ObjectRocket
a rackspace company

DBaaS on Kubernetes

KubeCon CloudNativeCon

ObjectRocket
a rackspace company

STEVE CROCE

PRODUCT MANAGER, ELASTICSEARCH

&

WESLEY ASHKINAZY

LEAD DEVELOPER, ELASTICSEARCH

Follow @OBJECTROCKET for slides and announcements.

KubeCon CloudNativeCon

About ObjectRocket

We help clients build better apps faster so they can focus on their mission, not their database.

Technology

DBaaS platform

Hassle-free hosting for **MongoDB®**, **Elasticsearch®** + **KIbana®**, and **Redis®**

Support

It's **the best hands-on support**, hands-down.

24x7x365 support from database experts with financially-backed SLAs.

Expertise

We're experts in scaling and supporting complex **production environments.**

What You’ll See Today

Why we’re adopting Kubernetes

The design choices that we faced

The choices that we made

What we’re going to be doing next

The Road to Kubernetes

Our Original Hosting Platform

Built Circa 2012

Built on OpenVZ

Custom orchestration, hardware, management systems

Assumes bare-metal environment

What’s Changed?

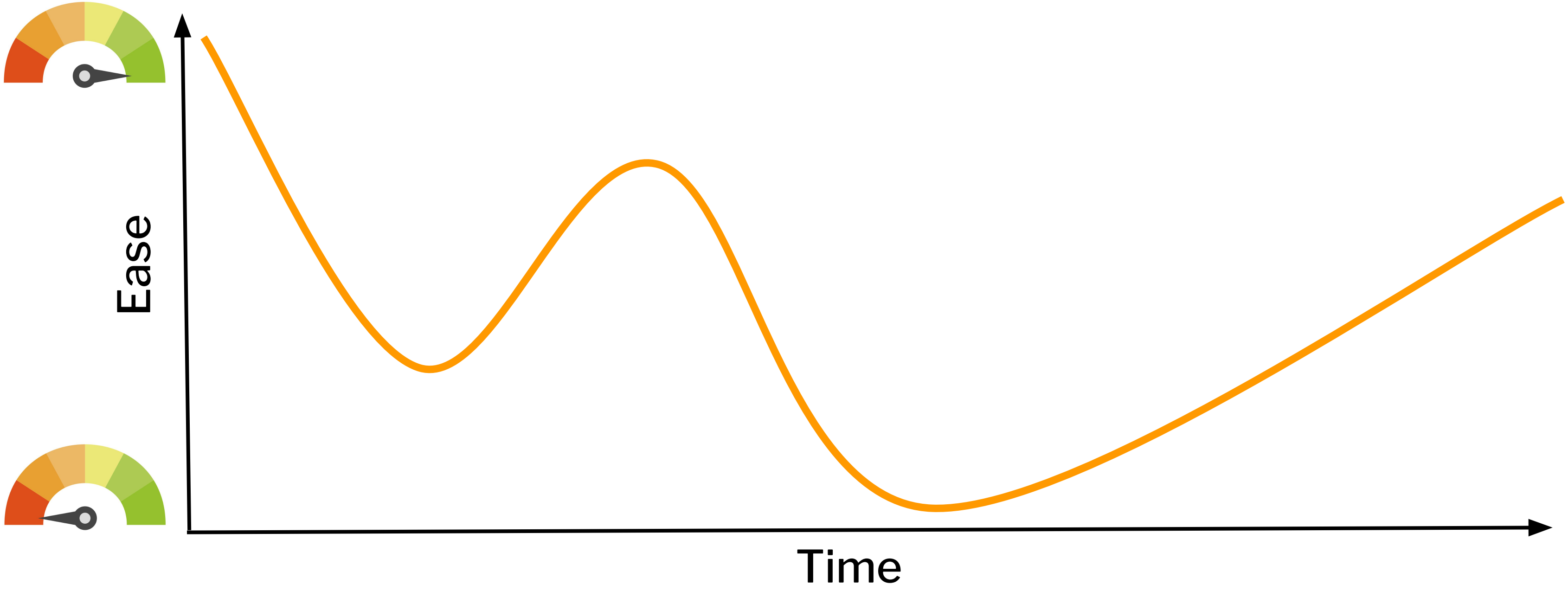
6 years is a long time in tech

Docker has become the de-facto container format

Custom orchestration is not a differentiator and awesome standard tools exist

Cloud usage has grown and not all need bare metal performance

Can Kubernetes Solve Our Problem?



ObjectRocket

Kubernetes + ObjectRocket:

Kubernetes Easy Meter

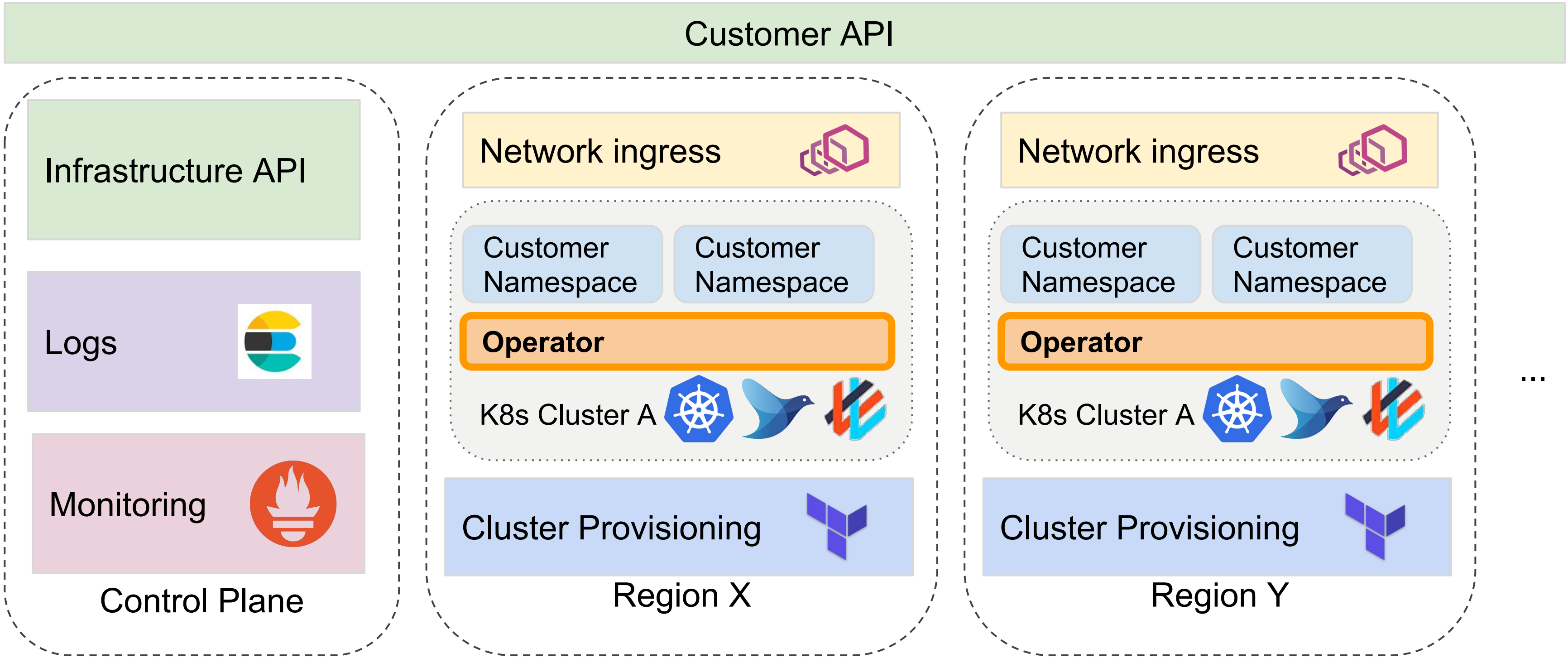
Pros

- Well-built, open source, modern orchestration
- First-class citizen in the clouds we want support
- People want to develop for it
- Operators make it easy… There are open source options everywhere
- It’s 🔥🔥🔥🔥

Cons

- Not built for stateful apps... databases are pretty damn stateful
- Most databases don’t tolerate disappearing resources
- All operators aren’t easy… we’ll need a whole lot more functionality

High-level Platform Architecture



What Our Service Must Do

01 **The Basics**

- Safely create a full cluster
- Make sure the cluster is healthy
- Delete all resources when asked

02 **Security**

- Handle multiple tenants
- Cluster Certificates
- Database User/Roles CRUD

03 **Database Administration**

- Cluster configuration and plugins
- Perform regular backups
- Running database utilities

04 **Service Features**

- Safely scale up/out/in/down
- Apply minor and patch updates
- Add-on dashboards and tooling

Database-as-a-Service: The Basics

What Our Service Must Do

01 The Basics

- Safely create a full cluster
- Make sure the cluster is healthy
- Delete all resources when asked

02 Security

- Handle multiple tenants
- Cluster Certificates
- Database User/Roles CRUD

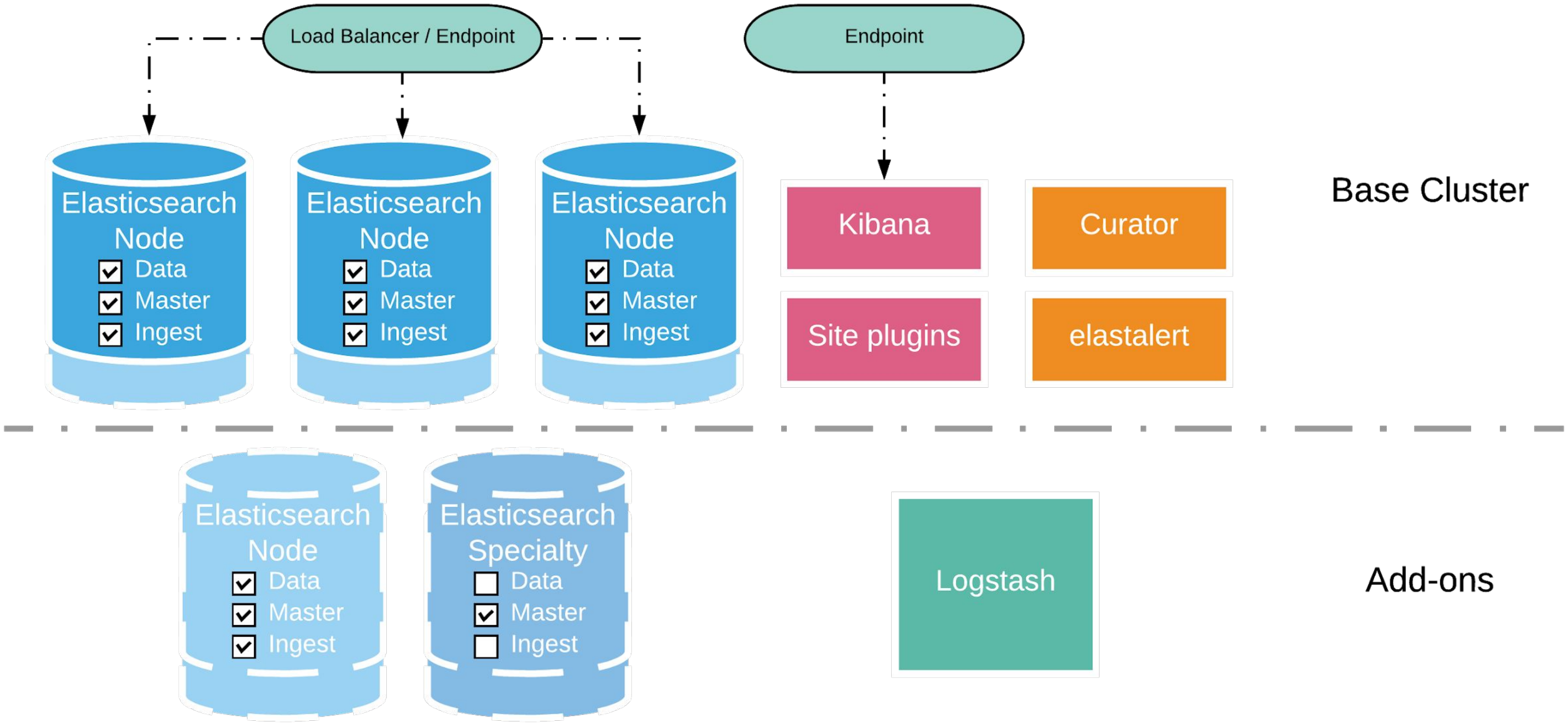
03 Database Administration

- Cluster configuration and plugins
- Perform regular backups
- Running database utilities

04 Service Features

- Safely scale up/out/in/down
- Apply minor and patch updates
- Add-on dashboards and tooling

Elasticsearch Deployment Architecture

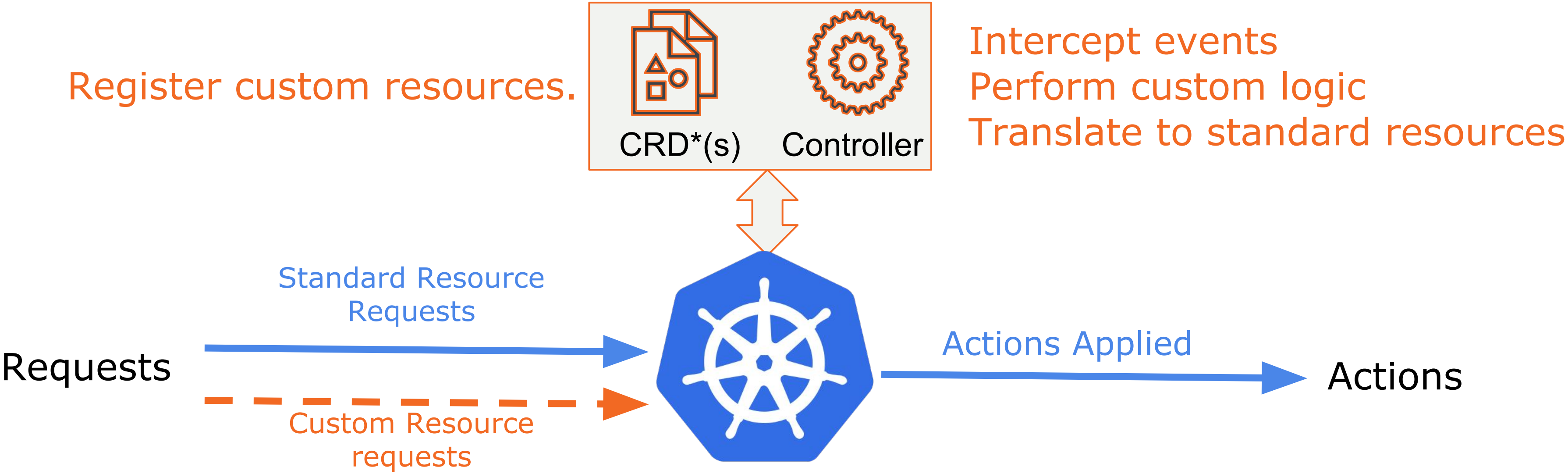


ObjectRocket

Operators: TLDR

Kubernetes Operators

Operators are a way to wrap business logic and manual operations around kubernetes features and components.



Register custom resources.

CRD*(s)

Controller

Intercept events
Perform custom logic
Translate to standard resources

Standard Resource Requests

Requests

Custom Resource requests

Actions Applied

Actions

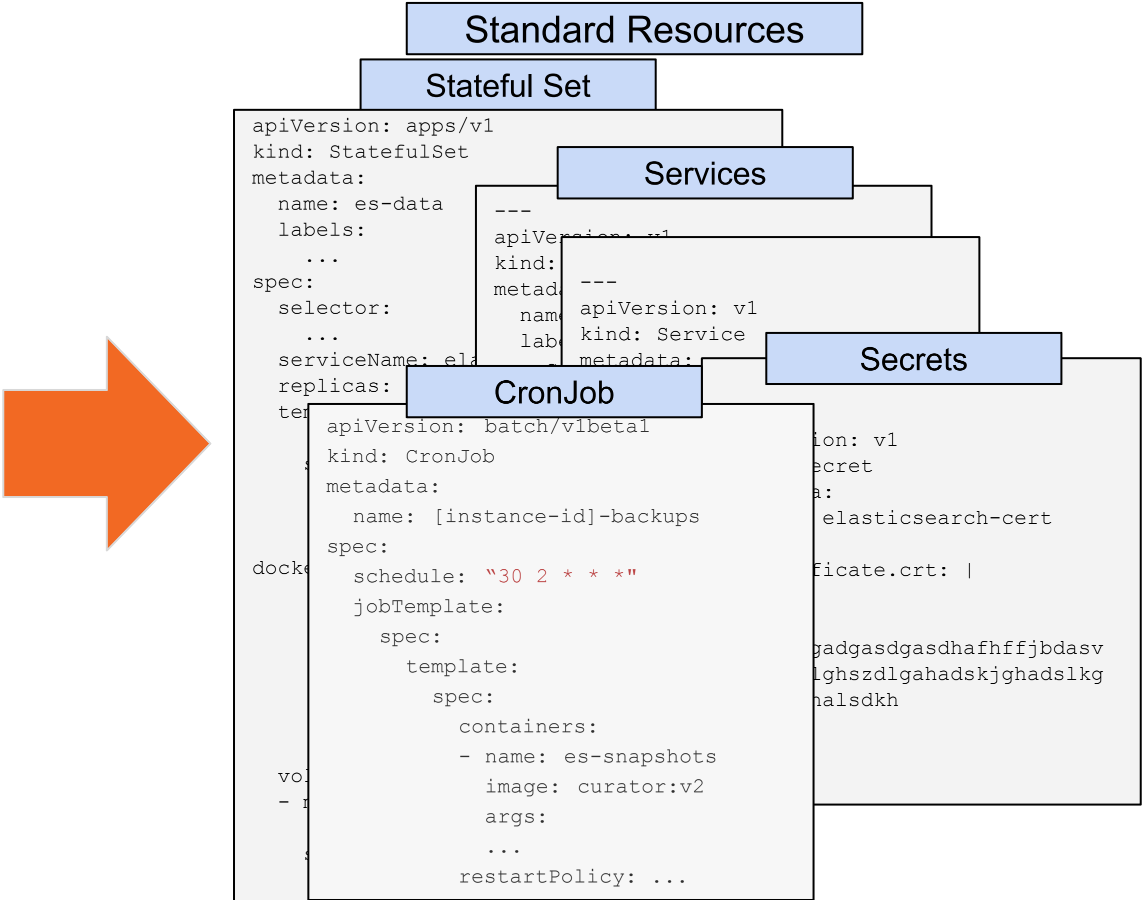
ObjectRocket

CRD = Custom Resource Definition

Operator Custom Resource Example

Custom Resource

```
apiVersion: elasticsearch.objectrocket.com/v1
kind: ElasticsearchMultiRole
metadata:
  name: multirole-deployment
  labels:
    instance-id: "exampleInstanceID"
spec:
  acls:
    - "0.0.0.0/0"
  userlist: |-
    {
      "exampleuser": {"hash": "...", "role":
["admin"]}
    }
  networkHost: 0.0.0.0
  elasticsearchImage:
objectrocket/elasticsearch:oss-6.4.0-v4
  multiRole:
    replicas: 3
    curator:
      curatorImage: "objectrocket/curator:0.0.1"
    javaOpts: "-Xms2048m -Xmx2048m"
    storageConfig:
      storageClass: "standard"
      size: 16
    resourceRestrictions:
      ...
```



```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: es-data
  labels:
    ...
spec:
  selector:
    ...
  serviceName: ela
  replicas:
  ter
docke
  vol
  - n
```

```
---
apiVe
kind:
metad
  name
  labe
```

```
---
apiVersion: v1
kind: Service
metadata:
```

```
ion: v1
ecret
a:
 elasticsearch-cert
ficate.crt: |
gadgasdgasdhafhffjbdasv
lghszdlgahadskjghadslkg
halsdkh
```

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: [instance-id]-backups
spec:
  schedule: "30 2 * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: es-snapshots
            image: curator:v2
            args:
            ...
          restartPolicy: ...
```

Operator Development Options

		Quick Ramp	Control	Latest and Greatest	Verdict
1	Off-the-Shelf Operator	●	◐	◐	- Best way to ramp - Not easily expandable to our end goal ; built 3-4 k8s versions ago
2	Operator Utility Library	◐	◐	◐	- Great for standardizing across operators - Helps bootstrap new operators - Available options didn't have sufficient community buy-in
3	Build from Scratch	○	●	●	- Most up front work - Allowed us to target our specific needs from the start

ObjectRocket

Note: The CoreOS Operator Framework did not yet exist.
We probably would have used that if it did.

Operator Development Options

		Quick Ramp	Control	Latest and Greatest	Verdict
1	Off-the-Shelf Operator	●	◐	◐	• Best way to ramp • Not easily expandable to our end goal, built 3-4 k8s versions ago WINNER: POC
2	Operator Utility Library	◐	◐	◐	• Great for standardizing across operators • Helps bootstrap new operators • Available options didn't have sufficient community buy-in
3	Build from Scratch	○	●	●	• Most up front work • Allows us to target our specific needs from the start WINNER: Production

ObjectRocket

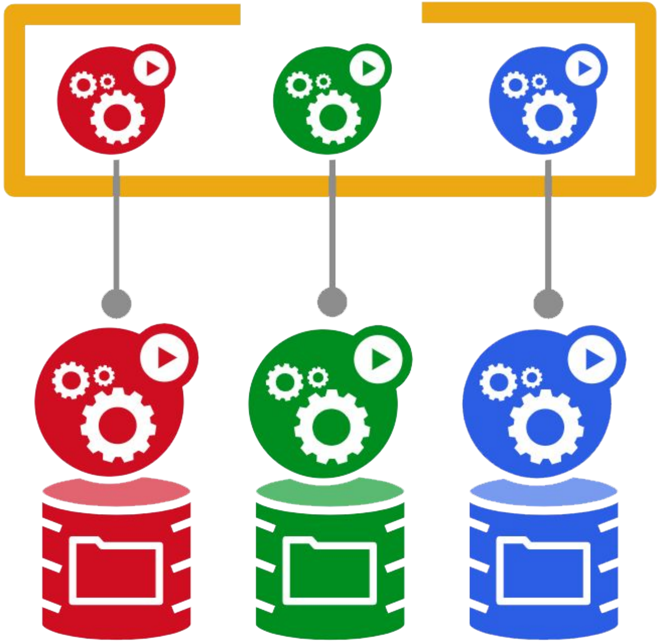
Note: The CoreOS Operator Framework did not yet exist.
We probably would have used that if it did.

Deployments with Persistence: Stateful Sets

Stateful Sets

You can’t have Databases in Kubernetes without them

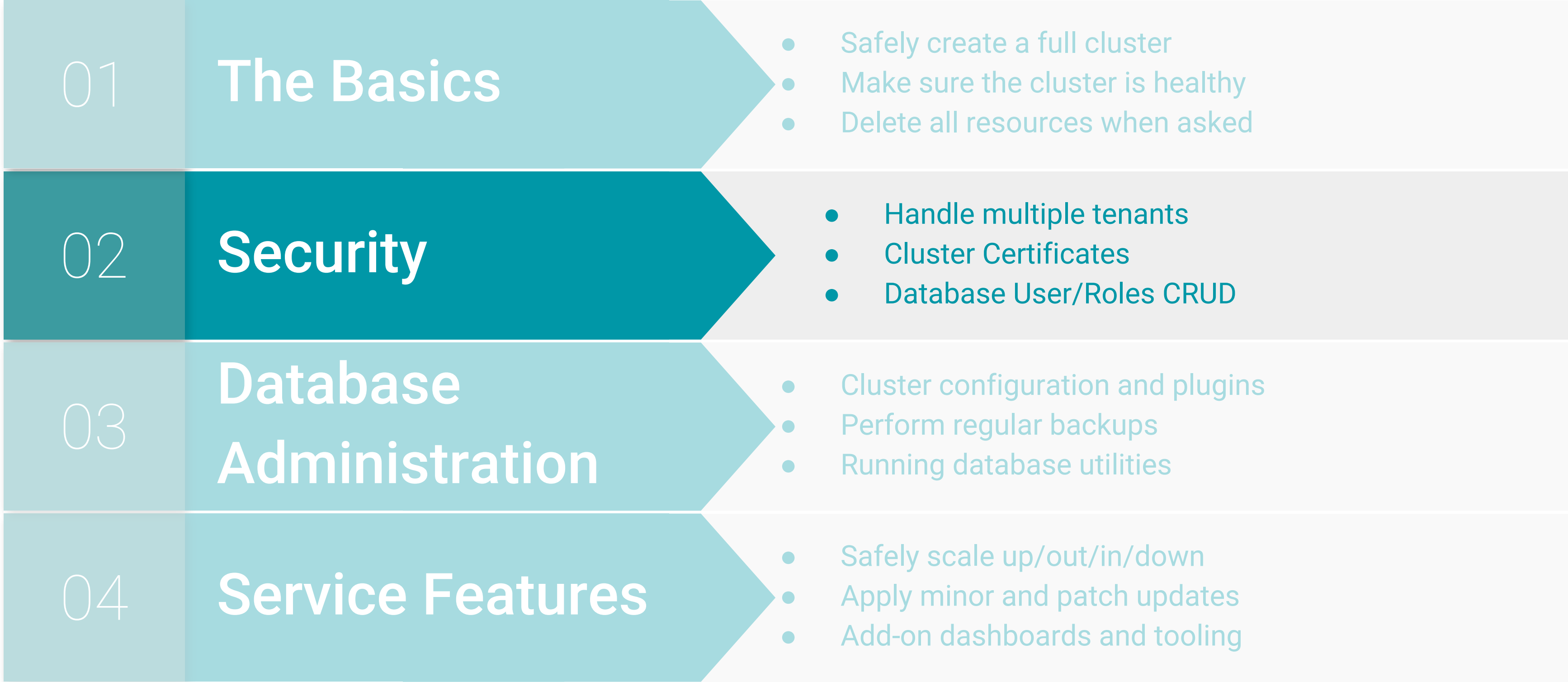
- Unique network identifiers
- Persistent storage
- Ordered, graceful deployment and scaling.
- Ordered, automated rolling updates.



ObjectRocket

Database-as-a-Service: Database Security

What Our Service Must Do



01 The Basics

- Safely create a full cluster
- Make sure the cluster is healthy
- Delete all resources when asked

02 Security

- Handle multiple tenants
- Cluster Certificates
- Database User/Roles CRUD

03 Database Administration

- Cluster configuration and plugins
- Perform regular backups
- Running database utilities

04 Service Features

- Safely scale up/out/in/down
- Apply minor and patch updates
- Add-on dashboards and tooling

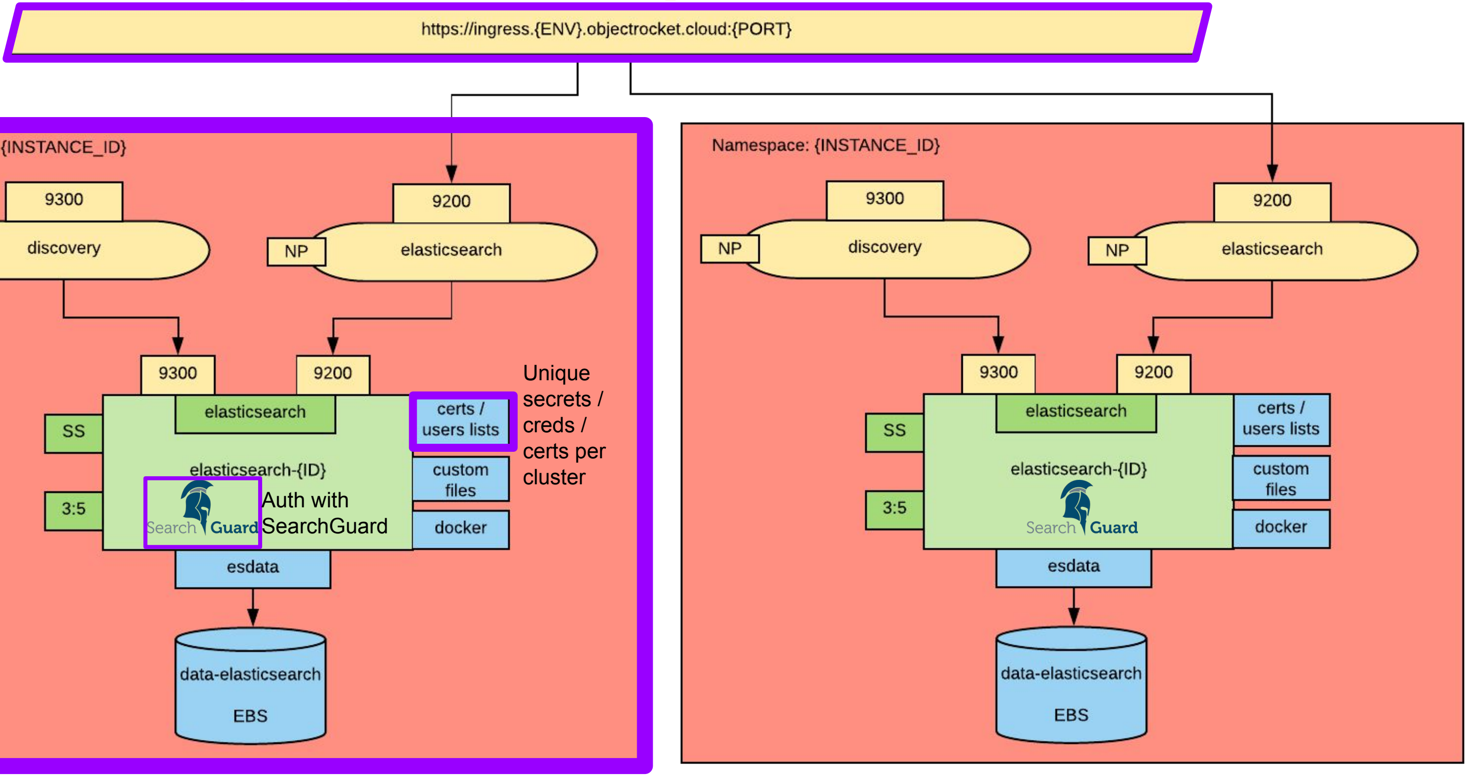
Multi-Tenancy

©2013 Grumpy Cat
GOOD FENCES MAKE GOOD NEIGHBORS
BAD NEIGHBORS MAKE GOOD FERTILIZER

Multi-Tenancy: Namespaces

Tenant Security by Namespace

Ingress ACLs via Envoy proxy



Auth with SearchGuard

Unique secrets / creds / certs per cluster

Namespace isolation and network encryption with Weave Net

ObjectRocket

User Management: Remote Command

ObjectRocket

KubeCon CloudNativeCon

User Management

Controlling the auth plugin that runs on the Elasticsearch containers

- User and role management will need to be executed by the operator
- Apache licensed Elasticsearch does not include an auth implementation

Apache 2.0 licensed Elasticsearch plugin that secures Elasticsearch by providing authentication and authorization.

- **Managed with pre-built command-line utility (sgadmin)**
- Community Edition
  - Internal user database for authentication
  - Role based permissions for authorization
  - Cluster and Index level permissions
  - Live reloads of user database

How to Execute Updates?

Option 1	Option 2	Option 3
Centralized Service	**k8s Job**	**Remote Command**
Standalone service, listening on a queue and executing its own copy of sgadmin	The operator starts up a k8s job which executes its own copy of sgadmin in instance namespace	The operator remotely executes a copy of sgadmin that exists on each member of the cluster

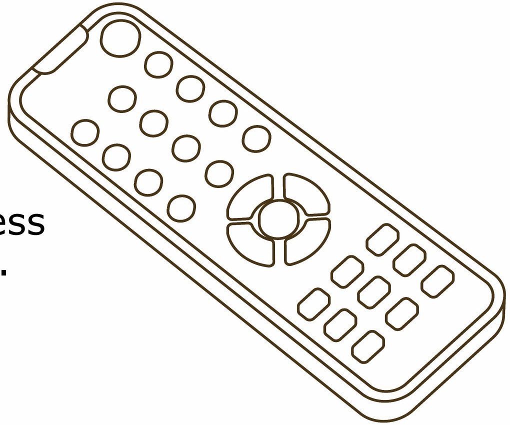
Remote Command

Pros

- **Security:** Uses the k8s API server port; minimizes open ports on the cluster
- **Security:** Leverages unique local admin certs we generate for each cluster
- **Simplicity:** Uses the version of sgadmin installed on each cluster, simplifying the support of multiple versions of the plugin
- **Efficiency:** No extra pods/run on demand means less wasted resources

Cons

- **Temporary multi-process container:** Introduces another process running in the Elasticsearch container breaking docker paradigms.
- **Risk:** client-go’s *remotecommand* library isn’t widely used



Database-as-a-Service: DB Administration

ObjectRocket

KubeCon | CloudNativeCon

What Our Service Must Do

01 **The Basics**

- Safely create a full cluster
- Make sure the cluster is healthy
- Delete all resources when asked

02 **Security**

- Handle multiple tenants
- Cluster Certificates
- Database User/Roles CRUD

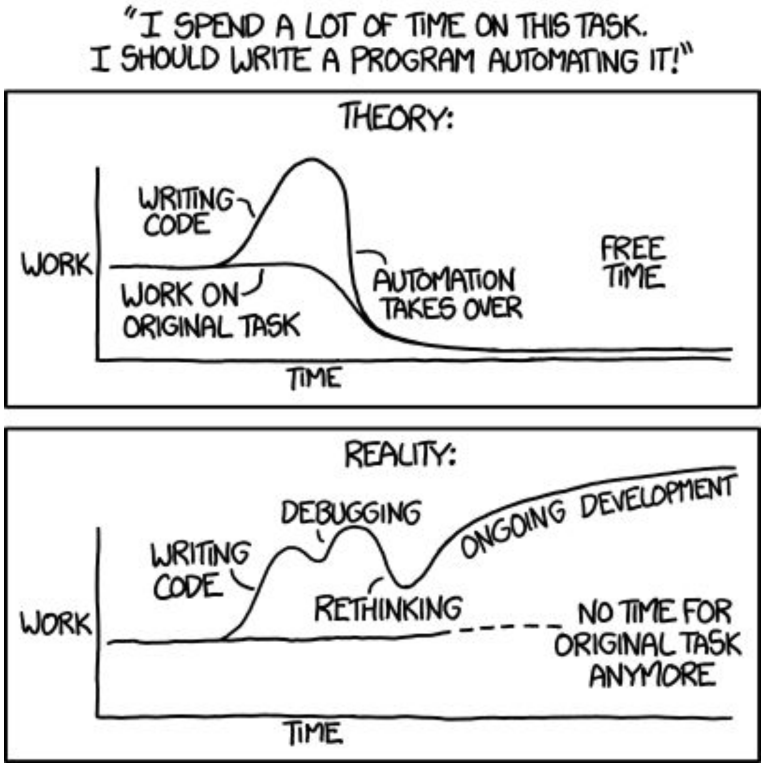
03 **Database Administration**

- Cluster configuration and plugins
- Perform regular backups
- Running database utilities

04 **Service Features**

- Safely scale up/out/in/down
- Apply minor and patch updates
- Add-on dashboards and tooling

Automating Administration Tasks



https://xkcd.com/1319/

ObjectRocket

Customization: Config Maps

ObjectRocket

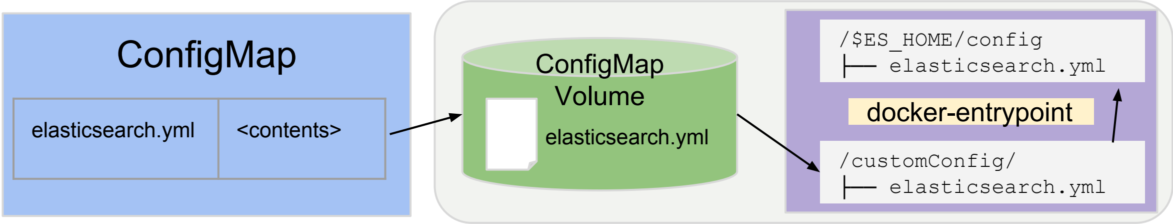
KubeCon CloudNativeCon

Config maps for custom configurations

Image defaults

```
$ES_HOME/config
├── elasticsearch.yml
├── log4j2.properties
├── jvm.options
├── certs
│   ├── ...
├── sgconfig
│   ├── ...
```

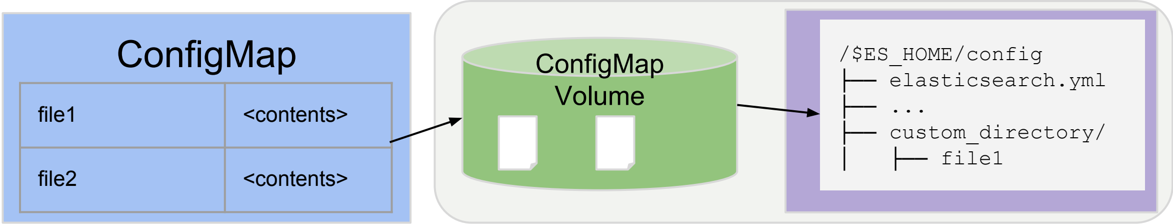
Something special in a specific file (e.g. elasticsearch.yml)



```
/$ES_HOME/config
├── elasticsearch.yml
```

```
/customConfig/
├── elasticsearch.yml
```

New directory for stopwords, synonyms, etc.



```
/$ES_HOME/config
├── elasticsearch.yml
├── ...
├── custom_directory/
│   ├── file1
```

Recurring Tasks: Cron management

ObjectRocket

KubeCon CloudNativeCon

Managing Recurring Tasks

Backups:

- Take a daily backup and retain the last 2 weeks
- **Implementation:** CronJob created with cluster that runs a daily backup

Elasticsearch Curator:

- Take regular actions against indexes (delete, create, aliases, etc.)
- **Implementation:** A CR is passed at any time that:
  - Creates a CronJob with the specified schedule
  - Provides the Curator configuration and action files via ConfigMaps

```
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: [instance-id]-backups
spec:
  schedule: “30 2 * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: es-snapshots
            image: curator:v2
            args:
            ...
          restartPolicy: ...
```

Database-as-a-Service: Service Features

What Our Service Must Do

01 **The Basics**

- Safely create a full cluster
- Make sure the cluster is healthy
- Delete all resources when asked

02 **Security**

- Handle multiple tenants
- Cluster Certificates
- Database User/Roles CRUD

03 **Database Administration**

- Cluster configuration and plugins
- Perform regular backups
- Running database utilities

04 **Service Features**

- Safely scale up/out/in/down
- Apply minor and patch updates
- Add-on dashboards and tooling

ObjectRocket

Services Features

DON'T KNOW IF FEATURE REALLY COMPLEX
OR PRODUCT OWNER REALLY STUPID
quickmeme.com

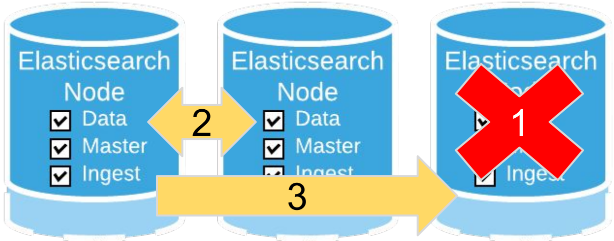
ObjectRocket

Safe Updates: StatefulSets

Elasticsearch Rolling Updates

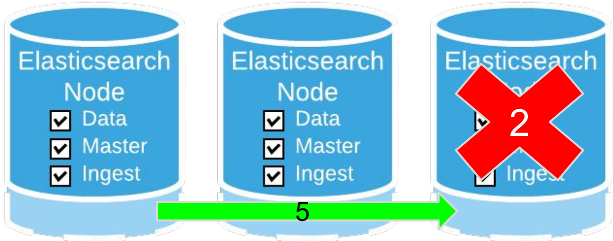
How to ensure that Elasticsearch cluster updates are performed safely and without customer impact?

The Wrong Way



1. Node disappears
2. Replicas promoted / New replicas created on live nodes
3. When node returns to cluster, data reshuffles
4. When green, repeat for every node restart

The Right Way



1. Cluster allocation disabled
2. Node disappears
3. Replicas on other nodes promoted to primary, but no data movement
4. When node returns to cluster, enable allocation
5. Elasticsearch verifies shards on returned node
6. When green, repeat for every node restart

StatefulSet Rolling Update Strategies

RollingUpdate

`.spec.updateStrategy.type = RollingUpdate`

- Automated, rolling update of pods
- Controller **will delete and recreate** each pod
- in the same order as pod termination
- Waits until an updated pod is running and ready prior to moving on

OnDelete Strategy

`.spec.updateStrategy.type = OnDelete`

- Legacy (1.6 and prior) behavior.
- Controller will **not automatically update** each pod
- User manually deletes pods / controller creates new pods
- User manage workflow

Partitioned RollingUpdate

Partitioned RollingUpdates apply changes to pods in reverse order, from {N-1..0}.

Kubernetes offers a couple of ways to safely tear down each pod:

- A grace period (time) is given before the pod is violently shutdown.
- `preHooks`: command/script to execute before the pod is violently shutdown.

Partitioned RollingUpdate enables control over when changes are applied to a member of the StatefulSet.

- Only pods with an ordinal >= the partition value will be updated when the StatefulSet's `.spec.template` is updated.
- Pods with an ordinal < the partition will not be updated, even if they are deleted

Complex Updates: State Management

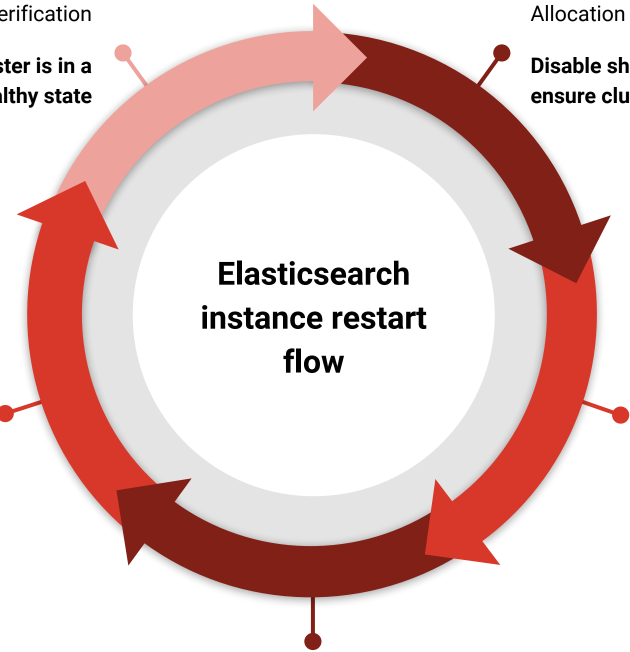
What do we mean by ‘State Management’?

For most Elasticsearch workflows Kubernetes manages:

- Pod Operations
- Pod Health
- Volume state

StatefulSets allow us to keep track of which members of the cluster have the new configuration

By leveraging labels we are able to manage what part of our process has been applied to an instance



Cluster Verification

Verify the cluster is in a healthy state

Allocation Management

Disable shard allocation and ensure cluster is healthy

Partition Management

Update ordinal to include the next member of the cluster

Pod/Cluster Health

Ensure Pod and the Cluster is healthy

Allocation Management

Enable shard allocation

ObjectRocket

StateMachine

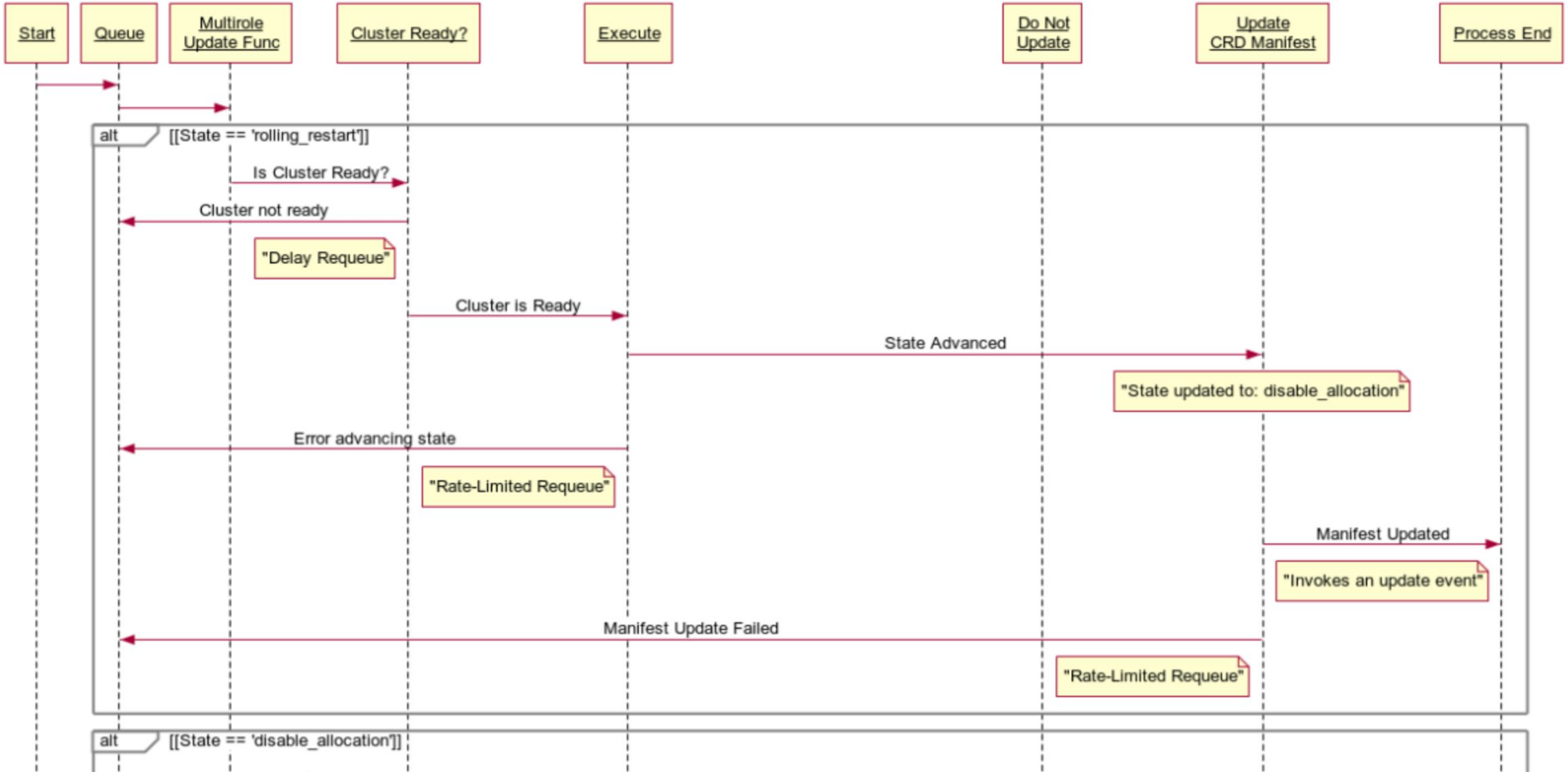
State Labels

```
RollingUpdateState      = "rolling_update"
BeginState              = "begin_state"
PodsReadyState          = "pods_ready"
SGInitState             = "sg_init"
FinalAddState           = "final_add_state"
StateCompleted          = "completed"
RollingRestartState     = "rolling_restart"
UpdateUserlistState     = "update_userlist"
UserlistRollback        = "userlist_rollback"
ForceRollingRestart     = "force_rolling_restart"
RollbackResourceStep    = "rollback_resource"
UpdateConfigFilesStep   = "update_config_files"
buildingState           = "building"
updatingState           = "updating"
disableAllocationState  = "disable_allocation"
```

StateMap Generator

```
func generateRollingRestartStateMap(numberNodes int) map[string]string {
        logrus.Info("Creating Safe Update State Map, number of nodes: ", numberNodes)
        stateMachine := make(map[string]string)
        state := RollingRestartState
        for i := numberNodes - 1; i > -1; i-- {
                newApplyPartitionState := fmt.Sprintf("%s%s%d", applyPartitionState, stateDelimiter, i)
                newVerifyPartitionApplied := fmt.Sprintf("%s%s%d", verifyPartitionState, stateDelimiter, i)
                newDisableAllocation := fmt.Sprintf("%s%s%d", disableAllocationState, stateDelimiter, i)
                newEnableAllocationState := fmt.Sprintf("%s%s%d", enableAllocationState, stateDelimiter, i)
                stateMachine[state] = newDisableAllocation
                stateMachine[newDisableAllocation] = newApplyPartitionState
                stateMachine[newApplyPartitionState] = newVerifyPartitionApplied
                stateMachine[newVerifyPartitionApplied] = newEnableAllocationState
                state = newEnableAllocationState
        }
        stateMachine[state] = finalState
        return stateMachine
}
```

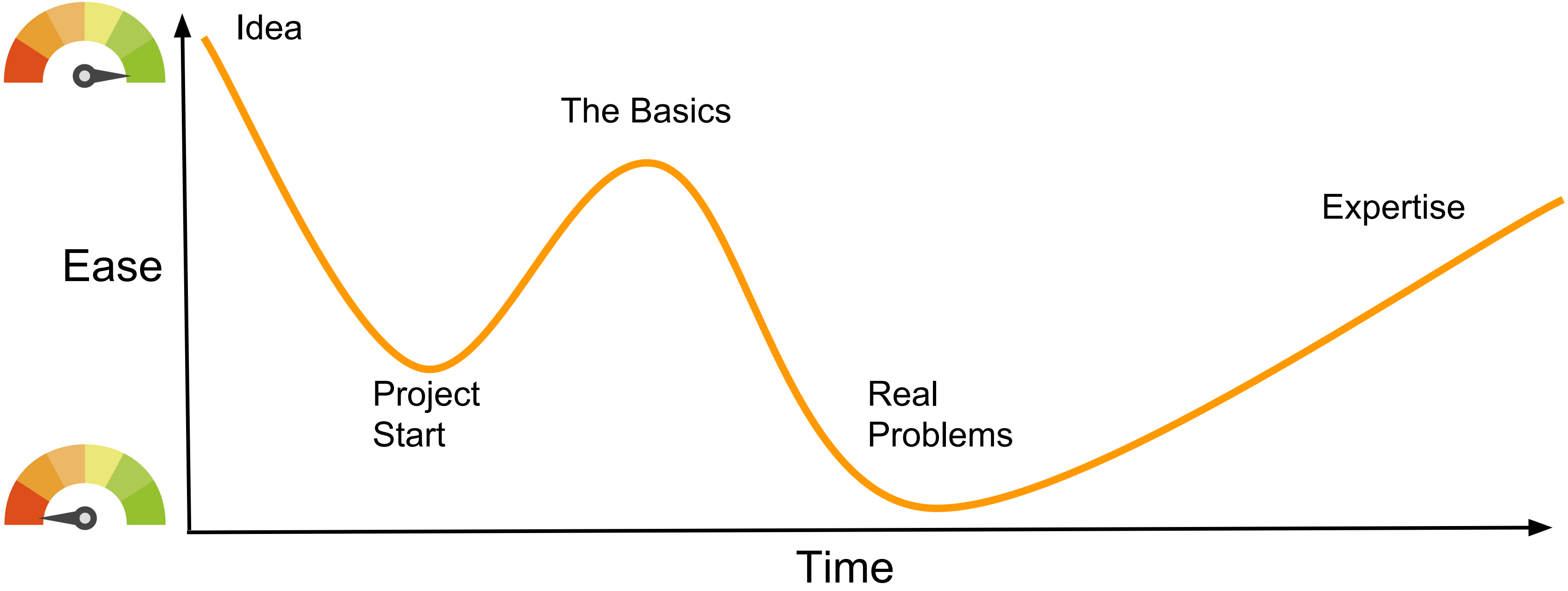
Example State Flow



ObjectRocket

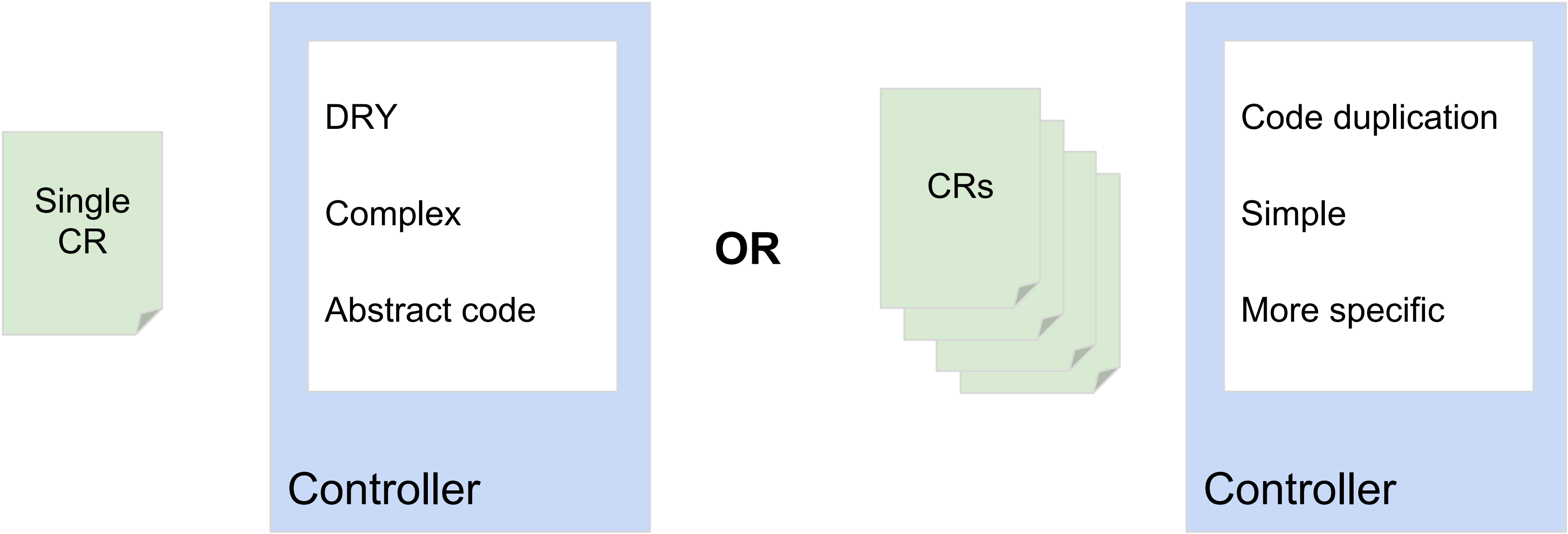
What We Learned

Our Kubernetes Learning Curve



ObjectRocket

DRY vs. Complexity



Single CR

DRY

Complex

Abstract code

Controller

OR

CRs

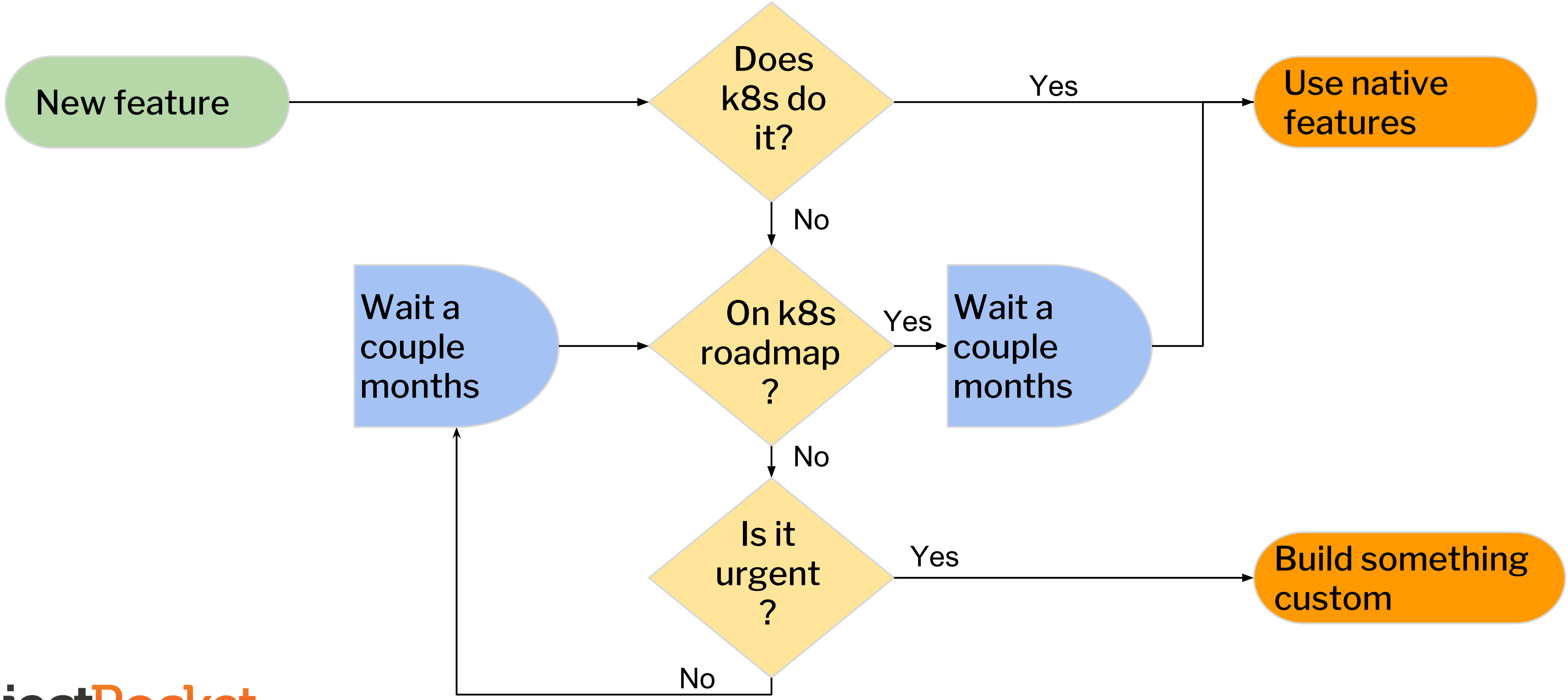
Code duplication

Simple

More specific

Controller

Velocity of Kubernetes



ObjectRocket

ObjectRocket
a rackspace company

Try Our Evaluation Program

Get **free database resources** on our new Kubernetes-based platform on AWS in the region of your choice.

MongoDB | Elasticsearch | Redis

Your experience and feedback during the program will play a critical role in helping us build the database platform you need.

https://www.objectrocket.com/or-on-k8s/

KubeCon CloudNativeCon

Check Out the Code

We’re going to open source our Elasticsearch operator

- It will be Apache 2.0 licensed
- It will arrive in Q1 of 2019

https://github.com/objectrocket

Coming Soon

We’re Hiring!

Engineering & Sales

Apply at **ObjectRocket.com/Careers**

ObjectRocket

KubeCon

CloudNativeCon