Kubernetes Fundamentals for Beginners

Presented by
Ole-Martin Hafslund

Agenda

Kubernetes and components with SAS Viya4 in a Cloud environment

- **What is Kubernetes (K8s)**
- Main K8s Components
- K8s Architecture
- K8s YAML Configuration File(s)
- Organizing your components with K8s Namespaces
- SAS Viya in Kubernetes – helpful tools

What is Kubernetes

- Kubernetes or K with 8 letters before the s = K8s
- Definitions
- Problem it solves

What is Kubernetes

Official definition of kubernetes

- *"Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. ... Kubernetes services, support, and tools are widely available. The name Kubernetes originates from Greek, meaning helmsman or pilot."*

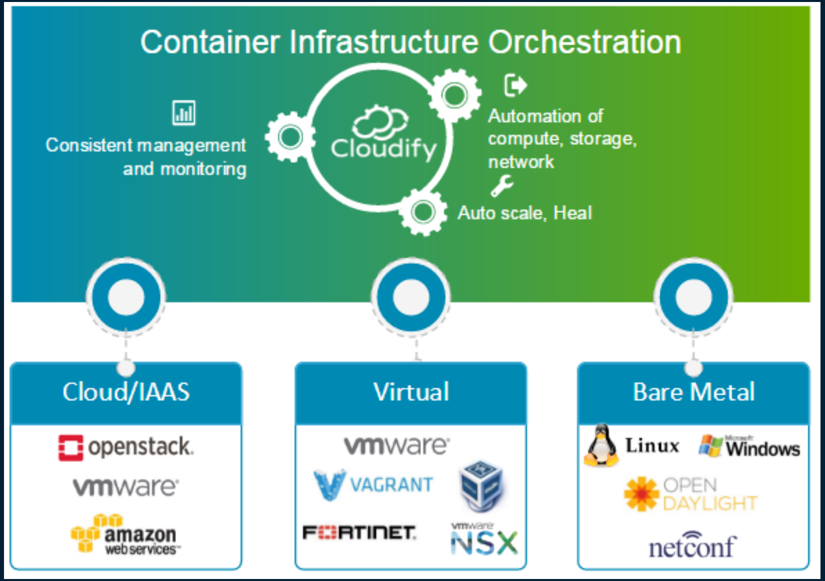
- Open Source container orchestration tool
- Developed by Google

- Its fundamental function is to handle containers

- K8s is agnostic to the type of software inside the containers
- There are many versions of Kubernetes and many vendors
- It can work with multiple Container Engines

What is Kubernetes

Container Infrastructure Orchestration

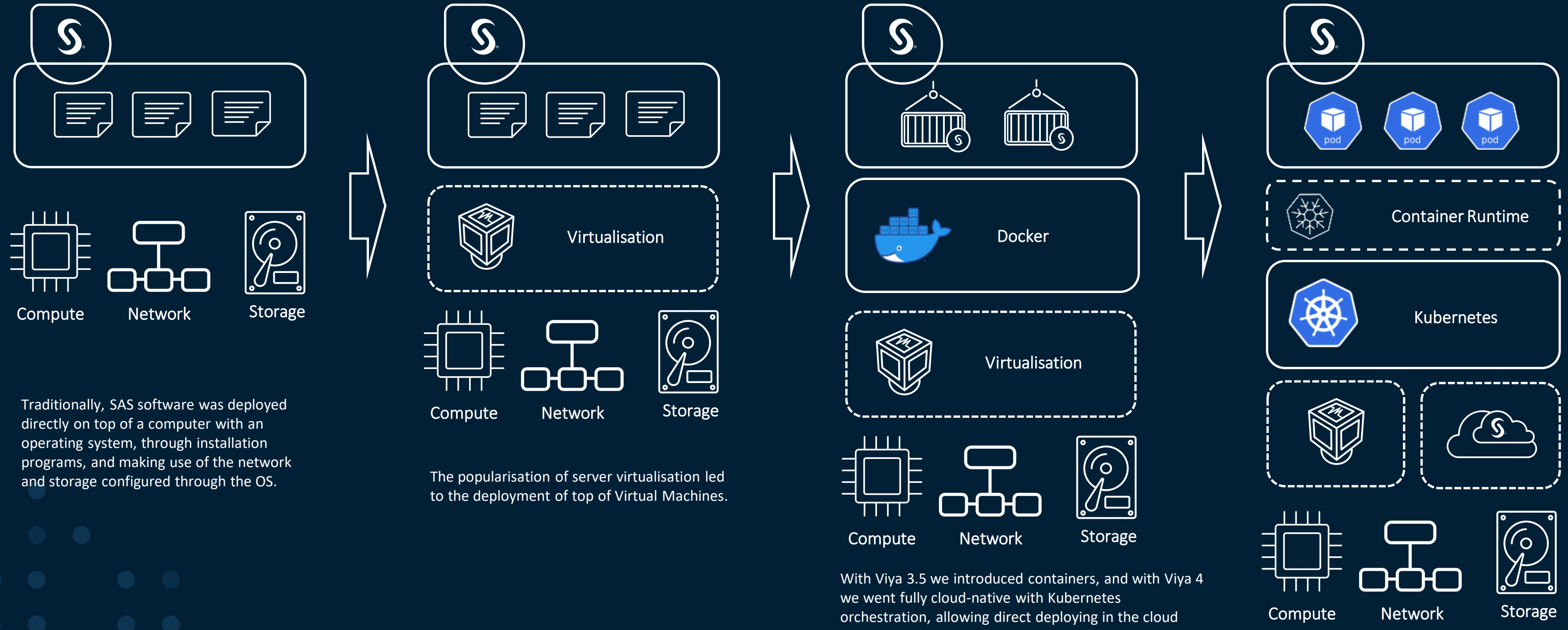


Kubernetes and containers- can never go wrong

- It’s a joke ☺

SAS Viya

Software installation and technology evolution



Traditionally, SAS software was deployed directly on top of a computer with an operating system, through installation programs, and making use of the network and storage configured through the OS.

The popularisation of server virtualisation led to the deployment of top of Virtual Machines.

With Viya 3.5 we introduced containers, and with Viya 4 we went fully cloud-native with Kubernetes orchestration, allowing direct deploying in the cloud and on-premises.

What is Kubernetes

How do we handle thousands of containers and run them

- Kubernetes is the pilot that handles the docker containers
- Kubernetes steers the «whale»

- There has been and is a trend going from gigantic Monolitic systems -> Microservices systems concisting of containers to control them.

- Kubernetes can handle things that scripts and manual labour would not cope up with.

- This demands a more proper way to manage thousands of containers across large enterprise systems

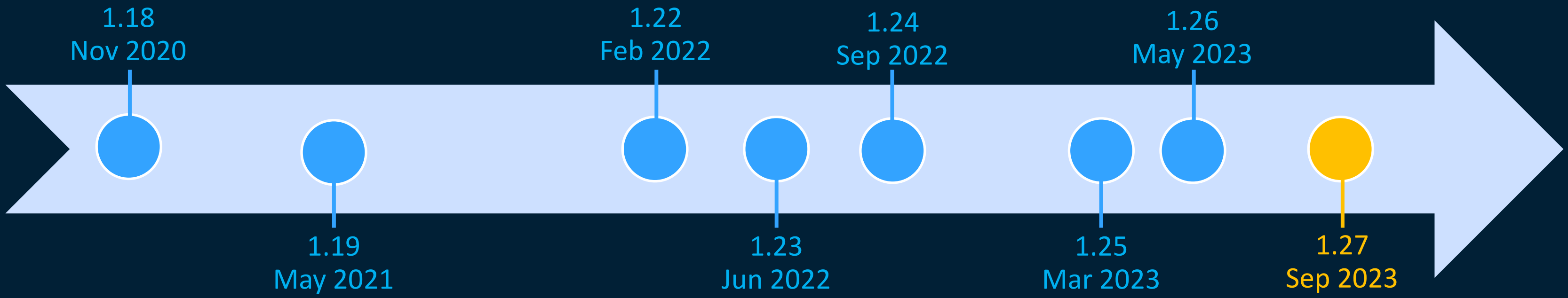
Kubernetes Cluster

- Kubernetes is a cluster technology
- A Kubernetes cluster is an instance of Kubernetes
- At Kubernetes v1.18 a cluster can have between 1 and 5000 Nodes
  - No more than 5,000 nodes
  - No more than 150,000 pods
  - No more than 300,000 containers
- You do not deploy an application to a specific computer (node)
- Kubernetes will determine the computer(s) that best matches the requirements of your application
  - Kubernetes will decide where to run the application based on a set of rules, or configuration

SAS Viya Kubernetes Support Timeline

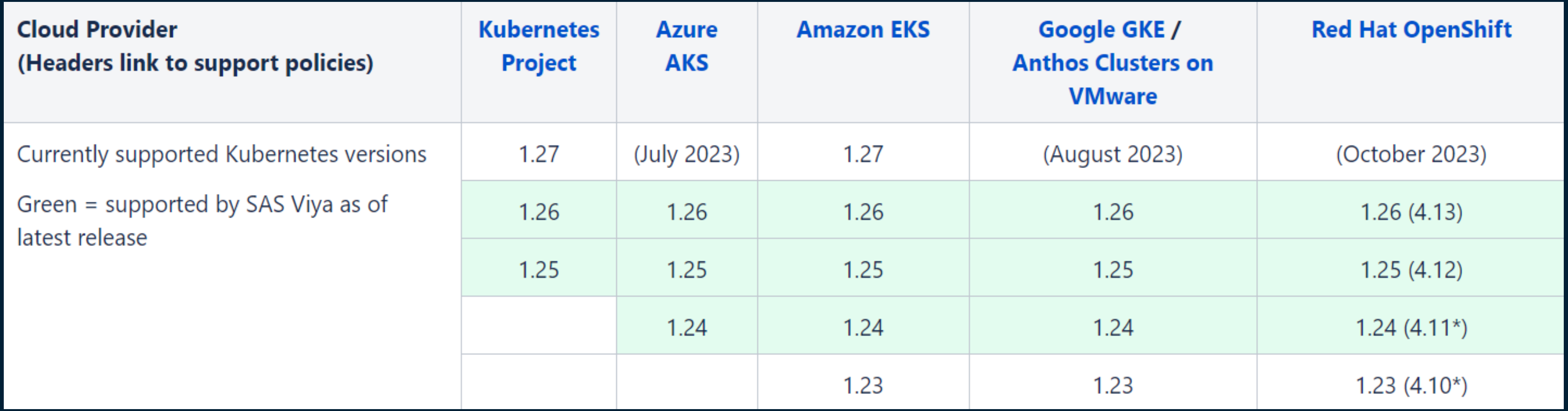
SAS Viya Kubernetes Support Timeline

History of SAS Viya Kubernetes Support



SAS Viya Kubernetes Support Levels

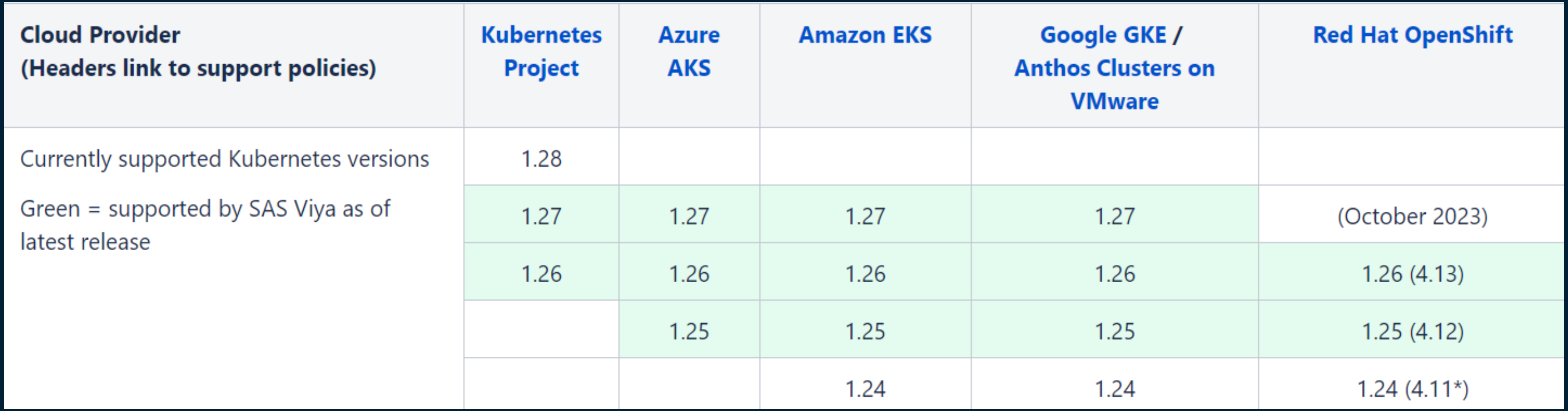
SAS Viya 2023.07 Stable



Cloud Provider (Headers link to support policies)	**Kubernetes Project**	**Azure AKS**	**Amazon EKS**	**Google GKE / Anthos Clusters on VMware**	**Red Hat OpenShift**
Currently supported Kubernetes versions	1.27	(July 2023)	1.27	(August 2023)	(October 2023)
Green = supported by SAS Viya as of latest release	1.26	1.26	1.26	1.26	1.26 (4.13)
	1.25	1.25	1.25	1.25	1.25 (4.12)
		1.24	1.24	1.24	1.24 (4.11*)
			1.23	1.23	1.23 (4.10*)

SAS Viya Kubernetes Support Levels

SAS Viya 2023.09 Stable (Planned)



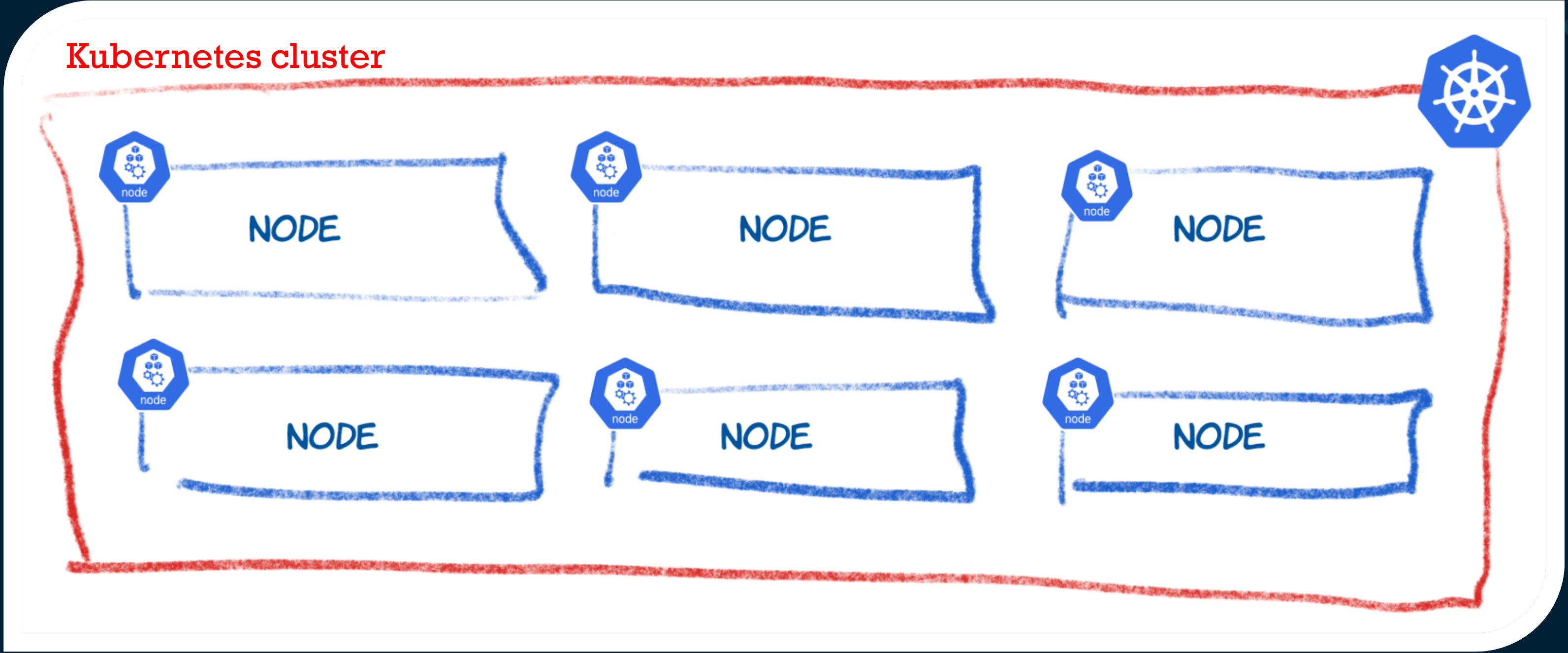
Cloud Provider (Headers link to support policies)	Kubernetes Project	Azure AKS	Amazon EKS	Google GKE / Anthos Clusters on VMware	Red Hat OpenShift
Currently supported Kubernetes versions	1.28				
Green = supported by SAS Viya as of latest release	1.27	1.27	1.27	1.27	(October 2023)
	1.26	1.26	1.26	1.26	1.26 (4.13)
		1.25	1.25	1.25	1.25 (4.12)
			1.24	1.24	1.24 (4.11*)

SAS Viya Kubernetes Support Timeline

Planning

- For SAS Viya, we target three Kubernetes version updates per year
  - Matches the cadence of the broader Kubernetes community timeline
  - Tentatively target the January, May, and September stable releases
- Timing is also influenced by key partner timelines
  - Especially Azure AKS and Red Hat OpenShift
  - But need to track all SAS Viya Kubernetes deployment targets

Kubernetes Cluster

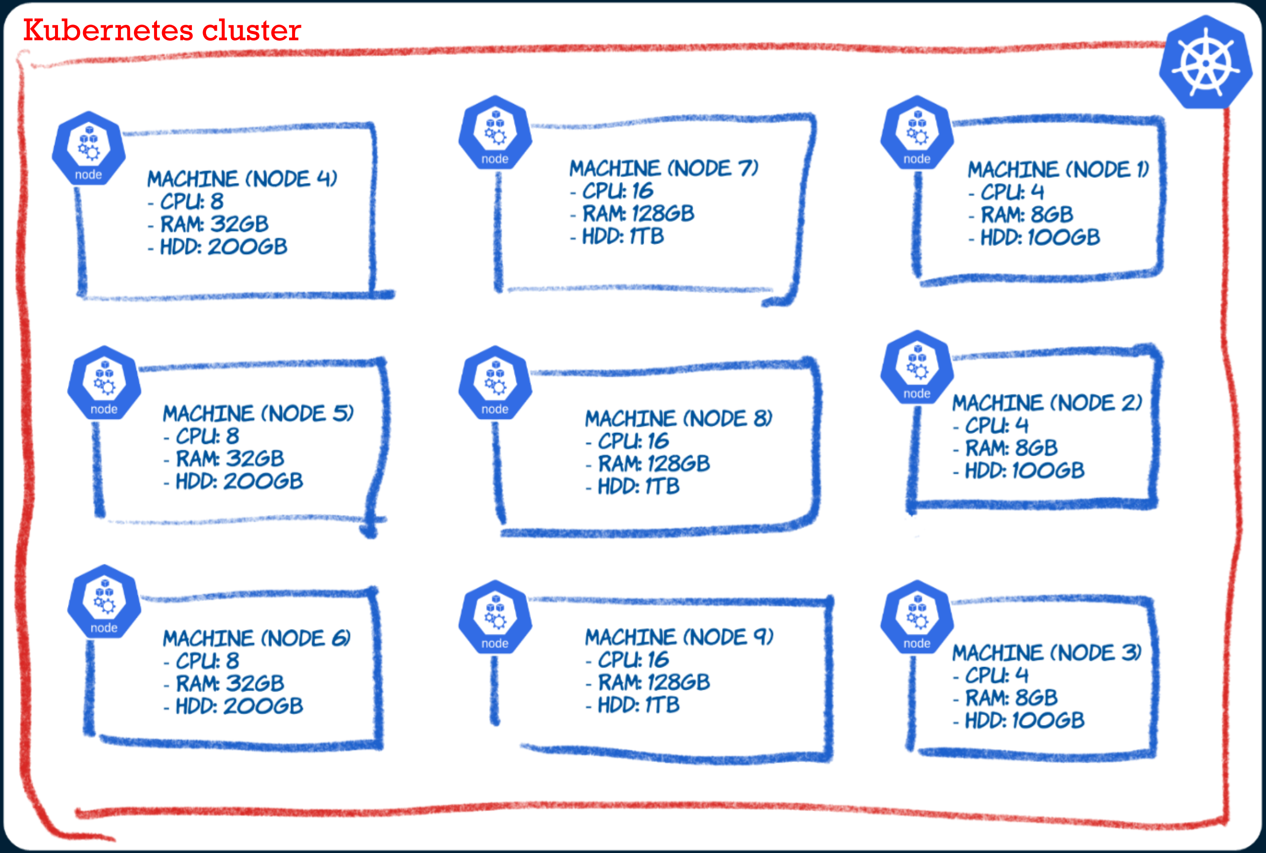


- A **Kubernetes cluster** is a set of machines (computers) for running containerized applications
- A cluster is made up of machines that are called Nodes

Kubernetes Nodes

- Each computer in the cluster is called a node
  - Nodes can be a physical or virtual instance of a server
  - Nodes can vary in size
- The nodes will host your applications, running as containers
- Nodes can be independently started and stopped
- The nodes could be spread across different data centers, but Kubernetes cannot change the Laws of Physics!
  - Kubernetes cannot increase bandwidth or lower network latency
  - So let's not get crazy with far flung Kubernetes clusters yet!
  - Remember the cluster is only as good as the underlying infrastructure

Kubernetes Nodes



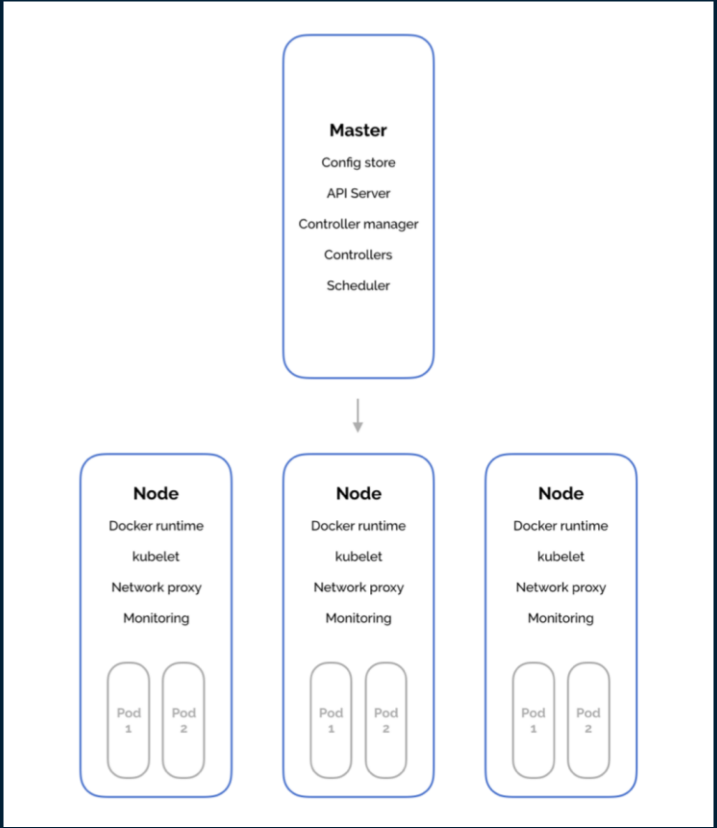
- Each computer (machine) in the cluster is called a ‘node’
- Nodes can vary in size (capacity)
- Nodes can be independently started and stopped

Kube-systems running K8s

- One of these machines is the master
- The rest are the worker nodes.

What is Kubernetes

How do we handle thousands of containers and run them



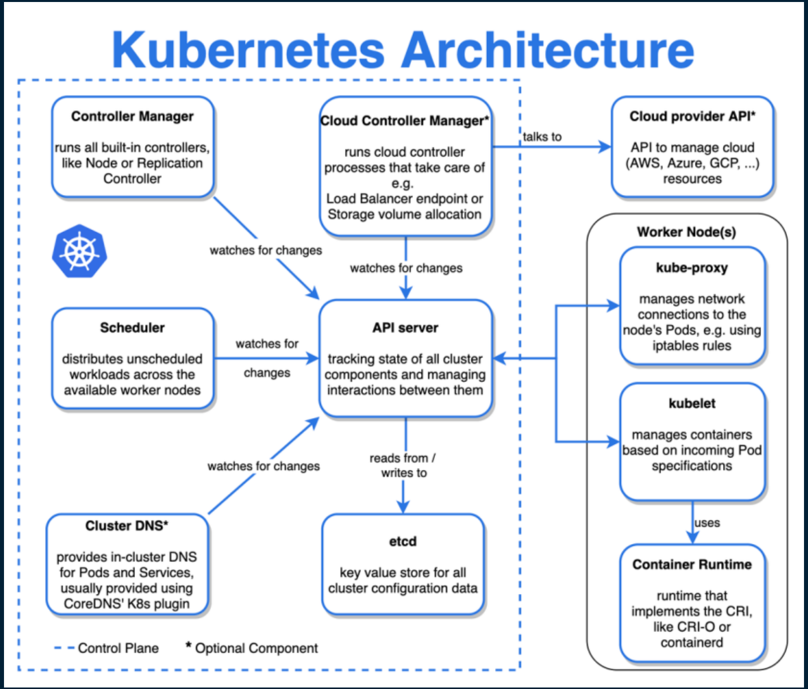
What is Kubernetes

How do we handle thousands of containers and run them

- What features do K8s as an orchestration tool offer?

1. High Availability – no downtime
2. Scalability – high performance
3. Disaster recovery – backup and restore of the applications.

- Container orchestration technology as K8s offers this


Kubernetes Architecture
Controller Manager
runs all built-in controllers, like Node or Replication Controller
Cloud Controller Manager*
runs cloud controller processes that take care of e.g. Load Balancer endpoint or Storage volume allocation
talks to
Cloud provider API*
API to manage cloud (AWS, Azure, GCP, ...) resources
watches for changes
watches for changes
Worker Node(s)
kube-proxy
manages network connections to the node's Pods, e.g. using iptables rules
Scheduler
distributes unscheduled workloads across the available worker nodes
watches for changes
API server
tracking state of all cluster components and managing interactions between them
kubelet
manages containers based on incoming Pod specifications
watches for changes
reads from / writes to
uses
Cluster DNS*
provides in-cluster DNS for Pods and Services, usually provided using CoreDNS' K8s plugin
etcd
key value store for all cluster configuration data
Container Runtime
runtime that implements the CRI, like CRI-O or containerd
Control Plane
* Optional Component

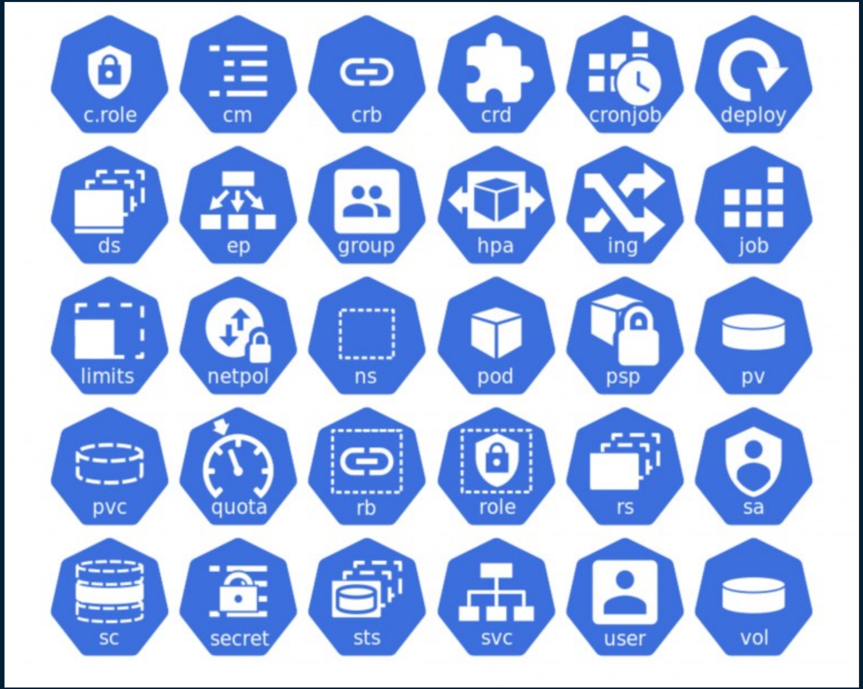
Agenda

Kubernetes and components with SAS Viya4 in a Cloud environment

- What is Kubernetes (K8s)
- **Main K8s Components**
- K8s Architecture
- K8s YAML Configuration File(s)
- Organizing your components with K8s Namespaces
- SAS Viya in Kubernetes

Main K8s Components

Most normal components - interaction



Will only look at a handful of these components that are in the picture to the left.

Like:
Pod
Deployment
Namespace
Storageclass
Persistent volume
Persistent Volume claim
Services
Secret
Ingress ...

Main K8s Components

Let’s take a look at these

Volumes

Secrets

ConfigMap

Pod

Ingress

Service

Deployment

Stateless

SatefulSet

Main K8s Components

Case – build web application with database

Node and Pod

Kubernetes Pods

- A Pod is the smallest deployable compute object in Kubernetes
- A Pod encapsulates an application's container and related storage resources
- A Pod can hold multiple containers, but usually contain one container
- Pods are hosted on the Nodes; Kubernetes will determine which Node will host a Pod
- A Pod is assigned a unique network identity (IP address), which is independent of the node on which the pod is running

Kubernetes Pod Security

Why it matters

- Customers requires workloads to run without elevated permissions
- Running unsecured workloads is asking for trouble
  – It exposes the underlying infrastructure and puts other workloads at risk
- Impacts a SAS Viya deployment
  – Requires an understanding of Kubernetes Pod Security
  – Specific configurations of SAS Viya are impacted
- Feel more confident to engage with customers that are applying pod security

Two flavors you may encounter in the field

Pod Security Policies	Pod Security Standards
• Introduced by the community as of Kubernetes version 1.10 • PSP for short • Removed as of Kubernetes version 1.25	• Introduced as of Kubernetes version 1.22 • PSS for short • Official successor to PSP as of Kubernetes version 1.25

Does not include SCCs from RHEL OCP

Kubernetes Pods

A Pod can hold multiple containers

- If 2 (or more) containers are tightly connected (coupled), it can be useful to co-locate the containers in a single pod, as they will "share" more things
- It also guarantees that the containers are running together on the same Node
- Two common patterns
  - Sidecar pattern
    - The two containers are tightly coupled, and the sidecar container adds function to the main service
    - The sidecar pattern is used to implement SSSD in the CAS pods
  - Init containers
    - Init containers run before the application container is started
    - The Init container pattern is used for the certificate framework (sas-certframe), which is responsible for setting up the required TLS objects ready for the main container

Main K8s Components

Node, Pod and container

node

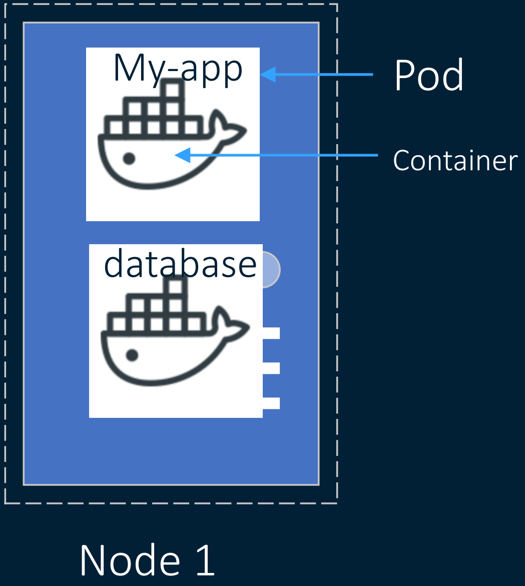
Node=
Server

pod

Pod =
Smallest unit in K8s

Abstraction over a container

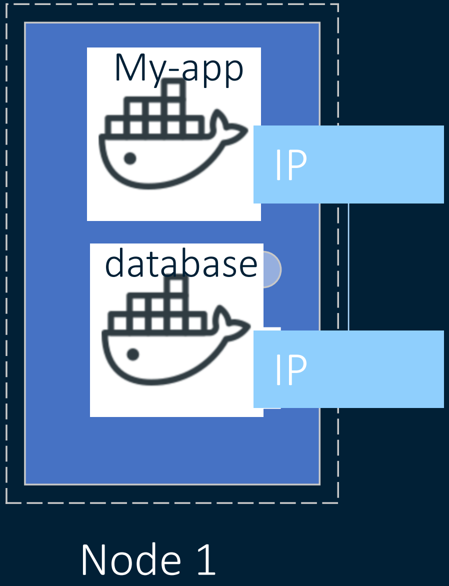
Running environment/
layer on top of the Container



Note: You only interact with kubernetes – not the Containers

Main K8s Components

Node, Pod and container



- Smallest unit of K8s = Pod
- Pod = Abstraction over a Container
- Normally 1 application per Pod
- Each pod gets its own IP address
- New IP address on re-creation of the pod

Note: You only interact with kubernetes – not the Containers

Kubernetes Service

- A Kubernetes Service is an abstraction which defines a logical set of Pods and a policy by which to access them

But why do I need a service?

- Pods are transient. They start, they die, get restarted, they can be randomly assigned to a Kubernetes node
- Pods don’t move from node to node, they are replaced (stopped & restarted)
- There may be more than one replica of a Pod
- So, we need a “static” way of accessing a Pod or group of Pods
- The **service** provides this mapping

Kubernetes Ingress & Ingress Controllers

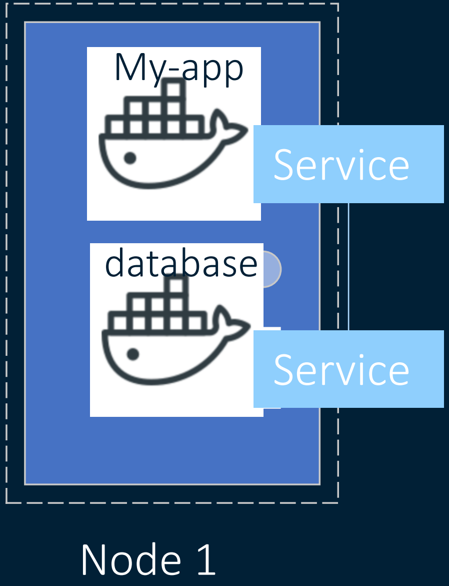
- Services support the communication from within the Cluster
- However, we would like to expose the services outside of the cluster

But how do I get to my application?

- An Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster
- Traffic routing is controlled by rules defined on the **Ingress** resource
- An **Ingress Controller** is responsible for fulfilling the Ingress
- Commonly used Ingress Controllers include
  - Traefix, NGINX, Istio, AWS ELB…

Main K8s Components

Service and Ingress

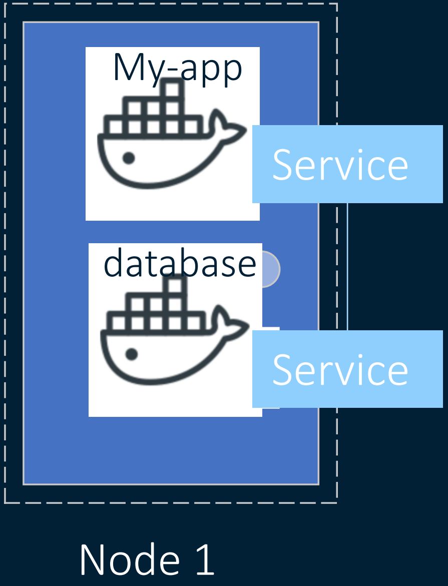


- Service = permanent static IP address
- Lifecycle of the Pod and the Service are not connected
- Service will keep its IP even if Pod dies

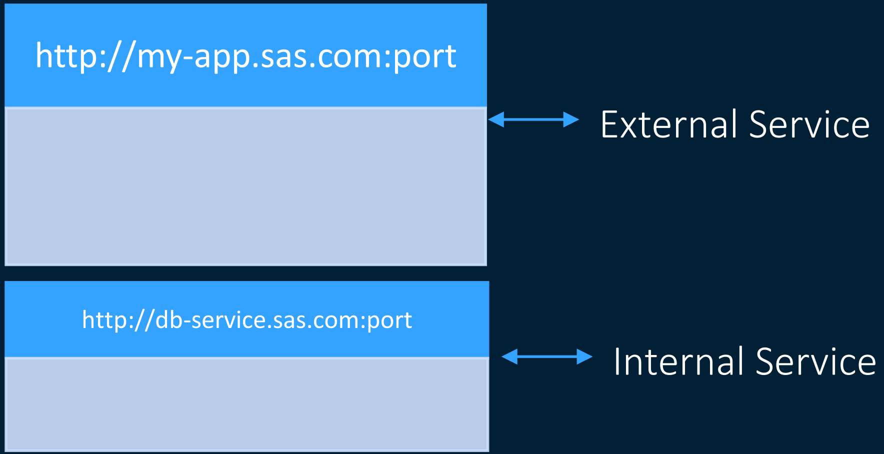
Note: You only interact with kubernetes – not the Containers

Main K8s Components

Service and Ingress



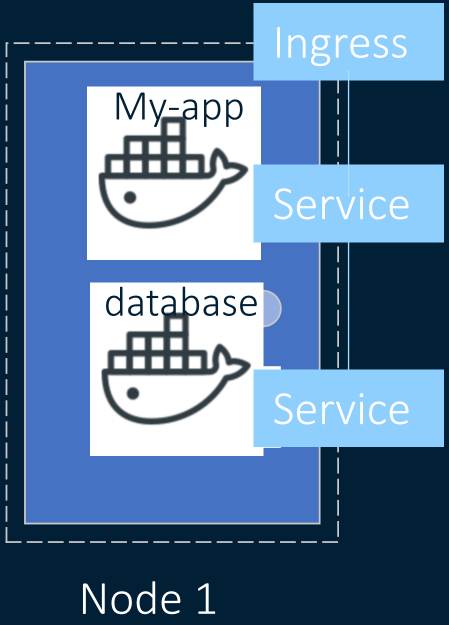
- App should be accessible from a browser – how?



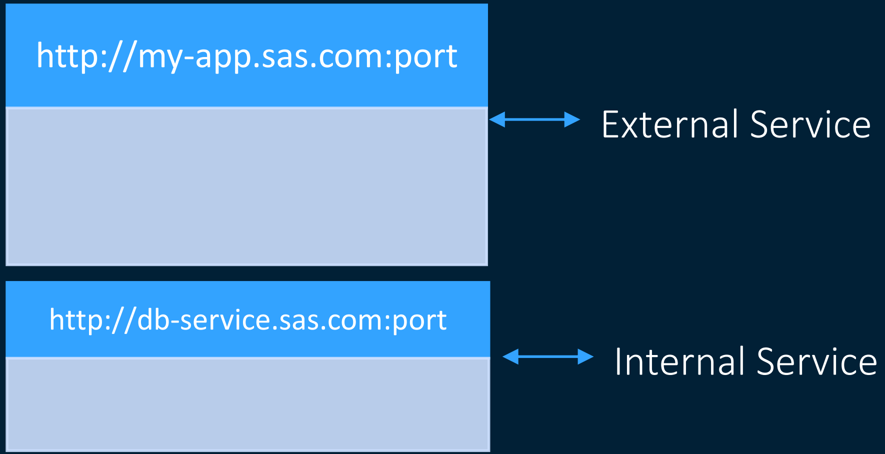
Note: You only interact with kubernetes – not the Containers

Main K8s Components

Service and Ingress



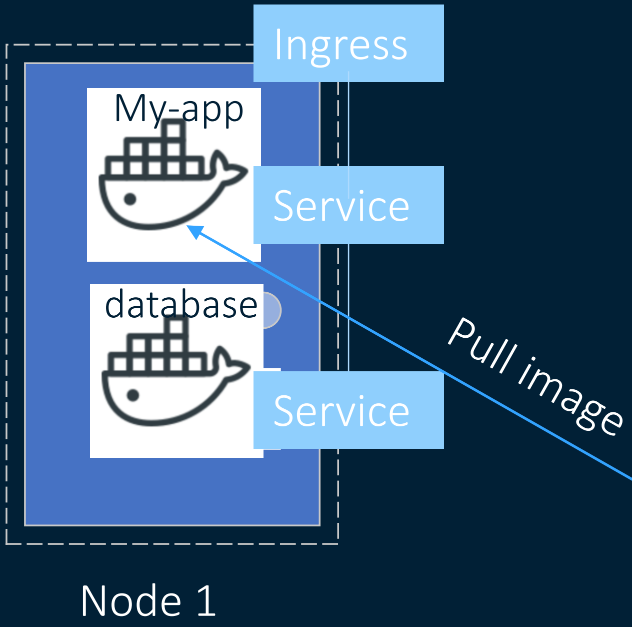
- App should be accessible from a browser – how?



Note: You only interact with kubernetes – not the Containers

Main K8s Components

Service and Ingress



Database URL usually registered in the built application

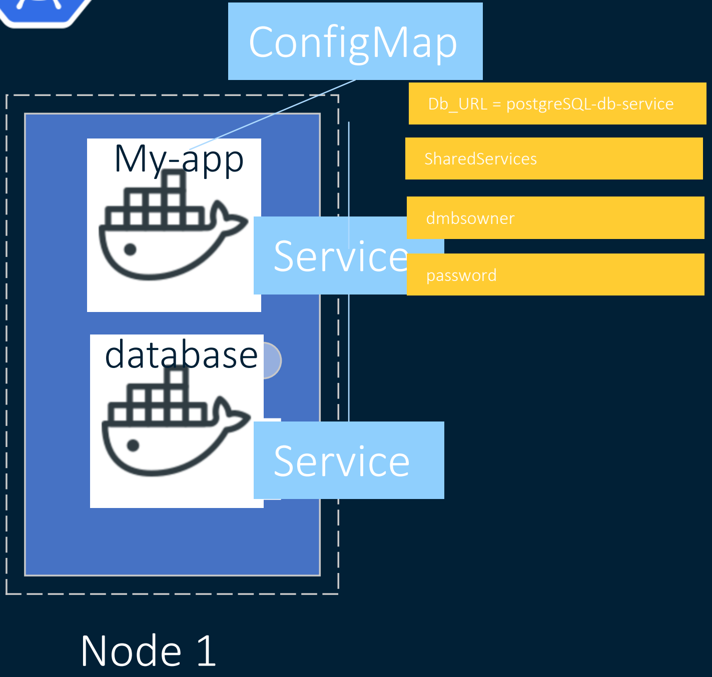
Re-build application

Push it to repository

Note: You only interact with kubernetes – not the Containers

Main K8s Components

ConfigMap



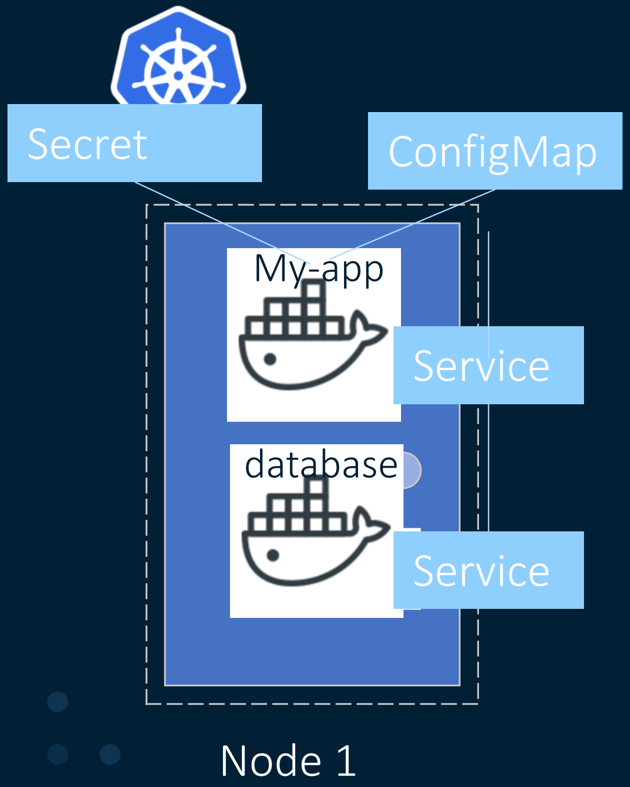
ConfigMap:

External configuration to your application

Note: You only interact with kubernetes – not the Containers

Main K8s Components

Secrets



Secret:

Used to store Secret data → Base64 encoded

Contains credentials

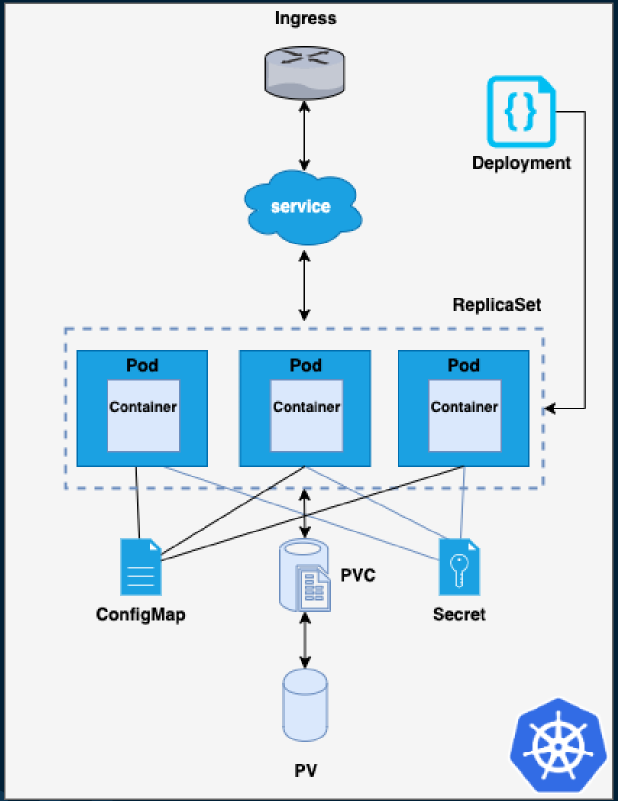
Certificates

Things you don't want other to view

Note: The built-in security mechanism is not enabled by default

Main K8s Components

Recap



- ✓ Pod
- ✓ Services
- ✓ Ingress
- ✓ ConfigMap
- ✓ Secrets

Agenda

Kubernetes and components with SAS Viya4 in a Cloud environment

- What is Kubernetes (K8s)
- Main K8s Components
- **K8s Architecture**
- K8s YAML Configuration File(s)
- Organizing your components with K8s Namespaces
- SAS Viya in Kubernetes – helpful tools

K8s Architecture

Volumes

Kubernetes Volumes

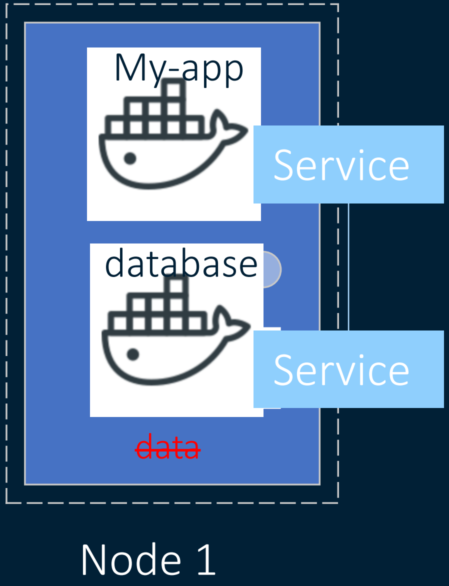
The problem: On-disk files in a Container are ephemeral

So how do you make use of different storage types and persist data?

- The Kubernetes Volume abstraction allows data to persist and/or be shared between Containers in a Pod or shared across Pods
- A Kubernetes volume is just a directory (possibly with some data in it) which is accessible to the Container(s) in a Pod
- Kubernetes supports many types of volumes, and a Pod can use any number of them simultaneously
- The lifetime of the volume depends on its type
- The Volume types include (and more):
  - local, hostPath, emptyDir
  - nfs, persistentVolumeClaim

K8s Architecture

Volumes

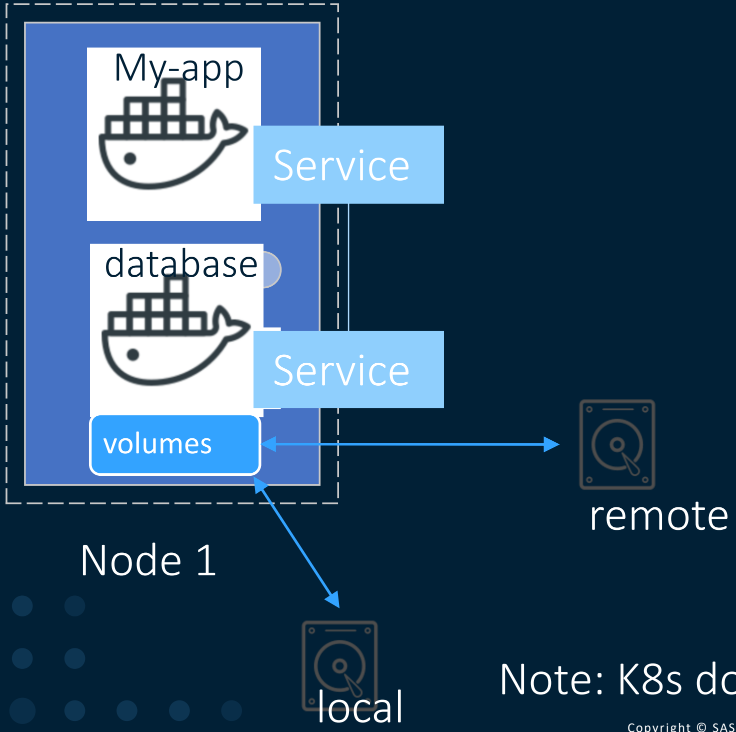


Data Storage:

Data stored inside the container is lost when restarting the pod

K8s Architecture

Volumes



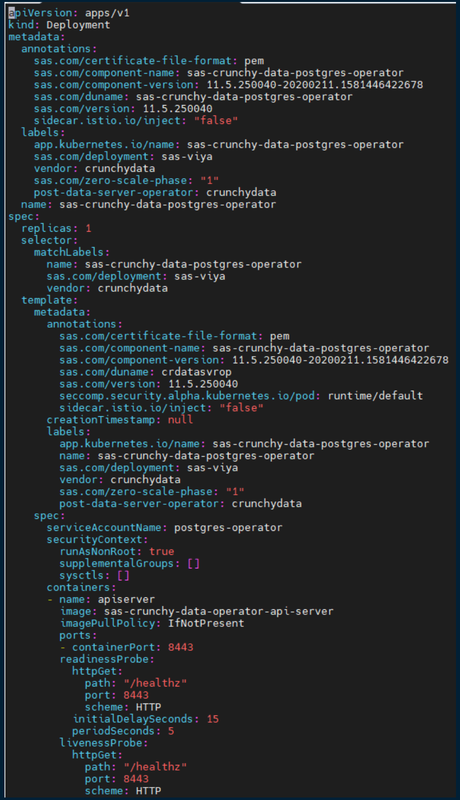
Volumes:

Storage on local machine or remote, outside of the K8s cluster

Note: K8s doesn't manage data persistence!

K8s Architecture

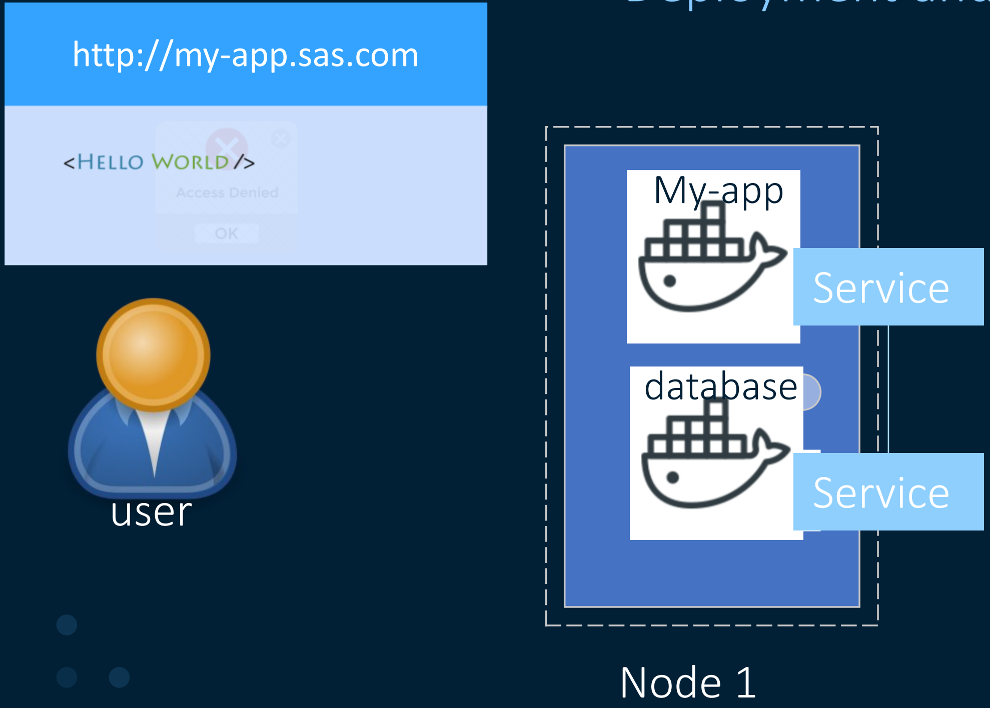
Deployment and Stateful Set



```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    sas.com/certificate-file-format: pem
    sas.com/component-name: sas-crunchy-data-postgres-operator
    sas.com/component-version: 11.5.250040-20200211.1581446422678
    sas.com/duname: sas-crunchy-data-postgres-operator
    sas.com/version: 11.5.250040
    sidecar.istio.io/inject: "false"
  labels:
    app.kubernetes.io/name: sas-crunchy-data-postgres-operator
    sas.com/deployment: sas-viya
    vendor: crunchydata
    sas.com/zero-scale-phase: "1"
    post-data-server-operator: crunchydata
  name: sas-crunchy-data-postgres-operator
spec:
  replicas: 1
  selector:
    matchLabels:
      name: sas-crunchy-data-postgres-operator
      sas.com/deployment: sas-viya
      vendor: crunchydata
  template:
    metadata:
      annotations:
        sas.com/certificate-file-format: pem
        sas.com/component-name: sas-crunchy-data-postgres-operator
        sas.com/component-version: 11.5.250040-20200211.1581446422678
        sas.com/duname: crdatasvrop
        sas.com/version: 11.5.250040
        seccomp.security.alpha.kubernetes.io/pod: runtime/default
        sidecar.istio.io/inject: "false"
      creationTimestamp: null
      labels:
        app.kubernetes.io/name: sas-crunchy-data-postgres-operator
        name: sas-crunchy-data-postgres-operator
        sas.com/deployment: sas-viya
        vendor: crunchydata
        sas.com/zero-scale-phase: "1"
        post-data-server-operator: crunchydata
    spec:
      serviceAccountName: postgres-operator
      securityContext:
        runAsNonRoot: true
        supplementalGroups: []
        sysctls: []
      containers:
      - name: apiserver
        image: sas-crunchy-data-operator-api-server
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8443
        readinessProbe:
          httpGet:
            path: "/healthz"
            port: 8443
            scheme: HTTP
          initialDelaySeconds: 15
          periodSeconds: 5
        livenessProbe:
          httpGet:
            path: "/healthz"
            port: 8443
            scheme: HTTP
```

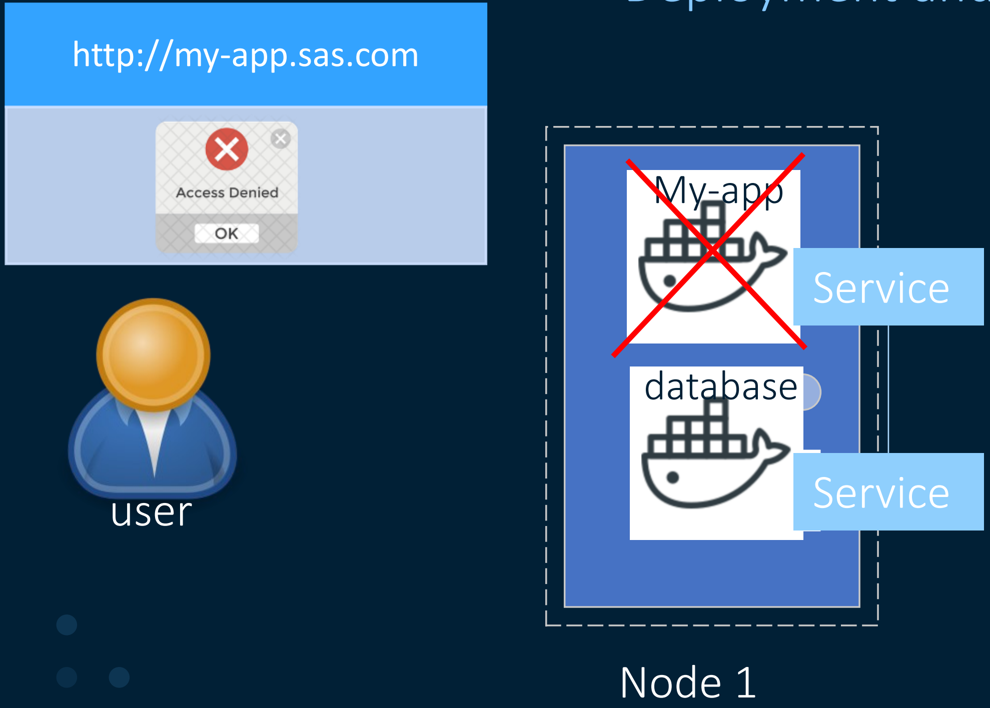
K8s Architecture

Deployment and Stateful Set



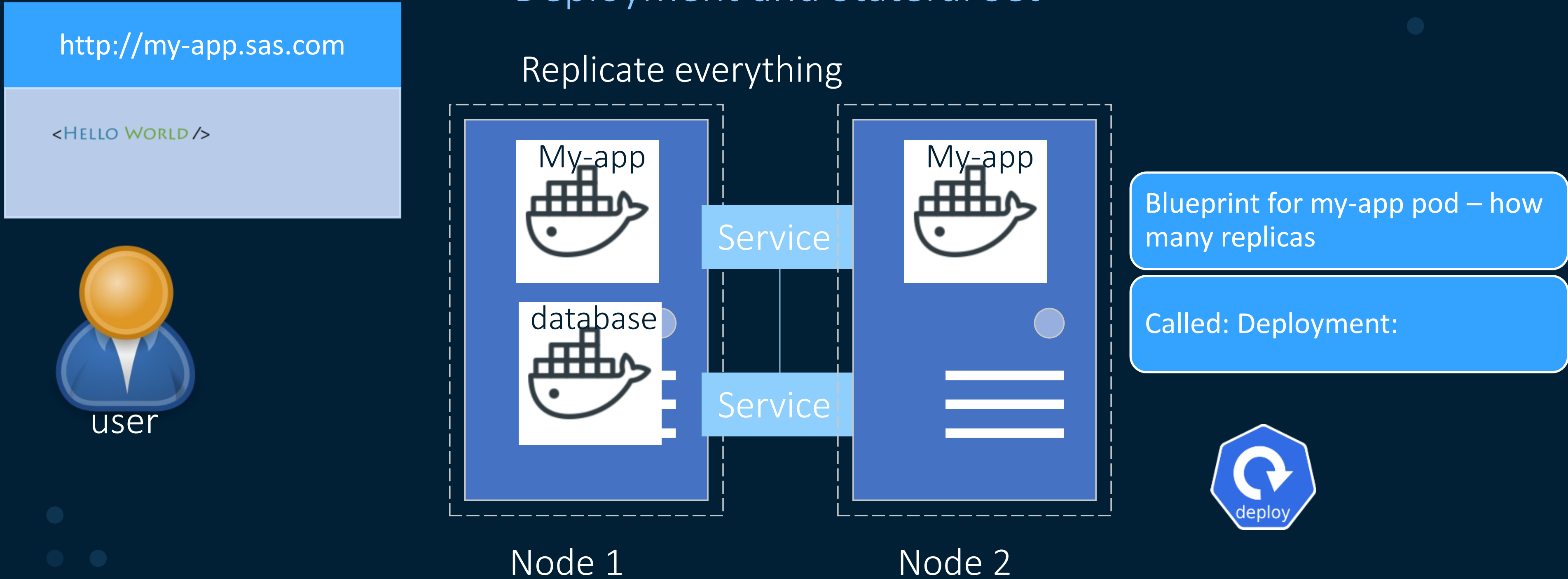
K8s Architecture

Deployment and Stateful Set



K8s Architecture

Deployment and Stateful Set

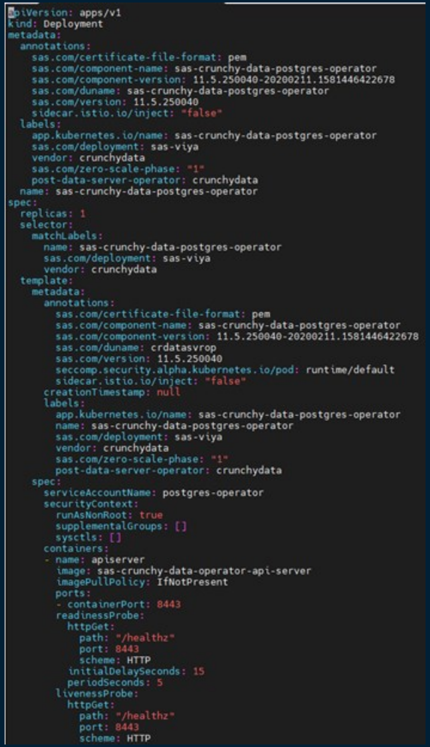


Blueprint for my-app pod – how many replicas

Called: Deployment:

K8s Architecture

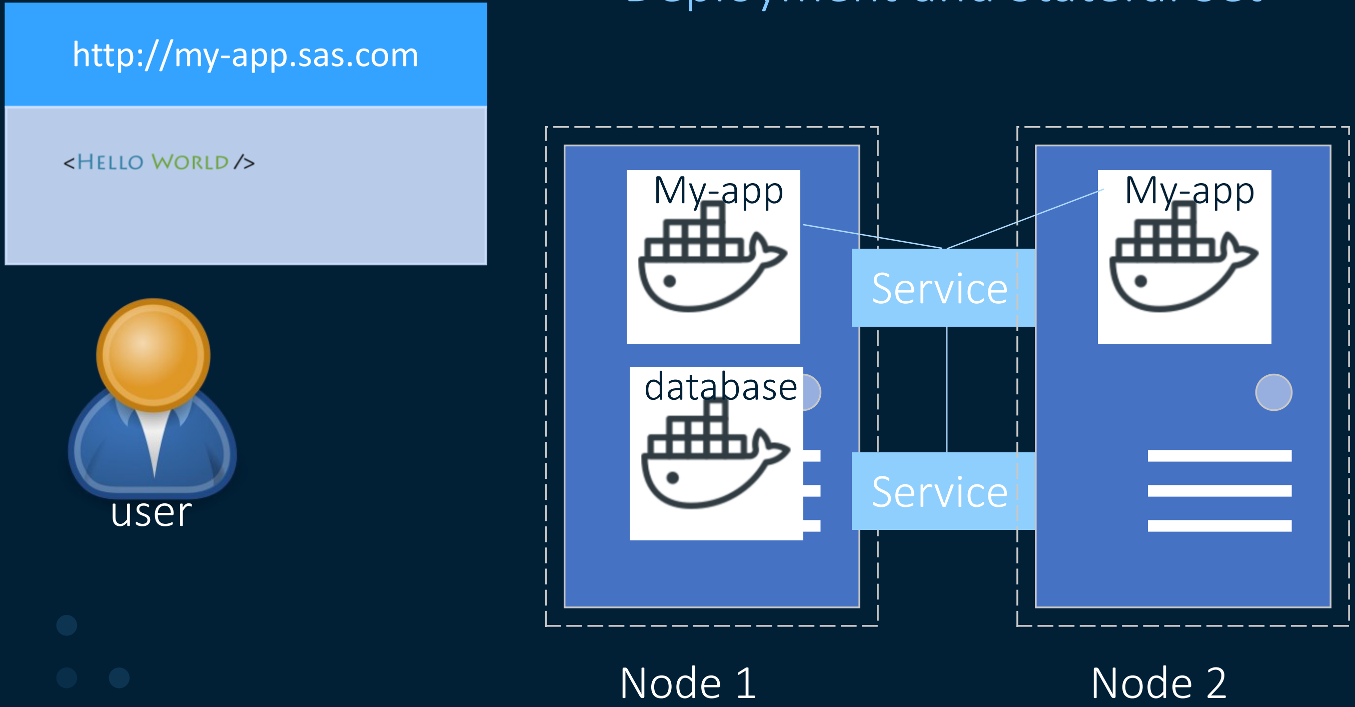
Deployment and Stateful Set



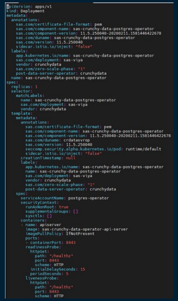
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    sas.com/certificate-file-format: pem
    sas.com/component-name: sas-crunchy-data-postgres-operator
    sas.com/component-version: 11.5.250040-20200211.1581446422678
    sas.com/duname: sas-crunchy-data-postgres-operator
    sas.com/version: 11.5.250040
    sidecar.istio.io/inject: "false"
  labels:
    app.kubernetes.io/name: sas-crunchy-data-postgres-operator
    sas.com/deployment: sas-viya
    vendor: crunchydata
    sas.com/zero-scale-phase: "1"
    post-data-server-operator: crunchydata
  name: sas-crunchy-data-postgres-operator
spec:
  replicas: 1
  selector:
    matchLabels:
      name: sas-crunchy-data-postgres-operator
      sas.com/deployment: sas-viya
      vendor: crunchydata
  template:
    metadata:
      annotations:
        sas.com/certificate-file-format: pem
        sas.com/component-name: sas-crunchy-data-postgres-operator
        sas.com/component-version: 11.5.250040-20200211.1581446422678
        sas.com/duname: crdatasvrop
        sas.com/version: 11.5.250040
        seccomp.security.alpha.kubernetes.io/pod: runtime/default
        sidecar.istio.io/inject: "false"
      creationTimestamp: null
      labels:
        app.kubernetes.io/name: sas-crunchy-data-postgres-operator
        name: sas-crunchy-data-postgres-operator
        sas.com/deployment: sas-viya
        vendor: crunchydata
        sas.com/zero-scale-phase: "1"
        post-data-server-operator: crunchydata
    spec:
      serviceAccountName: postgres-operator
      securityContext:
        runAsNonRoot: true
        supplementalGroups: []
        sysctls: []
      containers:
      - name: apiserver
        image: sas-crunchy-data-operator-api-server
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8443
        readinessProbe:
          httpGet:
            path: "/healthz"
            port: 8443
            scheme: HTTP
          initialDelaySeconds: 15
          periodSeconds: 5
        livenessProbe:
          httpGet:
            path: "/healthz"
            port: 8443
            scheme: HTTP
```

K8s Architecture

Deployment and Stateful Set



- Create Deployments
- Control the scaling
- Control the replica count

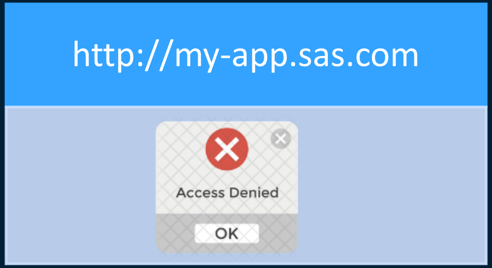


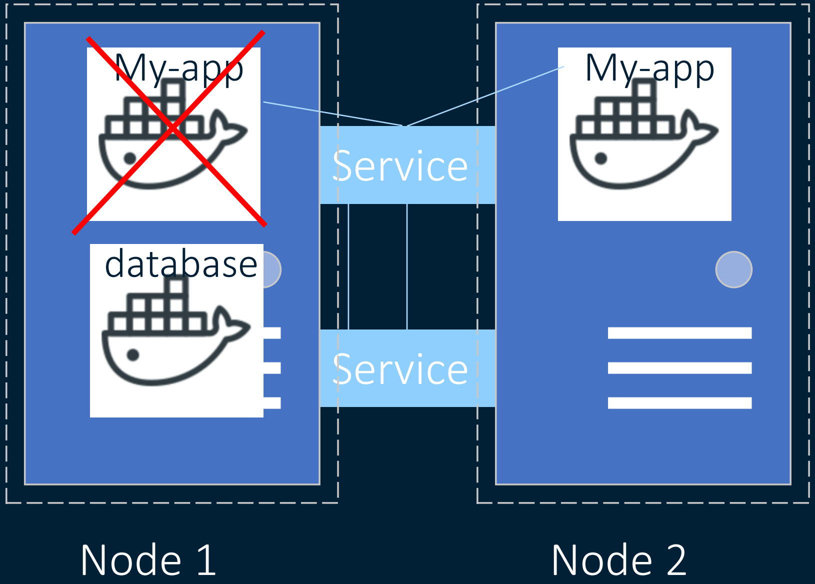
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  annotations:
    sas.com/certificate-file-format: pem
    sas.com/component-name: sas-crunchy-data-postgres-operator
    sas.com/component-version: 11.5.250040-20200211.1581446422678
    sas.com/duname: sas-crunchy-data-postgres-operator
    sas.com/version: 11.5.250040
    sidecar.istio.io/inject: "false"
  labels:
    app.kubernetes.io/name: sas-crunchy-data-postgres-operator
    sas.com/deployment: sas-viya
    vendor: crunchydata
    sas.com/zero-scale-phase: "1"
    post-data-server-operator: crunchydata
  name: sas-crunchy-data-postgres-operator
spec:
  replicas: 1
  selector:
    matchLabels:
      name: sas-crunchy-data-postgres-operator
      sas.com/deployment: sas-viya
      vendor: crunchydata
  template:
    metadata:
      annotations:
        sas.com/certificate-file-format: pem
        sas.com/component-name: sas-crunchy-data-postgres-operator
        sas.com/component-version: 11.5.250040-20200211.1581446422678
        sas.com/duname: crdatasvrop
        sas.com/version: 11.5.250040
        seccomp.security.alpha.kubernetes.io/pod: runtime/default
        sidecar.istio.io/inject: "false"
      creationTimestamp: null
      labels:
        app.kubernetes.io/name: sas-crunchy-data-postgres-operator
        name: sas-crunchy-data-postgres-operator
        sas.com/deployment: sas-viya
        vendor: crunchydata
        sas.com/zero-scale-phase: "1"
        post-data-server-operator: crunchydata
    spec:
      serviceAccountName: postgres-operator
      securityContext:
        runAsNonRoot: true
        supplementalGroups: []
        sysctls: []
      containers:
      - name: apiserver
        image: sas-crunchy-data-operator-api-server
        imagePullPolicy: IfNotPresent
        ports:
        - containerPort: 8443
        readinessProbe:
          httpGet:
            path: "/healthz"
            port: 8443
            scheme: HTTP
          initialDelaySeconds: 15
          periodSeconds: 5
        livenessProbe:
          httpGet:
            path: "/healthz"
            port: 8443
            scheme: HTTP
```

Note: You will mostly work with Deployments and not Pods

K8s Architecture

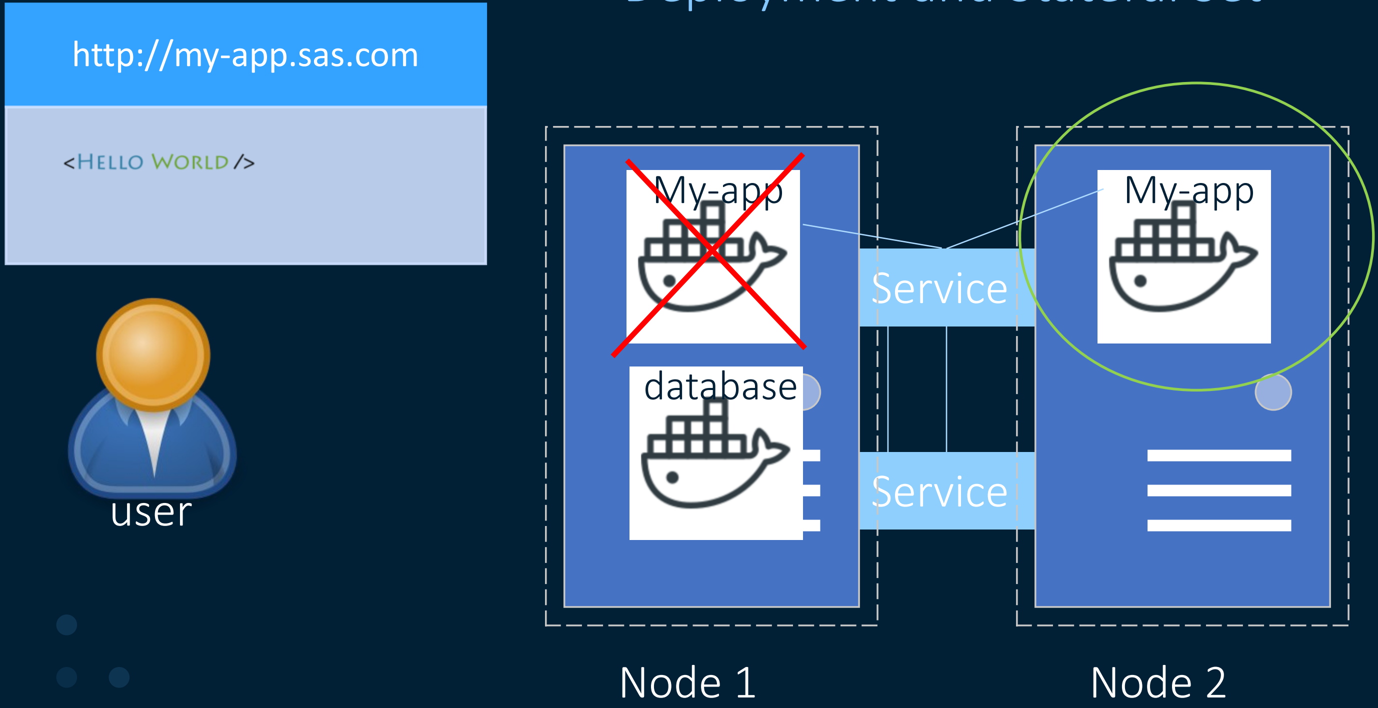
Deployment and Stateful Set





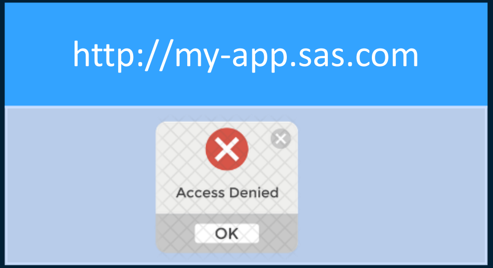
K8s Architecture

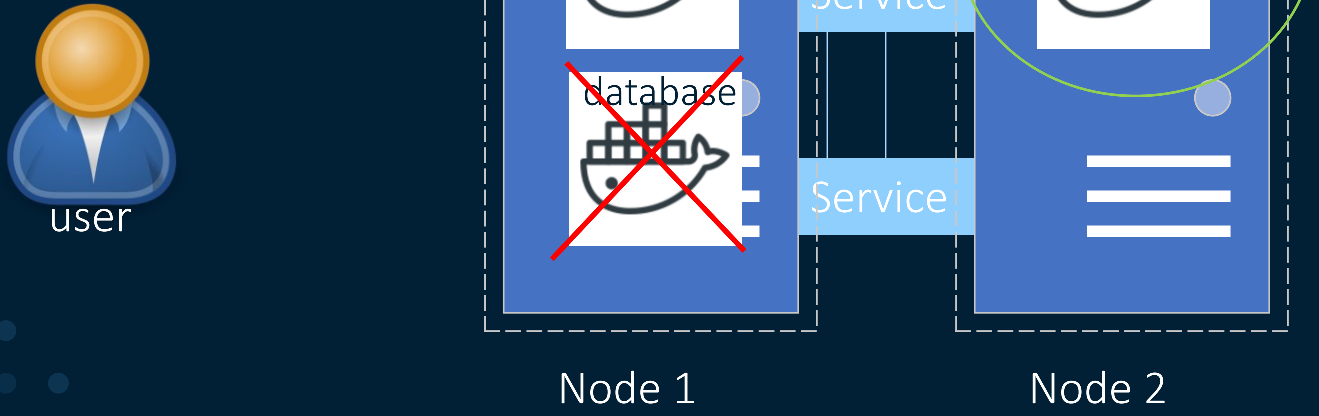
Deployment and Stateful Set



K8s Architecture

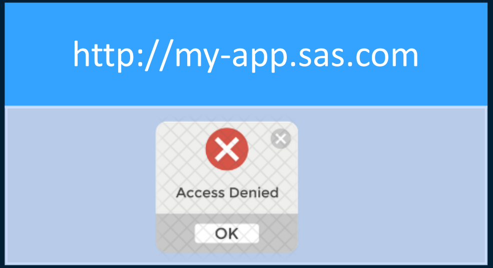
Deployment and Stateful Set

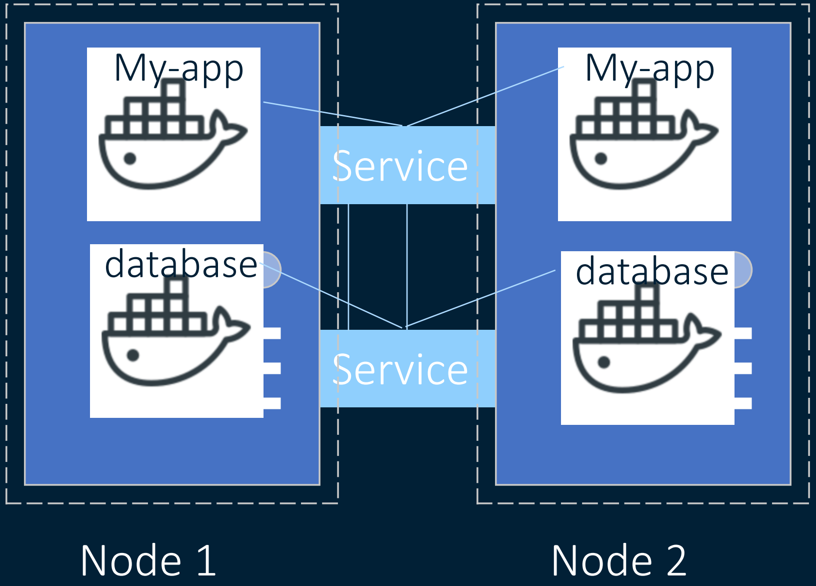




K8s Architecture

Deployment and Stateful Set

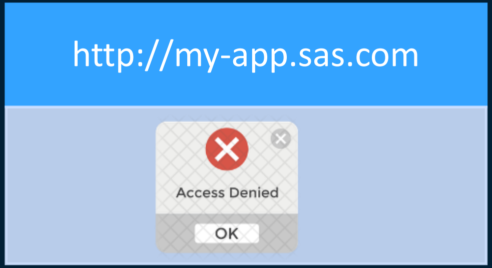


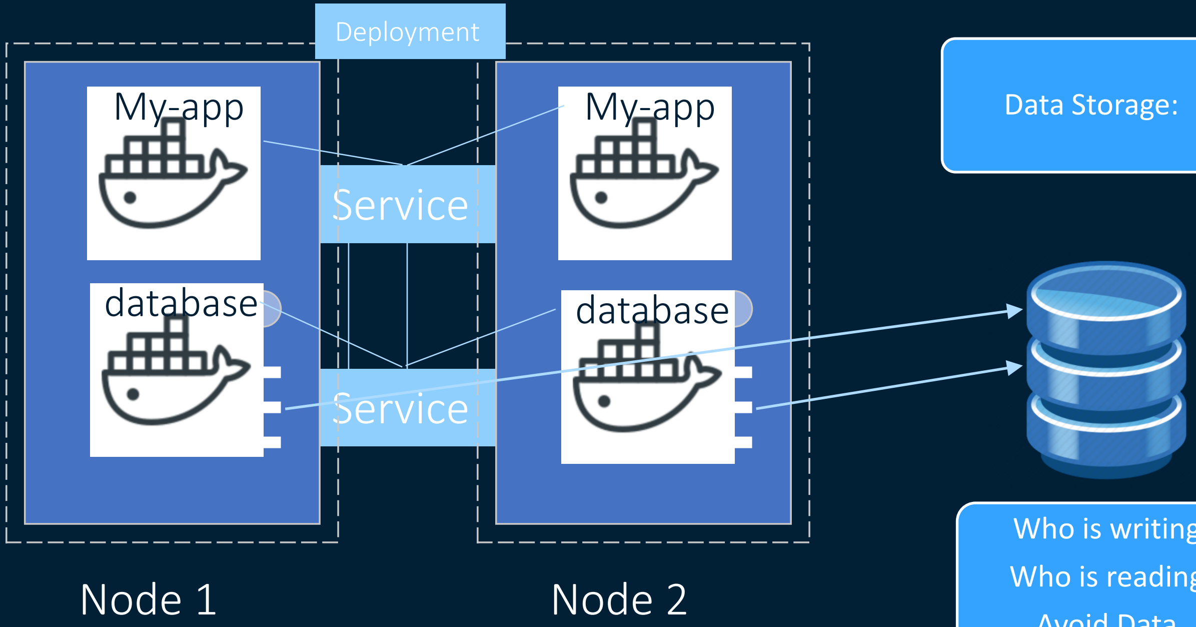


DBs can't be replicated using a Deployment

K8s Architecture

Deployment and Stateful Set





Data Storage:

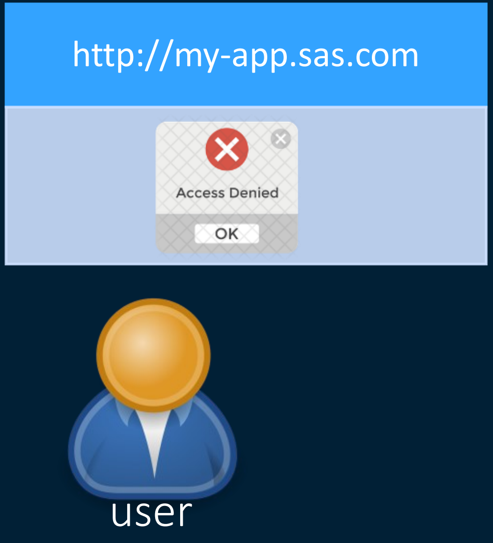
Who is writing

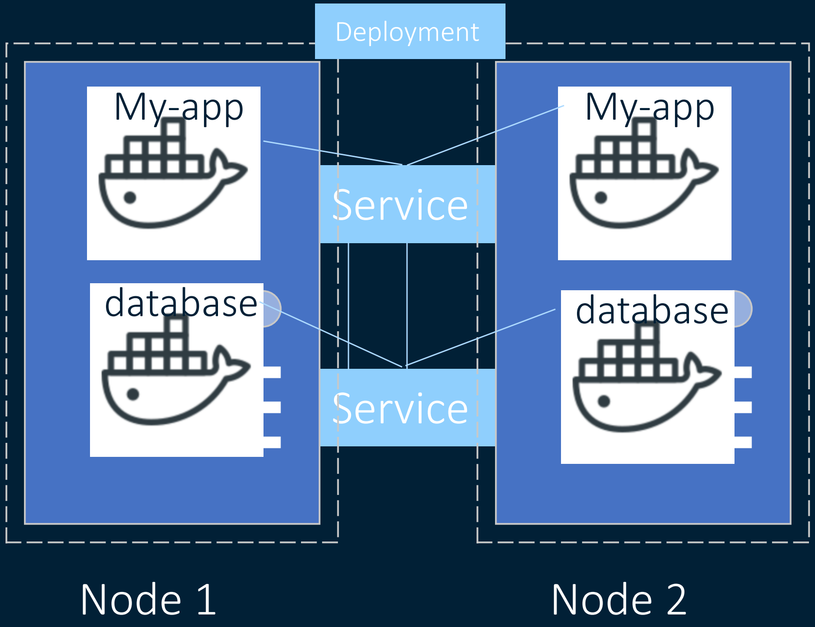
Who is reading

Avoid Data inconsistenciy

K8s Architecture

Deployment and Stateful Set

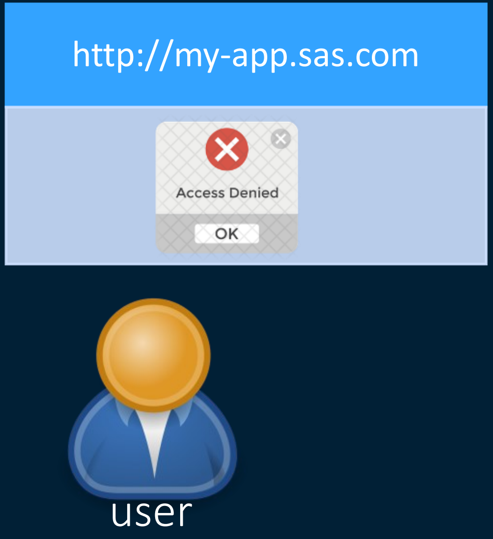


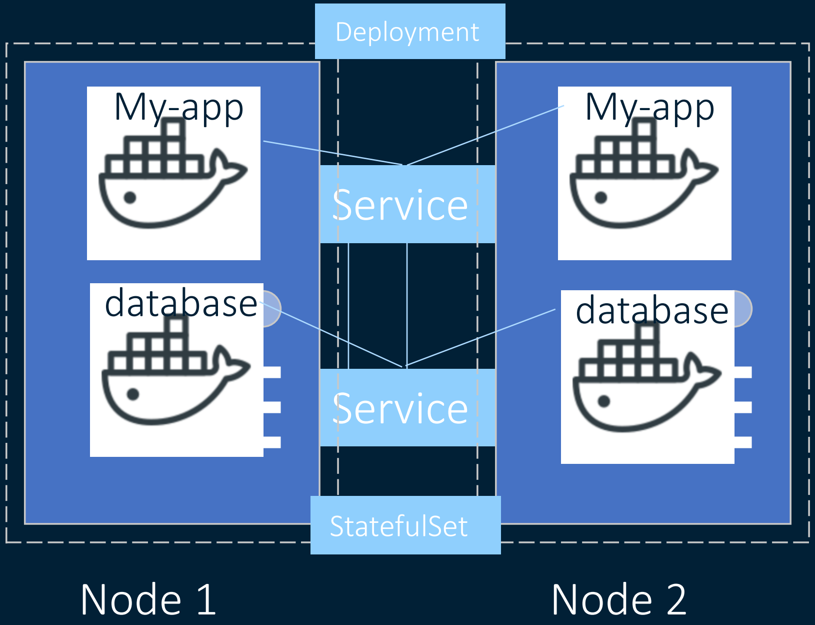


StatefulSet:
sts

K8s Architecture

Deployment and Stateful Set

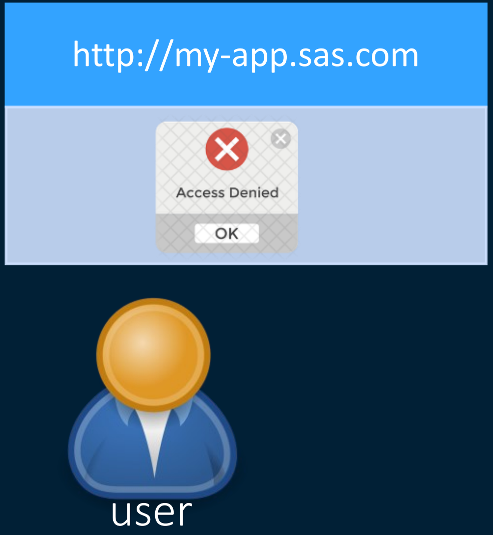


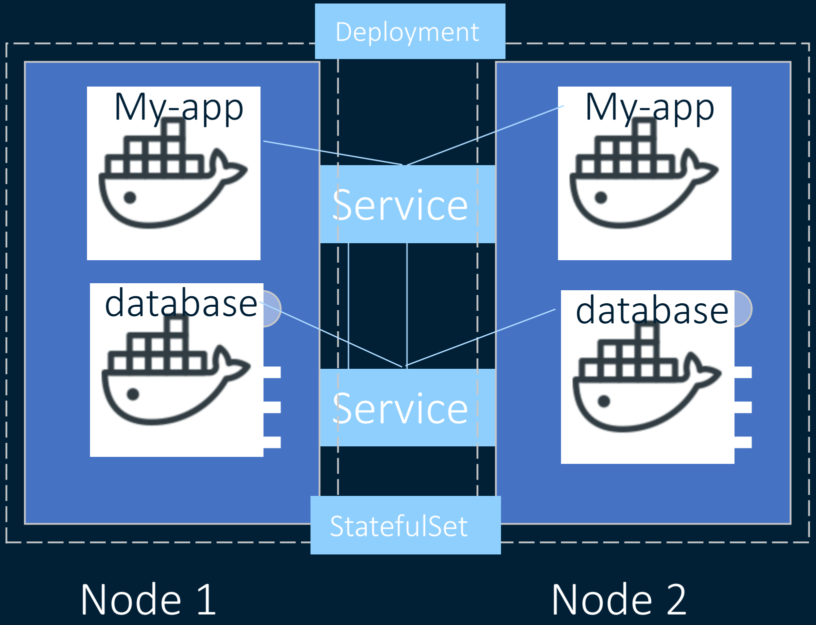


For STATEFUL apps

K8s Architecture

Deployment and Stateful Set



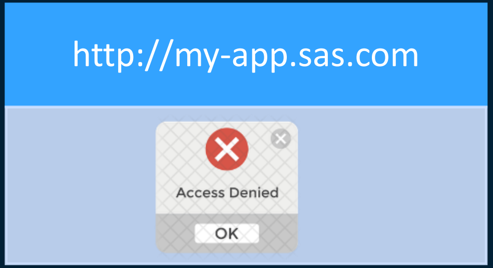


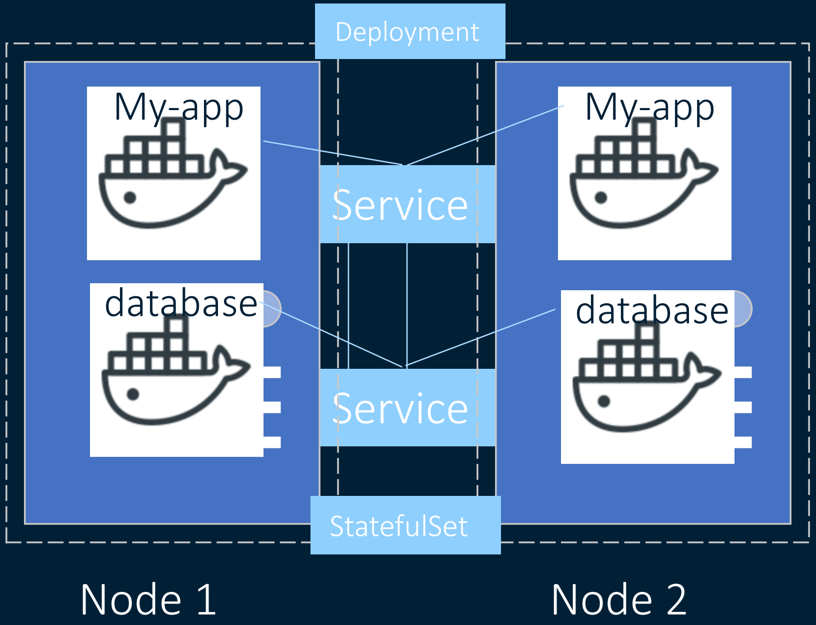
Deployment for stateless Apps

StatefulSet for stateful Apps or Databases

K8s Architecture

Deployment and Stateful Set

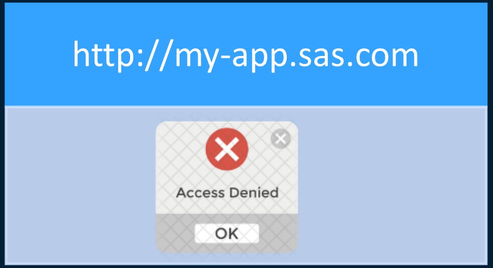


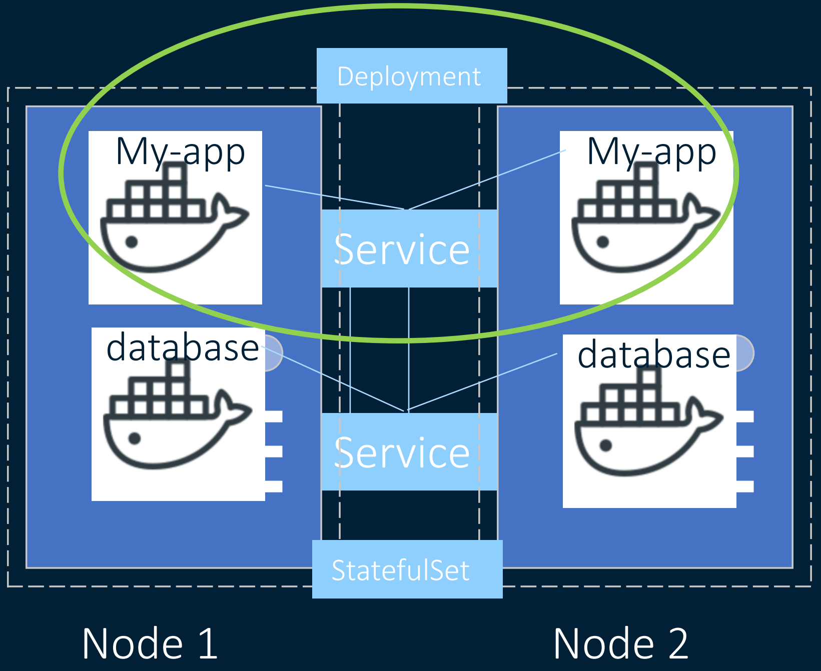


Deploying StatefulSet is not easy in Kubernetes

K8s Architecture

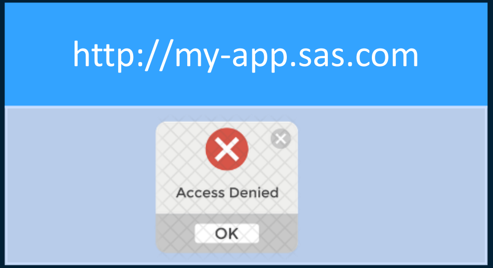
Deployment and Stateful Set

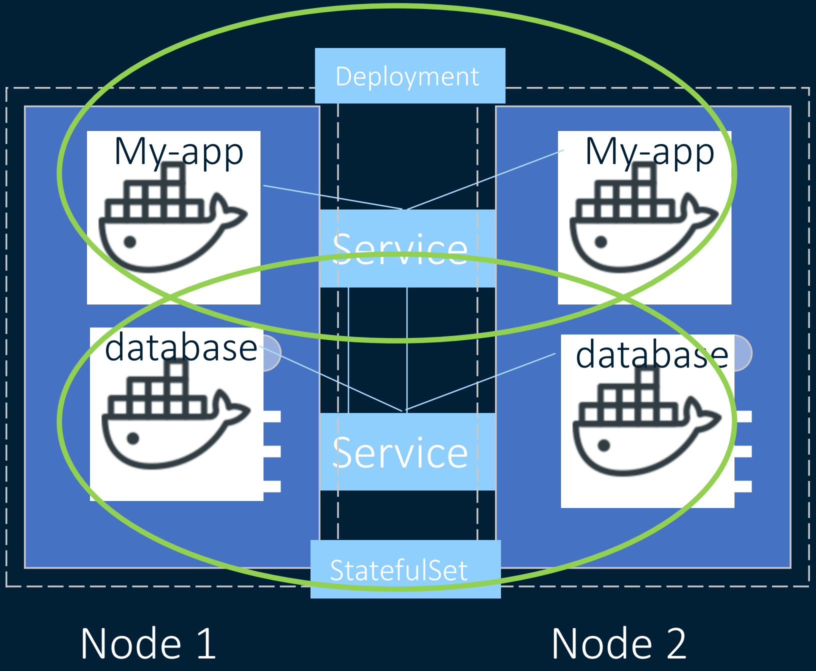




K8s Architecture

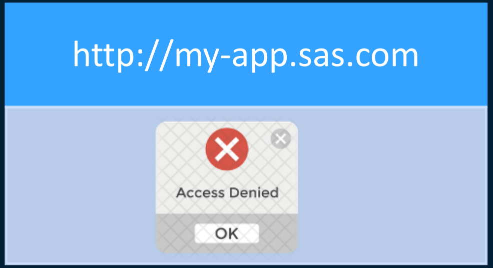
Deployment and Stateful Set

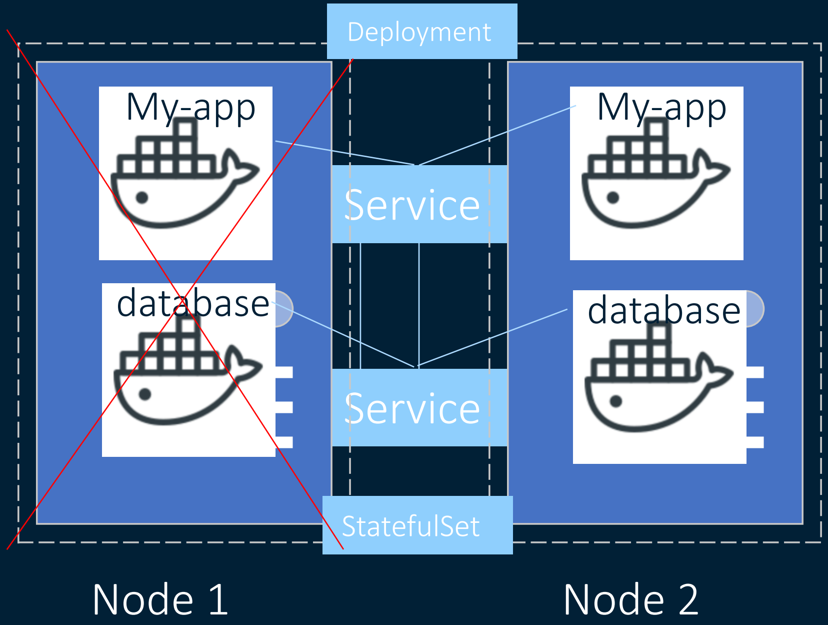




K8s Architecture

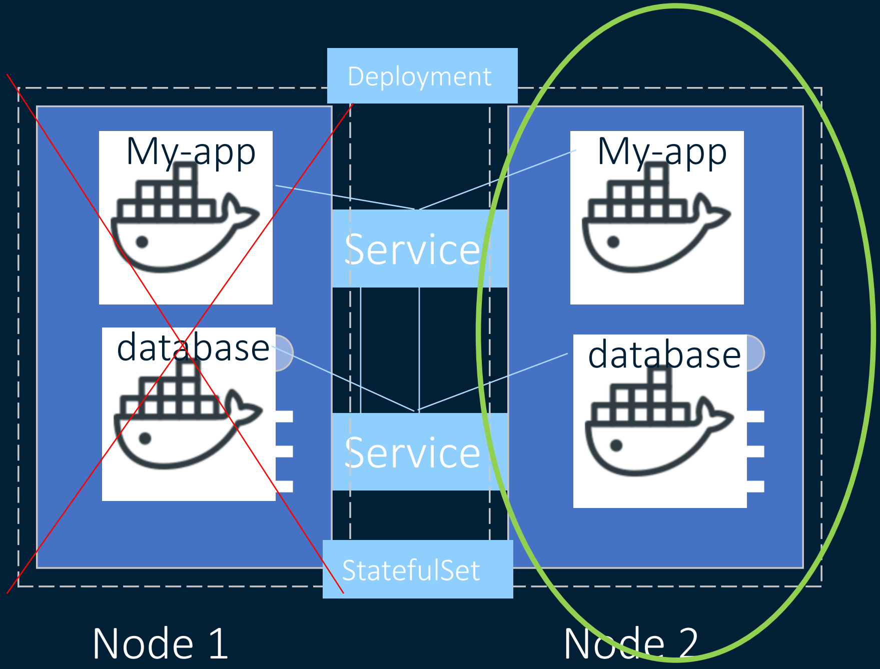
Deployment and Stateful Set





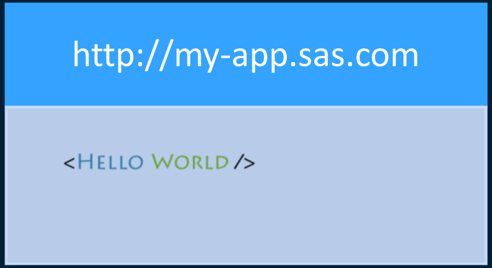
K8s Architecture

Deployment and Stateful Set



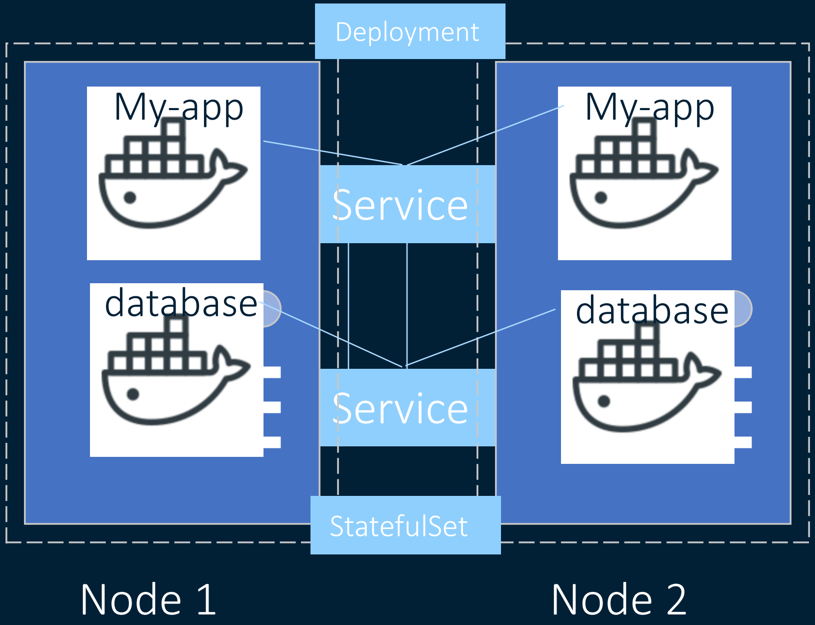
K8s Architecture

Deployment and Stateful Set



http://my-app.sas.com

<HELLO WORLD />



Now for the magic - two key concepts

Immutable architecture & declarative deployment

Immutability and declarative deployment are cornerstones of Kubernetes

- Immutable architecture
  - Immutable = unchanging over time or unable to be changed
  - You do not update the containers, if there is a new software version or software update you create a new container
  - Therefore, you will **NOT** apply maintenance as you did with SAS 9 or Viya 3
  - You deploy the new containers to replace the older version of software
  - This makes it easier to roll-back an upgrade
  - This also supports the concept of Blue-Green deployments

Now for the magic - two key concepts

Immutable architecture & declarative deployment

Immutability and declarative deployment are cornerstones of Kubernetes

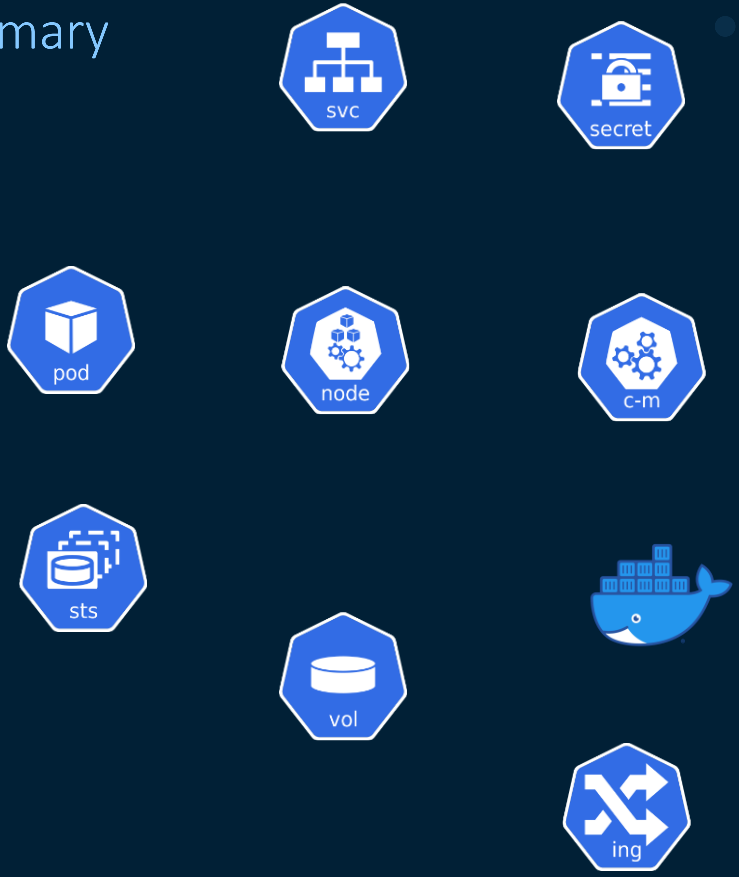
- Declarative deployment
  - You tell Kubernetes what the desired state should be (you declare it)
    - For example, “I want 3 CAS worker pods” or “I want 2 SAS Logon Manager pods”
  - If you declare that multiple instances of a pod or resource are required Kubernetes does its best to ensure that the desired state is achieved
    - If a resource stops, K8s will automatically start another instance to ensure the declared state
    - There is no guarantee that you will get what you ask for, the resources must be available to fulfill the request
  - Kubernetes uses **Deployments** to manage these requirements

K8s Architecture

Summary

- Main Kubernetes Components summarized
  - Container
  - Pod – abstraction over a container
  - Nodes – server instances
  - Services – static ip for pods
  - Ingress - to route traffic into cluster
  - External configuration – ConfigMaps
  - External configuration - Secrets
  - Data persistence - using volumes to disk
  - Pod blueprints with replicating mechanisms with Deployments and Stateful Sets

Is there more – oh, yes there is a lot more



The end Part 1

Pause for 10. min

sas.com