



*PASS

SQLSATURDAY

EDINBURGH | 01 FEB 2020

"Kubernetify"

Your Containers

Chris Taylor, Jarrin Consultancy



Chris Taylor

Principal Consultant, Jarrin Consultancy



@SQLGeordie



github.com/SQLGeordie/



chris.taylor@jarrinconsultancy.com



www.jarrinconsultancy.com/blog
www.chrisjarrintaylor.co.uk

SQL Server Specialists
Jarrin Consultancy

Experience

Worked with SQL Server since 2001

MCSE – Data Platform

Community

Newcastle DPaC (PASS) Leader

Power BI Newcastle (PBIUG) Leader

DataRelay Newcastle Organiser

Cricket/Football Coaching

Thank you to our Sponsors



..and special THANK YOU to our **Volunteers** (light blue #SQLSat927 polo shirt)

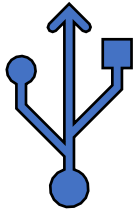
What we'll be doing today

- Containers
- Kubernetes
 - Kubectl
 - Minikube / Docker for Windows
 - Azure Kubernetes Service (AKS)
- (Little bit) Azure DevOps Pipelines
 - Build / Release

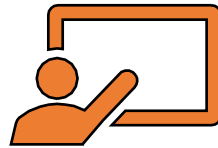
Not on the Agenda

- Deep Dive into Kubernetes Architecture / Internals
 - See Anthony Nocentino's Pluralsight course
 - <https://www.pluralsight.com/authors/anthony-nocentino>
 - <http://www.centinosystems.com/blog/author/aencentinosystems-com/>

Session Aim



Insight into
Kubernetes - Cloud
and On-Prem



Learn by example

- Demo's
- My Mistakes



Enough of a taste to
get the Kubernetes
bug and start
experimenting!

What are Containers?

- Next evolution in virtualisation
- Lightweight, standalone, executable package of software
 - Includes everything needed to run an application: code, runtime, system tools, system libraries and settings.
 - Separation of applications or services on the same container host
 - Isolated, resource controlled, and portable operating environment
 - Containerized software will always run the same, regardless of the environment
- Enables true independence between applications / infrastructure / developers / IT ops

“Basically, a container is an isolated place where an application can run without affecting the rest of the system, and without the system affecting the application.”

<https://docs.microsoft.com/en-us/virtualization/windowscontainers/quick-start/>

<https://www.docker.com/resources/what-container>

What is Kubernetes?

- Open source orchestration engine
 - Designed by Google / used extensively
 - Written in Go/Golang
 - Kubernetes v1.0 was released on July 21, 2015
- Leading orchestrator
 - Easy to deploy and maintain containers
 - Quick to spin up containers
 - High availability built-in
 - Big Data Clusters

Kubernetes is a portable, extensible open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation.

Kubernetes Features

- **Service discovery and load balancing**
 - No need to modify your application to use an unfamiliar service discovery mechanism. Kubernetes gives Pods their own IP addresses and a single DNS name for a set of Pods, and can load-balance across them.
- **Storage orchestration**
 - Automatically mount the storage system of your choice, whether from local storage, a public cloud provider such as [GCP](#) or [AWS](#), or a network storage system such as NFS, iSCSI, Gluster, Ceph, Cinder, or Flocker.
- **Automated rollouts and rollbacks**
 - Kubernetes progressively rolls out changes to your application or its configuration, while monitoring application health to ensure it doesn't kill all your instances at the same time. If something goes wrong, Kubernetes will rollback the change for you. Take advantage of a growing ecosystem of deployment solutions.
- **Batch execution**
 - In addition to services, Kubernetes can manage your batch and CI workloads, replacing containers that fail, if desired.
- **Automatic bin packing**
 - Automatically places containers based on their resource requirements and other constraints, while not sacrificing availability. Mix critical and best-effort workloads in order to drive up utilization and save even more resources.
- **Self-healing**
 - Restarts containers that fail, replaces and reschedules containers when nodes die, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.
- **Secret and configuration management**
 - Deploy and update secrets and application configuration without rebuilding your image and without exposing secrets in your stack configuration.
- **Horizontal scaling**
 - Scale your application up and down with a simple command, with a UI, or automatically based on CPU usage.

Kubernetes Features

- **Service discovery and load balancing**
 - No need to modify your application to use an unfamiliar service discovery mechanism. Kubernetes gives Pods their own IP addresses and a single DNS name for a set of Pods, and can load-balance across them.
- **Storage orchestration**
 - Automatically mount the storage system of your choice, whether from local storage, a public cloud provider such as [GCP](#) or [AWS](#), or a network storage system such as NFS, iSCSI, Gluster, Ceph, Cinder, or Flocker.
- **Automated rollouts and rollbacks**
 - Kubernetes progressively rolls out changes to your application or its configuration, while monitoring application health to ensure it doesn't kill all your instances at the same time. If something goes wrong, Kubernetes will rollback the change for you. Take advantage of a growing ecosystem of deployment solutions.
- **Batch execution**
 - In addition to services, Kubernetes can manage your batch and CI workloads, replacing containers that fail, if desired.
- **Automatic bin packing**
 - Automatically places containers based on their resource requirements and other constraints, while not sacrificing availability. Mix critical and best-effort workloads in order to drive up utilization and save even more resources.
- **Self-healing**
 - Restarts containers that fail, replaces and reschedules containers when nodes die, kills containers that don't respond to your user-defined health check, and doesn't advertise them to clients until they are ready to serve.
- **Secret and configuration management**
 - Deploy and update secrets and application configuration without rebuilding your image and without exposing secrets in your stack configuration.
- **Horizontal scaling**
 - Scale your application up and down with a simple command, with a UI, or automatically based on CPU usage.

What Kubernetes is not

- Does not limit the types of applications supported.
 - Kubernetes aims to support an extremely diverse variety of workloads, including stateless, stateful, and data-processing workloads. If an application can run in a container, it should run great on Kubernetes.
- Does not deploy source code and does not build your application.
 - Continuous Integration, Delivery, and Deployment (CI/CD) workflows are determined by organization cultures and preferences as well as technical requirements.
- Does not provide application-level services
 - E.g. middleware (for example, message buses), data-processing frameworks (e.g. Spark), databases (e.g. mysql), caches, nor cluster storage systems (e.g. Ceph) as built-in services.
 - Such components can run on Kubernetes, and/or can be accessed by applications running on Kubernetes through portable mechanisms, such as the Open Service Broker.
- Does not dictate logging, monitoring, or alerting solutions.
 - It provides some integrations as proof of concept, and mechanisms to collect and export metrics.
- Does not provide nor mandate a configuration language/system
 - It provides a declarative API that may be targeted by arbitrary forms of declarative specifications.
- Kubernetes is not a mere orchestration system.
 - Eliminates need for orchestration.
 - Orchestration is execution of a defined workflow: first do A, then B, then C.
 - Kubernetes is comprised of a set of independent, composable control processes that continuously drive the current state towards the provided desired state.
 - It shouldn't matter how you get from A to C.
 - Centralized control is also not required. This results in a system that is easier to use and more powerful, robust, resilient, and extensible.

What Kubernetes is not

- Does not limit the types of applications supported
 - Kubernetes aims to support an extremely diverse variety of workloads, including stateless, stateful, and data-processing workloads. If an application can run in a container, it should run great on Kubernetes.
- Does not deploy source code and does not build your application
 - Continuous Integration, Delivery, and Deployment (CI/CD) workflows are determined by organization cultures and preferences as well as technical requirements.
- Does not provide application-level services
 - E.g. middleware (for example, message buses), data-processing frameworks (e.g. Spark), databases (e.g. mysql), caches, nor cluster storage systems (e.g. Ceph) as built-in services.
 - Such components can run on Kubernetes, and/or can be accessed by applications running on Kubernetes through portable mechanisms, such as the Open Service Broker.
- Does not dictate logging, monitoring, or alerting solutions
 - It provides some integrations as proof of concept, and mechanisms to collect and export metrics.
- Does not provide nor mandate a configuration language/system
 - It provides a declarative API that may be targeted by arbitrary forms of declarative specifications.
- Kubernetes is not a mere orchestration system
 - Eliminates need for orchestration.
 - Orchestration is execution of a defined workflow: first do A, then B, then C.
 - Kubernetes is comprised of a set of independent, composable control processes that continuously drive the current state towards the provided desired state.
 - It shouldn't matter how you get from A to C.
 - Centralized control is also not required. This results in a system that is easier to use and more powerful, robust, resilient, and extensible.

What is k8s?

KUBERNETES

“Kubernetes means helmsman in Greek

(a person who drives or steers a ship).“



kubernetes

The symbol is the wheel of the ship.”



kubernetes

<https://kubernetes.io/docs/home>

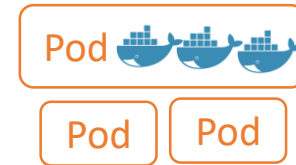
Kubernetes Architecture

- Containers



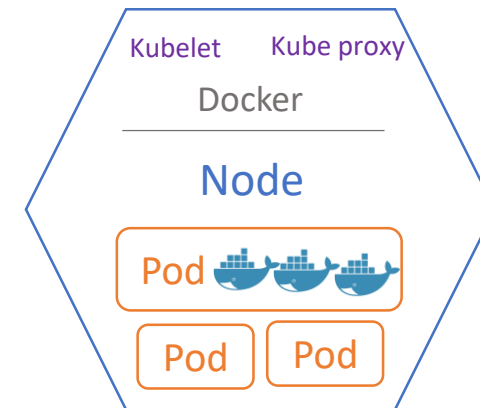
Kubernetes Architecture

- Containers
 - live in *Pods*
- Pod(s)
 - abstractions within *Nodes*



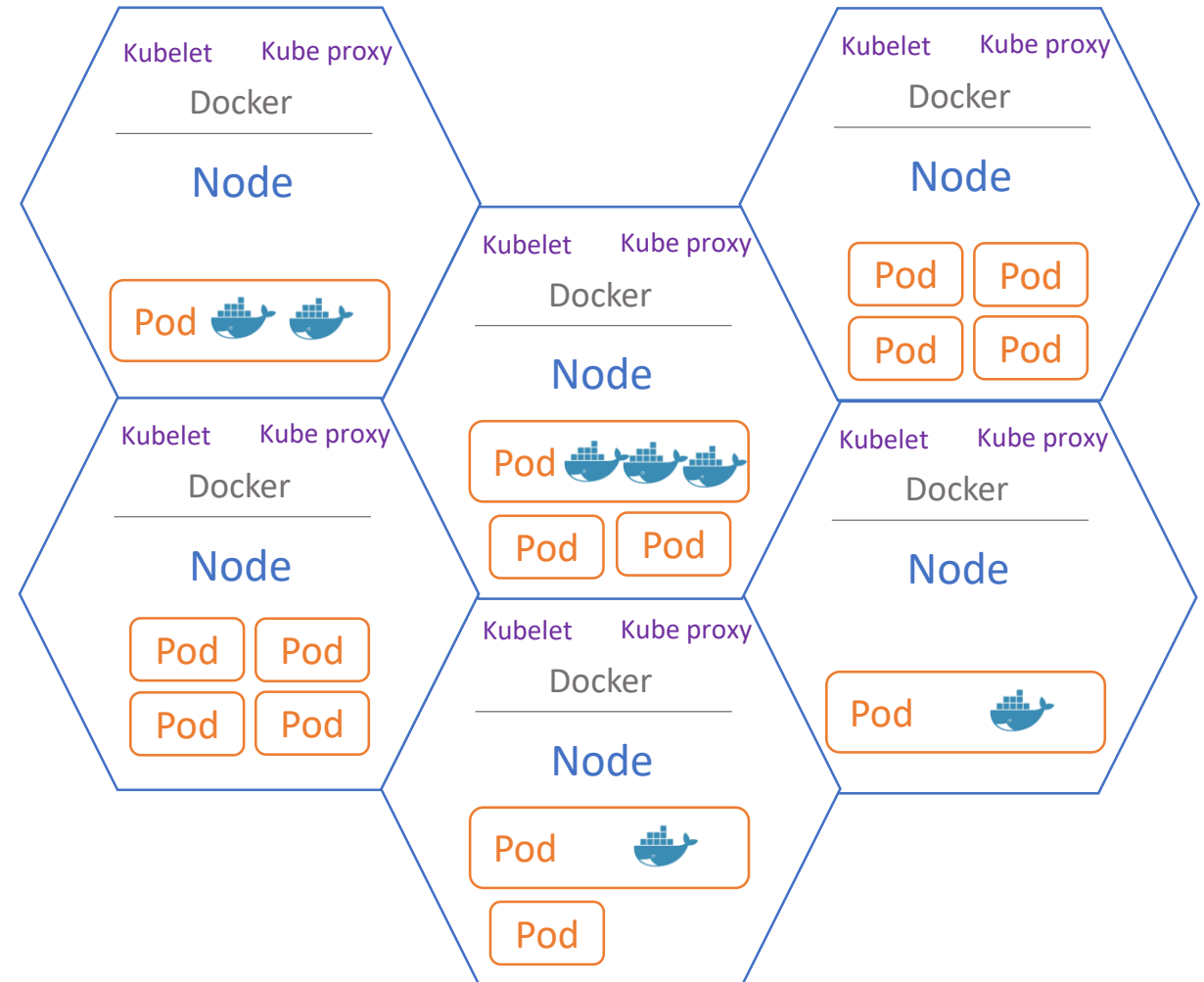
Kubernetes Architecture

- Containers
 - live in *Pods*
- Pod(s)
 - abstractions within *Nodes*
- Node(s)
 - Physical / Virtual machines



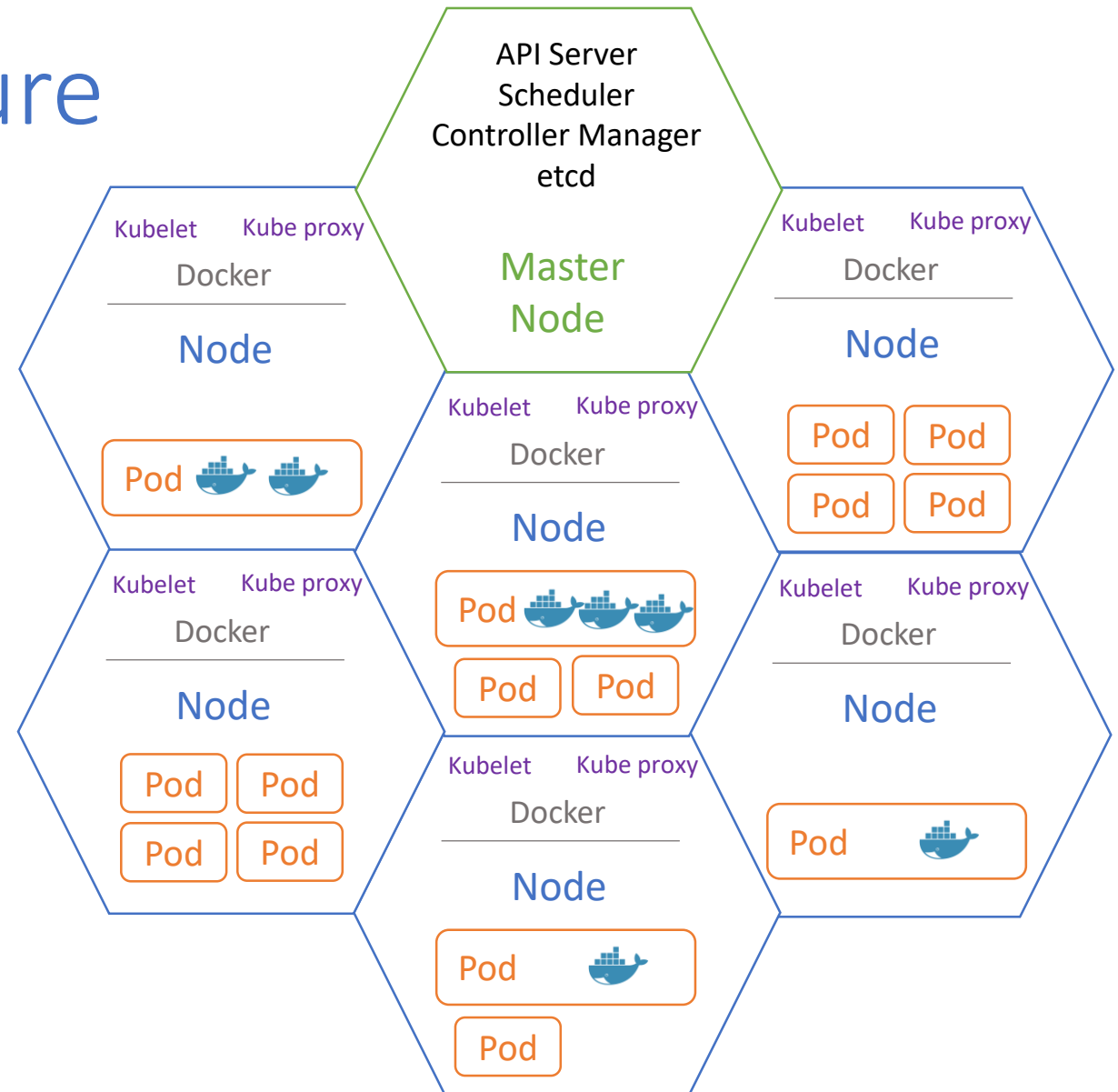
Kubernetes Architecture

- Containers
 - live in **Pods**
- Pod(s)
 - abstractions within **Nodes**
- Node(s)
 - Physical / Virtual machines
- Cluster(s)
 - Group of **Nodes**



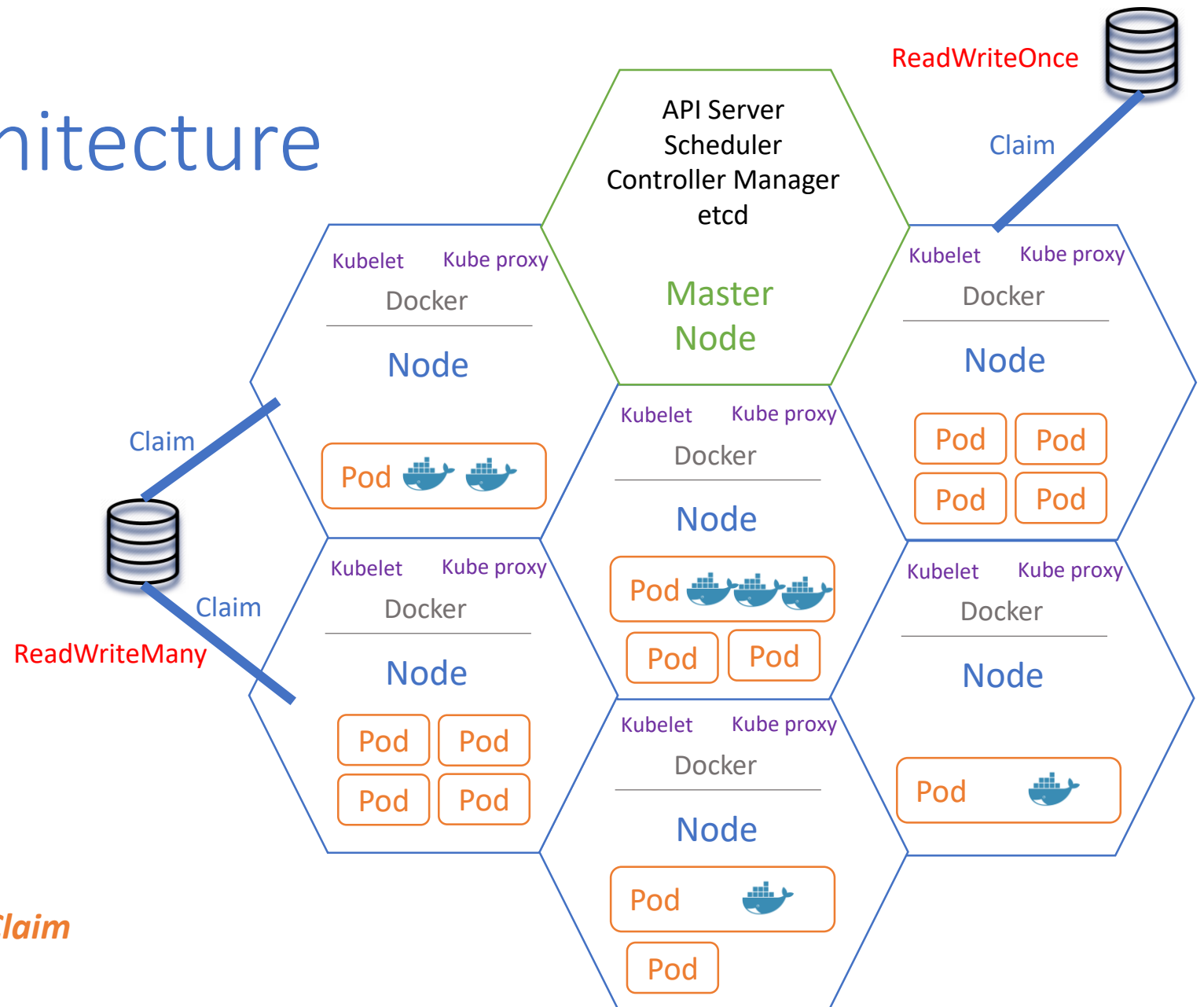
Kubernetes Architecture

- Containers
 - live in **Pods**
- Pod(s)
 - abstractions within **Nodes**
- Node(s)
 - Physical / Virtual machines
- Cluster(s)
 - Group of **Nodes**
- Master Node



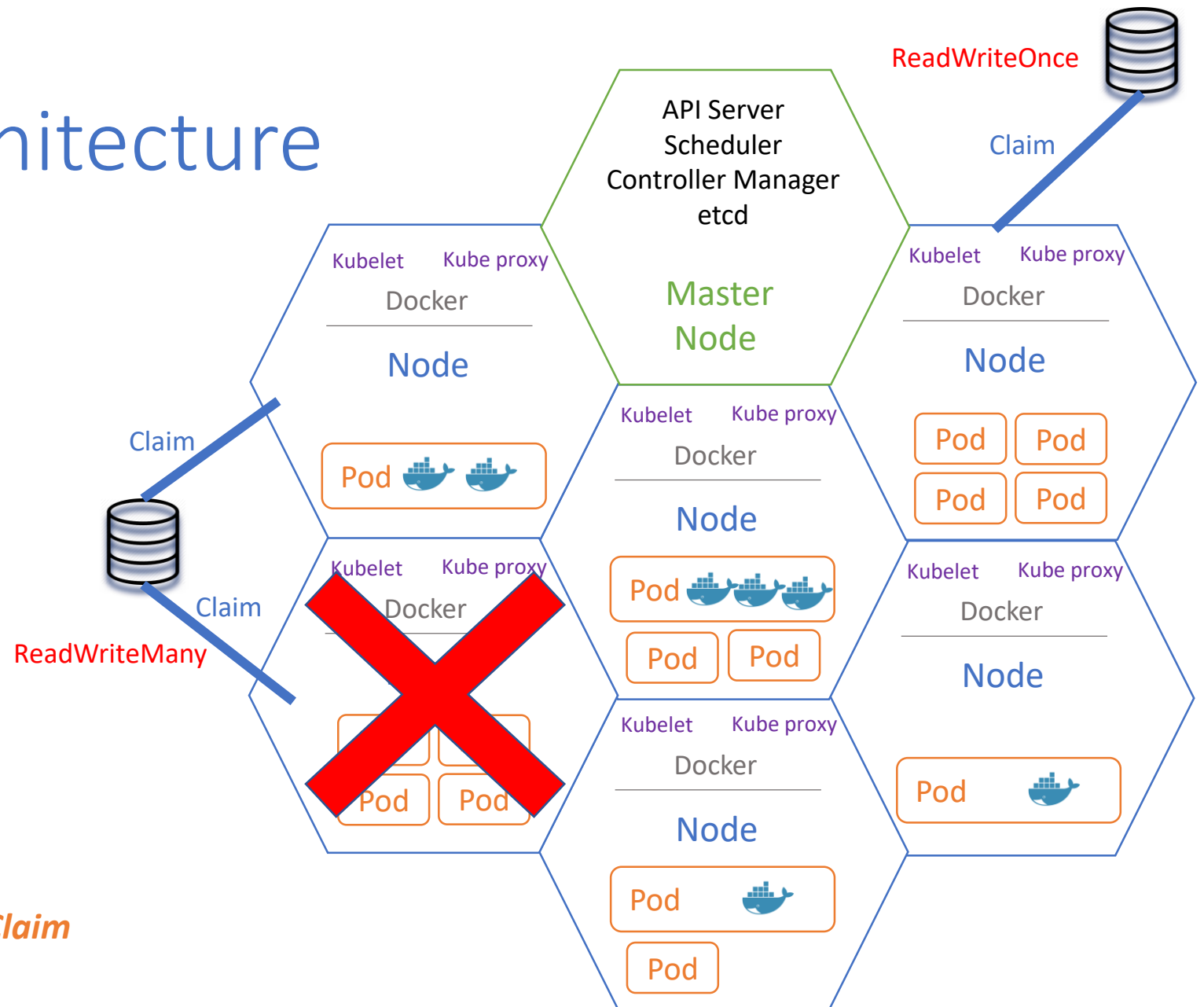
Kubernetes Architecture

- Containers
 - live in **Pods**
- Pod(s)
 - abstractions within **Nodes**
- Node(s)
 - Physical / Virtual machines
- Cluster(s)
 - Group of **Nodes**
- Master Node
- Storage
 - Volumes mounted through a **Claim**
 - Can be **Persisted**
 - Can be shared with other Nodes (RWO / ROX / RWX)



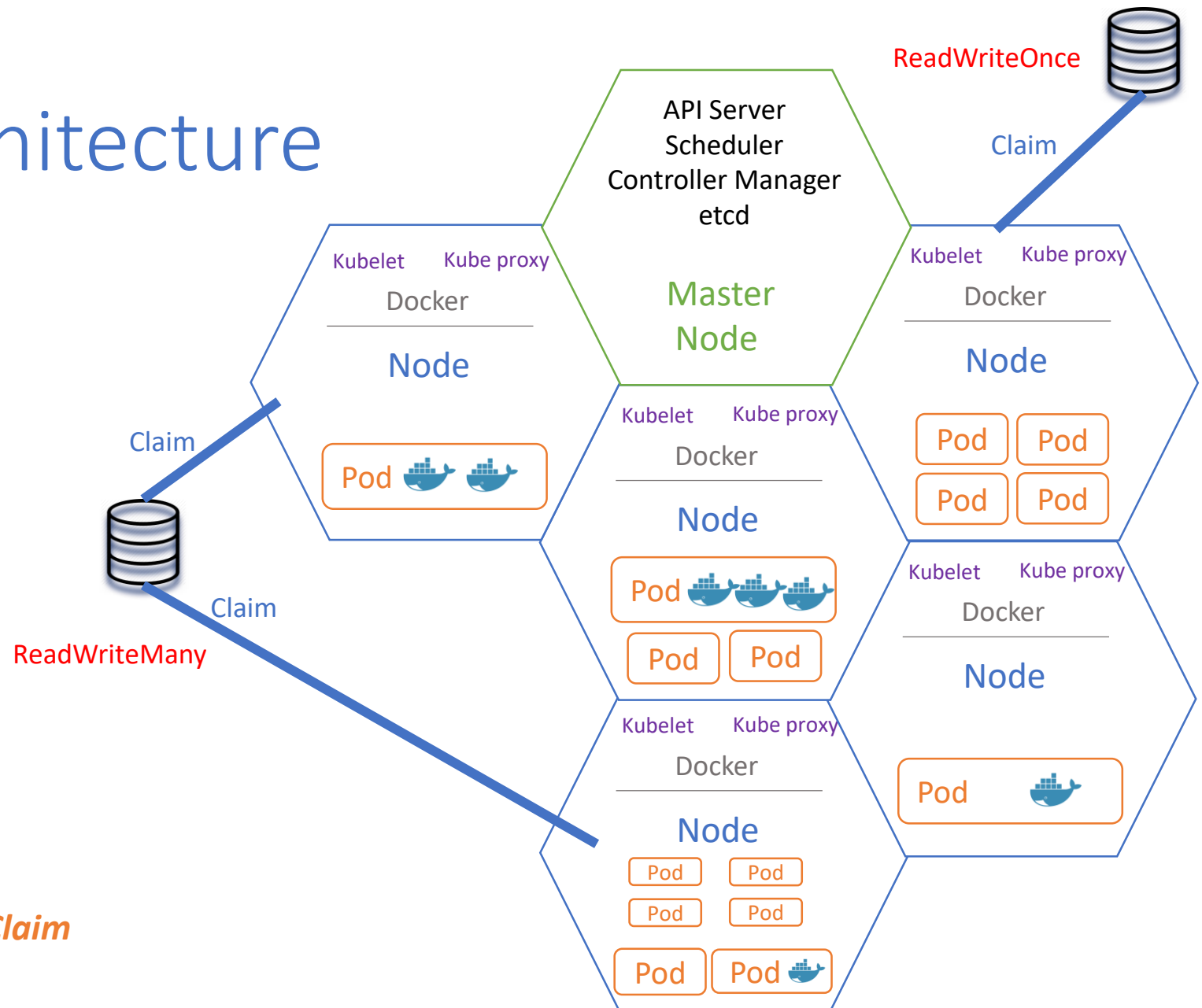
Kubernetes Architecture

- Containers
 - live in **Pods**
- Pod(s)
 - abstractions within **Nodes**
- Node(s)
 - Physical / Virtual machines
- Cluster(s)
 - Group of **Nodes**
- Master Node
- Storage
 - Volumes mounted through a **Claim**
 - Can be **Persisted**
 - Can be shared with other Nodes (RWO / ROX / RWX)



Kubernetes Architecture

- Containers
 - live in **Pods**
- Pod(s)
 - abstractions within **Nodes**
- Node(s)
 - Physical / Virtual machines
- Cluster(s)
 - Group of **Nodes**
- Master Node
- Storage
 - Volumes mounted through a **Claim**
 - Can be **Persisted**
 - Can be shared with other Nodes (RWO / ROX / RWX)

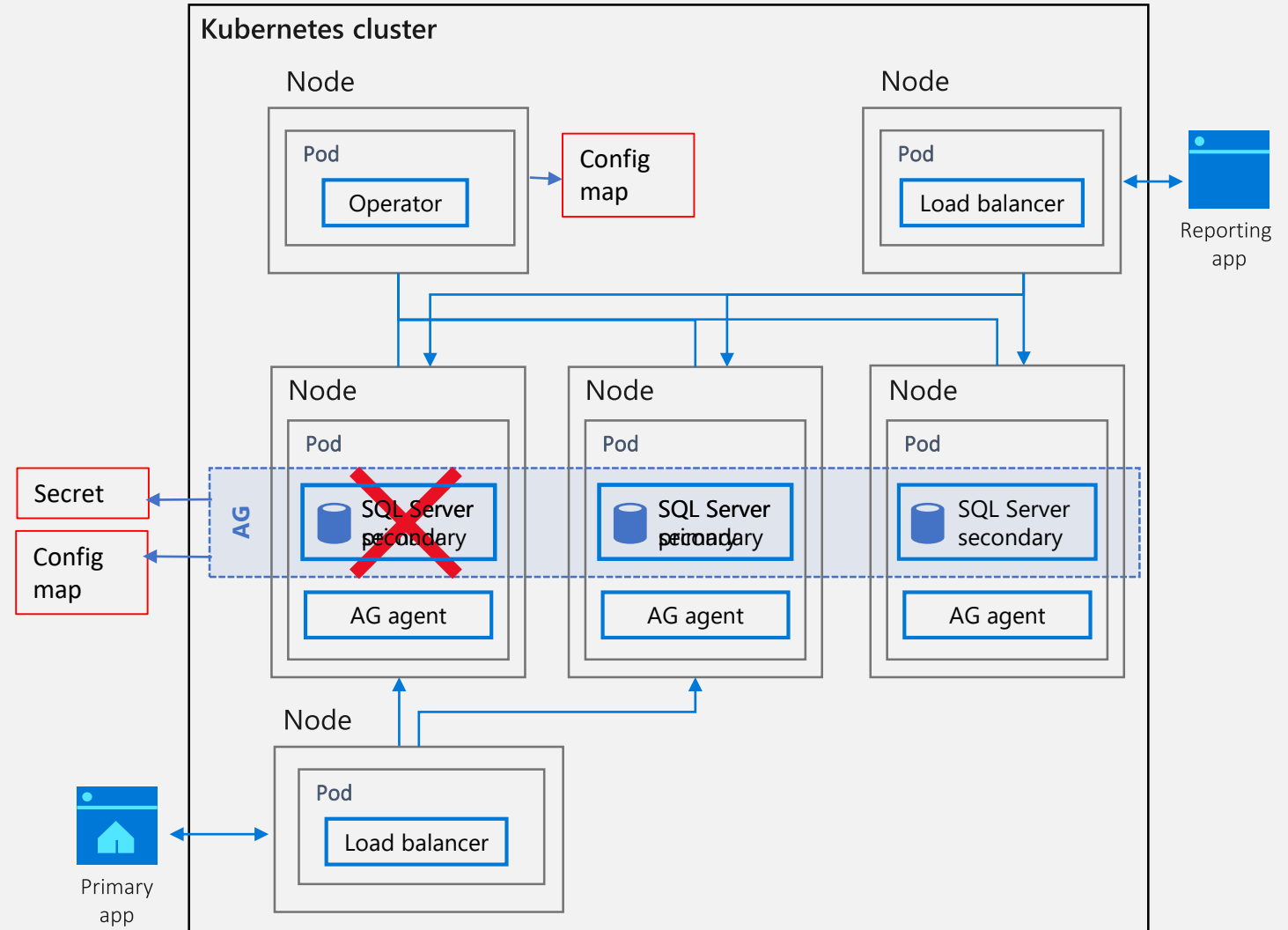


SQL Server 2019

Always On Availability Groups on Kubernetes

- Operator orchestrates
- AG concepts all apply
- Load Balancer for Primary App
- Load Balancer for Secondary Replica Readers
- No need for read write routing from primary listener.
- **ConfigMaps** = configuration settings with environment-specific (param , value)

Availability groups on Kubernetes



SQL Server 2019 – AOAG's/K8s not supported

Always On Availability Group Kubernetes operator not supported

- **Issue and customer impact:** The Kubernetes operator for Always On Availability Groups is not supported in this release candidate and will not be available at RTM.
- **Workaround:** None
- **Applies to:** SQL Server 2019 Release candidate

SQL Server 2019 – AOAG's/K8s not supported

Always On Availability Group Kubernetes operator not supported

- **Issue and customer impact:** The Kubernetes operator for Always On Availability Groups is not supported in this release candidate and will not be available at RTM.
- **Workaround:** None
- **Applies to:** SQL Server 2019 Release candidate

SQL Server 2019 – AOAG's/Containers not supported

Samples scripts for a SQL Server Always On Availability Group on SQL Server Containers, managed by Kubernetes

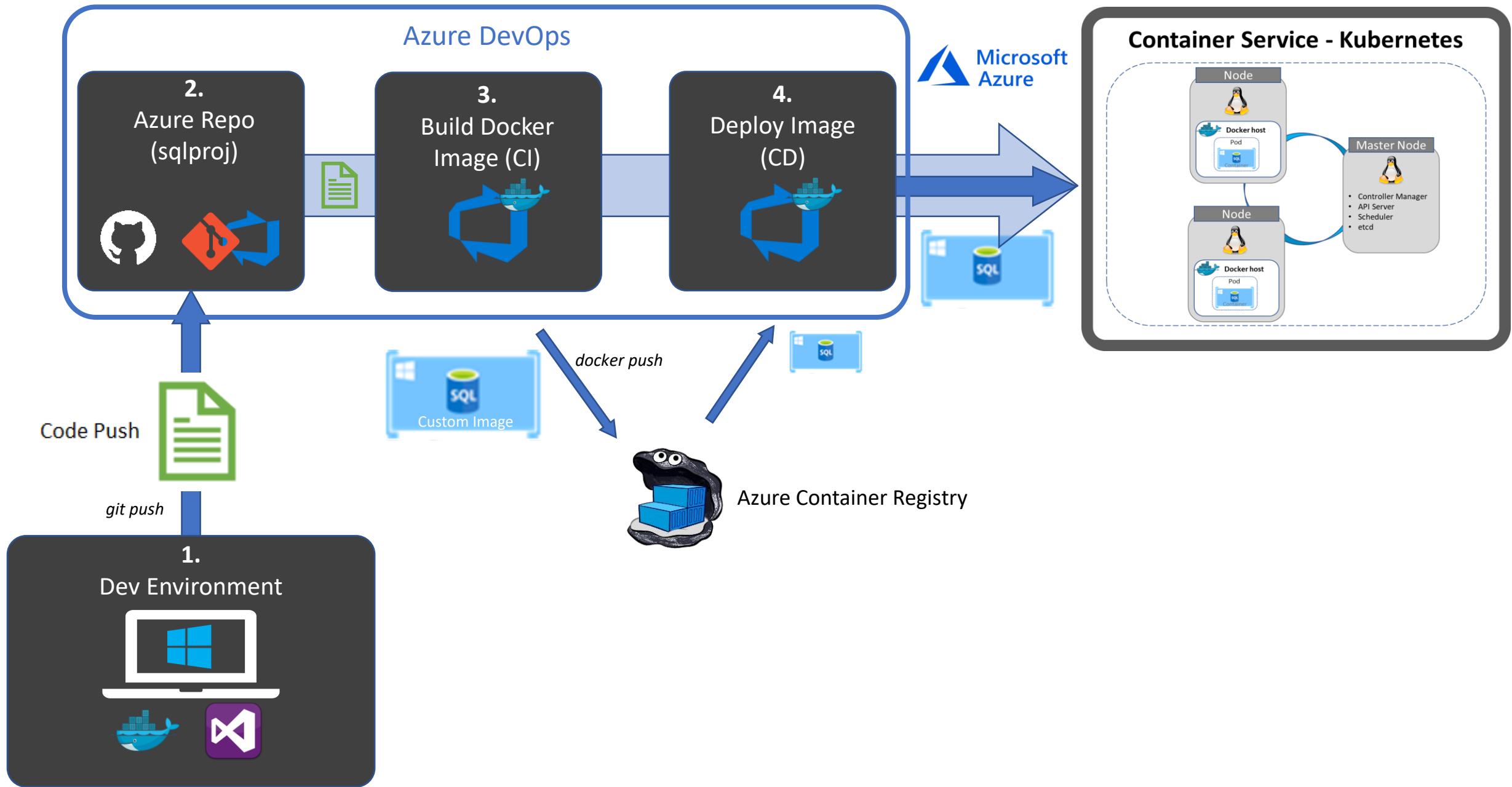
Availability Groups on SQL Server Containers was provided during SQL Server 2019 preview releases to demonstrate a potential capability. SQL Server 2019 does not support Availability Groups on containers.

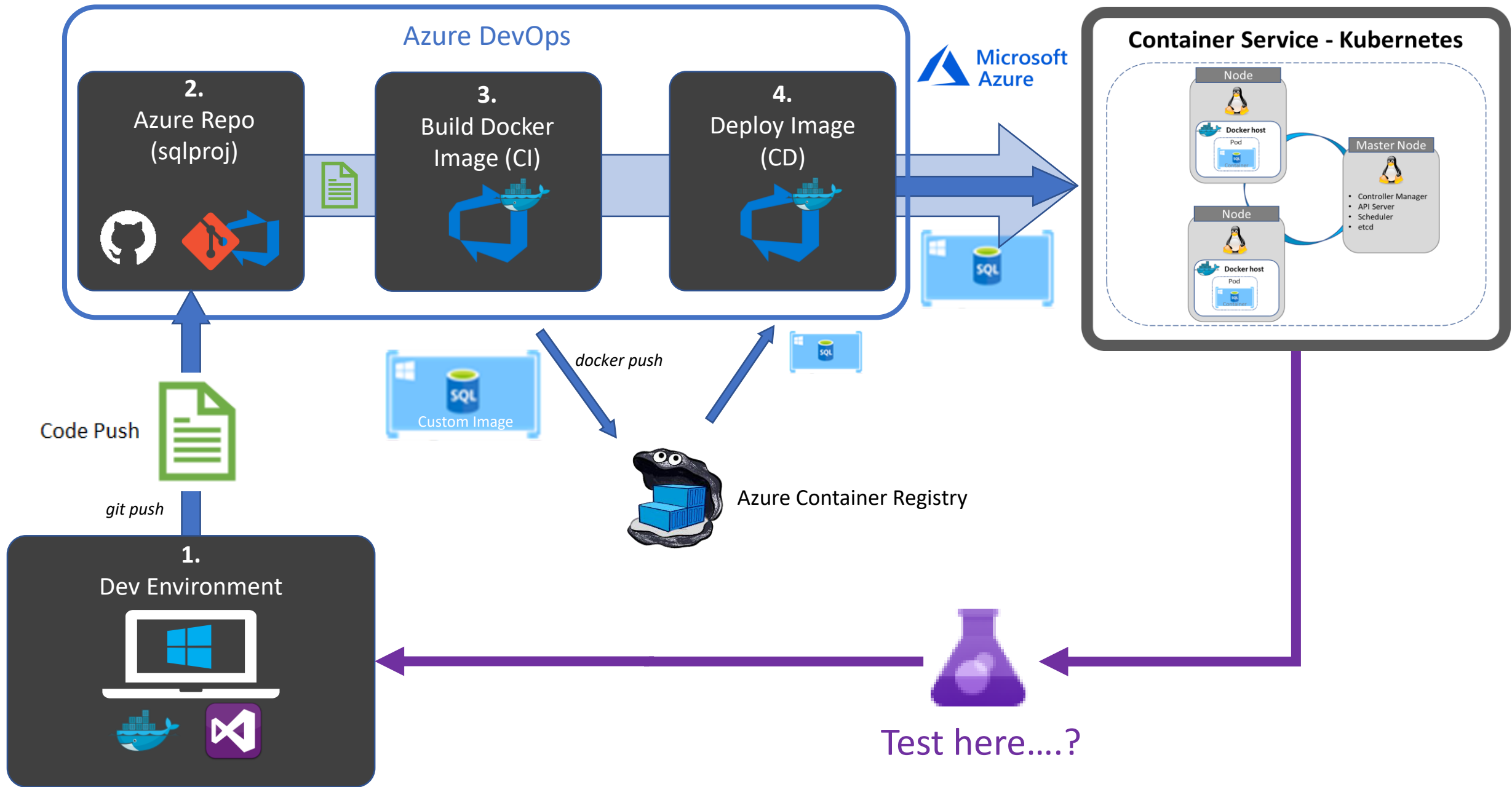
SQL Server 2019 – AOAG's/Containers not supported

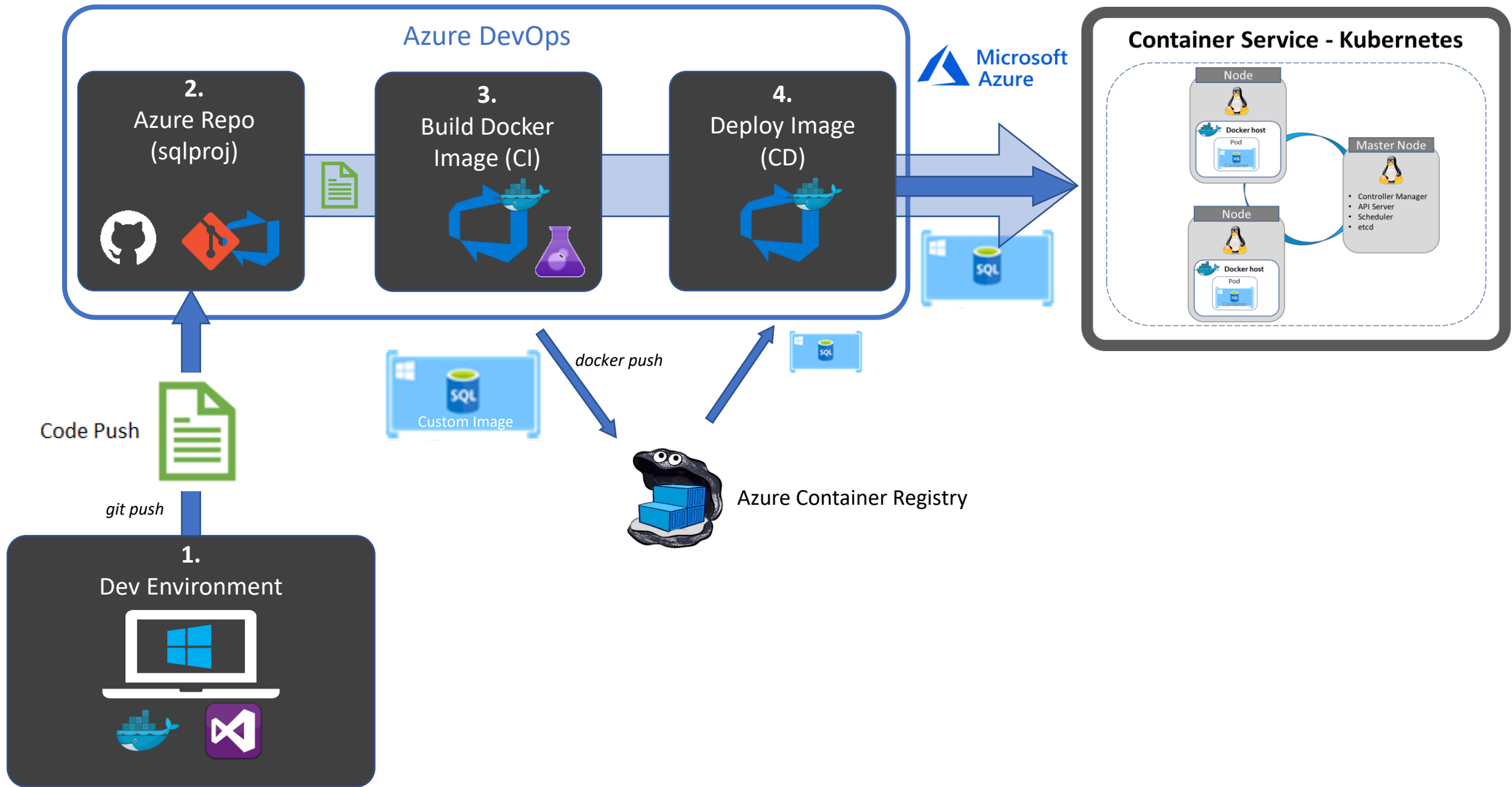
Samples scripts for a SQL Server Always On Availability Group on SQL Server Containers, managed by Kubernetes

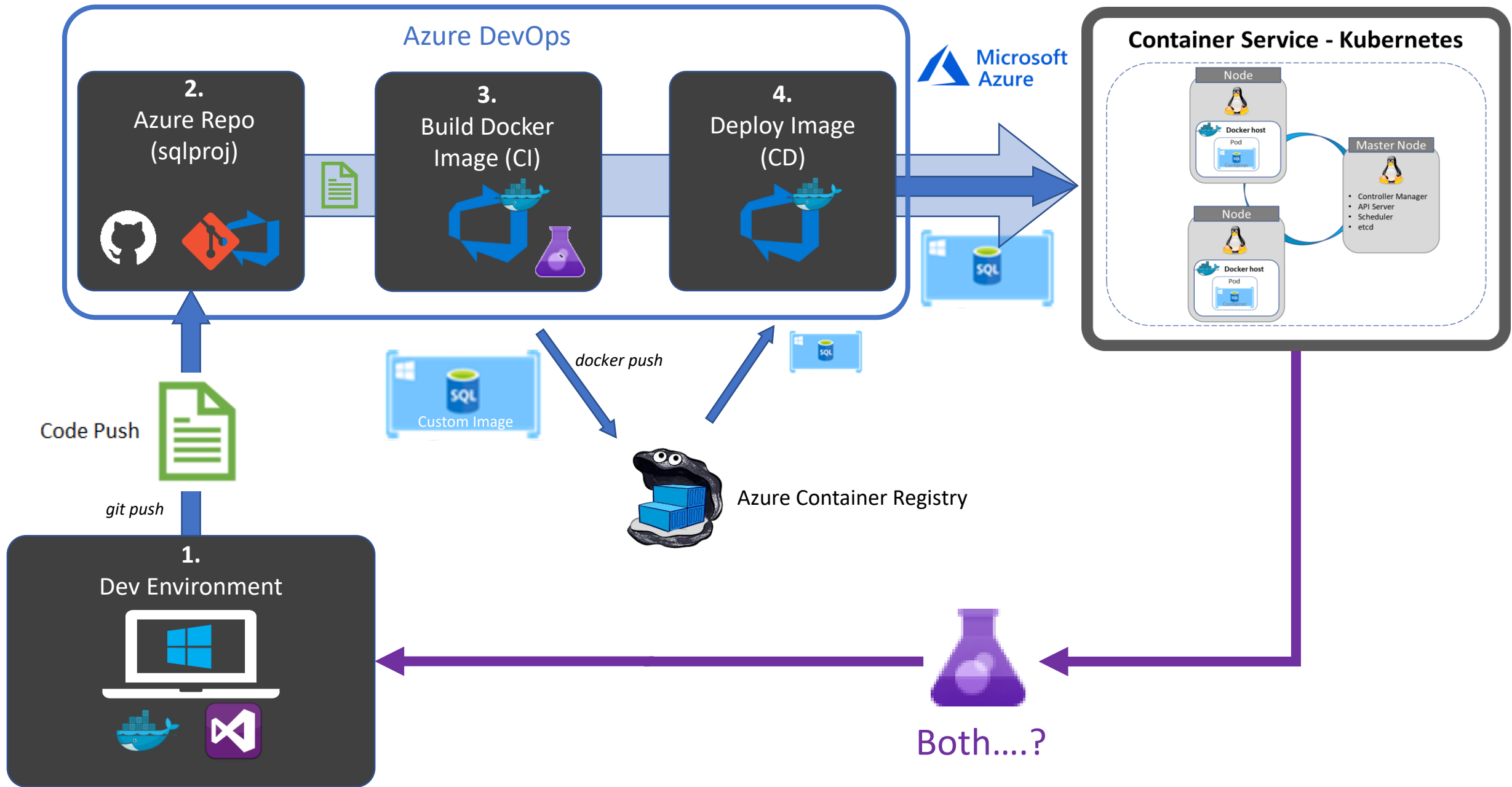
Availability Groups on SQL Server Containers was provided during SQL Server 2019 preview releases to demonstrate a potential capability. SQL Server 2019 does not support Availability Groups on containers.

DEMO










Move from Docker to K8s - kompose

```
version: '3.3'
services:
  db:
    build:
      context: ./docker/db/
      dockerfile: Dockerfile
    ports:
      - "1433:1433"
    environment:
      SA_PASSWORD: "Alaska2017"
      ACCEPT_EULA: "Y"
    healthcheck:
      test: sqlcmd -S db1.internal.prod.example.com -U SA -P [REDACTED] -Q 'select 1'
    networks:
      mynetwork:
        aliases:
          - db1.internal.prod.example.com
  web:
    build:
      context: ./docker/web/
      dockerfile: Dockerfile
    user: root
    depends_on:
      - db
    volumes:
      - ./code/
    ports:
      - "8080:8080"
    environment:
      DJANGO_SETTINGS_MODULE: SqlServerOnDocker.settings
    command: python3 manage.py runserver 0.0.0.0:8080
    restart: unless-stopped
    networks:
      mynetwork:
        aliases:
          - web1.internal.prod.example.com
networks:
  mynetwork:
    driver: bridge
```


Move from Docker to K8s - kompose

 Administrator: Windows PowerShell

```
PS C:\K8sDemo\kompose\SqlServerOnDocker> .\kompose -f docker-compose.yml convert
[33mWARN[0m Restart policy 'unless-stopped' in service web is not supported, convert it to 'always'
[33mWARN[0m Ignoring user directive. User to be specified as a UID (numeric).
[36mINFO[0m Kubernetes file "db-service.yaml" created
[36mINFO[0m Kubernetes file "web-service.yaml" created
[36mINFO[0m Kubernetes file "db-deployment.yaml" created
[36mINFO[0m Kubernetes file "web-deployment.yaml" created
PS C:\K8sDemo\kompose\SqlServerOnDocker> █
```

Move from Docker to K8s - kompose

Administrator: Windows PowerShell

```
PS C:\K8sDemo\kompose\SqlServerOnDocker> .\kompose -f docker-compose.yml convert
[33mWARN[0m Restart policy 'unless-stopped' in service web is not supported, convert it to 'always'
[33mWARN[0m Ignoring user directive. User to be specified as a UID (numeric).
[36mINFO[0m Kubernetes file "db-service.yaml" created
[36mINFO[0m Kubernetes file "web-service.yaml" created
[36mINFO[0m Kubernetes file "db-deployment.yaml" created
[36mINFO[0m Kubernetes file "web-deployment.yaml" created
PS C:\K8sDemo\kompose\SqlServerOnDocker>
```

K8s Summary

Pro's

- Runs on many platforms
- Scaling self-healing and recover quickly
- Manage infrastructure as a code
 - Hide infrastructure complexity
- Load balancers
- Speed of deployment
- Ability to absorb change quickly

Con's

- Big change from the norm!!!
- Minikube is hit n miss
 - Make sure you stop it when you shut down
- Make sure you set your resource limits
- Always read the change log before updating 😊

Upgrading Minikube / k8s ☺

```
AzureAD+ChrisTaylor@JCL-Precision15 MINGW64 /c/K8sDemo/Demo/1 - OnPrem - Minikube/Windows
$ kubectl apply -f sqlserver_persistentvolume_minikube.yaml
service/minikube-sqlserver-service unchanged
error: unable to recognize "sqlserver_persistentvolume_minikube.yaml": no matches for kind "Deployment" in version "apps/v1beta1"
```

```
AzureAD+ChrisTaylor@JCL-Precision15 MINGW64 /c/K8sDemo/Demo/1 - OnPrem - Minikube/Windows

$ kubectl apply -f sqlserver_persistentvolume_minikube.yaml
error: error validating "sqlserver_persistentvolume_minikube.yaml": error validating data
: ValidationError(Deployment.spec): missing required field "selector" in io.k8s.api.apps.
v1.DeploymentSpec; if you choose to ignore these errors, turn validation off with --valid
ate=false
```

<https://kubernetes.io/blog/2019/07/18/api-deprecations-in-1-16/>

Change Required

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: minikube-sqlserver
spec:
  replicas: 1
  template:
    metadata:
      labels:
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: minikube-sqlserver
spec:
  replicas: 1
  selector:
    matchLabels:
      app: minikube-mssql
  template:
    metadata:
      labels:
```

Summary

- Containers
- Kubernetes
 - Kubectl
 - Minikube / Docker for Windows
 - Azure Kubernetes Service (AKS)
- (Little bit) Azure DevOps Pipelines
 - Build / Release

Contact



@SQLGeordie



github.com/SQLGeordie/



chris.taylor@jarrinconsultancy.com



www.jarrinconsultancy.com/blog
www.chrisjarrintaylor.co.uk

Thank you

