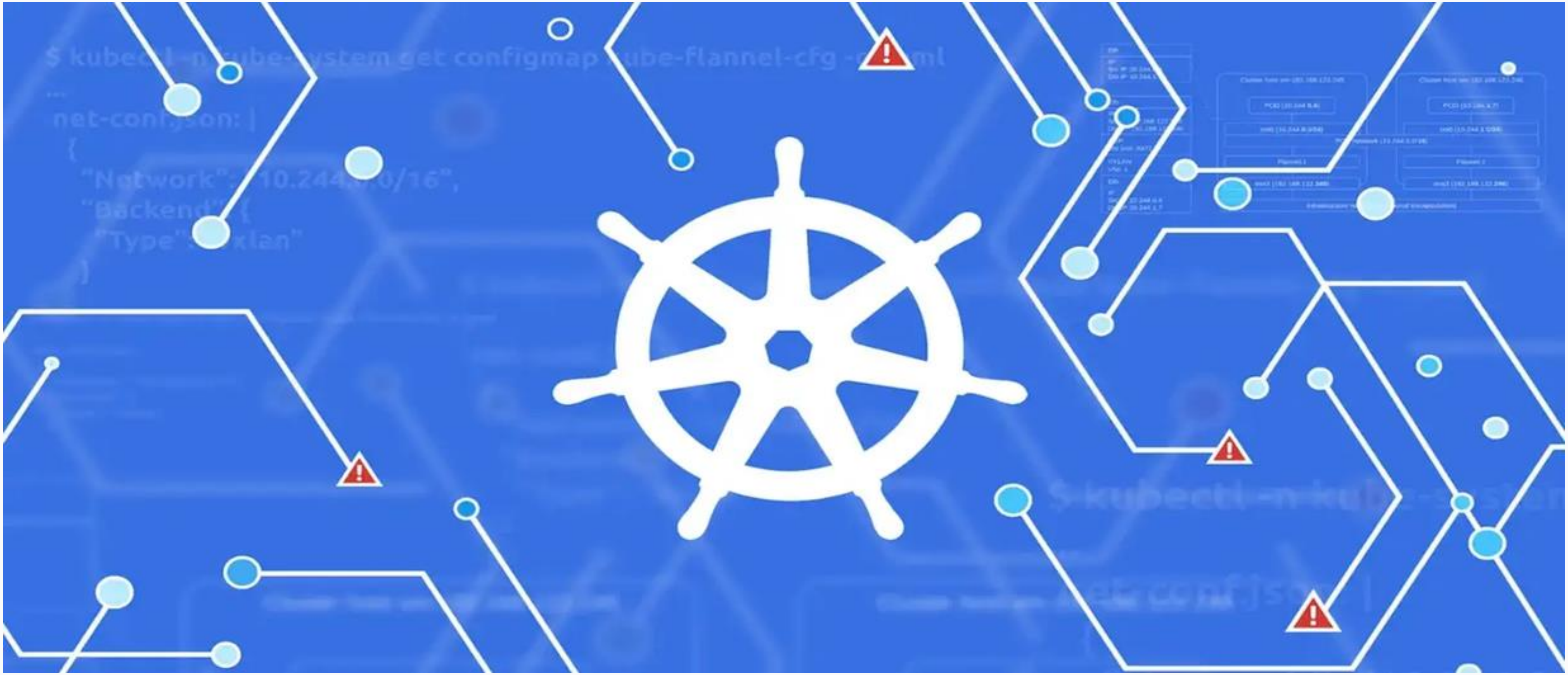
Building Resilient Stateful Apps in Kubernetes in the cloud


$ kubectl -n kube-system get configmap kube-flannel-cfg -o yaml
...
net-conf.json: |
{
"Network": "10.244.0.0/16",
"Backend": {
"Type": "vxlan"
}
$ kubectl -n kube-system
net-conf.json: |

slido

Join at slido.com #3033398

ⓘ Start presenting to display the joining instructions on this slide.

slido

Welcome! Where do you come from?

ⓘ Start presenting to display the poll results on this slide.

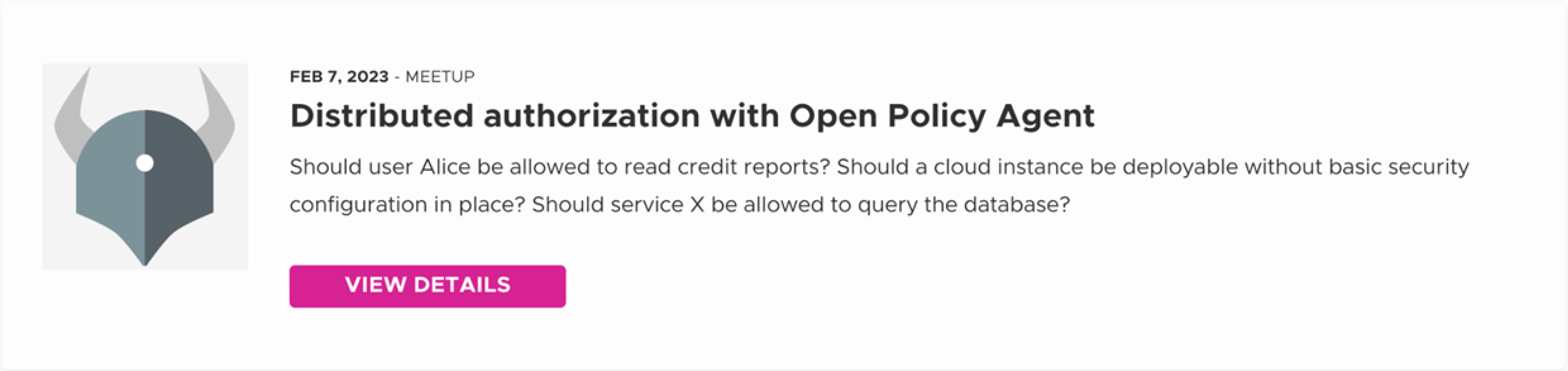
slido

Audience Q&A Session

ⓘ Start presenting to display the audience questions on this slide.

Next Events

February 7th at 17:30



FEB 7, 2023 - MEETUP

Distributed authorization with Open Policy Agent

Should user Alice be allowed to read credit reports? Should a cloud instance be deployable without basic security configuration in place? Should service X be allowed to query the database?

Mohammad Nofal

Application Modernization Architect – Global Black Belt Team – EMEA

@mohmd_nofal

Agenda

- The basics
- Performance
- Security
- Monitoring
- Resiliency

The Basics

What is a stateful application?

Stateful applications

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

Examples

MySQL, MariaDB, Postgres, SQL Server, Percona, MongoDB

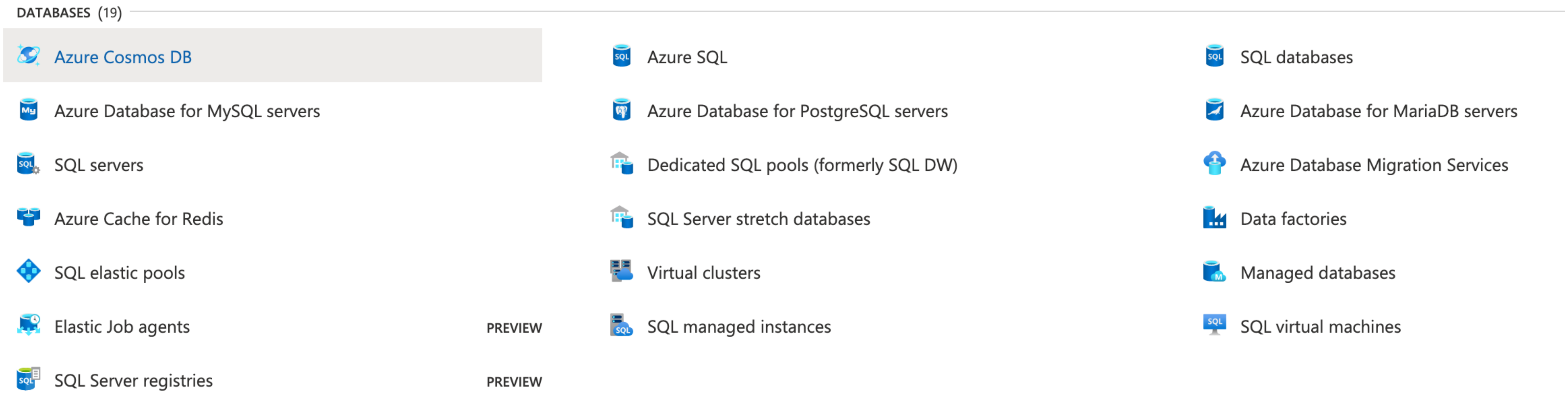
Cassandra, Kafka, Elastic Search + Kibana, Spark, Solr, Kubeflow

Wordpress, Redis, Jenkins

Rule of thumb

- 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



DATABASES (19)

- Azure Cosmos DB
- Azure Database for MySQL servers
- SQL servers
- Azure Cache for Redis
- SQL elastic pools
- Elastic Job agents PREVIEW
- SQL Server registries PREVIEW
- Azure SQL
- Azure Database for PostgreSQL servers
- Dedicated SQL pools (formerly SQL DW)
- SQL Server stretch databases
- Virtual clusters
- SQL managed instances
- SQL databases
- Azure Database for MariaDB servers
- Azure Database Migration Services
- Data factories
- Managed databases
- SQL virtual machines

When do customers build their own stateful data services?

- There is no managed/PaaS service offered
- Lift and shift of existing workload running on-premises or in other clouds, or hybrid deployments
- They require more control on their service or performance characteristics which the PaaS service doesn’t offer
- More control always comes with an operational cost, always weigh the cost vs benefits

Should I build stateful data services inside K8s or VMs?

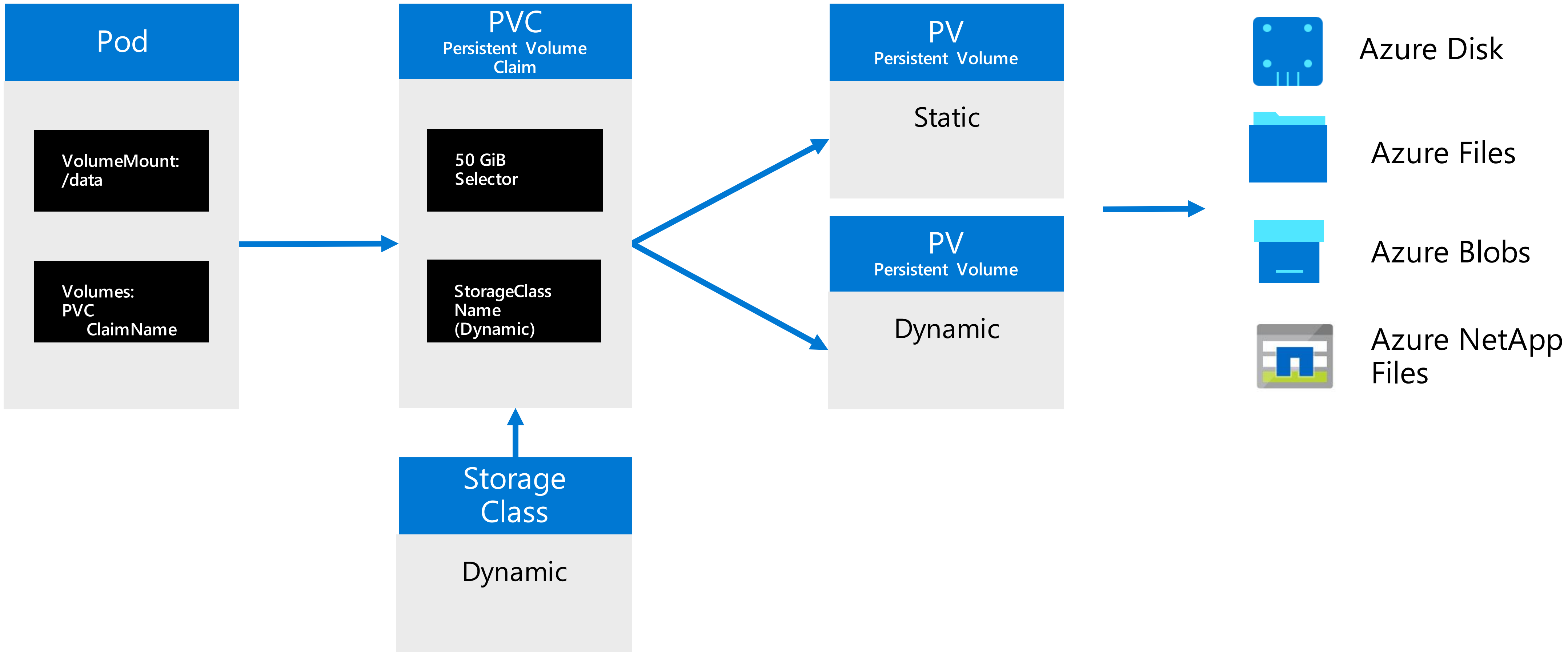
- Kubernetes and Kubernetes Statefulsets offer many features that make it easier to run stateful applications compared to just using VMs
- Many organizations are standardizing their operations on top of Kubernetes, as such building on the existing skill-set
- Kubernetes Operators are becoming more mature
- **Exception**: K8s has still some limitations when building service that should span clusters or regions

Kubernetes and Azure Storage Basics

Azure Storage options for stateful container workloads

	Azure Disk Storage	Azure File Storage	Azure Blob and Data Lake Storage	Azure NetApp Files via Trident
Workloads	Databases, bigdata, cache, CI/CD	Shared/user workspace, CMS, databases, AI/ML	Analytics on data lake, HPC	Analytics, HPC, Custom apps currently using NetApp
Access protocol	SCSI	SMB, NFS v4.1	Blobfuse, NFS v3.0	NFS (v3.0,v4.1), SMB (v2.1, v3.1)
Workloads	Databases, bigdata, cache, CI/CD	Shared/user workspace, CMS, databases, AI/ML	Analytics on data lake, long term retention, HPC	Analytics, HPC, Custom apps currently using NetApp
SKUs	Standard HDD, Standard SSD, Premium SSD, Premium SSDv2, Ultra	Standard HDD, Premium SSD	Standard HDD, Premium SSD	Standard, Premium, Ultra
Access modes	RWO, RWX (v1.21 in ZRS only)	RWO, RWX	RWO, RWX	RWO, RWX
Container type	Linux, Windows	Linux, Windows, ACI	Linux	Linux
Availability	LRS, ZRS	LRS, ZRS, GRS, RAGRS	LRS, ZRS, GRS, RAGRS	Single-zone

How to request peristent storage in Kubernetes?



Pod

VolumeMount: /data

Volumes: PVC ClaimName

PVC

Persistent Volume Claim

50 GiB Selector

StorageClass Name (Dynamic)

Storage Class

Dynamic

PV

Persistent Volume

Static

PV

Persistent Volume

Dynamic

Azure Disk

Azure Files

Azure Blobs

Azure NetApp Files

AKS Storage Classes today

5 storage classes for CSI volumes are offered by default >1.21

```
$ kubectl get storageclasses.storage.k8s.io
NAME                  PROVISIONER        RECLAIMPOLICY   VOLUMEBINDINGMODE     ALLOWVOLUMEEXPANSION   AGE
azurefile-csi-premium    file.csi.azure.com   Delete        Immediate            true               13d
azurefile-csi          file.csi.azure.com   Delete        Immediate            true               13d
default (default)       disk.csi.azure.com   Delete        WaitForFirstConsumer   true               13d
managed-csi-premium     disk.csi.azure.com   Delete        WaitForFirstConsumer   true               13d
managed-premium        disk.csi.azure.com   Delete        WaitForFirstConsumer   true               13d
```

CSI Drivers GA in AKS

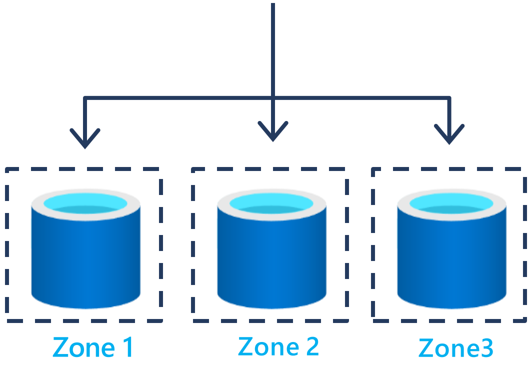
- CSI Drivers GA'ed in AKS in k8s v1.21 and is default now
- CSI is out of tree, i.e. Azure can apply changes to the drivers without touching the k8s core code base or waiting for k8s release cycles
- Offers new features in CSI
  - Volume Snapshots
  - Volume Clones
  - Resize Volumes (offline and online/preview)
  - Shared Volumes
  - Ephemeral Local Volumes

AKS CSI Drivers also bring additional improvements

- GA of Ultra Disk in AKS
- ZRS disks support for multi-zone clusters
- Larger file shares for Azure Files
- SMB 3.1.1 defaulted for Files in AKS
- NFS options for Blob (GA) and Files
- Private Endpoint support for Azure Files and Blobs
- ISV support for data protection on CSI volumes for Disk and Files

Zonal Redundant Disks (ZRS)

- ZRS means we copy the data “synchronously” to 3 zones
- Can be used as shared disk (more later)
- Available with CSI driver only
- **When to use**
  - **If you don’t have application level synchronous/asynchronous writes replication** (Elastic Search, MySQL, etc…)
- What to expect
  - 3 nodes in 3 AZs, pod with ZRS disk in zone [1] experiences a node failure or node is cordoned
  - Pod will be rescheduled in zone [2 or 3] and the disk will be attached with data intact

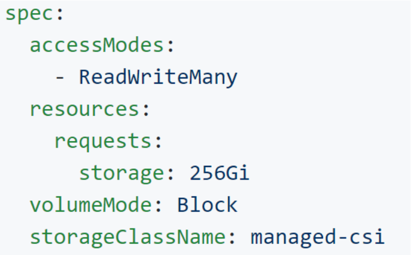


Zone redundant storage

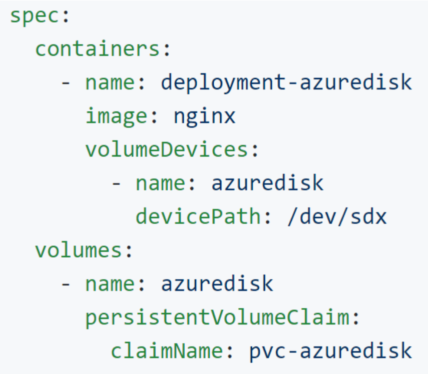
- **Synchronous writes** to 3 zones
- Protect against disk, node, rack and **zone** failures

Shared Disks

- Shared disk enables disk to be mounted to more than one node simultaneously
- Use ReadWriteMany (RWX) accessMode and Raw Block Device with Block volumeMode
- Specify devicePaths instead of mountPaths. Container will see a device instead of a mounted file system



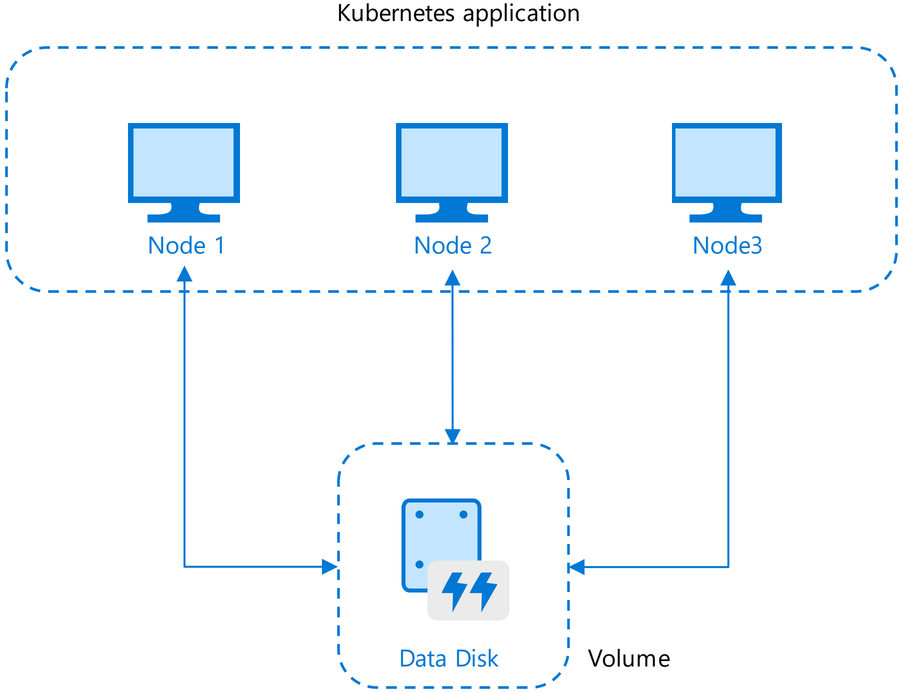
```
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 256Gi
  volumeMode: Block
  storageClassName: managed-csi
```



```
spec:
  containers:
    - name: deployment-azuredisk
      image: nginx
      volumeDevices:
        - name: azuredisk
          devicePath: /dev/sdx
  volumes:
    - name: azuredisk
      persistentVolumeClaim:
        claimName: pvc-azuredisk
```

- Can be used with applications that can manage writes, reads, locks, caches, fencing on raw block volumes (Pacemaker, corosync,etc..)
- Can be used with PPGs for lowest latency
- Supports ZRS disks

https://aka.ms/k8sRWXonShared



✓ Mount disk on multiple nodes

✓ Use as RWX raw block volumes

✓ Supported on Ultra Disks, Premium SSD and Standard SSD

Kubernetes StatefulSets

- provide guarantees on the ordering and uniqueness of deployed pods
- maintain 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

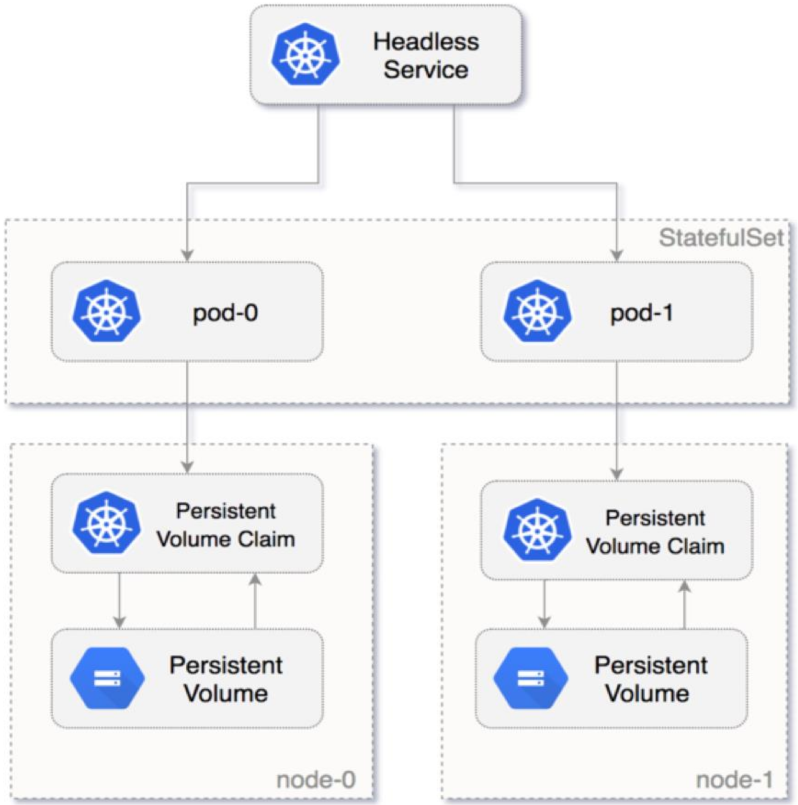
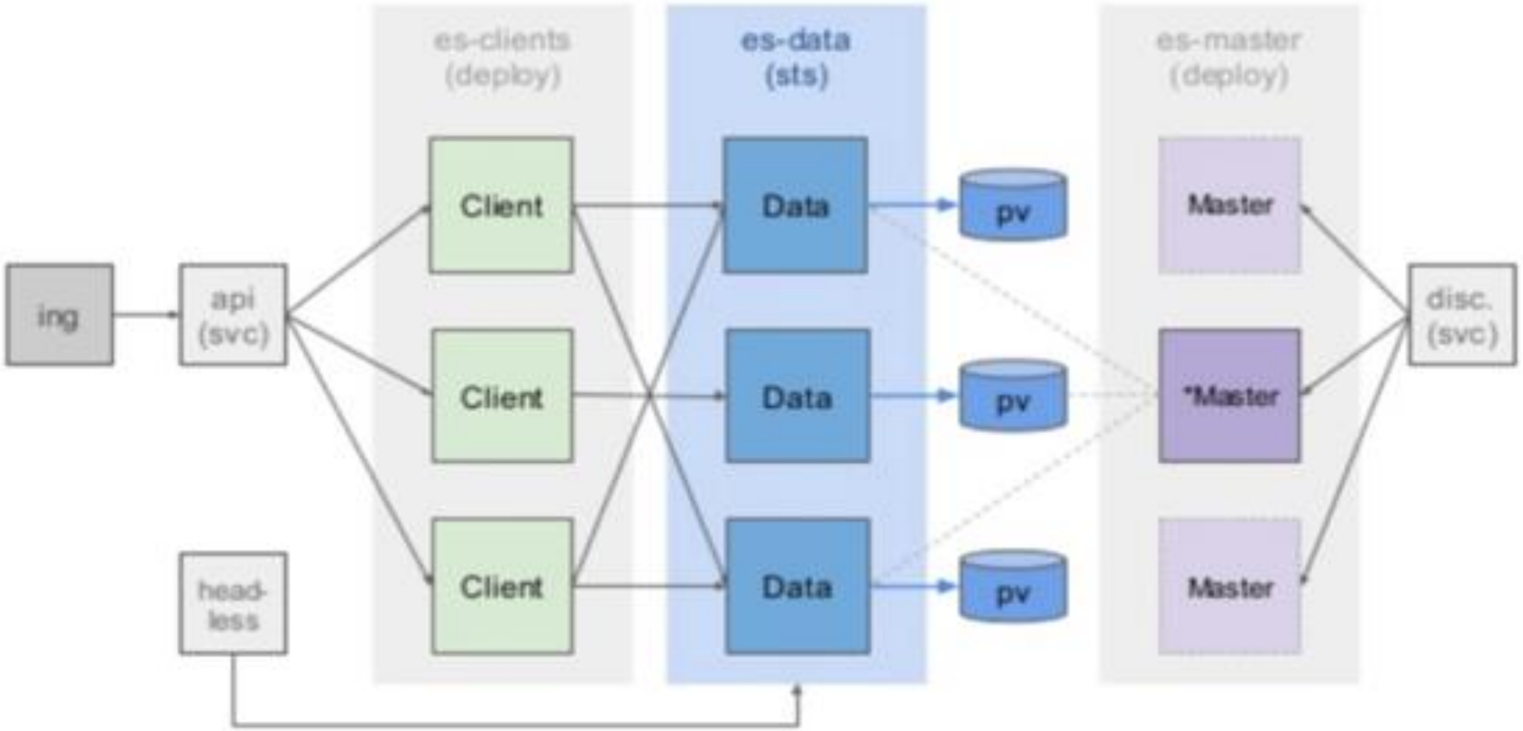


**Image source: Weaveworks Website

Our Application - ElasticSearch

elasticsearch

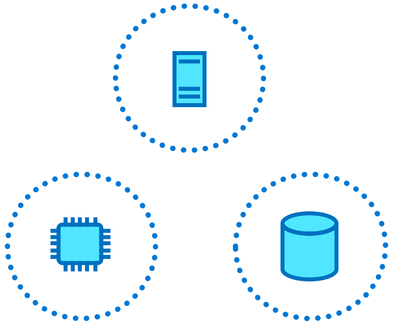


Application Requirements

- Performant
- Resilient and Operable
- Secure
- Cost Optimized

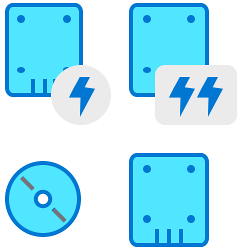
Performance

To optimize for workload performance with Azure Disks



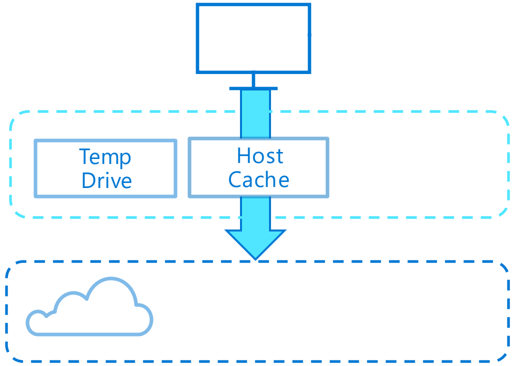
Choose the right VM

to optimize the storage



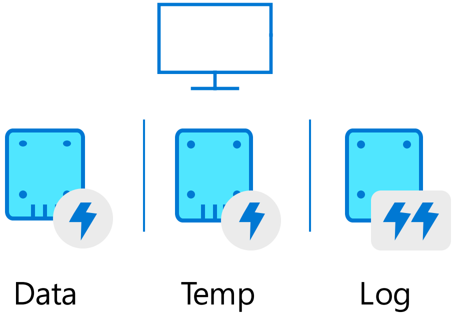
Choose the right disk

mapped to IOPS, BW and latency



Enable host cache

for improved IOPS and latency



Isolate files

to optimize read vs write traffic

Performance
Sizing

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

https://docs.microsoft.com/en-us/azure/virtual-machines/disks-benchmarks

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

https://docs.microsoft.com/en-us/azure/virtual-machines/disks-performance

Azure data Disk performance scaling options

Which performance solution is right for you?

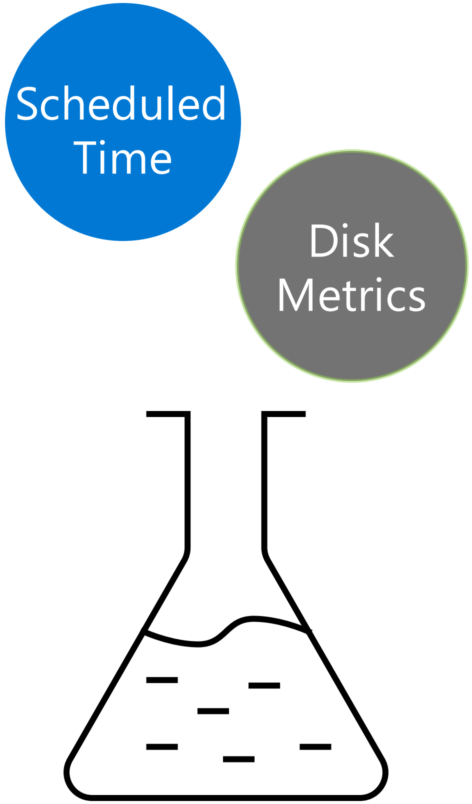
	Credit-based bursting	On-demand bursting	Performance tiers	Ultra Disk
Performance scaling	Recommended for unplanned events	Recommended for unplanned events	Recommended for planned events	Recommended for planned events
Duration of higher performance	Short-term	Short-term	Longer duration - sustained higher performance	Longer duration – sustained higher performance
Cost	Free, based on credit system	Enablement fee and cost per transaction	Fixed cost, you pay for the current performance tier	Fixed cost, you pay for the performance provisioned
Disk type	Premium SSD & Standard SSD on sizes less than and equal to 512GB	Premium SSD on sizes bigger than 512GB	All Premium SSDs	Ultra Disk
Latency	Low single digit ms	Low single digit ms	Low single digit ms	Sub millisecond
Enablement	Enabled by default	Manual enablement required	Manual enablement required	N/A; this is a standalone product

Ultra Disks

Dynamically configure Ultra Disks to meet your price-performance needs

Choose characteristics that meet your price-performance needs

Scale to meet business needs while reducing costs



Trigger Azure Disk update

```
"az disk update --set diskIopsReadWrite=10000
                --set diskMbpaReadWrite=156"
```

Seamless Scaling

In effect within 5 mins

Performance
Cluster OS Disk

Cluster Operating System Disk

- AKS offers 2 choices for the OS
  - Azure Disk
  - Ephemeral disk (OS resides in the VM cache)
- Ephemeral disks pros
  - Local disk with higher performance than Azure Disk
  - Its free of charge
  - Faster node reboot time
- Ephemeral disks cons
  - Not all VMs support ephemeral disks
  - Any data that was persisted on the node will be lost during reboots or moves between hosts

Ephemeral vs Azure 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
```

- Azure Disk (OS)

```
$ 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
```

Performance

Zones/Affinity

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

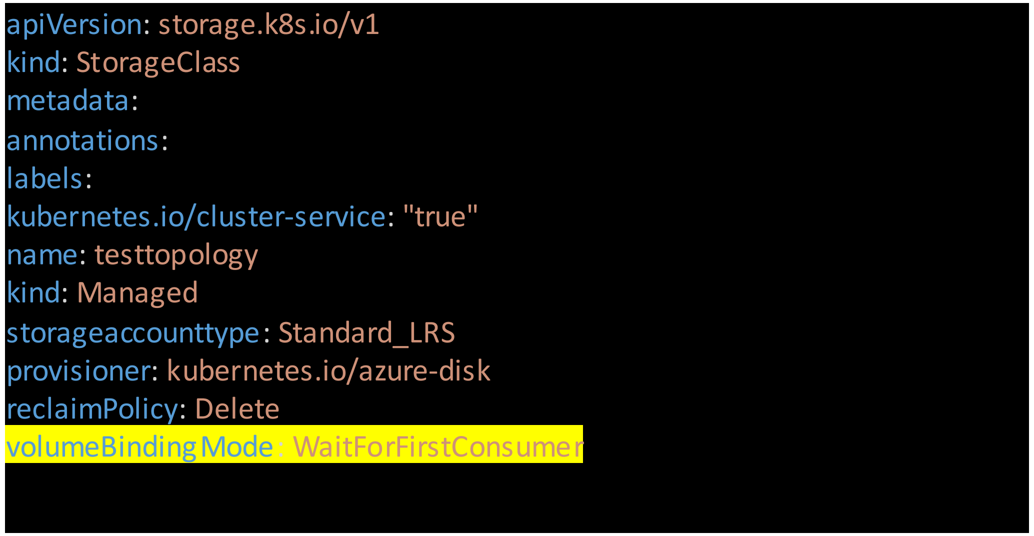
```
apiVersion: v1
kind: StatefulSet
…
spec:
containers:
- name: backend
…
affinity:
nodeAffinity:
requiredDuringSchedulingIgnoredDuringExecution:
nodeSelectorTerms:
- matchExpressions:
- key: topology.kubernetes.io/zone
operator: In
values:
- westeurope-1
- westeurope-2
```

frontend.yaml

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

Best Practice - Multi-Zones Pools Placement Issues

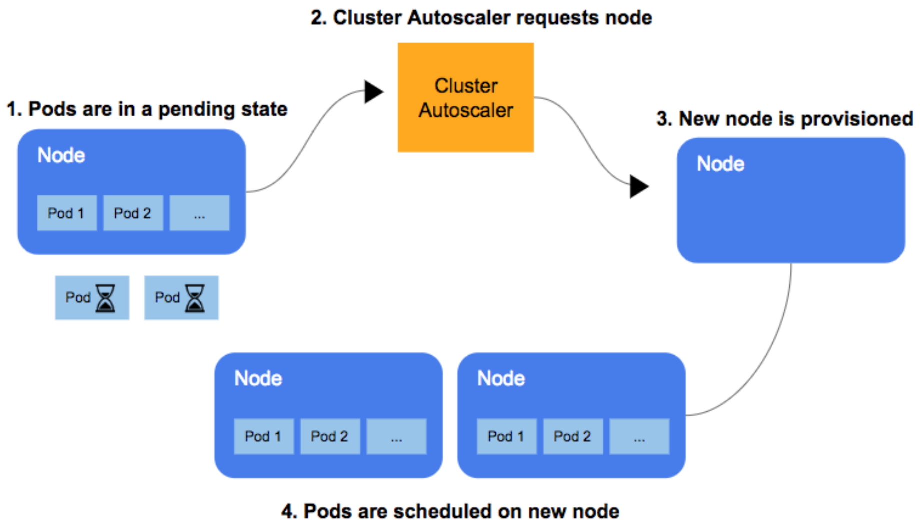
- Issue
  - By default dynamic disk provisioning is handled independently from pod scheduling
  - The disk/volume can be provisioned on a different node than the Pod
  - Disks/Volumes are zonal resources – Pod can be placed in the wrong zone
- Fix
  - StorageClass should be created with volumeBindingMode: WaitForFirstConsumer
  - Azure Storage Classes now support WaitForFirstConsumer by default
  - PVC will be unbound until the pod is created
- Note:
  - This problem doesn't manifest in regional volumes like Azure Files, or ZRS



```
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 – Cluster Autoscaler

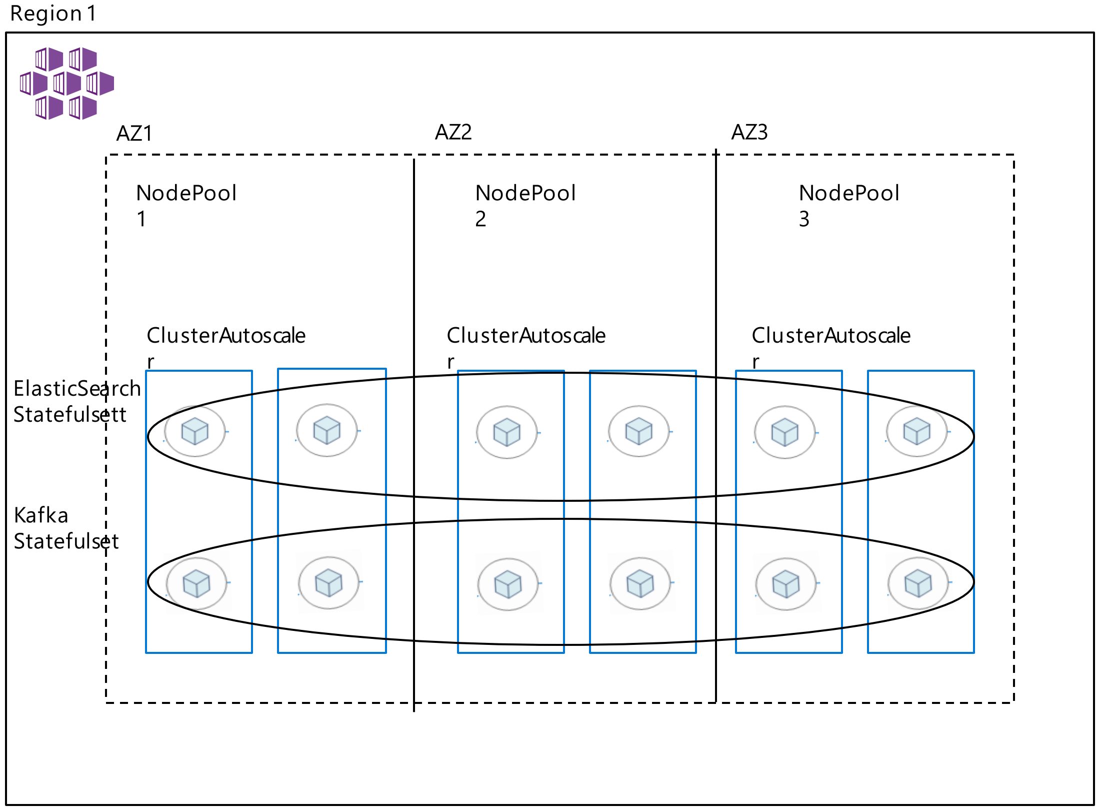
- Cluster Autoscaler is a standalone program that adjusts the size of a Kubernetes cluster to meet the current needs.
- When scale out (increase # of nodes) happen?
  - there are pods that failed to schedule on any of the current nodes due to insufficient resources.
  - adding a node similar to the nodes 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.


2. Cluster Autoscaler requests node
Cluster Autoscaler
1. Pods are in a pending state
Node
Pod 1
Pod 2
...
Pod
Pod
3. New node is provisioned
Node
Node
Pod 1
Pod 2
...
Node
Pod 1
Pod 2
...
4. Pods are scheduled on new node

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 provision 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 or regional volumes (azure files, ZRS), however, it remains a best practice to use node pool per zone when autoscaling is required.



Performance
Additional options

Best Practice – use your own StorageClass(es)

- Customers 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

```
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

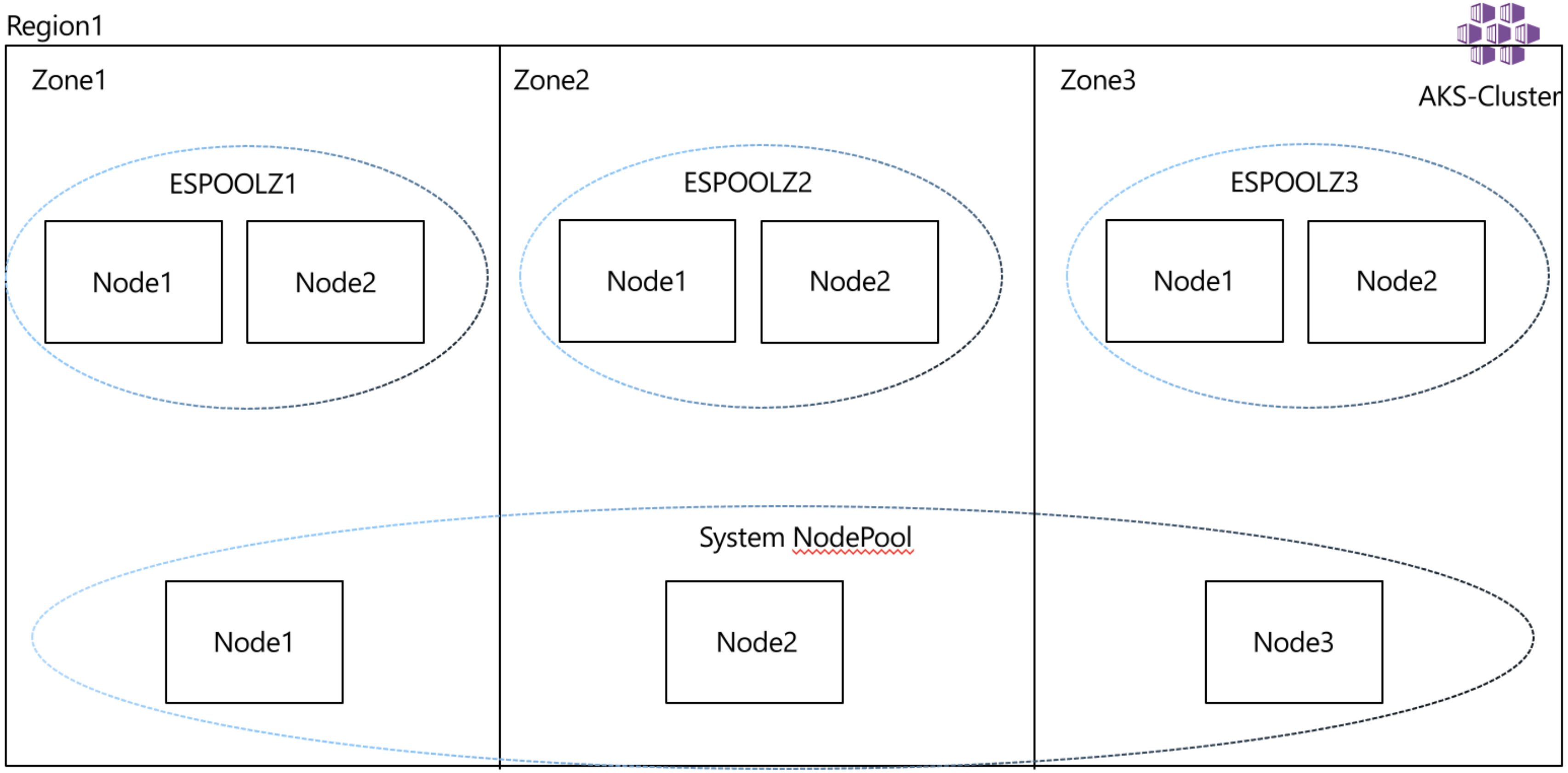
- Based on the upstream “Local Persistence Volume Static Provisioner”
  - https://github.com/kubernetes-sigs/sig-storage-local-static-provisioner
- Azure Implementation
  - https://github.com/Azure/kubernetes-volume-drivers/tree/master/local
- Allows you to use local disks on the VM such as local temporary disks and NVMe disks
- Great for distributed workloads like Cassandra, MongoDB, Elastic, etc that are distributed in nature and have high availability built into them
- 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 disks
  - 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
```

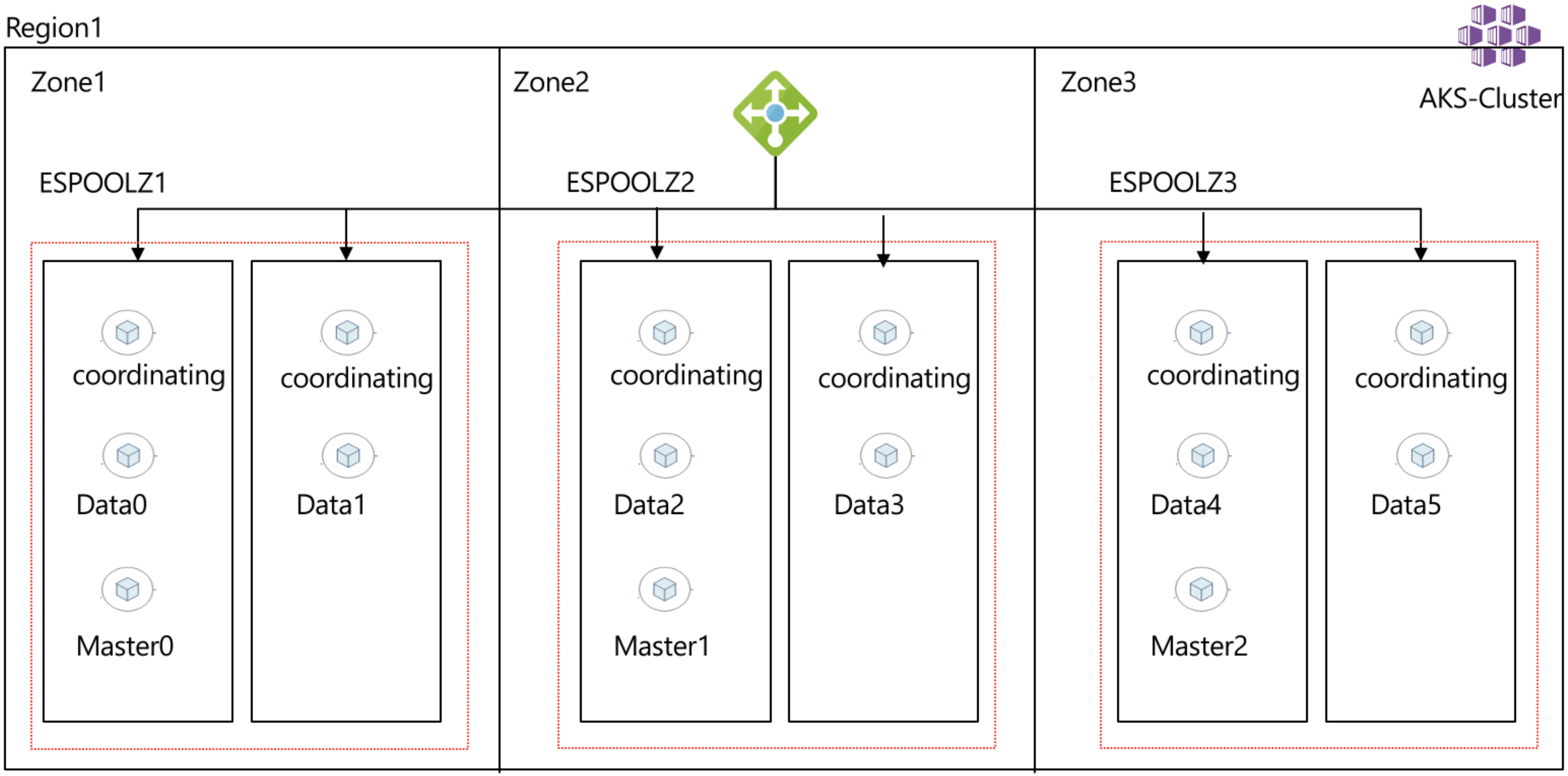
Our Cluster


Region1
AKS-Cluster
Zone1
Zone2
Zone3
ESPOOLZ1
Node1
Node2
ESPOOLZ2
Node1
Node2
ESPOOLZ3
Node1
Node2
System NodePool
Node1
Node2
Node3

Our Affinity Rules

```
affinity:
nodeAffinity:
requiredDuringSchedulingIgnoredDuringExecution:
nodeSelectorTerms:
- matchExpressions:
- key: topology.kubernetes.io/zone
operator: In
values:
- eastus-1
- eastus-2
- eastus-3
requiredDuringSchedulingIgnoredDuringExecution:
nodeSelectorTerms:
- matchExpressions:
- key: agentpool
operator: In
values:
- espoolz1
- espoolz2
- espoolz3
```

Our Elastic Search Cluster



Resiliency

Resiliency concepts

Robustness

- Build mechanisms into our software and process to accommodate expected problems
  - Discussed in depth in the previous section

Rebound

- The ability to recover after a traumatic event
- How well we recover/rebound from disruption?

Graceful extensibility [People Focused]

- How well we deal with a situation that is unexpected

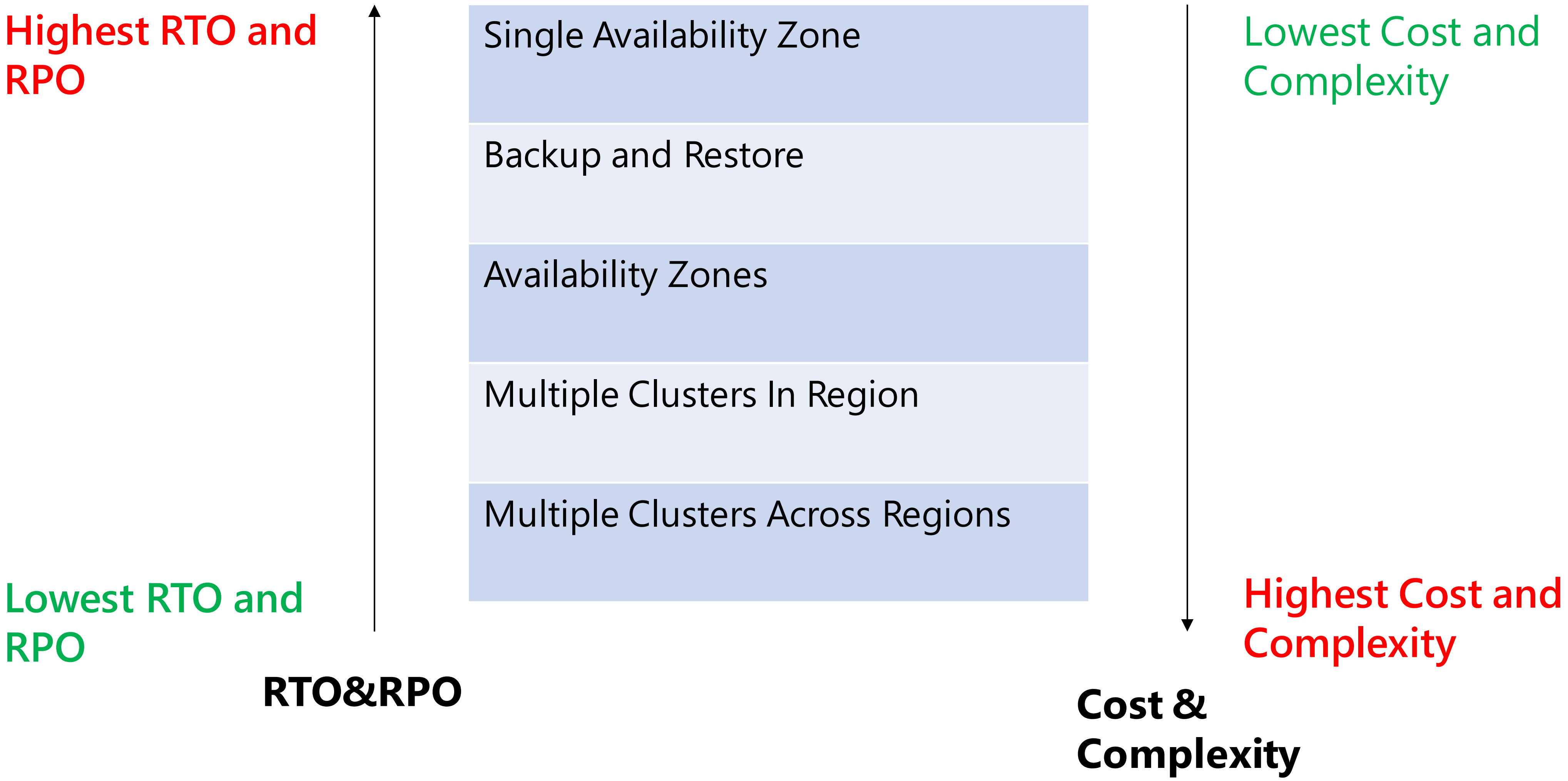
Sustained adaptability [People Focused]

- The ability to continually adapt to changing environment, stakeholders, and demands

As defined by David Woods in " Four Concepts for Resilience and the Implications for the Future of Resilience Engineering,"

Inspired by "Building Microservices" by "Sam Newman"

Resiliency Ladder



Highest RTO and RPO

Lowest RTO and RPO

RTO&RPO

Single Availability Zone

Backup and Restore

Availability Zones

Multiple Clusters In Region

Multiple Clusters Across Regions

Lowest Cost and Complexity

Highest Cost and Complexity

Cost & Complexity

General Principles

- Understand the availability requirements for your application
  - <=99.99% A single region with availability zones will suffice
  - >99.99% Multi-Region setup should be considered
- Don’t think of availability as a single big problem, decompose into smaller workable problems/milestones

Single Region

- Availability zones are always to be considered
  - Availability zones introduce a bit of complexity as discussed before, but the benefits out-weight the complexity
- Fault Domains are to be considered in regions where AZs aren't available (Availability==99.95%)
  - topology.kubernetes.io/zone in case of unzoned nodes will take values like FaultDomain=0
- Backup/Restore policy should be implemented for all components
  - Deployment artifacts (handled in code repository)
  - Data

[Rebound] Backup Options

- Azure snapshot API
  - Not K8s native, manual process, complex to manage at scale
- Azure disk backup
  - Not K8s native, Built-in automation, https://docs.microsoft.com/azure/backup/backup-managed-disks-cli
- CSI Snapshots
  - K8s native, but complex to manage at scale
- 3rd Party OSS solutions, i.e Velero
  - K8s native and automation is easy to implement

    VELERO
- 3rd Party paid solutions i.e. Kasten
  - Full solutions, many features, COST MONEY

    kasten by Veeam portworx by Pure Storage RACKWARE Zerto COMMVAULT
- [Roadmap] Azure Native K8s backup solution
  - private preview, expected to go live in H2Y23

Azure disk Backup Limitations

- Snapshot across region isn’t supported
- Copy snapshot across region isn’t supported

- The above makes it too hard to achieve multi-regional setup with native/1st party solutions (only applicable for disks)

- Velero Restic support can backup content only, can be considered as a free alternative
- 3rd Party solutions which offers storage abstraction can help
- A native API for cross region copy is coming

Velero

- Native Kubernetes OSS backup and restore solution by Heptio, now VMware
- Uses the underlying cloud provider API, i.e. Azure Snapshot API
- Sends a tarball for k8s config to blob storage and takes snapshots for azure disks
- Can be used to restore or migrate to new clusters
- Backups can be scheduled
- Support Backup/Restore hooks i.e. FSFREEZE
- Has RESTIC support, to backup content only (very handy)

Backup Solutions Comparison

	Snapshot API	Azure disk Backup	Velero	3rd Party Vendors
K8s Native	No	No	Yes	Yes
Learning Curve	Easy	Easy	Medium	Medium
Automation	Do it your own	Introduces automation but disks need to be chosen one by one	Just configure a schedule and the rest is handled	Just configure a schedule and the rest is handled
Application Aware	No	No	Yes, Backup/Restore hooks	Yes, Backup/Restore hooks
Cost	Minimal	Minimal	Minimal (operation cost)	$$$

Multi-Region Statefulsets

- Statefulsets can’t be stretched across clusters/regions and Kubernetes federation isn’t a feasible
- Best solution is application level synchronous writes replication, requires customization and know how.
- Some specialized operators can handle this i.e. Crunchy Data (PG), CoackroachDB, etc….

Cluster with Statefulsets Upgrade Options

1. **In place upgrades**
   - The simplest option, yet carries the risk in case of upgrade failure
   - Requires to have the right architectural design where you can tolerate a zone or a node pool failure
   - Never do the whole cluster at once, Control Plane First then NodePool/Zone by NodePool/Zone
2. **Blue\Green with new node pools**
   - Adds complexity, but provides some safeguards to the upgrade process
   - Carries a risk as you will need to upgrade the control plane first
3. **Blue\Green new clusters**
   - The safest option yet the most complex and requires high maturity
   - Some workloads are too complex to be handled in a blue/green pattern
   - Migrating data from one cluster to the other is the most difficult part
     - if downtime/writes pause is fine then Velero or detach/attach disks will suffice
     - if no downtime/no writes pause, then a 3rd party or a synchronous replication from the stateful set across clusters would be required

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/diskEncryptionSets/{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
```

RBAC to secure storage

- 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"]
```

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 when possible
- Make use of the Azure Advisor recommendations

Bonus Slide: should I use burstable VMs (B series)?

- 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

Note: This is not the same as VM level disk bursting : https://azure.microsoft.com/updates/virtual-machine-vm-bursting-is-now-generally-available-on-more-vm-types/

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

Container Storage partner ecosystem

Storage management	portworx by Pure Storage STORAGEOS ROBIN
Data protection	kasten by Veeam portworx by Pure Storage VELERO RackWare Zerto COMMVAULT
Managed Kubernetes	Giant Swarm ROBIN

https://docs.microsoft.com/azure/storage/solution-integration/validated-partners/container-solutions/partner-overview

CNCF Storage Projects (OSS)

Rook

Graduated

Storage management with block and file. Initiated as interface on CEPH FS.

Vitess

Graduated

Scalable Database management for mySQL

OpenEBS

Sandbox

Storage management for Block volumes

Longhorn

Sandbox

Distributed Block Storage with inbuilt data protection and DR

ChubaoFS

Sandbox

POSIX-compliant and S3-compatible filesystem

Piraeus-Datastore

Sandbox

Scalable local persistent volumes

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

Next Steps

- Demos and walkthroughs
  https://github.com/mohmdnofal/aks-best-practices/tree/master/stateful_workloads

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

- Customer Case Studies https://aka.ms/aks/casestudy

Thank You