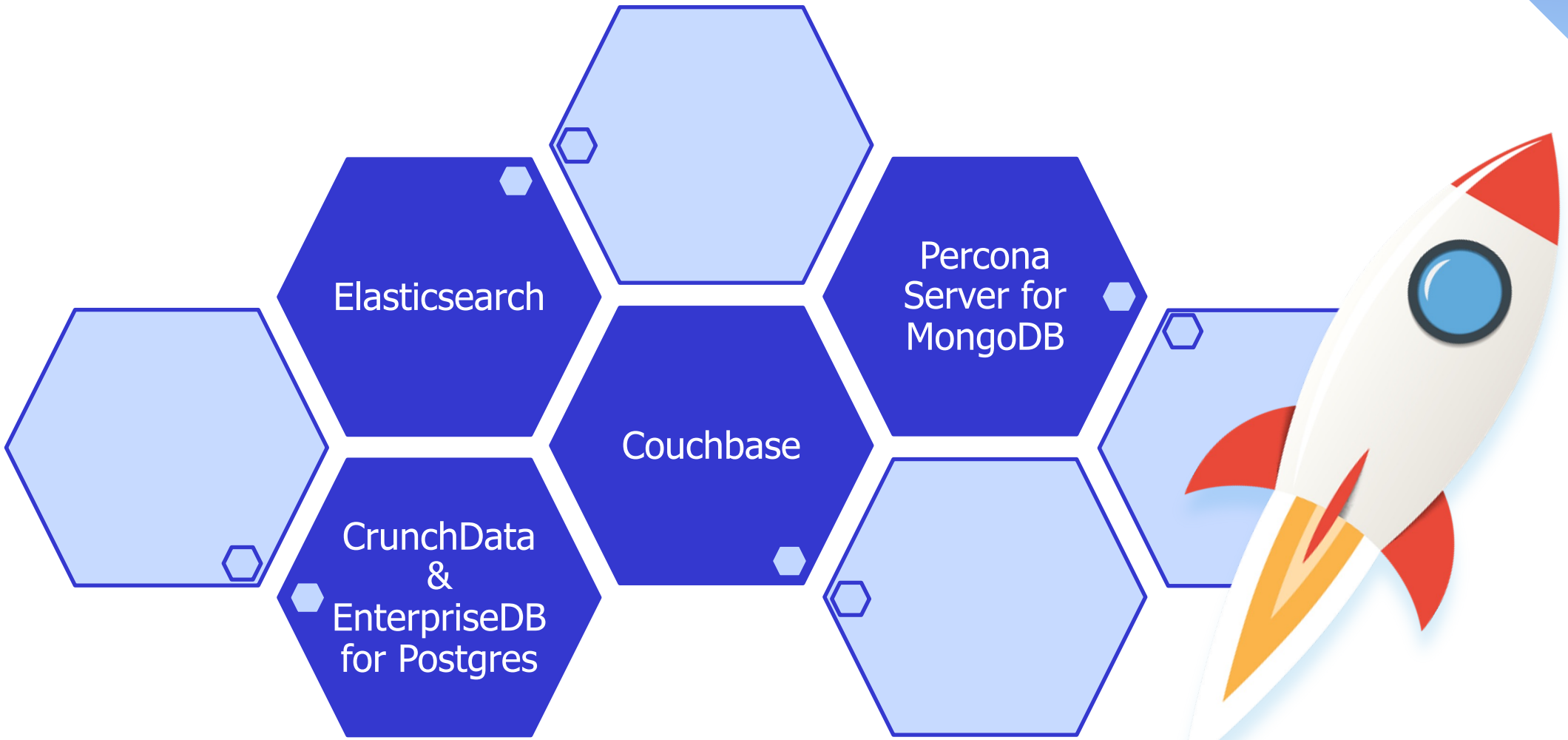
But What About Operators?

Operators

Accelerating Stateful Workloads on K8s



KASTEN by Veeam

What is an Operator?

An operator is a Kubernetes controller that **understands 2 domains: Kubernetes and something else**. By combining knowledge of both domains, it can automate tasks that usually require a human operator that understands both domains.

Jimmy Zelinskie, Former OpenShift Principal Product Manager

KASTEN by Veeam

Example Operator manifest using *Percona Operator for MongoDB*

```yaml
apiVersion: psmdb.percona.com/v1
kind: PerconaServerMongoDB
metadata:
  name: my-cluster-name
  finalizers:
    - delete-psmdb-pods-in-order
spec:
  crVersion: 1.13.0
  allowUnsafeConfigurations: false
  replsets:
  - name: rs0
    size: 3
    expose:
      enabled: false
      exposeType: ClusterIP
    resources:
      limits:
        cpu: "300m"
        memory: "0.5G"
      requests:
        cpu: "300m"
        memory: "0.5G"
    volumeSpec:
      persistentVolumeClaim:
        resources:
          requests:
            storage: 3Gi
  sharding:
    enabled: true
    configsvrReplSet:
      size: 3
.......
```

1x Cluster

3x Percona Server Replicas

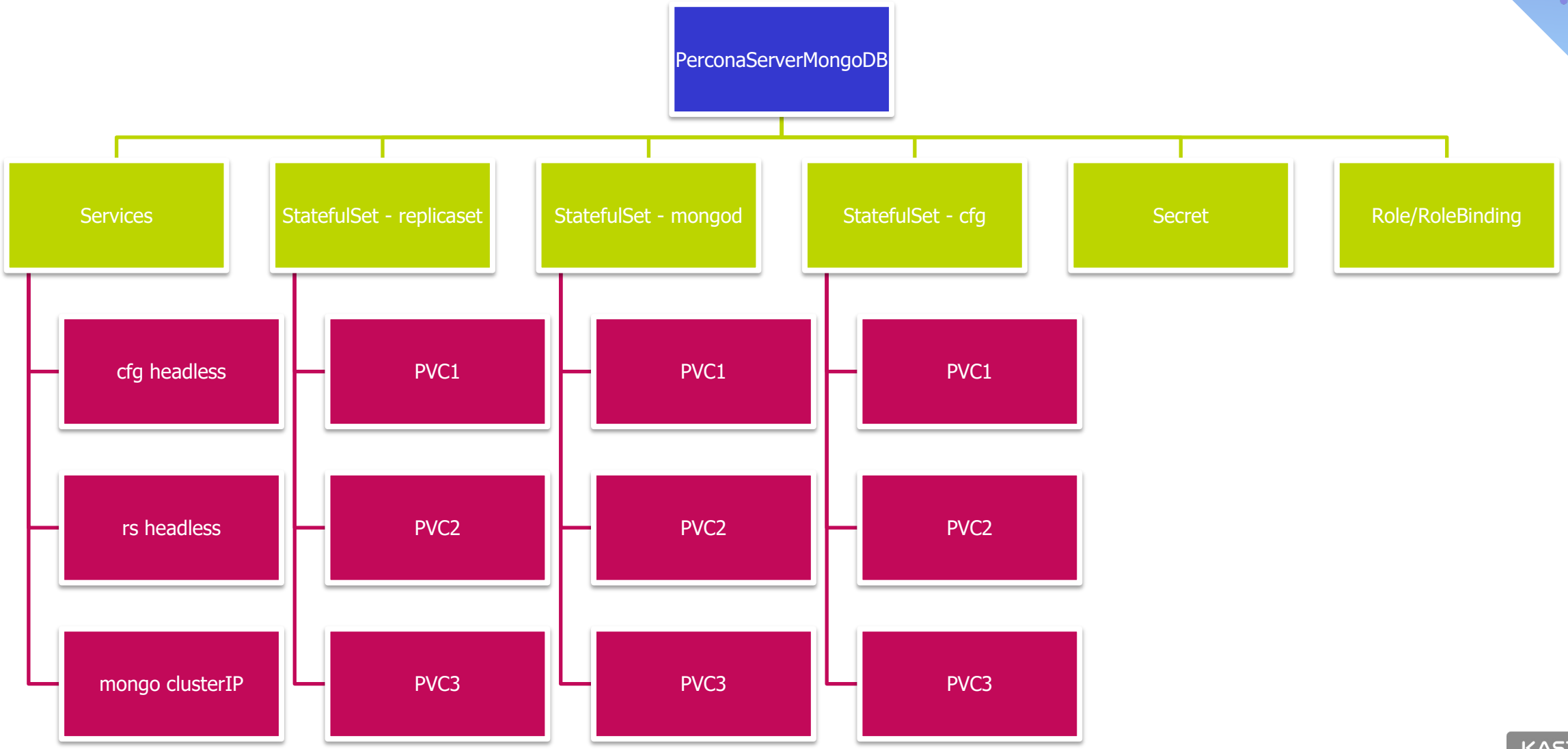
Each with its own PVC

3x Config Server Replicas

KASTEN by Veeam

Derived Resources

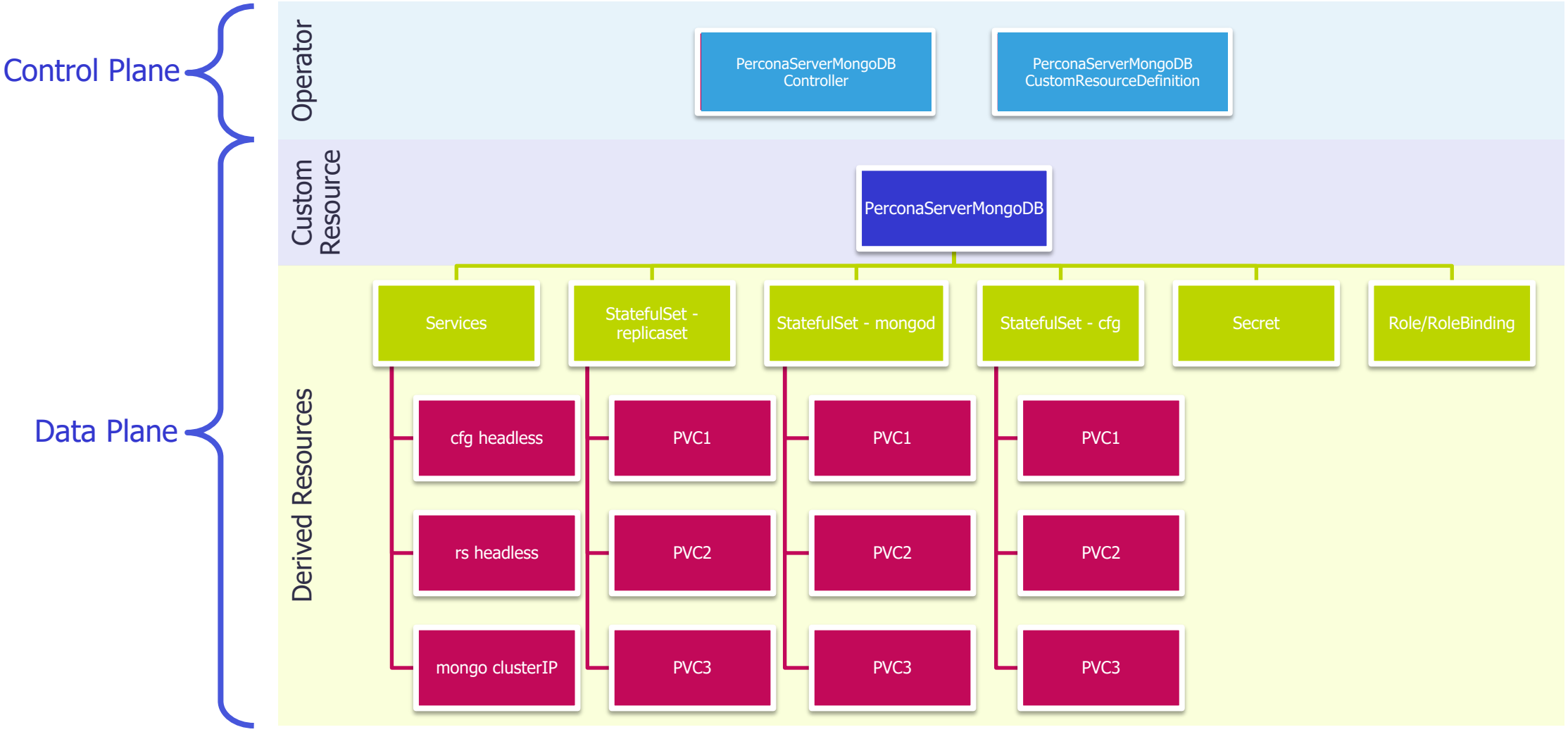
Created by the Operator



KASTEN by Veeam

Operator Dichotomy

Control v. Data



KASTEN by Veeam

"My Operator does backup for me…"

Leverage it!

- More Operators are providing built-in protection logic
- Data service backup does not protect your complete app
- Use Blueprints to work *with* the Operator, not *against* it!

```yaml
apiVersion: psmdb.percona.com/v1
kind: PerconaServerMongoDBBackup
metadata:
  name: backup1
spec:
  clusterName: my-cluster-name
  storageName: s3-us-west
```

```yaml
apiVersion: psmdb.percona.com/v1
kind: PerconaServerMongoDBRestore
metadata:
  name: restore1
spec:
  clusterName: my-cluster-name
  backupName: backup1
```


KASTEN by Veeam

```
6    backup:
7      outputArtifacts:
8        cloudObjects:
9          keyValue:
10           psmdbbackup: 'backup-{{ toDate "2006-01-02T15:04:05.999999999Z07:00" .Time | date "2006-01-02t15-04-05z07-00" }}'
11           namespace: "{{ .Object.metadata.namespace }}"
12     phases:
13     - func: KubeOps
14       name: createBackupCR
15       args:
16         operation: create
17         namespace: "{{ .Object.metadata.namespace }}"
18         spec: |-
19           apiVersion: psmdb.percona.com/v1
20           kind: PerconaServerMongoDBBackup
21           metadata:
22             finalizers:
23             - delete-backup
24             name: backup-{{ toDate "2006-01-02T15:04:05.999999999Z07:00" .Time | date "2006-01-02t15-04-05z07-00" }}
25           spec:
26             clusterName: {{ .Object.metadata.name }}
27             storageName: my-s3-storage
28     - func: Wait
29       name: waitBackupReady
30       args:
31         timeout: 360s
32         conditions:
33           anyOf:
34             - condition: '{{ if  eq "{ $.status.state }" "ready" }}true{{ else }}false{{ end }}'
35               objectReference:
36                 apiVersion: v1
37                 group: psmdb.percona.com
38                 resource: perconaservermongodbbackups
39                 name: 'backup-{{ toDate "2006-01-02T15:04:05.999999999Z07:00" .Time | date "2006-01-02t15-04-05z07-00" }}'
40                 namespace: "{{ .Object.metadata.namespace}}"
```

Blueprint "backup" action

Kanister Function

Create Operator-controlled Backup CR

Store generated Backup CR name in RestorePoint

Kanister Function

Wait for new Backup CR to reach "Ready" state

KASTEN by Veeam

```
55    restore:
56      inputArtifactNames:
57      - cloudObjects
58      phases:
59      - func: Wait
60        name: waitMongoDBClusterReady
61        args:
62          timeout: 300s
63          conditions:
64            anyOf:
65              - condition: '{{ if  eq "{ $.status.state }" "ready" }}true{{ else }}false{{ end }}'
66                objectReference:
67                  apiVersion: v1
68                  group: psmdb.percona.com
69                  resource: perconaservermongodbs
70                  name: "{{ .Object.metadata.name}}"
71                  namespace: "{{ .Object.metadata.namespace}}"
72      - func: KubeOps
73        name: createRestoreFromBackup
74        args:
75          operation: create
76          namespace: "{{ .Object.metadata.namespace }}"
77          spec: |-
78            apiVersion: psmdb.percona.com/v1
79            kind: PerconaServerMongoDBRestore
80            metadata:
81              name: restore-{{ toDate "2006-01-02T15:04:05.999999999Z07:00" .Time | date "2006-01-02t15-04-05z07-00" }}
82            spec:
83              clusterName: {{ .Object.metadata.name }}
84              backupName: {{ .ArtifactsIn.cloudObjects.KeyValue.psmdbbackup }}
85              {{- if .Options }}
86              {{- if .Options.pitr }}
87              pitr:
88                type: date
89                date: {{ .Options.pitr }}
90              {{- end }}
91              {{- end }}
92      - func: Wait
93        name: waitRestoreReady
94        args:
95          timeout: 720s
96          conditions:
97            anyOf:
98              - condition: '{{ if  eq "{ $.status.state }" "ready" }}true{{ else }}false{{ end }}'
99                objectReference:
100                 apiVersion: v1
101                 group: psmdb.percona.com
102                 resource: perconaservermongodbrestores
103                 name: 'restore-{{ toDate "2006-01-02T15:04:05.999999999Z07:00" .Time | date "2006-01-02t15-04-05z07-00" }}'
104                 namespace: "{{ .Object.metadata.namespace}}"
```

Blueprint "restore" action

Reference output of "backup" action

Wait for new MongoDB cluster CR to be "Ready"

Create Operator-controlled Restore resource

Using Backup CR name stored in RestorePoint

Wait for Restore CR to be "Ready"

KASTEN by Veeam

```
41    delete:
42      inputArtifactNames:
43      - cloudObjects
44      phases:
45      - func: KubeOps
46        name: deleteBackupCR
47        args:
48          operation: delete
49          objectReference:
50            apiVersion: v1
51            group: psmdb.percona.com
52            resource: perconaservermongodbbackups
53            name: '{{ .ArtifactsIn.cloudObjects.KeyValue.psmdbbackup }}'
54            namespace: '{{ .ArtifactsIn.cloudObjects.KeyValue.namespace }}'
```

Blueprint "delete" action

Reference output of "backup" action

Delete Backup CR

Using Backup CR name stored in RestorePoint

KASTEN by Veeam

Should I include the Derived Resources in my Policy?

If you're using a Blueprint:

- No need to backup derived resources
- Excluding them can save time & storage
- Leverage label-based Exclude Filters as part of your K10 Policy:
  - Ex. `app.kubernetes.io/managed-by=percona-server-mongodb-operator`

If you're NOT using a Blueprint:

- Derived resources MUST be protected by K10 Policy
- Validate with Operator author that restore via derived resources is supported (and if additional transforms are required)

KASTEN by Veeam

Should I backup the Operator itself (aka Control Plane)?

You can, but it doesn't mean you should:

- Yes, operators are CRDs, and Kasten K10 protects CRDs
- Need to ensure Control Plane restore is complete before attempting to restore CRs controlled by the Operator
- OpenShift Operators have additional APIs to manage Operator packaging and deployment – just reinstall!
- Recommended approach:
  1. Reinstall Operator (typically via cluster deployment IaC)
  2. Restore apps including Operator-controlled CRs via K10

KASTEN by Veeam