  **Federation with PingFederate:** third party alternative for AD FS  
    
  Seamless Single sign on: created a computer account (AZUREADSSOACC) on premise. Local Kerberos for auth (Kerberos tickets, not password hashes) to travel between on premise and cloud  
  (to do these, we’d need Azure AD Global Admin, and local AD Enterprise Admin)
* How to change password here? Go to myapps.microsoft.com and do it here  
  THEN **PASSWORD WRITEBACK**: Azure Ad Premium P1, P2

**User types in Azure AD:**

* Azure Active Directory: cloud native
* External Azure Active Directory (B2B)
* Invited user: B2B guest user who haven’t yet accepted invite
* Microsoft account: B2B guest user who had to create Microsoft account at invite redeem
* Windows Server AD: local AD identity which has been synced up to the Azure AD tenant

**How to enable MFA?** Via an conditional access policy

AD Licenses are licensed per user: all involved users, admins must be assigned to appropriate Azure AD license (Licenses blade in Azure AD tenant)

**PIM:** Privileged Identity Management (eg. give admins just in time, time restricted admin access to Azure AD or resources)

Access review process: role assignment/membership review (collaborate, give a guy PIM to review his own team, etc)  
- **Access Reviews** blade: Azure AD groups, apps  
- **Azure AD Privileged Identity Management** blade: Azure AD, Azure resource **roles**

**IdP:** Azure AD Identity Protection (p2 feature) – detect potential vulnerabilities, configure automatic remediation for events

* MFA registration policy
* User risk policy: user behavior threshold triggers this, then blocks/enables access to Azure AD backed apps
* Sign-in risk policy: failed signing in 3 times? Blocked  
    
  ISG: Intelligent Security Graph: AI that tracks risk values for users

**Azure AD Join:** join Windows 10 endpoint devices to Azure AD tenant  
This is for endpoint management, not server management.

Azure AD joined devices may or may not be connected to on premises Ads, Group Policy is not supported in Azure  
**TODO: read up on this**

**Azuer Enterprise Roaming** (same, P1, P2): public cloud equivalent of roaming user profiles in on-premises AD: users from Win10 Azure AD Join’d devices can sync their users, settings to the cloud

* Azure AD 🡪 Devices 🡪 Enterprise State Roaming  
  **Users May Sync** 🡪 All/selected  
  **App Data Across Devices** 🡪 All/selected