Microsoft Azure

Azure Kubernetes Service Summit

Managing stateful workloads in AKS

Mohammad Nofal

Cloud Native Apps Architect – Global Black Belt Team – EMEA

@mohmd_nofal

Agenda

- The basics
- Performance
- Security
- Monitoring
- Resiliency

The Basics

What is a stateful applicaiton?

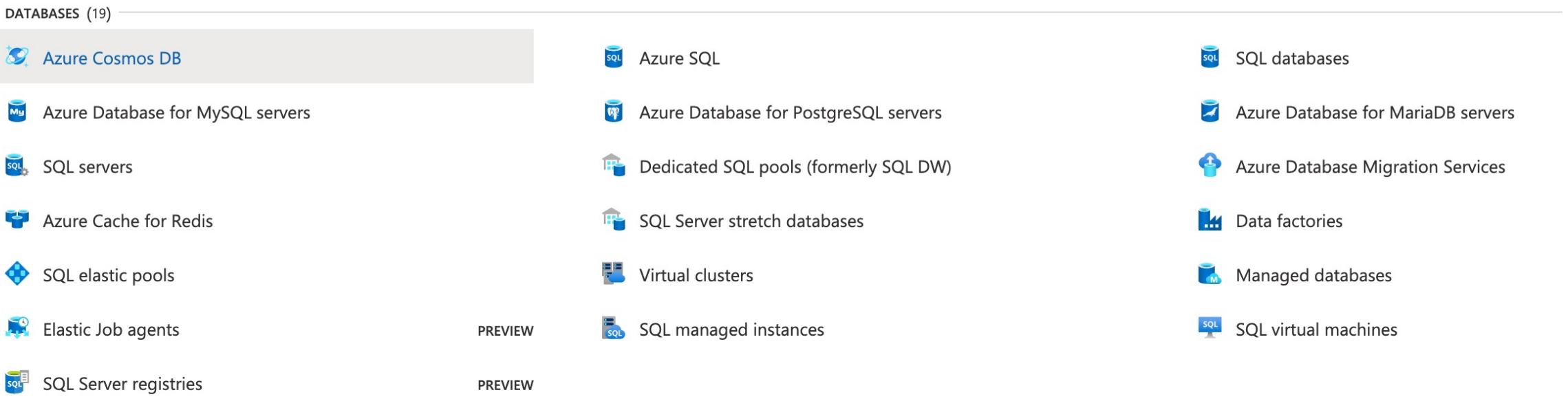
- Stateful applications
  - Applications that save data from the activities of one session so it can be used in the next session

- Examples
  - Mongo, Kafka, Cassandra, ElasticSearch, MySQL, etc..

The Essentials

- Avoid building your own stateful data service if you can
- Your priority is to make use of existing data/stateful services offered as a PaaS in Azure or the extensive partner eco-system in the marketplace
- Building and Managing your own stateful applications is expensive operationally

Azure Database Services



When do customers build their own stateful data services?

- There is no managed/PaaS service offered
- They require more control on their service which the PaaS service doesn’t offer
- More control always comes with an operational cost, always weight the Cost VS Benefits

Should I do stateful data services inside K8s or VMs?

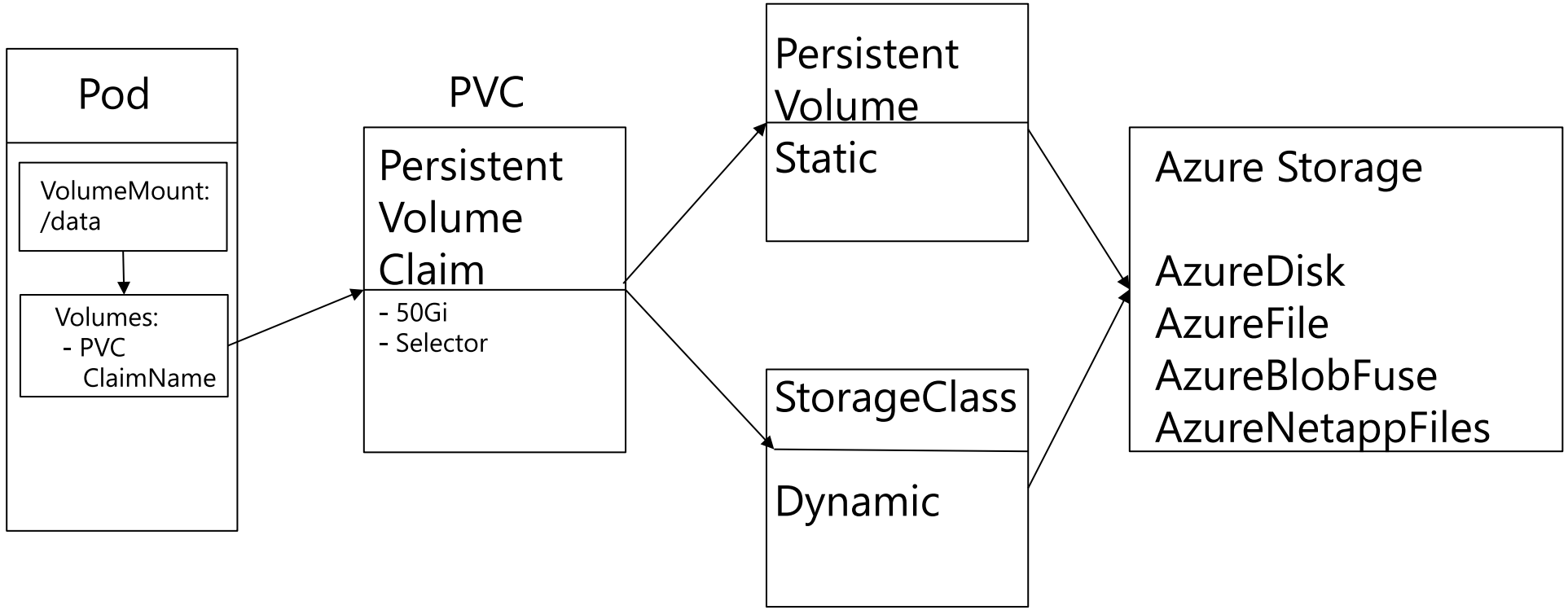
- Kubernetes and Kubernetes stateful sets offer many features that make it easier to run stateful applications comparing to plain VMs
- Many organizations are standardizing their operations on top of Kubernetes, as such building on the existing skillset
- Kubernetes Operators are maturing

Kubernetes and Azure Storage Basics

Kubernetes Access Modes and Azure Storage Options

Storage Type	Use Case	Example	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
Azure Disk	Structure Data	Database	✓		
Azure File	Shared Configuration	Content Management System	✓	✓	✓
Azure NetApp Files	Shared Configuration	Content Management System	✓	✓	✓
Azure Blob	Unstructured Data which require simple filesystem operations	Images	✓	✓	✓
Local VM Disk (Temporary Disk or NVME)	Distributed Storage Systems	FoundationDB	✓		
3rd Party Solutions	Depends	Depends	✓	✓	✓

How to request peristent storage in Kubernetes?



AKS Storage Classes

- 4 storage classes are offered by default

```
$ kubectl get storageclasses
NAME                PROVISIONER                RECLAIMPOLICY VOLUMEBINDINGMODE ALLOWVOLUMEEXPANSION AGE
azurefile           kubernetes.io/azure-file Delete         Immediate            true                 11d
azurefile-premium   kubernetes.io/azure-file Delete         Immediate            true                 11d
default (default)   kubernetes.io/azure-disk Delete         WaitForFirstConsumer true                 11d
managed-premium     kubernetes.io/azure-disk Delete         WaitForFirstConsumer true                 11d
```

CSI Drivers

- Default as of K8s 1.21, currently the feature require registration in AKS
- CSI is out of tree, meaning Azure can apply changes to the plugins without touching the k8s core code base
- Offers new features
  - Volume Snapshots
  - Volume Clones
  - Resize Volumes

Kubernetes StatefulSets

- provides guarantees about the ordering and uniqueness of deployed pods
- maintains a sticky identity for each of its pods
- when to use
  - stable, unique network identifier
  - stable, persistent storage
  - ordered, graceful deployment and scaling
  - ordered, automated rolling updates

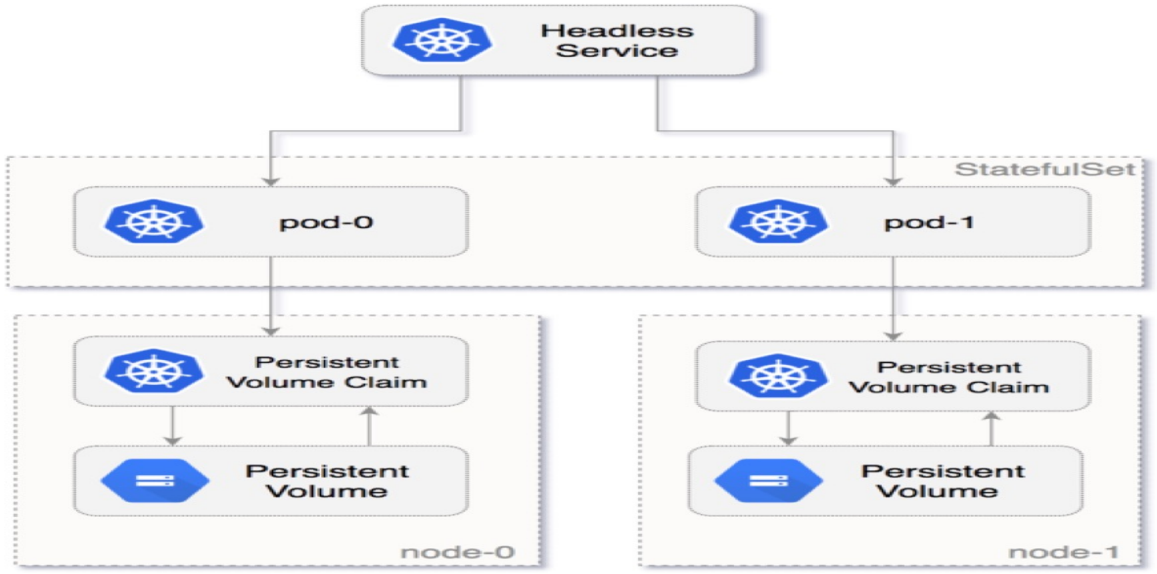
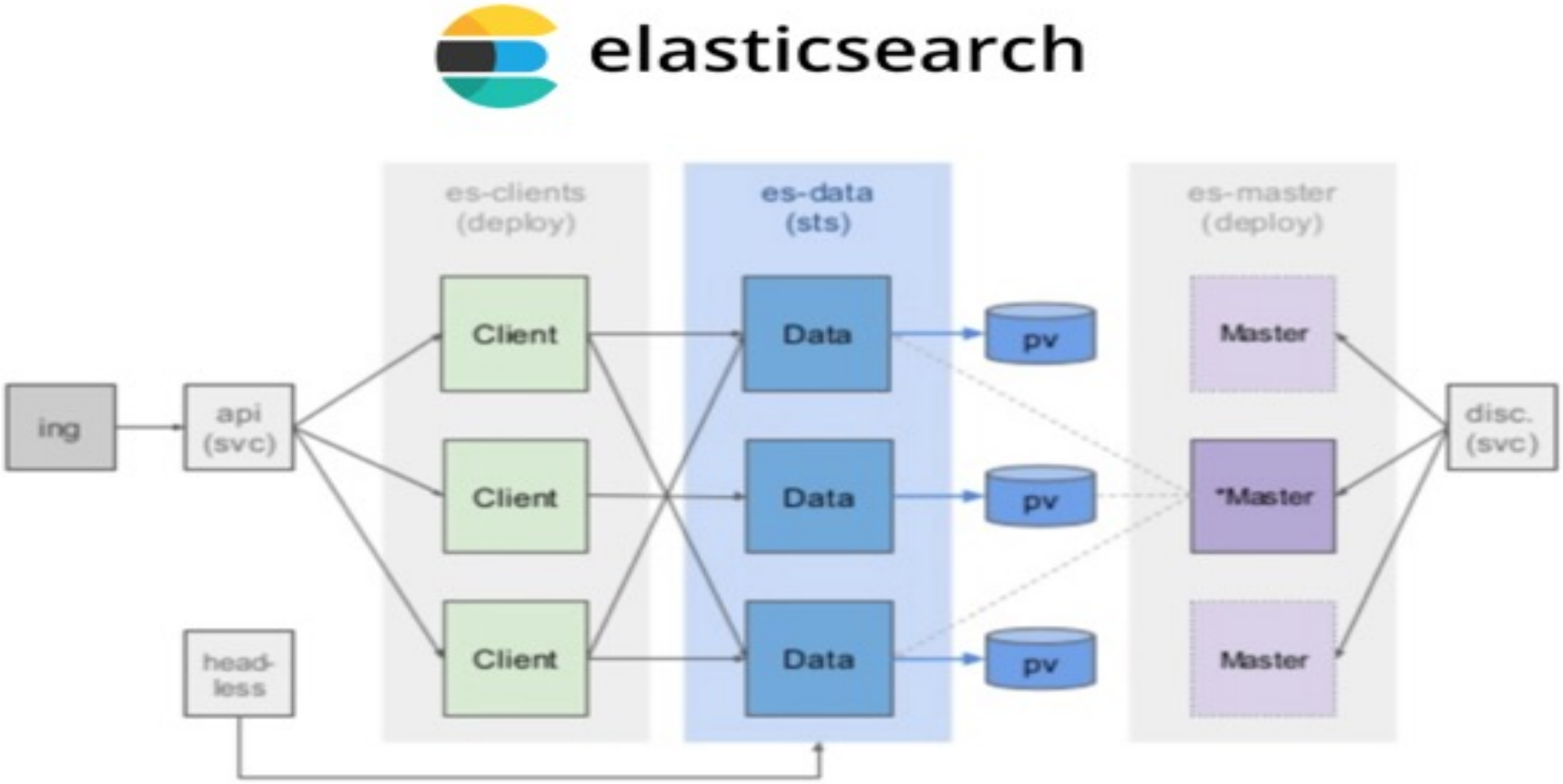


**Image source: Weaveworks Website

Our Application - ElasticSearch



Application Requirements

- Performant
- Resilient and Operable
- Cost Optimized
- Secure

Performance

Benchmark - Node Type

- Don’t just benchmark CPU and Memory you need to benchmark storage too!
- Number of data disks that can be attached to a node
- Node Throughput
- Ephemeral disks/temp storage attached to nodes
- Examples Azure Disks

Node Type and Size	Maximum Data Disks	Temp Storage SSD (GiB)	Max Uncached disk IOPS
Standard_D2ds_v4	4	75	3200
Standard_D4ds_v4	8	150	6400

Performance Sizing

- Input/output operations per second (IOPS)

- IO Request Size
  - Azure Premium SSD Disk IO Size = 256 KiB

- Throughput = IOPS X IO Size

- Example: Application Requires 10,000 IOPS with an average size of 64 KiB
  - Throughput = 10000 * 64 = 625 MBps
  - VM Example: Standard_D32ds_v4 which comes with 768MBps uncached throughput

Operating System Disk

- AKS offers 2 choices for the OS
  - Managed disk
  - Ephemeral disk (OS resides in the VM cache)
- Not all VMs support ephemeral disks
- Ephemeral disks benefits
  - Higher performance than managed disks
  - Its free of charge
  - Faster node reboot time
- As the name indicate any data that was persisted on the node will be flushed during reboots

Ephemeral vs Managed disk OS

- Ephemeral OS disk (+ ~65% better performance)

```
$ az aks nodepool add -n ephemeralos --cluster-name clusterName -g RG -c 1 -s Standard_D4ds_v4 --node-osdisk-size 100 --node-osdisk-type Ephemeral

$ fio --name=random-write --ioengine=posixaio --rw=randwrite --bs=4k --numjobs=1 --size=4g --iodepth=1 --runtime=60 --time_based --end_fsync=1

WRITE: bw=140MiB/s (147MB/s), 140MiB/s-140MiB/s (147MB/s-147MB/s), io=9251MiB (9701MB), run=66107-66107msec
```

- Managed OS Disk

```
$ az aks nodepool add -n managedos --cluster-name clusterName -g RG -c 1 -s Standard_D4ds_v4 --node-osdisk-size 100 --node-osdisk-type Managed

$ fio --name=random-write --ioengine=posixaio --rw=randwrite --bs=4k --numjobs=1 --size=4g --iodepth=1 --runtime=60 --time_based --end_fsync=1

WRITE: bw=84.8MiB/s (88.0MB/s), 84.8MiB/s-84.8MiB/s (88.0MB/s-88.0MB/s), io=8023MiB (8413MB), run=94563-94563msec
```

Zones, Pools, and Affinity

- AKS supports Node Pools and Availability Zones

```
az aks nodepool add \
--cluster-name $AKS_CLUSTER_NAME \
--name multi-zone-pool \
--resource-group $RG \
--zones 1 2 3
```

- Pools can live in N number of Zone(s)

```
az aks nodepool add \
--cluster-name $AKS_CLUSTER_NAME \
--name single-zone-pool \
--resource-group $RG \
--zones 1
```

- Pools can be scaled and auto-scaled indivually

```
az aks nodepool add \
--cluster-name $AKS_CLUSTER_NAME \
--name auto-scale-pool \
--enable-cluster-autoscaler  --min-count 1  --max-count 3
```

Best Practice – Affinity, clients next to where data is

- Clients need to be closer to the data which helps with performance and cost (cross zone charging)

backend.yaml

```
apiVersion: v1
kind: StatefulSet
…
spec:
containers:
- name: backend
…
affinity:
nodeAffinity:
requiredDuringSchedulingIgnoredDuringExecution:
nodeSelectorTerms:
- matchExpressions:
- key: failure-domain.beta.kubernetes.io/zone
operator: In
values:
- westeurope-1
- westeurope-2
```

frontend.yaml

```
apiVersion: v1
kind: Deployment
…
spec:
…
affinity:
podAffinity:
requiredDuringSchedulingIgnoredDuringExecution:
- labelSelector:
matchExpressions:
- key: app
operator: In
values:
- backend
topologyKey: failure-domain.beta.kubernetes.io/zone
```

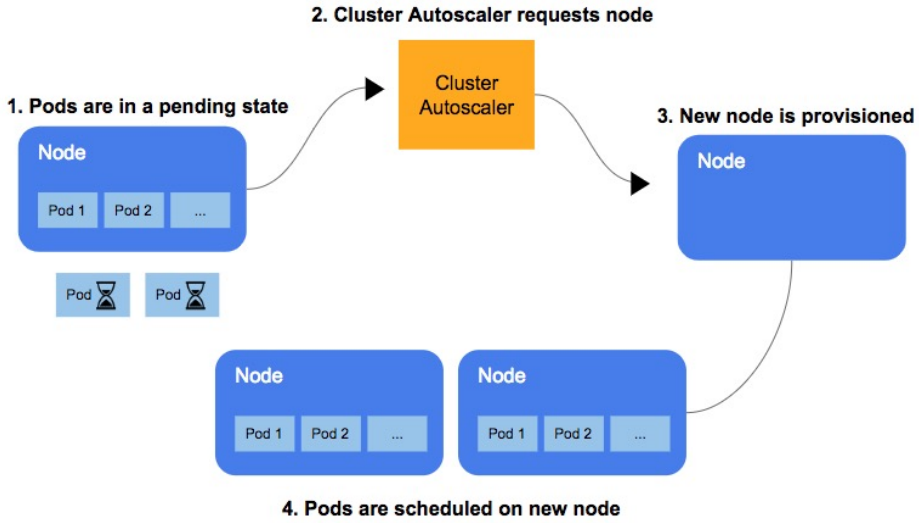
Best Practice - Multi-Zones Clusters Placement Issues

- Issue
  - By default dynamic provisioning is handled independently from pod scheduling
  - The volume can be provisioned on a different node than the Pod
  - Volumes are zonal resources – Pod can be placed in the wrong zone
- Fix
  - StorageClass should be created with volumeBindingMode: WaitForFirstConsumer
  - Azure Storage Classes now support WaitForFirstConsumer by default

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
annotations:
labels:
kubernetes.io/cluster-service: "true"
name: testtopology
kind: Managed
storageaccounttype: Standard_LRS
provisioner: kubernetes.io/azure-disk
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
```

Primer - Clustere Autoscaler

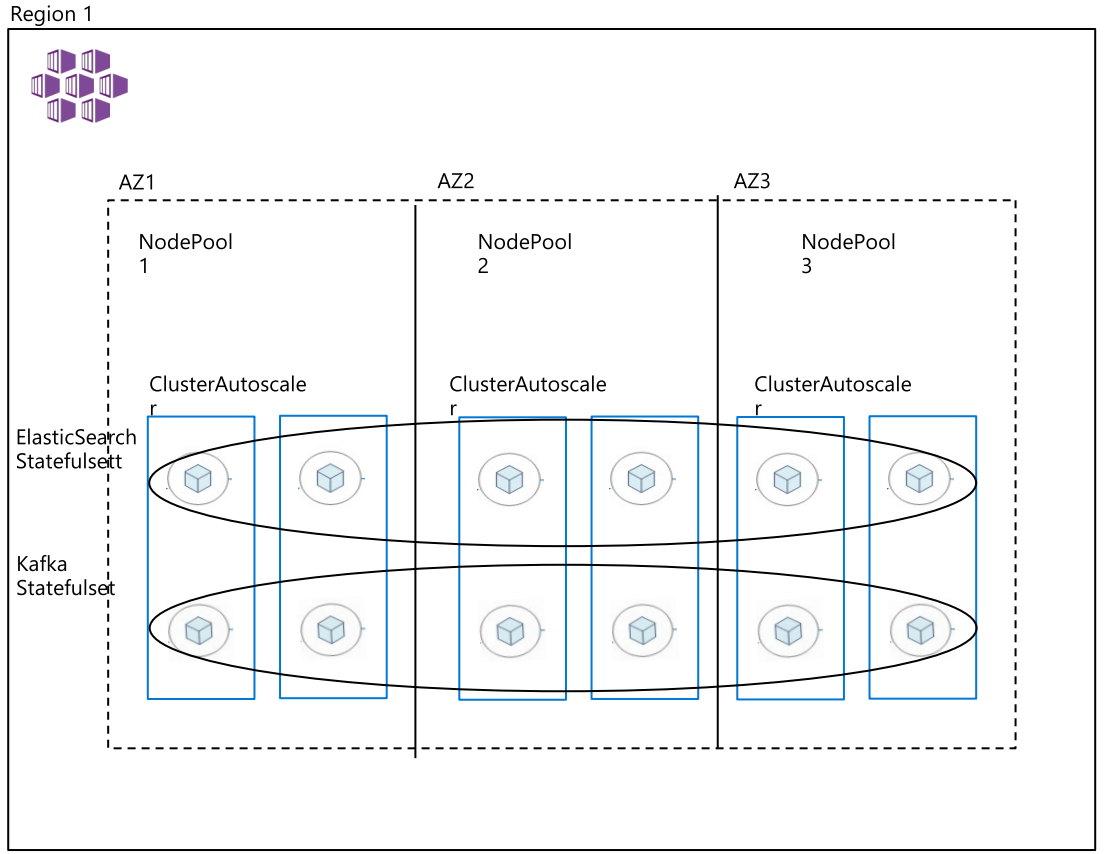
- Cluster Autoscaler is a standalone program that adjusts the size of a Kubernetes cluster to meet the current needs.
- When scale out (increase # of nodes) happen?
  - there are pods that failed to schedule on any of the current nodes due to insufficient resources.
  - adding a node similar to the nodes currently present in the cluster would help.
- When scale down (decrease # of nodes) happen?
  - when some nodes are consistently unneeded for a significant amount of time.



Credit : https://medium.com/kubecost/understanding-kubernetes-cluster-autoscaling-675099a1db92

Best Practice - Scaling with Multi-Zone Node Pools

- Issue
  - Cluster with a multi-zone nodepools
  - Pod gets scheduled on a zone that reached its upper scaling limit
  - CA relies on cloud provider to provison a node
  - Node can be placed in a different Zone
- Fix
  - Provision separate node pool per zone
  - **Note:** This issue doesn't manifest in stateless workloads, so multi-zone node pool is recommend in this case



Best Practice – use your own StorageClass(es)

- Cusomters are encouraged to create their own storage classes with the configuration that meet their workload requirements.
- Use default storage classes for guidance and learning!
- Volumes inherit their charactaristics from the storage classes at provisoning time
- Example

```yaml
allowVolumeExpansion: true
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: managed-premium-custom
parameters:
  cachingmode: ReadOnly
  kind: Managed
  storageaccounttype: Premium_LRS
  resourceGroup: storage-westeurope
  tags: costcenter=Finance   ##supported as of 1.19+
provisioner: kubernetes.io/azure-disk
reclaimPolicy: Delete
volumeBindingMode: WaitForFirstConsumer
```

Local Persistence Volume Static Provisioner For Azure

- Based on the upstream “Local Persistence Volume Static Provisioner”
  - https://github.com/kubernetes-sigs/sig-storage-local-static-provisioner
- Azure Implementation
  - https://github.com/Azure/kubernetes-volume-drivers/tree/master/local
- Allows you to use local VM disks such as local temporary disks and NVMe SSD disks
- Local temporary disks
  - SSDs attached to local hosts
  - Not available in all VM families
  - You can find it in i.e. Ddv4, Ddsv4, Edv4
- NVMe SSD
  - Available in Lsv2-series VMs, offers 8GB Memory and one 1.92TB NVMe per 8vCPU
  - Can go up to 19.2TB on 80vCPU L80s V2 VM

Example benchmark on Standard_L8s_v2

```
$kubectl get pv
NAME                CAPACITY ACCESS MODES RECLAIM POLICY  STATUS CLAIM                   STORAGECLASS AGE
local-pv-14f28886   1788Gi         RWO        Delete      Bound  default/dbench-pv-claim fast-disks   29m

$kubectl logs -f job/dbench
==================
= Dbench Summary =
==================
Random Read/Write IOPS: 145k/130k. BW: 1525MiB/s / 1269MiB/s
Average Latency (usec) Read/Write: 146.68/32.58
Sequential Read/Write: 2837MiB/s / 1326MiB/s
Mixed Random Read/Write IOPS: 103k/34.3k
```

Security

Azure Storage Service Encryption

- Azure Storage Service Encryption for data at rest (SSE) is enabled by default.
- BYOK using CMK (customer managed keys) Azure Disk Encryption
  - Supported in K8s 1.17+
  - Supports OS and Data Disks (Persistent Volumes)
  - Data Disks can be per Storage Class

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
name: hdd
provisioner: kubernetes.io/azure-disk
parameters:
skuname: Standard_LRS
kind: managed
diskEncryptionSetID:
"/subscriptions/{myAzureSubscriptionId}/resourceGroups/{myResourceGroup}/providers/Microsoft.Compute/disk
EncryptionSets/{myDiskEncryptionSetName}"
```

Host encryption

- Helps with ensuring data flows from the VM host to the storage service encrypted
- Encrypts temp disks and cache using platform managed keys
- Can be used on new clusters or new nodepools

```
$ az aks create --name myAKSCluster --resource-group myResourceGroup -s Standard_DS2_v2 -l westeurope --enable-encryption-at-host

Or

$ az aks nodepool add --name hostencrypt --cluster-name myAKSCluster --resource-group myResourceGroup -s Standard_DS2_v2 -l westeurope --enable-encryption-at-host
```

Securing Strorage Accounts

- You can decide who can work with volumes

```
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
name: storage-admin
namespace: production
rules:
…
resources: ["persistentvolumes"]
verbs: ["get", "list", "watch", "update"]
- apiGroups: [""]
resources: ["nodes"]
verbs: ["get", "list", "watch"]
- apiGroups: ["storage.k8s.io"]
resources: ["volumeattachments"]
verbs: ["get", "list", "watch", "update"]
```

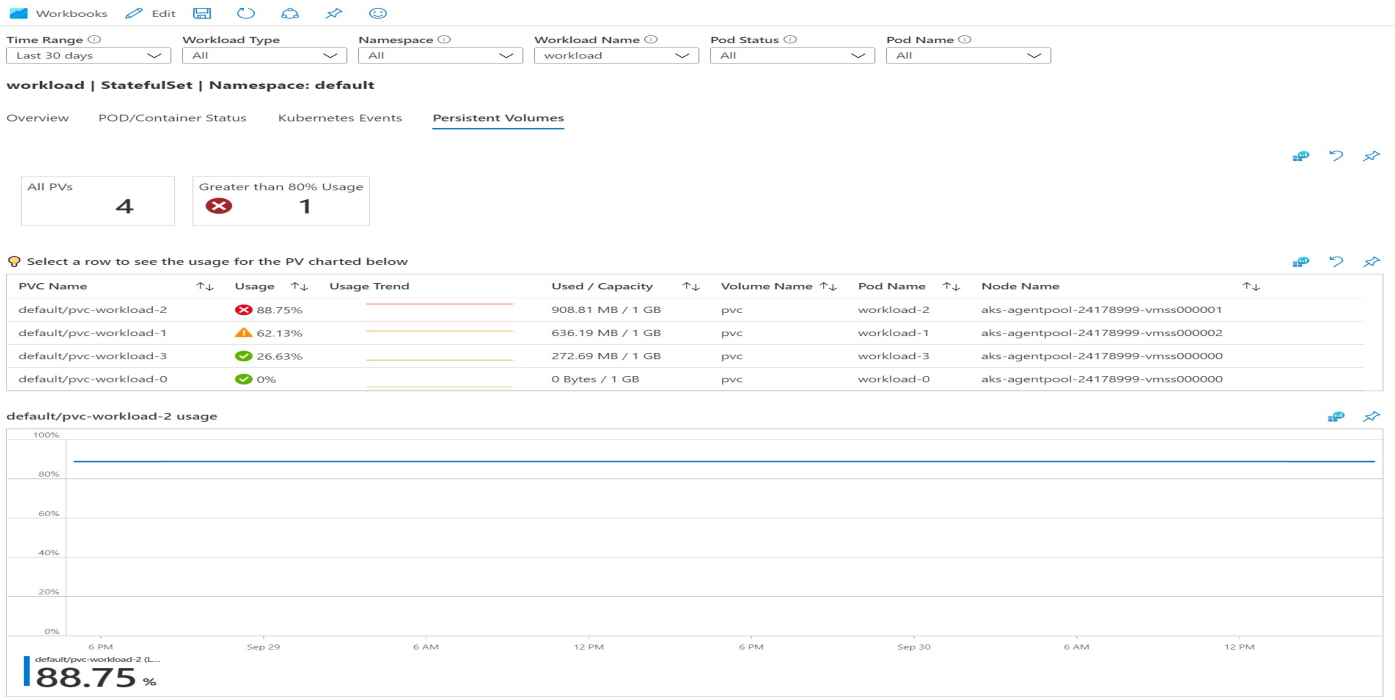
Monitoring

Node Resource Utilization

- All node resources should be monitored to ensure no bottlenecks
  - CPU Utilization
    - Container Insights: cpuUsagePercentage
    - Prometheus: node_cpu_utilisation
  - Memory Utilization
    - Container Insights: memoryRssPercentage
    - Prometheus: node_memory_MemAvailable_bytes
  - Bandwidth in/out
    - Container Insights: (bytes received/sent per second and errors per second)
    - Prometheus:
  - OS disk utilization and IO metrics
    - Container Insights: Node Disk IO metrics
    - Prometheus: note_network_receive/transmit_bytes/drop

Monitor Persistent Volumes

- Container Insights
  - pvUsedBytes
  - Make use of the workbooks



- Prometheus
  - kube_persistentvolume_*
  - Grafana has nice out of the box templates for monitoring persistent volumes

Monitor Statefulsets

- Container Insights
  - Controllers
- Prometheus
  - kube_statefulset_*
  - Grafana has out of the box dashboards which can be used

Other important tools

- Make use of the “Diagnose and solve problems” tab in the AKS portal
- Keep an eye on the advisor recommendations
- Enable Diagnostic settings (very verbose, exercise caution)

Resliency

Recommendations

- Check the session delivered during ignite for an extensive view on resiliency
  aka.ms/aks-best-practices-ignite
  github.com/mohmdnofal/aks-best-practices
- Consider Avaialbility Zones if exist in your region
- Implement a back/restore policy, most popular K8s OSS solution is Vmware velero
- Multi-region setup for a stateful workload is hard unless you’re using a shared filesystem or an overlay storage system

How to upgrade clusters with stateful workloads

- In place upgrade carry a risk, so better explore other options
- Example: New Node Pools
  - Ensure you have backup for your cluster
  - Ensure you have node pool per availability zone to ensure smooth operations
  - Add a new node pool with the new version
  - Modify your affinity rules if needed to accommodate for the new pool
  - Cordon and drain the nodes in the first pool, nodes should move the new pool
  - Repeat for the rest of the node pools

Cost Optimization

Recommendations

- Benchmark the required node type(s) and disk type(s) for optimal cost/performance balancer
- Use Cluster Autoscaler (the biggest chunk of savings is here)
- Use Node Pools for special workloads
- Use Affinity to reduce cross zone charges for multi-zone clusters/pools
- Use Spot node pools for transit workloads
- Use Ephemeral OS disks
- Make use of the Azure Advisor recommendations

Bonus Slide: should I use burstable VMs?

- Yes, for Test and Dev workloads
- No, for production clusters
  - Its difficult to set the correct resource quotas and limits, is it on the baseline performance (defeats the purpose) or on the burst limit (need dynamic quota and limits)
  - The other option is to not use resource quotas and limits, but this means no scaling

Management

Use K8s Operators When Exist

- Operators are class of k8s controllers
- Implement and manage custom resources
- Its packages applications for Kubernetes i.e. CouchDb, kafka, etc..
- https://operatorhub.io

3rd party storage providers

3rd Party Storage Solutions on Azure

Some In-Kubernetes Storage Solutions	    + HEKETI 
Some 3rd Party Azure Market Place NFS Solutions	 
Build your own	 

Azure vs. 3rd Party Storage Solutions

	Azure Storage	3rd Party
Operations	Low	Medium to High
Cost	$	$$$
Security	Inherit Benefits (shared responsibility)	Your Responsibility
Portability	Azure and Azure Stack	Any Data Center
Support	Azure Support	3rd party or in-house

Resources

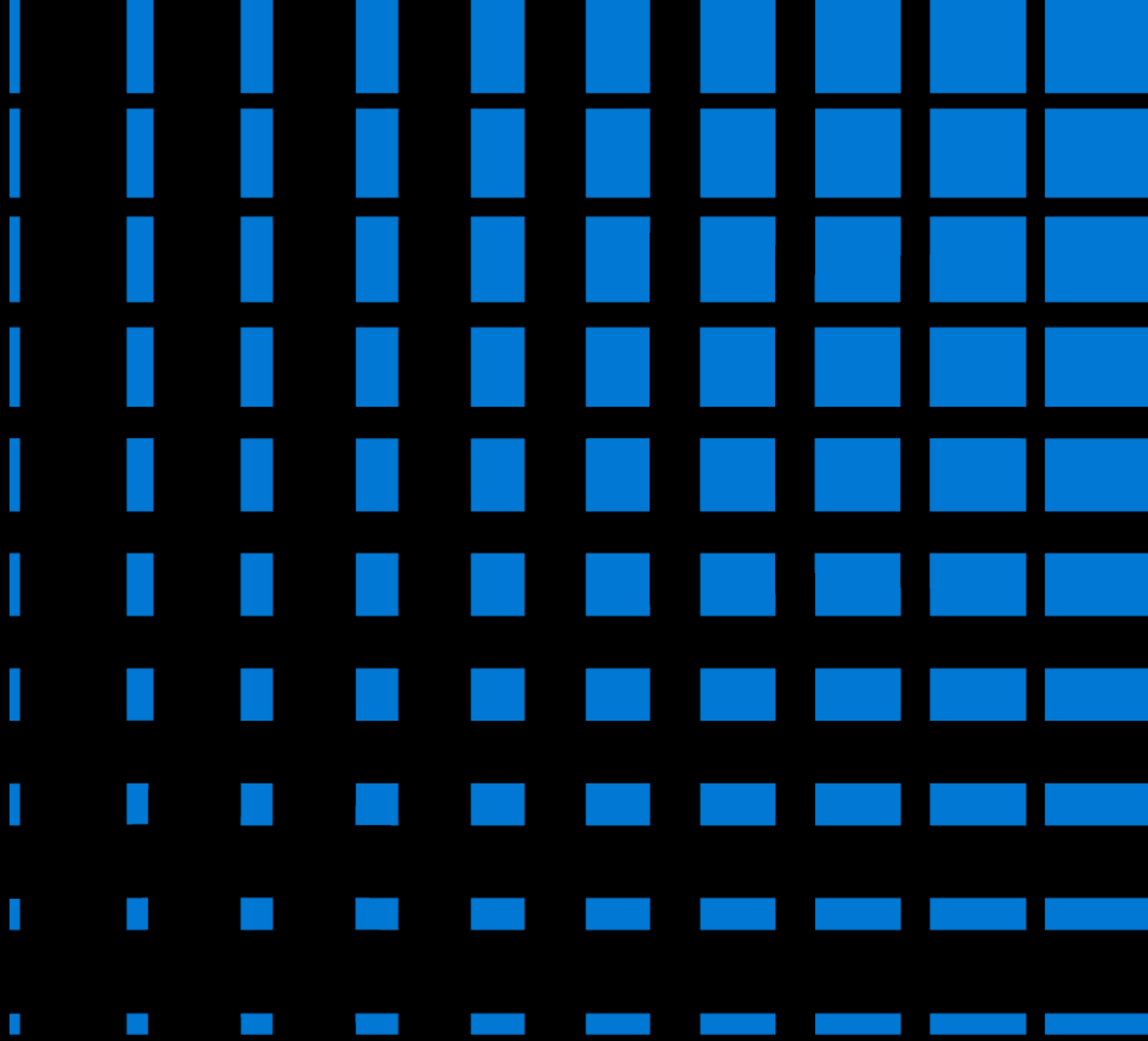
Resources

- AKS and best practices docs
  https://docs.microsoft.com/en-us/azure/aks
  https://docs.microsoft.com/en-us/azure/aks/best-practices

- Ignite talk on resiliency, slides, and demos
  https://aka.ms/aks-operational-best-practices
  https://github.com/mohmdnofal/aks-best-practices/tree/master/ha_cm

- This talk slides and demos
  https://github.com/mohmdnofal/aks-best-practices

Thank you!



Microsoft Azure