

Crossplane

Introduction & Deep Dive

<https://crossplane.io>

Nic Cope, Bob Haddleton, Matthias Luebken, Jared Watts





What is Crossplane?

- **Framework** for building cloud native **control planes**
 - No need to write any code
- Cloud providers have been managing their infrastructure with control planes for years
 - Crossplane helps you build your own - with your own **opinions**
- Extensible backend to manage **any infrastructure** in **any environment**
- Configurable frontend to expose **declarative APIs** (abstractions) for developer **self-service**



CNCF Project for the Community

- Crossplane is a neutral place for vendors and individuals to come together in enabling control planes
- Launched in Dec 2018 by creators of CNCF graduated Rook project
 - Accepted into Sandbox in June 2020
 - First major “stable” milestone [v1.0 released](#) in Dec 2020
 - Moved to Incubation September 2021
 - [v1.10](#) released last week
 - Aiming for Graduation early ‘23 - we need your help!

Project and Community Stats



7,000+

Stars

2x Increase YoY

60,000+

Contributions

2x Increase YoY

1,200+

Contributors

5x Increase YoY



6,000+

Followers

2x increase YoY



7,000+

Members

5x increase YoY



28M+

Pulls

10x increase YoY

The Basics

Managed Resources

Managed Resources Example: AWS

Networking

Databases

Kubernetes Clusters

IAM

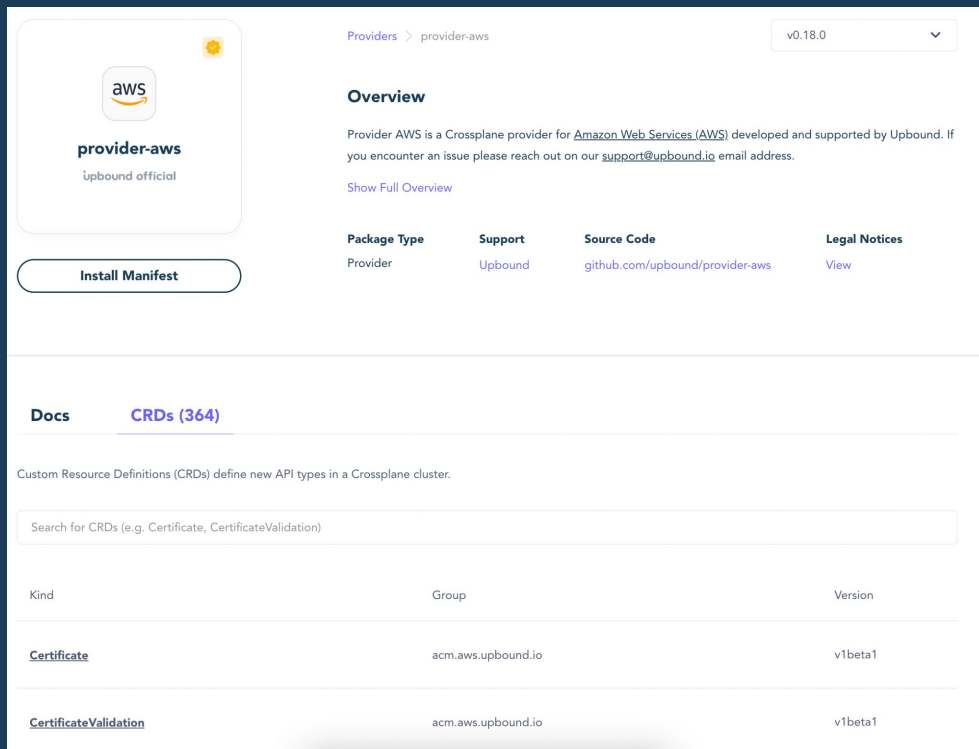
VMs

Message Queues

Caches

Certificates

...and much more...




The screenshot shows the Upbound Marketplace page for the **provider-aws**. The page includes a header with the Upbound logo and a version dropdown set to v0.18.0. The main content area is divided into sections: Overview, Package Type, Support, Source Code, and Legal Notices. The Overview section describes the provider as a Crossplane provider for Amazon Web Services (AWS) and provides a link to support@upbound.io. The Package Type section lists the provider as 'Provider'. The Source Code section provides a link to the GitHub repository. The Legal Notices section provides a link to view legal notices. Below these sections is a 'Docs' section with a link to 'CRDs (364)'. The CRDs section contains a search bar and a table listing Custom Resource Definitions (CRDs).

Kind	Group	Version
Certificate	acm.aws.upbound.io	v1beta1
CertificateValidation	acm.aws.upbound.io	v1beta1


Managed Resources

```
apiVersion: s3.aws.crossplane.io/v1beta1
kind: Bucket
metadata:
  name: crossplane-deepdive-demo-bucket
spec:
  forProvider:
    acl: private
    locationConstraint: eu-west-1
    paymentConfiguration:
      payer: BucketOwner
    versioningConfiguration:
      status: Enabled
  tagging:
    tagSet:
      - key: Name
        value: CrossplaneDeepDiveDemoBucket
```

Bucket overview

AWS Region	Amazon Resource Name (ARN)	Creation date
EU (Ireland) eu-west-1	 arn:aws:s3:::crossplane-deepdive-demo-bucket	May 16, 2022, 13:33:42 (UTC-05:00)

Tags (1)

Track storage cost or other criteria by tagging your bucket. [Learn more](#) 

Key	Value
Name	CrossplaneDeepDiveDemoBucket

Managed Resources

Status:
At Provider:
Arn: arn:aws:s3:::crossplane-deepdive-demo-bucket

Status contains values returned from the remote API and the condition of the resources.

Events:

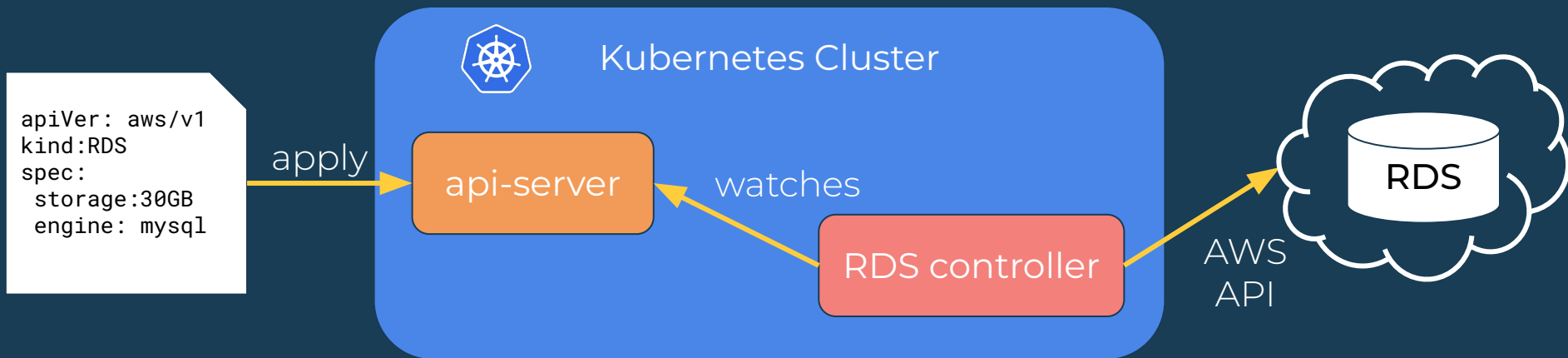
Type	Reason	Age	From
Message			
----	-----	----	----

Normal	CreatedExternalResource	6m8s	managed/bucket.s3.aws.crossplane.io
Successfully requested creation of external resource			

Managed Resources
Generate K8s Events

Managed Resource Reconciliation

- Controllers reconcile Managed Resources with cloud provider and on-prem APIs (e.g., GCP, AWS, or any API)



Control Plane Internal Stack

Controller

Controller

Controller

Controller

Custom Logic

Crossplane Runtime



Manage External APIs
Create/Update/Delete

Controller Runtime



Event, Watch, Request,
Reconciliation

Kubernetes API Machinery



CRDs, OpenAPI,
Persistence (etcd)

Kubernetes Runtime



Run Workloads, Ingress,
RBAC

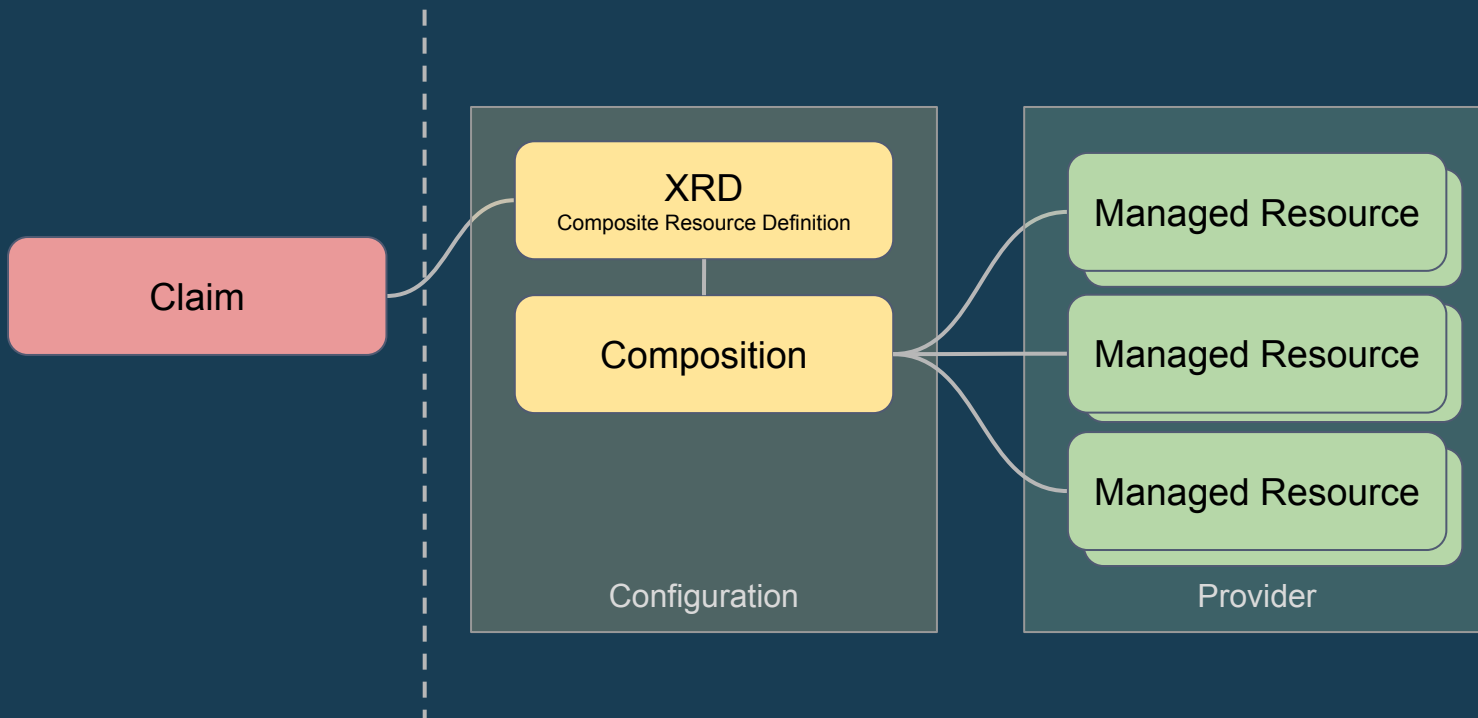
Building Your Control Plane

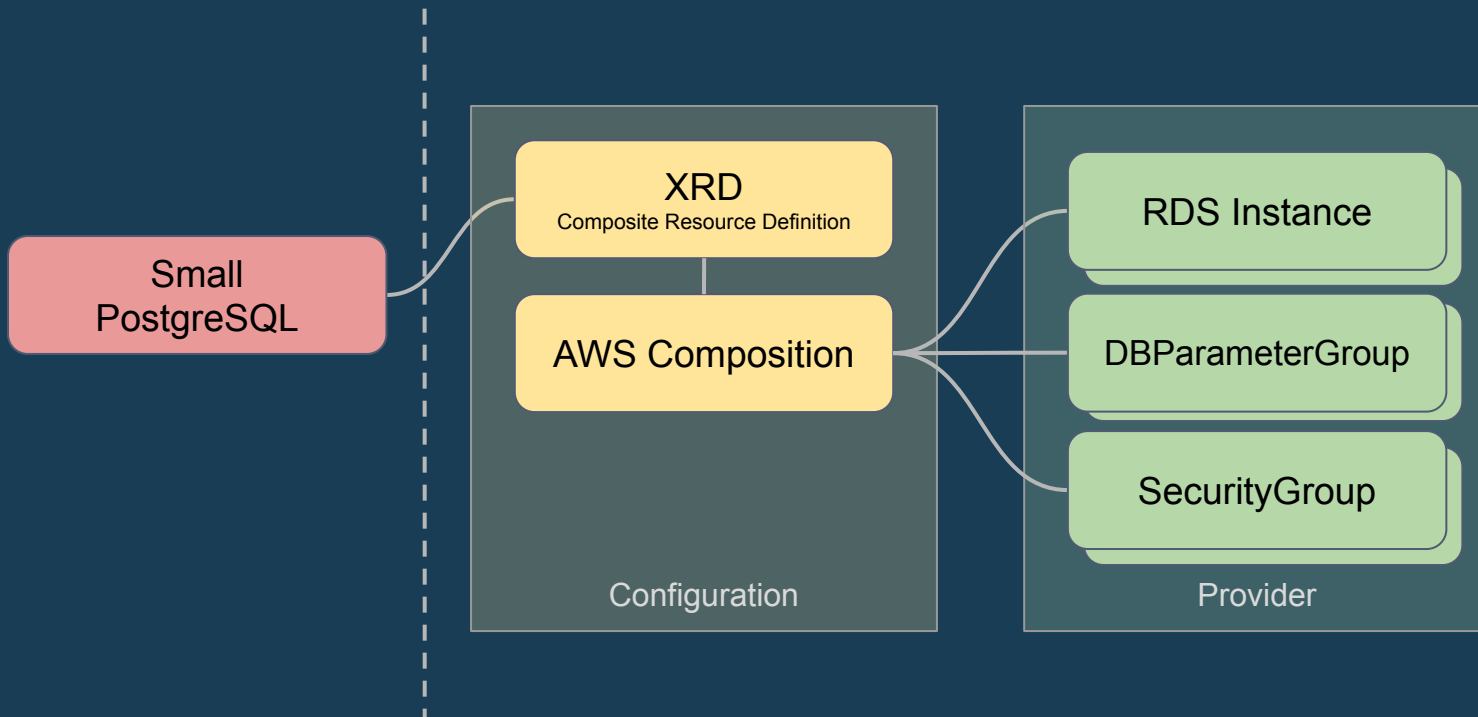
Composition



Build your own Platform API

- Assemble granular resources. E.g. from multiple clouds.
- Expose as higher level self-service API for your app teams
 - **Compose** GKE, NodePool, Network, Subnetwork
 - **Offer** as a single Cluster resource (API) with limited config for developers to self-service
- Hide infrastructure complexity and include policy guardrails
- All with K8s API - compatible with kubectl, GitOps, etc.
- **No code** required, it's all **declarative**





Composite Resources

First we create Composite Resource Definition (XRD) to declare our custom platform API

```
apiVersion: apiextensions.crossplane.io/v1
kind: CompositeResourceDefinition
metadata:
  name: xpostgresinstances.database.example.org
spec:
  group: database.example.org
  names:
    kind: XPostgreSQLInstance
    plural: xpostgresinstances
  versions:
    - name: v1alpha1
      served: true
      referenceable: true
      schema:
        openAPIV3Schema:
          type: object
          properties:
```

Custom API Group

Standard openAPIV3 Schema

Compositions

```
apiVersion: apiextensions.crossplane.io/v1
kind: Composition
metadata:
  name: xpostgresinstances.aws.database.example.org
spec:
  writeConnectionSecretsToNamespace: crossplane-system
  compositeTypeRef:
    apiVersion: database.example.org/v1alpha1
    kind: XPostgreSQLInstance
  resources:
    - name: parametergroup
      base:
        apiVersion: rds.aws.crossplane.io/v1alpha1
        kind: DBParameterGroup
```

Then we define
Composition which
implements XRD

XRD reference

List of Managed Resources
to Compose



Patches

patches:

- fromFieldPath: "spec.nodes.count"
toFieldPath: "spec.forProvider.scalingConfig.desiredSize"
- fromFieldPath: "spec.nodes.size"
toFieldPath: "spec.forProvider.instanceTypes[0]"

transforms:

- type: map

map:

small: t3.small
medium: t3.medium
large: t3.large

Patches enable propagation of data from Composite Resource (XR) down to composed Managed Resources (MR)

Copy of value from XR spec down to MR spec

Map transform to manipulate the config data

Extending Crossplane

Providers & Configurations



Current Extension Points

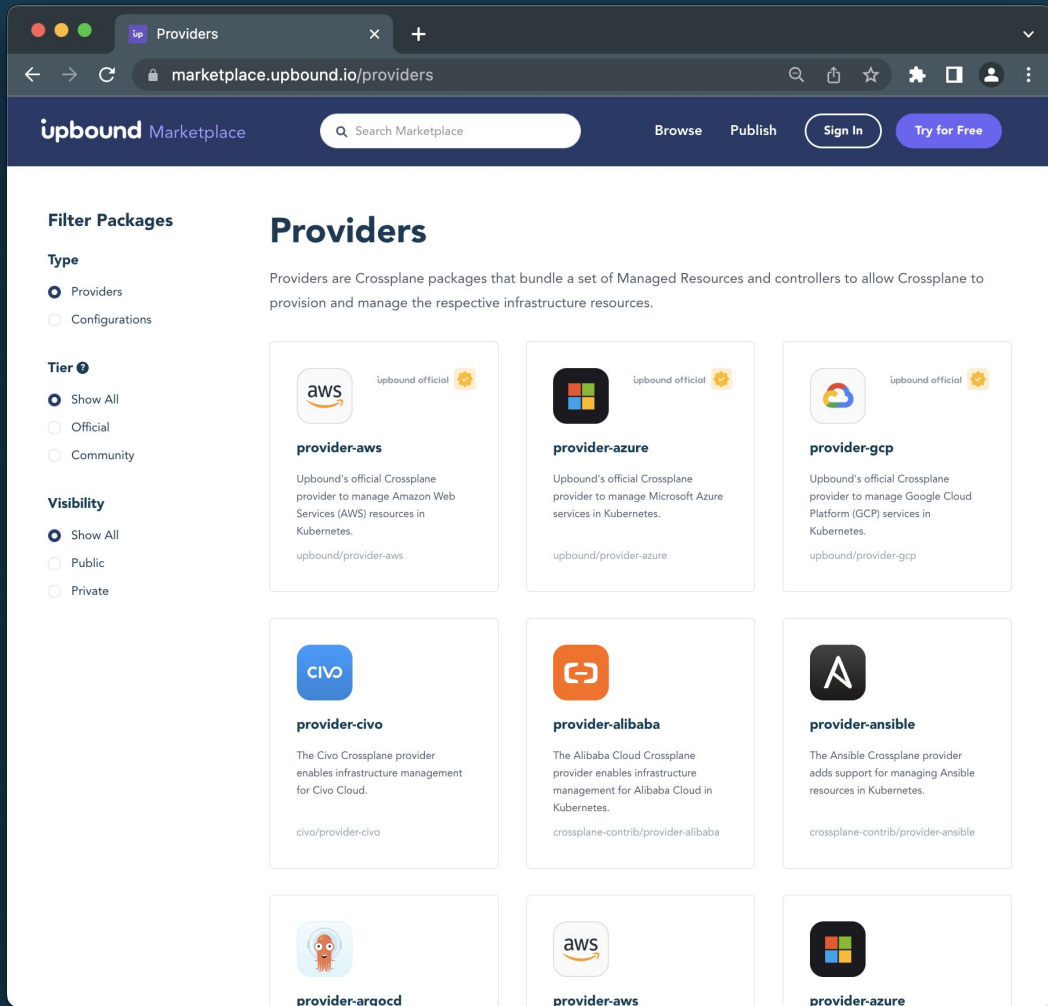
- Crossplane is a highly extensible framework
- **Providers**
 - You can build a provider to manage **anything** with an API
 - CRUD operations for cloud resources, on-prem services, etc.
- **Configurations**
 - Compose resources from providers
 - Define your control plane's declarative APIs and abstractions
 - These are what your devs see - it's how they consume the offerings of your control plane
- Both are Crossplane packages / opinionated OCI Images.

Crossplane Provider Ecosystem



Marketplace for all Extensions

Open for everyone



The screenshot shows the Upbound Marketplace interface for Providers. The browser address bar displays `marketplace.upbound.io/providers`. The page header includes the Upbound logo, a search bar, and links for Browse, Publish, Sign In, and Try for Free.

Filter Packages

- Type**
 - ☒ Providers
 - ☐ Configurations
- Tier**
 - ☒ Show All
 - ☐ Official
 - ☐ Community
- Visibility**
 - ☒ Show All
 - ☐ Public
 - ☐ Private

Providers

Providers are Crossplane packages that bundle a set of Managed Resources and controllers to allow Crossplane to provision and manage the respective infrastructure resources.

The page displays a grid of provider cards, each with a logo, name, description, and repository path:

- provider-aws** (Upbound official): Upbound's official Crossplane provider to manage Amazon Web Services (AWS) resources in Kubernetes. `upbound/provider-aws`
- provider-azure** (Upbound official): Upbound's official Crossplane provider to manage Microsoft Azure services in Kubernetes. `upbound/provider-azure`
- provider-gcp** (Upbound official): Upbound's official Crossplane provider to manage Google Cloud Platform (GCP) services in Kubernetes. `upbound/provider-gcp`
- provider-civo**: The Civo Crossplane provider enables infrastructure management for Civo Cloud. `civo/provider-civo`
- provider-alibaba**: The Alibaba Cloud Crossplane provider enables infrastructure management for Alibaba Cloud in Kubernetes. `crossplane-contrib/provider-alibaba`
- provider-ansible**: The Ansible Crossplane provider adds support for managing Ansible resources in Kubernetes. `crossplane-contrib/provider-ansible`
- provider-argocd**: (Logo visible)
- provider-aws**: (Logo visible)
- provider-azure**: (Logo visible)

Composition Functions

```
apiVersion: database.example.org/v1alpha1
kind: AcmeCoDatabase
metadata:
  name: example-db
spec:
  parameters:
    storageGB: 20
  compositionRef:
    name: example
```

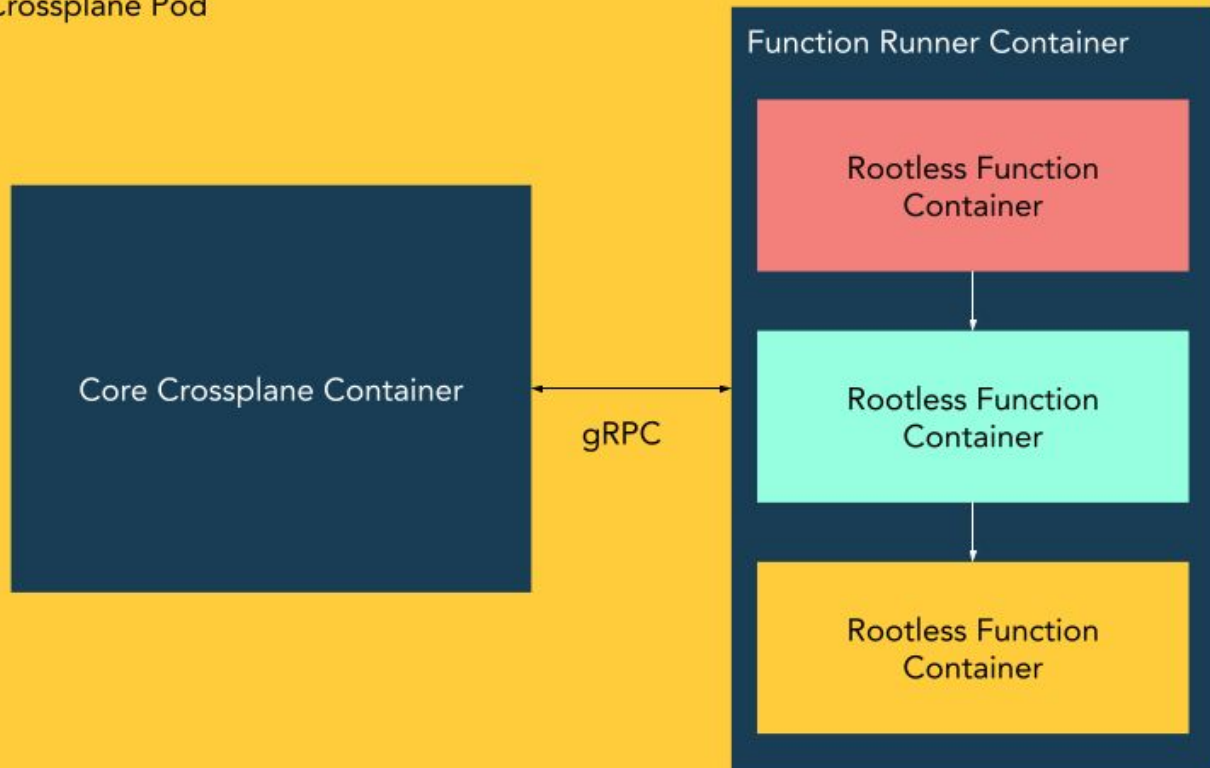
```
apiVersion: apiextensions.crossplane.io/v1
kind: Composition
metadata:
  name: example
spec:
  compositeTypeRef:
    apiVersion: database.example.org/v1alpha1
    kind: AcmeCoDatabase
  resources:
    - name: cloudsqlinstance
      base:
        apiVersion: database.gcp.crossplane.io/v1beta1
        kind: CloudSQLInstance
        spec:
          forProvider:
            databaseVersion: POSTGRES_9_6
            region: us-central1
            settings:
              tier: db-custom-1-3840
              dataDiskType: PD_SSD
      patches:
        - type: FromCompositeFieldPath
          fromFieldPath: spec.parameters.storageGB
          toFieldPath: spec.forProvider.settings.dataDiskSizeMb
          transforms:
            - type: math
              math:
                multiply: 1024
```



```
apiVersion: apiextensions.crossplane.io/v2alpha1
kind: Composition
metadata:
  name: example
spec:
  compositeTypeRef:
    apiVersion: database.example.org/v1alpha1
    kind: XPostgreSQLInstance
  functions:
  - name: my-cool-function
    type: Container
    container:
      image: xkpg.io/my-cool-function:0.1.0
```

```
apiVersion: apiextensions.crossplane.io/v1alpha1
kind: FunctionIO
config:
  apiVersion: database.example.org/v1alpha1
  kind: Config
  metadata:
    name: cloudsql
  spec:
    version: POSTGRES_9_6
observed:
  composite:
    resource:
      apiVersion: database.example.org/v1alpha1
      kind: XPostgreSQLInstance
      metadata:
        name: my-db
      spec:
        parameters:
          storageGB: 20
        compositionSelector:
          matchLabels:
            provider: gcp
      status:
        conditions:
          - type: Ready
            status: True
      connectionDetails:
        - name: uri
          value: postgresql://db.example.org:5432
```

Crossplane Pod



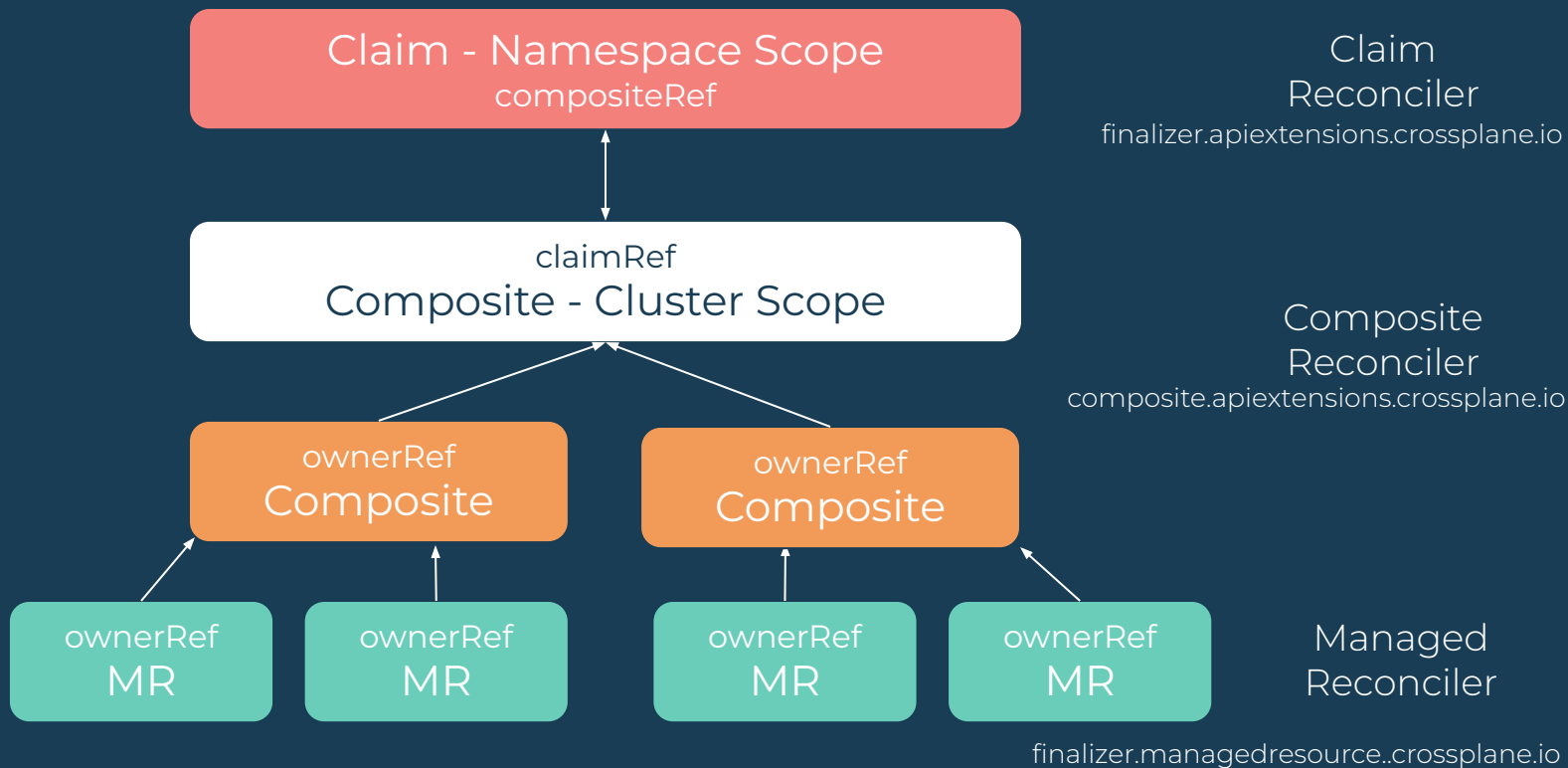
Crossplane Garbage Collection



Claim/Composite Delete Challenges

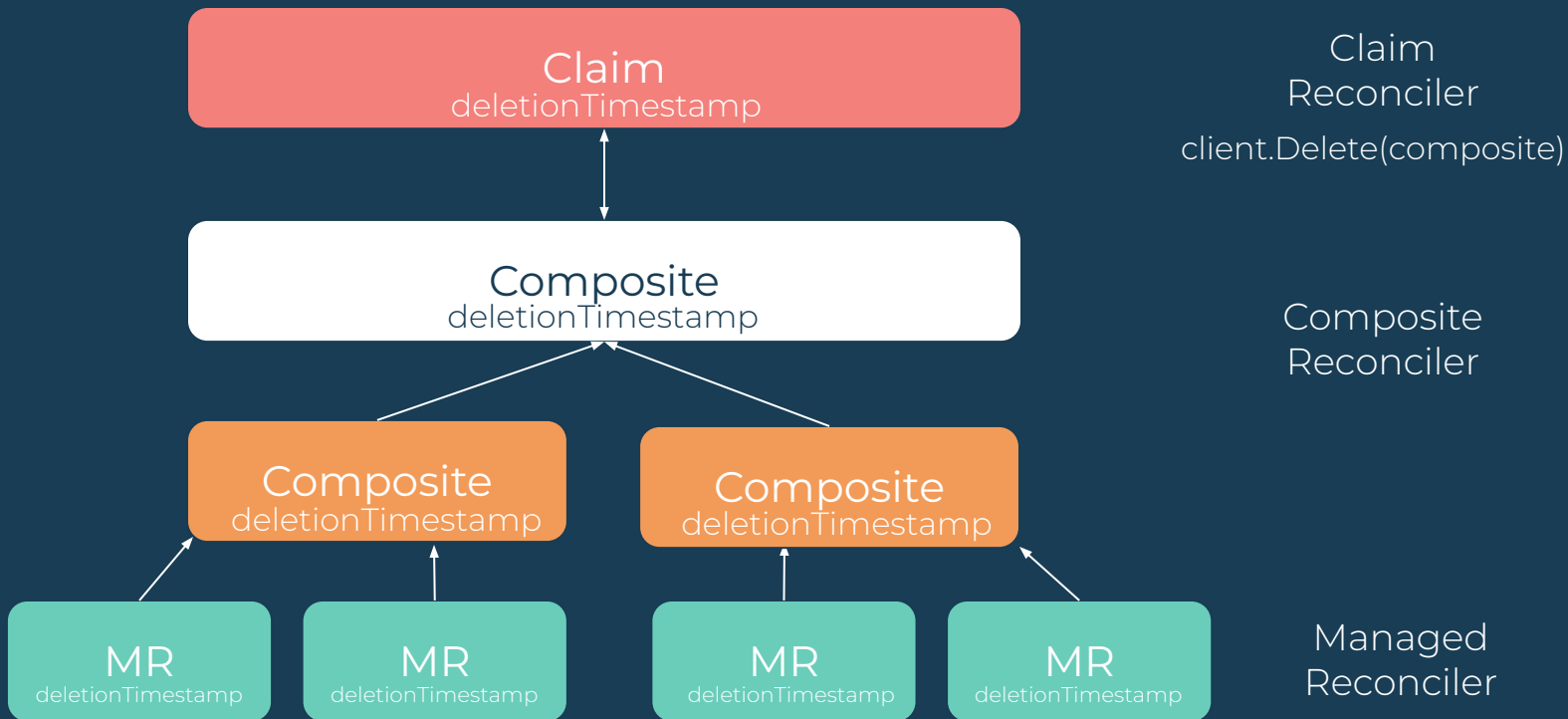
- Managed resources that fail to delete cleanly are invisible to the user of `kubectl delete`
- Order of deletion is undefined
- Foreground deletion on Composites didn't work because `blockOwnerDeletion` was not set on `ownerReferences`
- Claim is namespaced and Composite is cluster scoped
 - No `ownerReference` from Composite to Claim is possible
 - Foreground cascading deletion cannot be executed on Claim

Resource Relationships



Claim Delete - Background (default)

kubectl delete -n <namespace> claimName



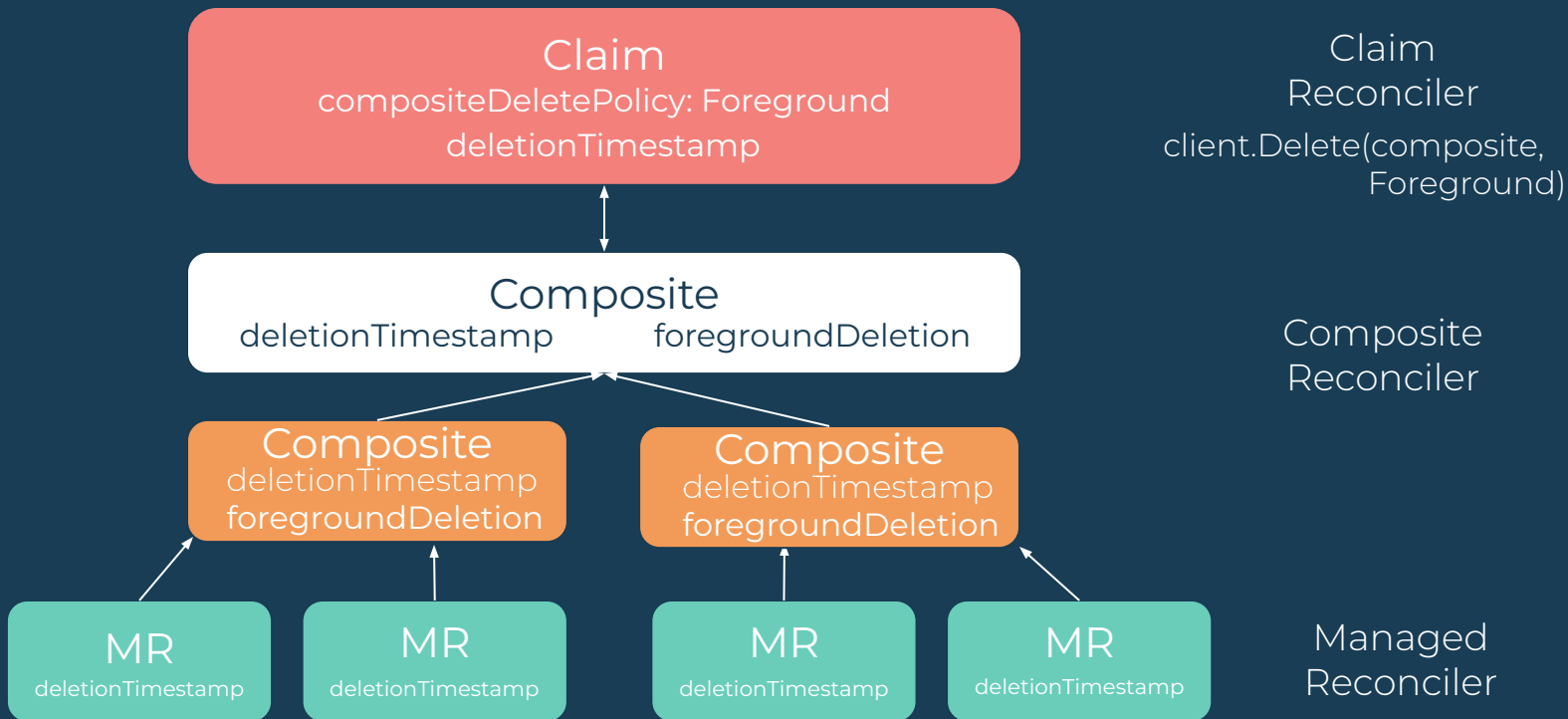


Foreground Deletion Support

- 1.10 adds compositeDeletePolicy attribute to Claim
 - Claim reconciler sets the PropagationPolicy in the Delete() API call for the Composite
 - Default value is Background
 - “Foreground” will simulate cascading Foreground deletion
- Added blockOwnerDeletion: true on all ownerReferences
 - Delete standalone Composites with *–cascade=foreground*
- Specifying the propagation policy via *–cascade* for kubectl delete on a Claim is still ignored

Claim Delete - Foreground

kubectl delete -n <namespace> claimName



Next Steps...

Community is everything



Get Involved

- Website: <https://crossplane.io/>
- Docs: <https://crossplane.io/docs>
- GitHub: <https://github.com/crossplane/crossplane>
- Slack: <https://slack.crossplane.io/>
- Blog: <https://blog.crossplane.io/>
- Twitter: https://twitter.com/crossplane_io
- Youtube: [Crossplane Youtube](#)



Questions?

While we have your attention...we're doing a Crossplane community and adoption survey - help our proposal for Graduation with the CNCF!

<https://tinyurl.com/3b5cyfap>

