Microsoft

Cloud Native Transformation of Applications Using Azure Kubernetes

Part 2 – Working with Kubernetes Objects

About the Speakers

Steve Caravajal, Ph.D. is Chief Cloud Solution Architect with extensive cloud experience, and 15+ yrs with Microsoft. He is focused on leading digital transformation and cloud native modernization for Microsoft's large strategic accounts. Steve is also Cloud Security lead, SME for multi-vendor cloud strategy, and K8s certified (CKA, CKAD).

David Hoerster, a former 6-time .NET MVP, has been working with the Microsoft.NET Framework since the early 1.0 betas and has recently found his next passion in Open Source technologies. He is currently a Cloud Solutions Architect at Microsoft specializing in application development and identity. He also recently earned his CKA and CKAD.

4 Week Agenda Overview

Week 1 – July 14
Containers, Azure Kubernetes Service (AKS), Azure Container Registry (ACR)
Establish foundation to enable advanced implementations and configuration in Weeks 2-4.

Week 2 – July 21
Storage, ConfigMaps, Namespace, Deployment Templates and YAML

Week 3 – July 28
AKS Networking, Managing Ingress and Container Security

Week 4 – Aug 4
Deploying a Distributed Application
Monitoring, and Service Mesh

Kubernetes Topics to be covered in 4 sessions

Overall Goal: Deploy a distributed app

Nodes / Pods	ReplicaSet	Deployment
Services	Namespace / Context	Storage / Volumes
config-map	Security / Secrets / AAD / KeyVault	Ingress / Egress
Monitoring / Logging	Data Management	Networking

Kubernetes Topics – Week 2

Nodes / Pods	ReplicaSet	Deployment
Services	Namespace / Context	Storage / Volumes
config-map	Security / Secrets / AAD / KeyVault	Ingress / Egress
Monitoring / Logging	Data Management	Networking

Learning Expectations for the 4 weeks

- What is containerization?
- Kubernetes architecture
- Storage, Clusters, nodes, and pods
- Deployments, jobs, and services
- Getting an application up and running
- Working with labels
- Handling application upgrades
- Configuration data
- Running jobs
- Production deployments
- Monitoring and logging
- Security in Kubernetes
- Kubernetes Networking

Learning Expectations for Week 2

Understand `kubectl` and its uses

Learn YAML schema elements for major Kubernetes objects

Deeper dive into major Kubernetes objects, such as

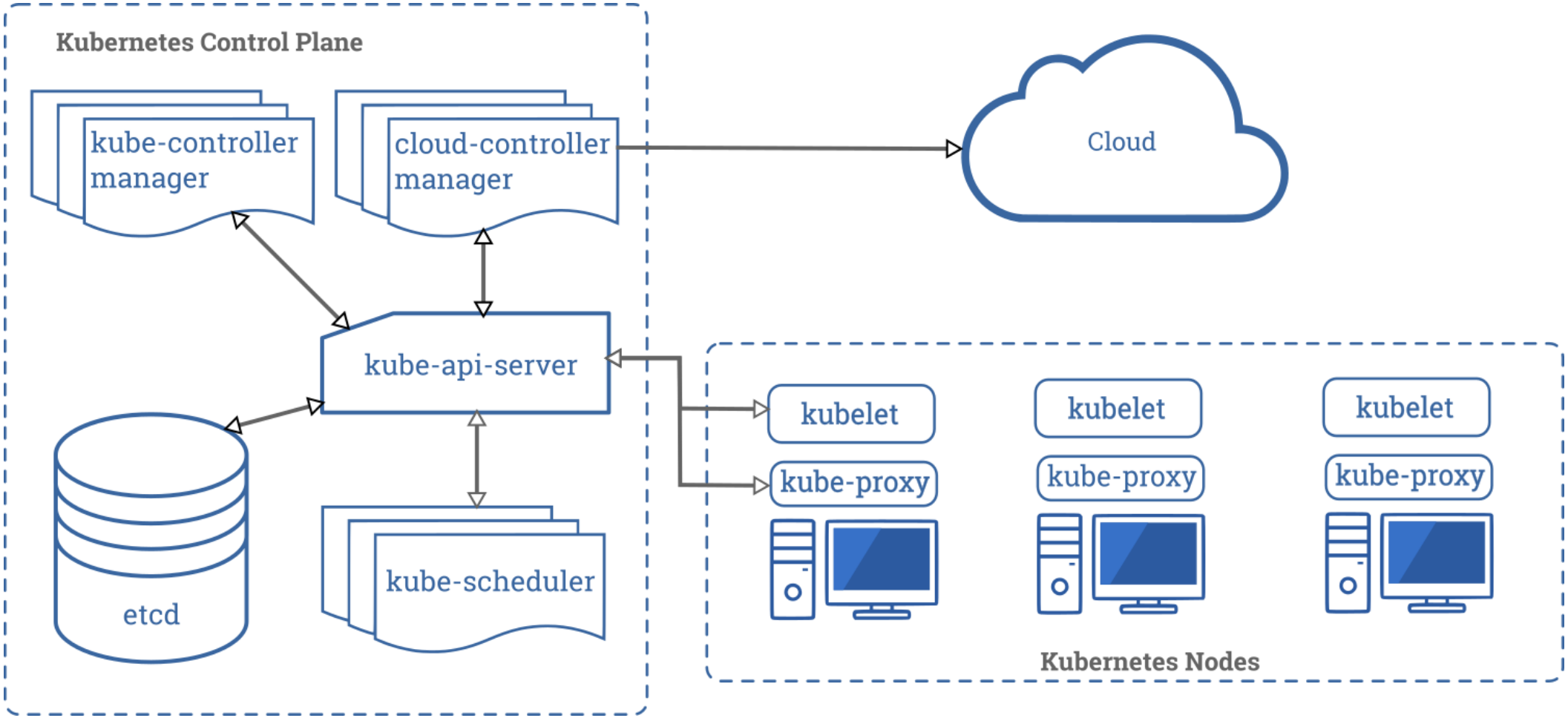
- Pods, Deployments, Services
- ConfigMaps, Secrets
- Storage and Mounting Storage

Bringing them Together for Deploying Applications

Microsoft

Cluster Components Review

Cluster Components



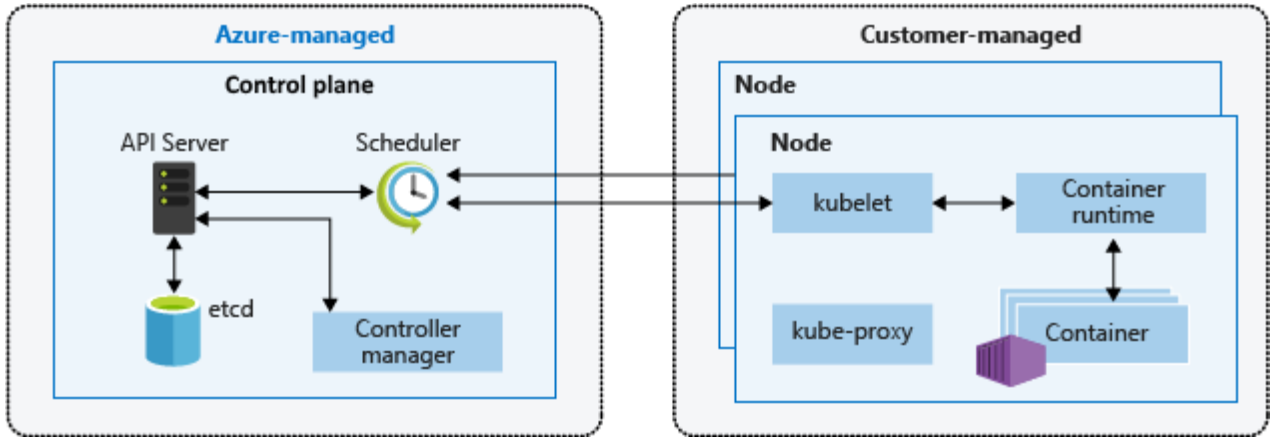
Source: https://kubernetes.io/docs/concepts/overview/components/

Cluster Components

- Control Plane Components
  - API Server
    - Exposes the Kubernetes API
    - Primary interaction point for users
  - Scheduler
    - Places pods on nodes
    - Keeps track of resource requirements and utilization
  - Etcd
    - Name/value store for all Kubernetes cluster data
  - Controller
    - Handles node health, replication, endpoint management, access tokens

Cluster Components

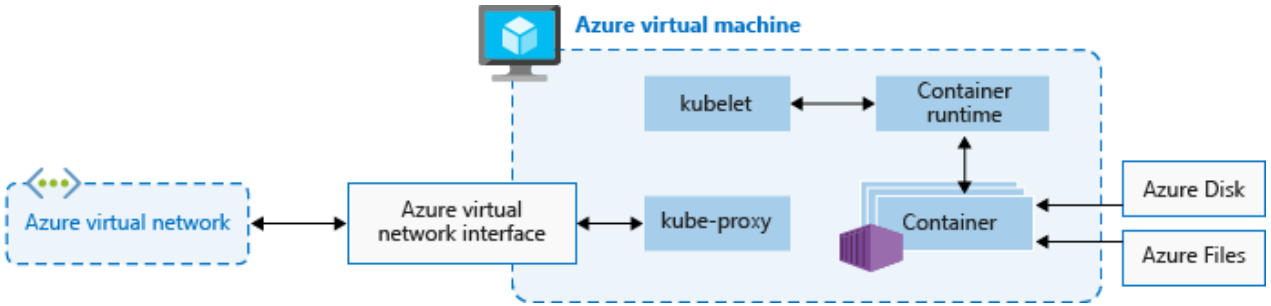
- Control Plane Components with AKS
  - These components are managed by Azure
  - AKS users do not have access
  - The PaaS part of AKS
- AKS Handles…
  - Backup and restore of control plane components
  - Backup and restore of etcd component
  - All control plane administrative operations



Source: https://docs.microsoft.com/en-us/azure/aks/media/concepts-clusters-workloads/control-plane-and-nod

Cluster Node Pools

- The IaaS part of AKS
- Groups of VMs that are your Kubernetes data plane
  - A node pool is a collection of VMs with the same configuration
- AKS can have multiple node pools
  - For example, a Linux node pool and a Windows node pool



Source: https://docs.microsoft.com/en-us/azure/aks/media/concepts-clusters-workloads/aks-node-resource-interactions.png

Microsoft

Using the Command Line - kubectl

Interacting with Your AKS Cluster

- `kubectl` is the primary Kubernetes command line tool
- Install `kubectl` by either
  - Using the Azure Command Line
    - `az aks install-cli`
  - Download compiled binary
    - https://kubernetes.io/docs/tasks/tools/install-kubectl/
- Set up `kubectl` to use AKS cluster credentials
  - `az aks get-credentials --resource-group myResourceGroup --name myAKSCluster`

Interacting with Your AKS Cluster

- Now that `kubectl` is installed and configured, let's check on the status of the AKS cluster
  - **`kubectl get nodes`**
  - This displays nodes in all node pools along with status

```
c:\code\git
λ kubectl get nodes
NAME                                STATUS   ROLES   AGE   VERSION
aks-agentpool-51725664-vmss000000   Ready    agent   11h   v1.18.2
aks-agentpool-51725664-vmss000001   Ready    agent   11h   v1.18.2
aks-agentpool-51725664-vmss000002   Ready    agent   11h   v1.18.2
```

- Use 'describe' to get detailed information about a node
  - **`kubectl describe node aks-agentpool-51725664-vmss00000`**
- Provides info about labels, events, pods, metrics, etc.

Interacting with Your AKS Cluster - Describe Node I

```
c:\code\git
λ kubectl describe node aks-fabeastpool-15160931-vmss000000
Name:               aks-fabeastpool-15160931-vmss000000
Roles:              agent
Labels:             agentpool=fabeastpool
                    beta.kubernetes.io/arch=amd64
                    beta.kubernetes.io/instance-type=Standard_DS11_v2
                    beta.kubernetes.io/os=linux
                    failure-domain.beta.kubernetes.io/region=eastus
                    failure-domain.beta.kubernetes.io/zone=0
                    kubernetes.azure.com/cluster=MC_global-fab-cluster-rg_fabk-east_eastus
                    kubernetes.azure.com/node-image-version=AKSUbuntu-1604-2020.06.10
                    kubernetes.azure.com/role=agent
                    kubernetes.io/arch=amd64
                    kubernetes.io/hostname=aks-fabeastpool-15160931-vmss000000
                    kubernetes.io/os=linux
                    kubernetes.io/role=agent
                    node-role.kubernetes.io/agent=
                    node.kubernetes.io/instance-type=Standard_DS11_v2
                    storageprofile=managed
                    storagetier=Premium_LRS
                    topology.kubernetes.io/region=eastus
                    topology.kubernetes.io/zone=0
Annotations:        csi.volume.kubernetes.io/nodeid: {"secrets-store.csi.k8s.io":"aks-fabeastpool-15160931-vmss000000"}
                    node.alpha.kubernetes.io/ttl: 0
                    volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp:  Wed, 24 Jun 2020 12:10:24 -0400
Taints:             <none>
Unschedulable:      false
Lease:
  HolderIdentity:  aks-fabeastpool-15160931-vmss000000
  AcquireTime:     <unset>
  RenewTime:       Tue, 07 Jul 2020 10:42:59 -0400
Conditions:
  Type                 Status  LastHeartbeatTime                 LastTransitionTime                Reason                       Message
  ----                 ------  -----------------                 ------------------                ------                       -------
  NetworkUnavailable   False   Wed, 24 Jun 2020 12:11:34 -0400   Wed, 24 Jun 2020 12:11:34 -0400   RouteCreated                 RouteController created a rout
e
  MemoryPressure       False   Tue, 07 Jul 2020 10:38:07 -0400   Wed, 24 Jun 2020 12:10:24 -0400   KubeletHasSufficientMemory   kubelet has sufficient memory
available
  DiskPressure         False   Tue, 07 Jul 2020 10:38:07 -0400   Wed, 24 Jun 2020 12:10:24 -0400   KubeletHasNoDiskPressure     kubelet has no disk pressure
  PIDPressure          False   Tue, 07 Jul 2020 10:38:07 -0400   Wed, 24 Jun 2020 12:10:24 -0400   KubeletHasSufficientPID      kubelet has sufficient PID ava
ilable
  Ready                True    Tue, 07 Jul 2020 10:38:07 -0400   Wed, 24 Jun 2020 12:10:34 -0400   KubeletReady                 kubelet is posting ready statu
s. AppArmor enabled
Addresses:
  Hostname:    aks-fabeastpool-15160931-vmss000000
  InternalIP:  172.17.0.4
Capacity:
  attachable-volumes-azure-disk:  8
  cpu:                            2
  ephemeral-storage:              129901008Ki
  hugepages-1Gi:                  0
  hugepages-2Mi:                  0
  memory:                         14338988Ki
  pods:                           110
```

Interacting with Your AKS Cluster - Describe Node II

```
System Info:
  Machine ID:                 5204c12129ef4b4d882fce20b282369c
  System UUID:                F6F234CD-A2C5-BA46-835B-BA7B08E5B53B
  Boot ID:                    5b43f7fe-2012-4cc0-a9f4-a6b893ba198b
  Kernel Version:             4.15.0-1089-azure
  OS Image:                   Ubuntu 16.04.6 LTS
  Operating System:           linux
  Architecture:               amd64
  Container Runtime Version:  docker://3.0.10+azure
  Kubelet Version:            v1.18.2
  Kube-Proxy Version:         v1.18.2
PodCIDR:                      10.244.1.0/24
PodCIDRs:                     10.244.1.0/24
ProviderID:                   azure:///subscriptions/[redacted]/resourceGroups/[redacted]/providers/M
icrosoft.Compute/virtualMachineScaleSets/aks-fabeastpool-15160931-vmss/virtualMachines/0
Non-terminated Pods:          (28 in total)
  Namespace                   Name                                                       CPU Requests  CPU Limits  Memory Requests  Memory Limits  AGE
  ---------                   ----                                                       ------------  ----------  ---------------  -------------  ---
  chat                        chat-svc-1                                                 10m (0%)      100m (5%)   10Mi (0%)        50Mi (0%)      11d
  chat                        chat-svc-3                                                 10m (0%)      100m (5%)   10Mi (0%)        50Mi (0%)      11d
  chat                        lighthouse-1                                               10m (0%)      100m (5%)   10Mi (0%)        50Mi (0%)      11d
  dapr-monitoring             dapr-prom-release-prometheus-alertmanager-5cdb75d748-xvpsj 0 (0%)        0 (0%)      0 (0%)           0 (0%)         11d
  dapr-monitoring             dapr-prom-release-prometheus-node-exporter-6qfm6           0 (0%)        0 (0%)      0 (0%)           0 (0%)         11d
  dapr                        dapr-localforwarder-5c7b4b48db-n2qlm                       0 (0%)        0 (0%)      0 (0%)           0 (0%)         11d
  dapr                        dapr-placement-84f9cd87b7-n694x                            0 (0%)        0 (0%)      0 (0%)           0 (0%)         11d
  dapr                        dapr-sentry-58c576ff98-zhh9k                               0 (0%)        0 (0%)      0 (0%)           0 (0%)         11d
  default                     csi-secrets-store-provider-azure-4gq7j                     50m (2%)      50m (2%)    100Mi (0%)       100Mi (0%)     11d
  default                     nmi-wtfsn                                                  100m (5%)     200m (10%)  256Mi (2%)       512Mi (4%)     11d
  fluxcd                      memcached-5bd7849b84-nvwx6                                 0 (0%)        0 (0%)      0 (0%)           0 (0%)         12d
  hipster                     adservice-c5cf7fd66-4nlqh                                  200m (10%)    300m (15%)  180Mi (1%)       300Mi (2%)     12d
  hipster                     cartservice-687d457bc8-b9xf2                               200m (10%)    300m (15%)  64Mi (0%)        128Mi (1%)     12d
  hipster                     frontend-5cb77cdcc-h7hlr                                   100m (5%)     200m (10%)  64Mi (0%)        128Mi (1%)     12d
  hipster                     loadgenerator-54fd7cf49c-m6qdg                             300m (15%)    500m (26%)  256Mi (2%)       512Mi (4%)     12d
  hipster                     shippingservice-74b47cd797-vml6d                           100m (5%)     200m (10%)  64Mi (0%)        128Mi (1%)     12d
  ingress-kong                kong-ingress-kong-6fff9b4699-ccsh7                         0 (0%)        0 (0%)      0 (0%)           0 (0%)         12d
  kube-system                 kube-proxy-4j4dg                                           100m (5%)     0 (0%)      0 (0%)           0 (0%)         12d
  kube-system                 kubernetes-dashboard-55bf89b759-gsdxt                      100m (5%)     100m (5%)   50Mi (0%)        500Mi (4%)     12d
  kubevious                   kubevious-ui-584bc6dddc-7g69l                              25m (1%)      100m (5%)   100Mi (0%)       50Mi (0%)      11d
  linkerd                     linkerd-identity-7bcdfd7dff-6nwpt                          10m (0%)      100m (5%)   10Mi (0%)        50Mi (0%)      12d
  linkerd                     linkerd-proxy-injector-f4b98f4f4-98pj9                     10m (0%)      100m (5%)   10Mi (0%)        50Mi (0%)      12d
  minecraft                   bds-7cdb98b5b5-bc648                                       10m (0%)      100m (5%)   10Mi (0%)        50Mi (0%)      6d17h
  secretpod                   secretpod-deploy-5dfdb7b68d-9kq9d                          100m (5%)     200m (10%)  64Mi (0%)        128Mi (1%)     8d
  secretpod                   secretpod-deploy-green-6cf5554455-v45j8                    100m (5%)     200m (10%)  64Mi (0%)        128Mi (1%)     8d
  secretstore                 csi-secrets-store-secrets-store-csi-driver-kmnjf           0 (0%)        0 (0%)      0 (0%)           0 (0%)         11d
  weave                       weave-scope-agent-6rp4w                                    100m (5%)     0 (0%)      100Mi (0%)       0 (0%)         11d
  weave                       weave-scope-app-86f5f56988-2lrj2                           0 (0%)        0 (0%)      0 (0%)           0 (0%)         11d
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource                    Requests      Limits
  --------                    --------      ------
  cpu                         1635m (86%)   2950m (155%)
  memory                      1422Mi (13%)  2914Mi (26%)
  ephemeral-storage           0 (0%)        0 (0%)
  hugepages-1Gi               0 (0%)        0 (0%)
  hugepages-2Mi               0 (0%)        0 (0%)
  attachable-volumes-azure-disk  0          0
```

Interacting with Your AKS Cluster

- Get basic info about your AKS cluster
  - **`kubectl cluster-info`**

```
c:\code\git
λ kubectl cluster-info
Kubernetes master is running at https://aks-demo-dns-c1b9cbd3.hcp.eastus.azmk8s.io:443
CoreDNS is running at https://aks-demo-dns-c1b9cbd3.hcp.eastus.azmk8s.io:443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy
Metrics-server is running at https://aks-demo-dns-c1b9cbd3.hcp.eastus.azmk8s.io:443/api/v1/namespaces/kube-system/services/https:metrics-server:/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

- Get detailed information about your AKS cluster
  - **`kubectl cluster-info dump`**
  - Warning – this is a lot of information!

Connecting to Multiple AKS Clusters

`kubectl` uses a *context* to specify what cluster to connect to as which user

Context information kept in $HOME/.kube/config

Clusters and Users sections can contain certificate data

Contexts section defines the association of context and user instances

```
apiVersion: v1
kind: Config
current-context: some-dev-cluster
clusters:
- cluster:
    certificate-authority-data: …
    server: <cluster-url>
  name: dev-cluster
users:
- name: some-dev
  user:
    client-certificate-data: …
    client-key-data: …
    token: …
contexts:
- context:
    cluster: dev-cluster
    user: some-dev
  name: some-dev-cluster
```

Connecting to Multiple AKS Clusters

- List defined contexts
  - `kubectl config get-contexts`

```
c:\code\git
λ kubectl config get-contexts
CURRENT   NAME                        CLUSTER     AUTHINFO                            NAMESPACE
          aks-demo                    aks-demo    clusterUser_aks-demo-rg_aks-demo
          fabk-east-adelev-hipster    fabk-east   adelev-hipster
*         fabk-east-admin             fabk-east   fabk-east-admin
          fabk-east-gradya-itgroup    fabk-east   gradya-itgroup
          fabk-west-admin             fabk-west   fabk-west-admin
          minikube                    minikube    minikube
```

- Switch to a different context
  - `kubectl config use-context aks-demo`
- Get nodes from new cluster
  - `kubectl get nodes`

```
c:\code\git
λ kubectl config use-context aks-demo
Switched to context "aks-demo".

c:\code\git
λ kubectl get nodes
NAME                                STATUS   ROLES   AGE   VERSION
aks-agentpool-51725664-vmss000000   Ready    agent   11h   v1.18.2
aks-agentpool-51725664-vmss000001   Ready    agent   11h   v1.18.2
aks-agentpool-51725664-vmss000002   Ready    agent   11h   v1.18.2
```

Microsoft

Essential Kubernetes Objects

Kubernetes Objects

- Resources in a Kubernetes cluster are represented by *objects*
- They are persistent and represent the state of the cluster
- They include
  - Containerized apps deployed to the cluster
  - Resources, such as storage, available to those apps
  - Policies stating how those apps should behave
- You use `kubectl` to tell the Kubernetes API what objects to create
- When applied, these objects represent the cluster's desired state

Kubernetes Objects

- Specification described in a YAML file
- Each object has its own schema
- But each object has common sections and follows similar pattern
- Simple spec is at the right. Required fields are highlighted
  - `apiVersion` is the version of the Kubernetes API to use
  - `kind` is the type of object to create
  - `metadata` contains data to help uniquely identify the object, such as a name and labels (key/value pairs)
  - `spec` is that desired state of that object

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod
  labels:
    run: sample-pod
spec:
  containers:
  - image: nginx
    name: sample-pod-nginx
  restartPolicy: Never
```

Kubernetes Objects - Pods

- Basic building block in Kubernetes
- Represents one or more containers
- Smallest unit of deployment
- Name should be unique across a namespace (more on this later!)
- Labels are key/value pairs
  - Useful for pod selection later
- Spec defines
  - Name of the container (friendly name)
  - Image name of the container (repository name, such as docker hub)
  - Optional info like ports to expose

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod
  labels:
    run: sample-pod
spec:
  containers:
  - image: nginx
    name: sample-pod-nginx
    ports:
    - containerPort: 80
```

Deploying Kubernetes Objects - Create Pod Manifest

Using `kubectl` imperative commands

```
kubectl run sample-pod --image=nginx
--restart=Never
```

Create declarative YAML file and deploy it

```
kubectl run sample-pod --image=nginx
--restart=Never --dry-run -o yaml >
sample-pod.yaml
```

Modify file as necessary

Apply it to AKS cluster

```
kubectl apply -f sample-pod.yaml
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod
  labels:
    run: sample-pod
spec:
  containers:
  - image: nginx
    name: sample-pod-nginx
  restartPolicy: Never
```

Deploying Kubernetes Objects - Check Pod Status

- Check status (regardless of method)
  - `kubectl get pods`
- Default Information Provided
  - NAME = Name of pod
  - READY (H/T) = Number of healthy containers (H) and total number of containers (T)
  - STATUS = health of pod
  - RESTARTS = number of times pod has been restarted
  - AGE = time since pod was last restarted

```
NAME          READY   STATUS    RESTARTS   AGE
sample-pod    1/1     Running   0          12s
```

```
NAME                        READY   STATUS    RESTARTS   AGE
chat-svc-0                  2/2     Running   0          5d21h
chat-svc-1                  2/2     Running   0          5d21h
chat-svc-2                  2/2     Running   0          5d21h
chat-svc-3                  2/2     Running   0          5d21h
chat-ui-788b4c4976-xp2l7    2/2     Running   0          5d21h
lighthouse-0                2/2     Running   0          5d21h
lighthouse-1                2/2     Running   0          5d21h
lighthouse-2                2/2     Running   0          5d21h
mock-svr-5d5dc8f685-vkrr6   2/2     Running   0          5d21h
```

Kubernetes Objects - Pods

- Pod specification includes
  - Commands and args to run
  - ENV variables and secrets
  - Probes
  - Mounted storage
  - Initialization containers
- Specification is then used for controller objects, such as
  - Deployments
  - DaemonSets
  - StatefulSets

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: sample-pod
    team: marketing
  name: sample-pod
spec:
  containers:
  - command: [“/bin/sh”, “-c”, “sleep 3600”]
    env:
    - name: DATA
      value: /var/data
    image: busybox
    name: sample-pod
```

Deploying Kubernetes Objects - Troubleshooting Status

- Sometimes pods have problems
- Status has several issue values
- CrashLoopBackoff means there's a problem with the container and Kubernetes is trying to restart it
  - You'll see restarts increase
  - Sometimes RunContainerError means there's an issue running the container
    - I had a typo in my startup command

```
c:\code\git\aks-training\part2\yaml
λ kubectl get pods -l app=sample-pod
NAME          READY   STATUS             RESTARTS   AGE
sample-pod    0/1     CrashLoopBackOff   1          30s
```

```
c:\code\git\aks-training\part2\yaml
λ kubectl get pods -l app=sample-pod
NAME          READY   STATUS             RESTARTS   AGE
sample-pod    0/1     RunContainerError  6          6m14s
```

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    app: sample-pod
    team: marketing
  name: sample-pod
spec:
  containers:
  - command: ["/bin", "-c", "touch /tmp/healthy; sleep 3600"]
    env:
    - name: DATA
      value: /var/data
    livenessProbe:
      exec:
        command: ["cat", "/tmp/healthy"]
      initialDelaySeconds: 5
      periodSeconds: 5
    image: busybox
    name: sample-pod
```

Deploying Kubernetes Objects - Proactive Measures

- You don’t want to wait for bad statuses to know there’s a problem
- How else can you have Kubernetes monitor your pods for health?
- How do you know if your pod is responding, healthy, alive?

- Probes!

Kubernetes Objects - Probes for Pods

- Pods define probes so that K8s knows pod health
  - Liveness - when to restart a container
  - Readiness - when a pod can accept traffic
  - Startup - when a pod has started
- Probes can be based on commands (“exec”) or http/tcp
- Example on right defines Liveness
  - Use similar spec for Readiness and Startup

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: sample-nginx
    team: marketing
  name: sample-nginx
spec:
  containers:
  - env:
    - name: DATA
      value: /var/data
    image: nginx
    name: sample-nginx
    ports:
    - containerPort: 80
    livenessProbe:
      httpGet:
        path: /
        port: 80
      initialDelaySeconds: 5
      periodSeconds: 5
```

Kubernetes Objects - Storage for Pods

- Container file system lives as long as the container lives
  - Restarting container loses file system changes
- Volumes allow for independent storage for the pod
- Many types of volumes
  - emptyDir
  - hostPath
  - azure (blob and files)
  - Many more

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: sample-store
    team: data-team
  name: sample-store
spec:
  containers:
  - command: ["/bin/sh", "-c", "touch /mnt/data/healthy; sleep 3600"]
    env:
    - name: DATA
      value: /var/data
    livenessProbe:
      exec:
        command: ["cat", "/mnt/data/healthy"]
      initialDelaySeconds: 5
      periodSeconds: 5
    image: busybox
    name: sample-pod
    volumeMounts:
    - name: pod-storage
      mountPath: /mnt/data
  volumes:
    - name: pod-storage
      emptyDir: {}
```

Kubernetes Objects - Storage Volumes

- Volume definition outside of containers
  - Can be applied to multiple containers
- Volumes survive container restarts
- Sharing volumes across containers in a pod allows for
  - Initialization of data by Init Containers
  - Single definition of storage but different mount locations
  - Survivability

Kubernetes Objects - Init Containers and Storage

- InitContainers
  - Runs before “regular” containers
  - Must run to completion
  - Allow you to keep your app containers free of startup utils, etc.
- YAML describes:
  - NGINX web server (“nginx” image)
  - BusyBox container that runs when pod starts (“busybox” image)
    - BusyBox creates a simple web page in “contentdir”
  - Volume defined as an emptyDir
    - Mounted in BusyBox at /work-dir
    - Mounted in NGINX at /usr/share/nginx/html
  - BusyBox first runs, creates HTML file
  - NGINX then runs, serves HTML file
    - LivenessProbe checks the file exists

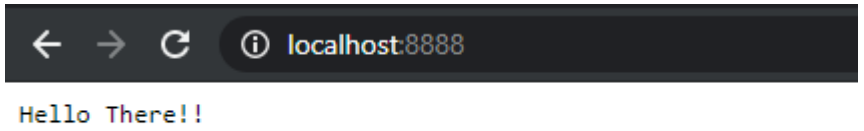
```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: sample-web
    team: web-team
  name: sample-init
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
    - name: contentdir
      mountPath: /usr/share/nginx/html
    livenessProbe:
      httpGet:
        path: /
        port: 80
  initContainers:
  - name: install
    image: busybox
    command: ["/bin/sh", "-c", "echo 'Hello There!!' > /work-dir/index.html"]
    volumeMounts:
    - name: contentdir
      mountPath: /work-dir
  volumes:
  - name: contentdir
    emptyDir: {}
```

Kubernetes Objects - Init Containers and Storage

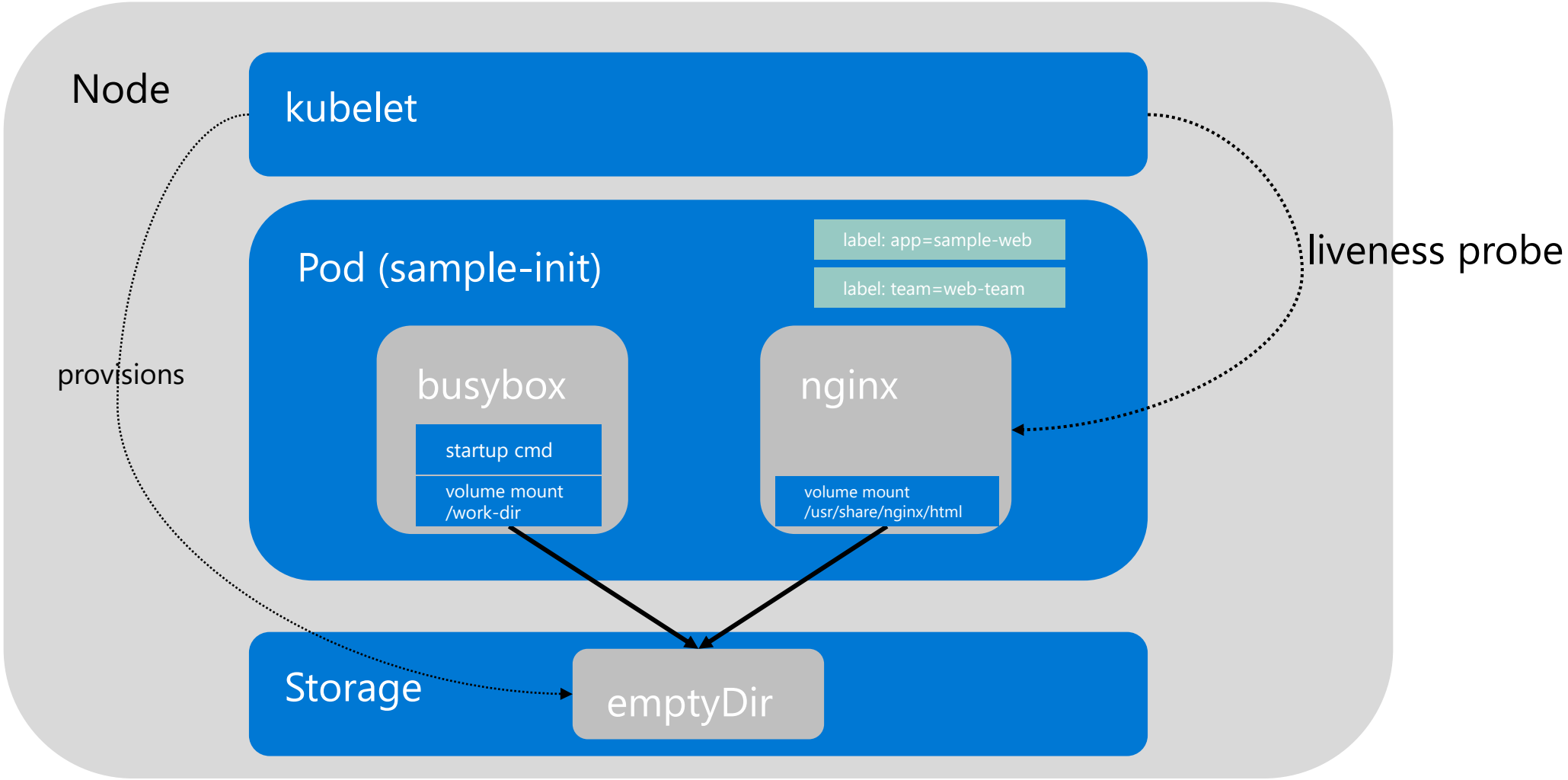
- Deploy the pod YAML
  - `kubectl apply -f sample-init.yaml`
- Watching status, note `Init` status
- Once running, access pod by using
  - `kubectl port-forward sample-init 8888:80`
  - `sample-init` is pod name
  - 8888:80 forwards your request of port 8888 to pod's port 80
- Browse using localhost:8888
- Also note the pod selector
  - `kubectl get pods -l app=sample-web`
  - This retrieves all pods who have a label of app set to the value "sample-web"

```
λ kubectl get pods -l app=sample-web --watch
NAME          READY   STATUS    RESTARTS   AGE
sample-init   0/1     Pending   0          0s
sample-init   0/1     Pending   0          0s
sample-init   0/1     Init:0/1   0          0s
sample-init   0/1     PodInitializing   0          3s
sample-init   1/1     Running           0          4s
```

```
c:\code\git
λ kubectl port-forward sample-init 8888:80
Forwarding from 127.0.0.1:8888 -> 80
Forwarding from [::1]:8888 -> 80
```



Kubernetes Objects - Pods



Next Steps

- We’ve deployed a pod
- Provided some startup commands
- Added environment variables
- Included probes for liveness and readiness
- Defined storage and mounted it
- Created initialization containers
- But what if…
  - We want multiple copies of a pod running?
  - We want to handle how rollouts of new versions of pods is handled?
  - We want to roll back to a previous version?
  - We want to scale the number of pods running (both up and down)?

Microsoft

Kubernetes Deployments

Kubernetes Objects - Deployments

- Describes a *desired state*
- Deployments control ReplicaSets
  - Ensures a specified number of pod replicas are running at any given time
  - Generally you would use Deployments to control these
- Deployment YAML wraps a pod YAML spec
  - Example deploys 3 pods each running an `nginx` container
  - `matchLabels` selects pods who have those labels and values

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: sample-deploy
    team: dev-team
  name: sample-deploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: sample-deploy
  template:
    metadata:
      labels:
        app: sample-deploy
    spec:
      containers:
      - image: nginx
        name: nginx
```

Kubernetes Objects - Deployments

- Retrieve the `sample-deploy` deployment by name

```
c:\code\git
λ kubectl get deploy sample-deploy
NAME            READY   UP-TO-DATE   AVAILABLE   AGE
sample-deploy   3/3     3            3           9m44s
```

- Retrieve all objects with a label of `app` set to the `sample-deploy` value

```
c:\code\git
λ kubectl get all -l app=sample-deploy
NAME                                    READY   STATUS    RESTARTS   AGE
pod/sample-deploy-84599f456f-nkp4p      1/1     Running   0          10m
pod/sample-deploy-84599f456f-r4mcx      1/1     Running   0          10m
pod/sample-deploy-84599f456f-tl64q      1/1     Running   0          10m

NAME                             READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/sample-deploy    3/3     3            3           10m

NAME                                        DESIRED   CURRENT   READY   AGE
replicaset.apps/sample-deploy-84599f456f    3         3         3       10m
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: sample-deploy
    team: dev-team
  name: sample-deploy
spec:
  replicas: 3
  selector:
    matchLabels:
      app: sample-deploy
  template:
    metadata:
      labels:
        app: sample-deploy
    spec:
      containers:
      - image: nginx
        name: nginx
```

Kubernetes Objects - Scaling a Deployment

- Altering number of pods in a deployment is a kubectl command or applying a YAML update
  - `kubectl scale deploy sample-deploy --replicas=4`
- Scale it back down
  - `kubectl scale deploy sample-deploy --replicas=3`
- What if we have to update the image that `sample-deploy` is using?

```
c:\code\git\aks-training\part2\yaml\voting-app-lab
λ kubectl scale deploy sample-deploy --replicas=4
deployment.apps/sample-deploy scaled

c:\code\git\aks-training\part2\yaml\voting-app-lab
λ kubectl get pods -l app=sample-deploy
NAME                              READY   STATUS    RESTARTS   AGE
sample-deploy-c86dd7484-ks6qt     1/1     Running   0          4d20h
sample-deploy-c86dd7484-qgt96     1/1     Running   0          15s
sample-deploy-c86dd7484-sxnmk     1/1     Running   0          4d20h
sample-deploy-c86dd7484-twsml     1/1     Running   0          3m30s

c:\code\git\aks-training\part2\yaml\voting-app-lab
λ kubectl scale deploy sample-deploy --replicas=3
deployment.apps/sample-deploy scaled

c:\code\git\aks-training\part2\yaml\voting-app-lab
λ kubectl get pods -l app=sample-deploy
NAME                              READY   STATUS    RESTARTS   AGE
sample-deploy-c86dd7484-ks6qt     1/1     Running   0          4d20h
sample-deploy-c86dd7484-sxnmk     1/1     Running   0          4d20h
sample-deploy-c86dd7484-twsml     1/1     Running   0          3m44s
```

Kubernetes Objects - Deployment Status

- Note that pods have a hash
- Allows for unique pod name and is a hash of the pod template
- ReplicaSet has similar hash

```
c:\code\git
λ kubectl get all -l app=sample-deploy
NAME                                    READY   STATUS    RESTARTS   AGE
pod/sample-deploy-84599f456f-nkp4p      1/1     Running   0          10m
pod/sample-deploy-84599f456f-r4mcx      1/1     Running   0          10m
pod/sample-deploy-84599f456f-tl64q      1/1     Running   0          10m

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/sample-deploy     3/3     3            3           10m

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/sample-deploy-84599f456f     3         3         3       10m
```

Kubernetes Objects - Deployment Updates

- Update deployment image

  ```
  kubectl set image deploy sample-deploy nginx=nginx:1.16.1 --record
  ```

- We'll see pods terminate and new ones start up
- New ReplicaSet created
- New Pods created, using same hash from new ReplicaSet

```
c:\code\git
λ kubectl set image deploy sample-deploy nginx=nginx:1.16.1 --record
deployment.apps/sample-deploy image updated

c:\code\git
λ kubectl get pods -l app=sample-deploy --watch
NAME                              READY   STATUS              RESTARTS   AGE
sample-deploy-84599f456f-nkp4p    1/1     Running             0          20m
sample-deploy-84599f456f-tl64q    1/1     Running             0          20m
sample-deploy-c86dd7484-sb6vv     0/1     ContainerCreating   0          12s
sample-deploy-c86dd7484-xt9w8     1/1     Running             0          27s
sample-deploy-c86dd7484-sb6vv     1/1     Running             0          15s
sample-deploy-84599f456f-nkp4p    1/1     Terminating         0          20m
sample-deploy-c86dd7484-hvxlc     0/1     Pending             0          0s
sample-deploy-c86dd7484-hvxlc     0/1     Pending             0          0s
sample-deploy-c86dd7484-hvxlc     0/1     ContainerCreating   0          0s
sample-deploy-84599f456f-nkp4p    0/1     Terminating         0          20m
sample-deploy-c86dd7484-hvxlc     1/1     Running             0          2s
sample-deploy-84599f456f-tl64q    1/1     Terminating         0          20m
sample-deploy-84599f456f-nkp4p    0/1     Terminating         0          20m
sample-deploy-84599f456f-nkp4p    0/1     Terminating         0          20m
sample-deploy-84599f456f-tl64q    0/1     Terminating         0          20m
sample-deploy-84599f456f-tl64q    0/1     Terminating         0          20m
sample-deploy-84599f456f-tl64q    0/1     Terminating         0          20m
```

```
c:\code\git
λ kubectl get rs
NAME                        DESIRED   CURRENT   READY   AGE
sample-deploy-84599f456f    0         0         0       21m
sample-deploy-c86dd7484     3         3         3       78s
```

```
c:\code\git
λ kubectl get pods -l app=sample-deploy
NAME                              READY   STATUS    RESTARTS   AGE
sample-deploy-c86dd7484-hvxlc     1/1     Running   0          5m16s
sample-deploy-c86dd7484-sb6vv     1/1     Running   0          5m31s
sample-deploy-c86dd7484-xt9w8     1/1     Running   0          5m46s
```

Kubernetes Objects - Deployment Updates

```
c:\code\git
λ kubectl describe deploy sample-deploy
Name:                   sample-deploy
Namespace:              default
CreationTimestamp:      Wed, 08 Jul 2020 10:40:27 -0400
Labels:                 app=sample-deploy
                        team=dev-team
Annotations:            deployment.kubernetes.io/revision: 2
                        kubernetes.io/change-cause: kubectl set image deploy sample-deploy nginx=nginx:1.16.1 --record=true
Selector:               app=sample-deploy
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=sample-deploy
  Containers:
   nginx:
    Image:        nginx:1.16.1
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   sample-deploy-c86dd7484 (3/3 replicas created)
Events:
  Type    Reason             Age    From                   Message
  ----    ------             ----   ----                   -------
  Normal  ScalingReplicaSet  26m    deployment-controller  Scaled up replica set sample-deploy-84599f456f to 3
  Normal  ScalingReplicaSet  6m46s  deployment-controller  Scaled up replica set sample-deploy-c86dd7484 to 1
  Normal  ScalingReplicaSet  6m31s  deployment-controller  Scaled down replica set sample-deploy-84599f456f to 2
  Normal  ScalingReplicaSet  6m31s  deployment-controller  Scaled up replica set sample-deploy-c86dd7484 to 2
  Normal  ScalingReplicaSet  6m16s  deployment-controller  Scaled down replica set sample-deploy-84599f456f to 1
  Normal  ScalingReplicaSet  6m16s  deployment-controller  Scaled up replica set sample-deploy-c86dd7484 to 3
  Normal  ScalingReplicaSet  6m14s  deployment-controller  Scaled down replica set sample-deploy-84599f456f to 0
```

Kubernetes Objects - Rolling Back a Deployment

```
c:\code\git
λ k rollout history deploy sample-deploy
deployment.apps/sample-deploy
REVISION  CHANGE-CAUSE
1         <none>
2         kubectl set image deploy sample-deploy nginx=nginx:1.16.1 --record=true


c:\code\git
λ kubectl rollout undo deploy sample-deploy
deployment.apps/sample-deploy rolled back
```

```
c:\code\git
λ kubectl get pods -l app=sample-deploy
NAME                              READY   STATUS    RESTARTS   AGE
sample-deploy-84599f456f-hm29b    1/1     Running   0          2m5s
sample-deploy-84599f456f-kvtc6    1/1     Running   0          2m10s
sample-deploy-84599f456f-stwcj    1/1     Running   0          2m8s
```

```
c:\code\git
λ kubectl describe deploy sample-deploy
Name:                   sample-deploy
Namespace:              default
CreationTimestamp:      Wed, 08 Jul 2020 10:40:27 -0400
Labels:                 app=sample-deploy
                        team=dev-team
Annotations:            deployment.kubernetes.io/revision: 3
Selector:               app=sample-deploy
Replicas:               3 desired | 3 updated | 3 total | 3 available | 0 unavailable
StrategyType:           RollingUpdate
MinReadySeconds:        0
RollingUpdateStrategy:  25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=sample-deploy
  Containers:
   nginx:
    Image:        nginx
    Port:         <none>
    Host Port:    <none>
    Environment:  <none>
    Mounts:       <none>
  Volumes:        <none>
Conditions:
  Type           Status  Reason
  ----           ------  ------
  Available      True    MinimumReplicasAvailable
  Progressing    True    NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   sample-deploy-84599f456f (3/3 replicas created)
Events:
  Type    Reason             Age                From                   Message
  ----    ------             ----               ----                   -------
  Normal  ScalingReplicaSet  31m                deployment-controller  Scaled up replica set sample-deploy-84599f456f to 3
  Normal  ScalingReplicaSet  11m                deployment-controller  Scaled up replica set sample-deploy-c86dd7484 to 1
  Normal  ScalingReplicaSet  10m                deployment-controller  Scaled down replica set sample-deploy-84599f456f to 2
  Normal  ScalingReplicaSet  10m                deployment-controller  Scaled up replica set sample-deploy-c86dd7484 to 2
  Normal  ScalingReplicaSet  10m                deployment-controller  Scaled down replica set sample-deploy-84599f456f to 1
  Normal  ScalingReplicaSet  10m                deployment-controller  Scaled up replica set sample-deploy-c86dd7484 to 3
  Normal  ScalingReplicaSet  10m                deployment-controller  Scaled down replica set sample-deploy-84599f456f to 0
  Normal  ScalingReplicaSet  35s                deployment-controller  Scaled up replica set sample-deploy-84599f456f to 1
  Normal  ScalingReplicaSet  33s                deployment-controller  Scaled down replica set sample-deploy-c86dd7484 to 2
  Normal  ScalingReplicaSet  33s                deployment-controller  Scaled up replica set sample-deploy-84599f456f to 2
  Normal  ScalingReplicaSet  28s (x3 over 30s)  deployment-controller  (combined from similar events): Scaled down replica set sample-deploy
-c86dd7484 to 0
```

Next Steps…

- OK, we can
  - Deploy pods
  - Have environment variables, probes, storage
  - Deploy multiple copies of pods
  - Track deployments
  - Roll back deployments

- But how can we…
  - Expose pods inside or outside the cluster?
  - Have pods called without having to use `port-forward`?

Microsoft

Kubernetes Services

Kubernetes Objects - Services

- Each pod gets its own IP
- IPs can be allocated dynamically
- What happens when pod restarts and new IP assigned?
  - How can client apps connect reliably?
  - In deploy update below, pod IPs changed during image update

```
c:\code\git
λ kubectl get pods -l app=sample-deploy -o wide
NAME                              READY   STATUS    RESTARTS   AGE     IP            NODE                                NOMINATED NODE   READINESS GATES
sample-deploy-84599f456f-hm29b    1/1     Running   0          8m55s   10.244.1.13   aks-agentpool-51725664-vmss000000   <none>           <none>
sample-deploy-84599f456f-kvtc6    1/1     Running   0          9m      10.244.1.12   aks-agentpool-51725664-vmss000000   <none>           <none>
sample-deploy-84599f456f-stwcj    1/1     Running   0          8m58s   10.244.2.13   aks-agentpool-51725664-vmss000001   <none>           <none>
```

```
c:\code\git
λ kubectl get pods -l app=sample-deploy -o wide
NAME                              READY   STATUS    RESTARTS   AGE   IP            NODE                                NOMINATED NODE   READINESS GATES
sample-deploy-c86dd7484-7h67b     1/1     Running   0          20s   10.244.0.10   aks-agentpool-51725664-vmss000002   <none>           <none>
sample-deploy-c86dd7484-ks6qt     1/1     Running   0          18s   10.244.2.14   aks-agentpool-51725664-vmss000001   <none>           <none>
sample-deploy-c86dd7484-sxnmk     1/1     Running   0          22s   10.244.0.9    aks-agentpool-51725664-vmss000002   <none>           <none>
```

- Can a Kubernetes object be created to abstract the backend pods?

Kubernetes Objects - Services

- Create a service imperatively or via YAML
  - `kubectl expose deploy sample-deploy --name sample-deploy-svc --port=80 --type=LoadBalancer`
- `type` can be one these values
  - `LoadBalancer` - provisions a public IP with Azure (cloud provider)
  - `ClusterIP` (default) - available only within cluster
  - `NodePort` - exposes pods via port on the K8s node
  - `External` - exposes an external service to cluster (e.g. Azure Cosmos DB)

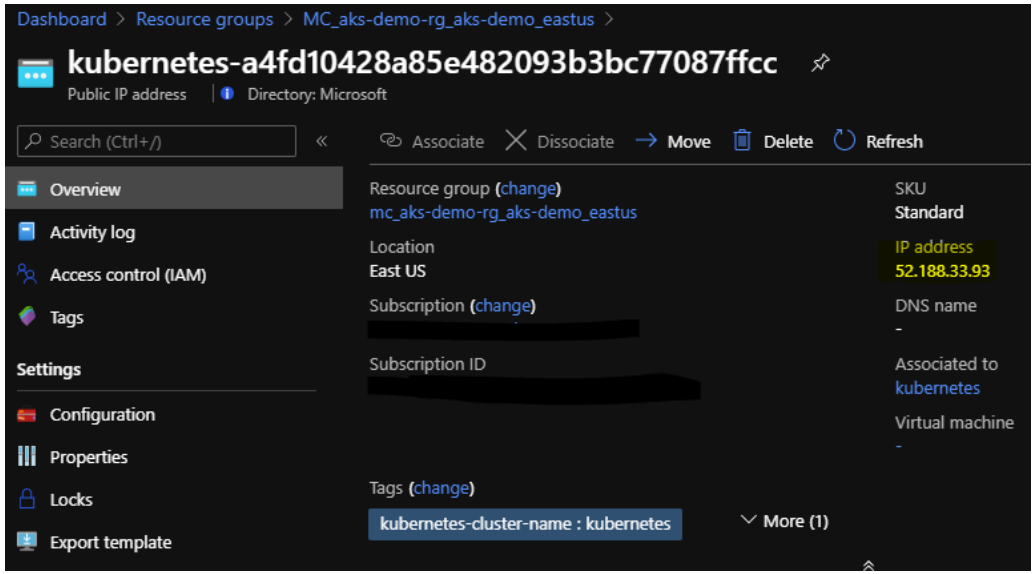
```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: sample-deploy
    team: dev-team
  name: sample-deploy-svc
spec:
  type: LoadBalancer
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: sample-deploy
```

Kubernetes Objects - Services

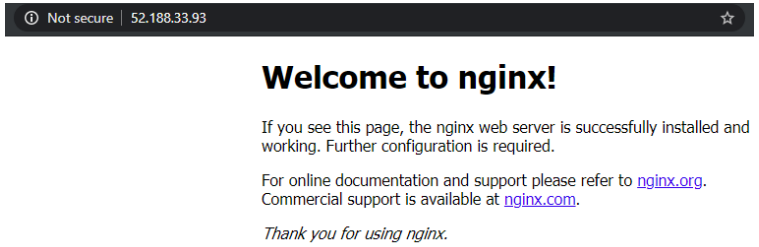
- View service information

```
c:\code\git
λ kubectl get svc -l app=sample-deploy
NAME                 TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)        AGE
sample-deploy-svc    LoadBalancer   10.0.199.211   52.188.33.93   80:30432/TCP   2m44s
```

- Public IP provisioned in Azure



- Navigate to LoadBalancer exposed service



Kubernetes Objects - Services

- The sample-deploy Deployment defines label for pods of app=sample-deploy
- Service sample-deploy-svc abstracts pods with those labels of app=sample-deploy
- Service creates endpoints that map to pod IPs

```
c:\code\git
λ kubectl get endpoints sample-deploy-svc
NAME                ENDPOINTS                                       AGE
sample-deploy-svc   10.244.0.10:80,10.244.0.9:80,10.244.2.14:80   19m

c:\code\git
λ kubectl get pods -l app=sample-deploy -o wide
NAME                              READY   STATUS    RESTARTS   AGE   IP
sample-deploy-c86dd7484-7h67b     1/1     Running   0          26m   10.244.0.10
sample-deploy-c86dd7484-ks6qt     1/1     Running   0          26m   10.244.2.14
sample-deploy-c86dd7484-sxnmk     1/1     Running   0          26m   10.244.0.9
```

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: sample-deploy
    team: dev-team
  name: sample-deploy-svc
spec:
  type: LoadBalancer
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: sample-deploy
```

Kubernetes Objects - Services

- Do I need to expose my services using LoadBalancer if I want external access?
  - I could use a lot of public IP address
- No - more to come next week on this!!!
  - Hint -- Ingress

Microsoft

Reusing and Grouping Kubernetes Objects

Reusing Kubernetes Objects

- In previous examples, we saw defined in individual YAML
  - Environment Variables
  - Secrets
  - Storage
- What if we had to provide these to multiple objects
- Can we create them separately and then apply to many objects?
- You betcha!

Kubernetes Objects – ConfigMaps

- ConfigMaps allow you to define configuration separately from applications
- Can specify name/value pairs
- Can also specify file-like configuration

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: sample-config
data:
  data-vol: “/data”
  max-count: “100”
  text.conf: |
    text.color=blue
    font.size=12
```

Kubernetes Objects - ConfigMaps

- Retrieve ConfigMap
  - `kubectl get cm sample-config`
  - Retrieves basic info about the ConfigMap
- Describe ConfigMap
  - `kubectl describe cm sample-config`
  - Retrieves values of the ConfigMap
  - Notice “text.conf”, which was a multi-value config item

- How do I use these?

```
c:\code\git
λ kubectl get cm sample-config
NAME            DATA   AGE
sample-config   3      90s

c:\code\git
λ kubectl describe cm sample-config
Name:         sample-config
Namespace:    default
Labels:       <none>
Annotations:
Data
====
data-vol:
----
“/data”
max-count:
----
“100”
text.conf:
----
text.color=blue
font.size=12
```

Kubernetes Objects - Using ConfigMaps in Pods

- Map an ENV variable to ConfigMap item

```
c:\code\git
λ kubectl logs sample-pod-config
KUBERNETES_SERVICE_PORT=443
KUBERNETES_PORT=tcp://10.0.0.1:443
HOSTNAME=sample-pod-config
SAMPLE_DEPLOY_SVC_SERVICE_PORT=80
SAMPLE_DEPLOY_SVC_PORT=tcp://10.0.199.211:80
SHLVL=1
HOME=/root
SAMPLE_DEPLOY_SVC_PORT_80_TCP_ADDR=10.0.199.211
DATA_DIR="/data"
SAMPLE_DEPLOY_SVC_PORT_80_TCP_PORT=80
SAMPLE_DEPLOY_SVC_PORT_80_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP_ADDR=10.0.0.1
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
KUBERNETES_PORT_443_TCP_PORT=443
SAMPLE_DEPLOY_SVC_PORT_80_TCP=tcp://10.0.199.211:80
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP=tcp://10.0.0.1:443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_SERVICE_HOST=10.0.0.1
PWD=/
SAMPLE_DEPLOY_SVC_SERVICE_HOST=10.0.199.211
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod-config
spec:
  containers:
    - name: main
      image: busybox
      command: [ "/bin/sh", "-c", "env; sleep 3600" ]
      env:
        - name: DATA_DIR
          valueFrom:
            configMapKeyRef:
              name: sample-config
              key: data-vol
```

Kubernetes Objects - Using ConfigMaps in Pods

- Map all ConfigMap items to ENV variables

```
c:\code\git
λ kubectl logs sample-pod-config2
KUBERNETES_PORT=tcp://10.0.0.1:443
KUBERNETES_SERVICE_PORT=443
HOSTNAME=sample-pod-config2
SAMPLE_DEPLOY_SVC_SERVICE_PORT=80
SAMPLE_DEPLOY_SVC_PORT=tcp://10.0.199.211:80
SHLVL=1
HOME=/root
text.conf=text.color=blue
font.size=12

SAMPLE_DEPLOY_SVC_PORT_80_TCP_ADDR=10.0.199.211
data-vol="/data"
SAMPLE_DEPLOY_SVC_PORT_80_TCP_PORT=80
SAMPLE_DEPLOY_SVC_PORT_80_TCP_PROTO=tcp
max-count="100"
KUBERNETES_PORT_443_TCP_ADDR=10.0.0.1
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
SAMPLE_DEPLOY_SVC_PORT_80_TCP=tcp://10.0.199.211:80
KUBERNETES_PORT_443_TCP_PORT=443
KUBERNETES_PORT_443_TCP_PROTO=tcp
KUBERNETES_PORT_443_TCP=tcp://10.0.0.1:443
KUBERNETES_SERVICE_PORT_HTTPS=443
KUBERNETES_SERVICE_HOST=10.0.0.1
PWD=/
SAMPLE_DEPLOY_SVC_SERVICE_HOST=10.0.199.211
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod-config2
spec:
  containers:
    - name: main
      image: busybox
      command: [ "/bin/sh", "-c", "env; sleep 3600" ]
      envFrom:
        configMapRef:
          name: sample-config
```

Kubernetes Objects - Using ConfigMaps in Pods

- Map ConfigMap items to volumes
  - Can be more secure than just using ENV vars
- Each config item is a file in the volume

```
c:\code\git
λ kubectl logs sample-pod-config3
text.color=blue
font.size=12

c:\code\git
λ kubectl exec sample-pod-config3 -it -- ls -l /mnt/env
total 0
lrwxrwxrwx    1 root     root           15 Jul  8 19:34 data-vol -> ..data/data-vol
lrwxrwxrwx    1 root     root           16 Jul  8 19:34 max-count -> ..data/max-count
lrwxrwxrwx    1 root     root           16 Jul  8 19:34 text.conf -> ..data/text.conf
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod-config3
spec:
  containers:
    - name: main
      image: busybox
      command: [ "/bin/sh", "-c", "cat
/mnt/env/text.conf; sleep 3600" ]
      volumeMounts:
      - name: config-vol
        mountPath: /mnt/env
  volumes:
  - name: config-vol
    configMap:
      name: sample-config
```

Handling Sensitive Values

- Mapping ConfigMap values to ENV or volumes is great
- BUT…
  - What about more sensitive values?
- Kubernetes also has an *object* called a Secret
  - `kubectl create secret generic sample-secret --from-literal=favoriteFood=hot-pockets`
- I can use this to create a YAML file, too

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: sample-secret
data:
  favoriteFood: aG90LXBvY2tldHM=
```

Kubernetes Objects - Referencing Secrets in Pods

- Very similar to ConfigMaps
  - Set certain secrets to ENV Vars using `valueFrom` and `secretKeyRef`
  - Set them to ENV Vars using `envFrom` and `secretRef`
  - Set them in `volumeMounts` using the `secret` type of volume
- Then reference them in your app as you normally would

Kubernetes Objects - Referencing Secrets in Pods

Individual ENV Var

```
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod-secret1
spec:
  containers:
    - name: main
      image: busybox
      command: [ "/bin/sh", "-c", "env;
sleep 3600" ]
      env:
      - name: FAVORITE_FOOD
        valueFrom:
          secretKeyRef:
            name: sample-secret
            key: favoriteFood
```

All ENV Var

```
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod-secret2
spec:
  containers:
    - name: main
      image: busybox
      command: [ "/bin/sh", "-c",
"env; sleep 3600" ]
      envFrom:
      - secretRef:
          name: sample-secret
```

Mount as Volume

```
apiVersion: v1
kind: Pod
metadata:
  name: sample-pod-secret3
spec:
  containers:
    - name: main
      image: busybox
      command: [ "/bin/sh", "-c",
"ls -l /mnt/secrets; sleep 3600" ]
      volumeMounts:
      - name: secret-vol
        mountPath: /mnt/secrets
  volumes:
    - name: secret-vol
      secret:
        secretName: sample-secret
```

Kubernetes Objects – Advanced Storage (PV and PVC)

- Up to now, Storage defined alongside Pods in Deployments
- What about shared storage?
  - Defined outside Deployments
  - Used across multiple Deployments
- `PersistentVolume` and `PersistentVolumeClaims` help with this
  - `PersistentVolume` (PV) is provisioned storage in the cluster and have a lifecycle outside the Pod
  - `PersistentVolumeClaim` (PVC) is a request for storage and cluster process will attempt to match a PVC to a PV. A PVC is used as a volume in a Pod.
  - Match is based on attributes such as type of storage, access mode, and request size
- Provisioning of storage can either be Static or Dynamic
  - **Static** matches a PVC with pre-existing PVs. PV is created by an admin ahead of time
  - **Dynamic** creates a PV based on the PVC when no existing PVs satisfy the PVC

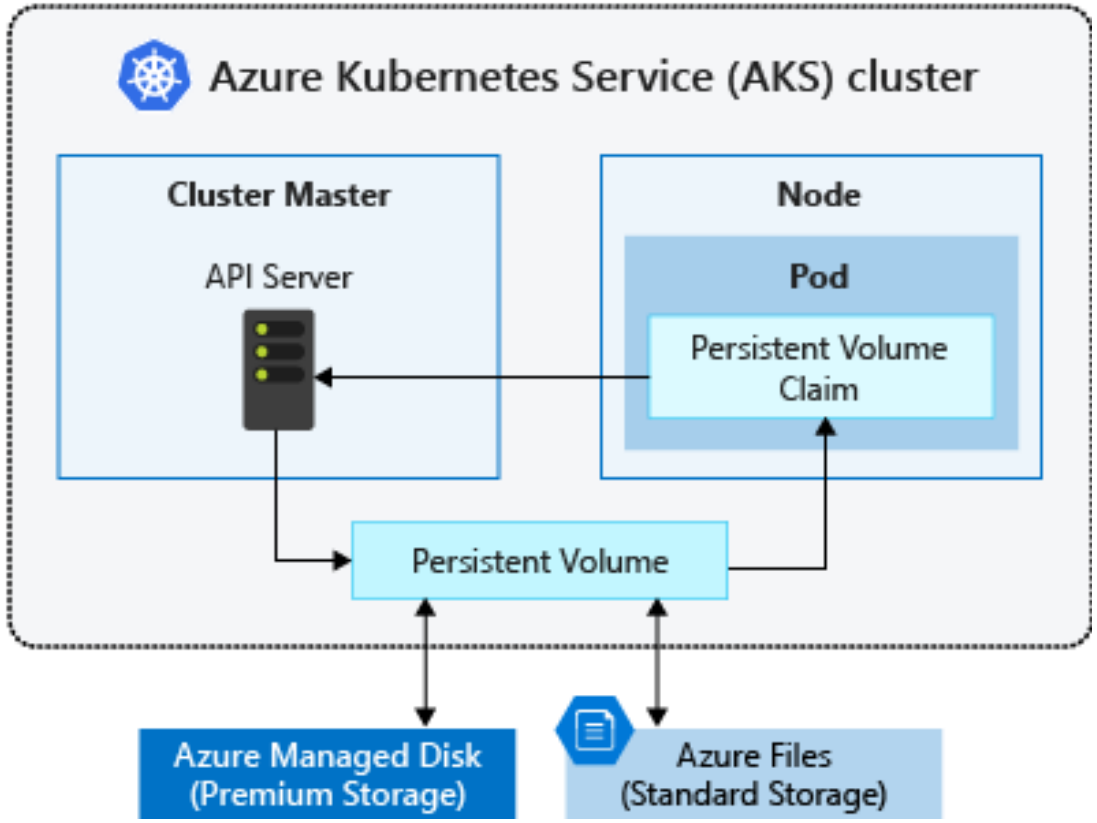


Image Source: https://docs.microsoft.com/en-us/azure/aks/media/concepts-storage/aks-storage-options.png

Kubernetes Objects - Create a Dynamic PV in Azure

- The PV will be created dynamically
- The following happens
  - Storage account created

fe67d129826e249ba9c3850	Storage account	East US

  - Secret created with storage name and key

```
λ kubectl describe secret azure-storage-account-fe67d129826e249ba9c3850-secret
Name:         azure-storage-account-fe67d129826e249ba9c3850-secret
Namespace:    default
Labels:       <none>
Annotations:  <none>

Type:  Opaque

Data
====
azurestorageaccountkey:   88 bytes
azurestorageaccountname:  23 bytes
```

  - PV created

```
c:\code\git
λ kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                 STORAGECLASS   REASON   AGE
pvc-dfd0a814-0402-4146-bf01-264aa7dabf61   5Gi        RWX            Delete           Bound    default/sample-pvc    azurefile              6m59s
```

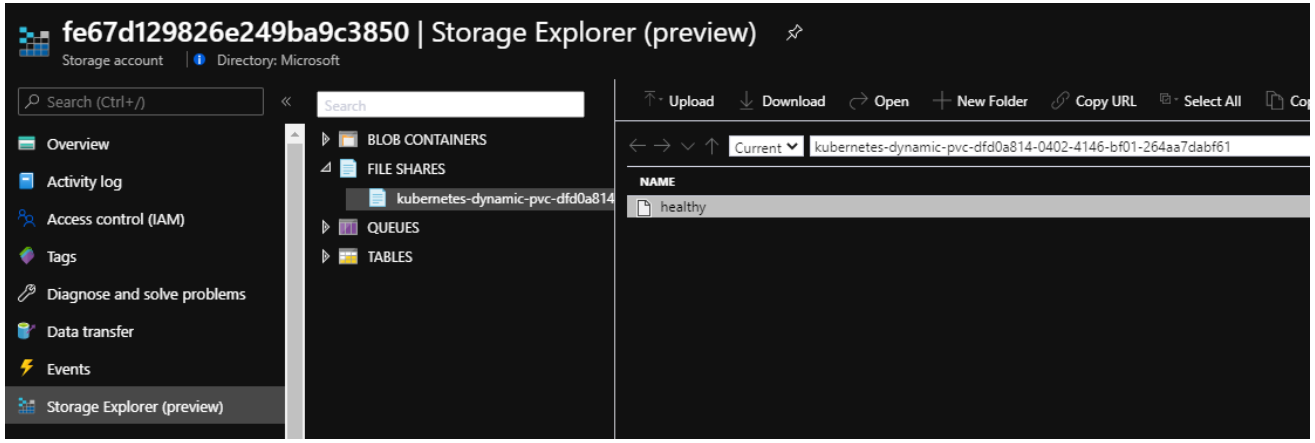
  - PVC created

```
c:\code\git
λ kubectl get pvc
NAME         STATUS   VOLUME                                     CAPACITY   ACCESS MODES   STORAGECLASS   AGE
sample-pvc   Bound    pvc-dfd0a814-0402-4146-bf01-264aa7dabf61   5Gi        RWX            azurefile      8m8s
```

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: sample-pvc
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: azurefile
  resources:
    requests:
      storage: 5Gi
```

Kubernetes Objects - Use a PVC in a Pod (Azure Files)

- Using the volume example from earlier, but use the Azure Files PV/PVC
- Only change is in `volumes`, where claim is associated
- The ‘healthy’ file shows up




```yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    app: sample-store-azurefiles
    team: azure-team
  name: sample-store-azurefiles
spec:
  containers:
  - command: ["/bin/sh", "-c", "touch /mnt/data/healthy; sleep 3600"]
    livenessProbe:
      exec:
        command: ["cat", "/mnt/data/healthy"]
      initialDelaySeconds: 5
      periodSeconds: 5
    image: busybox
    name: sample-pod
    volumeMounts:
    - name: pod-azure-storage
      mountPath: /mnt/data
  volumes:
    - name: pod-azure-storage
      persistentVolumeClaim:
        claimName: sample-pvc
```

Kubernetes Objects – Namespaces

- Namespaces allow you to create virtual clusters within your physical cluster
- Provides a scope for names of resources
  - Names of resources need to be unique within a namespace
- Not all Kubernetes objects are namespaced
  - `kubectl api-resources` lists Kubernetes resources and if they can be namespaced
- Why use namespaces?
  - Environment has multiple users spread across multiple teams and/or projects
  - Divide cluster resources between multiple users via resource quotas
  - You need logical division between different groups of resources (e.g. applications)
- Namespaces also adjust how you refer to objects
  - Service goes from “my-service” to “my-service.my-ns” as a URL

Kubernetes Objects - Using Namespaces

- Previous Deployment example, selected pods using label selector
- May want to group pods, deployment, and service together
  - Use a namespace!

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: sample
```

  - Associate the namespace with deployment and service

```
c:\code\git
λ kubectl get all -n sample
NAME                                    READY   STATUS    RESTARTS   AGE
pod/sample-deploy-84599f456f-6qqwg      1/1     Running   0          4m14s
pod/sample-deploy-84599f456f-7gcvb      1/1     Running   0          4m14s
pod/sample-deploy-84599f456f-j28fn      1/1     Running   0          4m14s

NAME                        TYPE           CLUSTER-IP   EXTERNAL-IP    PORT(S)        AGE
service/sample-deploy-svc   LoadBalancer   10.0.58.54   52.226.97.69   80:31619/TCP   4m4s

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/sample-deploy     3/3     3            3           4m14s

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/sample-deploy-84599f456f     3         3         3       4m14s
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: sample-deploy
    team: dev-team
  name: sample-deploy
  namespace: sample
spec:
  replicas: 3
  selector:
    matchLabels:
      app: sample-deploy
  template:
    metadata:
      labels:
        app: sample-deploy
    spec:
      containers:
      - image: nginx
        name: nginx
```

```yaml
apiVersion: v1
kind: Service
metadata:
  labels:
    app: sample-deploy
    team: dev-team
  name: sample-deploy-svc
  namespace: sample
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: sample-deploy
  type: LoadBalancer
```

Kubernetes Objects - Using Namespaces

- Only view resources in specific namespaces passing (-n <ns-name>) to `kubectl`
  - `kubectl get all -n sample`
- View all resources across all namespaces passing `--all-namespaces` flag to `kubectl`
  - `kubectl get pods --all-namespaces`
- Not passing a namespace flag pulls resources only in the `default` namespace
  - `kubectl get pods` (returns only pods in the default namespace)

Kubernetes Objects – Working with Namespaces

	Imperative Style	Declarative Style

Create a namespace	`kubectl create ns test`	`apiVersion: v1`
		`kind: Namespace`
		`metadata:`
		`  name: test`

List all namespaces	`kubectl get ns`	n/a

Deploy a Pod to the `test` namespace	`kubectl run my-pod --image=nginx –restart=Never -n test`	`apiVersion: v1`
		`kind: Pod`
		`metadata:`
		`  labels:`
		`  run: my-pod`
		`  name: my-pod`
		`  namespace: test`
		`spec:`
		`  containers:`
		`  - image: nginx`
		`  name: my-pod`
		`  restartPolicy: Never`

Kubernetes Objects - Working with Namespaces

- Resources provisioned in a namespace are only available to other resources in that namespace
  - ConfigMaps and Secrets available to resources only in the same namespace
  - PVCs also scoped to a namespace
- Some resources are not scoped to a namespace and are available across all, for example
  - PersistentVolumes
  - Nodes
- Check `kubectl api-resources` for which resources are namespace scoped

Microsoft

Bringing It Together

Bringing Things Together - Voting App

- Create the namespace

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: voting
```

- Create back end deploy and place it in voting ns

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: azure-vote-back
  namespace: voting
spec:
  replicas: 1
  selector:
    matchLabels:
      app: azure-vote-back
  template:
    metadata:
      labels:
        app: azure-vote-back
    spec:
      containers:
      - name: azure-vote-back
        image: redis
        ports:
        - containerPort: 6379
          name: redis
```

- Create back end service and add it to voting ns

```yaml
apiVersion: v1
kind: Service
metadata:
  name: azure-vote-back
  namespace: voting
spec:
  ports:
  - port: 6379
  selector:
    app: azure-vote-back
  type: ClusterIP  #default value
```

Bringing Things Together - Voting App

- Create the front end deployment and service, both in the voting ns

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: azure-vote-front
  namespace: voting
spec:
  replicas: 1
  selector:
    matchLabels:
      app: azure-vote-front
  template:
    metadata:
      labels:
        app: azure-vote-front
    spec:
      containers:
      - name: azure-vote-front
        image: microsoft/azure-vote-front:v1
        ports:
        - containerPort: 80
        env:
        - name: REDIS
          value: "azure-vote-back.voting"
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: azure-vote-front
  namespace: voting
spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: azure-vote-front
```

Note that the location of the Redis service has the namespace suffix. If this was missing, service would not be found!

Bringing Things Together - Voting App

- Check on status via `kubectl`

```
c:\code\git
λ kubectl get all -n voting
NAME                                    READY   STATUS    RESTARTS   AGE
pod/azure-vote-back-7d6d77f4ff-659h2    1/1     Running   0          17m
pod/azure-vote-front-65cdfc98d6-4rbs7   1/1     Running   0          17m

NAME                       TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)        AGE
service/azure-vote-back    ClusterIP      10.0.252.176   <none>          6379/TCP       17m
service/azure-vote-front   LoadBalancer   10.0.215.9     52.186.43.166   80:30556/TCP   17m

NAME                               READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/azure-vote-back    1/1     1            1           17m
deployment.apps/azure-vote-front   1/1     1            1           17m

NAME                                          DESIRED   CURRENT   READY   AGE
replicaset.apps/azure-vote-back-7d6d77f4ff    1         1         1       17m
replicaset.apps/azure-vote-front-65cdfc98d6   1         1         1       17m
```

Bringing Things Together - Voting App

- Scale up the front-end via kubectl

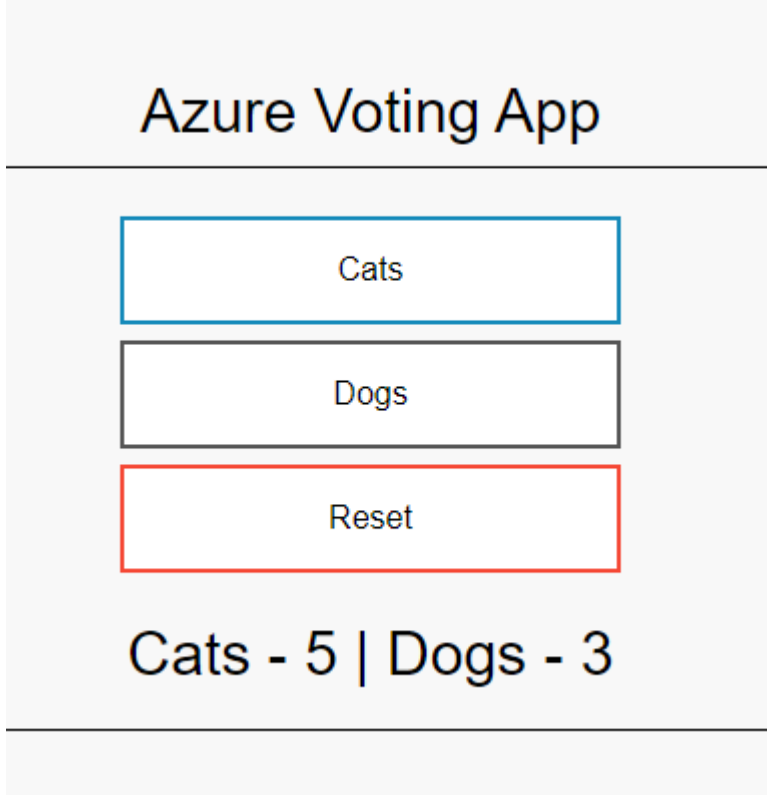
```
c:\code\git
λ kubectl scale deploy azure-vote-front --replicas=3 -n voting
deployment.apps/azure-vote-front scaled
```

```
c:\code\git
λ kubectl get all -n voting
NAME                                     READY   STATUS    RESTARTS   AGE
pod/azure-vote-back-7d6d77f4ff-659h2     1/1     Running   0          3h4m
pod/azure-vote-front-65cdfc98d6-52qr6    1/1     Running   0          2m
pod/azure-vote-front-65cdfc98d6-khvz6    1/1     Running   0          2m
pod/azure-vote-front-65cdfc98d6-r2nx9    1/1     Running   0          3m17s

NAME                        TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)        AGE
service/azure-vote-back     ClusterIP      10.0.252.176   <none>          6379/TCP       3h4m
service/azure-vote-front    LoadBalancer   10.0.215.9     52.186.43.166   80:30556/TCP   3h4m

NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/azure-vote-back      1/1     1            1           3h4m
deployment.apps/azure-vote-front     3/3     3            3           3h4m

NAME                                            DESIRED   CURRENT   READY   AGE
replicaset.apps/azure-vote-back-7d6d77f4ff      1         1         1       3h4m
replicaset.apps/azure-vote-front-65cdfc98d6     3         3         3       3h4m
```



Bringing Things Together - Voting App

- Update the front end image to v2
  - Can do via YAML (`kubectl apply -f sample-voting-app.yaml`)
  - Can also use `kubectl`

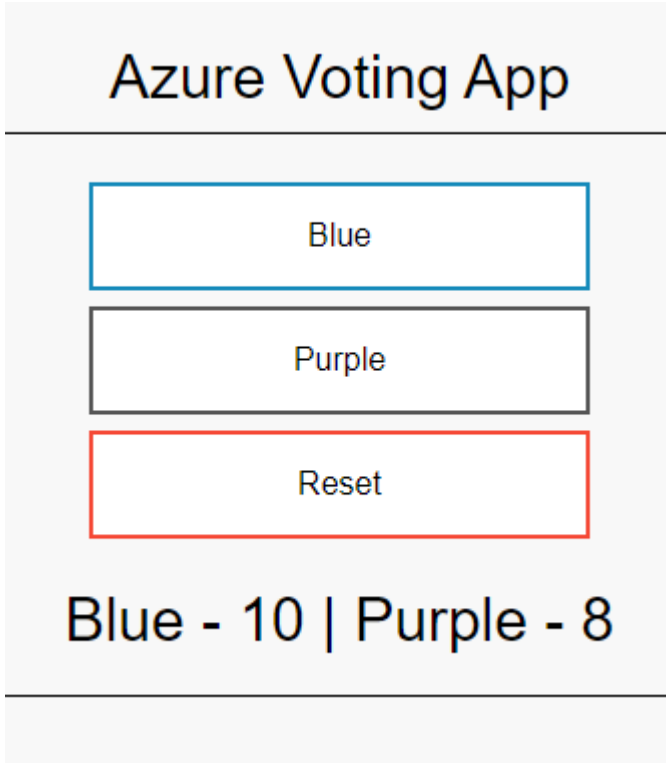
```
c:\code\git
λ kubectl set image deploy azure-vote-front azure-vote-front=microsoft/azure-vote-front:v2 -n voting --record
deployment.apps/azure-vote-front image updated
```

```
c:\code\git
λ kubectl get all -n voting
NAME                                      READY   STATUS        RESTARTS   AGE
pod/azure-vote-back-7d6d77f4ff-659h2      1/1     Running       0          3h18m
pod/azure-vote-front-65cdfc98d6-f6psf     1/1     Terminating   0          76s
pod/azure-vote-front-65cdfc98d6-sn7k9     1/1     Terminating   0          83s
pod/azure-vote-front-65cdfc98d6-xmbsc     1/1     Terminating   0          83s
pod/azure-vote-front-85fc599dd5-4r8pn     1/1     Running       0          16s
pod/azure-vote-front-85fc599dd5-6d986     1/1     Running       0          8s
pod/azure-vote-front-85fc599dd5-9rwmz     1/1     Running       0          16s

NAME                   TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)        AGE
service/azure-vote-back    ClusterIP      10.0.252.176   <none>          6379/TCP       3h18m
service/azure-vote-front   LoadBalancer   10.0.215.9     52.186.43.166   80:30556/TCP   3h18m

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/azure-vote-back   1/1     1            1           3h18m
deployment.apps/azure-vote-front  3/3     3            3           3h18m

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/azure-vote-back-7d6d77f4ff   1         1         1       3h18m
replicaset.apps/azure-vote-front-65cdfc98d6  0         0         0       3h18m
replicaset.apps/azure-vote-front-85fc599dd5  3         3         3       176m
```



Bringing Things Together - Voting App

- We like Cats and Dogs better…roll back to v1!
  - You can update the YAML and re-apply (`kubectl apply -f sample-voting-app.yaml`)
  - You can also use `kubectl`

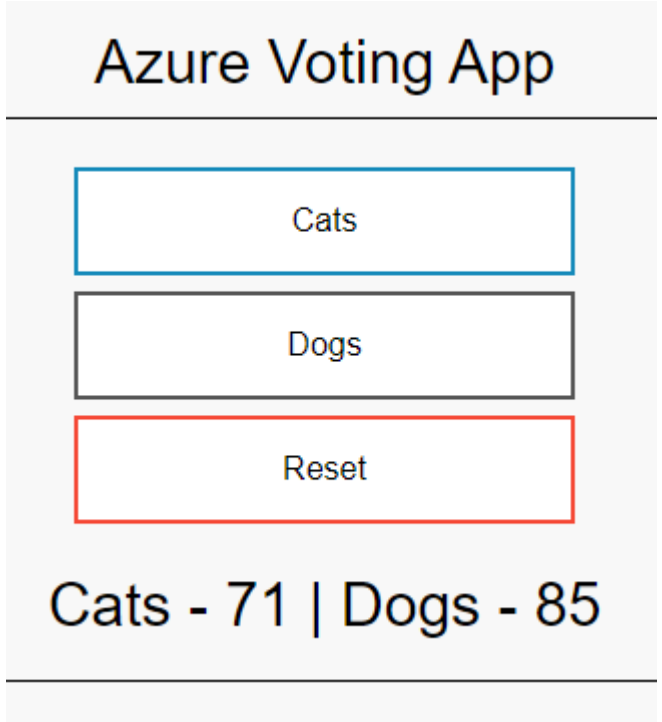
```
c:\code\git
λ kubectl rollout undo deploy azure-vote-front -n voting
deployment.apps/azure-vote-front rolled back
```

```
c:\code\git
λ kubectl get all -n voting
NAME                                     READY   STATUS        RESTARTS   AGE
pod/azure-vote-back-7d6d77f4ff-659h2     1/1     Running       0          3h23m
pod/azure-vote-front-65cdfc98d6-hdm7v    1/1     Running       0          8s
pod/azure-vote-front-65cdfc98d6-rpz48    1/1     Running       0          16s
pod/azure-vote-front-65cdfc98d6-vwcgk    1/1     Running       0          16s
pod/azure-vote-front-85fc599dd5-4r8pn    1/1     Terminating   0          5m3s
pod/azure-vote-front-85fc599dd5-6d986    1/1     Terminating   0          4m55s
pod/azure-vote-front-85fc599dd5-9rwmz    0/1     Terminating   0          5m3s

NAME                  TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)        AGE
service/azure-vote-back    ClusterIP      10.0.252.176   <none>         6379/TCP       3h23m
service/azure-vote-front   LoadBalancer   10.0.215.9     52.186.43.166  80:30556/TCP   3h23m

NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/azure-vote-back      1/1     1            1           3h23m
deployment.apps/azure-vote-front     3/3     3            3           3h23m

NAME                                            DESIRED   CURRENT   READY   AGE
replicaset.apps/azure-vote-back-7d6d77f4ff      1         1         1       3h23m
replicaset.apps/azure-vote-front-65cdfc98d6     3         3         3       3h23m
replicaset.apps/azure-vote-front-85fc599dd5     0         0         0       3h1m
```



Exploring Further

- In the lab exercise, you'll
  - Create liveness and readiness probes for the front-end pods
  - Move the location of the Redis backend to a ConfigMap and have the front-end pods refer to it
  - Externalize Redis persistence to Azure Storage using a dynamic PersistentVolume
- Challenge #1: Create a 'blue' and a 'green' version of the voting app deployment
  - Point your service to either blue or green to simulate a blue/green style deployment
- Challenge #2: Look into StatefulSets and create a Redis cluster in AKS!