Kubebuilder

Tutorial

Ondrej Sery

osery@purestorage.com

PURESTORAGE®

PureStorage

Delivering the Modern Data Experience

GROWTH

+$50B

Total Addressable Market

$1.64B

FY20 Annual Revenue

—

Subscription Services up **37%** Year-over-year (Q1 FY21)

CUSTOMERS

7,800+

Customers

—

1,700

New Customers in FY20

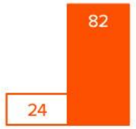
45%
of Fortune 500 Companies

33%+
of Bookings from Cloud Customers (Q1 FY21)

LEADERSHIP

Gartner Magic Quadrant

For the sixth year in a row, Pure Storage is a leader in the Gartner Magic Quadrant.



Net Promoter Score

In the top 1% of B2B companies.

COMPANY MILESTONES

2009	Founded
2011	Pioneer in All-Flash
2012	Introduction of FlashArray™
Late 2012	First international office open in Europe
Mid 2014	Evergreen™ Storage Non-Disruptive Upgrades
Early 2015	Pure1® Cloud-Based Management
Late 2015	IPO (PSTG)
2016	Introduction of FlashBlade®
Early 2018	Pure introduces the first AI-ready Infrastructure (AIRI™)
Mid 2018	Launch of as-a-Service model (Pure as-a-Service)
2018	Cloud Data Services
2018	$1+ Billion in revenue
2019	Compuverde Acquisition
2019	Expands FlashArray™ Family
2019	Named a Leader: Gartner Magic Quadrant
2020	Pure as-a-Service celebrates 2-year anniversary

Kubernetes Operators

THE WHY

“Operating” Kubernetes Applications

- **Stateless applications**
  - Easy enough using Kubernetes alone
  - New version & rolling update

- **Stateful applications**
  - R/O mode, Backup, Migration, Upgrade, R/W mode
  - Manually by a human “operator” with domain knowledge

⇒ *Automated “operator” with domain knowledge*

PURESTORAGE®

Custom Resource Definitions

- New Kubernetes resources/Kinds
  - Domain specific description of the desired state.
  - e.g., Prometheus, PrometheusRule

```yaml
apiVersion: monitoring.coreos.com/v1
kind: Prometheus
metadata:
  name: prometheus
spec:
  serviceAccountName: prometheus
  serviceMonitorSelector:
    matchLabels:
      team: frontend
  resources:
    requests:
      memory: 400Mi
  enableAdminAPI: true
```

```yaml
apiVersion: monitoring.coreos.com/v1
kind: PrometheusRule
metadata:
  labels:
    prometheus: example
    role: alert-rules
  name: prometheus-example-rules
spec:
  groups:
  - name: ./example.rules
    rules:
    - alert: ExampleAlert
      expr: vector(1)
```

Kubernetes Control Loop

OBSERVE → CHECK DIFFERENCE → TAKE ACTION

- Observe:
  - Kubernetes resources/Kinds with the desired state
  - *E.g., replica set with 3 replicas*
- Check Difference:
  - Between desired and actual state
  - *E.g., only 2 running pods*
- Take Action:
  - To remove the difference
  - *E.g., start another pod*

PURESTORAGE®

Tooling

- **Kubebuilder**
- Operator SDK
  - Helm
  - Ansible
  - Golang (now using Kubebuilder)
- KUDO
- Metacontroller

Kubebuilder

THE HOW

Kubebuilder Overview

- Generates a project skeleton for an operator
  - Go schema & API definitions
  - Go controller skeletons
  - Deployment yaml files
  - Test skeletons
  - Dockerfile
  - Makefile
- Covers also evolution
  - Add new (versions of) resources

Tutorial: Coffee Maker Operator

- **Existing:** Coffee Maker App
  - GitHub: github.com/osery/coffee-maker
  - Docker image: ghcr.io/osery/coffee-maker:latest

- **Our goal:** *Coffee Maker K8s Operator*
  - CRD-based API for the Coffee Maker App
  - Controller
    - Call the coffee maker REST API
    - New beverage
    - Status updates

New Kubebuilder Project

- We need a new go module:

```
$ go mod init coffee.demo.purestorage.com
```

- Generate the project skeleton:

```
$ kubebuilder init --domain coffee.demo.purestorage.com
```

Let's try it and explore a bit…

New API

- Let’s create a Coffee CRD:

```
$ kubebuilder create api --group beverage --version v1 --kind Coffee
```

- *Generates:*
    - Golang custom resource types with json bindings
    - K8s CRD and RBAC yaml definitions
    - Example custom resource yaml files
    - Controller skeleton
    - Boilerplate registration code, etc.

Add Fields

- Manual edits to add/remove/update the actual payload
- Special types:
  - resource.Quantity (e.g., “1.5Gi”)
  - metav1.Time (time.Time with correct yaml/json marshalling)
- Our “Coffee” fields:
  - `Spec:`
    - `type = {espresso, americano, latte}`
    - `extraSugar = {true, false}`
  - `Status:`
    - `status = {queued, brewing, done, failed}`

Build and Deploy CRDs

- Regenerate everything and build:

```
$ make manifests
```

- Install CRD definitions into a K8s cluster (current `kubectl` default)

```
$ make install
```

- Uninstall CRD definitions

```
$ make uninstall
```

PURESTORAGE®

Customizing CRD

- Mark additional fields in the default ”`kubectl get`” display:

```
// +kubebuilder:printcolumn:name="Type",type=string,JSONPath=`.spec.type`
// +kubebuilder:printcolumn:name="ExtraSugar",type=boolean,JSONPath=`.spec.extraSugar`
// +kubebuilder:printcolumn:name="Status",type=string,JSONPath=`.status.status`
```

- Ensure only allowed values are used as coffee Type.

```
// +kubebuilder:validation:Enum=espresso;latte;americano
```

Controller

- Get K8s resource
- Get REST resource
- Create a new if missing
- Update `Status`
- Requeue until in status done

Build and Deploy Controller

- Build a push docker image:

```
$ make docker-build docker-push IMG=repository/image:tag
```

- Install the controller into a K8s cluster

```
$ make deploy IMG=repository/image:tag
```

RESOURCES

- Kubebuilder
    - GitHub: https://github.com/kubernetes-sigs/kubebuilder
    - Book: https://kubebuilder.io
- K3s: https://k3s.io
- Demo materials:
    - Will upload to https://github.com/osery/2020-11-24-kubebuilder-tutorial after the talk...

Q&A