AKS, ACI, VK, OSS Tools

Ben Griffin | Azure Open Source GBB | OSS Tech Lead | Australia & NZ

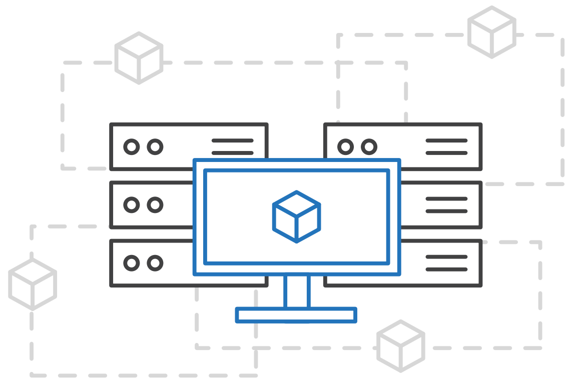
ben.griffin@microsoft.com | @benitogriffin

Kubernetes: empowering you to do more

- Deploy your applications quickly and predictably
- Scale your applications on the fly
- Roll out new features seamlessly
- Limit hardware usage to required resources only

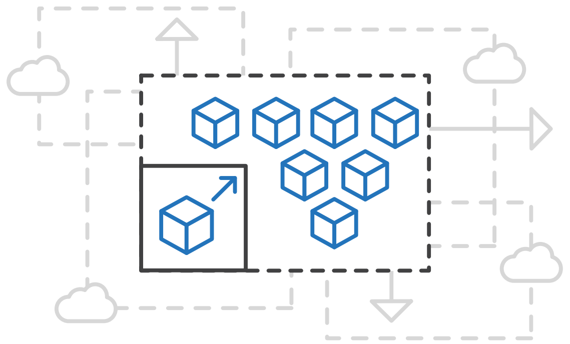
Azure Kubernetes Service (AKS)

Simplify the deployment, management, and operations of Kubernetes



Focus on your containers not the infrastructure

Work how you want with open-source APIs



Scale and run applications with confidence

The elements of orchestration

- Scheduling
- Affinity/anti-affinity
- Health monitoring
- Failover
- Scaling
- Networking
- Service discovery
- Coordinated app upgrades

What is Kubernetes

- Kubernetes is a container orchestrator comes from the Greek word κυβερνήτης:, which means *helmsman* or *ship pilot*. With this analogy in mind, we can think of Kubernetes as the manager for shipping containers
- Kubernetes was started by Google and is highly inspired by the Google Borg and later Omega system with its v1.0 release in July 2015, Google donated it to the Cloud Native Computing Foundation (CNCF)
- Brendan Burns who worked for Google, was one of the co-founders and now works for Microsoft and leads the containers team
- Kubernetes manages applications and not machines
- Kubernetes is typically used with microservice design patterns

What are the features of Kubernetes ?

- Building clusters from virtual or physical infrastructure (cloud & bare metal)
- High availability configurations
- Automated deployment rollout and rollback
- Seamless horizontal scaling
- Secret Management
- Service discovery and load balancing
- Simple log collection
- Stateful application support
- Persistent volume management
- CPU/memory quotas

AKS – Azure Kubernetes Service
Current feature set

- AKS is “Kubernetes as a service”
- 100% upstream Kubernetes
- Deploy in your own VNET
- RBAC with AAD Integration (Preview)
- GPU Support – Deploy Azure VM Type NC|ND|NV
- Terraform Support
- HTTP Application Routing
- Monitoring Support
- Free Hosted Control Plane

What are the features of AKS - GA

Feature	Description
Hosted control plane	No VMs to manage, control plane availability handled by Azure
Nodes in new VNET	Cluster nodes provisioned into a new VNET in your subscription
Node scale up/down (upto 50 VM's)	Increase or decrease cluster capacity using the Azure CLI or Portal or REST API
Kubernetes patch-release upgrades	Upgrade your cluster between Kubernetes patch releases (e.g. 1.7 to 1.8)
Kubernetes minor-release upgrades	Upgrade your cluster between Kubernetes minor releases (e.g. 1.7.7 to 1.7.9)
Node OS updates (on upgrade)	Node operating system is automatically updated including most recent security fixes on every upgrade, including patch or minor release upgrades
Node OS security updates (nightly)	Nodes automatically apply operating system security patches on a nightly schedule, does not include node reboot
Persistent disks	Attach Kubernetes persistent volumes to your applications with support for Azure Files and persistent disks
Azure load balancer	Automatically provision and attach Azure Load Balancers to your Kubernetes cluster
User access profiles	Users obtain cluster access using Azure APIs rather than requiring cluster SSH access

AKS Agent Nodes (Ubuntu VM’s)

- Customers only pay for the nodes in their cluster at no additional cost over the base VM price. Three node cluster is charged as three nodes

- There is no cost for the control plane, where the Kubernetes masters sit

- Managed solution, so no VM’s to manage, however you need to schedule a reboot to apply fixes, i.e. updated kernels

Kubernetes Add-Ons

Add-ons extend the functionality of Kubernetes.

- Kube-dns
- Heapster
- Dashboard
- Registry
- Storage-class
- Tiller
- metrics-server (will replace tiller in 1.11)
- http_application_routing (AKS specific add-on)

Release automation tools

Simplifying the Kubernetes experience

HELM

The package manager for Kubernetes

DRAFT

Streamlined Kubernetes development

BRIGADE

Event-driven scripting for Kubernetes

KASHTI

Visualization dashboard for Brigade

Helm

The best way to find, share, and use software built for Kubernetes

Manage complexity

Charts can describe complex apps; provide repeatable app installs, and serve as a single point of authority

Easy updates

Take the pain out of updates with in-place upgrades and custom hooks

Simple sharing

Charts are easy to version, share, and host on public or private servers

Rollbacks

Use `helm rollout` to roll back to an older version of a release with ease

Helm: The Kubernetes Package Manager

Helm is a tool that streamlines installing and managing Kubernetes applications. Think of it like apt/yum/homebrew for Kubernetes.

- Helm has two parts: a client (helm) and a server (tiller)

- Tiller runs inside of your Kubernetes cluster, and manages releases (installations) of your charts.
- Helm runs on your laptop, CI/CD, or wherever you want it to run.

Tiller server

- In-cluster server that interacts with the client and interfaces with the Kubernetes API server

Helm (Client) Architecture

Helm client is the CLI for end users

- Written in Go, uses gRPC to interact with the server
- Sends charts and values to Tiller for install, upgrade, etc.
- Charts can be stored on disk, or fetched from remote chart repositories
  - Any HTTP server that can house YAML/tar files (Azure, GitHub pages, etc.)
  - Public repo with community supported charts (eg – Jenkins, Mongo, etc.)

Charts are Helm packages that contain :

- Metadata – description of the package in Chart.yaml
- Kubernetes resource definitions (manifest files)
- Configuration – values.yaml
- Documentation

https://github.com/kubernetes/charts

Draft: Streamlines Kubernetes Development

```
# To set up draft (after prerequisites are installed)
$ draft init

# To containerize your application based on Draft packs
$ draft create

# To deploy your application to a Kubernetes dev sandbox, accessible using draft
# to connect over a secured tunnel.
$ draft up
```

Use a local editor to modify the application, with changes deployed to Kubernetes in seconds.

Draft: Streamlines Kubernetes Development

What Draft Does Not Do

Draft should not be used as the following:

- A tool to be used in production deployments
- A tool to be used in replacement for a CI/CD pipeline
- A PaaS like Deis Workflow

Azure Kubernetes Service (AKS)

Deployment : Get started easily

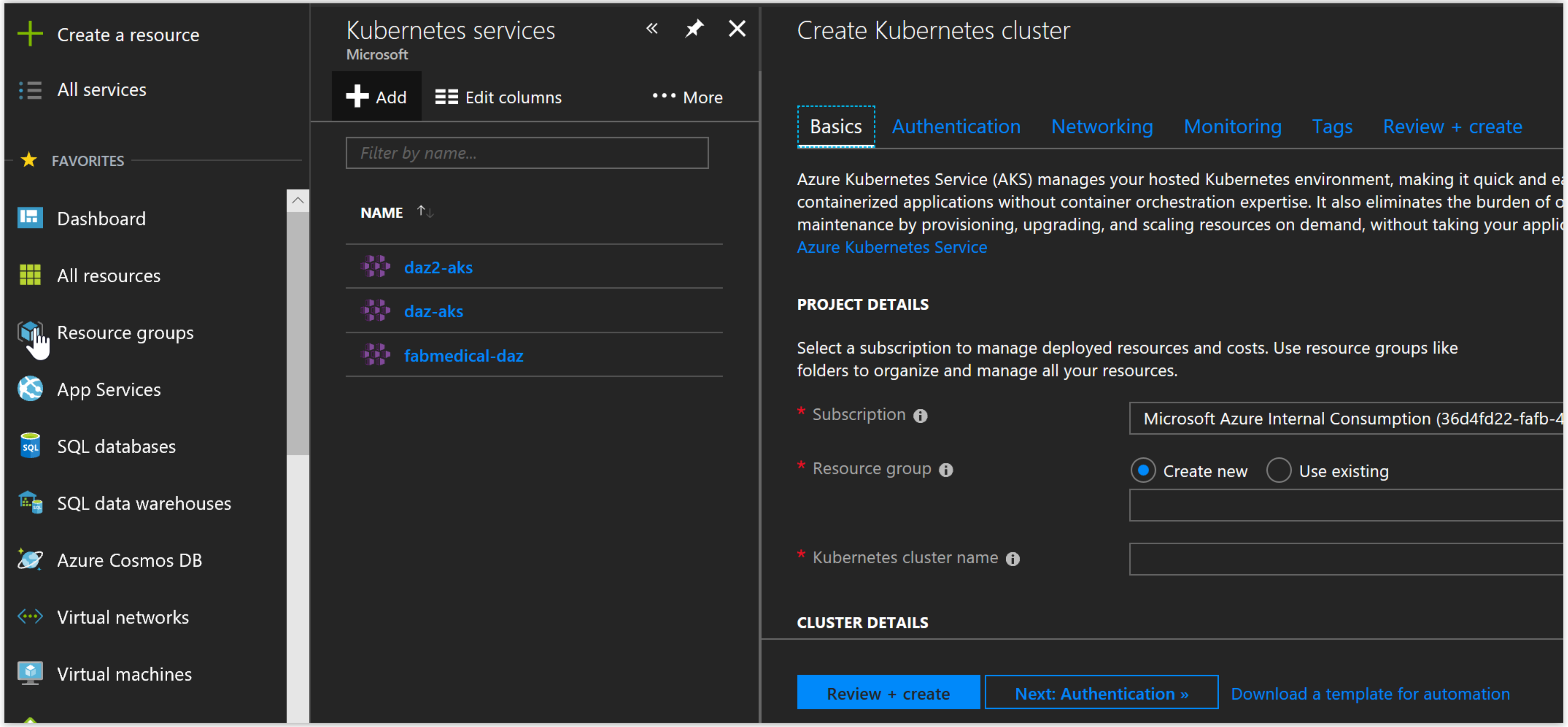
```
$ az group create --name my-ResourceGroup --location southeastasia

$ az aks create -g myResourceGroup -n myCluster -c 3  --generate-ssh-keys
\ Running ..

$ az aks install-cli
Downloading client to /usr/local/bin/kubectl ..

$ az aks get-credentials -g myResourceGroup -n myCluster
Merged "myCluster" as current context ..

$ kubectl get nodes
NAME                        STATUS    AGE       VERSION
aks-mycluster-36851231-0    Ready     4m        v1.8.1
aks-mycluster-36851231-1    Ready     4m        v1.8.1
aks-mycluster-36851231-2    Ready     4m        v1.8.1
```

Azure Kubernetes Service (AKS)

```
$ az aks list -o table
Name                Location    ResourceGroup    KubernetesRelease
ProvisioningState
------------------  ----------  --------------  -------------------  ----------------
---
myCluster           westus2     myResourceGroup              1.7.7  Succeeded

$ az aks upgrade -g myResourceGroup -n myCluster --kubernetes-version 1.8.1
\ Running ..

$ kubectl get nodes
NAME                        STATUS    AGE       VERSION
aks-mycluster-36851231-0   Ready     12m       v1.8.1
aks-mycluster-36851231-1   Ready     8m        v1.8.1
aks-mycluster-36851231-2   Ready     3m        v1.8.1

$ az aks scale -g myResourceGroup -n myCluster --agent-count 10
\ Running ..
```

Azure Kubernetes Service (AKS)

Deployment from the Azure Portal



daz2-aks
daz-aks
fabmedical-daz

Create Kubernetes cluster

Basics | Authentication | Networking | Monitoring | Tags | Review + create

Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment, making it quick and ea
containerized applications without container orchestration expertise. It also eliminates the burden of o
maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applic
Azure Kubernetes Service

PROJECT DETAILS

Select a subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

* Subscription: Microsoft Azure Internal Consumption (36d4fd22-fafb-4

* Resource group: [✓ Create new] Use existing

* Kubernetes cluster name:

CLUSTER DETAILS

Config Maps and Secrets

A good practice when writing applications is to separate application code from configuration. We want to enable application authors to easily employ this pattern within Kubernetes. While the Secrets API allows separating information like credentials and keys from an application, no object existed in the past for ordinary, non-secret configuration.

Secret values are base64 encoded and automatically decoded for pods.

The ConfigMap API is simple conceptually. From a data perspective, the ConfigMap type is just a set of key-value pairs. Applications are configured in different ways, so we need to be flexible about how we let users store and consume configuration data. There are three ways to consume a ConfigMap in a pod:

- Command line arguments
- Environment variables
- Files in a volume

Namespaces

- Allow for multiple virtual clusters backed by the same physical cluster
- Logical separation
- Namespace used in FQDN of Kubernetes services
  - Eg - <service-name>.<namespace-name>.svc.cluster.local
- Every Kubernetes resource type is scoped to a namespace (except for nodes, persistentVolumes, etc.)
- Intended for environments with many users, teams, projects

Role Based Access Control

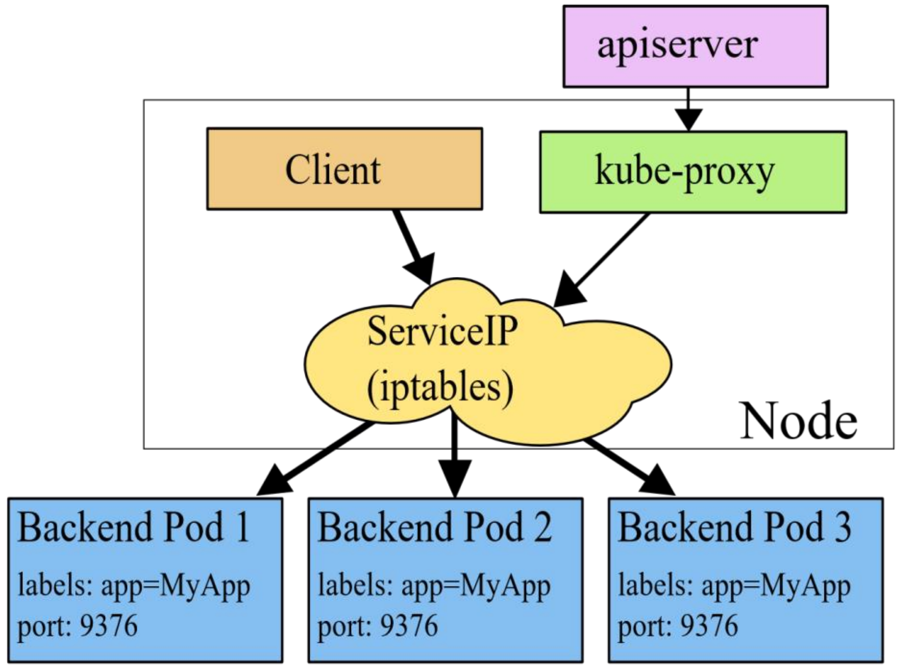
- Allows for dynamic policies against k8s API for authZ
- In version 1.6, this is a beta feature
- Can create both "Roles" and "ClusterRoles"
- Can configure api-server and kubectl to use Azure AD for access
- To grant permissions:
  - Use "RoleBindings" for within namespaces
  - Use "ClusterRoleBindings" for cluster-wide access
  - Can contain users, groups, service accounts
  - Control access to specific resources or API verbs

```yaml
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["batch", "extensions"]
  resources: ["jobs"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

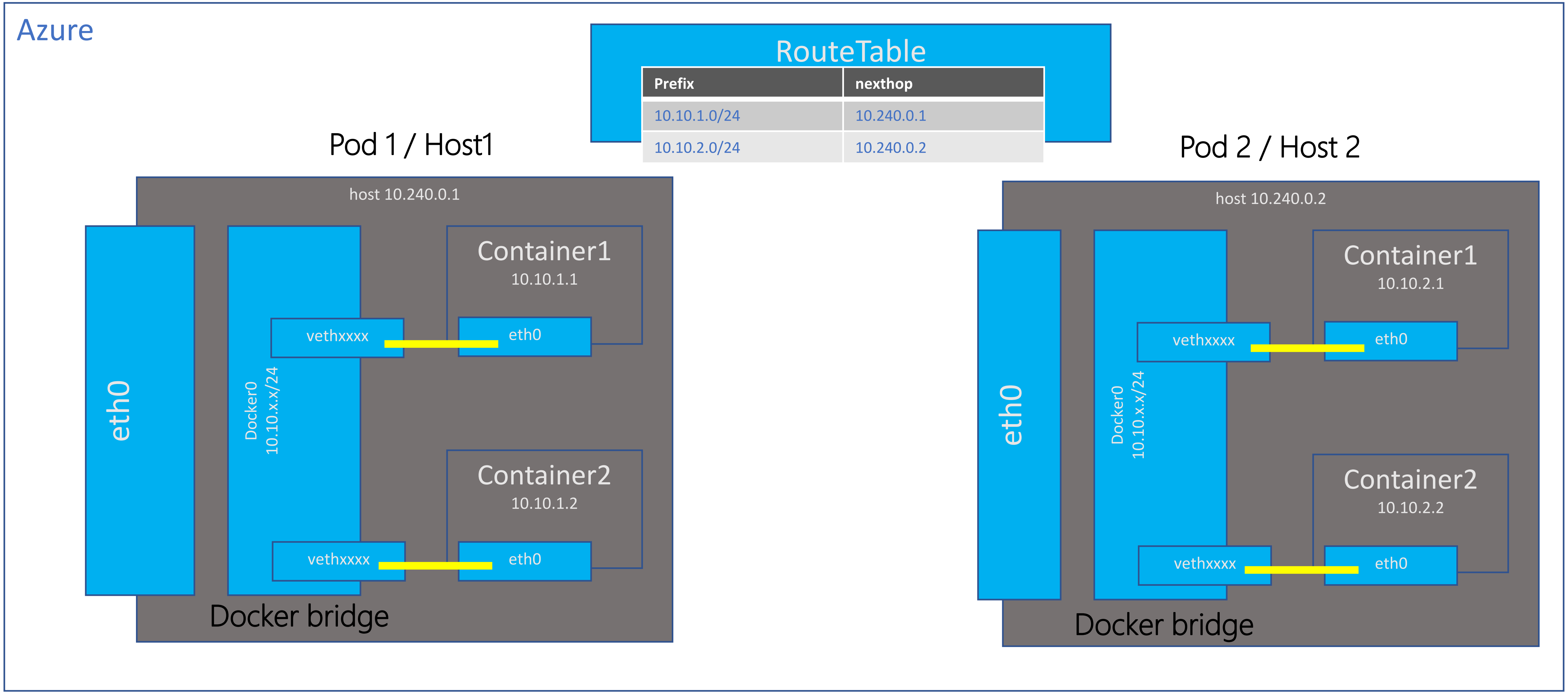
Networking in Kubernetes - Summary

Kubernetes Networking model introduces 3 methods of communications:

- Pod-to-Pod communication directly by IP address. Kubernetes has a Pod-IP wide metric simplifying communication.
- Pod-to-Service Communication – Client traffic is directed to service virtual IP by iptables rules that are modified by the kub-proxy process (running on all hosts) and directed to the correct Pod.
- External-to-Internal Communication – external access is captured by an external load balancer which targets nodes in a cluster. The Pod-to-Service flow stays the same.



Pod Network in AKS



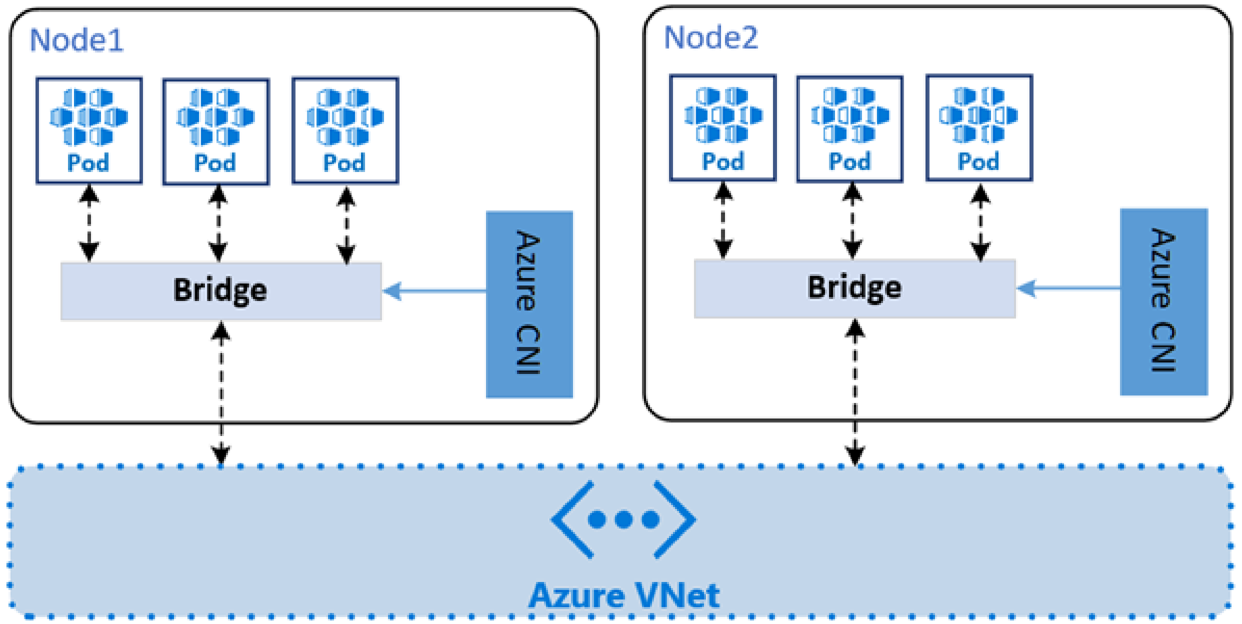
Prefix	nexthop
10.10.1.0/24	10.240.0.1
10.10.2.0/24	10.240.0.2

Kubernetes Networking : Basis vs Advanced

Basic – Uses kubenet plugin

Advanced – Uses CNI plugin, AKS can communicate with your existing VNET’s, and your pods can access services via the virtual VNet endpoints

Network design



Nodes in an AKS cluster configured for Advanced networking use the Azure Container Networking Interface (CNI) Kubernetes plugin.

https://docs.microsoft.com/en-us/azure/aks/networking-overview
https://github.com/Azure/azure-container-networking/blob/master/docs/acs.md

Kubernetes Networking : Ingress

All traffic that ends up at an edge router is either dropped or forwarded elsewhere
Ingress is a collection of rules that allow inbound connections to reach the cluster services
An Ingress controller is responsible for fulfilling the Ingress, usually with a loadbalancer, though it may also configure your edge router or additional frontends to help handle the traffic in an HA manner

- Several ways of calling a service from outside the cluster
  - Callable from within the cluster only
  - Load Balanced on a fixed IP (Type=LoadBalancer)
  - Mapped to a specific port on every Node (Type=NodePort)
  - An Ingress resource is a service that acts as a reverse proxy and is exposed on a specific IP.

Kubernetes Network policy

- By default, pods are non-isolated
- Recommend to create default policies that apply to all pods
  - Default deny all ingress traffic
  - Default allow all ingress traffic
  - Default deny all egress traffic.
  - Default allow all egress traffic
  - Default deny all ingress and all egress traffic

Network policy recipes
https://github.com/ahmetb/kubernetes-network-policy-recipes

Azure’s Network Policy Manager (Azure-NPM) Supported mode : acs-engine Only

- We are happy to announce the availability of Azure’s Network Policy Manager (Azure-NPM), Azure’s implementation of Kubernetes Network Policies.

- The Azure-NPM supports all the capabilities described in Kubernetes’ Policy specification
- https://kubernetes.io/docs/concepts/services-networking/network-policies/

- Docker image : Azure Network Policy Manager : https://hub.docker.com/r/containernetworking/azure-npm/

- It’s open source. The code is available on our Github Repository https:/github.com/Azure/azure-container-networking/tree/master/npm

- Integration with acs-engine is finished and networkPolicy==”azure” is supported in cluster configuration file.
- It is coming soon to AKS clusters.
- This is supported on Linux nodes. Windows support will come in near future.
- We have tested this with Azure CNI.

Network policy

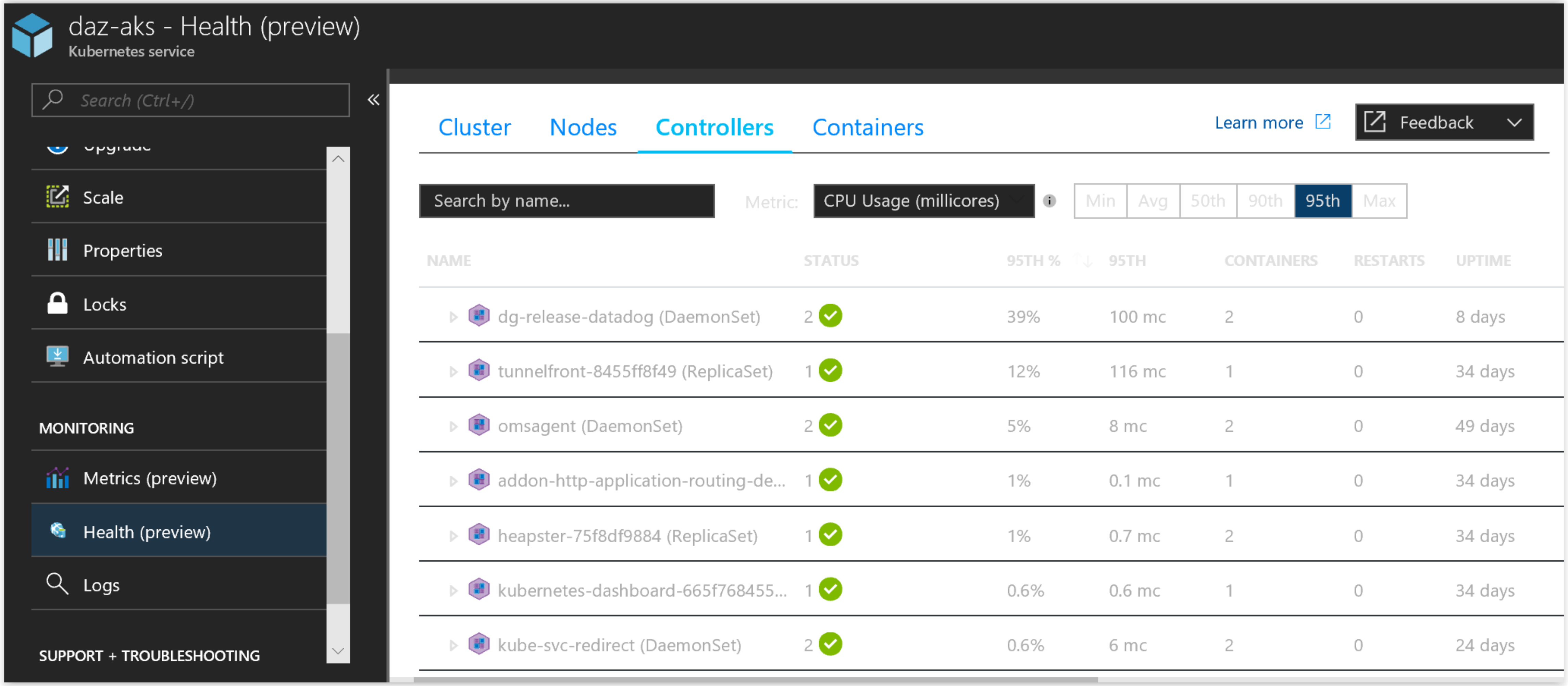
- Pod Selector
- PolicyTypes
  - Ingress, egress

Ingress

- namespaceSelector
- podSelector
- ipBlock

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

AKS Health Monitoring

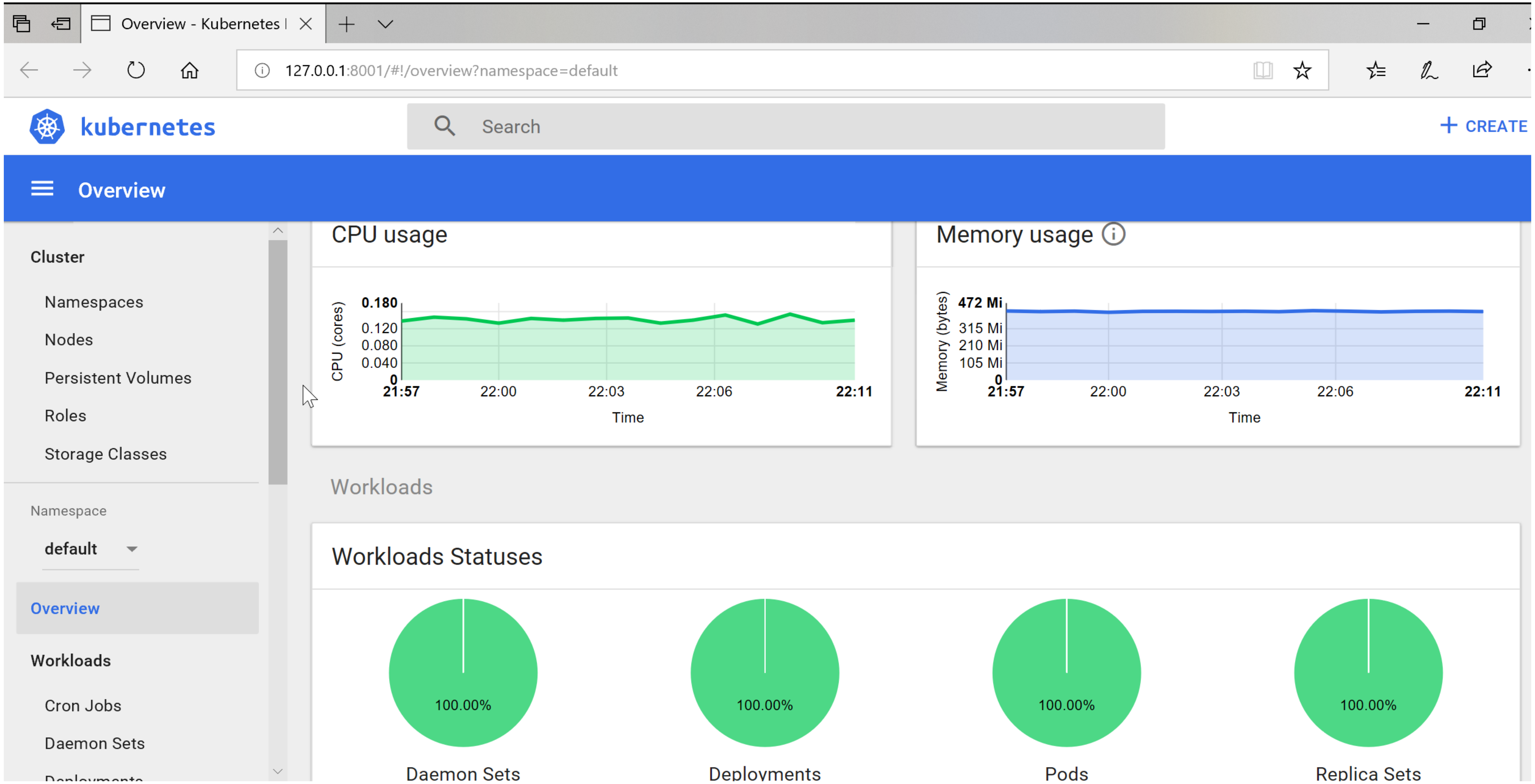


daz-aks - Health (preview)
Kubernetes service

Metric: CPU Usage (millicores)   Min | Avg | 50th | 90th | [✓ 95th] | Max

NAME	STATUS	95TH %	95TH	CONTAINERS	RESTARTS	UPTIME
dg-release-datadog (DaemonSet)	2 ✓	39%	100 mc	2	0	8 days
tunnelfront-8455ff8f49 (ReplicaSet)	1 ✓	12%	116 mc	1	0	34 days
omsagent (DaemonSet)	2 ✓	5%	8 mc	2	0	49 days
addon-http-application-routing-de...	1 ✓	1%	0.1 mc	1	0	34 days
heapster-75f8df9884 (ReplicaSet)	1 ✓	1%	0.7 mc	2	0	34 days
kubernetes-dashboard-665f768455...	1 ✓	0.6%	0.6 mc	1	0	34 days
kube-svc-redirect (DaemonSet)	2 ✓	0.6%	6 mc	2	0	24 days

Kubernetes Dashboard



Kubernetes Dashboard

Kubernetes includes a web dashboard that can be used for basic management operations.

This command creates a proxy between your development system and the Kubernetes API, and opens a web browser to the Kubernetes dashboard.

```
# Run this command on your desktop (Windows, MacOS), so you'll need to install
# azure-cli and kubectl

    $ az aks browse -g k8s-cluster-aks-rg -n k8s-cluster-aks
```

Goto https://127.0.0.1:8001/ in your browser to access the dashboard.

Kubernetes Storage

Volumes – A Kubernetes abstraction for persistent storage. Kubernetes supports many types of volumes, such as NFS, Ceph, GlusterFS, local directory, etc.

On-disk files in a container are ephemeral, which presents some problems for non-trivial applications when running in containers.

First, when a container crashes, kubelet will restart it, but the files will be lost - the container starts with a clean state.

Second, when running containers together in a Pod it is often necessary to share files between those containers. The Kubernetes Volume abstraction solves both of these problems.

Kubernetes Storage : Volume Types

- Volume types:
  - Azure Disk
  - Azure Files
  - cephfs
  - flocker
  - glusterfs
  - nfs
  - portworxVolume

Azure Files & Azure Disk

Storage Class

StorageClass – allows for dynamic provisioning of PersistentVolumes

Name: myDefaultStorageClass
Provisioner: kubernetes.io/azure-disk

Persistant Volume Claim

A PVC is a request for a persistant volume based on a storage class.

Mount in Pod.yaml

In a POD a persistant volume can be mounted.

Kubernetes Storage : Access Modes

Volume Plugin	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
AzureFiles	✓	✓	✓
AzureDisk	✓	-	-
CephFS	✓	✓	✓
Cinder	✓	-	-
FlexVolume	✓	✓	-
Flocker	✓	-	-
Glusterfs	✓	✓	✓
NFS	✓	✓	✓
PortworxVolume	✓	-	✓

https://kubernetes.io/docs/concepts/storage/persistent-volumes/#persistent-volumes

Kubernetes Storage : storage in yaml format

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: wp-pv-claim
  labels:
    app: wordpress
  annotations:
    volume.beta.kubernetes.io/storage-class: premium-managed
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 32Gi
```

https://docs.microsoft.com/en-us/azure/aks/azure-disks-dynamic-pv

https://docs.microsoft.com/en-us/azure/aks/azure-files-dynamic-pv

https://github.com/Azure/blackbelt-aks-hackfest/blob/master/labs/day2-labs/persistent-volumes.md

Application Scaling Strategies

These functions should be coordinated

Infrastructure Level

- Increase/decrease the number of VM's running in the cluster
- Azure infrastructure function
- via AKS API/CLI (utilizes Azure Availability Sets)
- Node auto-scaling coming to AKS soon

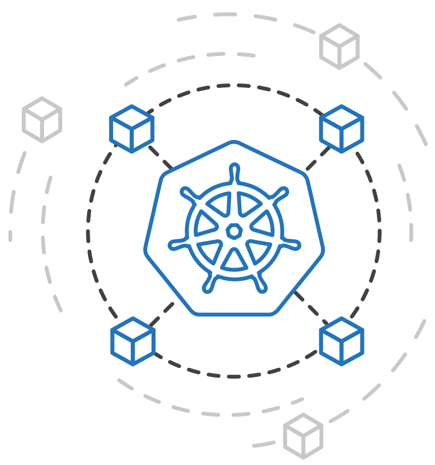
Application Level

- Increase/decrease the number of pods for a given service
- Handled by the orchestrator or the application itself
- Kubernetes Horizontal Pod Autoscaling

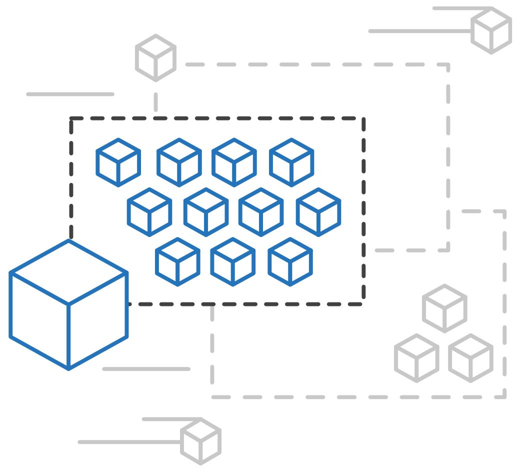
https://docs.microsoft.com/en-us/azure/aks/autoscaler

Azure Container Instances (ACI)

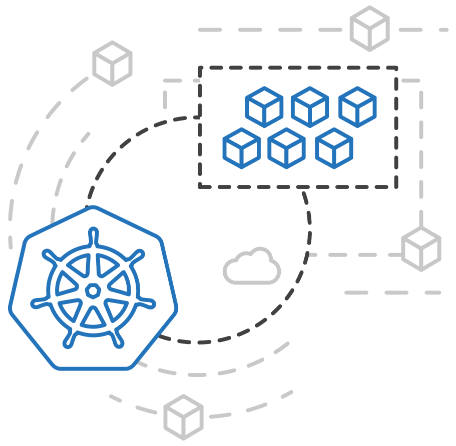
ACI Connector for Kubernetes



Kubernetes provides rich orchestration capabilities



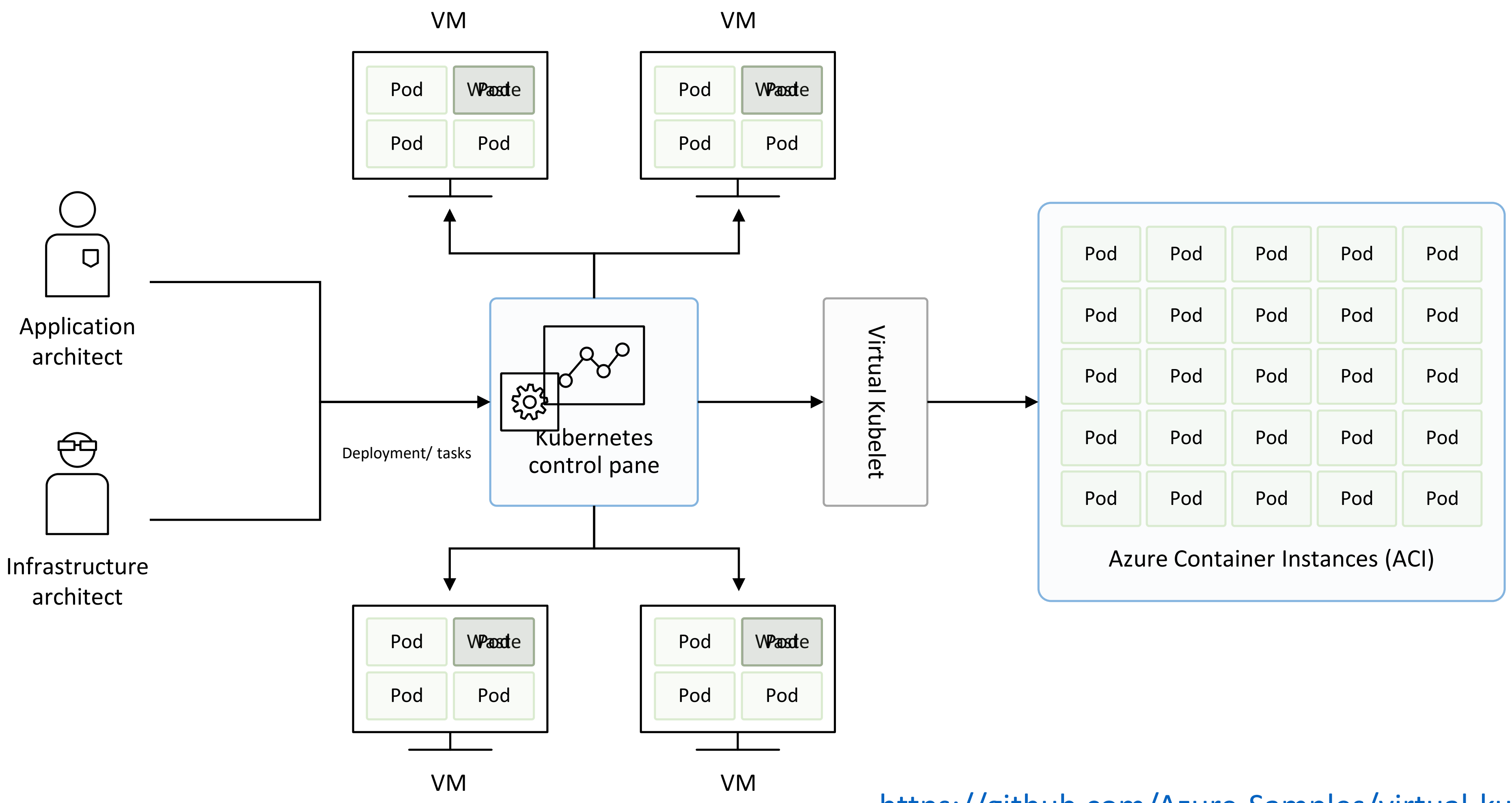
ACI provides infinite container-based scale



The ACI Connector for K8s brings them together

Azure Container Instances (ACI)

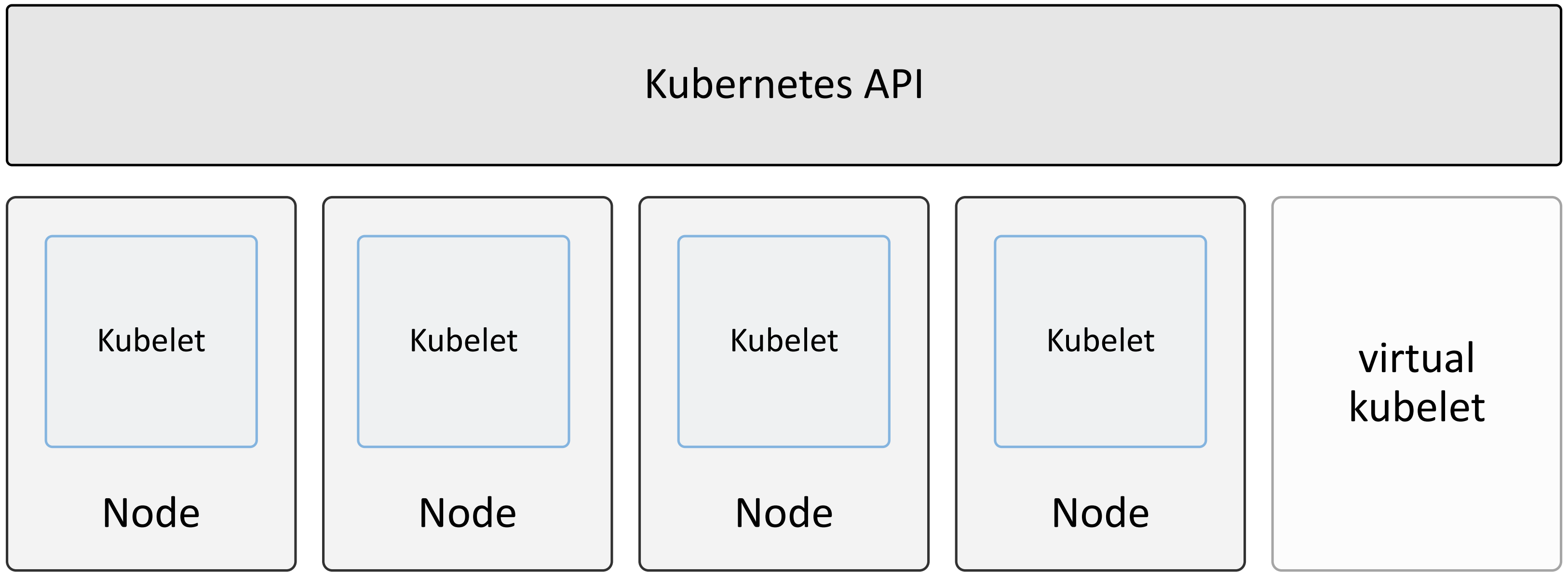
Bursting with Virtual Kubelet



https://github.com/Azure-Samples/virtual-kubelet-aci-burst

Azure Container Instances (ACI)

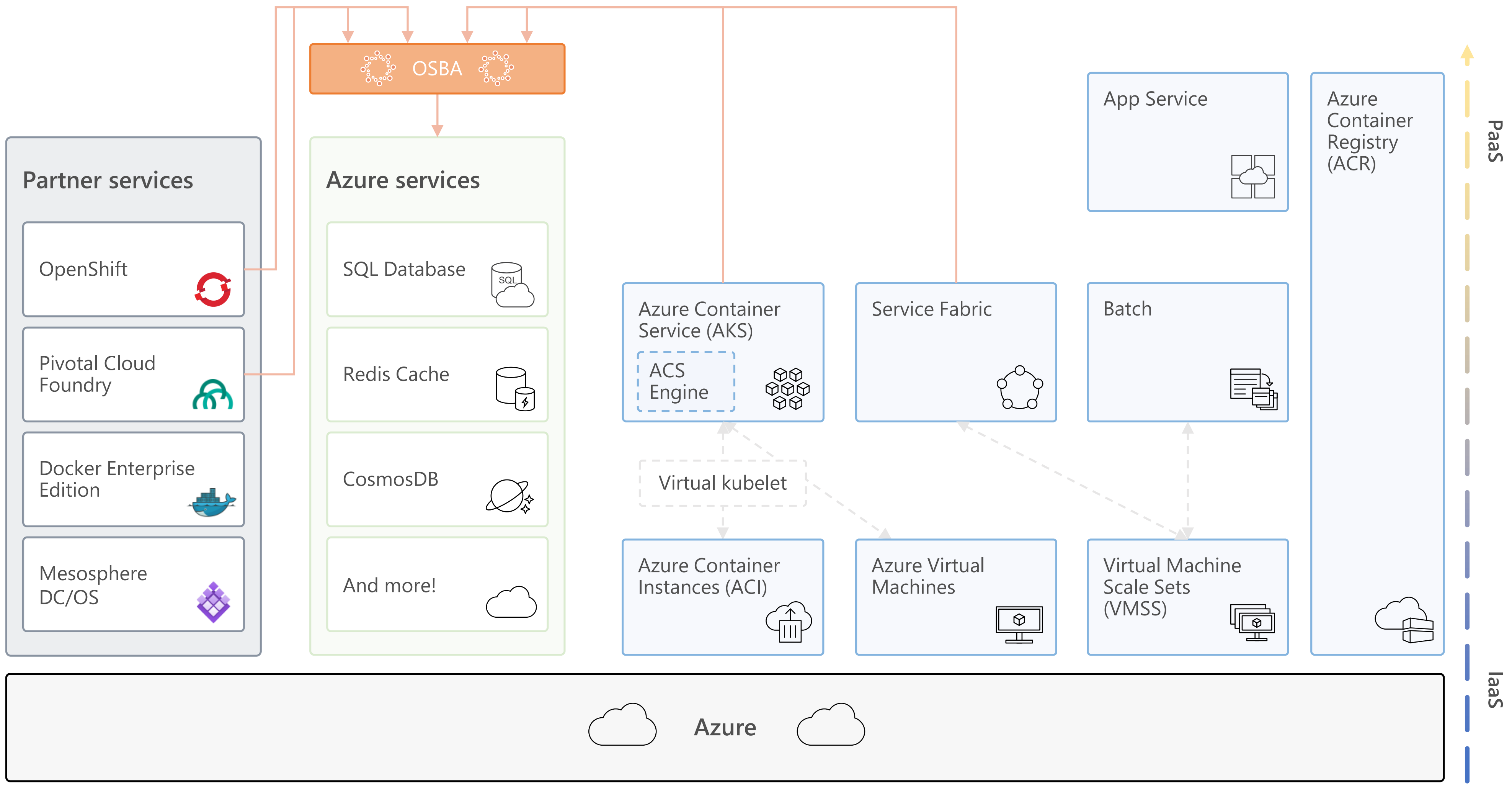
Virtual Kubelet



Typical kubelets implement the pod and container operations for each node as usual.

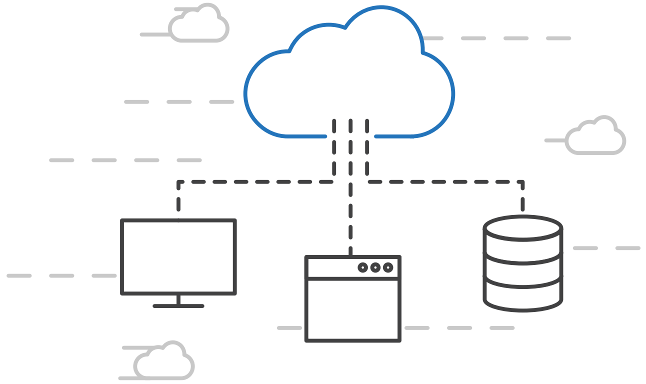
Virtual kublet registers itself as a “node” and allows developers to program their own behaviors for operations on pods and containers.

Open Service Broker for Azure (OSBA)

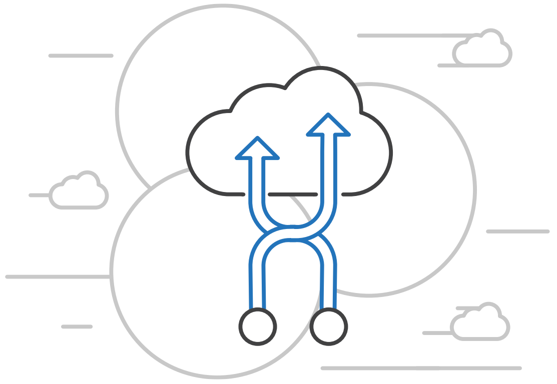


Open Service Broker for Azure (OSBA)

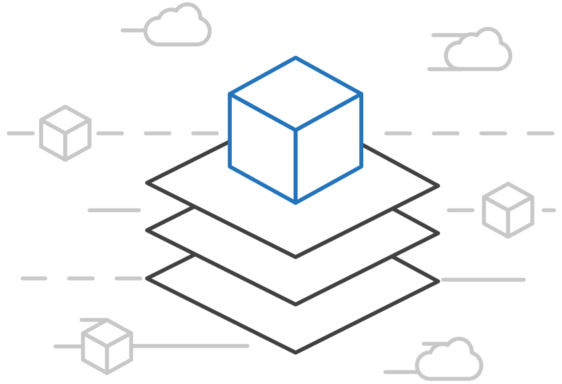
Connecting containers to Azure services and platforms



A standardized way to connect with Azure services



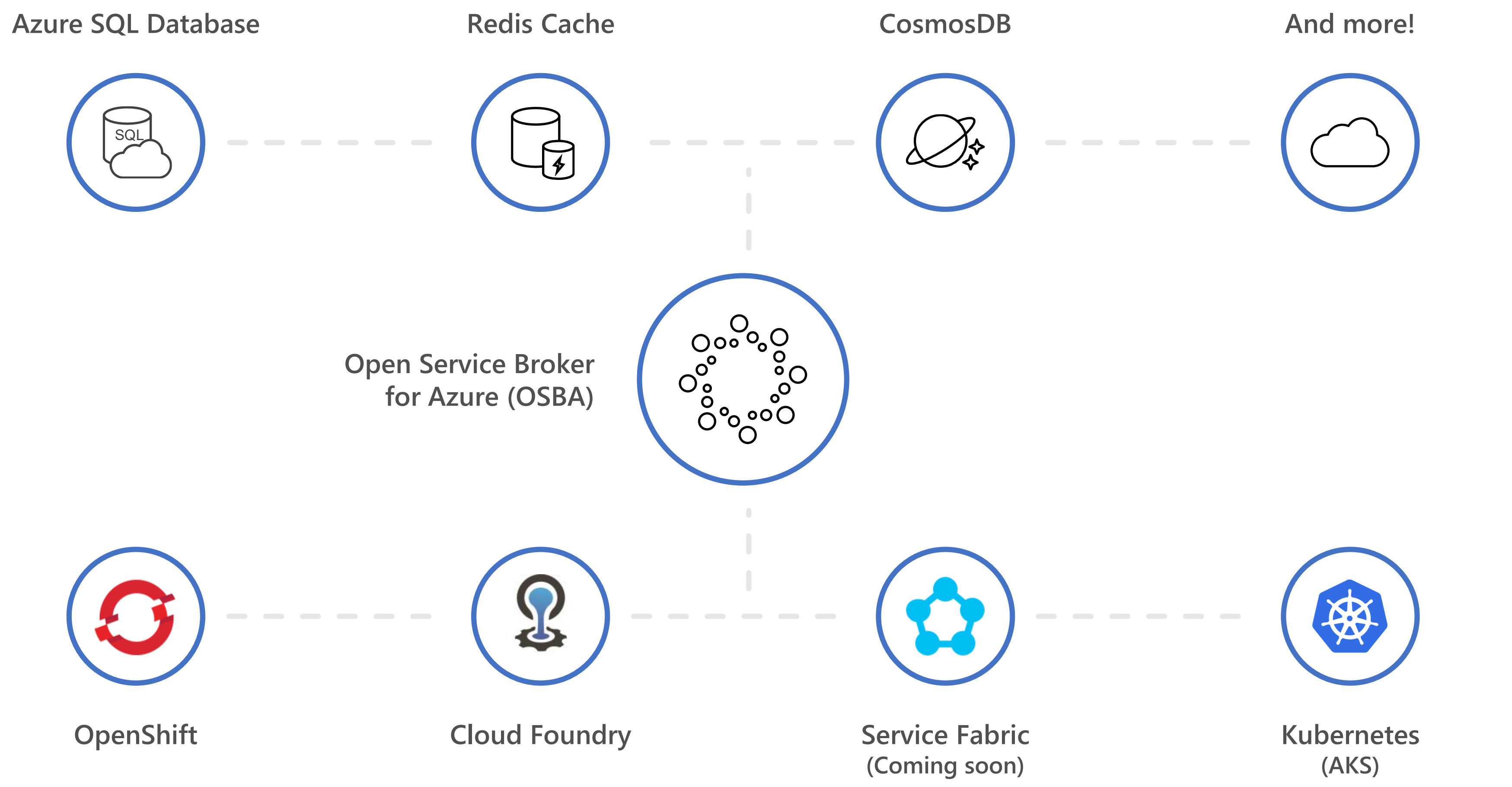
Simple and flexible service integration



Compatible across numerous platforms

Open Service Broker for Azure (OSBA)

An implementation of the Open Service Broker API



acs-engine

The Azure Container Service Engine (acs-engine) generates ARM (Azure Resource Manager) templates for Docker enabled clusters on Microsoft Azure with your choice of DC/OS, Kubernetes, or Swarm orchestrators.

The input to the tool is a cluster definition

Allows you to deploy the latest, but untested version of Kubernetes

acs-engine is not supported by Microsoft

https://github.com/Azure/acs-engine/releases

acs-engine

You can customize your Docker enabled cluster in many ways including:

- Choice of DC/OS, Kubernetes, OpenShift, Swarm Mode, or Swarm orchestrators
- Multiple agent pools where each agent pool can specify:
  - Standard or premium VM Sizes, including GPU optimized VM sizes
  - Node count
  - Virtual Machine ScaleSets or Availability Sets
  - Storage Account Disks or Managed Disks
  - OS (Linux and Windows – early alpha) and distro (Other than ubuntu, not recommended)
- Custom VNET
- Extensions

Kubernetes Monitoring Solutions

DATADOG

Sysdig

New Relic.

(AKS) container health (preview)

Prometheus

+More

Brigade

Run scriptable, automated tasks in the cloud — as part of your Kubernetes cluster

Simple, powerful pipes

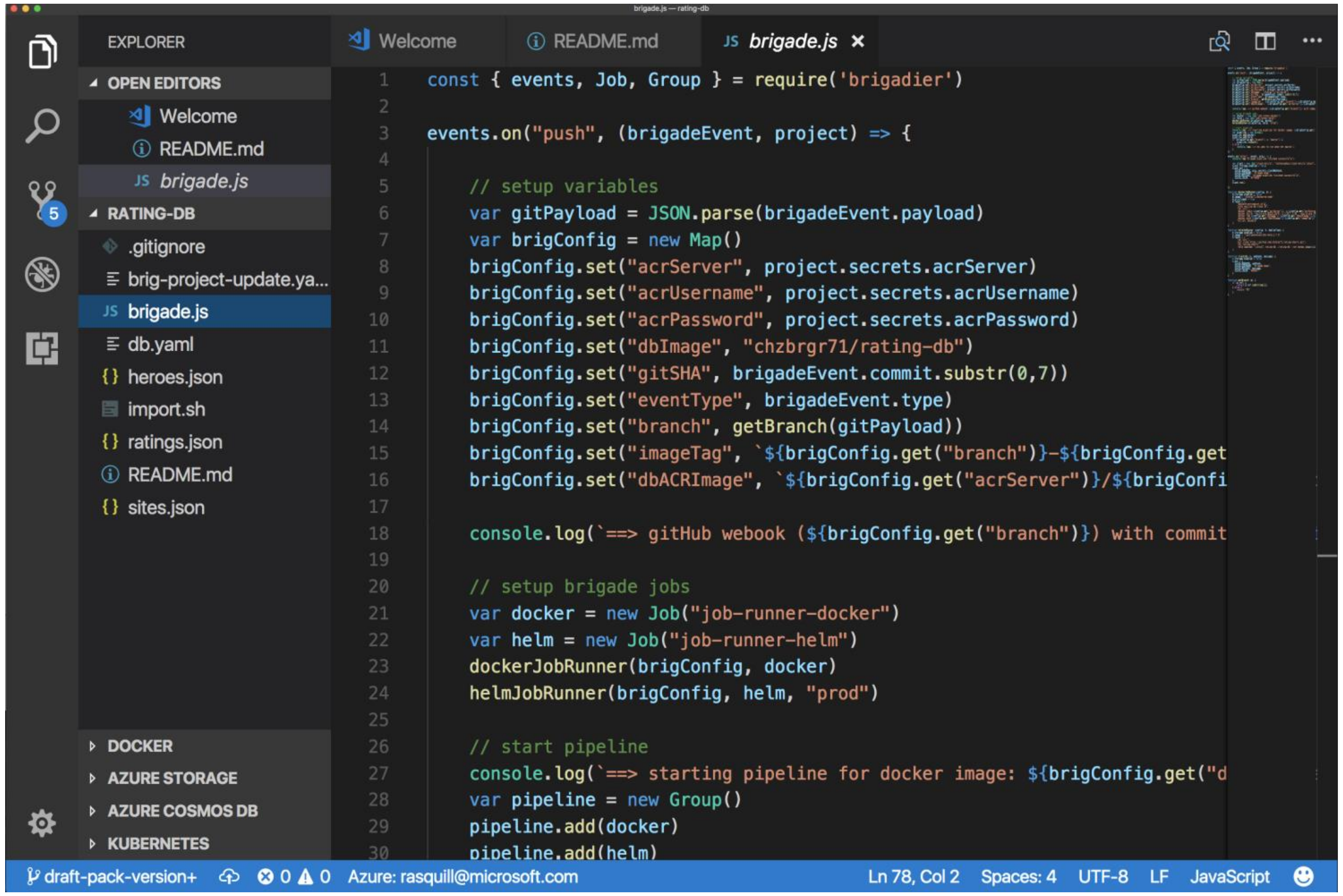
Each project gets a `brigade.js` config file, which is where you can write dynamic, interwoven pipelines and tasks for your Kubernetes cluster

Runs inside your cluster

By running Brigade as a service inside your Kubernetes cluster, you can harness the power of millions of available Docker images

Brigade

Brigade in action



```
const { events, Job, Group } = require('brigadier')

events.on("push", (brigadeEvent, project) => {

    // setup variables
    var gitPayload = JSON.parse(brigadeEvent.payload)
    var brigConfig = new Map()
    brigConfig.set("acrServer", project.secrets.acrServer)
    brigConfig.set("acrUsername", project.secrets.acrUsername)
    brigConfig.set("acrPassword", project.secrets.acrPassword)
    brigConfig.set("dbImage", "chzbrgr71/rating-db")
    brigConfig.set("gitSHA", brigadeEvent.commit.substr(0,7))
    brigConfig.set("eventType", brigadeEvent.type)
    brigConfig.set("branch", getBranch(gitPayload))
    brigConfig.set("imageTag", `${brigConfig.get("branch")}-${brigConfig.get
    brigConfig.set("dbACRImage", `${brigConfig.get("acrServer")}/${brigConfi

    console.log(`==> gitHub webook (${brigConfig.get("branch")}) with commit

    // setup brigade jobs
    var docker = new Job("job-runner-docker")
    var helm = new Job("job-runner-helm")
    dockerJobRunner(brigConfig, docker)
    helmJobRunner(brigConfig, helm, "prod")

    // start pipeline
    console.log(`==> starting pipeline for docker image: ${brigConfig.get("d
    var pipeline = new Group()
    pipeline.add(docker)
    pipeline.add(helm)
```

Azure

Siemens Health leverages technology to connect medical devices to the cloud through AKS

Digitization and networking between healthcare providers and software development companies are essential to value-based care. Moving from the development of value-added services into becoming more of a platform provider, it became important for Siemens to adopt a microservices approach to application delivery. To that end, Siemens adopted Azure Container Service (AKS) to run their microservices-based apps. AKS puts Siemens in a position not only to deploy business logic in Docker containers—including the orchestration—but also enables them to use an applicant gateway and API management to manage exposure, control, and to meter the access continuously. With their cloud-based development approach, Siemens has driven newfound product development agility. This project is already having a positive impact within the healthcare industry.

SIEMENS

Products and services	Organization size	Industry	Country	Business need
Microsoft Azure Container Service	100,000+ employees	Healthcare	Germany	Faster application development

Customer success stories

https://azure.microsoft.com/en-gb/case-studies/?service=kubernetes-service

https://customers.microsoft.com/en-gb/story/varian-health-provider-azure

https://customers.microsoft.com/en-gb/story/tryg-insurance-azure

Microsoft

Thank You!