AKS – Anatomy of Deployment steps

Day-0 Activities

Plan Network topology

- **Hub-n-Spoke Or Dedicated**
  - Preferred is *Hub-n-Spoke* as it makes entire topology flexible
  - In both the cases decide on the list of VNETs and Subnets to used (Existing or New). At the minimum follwoing is the list that are needed -

    VNET address space -

    - **AKS+ VNET** - /16 at least or more
    - **AKS subnet** - A completely dedicated Subnet for AKS cluster. No other resources to be planned on this.

      /21, 22 for Dev/UAT and **/18, /20** for Prod id safe to choose. *If Address space is an issue then Kubenet*. This should be a dedicated subnet for AKS cluster.

      The question that should debated at this stage are -

      - How many micro-services approximately to be deployed (*now* and *in future*)
      - What would the minimum and maximum no. of replicas for each
      - How much *CPU* and *Memory* that each micro-services could consume approximately
      - And based on all these –
        - What is Size of each *Node* (VM) i.e how many *Cores of CPU* and how much *GB of Runtime Memory*
        - how many *Nodes* (VMs) that the cluster could expect (*initially and when scaled up?*) – basically *minimum* and *maximum* no. of such *Nodes*
      - Finally *maximum* number of pods or app replicas you want in each *Node* – Ideally whatever be the size of the *Node*, this value should not go beyond 40-50; not an hard and fast rule but with standard VM sizes like 8 Core 16 GB, 40-50 *Pods* per *Node* is good enough Based on all these info, let us try to define a formulae to decide what would be the address space of VNET and Subnet for AKS.

      Let us assume,

      Np = Max. Number of Pods in each Node (Ideally should not be more ethan 40-50)

      Nd = Max. Number of Nodes possible (approx.)

      Then the total no. of addresses that you would need in AKS Subnet = ***(Np * (Nd + 1) + Np)***

+1 for reserved IP by system for each Node

+Np for additional IPs you might need while Upgrade – normally K8s system will pull down one Node at a time, transfer all workloads to another Node and then upgrade the previous Node

It is advisable to keep some more in buffer based on workloads and other unforeseen situations

What we have seen, for high end workloads, ideally for a *DEV-UAT* cluster, we should go with **/21 or /22** which means around 2k or 1k *Nodes*.

PROD cluster should have a bit more like **/18 or /20** which means around 16k or 4k *Nodes*.

- **ACR + KeyValut subnet** - This is to be used for Private endpoint for these two resources
- **APIM subnet** - One dedicated subnet for APIM. Two options are there - External VNET or Internal VNET.

  In case of *Internal* - the endpoint is completely private.

  In cade of *External* - the ingress to VNET is allowed by default and hence a proper NSG has to be added to control ingress access only from *Application Gateway*

  The address

  - *Same VNET or Separate VNET*? If one APIM across entire org then should be in a separate VNET and then peering with AKS VNET
  - *Address Space* - **/29** is enough for both DEV and PROD
- **Application Gateway+ VNET** - App G/W, Bastion Host
  - *Same VNET or Separate VNET*? If one App G/W across entire org then should be in a separate VNET and then peering
  - **/27** should be a good choice for Dev/UAT and Prod both. Select **Standard_V2** at least and **WAF_V2** if WAF service is needed as well
- **DevOps VNET** - Self-hosted devops agents - Linux or Windows
- **NSGs to be decided Upfront**
  - Decide basic NSGs for all the subnets
  - Some important *Inbound* rules to be followed -
    - App G/W allows communication only from Azure Front door
    - APIM should accept traffic only from App G/W
    - AKS subnet should accept traffic only from APIM and/or On-Prem VNET gateway
  - Some important *Outbound* rules to be followed -
    - AKS Subnet outbound would always use the public *Standard LoadBalancer* created during Cluster creatiopn process. To change that behaviour - add appripriate outbound rules and anyone of the following
      - Nat Gateway associated with AKS subnet
      - UDR through a Firewall Subnet
      - Forcing communication directly through web corporate proxy - this needs some amount scripting. Setting *http_proxy* or *https_proxy* can be deployed as a *Daemonset* on every AKS cluster node and force Nodes to send raffic through proxy

Plan Communication to Azure Services

- **Private/Service Endpopints**

  Plan to use for *Private Endpoints* (Or atleast *Service Endpoints*) wherever possible for communiction with Azure resources like Storage, SQL, CosmosDB etc. This makes communication secure and fast

- **Service Principal or AAD Pod Identity**
  - Service Principal with limited access to variopus Azure resources. This make sure any communication from within tthe PODs are all secured and legitimate within Azure

- **Service Principal for AKS cluster**
  - Allowing access to AKS subnet and various other azure resources; create Service Principal and Define roles to itt and assign it to designated azure resource(s) like VNET etc. Role should be as per the requirement and follow least access principal

Plan User Roles within AKS cluster

- 4 Primary roles to be considered - *Cluster Admi*n, *Architects*, *Managers* and *Developers*. Based on requirement more cna be thought of
- All 4 roles should be added as an Azure AD group
- These groups would be mapped onto AKS cluster post creation of the cluster process
- Actual *Role Assignment* and *Role Bindings* would happen on **Day-2**

Plan DevOps setup

- Self Hosted agent - This ideally should be Ubuntu machine with at least 2 Core 8 GB spec (Or Windows macin with similar configuration)
- Same agent machine can be used both as *Build Agent* and *Deployment Agent*
- The VNET (as described aloe) should peered with AKS VNET
- ACR/KeyVault VNETs should be peered with DevOps VNET to make sure that CI/CD pipeline can access ACR images and KeyVault keys during their run

Plan ACR setup

- Ideally two ACRs - one for DEV and one for PROD
- Private Emndpoint support
- DevOps VNET should be able to access the ACR Subnet
- No Admin access enabled; Service principal to be used for accessing it from within the cluster as well as DevOps pipeline(s)

Plan KeyVault setup

- Ideally One KeyVault - Managing the keys from different apps and environments with varying access
- Private Endpoint support
- DevOps VNET should be able to access the KeyVault Subnet. During CI/CD process all keys are downloaded from *KeyVault* and mapped to Application variables

Day-1 Activities

Approach

- 3 Step Approach - *Pre-Provision, Provision, Post-Provision*
- Each step would be sharing one script e.g. *PreConfig.ps1, Setup.ps1, PostConfig.ps1*
- Deployment scripts can be anything as per comfort and choice - preferred is *PowerShell* and *ARM template* combination as that is Native Azure; but Terraform can be anbother god option with more human-resdable scripts.In all the cases, the approach shouild be to decouple the entiere prcoess into 3 steps as described above

Scripts (ARM template apporach)

- **Folder Structure** -
  - **Deployments**
    - **SetUp** - Contains follwoing PowerShell scripts -
      - *PreConfig.ps1* - Sets up all Infra as decided on ***Day-0*** planning

        (*Associted Roles* - **Cluster Admin**)

        1. Creates Service Principal for AKS cluster
        2. Creates Service Principal for ACR
        3. Deploys entire Network setup using ARM templates
        4. Deploys ACR using ARM templates
        5. Deploys KeyVault using ARM templates
        6. Saves above two Service Principals into KeyVault for later use
        7. Assigns ***Network Contributor*** role to the AKS *Service Principal* on AKS VNET

      - *Setup.ps1* - Creates Cluster with all custom parameters

        (*Associted Roles* - **Cluster Admin**)

        1. Creates AKS cluster
        2. Uses AKS *Service Principal* from above step
        3. Uses AKS VNET and Subnet created in teh above step

4. Defines VM size - 4 Core 16 GB is standard; to be deicded baded on your workload

   (*As explained above in **Day-0** activities*)

5. Defines Max Nodes count - As explained above in **Day-0** activities

6. Defines Max Pods count - As explained above in **Day-0** activities. Again reiterating, this should ideally not go beyond *40 - 50 PODs per Node*; for better performance with Service Discovery within the cluster. Having more than that results in serious performance issues

7. Defines Node set type - this can be *AvailabilitySet* (*aka *AS*) or *VMSS*. *AS* provides - *Fault Domain* and *Update Domain* - so for HA you should go with *AS* support

8. Users can chose to go for *Availability Zone* (*aka AZ*) support as well (*If the region is included in our supported regions list*)

9. Enables AddOns like - *Monitoring*, *VIrtual Nodes* etc.

10. Defines Azure AD credentials (*from **Day-0** activities*). This would be used for RBAC within the cluster

11. Enables *Cluster AutoScaler* for automatic horizontal Node scaling

- *PostConfig.ps1* - Setsup Cluster created above with additional tools and services (*if needed*)

  (*Associted Roles -* **Cluster Admin**, **Architects**)

  1. Installs Ingress Controller - NGINX is installed by default (using Helm chart) but script can be easily modified to install other such tools. Ingress is deploed as an ILB (*Internal Load Balancer*).

     The Private IP of ILB can be then added as a RecordSet through a Private DNS Zone to get a custom domain name

  2. Deploys APIM which points to the Private IP or Custom Domain name to call backend APIs. APIms hould define a Health service to check backend status and returns some response (*HTTP status codes between 200 - 399*)

  3. Deploys Application Gateway with WAF_V2 support.This can be mdofied to use Standard_V2 tier also.

     Application Gateway would have Private IP of APIM as in the backend pool. The health service as above would be used a Health probe by App G/W to monitor the backend and keep the links alive

- **Templates**

  1. Contains all the ARM templates for - *Virtual Network, ACR, KeyVault, Application Gateway*
  2. The Master PowerShell scripts (*as described above*) in the Setup folder would be calling each of these templates and deploy resources 1-by-1
  3. For terraform scripts, this folder can be used for storing terraform templates for each resource

- **YAMLs**
    1. Contains primarily 2 folders - *Cluster Admin, RBAC, Ingress*
    2. **Cluster Admin**
        - Contains scripts for Adding ***Cluster Admin*** group created as part of ***Day-0*** activities
        - Contais scripts for defining *Persisten Volume(s)* (*as discussed in **Day-0** activities*)
    3. **RBAC**
        - Contains RBAC roles and bindings definition for ***Architects*** group created as part of ***Day-0*** activities
        - Contains RBAC roles and bindings definition for ***Developers*** group created as part of ***Day-0*** activities
    4. **Ingress**
        - Contains *Ingress* object as a sample
        - Install One *Ingress* object per NameSpace
        - Define *Service Routes* as per requirements

Day-2 Activities

Cluster Hardening

(*Associted Roles -* **Architects, Managers, Developers(?)**)

- **Network Policy** (*East-West Traffic*)
    1. NetPol folder under YAMLs folder (above) would contain sample Network Policy file.This can be modified crested more specific policies
    2. Define Ingress policy for each micro service (aka tyoe of PODs)
    3. Define Egress policy for each micro service (aka tyoe of PODs)
    4. Include any socaial IPs to be allowed
    5. Exclude any IPs to Disallowed

- **Secrets**
    - Create All Secrets needed by the micro services
    - Majorly Docker Secrets, Storage Secrets, DB Secrets etc.

- **Azure Policy**
    1. Go to Azure Portal and Select Policy
    2. Filter by Kubernetes keyworkd
    3. Add All relevant built-in policies on the cluster