KubeDirector: Open Source Project for Stateful Applications on Kubernetes
Today’s Speakers

Tom Phelan
Co-Founder and Chief Architect
BlueData
@tapbluedata

Joel Baxter
Distinguished Engineer
BlueData
@joel_k_baxter
Agenda

- Kubernetes (K8s) and Stateful / Stateless Applications
- Complex Stateful Applications on Kubernetes
- BlueData, BlueK8s, and KubeDirector
- KubeDirector Deep Dive
- KubeDirector Demonstration
- Key Takeaways
What is Kubernetes (K8s?)

• Open source “platform” for container orchestration

• Platform **building blocks** vs. turnkey platform
  – [https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/#what-kubernetes-is-not](https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/#what-kubernetes-is-not)

• Top use case is **stateless / microservices** deployments

• Evolving for **stateful** applications
Stateless Applications on K8s

• **Stateless**
  – Each application service instance is configured identically
  – All information stored remotely
  – “Remotely” refers to some persistent storage that has a life span different from that of the container
  – Frequently referred to as “cattle”
Stateful Applications on K8s?

- **Stateful**
  - Each application service instance is configured differently
  - Critical information stored locally
  - “Locally” means that the application running in the container accesses the information via file system reads/writes rather than some remote access protocol
  - Frequently referred to as “**pets**”
Complex Stateful Applications

• Big Data / AI / Machine Learning / Deep Learning

What do all these applications have in common?
– Require large amounts of data
– Use distributed processing, multiple tools / services
– When on-prem, typically deployed on bare-metal
– Do not have a cloud native architecture
  • No microservices
  • Application instance-specific state
Example: Hadoop in Containers

Running Hadoop clusters in containers:
Kubernetes – Components

- Objects
- Pods
- Statefulsets
- PersistentVolumes
- Operators
- Custom Resource Definitions
**Kubernetes – Operators**

- **Operator**
  - A way of packaging, deploying, and managing a given application

- **Operator Framework**
  - A set of developer and runtime tools to help accelerate the writing of a Operator

- **Operator SDK**
  - An SDK that further hides the complexities of the Kubernetes API

Source: [https://coreos.com/operators](https://coreos.com/operators)
Kubernetes – Operators

• Application-specific means a new operator needs to be written for each application
What to Do?

• There needs to be an easier way to deploy and manage clusters running complex stateful applications
BlueK8s and KubeDirector

- BlueK8s is an Apache open source initiative focused on bringing enterprise support for complex stateful applications to Kubernetes.

- A series of open source projects will be rolled out under the BlueK8s umbrella.
  - The first major project is “KubeDirector”: [https://github.com/bluek8s/kubedirector](https://github.com/bluek8s/kubedirector)

Operation: Stateful BlueK8s and KubeDirector
Motivation

• Why create KubeDirector? Why use it?
  – E.g. why not app-specific operators, Helm, Kubeflow...

• Reframed: which architecture enables features we want (current or future)?

• Find sweet spot for users between two extremes:
  – Direct use of K8s APIs & “generic” deployment
  – Hardcoded application-specific solutions

• Abstractions + features guided by domain focus
Domain Focus

• Interested in best supporting apps that:
  – Are scale-out
  – May have “non cloud-native” service architecture
  – Have stateful cluster members
  – Need to access data lakes
  – Have user roles w/ distinct workflows and privileges
  – Integrate w/ enterprise services for authentication, certificate and license management, etc.
KubeDirector Overview

• KubeDirector is a K8s “custom controller”
• Watches for custom resources (CRs) to appear/change
• Creates/modifies standard K8s resources (StatefulSets etc.) in response, to implement specs from CRs
• Differs from normal Kubernetes Operator pattern:
  – No app-specific logic in KubeDirector code
  – App deployment is data-driven from external app definitions
  – Supports interactions among different apps + other objects
Deploy KubeDirector to K8s

```bash
kubectl create -f kubedirector/deployment.yaml
```
Separation of Concerns

• Application experts (on-site or elsewhere)
  – Responsible for making app images/metadata/configscripts
  – No need to write Go code or understand Operator concepts

• Administrators (on-site)
  – Select which apps are available to end users
  – Change app versions independently of KubeDirector upgrade

• End users
  – Pick from menu of applications and config choices
Alternatives Comparison 1/2

• Support distinctions between IT, app expert, project manager, and data scientist
  – Unlike Helm 2/3 & Kubeflow
• Integrate with K8s user authentication and ACLs
  – Unlike Helm 2 (Tiller)
• Support post-deployment autoremediation, autoscale, and other lifecycle events w/ app-specific logic
  – Unlike Helm 3 & Kubeflow
Alternatives Comparison 2/2

• Also a couple of behaviors not found in app-specific operators, and not a picnic in other solutions:
  – Support end-user import of new application types
  – Apply common features across multiple application types from different developers
KubeDirector Concepts
Custom Resource Definitions

• Primary CRD: KubeDirectorCluster
  – Models any kind of application instance launchable by KubeDirector

• Other CRDs for related objects, e.g.
  – App definitions (KubeDirectorApp)
  – DataTaps and other shared storage
  – Config sets for AD/LDAP integration for containers
  – Machine Learning models

• This talk will concentrate on KubeDirectorCluster/App
• KubeDirector Administration
• Application Preparation
• Application Instance Deployment
Deployment

- Create custom resource definitions (CRDs) in your K8s cluster
- Deploy KubeDirector
  - Normally runs in a Pod on same K8s cluster
  - Authenticates to K8s API w/ privileged service account
- Configure KubeDirector global settings
  - E.g. supply app definitions, set types of service & storage
• KubeDirector Administration
• Application Preparation
• Application Instance Deployment
App Definition Metadata

• App identifier/description/version
• Service endpoints
• Available “roles”, and container image per role
• Available deploy-time choices, and their effects on services per role
• Info for optional runtime setup package
• And more!
App Definition Example 1/3

apiVersion: kubedirector.bluedata.io/v1alpha1
kind: KubeDirectorApp
metadata:
  name: spark221e2

spec:
  label:
    name: Spark 2.2.1 on centos7x with Jupyter
  default_image_repo_tag: docker.io/bluedata/sparkbase:2.0
  default_config_package:
    package_url: https://s3.amazonaws.com/mybucket/spark221e2/appconfig.tgz
roles:
- id: controller
  cardinality: 1
- id: worker
  cardinality: 0+

services:
- id: spark
  label:
    name: Spark master
  endpoint:
    port: 7077

  - id: spark_master_ui
    label:
      name: Spark master (web UI)
    endpoint:
      port: 8080
      is_dashboard: true
      url_scheme: http

  - id: spark_worker_ui
    label:
      name: Spark worker (web UI)
    endpoint:
      port: 8081
      is_dashboard: true
      url_scheme: http
config:

selected_roles:
- controller
- worker

role_services:
- role_id: controller
  service_ids:
  - spark
  - spark_master_ui
- role_id: worker
  service_ids:
  - spark_worker_ui
Application Setup Package

• Optional tgz injected into each container, contains:
  – Entrypoint script
  – Standard script functions for reading deployment info
  – Any artifacts (config file templates etc.) required for setup

• Entrypoint script will be invoked at lifecycle events:
  – This container has just been created
  – Some other member(s) added to or removed from the cluster
Setup Script Actions

• Perform any setup that requires runtime info
  – E.g. FQDNs of other member(s) of the cluster

• Enable and start appropriate services
  – Can query the role of current node
  – Services-to-start depend on role and deploy-time choices

• Can use features of KubeDirector Agent in future
• KubeDirector Administration
• Application Preparation
• Application Instance Deployment
apiVersion: "kubedirector.bluedata.io/v1alpha1"
kind: "KubeDirectorCluster"
metadata:
  name: "spark-instance"
spec:
  app: spark221e2
  roles:
  - id: controller
    resources:
      limits:
        memory: "4Gi"
  - id: worker
    members: 2
Cluster Creation Sequence 2/2

KubeDirector

Kubernetes API

statefulset 1

statefulset 2

CR

pod

svc

svc

svc

pod

pod

pod

container image repository

application configuration artifacts
Other Operations

- Shrink & expand of role member count is handled similarly
- All resources are automatically cleaned up if CR is deleted (because CR is their “owner”)
- End user can read the CR to see current status, service objects, event history, etc.
Key Takeaways

• Running complex stateful applications on Kubernetes is challenging today
• The goal of BlueK8s and KubeDirector is to make it easier to run such applications on Kubernetes
• Learn more about KubeDirector:
  – https://github.com/bluek8s/kubedirector/wiki
Join the KubeDirector Community!

Tom Phelan
@tapbluedata

Joel Baxter
@joel_k_baxter

https://github.com/bluek8s