#Secure Baseline cluster

Ref : https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/containers/aks/secure-baseline-aks

What's needed

Networking Topology

Ref: https://github.com/mspnp/aks-secure-baseline/blob/main/networking/topology.md

- Prefer hub/spoke network topology.
- Multiple advantages of this model
    - Shared resources across networks - firewall and DNS. Save $
    - Extendable architecture - new workloads can be added to new spokes instead of updating existing network topology
    - Mandate a WAF (web application firewall) service to govern HTTP traffic

AKS will be deployed to either a new or an existing spoke. A typical breakdown is /18(16382 ips) or /22 (1022 ips) + /27 (for ingress) for the cluster nodes and ingress. Keep in mind that under the CNI networking model for AKS - all pods and nodes will have an IP address. You will also need to account for

Upgrades - AKS updates nodes regularly to make sure the underlying virtual machines are up to date on security features and other system patches. During an upgrade process, AKS creates a node that temporarily hosts the pods, while the upgrade node is cordoned and drained. That temporary node is assigned an IP address from the cluster subnet.

For pods, you might need additional addresses depending on your strategy. For rolling updates, you'll need addresses for the temporary pods that run the workload while the actual pods are updated. If you use the replace strategy, pods are removed, and the new ones are created. So, addresses associated with the old pods are reused.

Scalability Because AKS is dynamic, as a best practice you want to make use of that elasticity either by using HPA or node scaleout. Suppose you want to scale out by 200%. You'll need two times the number of addresses for all those scaled-out nodes.

Cluster Compute

Deploy Ingress resources

What type of ingress do you want or need? We see three different flavors

1. Nginx/Traefik - i.e an ingress controller deployed within the cluster which is typically fronted by an internal azure load balancer. The Azure LB is outside the cluster and forwards the traffic to the ingress controller. The ingress controller can or cannot terminate SSL. MS recommeds having an app gateway in front of the LB (app gateway shared across other services so it can potentially be in the hub network) which has WAF enabled.
2. App Gatewway Ingress - this is AKS specific but you can have a dedicated app gateway for your cluster and not have to deal with separate ingress controller. MS says this is faster because it talks to the pods

directly and does not use the cluster resource (by virtue of it being outside the cluster). One thing to point out is that if you terminate TLS here, traffic is unencrypted from the app gateway to your cluster (its is still in your private network but its uncrypted as opposed to 1 above where its encrypted till it hits the subnet where the ingress controller is)
3. Service Mesh eg Istio - offers much more than a regular ingress controller (albeit at a cost) but you would still require an app gateway in front for security.

If using 1 and 3 - consider using `podAntiAffinity` to spread out the load and ensure business continuity.

TODO : https://github.com/kubernetes/community/blob/master/contributors/design-proposals/scheduling/podaffinity.md#anti-affinity `{{LabelSelector: <selector that matches S's pods>, TopologyKey: "node"}`

Egress traffic flow

https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/containers/aks/secure-baseline-aks#egress-traffic-flow

For zero-trust control and the ability to inspect traffic, all egress traffic from the cluster moves through Azure Firewall. You can implement that choice using user-defined routes (UDRs). The next hop of the route is the private IP address of the Azure Firewall. Here, Azure Firewall decides whether to block or allow the egress traffic. That decision is based on the specific rules defined in the Azure Firewall or the built-in threat intelligence rules.

Pod-to-pod traffic

By default, a pod can accept traffic from any other pod in the cluster. Kubernetes `NetworkPolicy` is used to restrict network traffic between pods. Starting out, maybe restrict communication between namespaces and avoid between pods in same namespace - use namespaces as your segregation boundary...similar to using resource groups versus resource tag in most (not all) cases.

> Enable network policy when the cluster is provisioned because it can't be added later.

-Azure Network Policy is recommended, which requires Azure Container Networking Interface (CNI), see the note below.

- Calico Network Policy, a well-known open-source option. Consider Calico if you need to manage cluster-wide network policies. Calico isn't covered under standard Azure support.

Container image reference

Policy management

Identity Management

Integrate Azure AD for the cluster

Integrate Azure AD for the workload

Secure data flow

Secure the network flow

Add secret management

Business continuity

Scalability

Cluster and node availability

Availability and multi-region support

Operations

Cluster and workload CI/CD pipelines

Cluster health and metrics

Cost management and reporting