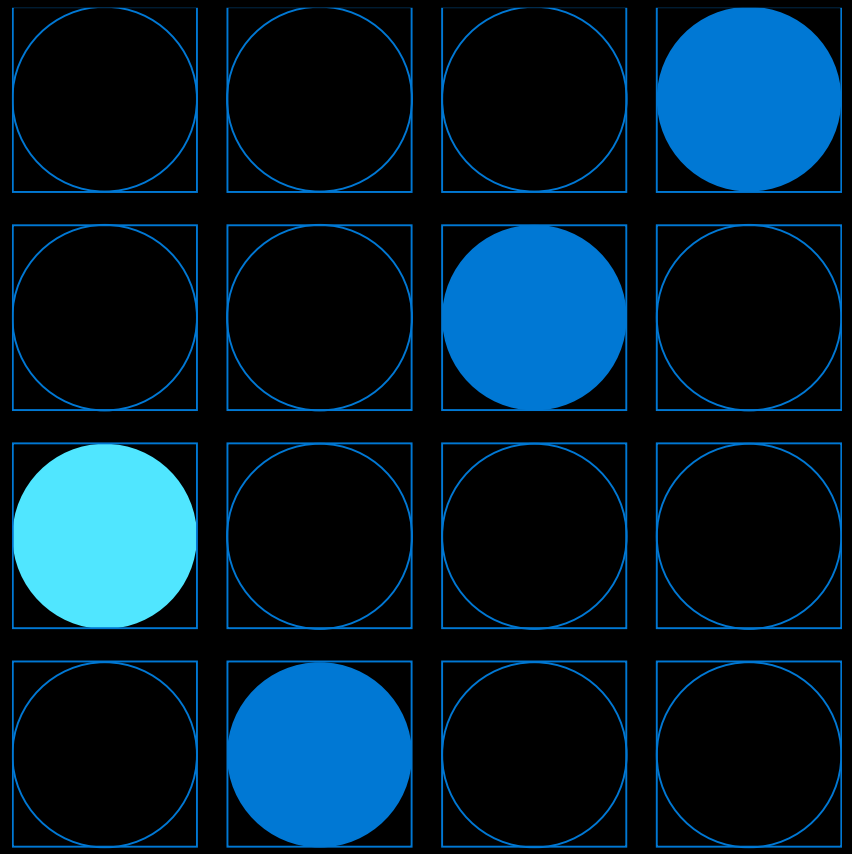
Microsoft Azure

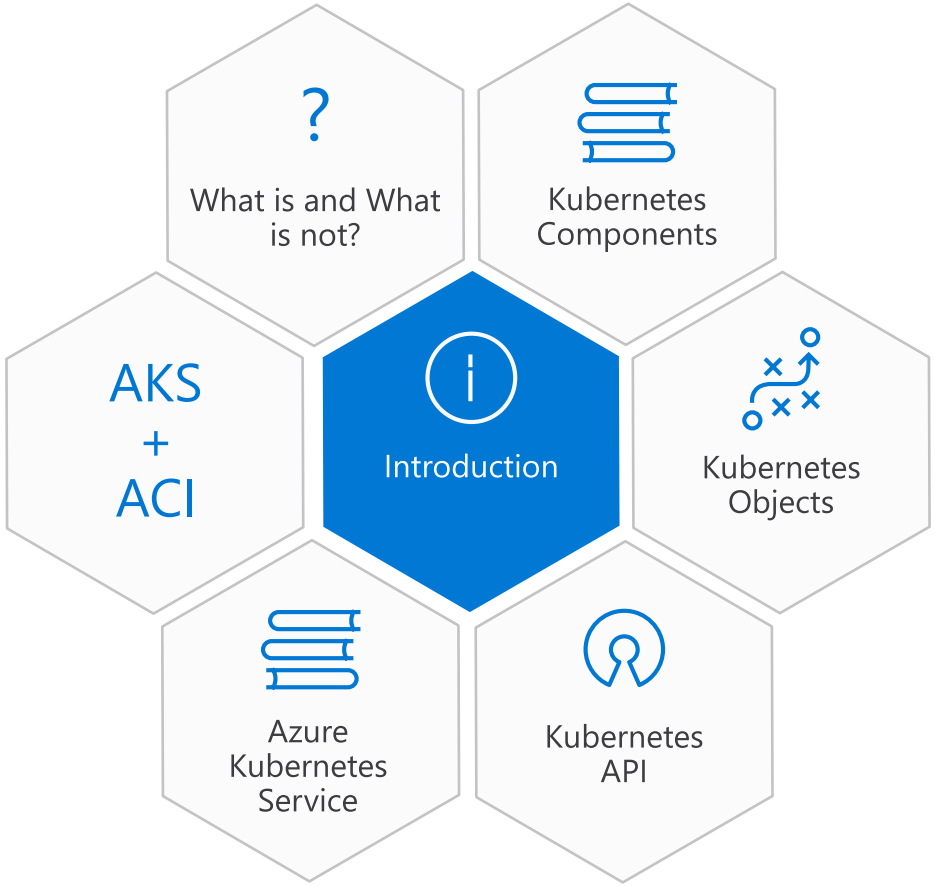
Microsoft Azure DevDays



Microsoft Azure

Kubernetes

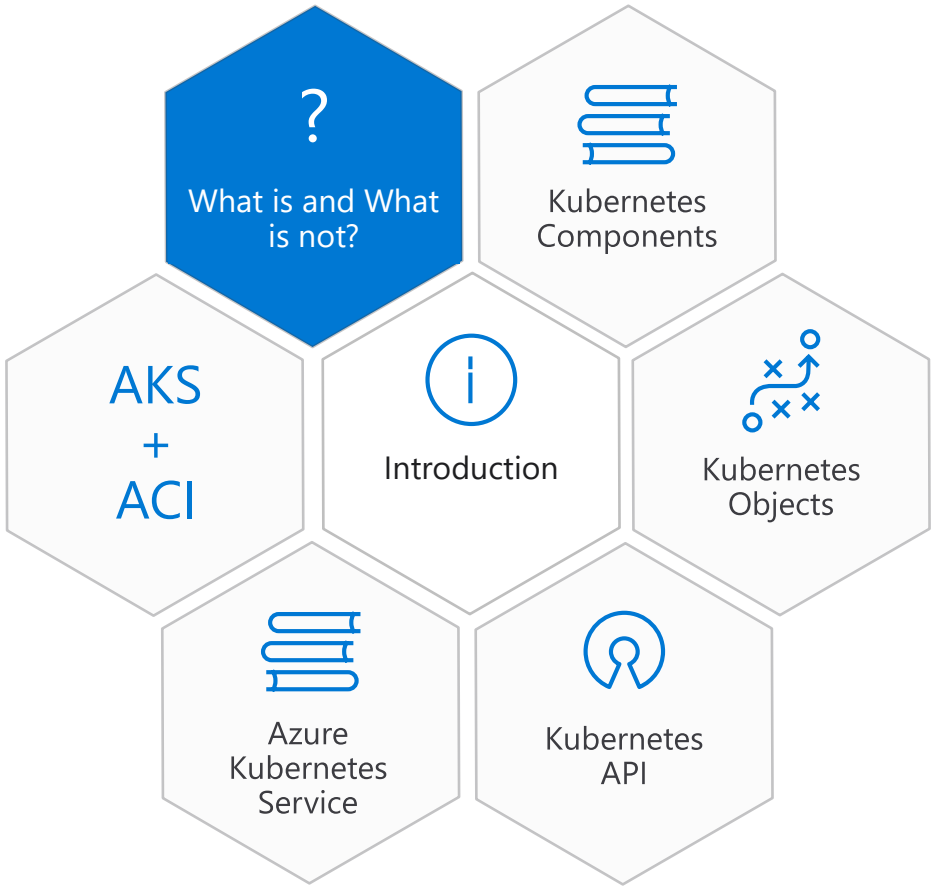
Introduction



Features of Kubernetes

1. Automatic Binpacking
2. Service Discovery & Load Balancing
3. Storage Orchestration
4. Self Healing
5. Secret & Configuration Management
6. Batch Execution
7. Horizontal Scaling
8. Automatic Rollbacks & Rollouts

What is and What is not Kubernetes?

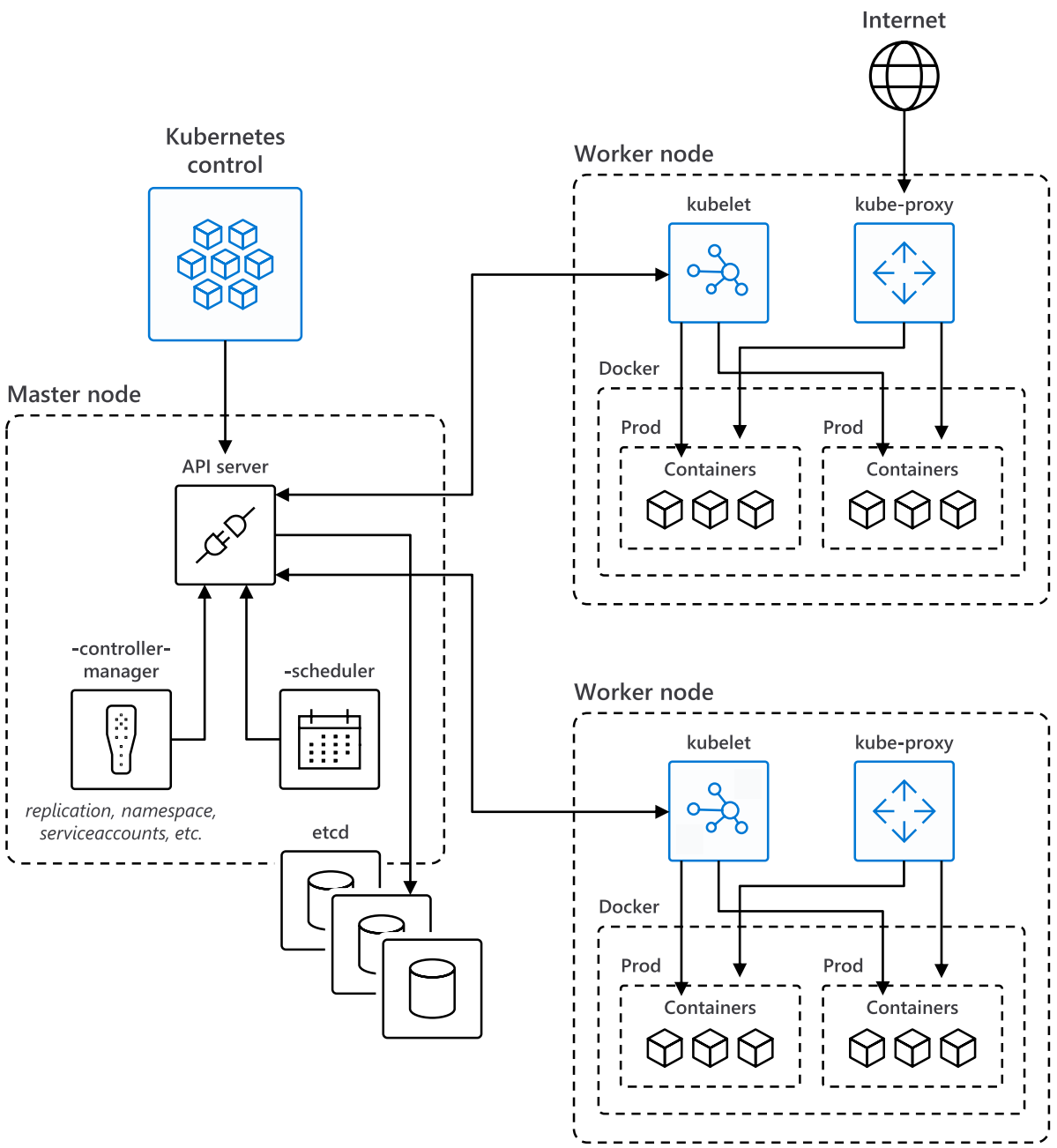


What is Kubernetes?

- Kubernetes (K8s) is an open source project that was released by Google in June, 2014.
- Kubernetes is an orchestration framework for Docker containers which helps expose containers as services to the outside world
- It is a multi-container management solution.

How Kubernetes works?

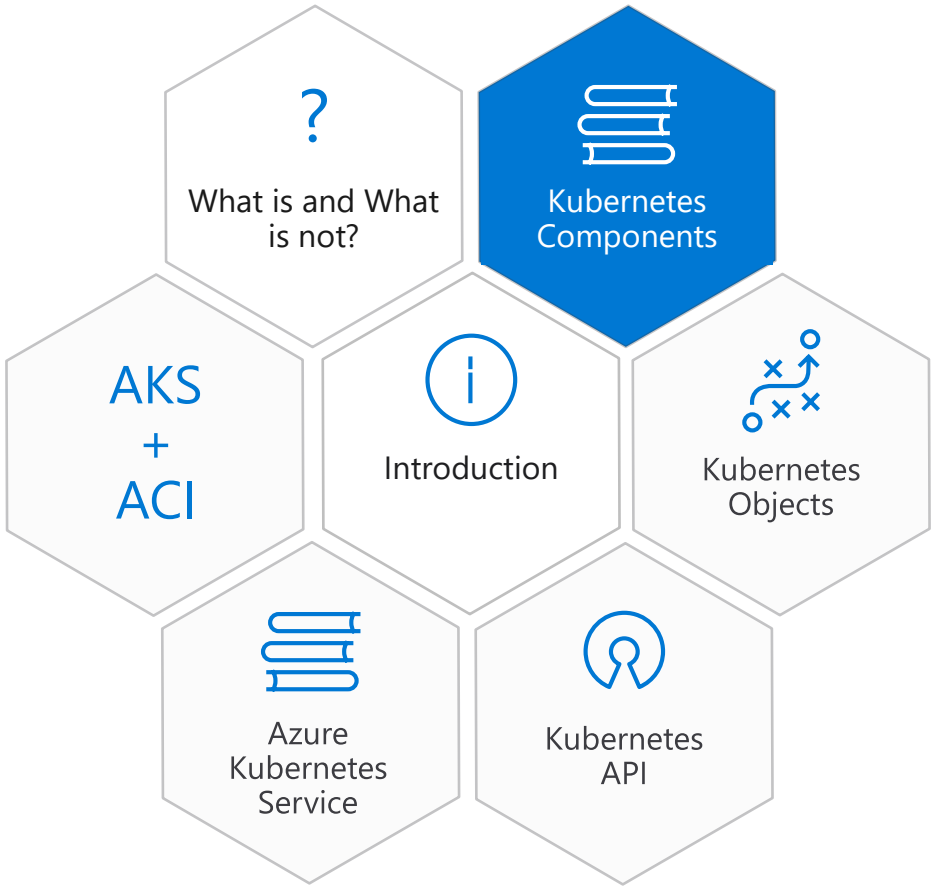
1. Kubernetes users communicate with API server and apply desired state
2. Master nodes actively enforce desired state on worker nodes
3. Worker nodes support communication between containers
4. Worker nodes support communication from the Internet



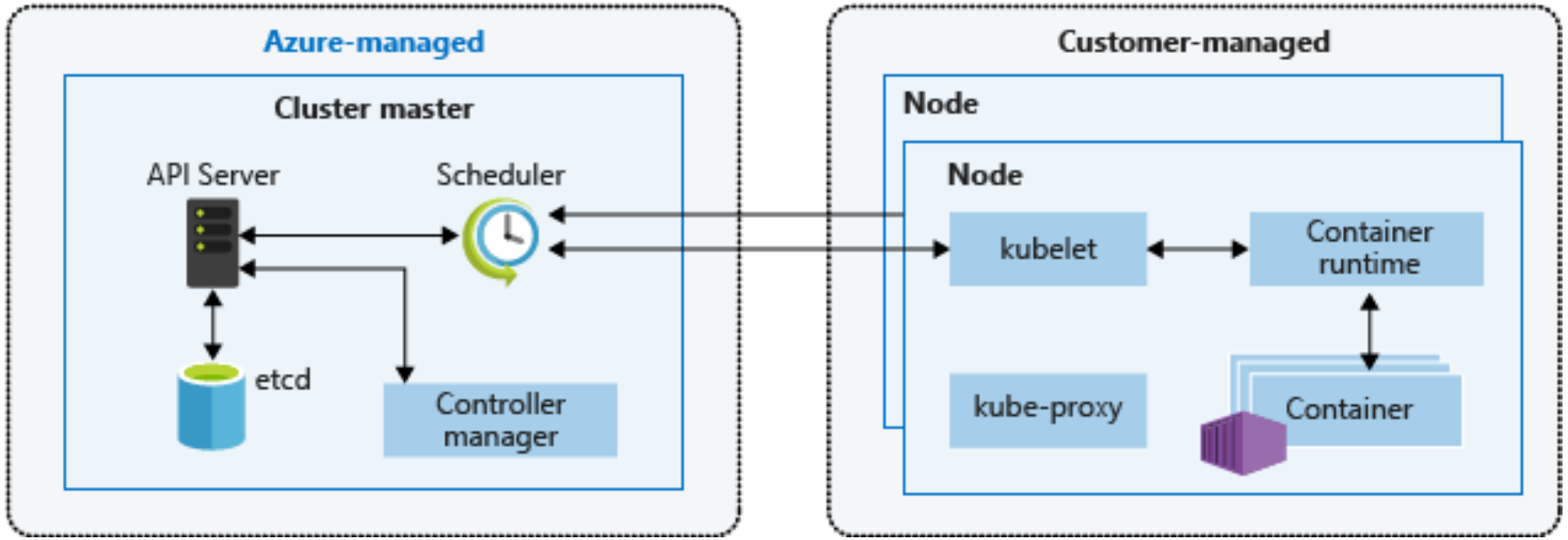
What is not Kubernetes?

- Kubernetes is not a containerization platform
- Does not limit the types of applications supported
- Does not deploy source code and does not build your application
- Does not provide application-level services
- Does not dictate logging, monitoring, or alerting solutions
- Does not provide nor mandate a configuration language/system (e.g Jsonnet)
- Does not provide nor adopt any comprehensive machine configuration, maintenance, management, or self-healing systems

Kubernetes Components



Kubernetes cluster architecture



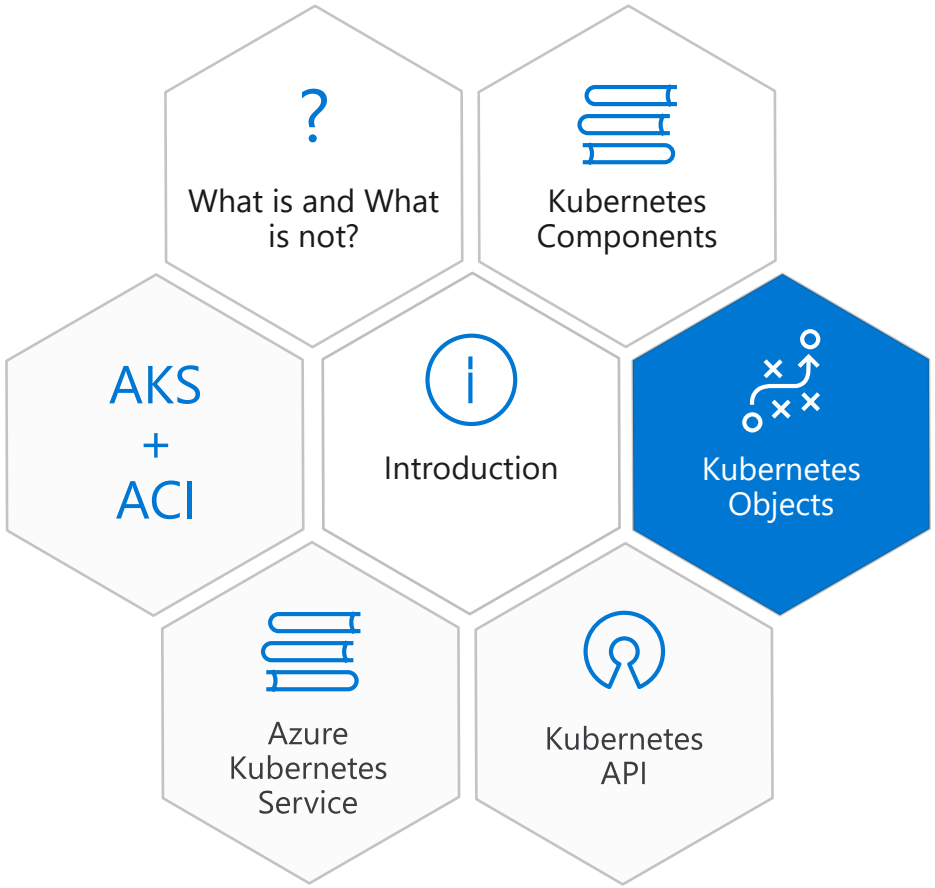
Master Components

- kube-apiserver
- Etcd
- Kube-scheduler
- Kube-controller-manager
- Cloud-controller-manager

Node Components

- kubelet
- Kube-proxy
- Container Runtime

Kubernetes Objects



Kubernetes Objects

- What containerized applications are running (and on which nodes)
- The resources available to those applications
- The policies around how those applications behave, such as restart policies, upgrades, and fault-tolerance

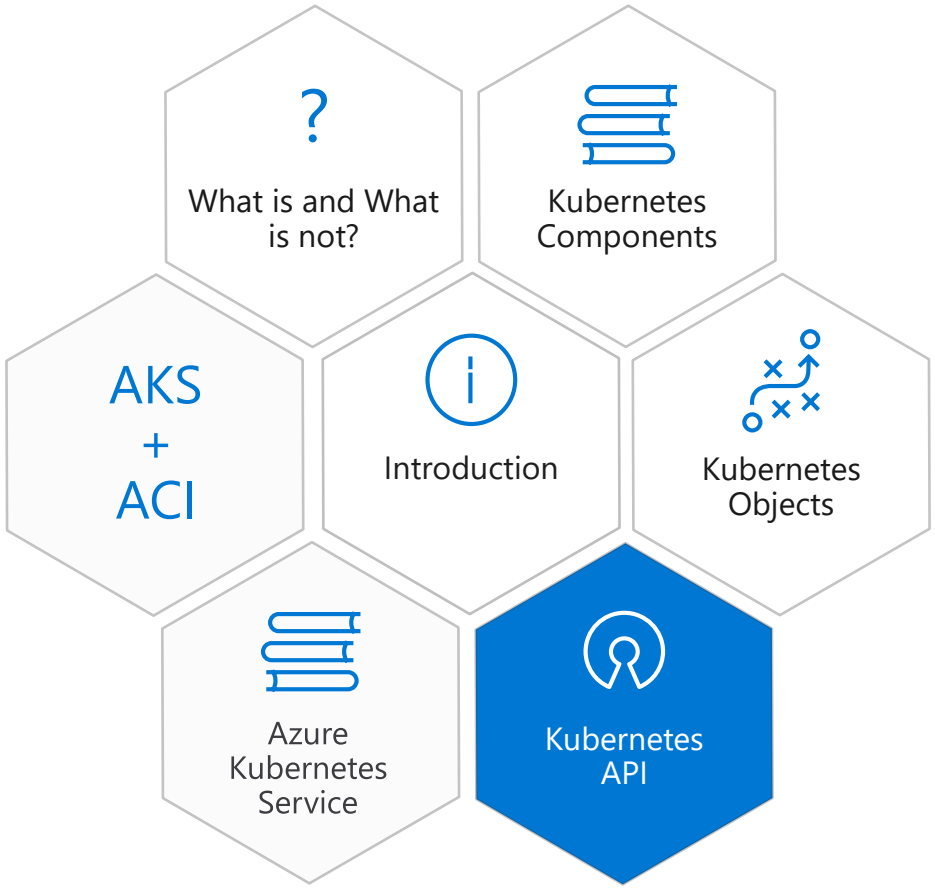
Describing a Kubernetes Object

```yaml
application/deployment.yaml
apiVersion: apps/v1 # for versions before 1.9.0 use apps/v1beta2
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # tells deployment to run 2 pods matching the template
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Required Fields

- apiVersion - Which version of the Kubernetes API you're using to create this object
- kind - What kind of object you want to create
- metadata - Data that helps uniquely identify the object, including a name string, UID, and optional namespace

Kubernetes API



Kinds

- Objects represent a persistent entity in the system
- Lists are collections of resources of one (usually) or more (occasionally) kinds
- Simple kinds are used for specific actions on objects and for non-persistent entities

Objects

Creating an API object is a record of intent - once created, the system will work to ensure that resource exists. All API objects have common metadata.

An object may have multiple resources that clients can use to perform specific actions that create, update, delete, or get.

Lists

The name of a list kind must end with "List". Lists have a limited set of common metadata. All lists use the required "items" field to contain the array of objects they return. Any kind that has the "items" field must be a list kind.

Most objects defined in the system should have an endpoint that returns the full set of resources, as well as zero or more endpoints that return subsets of the full list. Some objects may be singletons (the current user, the system defaults) and may not have lists.

Simple

Given their limited scope, they have the same set of limited common metadata as lists.

For instance, the "Status" kind is returned when errors occur and is not persisted in the system.

Many simple resources are "subresources", which are rooted at API paths of specific resources. When resources wish to expose alternative actions or views that are closely coupled to a single resource, they should do so using new sub-resources.

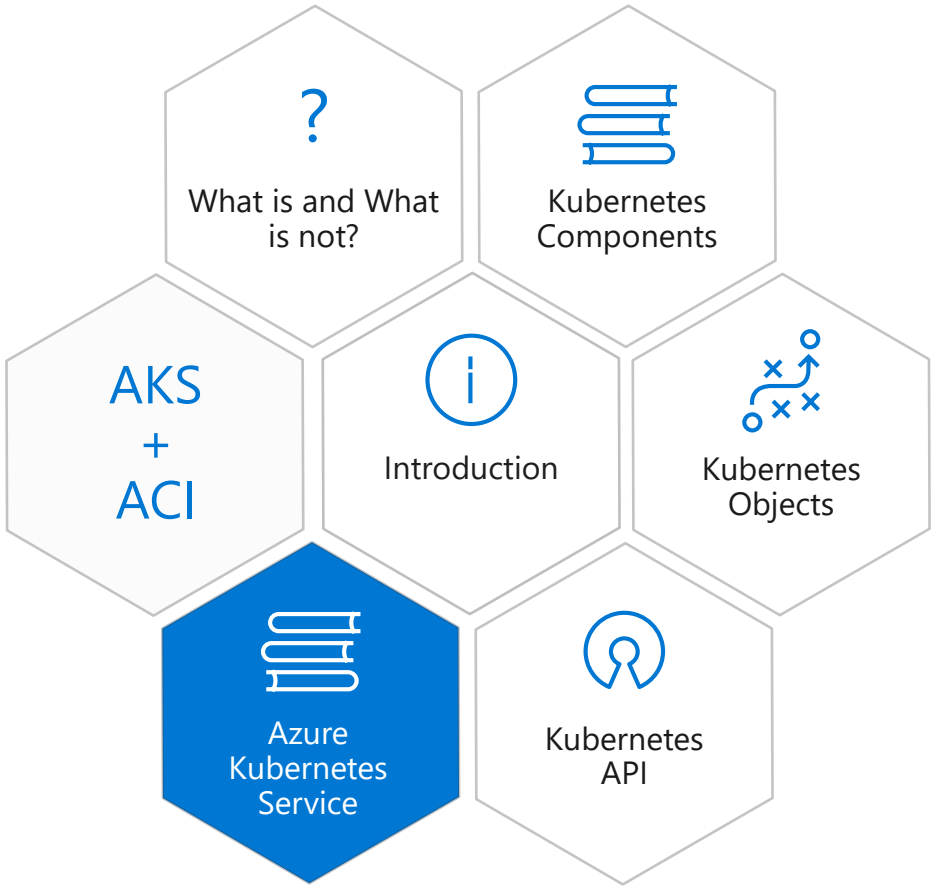
Simple

/binding Used to bind a resource representing a user request (e.g., Pod, PersistentVolumeClaim) to a cluster infrastructure resource (e.g., Node, PersistentVolume)

/status Used to write just the status portion of a resource. For example, the /pods endpoint only allows updates to metadata and spec, since those reflect end-user intent

/scale Used to read and write the count of a resource in a manner that is independent of the specific resource schema.

Azure Kubernetes Service



AKS: Simplify the deployment, management, and operations of Kubernetes

Deploy and manage Kubernetes with ease

Accelerate containerized application development

Set up CI/CD in a few clicks

Secure your Kubernetes environment

Scale and run applications with confidence

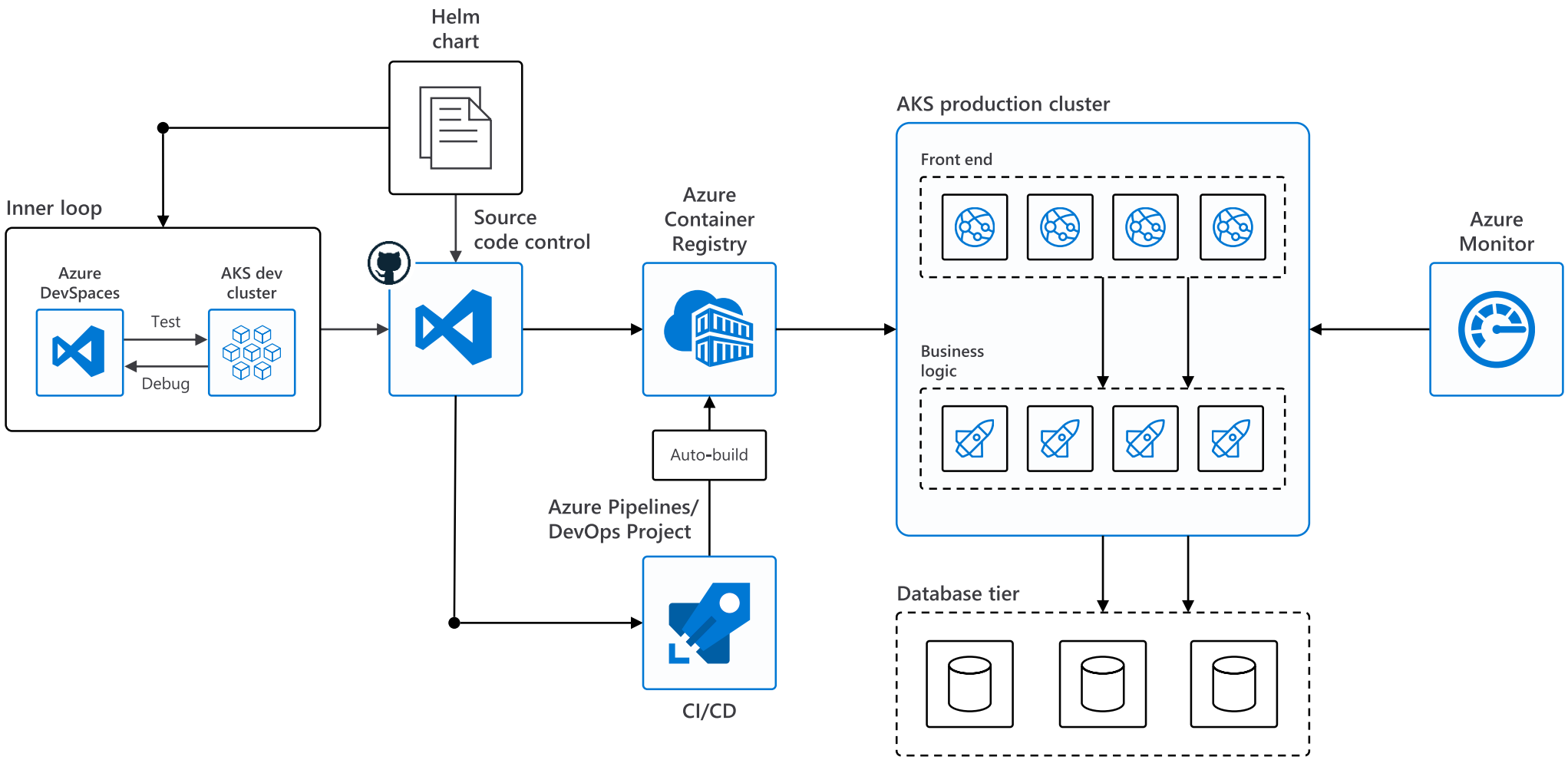
Work how you want with open-source tools and APIs

Azure makes Kubernetes easy

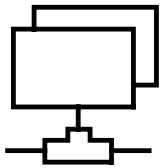
Deploy and manage Kubernetes with ease

Task	The old way	With Azure
Create a cluster	Provision network and VMs Install dozens of system components including etcd Create and install certificates Register agent nodes with control plane	AZ AKS create
Upgrade a cluster	Upgrade your master nodes Cordon/drain and upgrade worker nodes individually	AZ AKS upgrade
Scale a cluster	Provision new VMs Install system components Register nodes with API server	AZ AKS scale

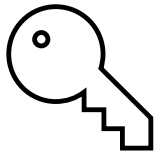
End-to-end experience



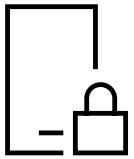
Secure your Kubernetes environment



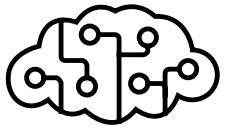
Control access through AAD and RBAC



Safeguard keys and secrets with Key Vault

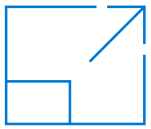


Secure network communications with VNET and CNI



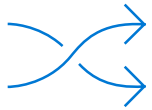
Compliant Kubernetes service with certifications covering SOC, HIPAA, and PCI

Scale and run with confidence

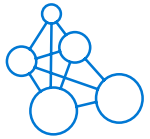


Built-in
auto scaling

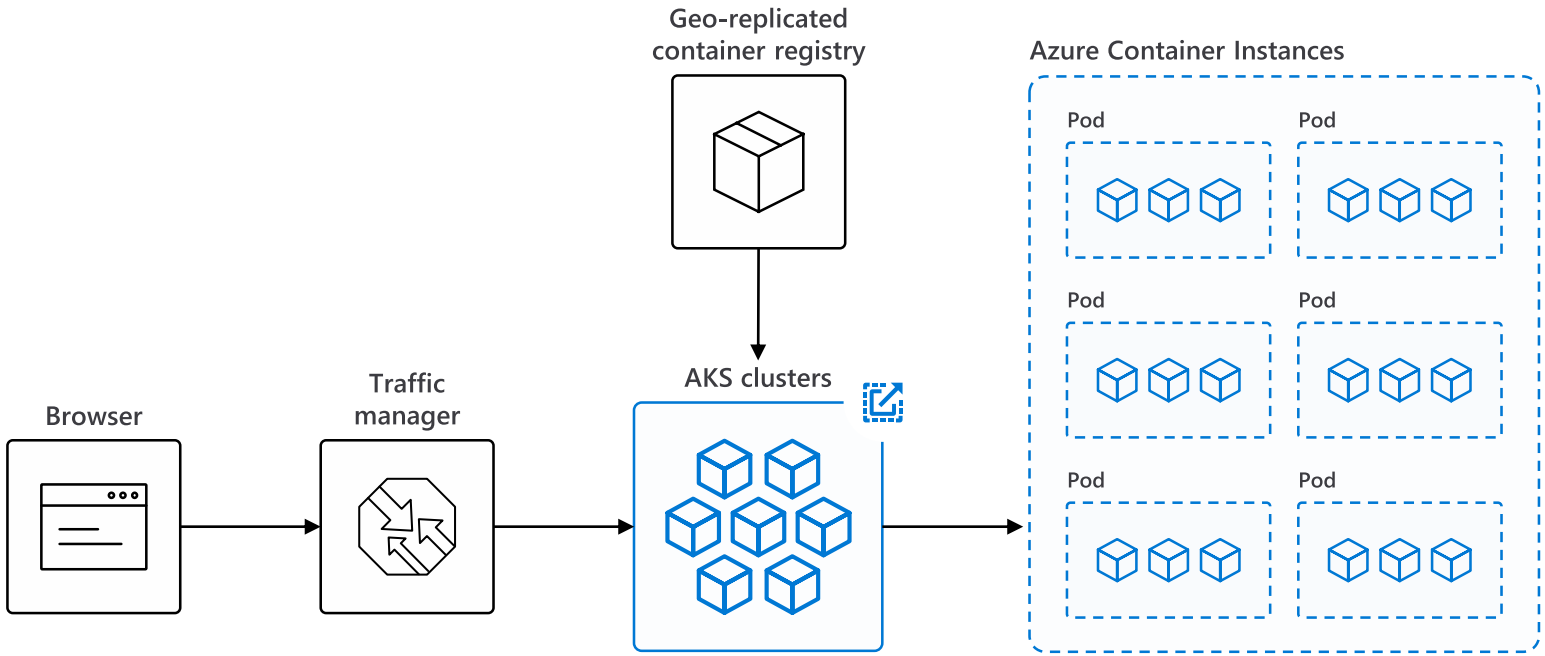
Global
data center



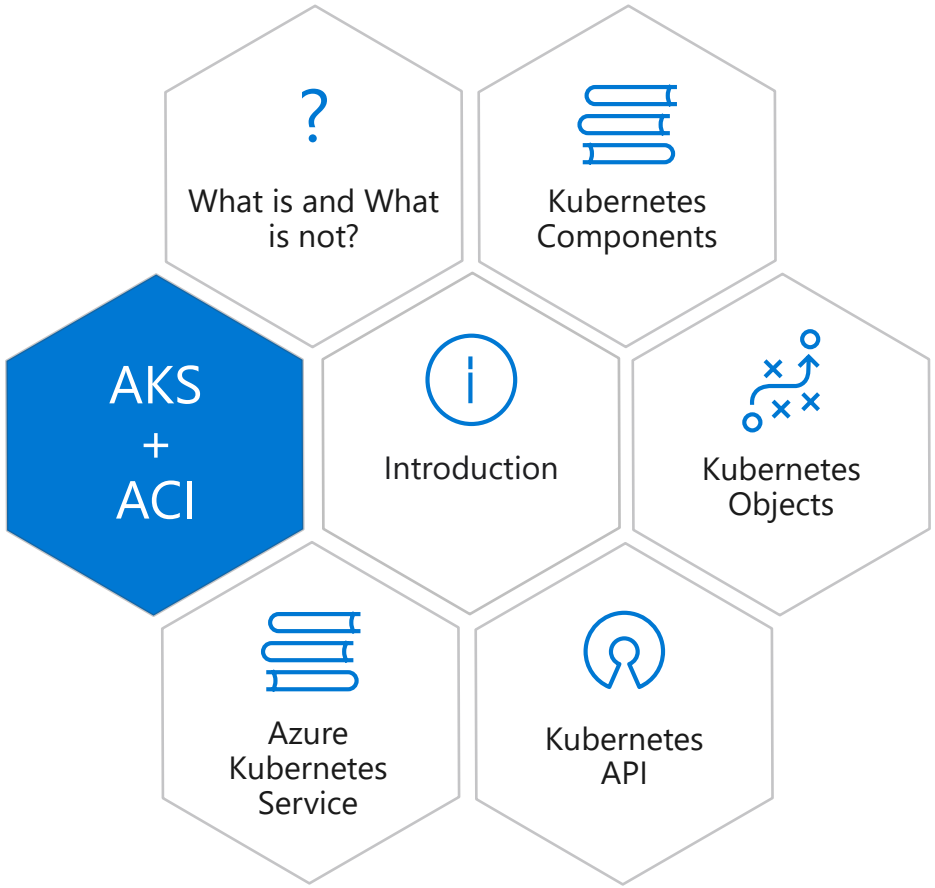
Elastically burst
using ACI



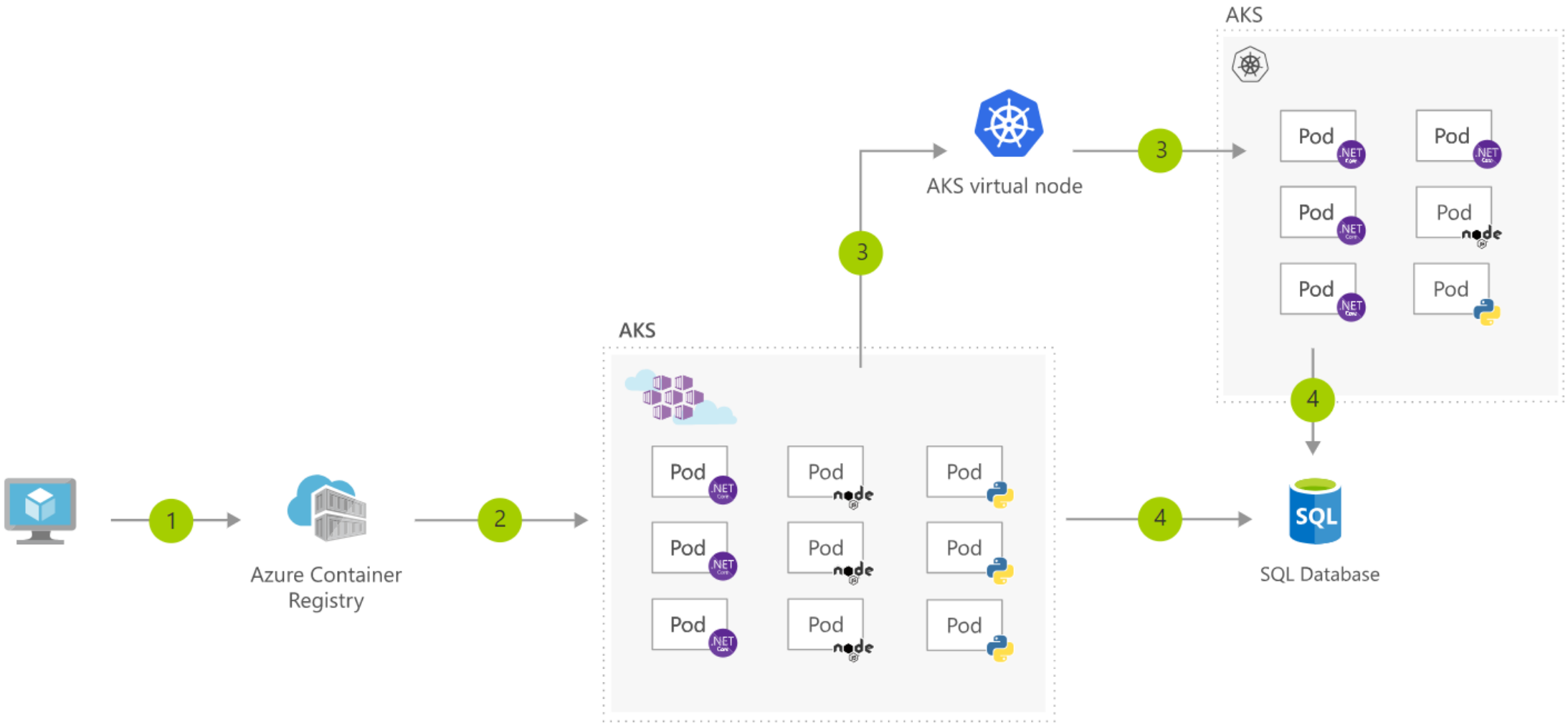
Geo-replicated
container registry



Azure Kubernetes Service



AKS and ACI


AKS
AKS virtual node
3
Pod
Pod
Pod
Pod
Pod
Pod
3
AKS
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod
Pod
1
Azure Container Registry
2
4
4
SQL
SQL Database

Microsoft

Thank You