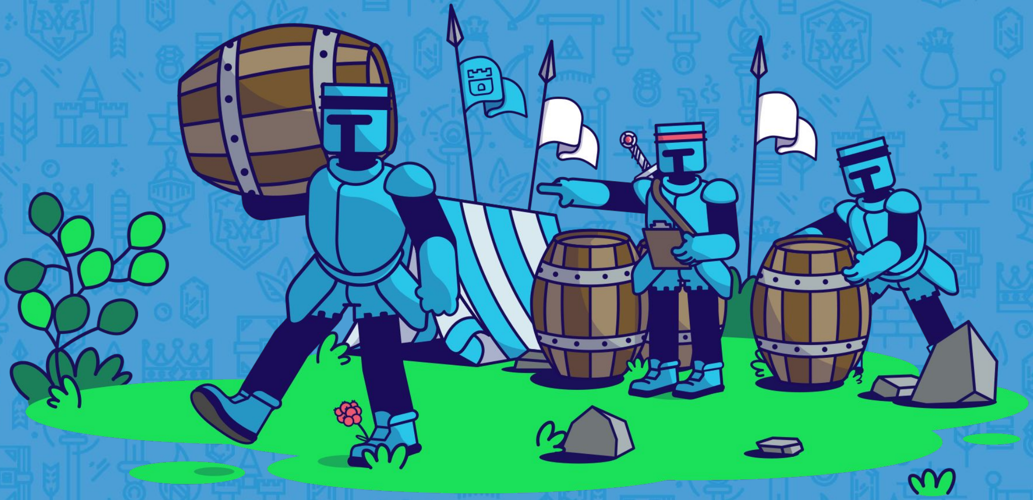
Rook Deep Dive

Jared Watts
Rook Senior Maintainer
Upbound Founding Engineer

https://rook.io/
https://github.com/rook/rook



What is Rook?

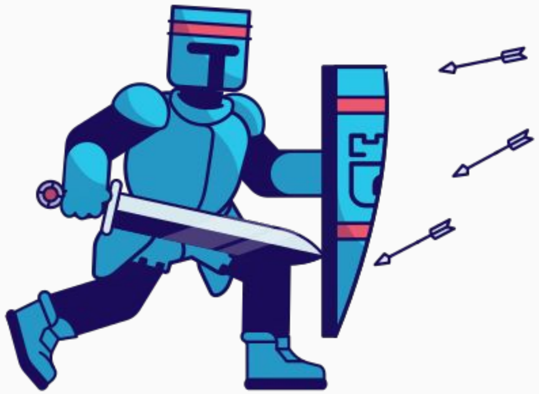
- Cloud-Native Storage Orchestrator
- Extends Kubernetes with custom types and controllers
- Automates deployment, bootstrapping, configuration, provisioning, scaling, upgrading, migration, disaster recovery, monitoring, and resource management
- Framework for many storage providers and solutions
- Open Source (Apache 2.0)
- Hosted by the Cloud-Native Computing Foundation (CNCF)

Rook Framework for Storage Solutions

- Rook is more than just a collection of Operators and CRDs
- **Framework** for storage providers to integrate their solutions into cloud-native environments
    - Storage resource normalization
    - Operator patterns/plumbing
    - Common policies, specs, logic
    - Testing effort
- Ceph, CockroachDB, Minio, NFS, Cassandra, EdgeFS, and more...

New Rook Operators

Apache Cassandra



Nexenta EdgeFS

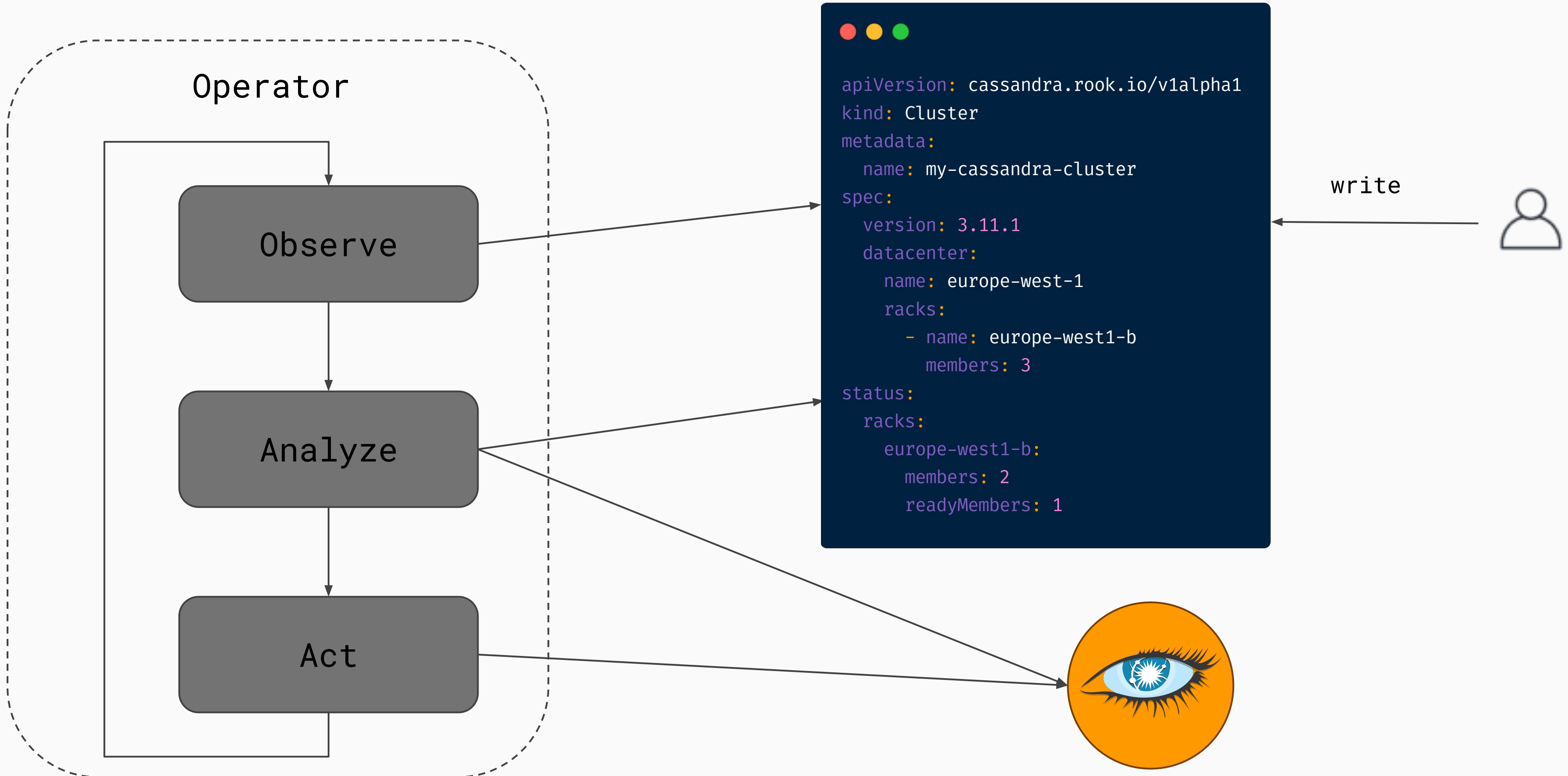
Apache Cassandra

- Cassandra is an open-source, distributed NoSQL database that can handle large amounts of data on commodity hardware
  - Highly available with no single point of failure
  - Horizontal scaling
- Great candidate for building an operator to manage it
- Designed and implemented by **Yannis Zarkadas** as part of his graduate thesis, sponsored by **Arrikto**
  - Github: **@yanniszark**

Can StatefulSets manage a reliable Cassandra cluster?

- Confined to 1 rack - no cluster hierarchy, multiple racks, etc.
- Safe scale down procedure is complicated and time consuming
  - nodetool decommission
  - stream data
  - member leaves
- Issues with seeds, multi-zone deployments, loss of persistence, backups/restores, extensibility
- Cassandra needs an operator!

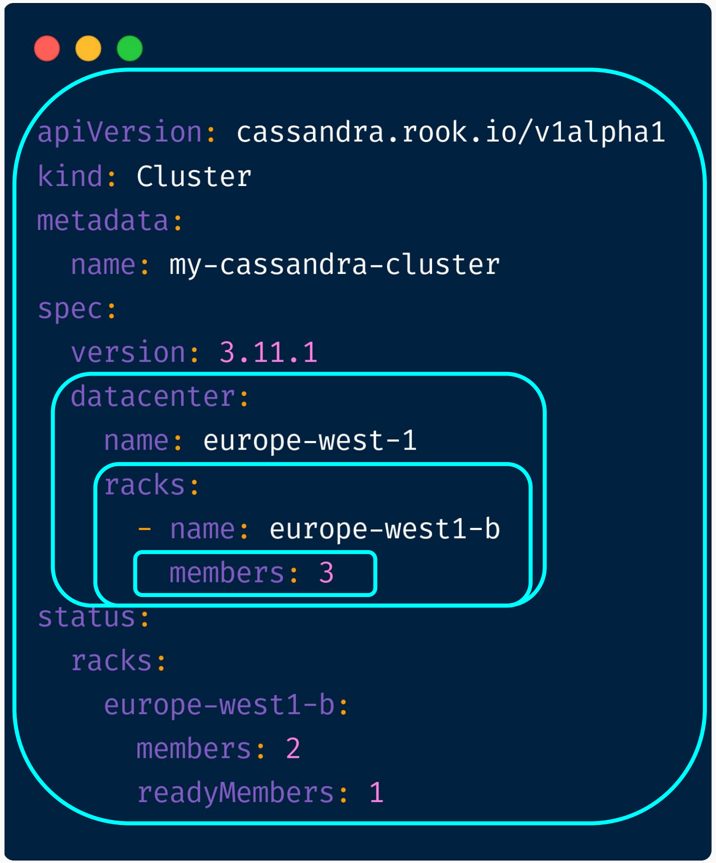
Cassandra Operator



```
apiVersion: cassandra.rook.io/v1alpha1
kind: Cluster
metadata:
  name: my-cassandra-cluster
spec:
  version: 3.11.1
  datacenter:
    name: europe-west-1
    racks:
      - name: europe-west1-b
        members: 3
status:
  racks:
    europe-west1-b:
      members: 2
      readyMembers: 1
```

Cassandra as Kubernetes Resources

cassandra	kubernetes
Member	Pod
Rack	StatefulSet
Datacenter	StatefulSets
Cluster	Cluster Custom Resource



```yaml
apiVersion: cassandra.rook.io/v1alpha1
kind: Cluster
metadata:
  name: my-cassandra-cluster
spec:
  version: 3.11.1
  datacenter:
    name: europe-west-1
    racks:
      - name: europe-west1-b
        members: 3
status:
  racks:
    europe-west1-b:
      members: 2
      readyMembers: 1
```

Cassandra Operator Features

- Cluster creation and bootstrapping
- Auto scaling - grow the cluster with new members
- Scale-down (safely!) - decommission, stream data, leave
- Handle failed node and replace with new members
- More to come...

EdgeFS

- Natively designed to be global storage
- Based on immutable blocks similar to `Git`
  - modifications are globally unique and versioned
  - modification results in a new identity
  - caches are always in a consistent state
  - allows global fault tolerance, global scalability
- Segmented storage - stitching clouds into one single geo-namespace
  - ISGW - Inter-Segment GateWay

EdgeFS - metadata only

- Mode to (initially) transfer metadata only across segments
- Full file listings and info are available globally *fast*
- Data chunks will be fetched lazily (on demand)
- Critical for enabling a globally remote client to start consuming data as soon as it is created

EdgeFS - deduplication and recovery

- Global deduplication - multiple identical chunks are only stored once
- Built in disaster recovery: lost data chunks can be recovered from remote segments transparently
- Client sees a temporary loss in throughput, but no errors while fetching from remote
- Local site cache is repopulated with recovered data

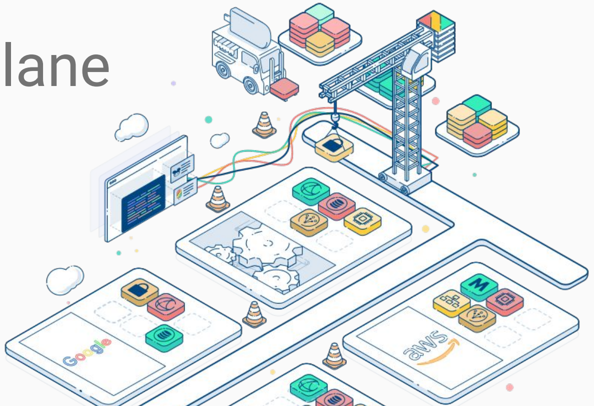
Software runs in many environments

- Kubernetes runs great everywhere
- Rook is great on-premises
- Cloud provider managed services are great in the cloud
- So, can we have both?
- Manage our infrastructure, platform services, storage, and applications all from one place: `kubectl`
- Portable abstractions for all our storage needs

Power of Portability

- Power of **choice** - cost, features, availability, compliance, etc.
- Take our **data** wherever Kubernetes goes
- **Pod** and **Volume** abstractions enables portability
  - What about databases, buckets, message queues, data pipelines, etc.?
- **Crossplane** - open source multicloud control plane
  - https://crossplane.io/

Crossplane

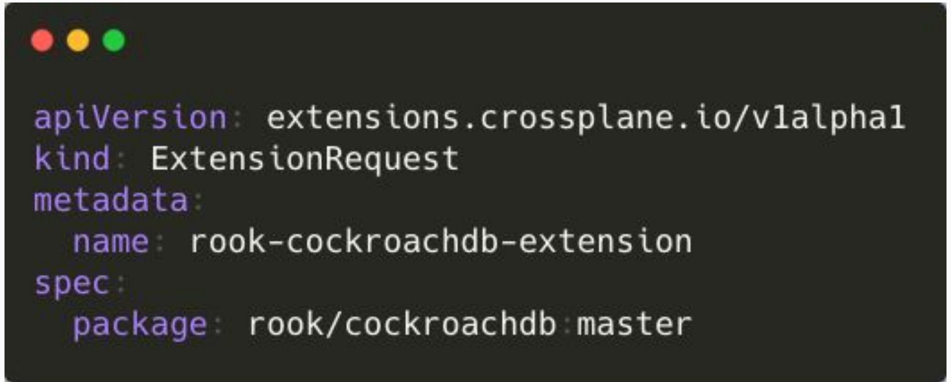


Dynamic provisioning for new storage types

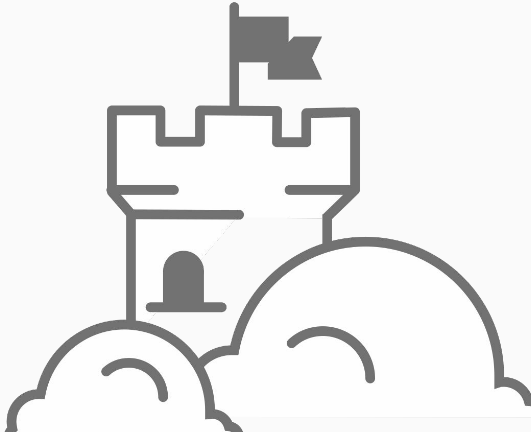
- Similar pattern to **StorageClass** and **PersistentVolumeClaim**
- **ResourceClass** - a “blueprint” created by the administrator
  - all environment specifics to create a “class” of storage
  - *Fast, Standard, Cheap, etc.*
- **ResourceClaim** - user defined, expresses the general need for a type of storage
- Storage is created on demand as needed
- Enables **portability** and the power of choice
- ***Write once, run anywhere***

Extending Crossplane

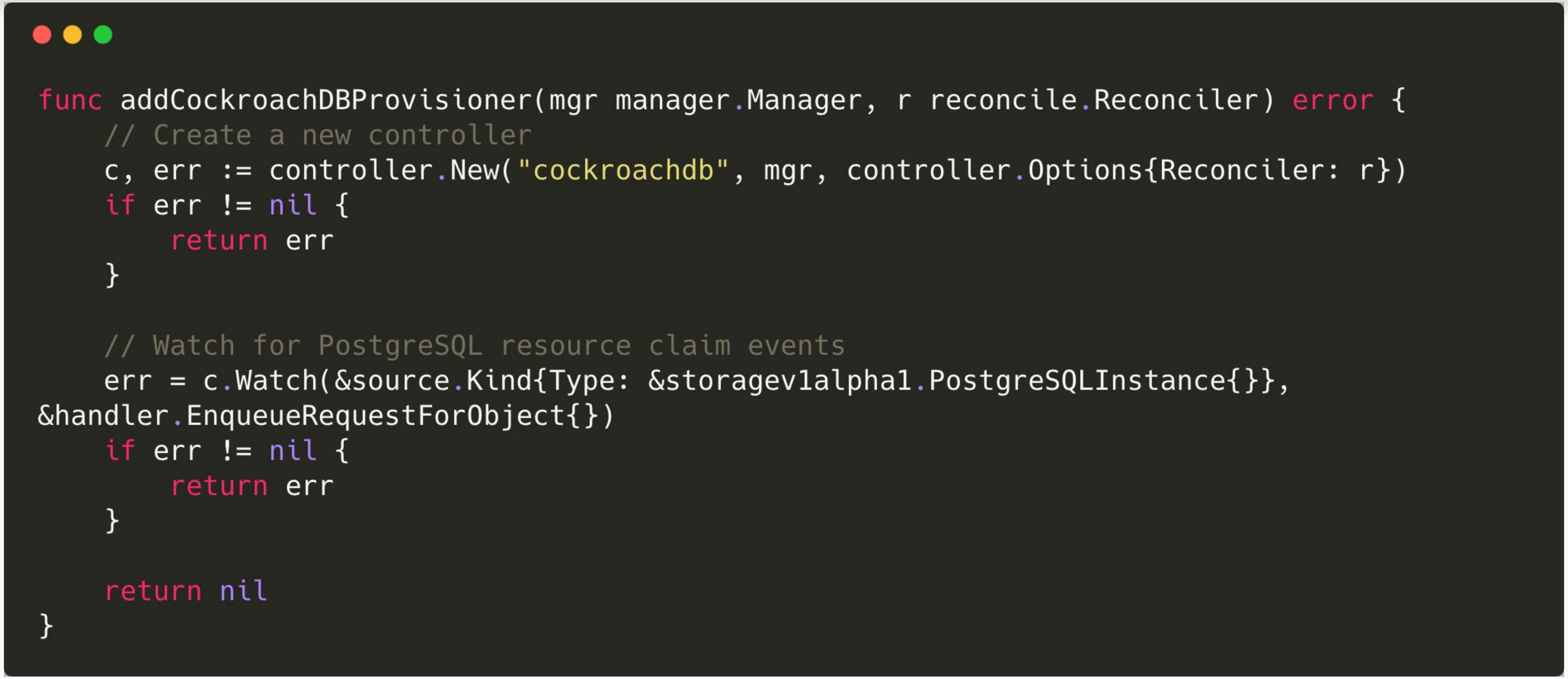
- Add new out-of-tree functionality to Crossplane with an Extension
- Let's extend Crossplane with Rook!
- Now we can dynamically provision new storage types in-cluster (on-premises) too



```yaml
apiVersion: extensions.crossplane.io/v1alpha1
kind: ExtensionRequest
metadata:
  name: rook-cockroachdb-extension
spec:
  package: rook/cockroachdb:master
```

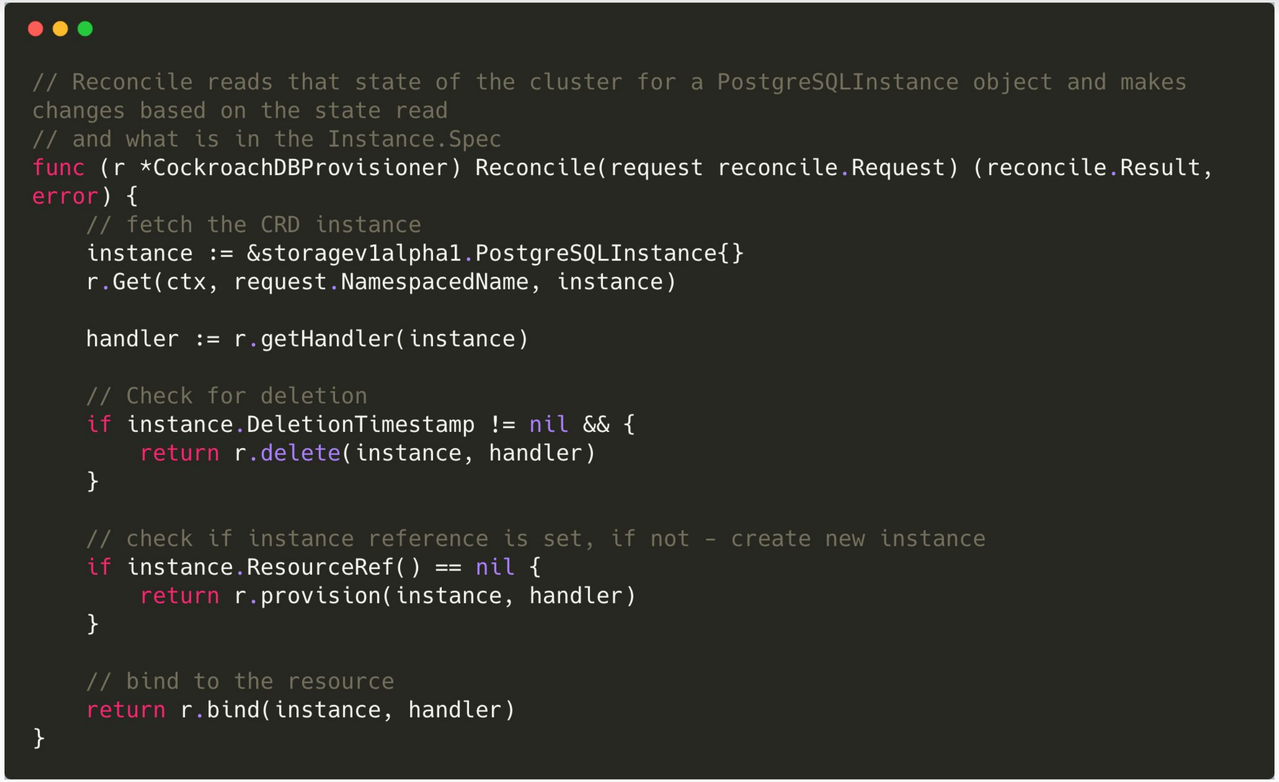


CockroachDB Dynamic Provisioner - Observe



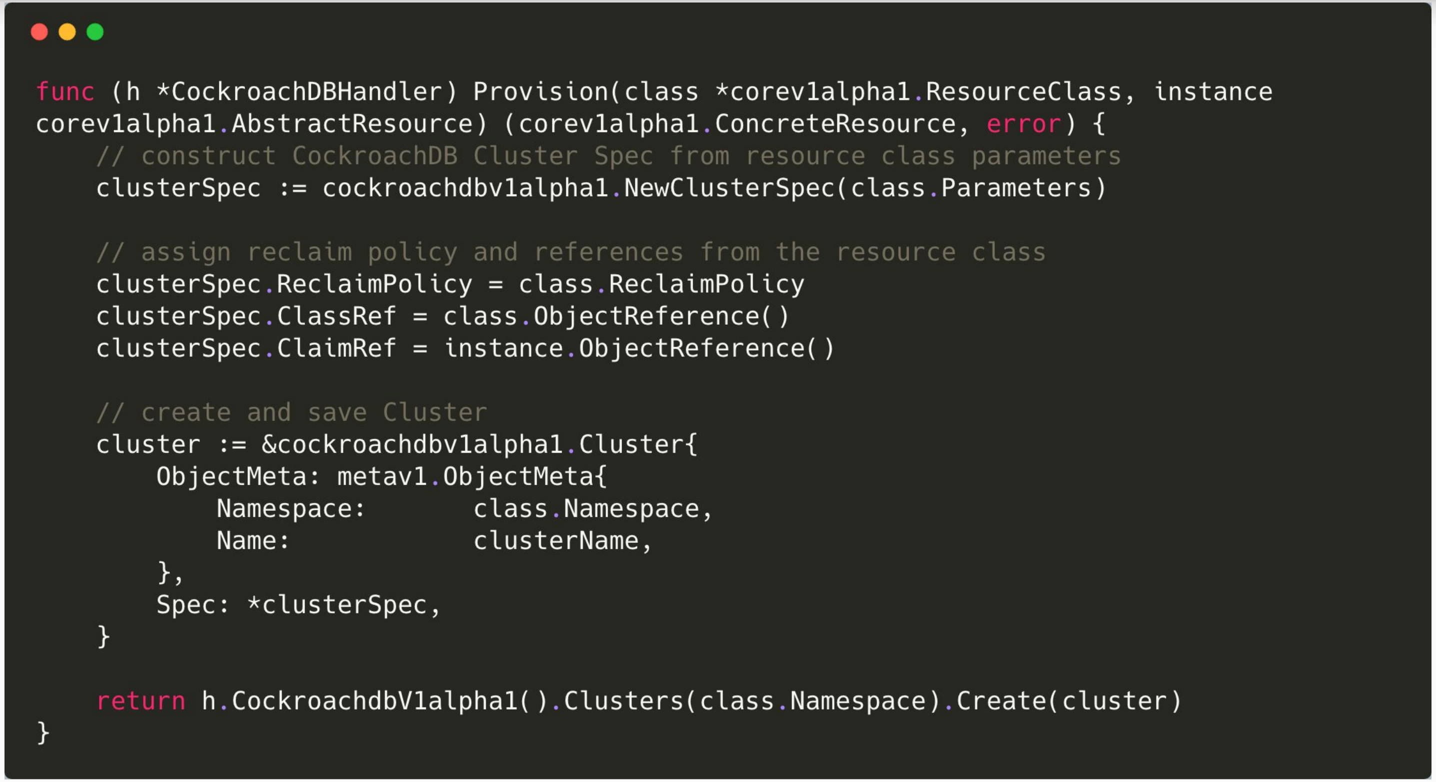
```go
func addCockroachDBProvisioner(mgr manager.Manager, r reconcile.Reconciler) error {
    // Create a new controller
    c, err := controller.New("cockroachdb", mgr, controller.Options{Reconciler: r})
    if err != nil {
        return err
    }

    // Watch for PostgreSQL resource claim events
    err = c.Watch(&source.Kind{Type: &storagev1alpha1.PostgreSQLInstance{}},
&handler.EnqueueRequestForObject{})
    if err != nil {
        return err
    }

    return nil
}
```

CockroachDB Dynamic Provisioner - Analyze



```go
// Reconcile reads that state of the cluster for a PostgreSQLInstance object and makes
changes based on the state read
// and what is in the Instance.Spec
func (r *CockroachDBProvisioner) Reconcile(request reconcile.Request) (reconcile.Result,
error) {
	// fetch the CRD instance
	instance := &storagev1alpha1.PostgreSQLInstance{}
	r.Get(ctx, request.NamespacedName, instance)

	handler := r.getHandler(instance)

	// Check for deletion
	if instance.DeletionTimestamp != nil && {
		return r.delete(instance, handler)
	}

	// check if instance reference is set, if not - create new instance
	if instance.ResourceRef() == nil {
		return r.provision(instance, handler)
	}

	// bind to the resource
	return r.bind(instance, handler)
}
```

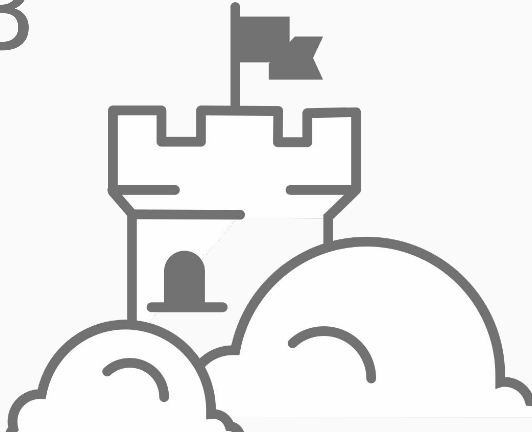
CockroachDB Dynamic Provisioner - Act



```go
func (h *CockroachDBHandler) Provision(class *corev1alpha1.ResourceClass, instance
corev1alpha1.AbstractResource) (corev1alpha1.ConcreteResource, error) {
    // construct CockroachDB Cluster Spec from resource class parameters
    clusterSpec := cockroachdbv1alpha1.NewClusterSpec(class.Parameters)

    // assign reclaim policy and references from the resource class
    clusterSpec.ReclaimPolicy = class.ReclaimPolicy
    clusterSpec.ClassRef = class.ObjectReference()
    clusterSpec.ClaimRef = instance.ObjectReference()

    // create and save Cluster
    cluster := &cockroachdbv1alpha1.Cluster{
        ObjectMeta: metav1.ObjectMeta{
            Namespace:        class.Namespace,
            Name:             clusterName,
        },
        Spec: *clusterSpec,
    }

    return h.CockroachdbV1alpha1().Clusters(class.Namespace).Create(cluster)
}
```

Demo

Extending Crossplane with Rook-CockroachDB



What did we cover today?

- Rook is a cloud-native storage orchestrator
- Framework to create storage operators that deploy, configure, and manage many storage solutions in Kubernetes
- Apache Cassandra - scale-up, scale-down, node failover
- EdgeFS - globally distributed storage in a single geo-namespace
- Extend Crossplane with new multi-cloud functionality
- Dynamically provision all sorts of storage types in the cloud and on-premises with Crossplane & Rook

How to get involved?

- Contribute to Rook and Crossplane
  - https://rook.io/
  - https://crossplane.io/
- Slack
  - https://slack.rook.io/
  - https://slack.crossplane.io/
- Twitter - @rook_io & @crossplane_io
- Forums - rook-dev & crossplane-dev on google groups
- Community Meetings

Rook sessions at Kubecon

Meet the Maintainers

Wednesday, **12:30** @ CNCF Answer Bar

Keep the Space Shuttle Flying: Writing Robust Operators

Wednesday, **15:55** @ Hall 8.1 G2

Rook, Ceph, and ARM: A Caffeinated Tutorial

Wednesday, **16:45** @ Hall 8.0 D2

Thank you!

https://rook.io/

https://crossplane.io/