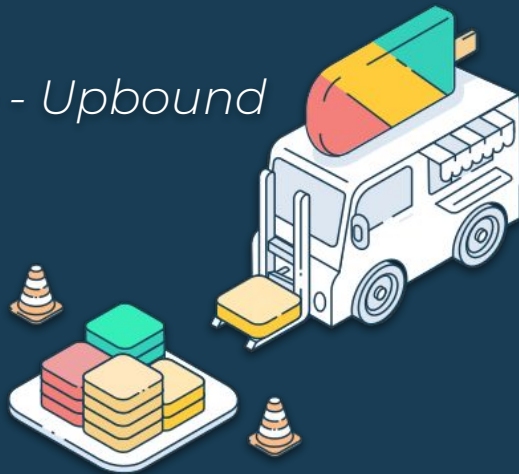


Crossplane

<https://crossplane.io>

Jared Watts - Upbound

Mark Anderson-Trocmé - Upbound





What is Crossplane?

- Your cloud native **control plane**
 - Provision/manage **all** of your resources
- **Compose** those resources into high level **abstractions**
 - Give your developers self-service provisioning
- Kubernetes is a great control plane for containers
 - Crossplane teaches it how to manage **everything** else
- Cloud providers have used control planes for years
 - Now it's your turn to build your own!



CNCF project ready for Graduation!

- [2,500+](#) contributors to the project
- [Top 10%](#) of all CNCF projects for PR authors
- [850+](#) PR authors
- [400+](#) companies contributing
- [150+](#) companies with maintainers
- Steering committee has members from **Apple**, **Nokia**, and **Upbound** to lead and steward the project
- Lots of adopters in **production** and at **scale** ([ADOPTERS.md](#))
 - Nike, Autodesk, Grafana, NASA Science Cloud, Elastic, Akamai, SAP, IBM, VMWare Tanzu, Nokia, etc.
- Full proposal: [cncf/toc#1397](#)

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 interface. On the left, there's a sidebar for the 'provider-family-aws' package, which includes an 'Install Package Manifest' button, a 'Trust Tier' section marked as 'Official', a 'Source' link to a GitHub repository, and a 'Languages' section listing 'python' and 'kcl'. The main content area shows the 'Overview' for the 'provider-family-aws' package, version v1.17.0. It describes it as 'Upbound's official Crossplane provider to manage Amazon Web Services (AWS) config services in Kubernetes.' Below this is a tabbed interface with 'Providers' selected, showing a list of providers. The list has columns for 'Provider', 'Version', and 'Repository'. The providers listed are: provider-aws-accessanalyzer, provider-aws-account, provider-aws-acm, provider-aws-acmpca, provider-aws-amp, provider-aws-amplify, provider-aws-apigateway, and provider-aws-apigatewayv2. All versions are marked as 'latest'.

Provider	Version	Repository
provider-aws-accessanalyzer	latest	upbound/provider-aws-accessanalyzer
provider-aws-account	latest	upbound/provider-aws-account
provider-aws-acm	latest	upbound/provider-aws-acm
provider-aws-acmpca	latest	upbound/provider-aws-acmpca
provider-aws-amp	latest	upbound/provider-aws-amp
provider-aws-amplify	latest	upbound/provider-aws-amplify
provider-aws-apigateway	latest	upbound/provider-aws-apigateway
provider-aws-apigatewayv2	latest	upbound/provider-aws-apigatewayv2


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	April 21, 2023, 15:00:23 (UTC+01:00)

Tags (1)

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

Key	Value
Name	CrossplaneDeepDiveDemoBucket

Managed Resources

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

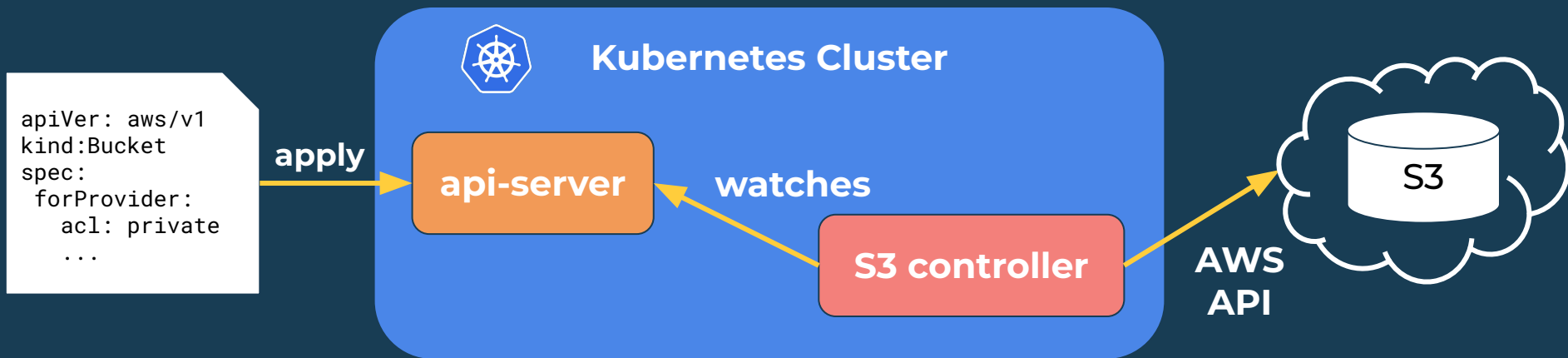
```
Status:
  At Provider:
    Arn:  arn:aws:s3:::crossplane-deepdive-demo-bucket
```

```
Events:
  Type      Age      From                                Message
  ----      -
  Normal    6m8s    bucket.s3.aws.crossplane.io    Successfully created external resource
```

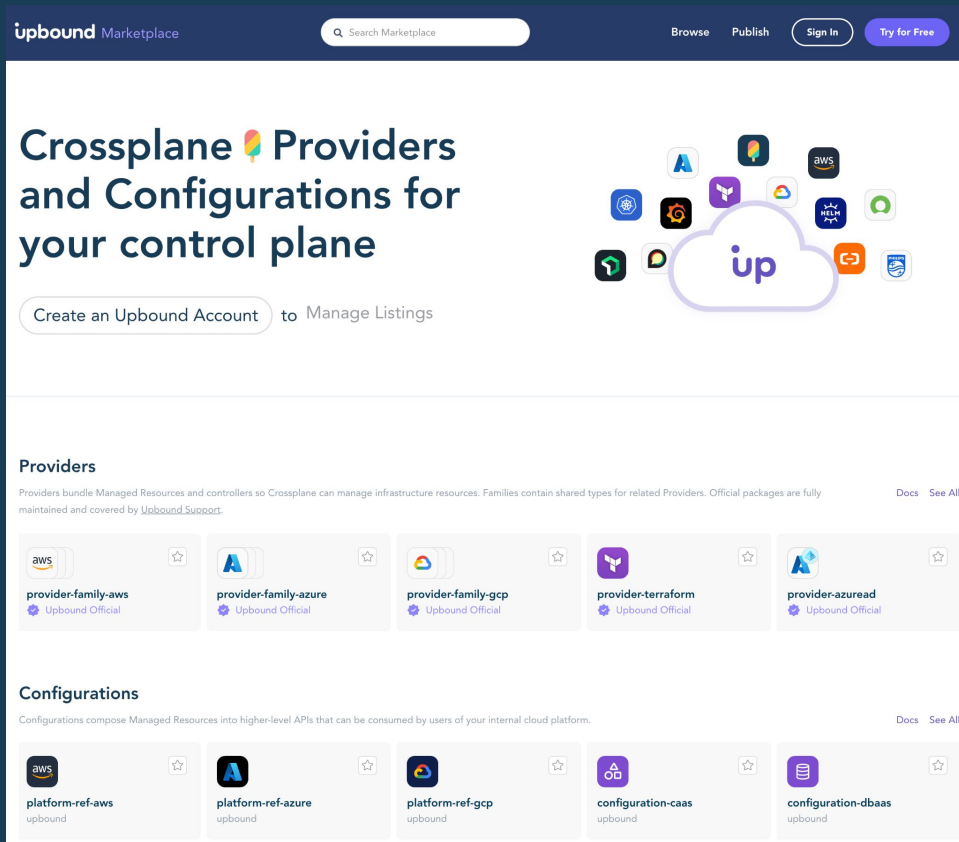
Managed Resources
Generate K8s Events

Managed Resource Reconciliation

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



Lots of Extensions for Crossplane



The screenshot shows the Upbound Marketplace interface. At the top, there's a navigation bar with 'upbound Marketplace', a search bar, and links for 'Browse', 'Publish', 'Sign In', and 'Try for Free'. The main heading is 'Crossplane Providers and Configurations for your control plane'. Below this is a button 'Create an Upbound Account to Manage Listings'. A central graphic shows the 'up' logo in a cloud, surrounded by various cloud provider icons like AWS, Azure, GCP, and others. The page is divided into two sections: 'Providers' and 'Configurations'. Each section has a descriptive paragraph and a list of items with icons, names, and 'Upbound Official' badges.

Crossplane Providers and Configurations for your control plane

Create an Upbound Account to Manage Listings

Providers

Providers bundle Managed Resources and controllers so Crossplane can manage infrastructure resources. Families contain shared types for related Providers. Official packages are fully maintained and covered by Upbound Support.

- provider-family-aws (Upbound Official)
- provider-family-azure (Upbound Official)
- provider-family-gcp (Upbound Official)
- provider-terraform (Upbound Official)
- provider-azuread (Upbound Official)

Configurations

Configurations compose Managed Resources into higher-level APIs that can be consumed by users of your internal cloud platform.

- platform-ref-aws (upbound)
- platform-ref-azure (upbound)
- platform-ref-gcp (upbound)
- configuration-caas (upbound)
- configuration-dbaas (upbound)

Providers, Functions,
Configurations

Discover and share extensions
with the community

<https://marketplace.upbound.io>

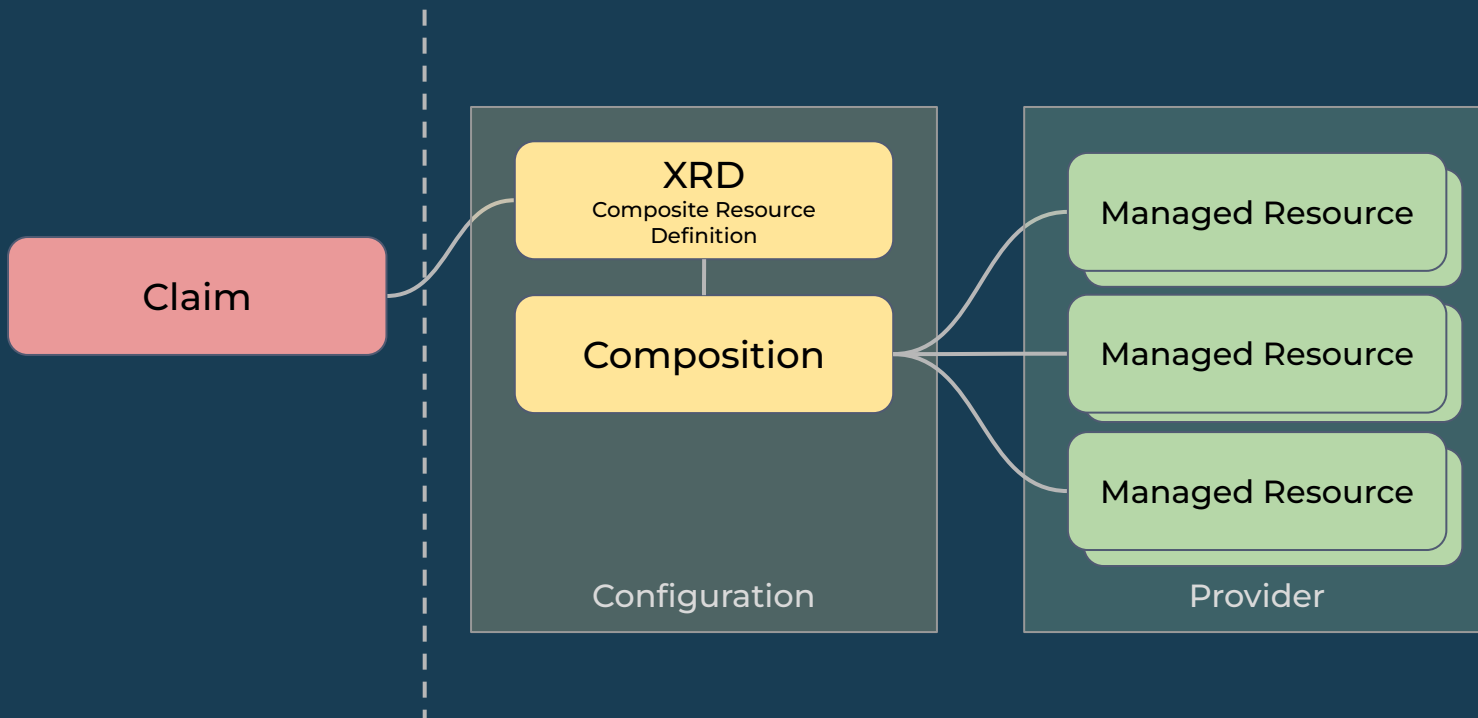
Building Your Control Plane

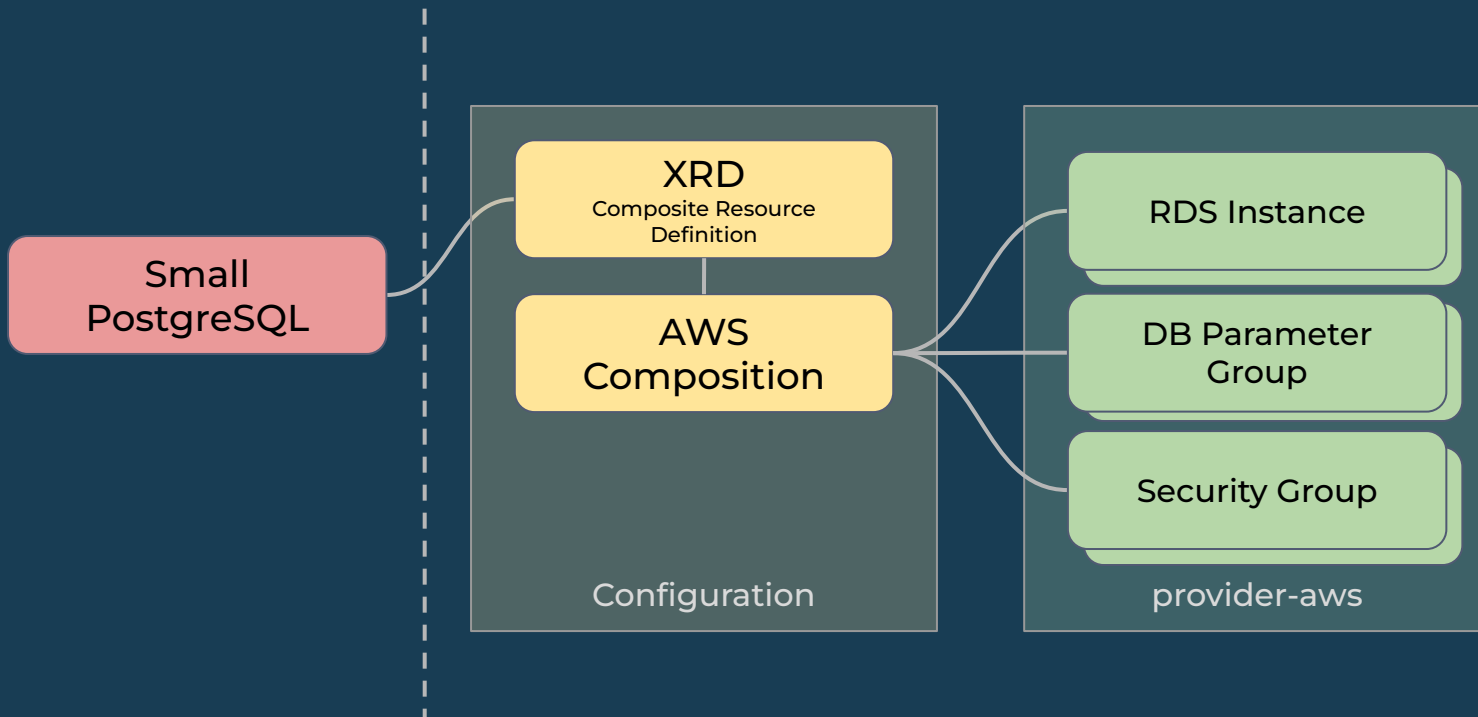
Composition and Functions



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** abstraction (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**





Composite Resources

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

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

Custom API Group

Standard openAPIV3 Schema

Compositions

```
apiVersion: apiextensions.crossplane.io/v1
kind: Composition
metadata:
  name: nosqls.database.example.com
spec:
  compositeTypeRef:
    apiVersion: database.example.com/v1alpha1
    kind: NoSQL
  mode: Pipeline
  pipeline:
    - step: generate-resources
      functionRef:
        name: function-acme-func
        input: {}
    - step: filter-resources
      functionRef:
        name: function-filter
        input: {}
```

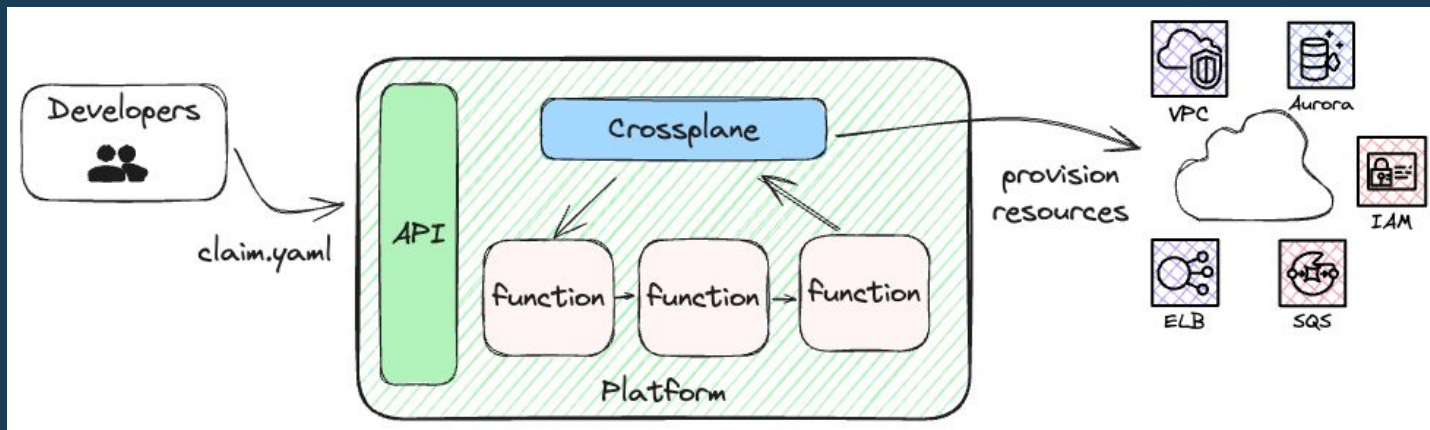
Then we define a Composition which implements the XRD

The XRD this Composition is for

Pipeline of functions to execute that will generate managed resources

How do Functions work?

- Run a pipeline of simple functions to compose resources
- Written in your language of choice with your unique logic
- Sweet spot between “no code” ↔ building an entire controller
 - Focus only on your platform’s unique needs
 - Crossplane does the heavy lifting of resources CRUD, reconciling, finalizers, owner refs, etc.





Composition Functions have graduated

- Functions are now V1 in [v1.17+](#)
 - Mature, stable, production ready
 - Crossplane is all in on Functions
 - If ever you felt like your hands were tied when writing compositions with Patch & Transform, no more - anything is possible



How can I start using Functions?

- You don't **have** to write any code to start using functions
 - Community has built an ecosystem of “reusable functions”
 - Ready to use directly from your compositions
- Spectrum of no-code → low-code → full-code
 - Higher level config languages
 - General purpose programming languages
 - Choose what you're comfortable with and mix-n-match!
- Let's see some examples...

Templating

```
pipeline:
- step: render-templates
  functionRef:
    name: function-go-templating
  input:
    apiVersion: gotemplating.fn.crossplane.io/v1beta1
    kind: GoTemplate
    inline:
      template: |
        {{- range $i := until ( .observed.composite.resource.spec.count | int ) }}
        apiVersion: iam.aws.upbound.io/v1beta1
        kind: AccessKey
        spec:
          forProvider:
            userSelector:
              matchLabels:
                crossplane.io/name: user-{{ $i }}
        {{- end }}
```

```
pipeline:
- step: render-instances
  functionRef:
    name: kcl-function
  input:
    apiVersion: krm.kcl.dev/v1alpha1
    kind: KCLInput
    spec:
      source: |
        regions = ["us-east-1", "us-east-2"]
        items = [{
          apiVersion: "ec2.aws.upbound.io/v1beta1"
          kind: "Instance"
          metadata.name = "instance-" + r
          spec.forProvider: {
            ami: "ami-0d9858aa3c6322f73"
            instanceType: "t2.micro"
            region: r
          }
        } for r in regions]
```

```
// if a base ARN exists, render a policy with all ARNs.
if baseARN != "unknown" {
    let allTuples = list.Concat([
        [baseARN, baseARN + "/*"],
        [
            for a in additionalARNs {[a, a + "/*"]},
        ],
    ])
    let allResources = list.FlattenN( allTuples, 1)
    response: desired: resources: iam_policy: resource: {
        apiVersion: "iam.aws.upbound.io/v1beta1"
        kind: "Policy"
        metadata: {
            name: "${compName}-access-policy"
        }
        spec: {
            forProvider: {
                path: "/"
                policy: json.Marshal({
                    Version: "2012-10-17"
                    Statement: [
                        {
                            Sid: "S3BucketAccess"
                            Action: [
                                "s3:GetObject",
                                "s3:PutObject",
                            ]
                            Effect: "Allow"
                            Resource: allResources
                        }
                    ],
                })
            }
        }
    }
}
```

```
desired {
  resources {
    ["cm-obj"] = new {
      resource = new Object {
        spec {
          forProvider {
            manifest = new ConfigMap {
              metadata {
                namespace = "crossplane-system"
              }
              data {
                ["env"] = "prod"
                ["team"] = "core"
              }
            }
          }
        }
      }
    }
  }
}
```

Full Code - General Purpose Programming

```
// RunFunction observes an example composite resource (XR). It simple adds one
// S3 bucket to the desired state.
func (f *Function) RunFunction(_ context.Context, req *fnv1beta1.RunFunctionRequest)
(*fnv1beta1.RunFunctionResponse, error) {
    f.log.Info("Running Function", "tag", req.GetMeta().GetTag())
    rsp := response.To(req, response.DefaultTTL)

    // create a single test S3 bucket
    _ = v1beta1.AddToScheme(composed.Scheme)
    name := "test-bucket"
    b := &v1beta1.Bucket{
        ObjectMeta: metav1.ObjectMeta{
            Annotations: map[string]string{
                "crossplane.io/external-name": name,
            },
        },
        Spec: v1beta1.BucketSpec{
            ForProvider: v1beta1.BucketParameters{
                Region: ptr.To[string]("us-east-2"),
            },
        },
    }

    ...

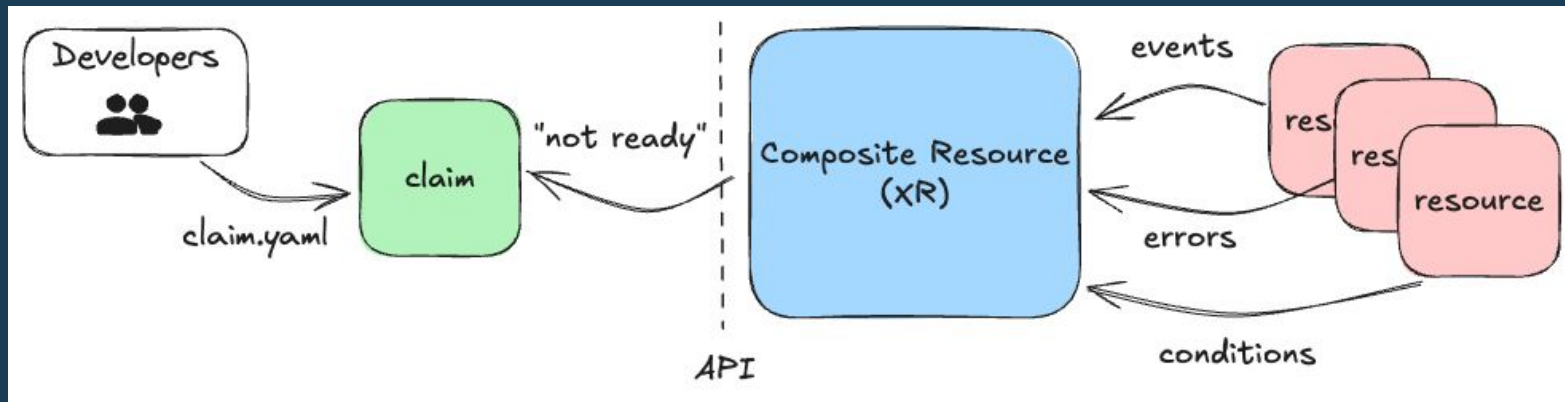
    return rsp, nil
}
```

Claim Communication

Help your devs find root cause

Claim communication

- Long standing user experience problem in Crossplane
- Composition model has rich details - under the covers
 - events, status, conditions, error messages
- Details rarely get surfaced to the claim for your developers
 - “separation of concerns” gone too far?



Demo - Claim Communication

github.com/jbw976/demo-xfn-claim-com

Declarative claim communications

```
pipeline:
# previous pipeline steps omitted
- step: status-handler
  functionRef:
    name: function-status-transformer
  input:
    apiVersion: function-status-transformer.fn.crossplane.io/v1beta1
    kind: StatusTransformation
    statusConditionHooks:
      - matchers:
          - resources:
              - name: "cloudsql-instance"
            conditions:
              - type: Synced
                status: "False"
                reason: ReconcileError
                message: "failed to create the database: (?P<Error>.+)"
          setConditions:
            - target: CompositeAndClaim
              condition:
                type: DatabaseReady
                status: "False"
                reason: FailedToCreate
                message: "Encountered an error creating the database: {{ .Error }}"
```

Thanks

[@dalton-hill-0!](#)

Package Manager

Pull and manage packages with ease



ImageConfig API

- Configure **credentials** to pull images and their dependencies
 - Pull private package dependencies
 - Pull images from multiple private registries
- Image verification: Safer supply chain
 - Alpha in 1.18
 - Verify image signatures



Dependency Version Upgrades

- Crossplane will **upgrade** the dependencies of a package when a new version is installed
- Support package installation using **digests**
 - Crossplane used to only allow installing packages with tags now it supports digests
 - Adds to the secure supply chain by guaranteeing images you are pulling the right image with a SHA

Demo - ImageConfig

Package Ease of use & Safety

github.com/markandersonontrocme/demo-imageconfig

Composition Environments



v1.18 matures EnvironmentConfig to Beta

- But “native support” has been removed from Compositions
 - API wasn't right shape, high complexity, maintenance burden
 - **Lesson:** don't leave features in Alpha for too long
- v1.18+ supported experience with Functions
 - [function-environment-configs](#) as first pipeline step to select/filter/merge **EnvironmentConfigs**
 - Rest of function pipeline just uses the data like normal
- Migration support
 - Automated migration tooling - **crossplane beta convert**
 - Migration guide and docs

EnvironmentConfig Beta example

```
pipeline:
- step: environmentConfigs
  functionRef:
    name: function-environment-configs
  input:
    apiVersion: environmentconfigs.fn.crossplane.io/v1beta1
    kind: Input
    spec:
      environmentConfigs:
        - type: Reference
          ref:
            name: example-config
# the environment is be passed to the next function in the pipeline as part of the context
- step: go-templating
  functionRef:
    name: function-go-templating
  input:
    apiVersion: gotemplating.fn.crossplane.io/v1beta1
    kind: GoTemplate
    source: Inline
    inline:
      template: |
        ---
        apiVersion: example.crossplane.io/v1
        kind: XR
        status:
          fromEnv: {{ index .context "apiextensions.crossplane.io/environment" "data" "key" }}
```

Community is everything



Contributors welcome

- Getting started guide now available!
 - [contributing](#) folder in Crossplane GitHub repo
- Code contributions in core and ecosystem
 - functions are 🔥 right now!
- Docs contributions
 - share your expertise with others
- Good first issues, P0/P1 issues, roadmap
- Mentorship opportunities



LFX Mentorship

- Thank you [@enesonus](#) for an awesome '24 summer term!
- Focused on **validate** features and improvements
 - unknown field checks, catch more errors
 - download/cache full dependency graph
 - 150x speed-up!
- Blog post to learn more about his experience:
 - <https://blog.crossplane.io/lfx-mentorship-2024/>
- We'll do more mentorship in the future...





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](#)



Calling all Crossplane Adopters!

🚓 We'd love to hear about your adoption of Crossplane, please share your story in [ADOPTERS.md](#) in the crossplane/crossplane repo 🚓

