Crossplane

The Cloud-Native Framework for Platform Engineering

Jared Watts, Nic Cope
Steering Committee / Maintainers

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!

The Basics

Managed Resources

Managed Resources Example: AWS

Networking

Databases

Kubernetes Clusters

IAM

VMs

Message Queues

Caches

Certificates

...and much more...

upbound Marketplace

Search the Upbound Marketplace

Browse Packages | Publish | Learn More | Sign In | Sign Up Free

aws

provider-family-aws

Official

By Upbound

Install Package Manifest

Trust Tier

Official

Source

github.com/crossplane-contrib/prov...

Languages

python

kcl

Legal

Read the legal terms of service

Home > Providers > provider-family-aws

Overview

Version: v1.17.0 | Star Package

Upbound's official Crossplane provider to manage Amazon Web Services (AWS) config services in Kubernetes.

Readme | Documentation | Providers | ProviderConfig | Provenance | SBOM

Search...

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

Crossplane

https://github.com/crossplane-contrib/provider-upjet-aws

Managed Resources

```yaml
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.

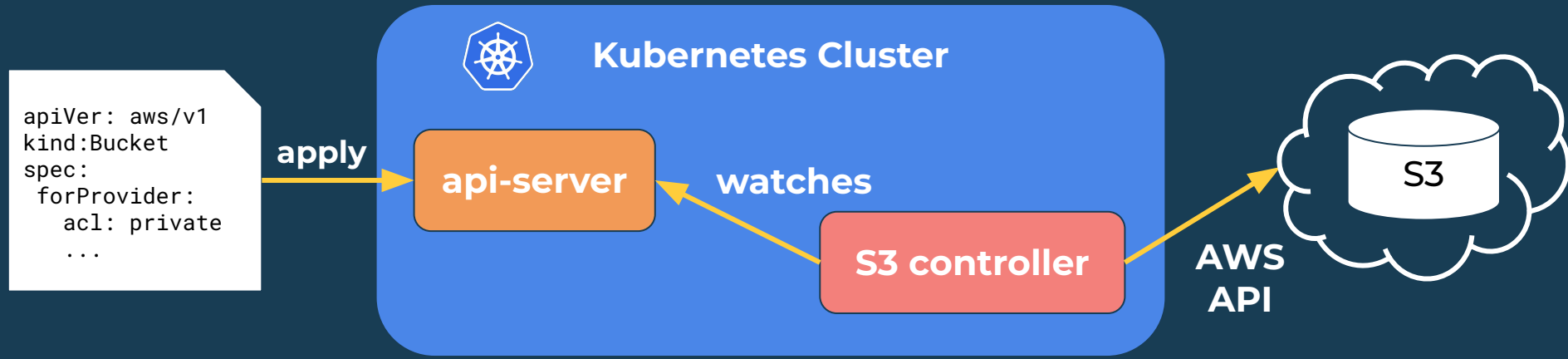
```
Status:
  At Provider:
    Arn:  arn:aws:s3:::crossplane-deepdive-demo-bucket

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)



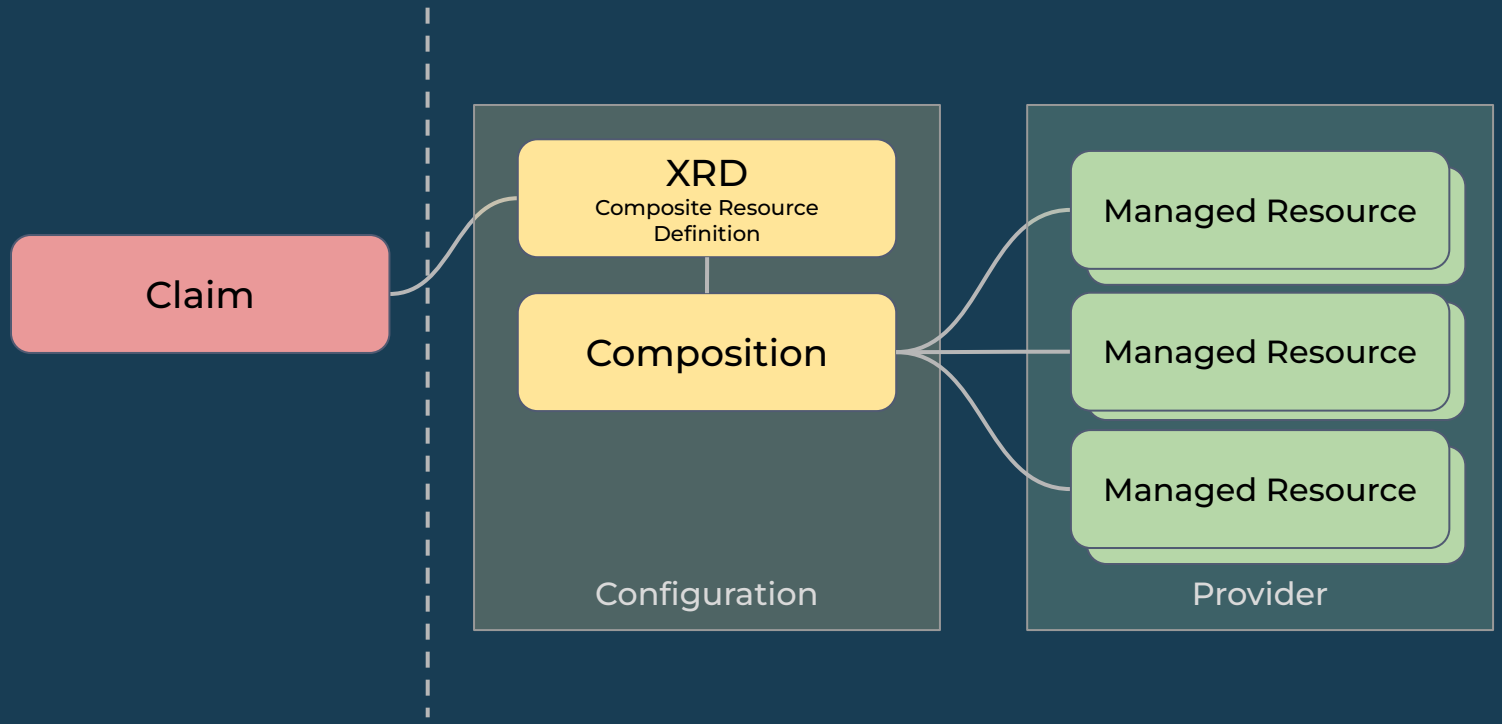
Building Your Control Plane

Composition and Functions

Crossplane

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 simple `Cluster` abstraction (API) with limited config for developers to self-service
- Hide infrastructure complexity and codify a “golden path”
- All with K8s API - compatible with kubectl, GitOps, etc.
- No code **required**


Claim
XRD
Composite Resource Definition
Composition
Configuration
Managed Resource
Managed Resource
Managed Resource
Provider

Crossplane

Small PostgreSQL

```
apiVersion: acme.com/v1
kind: Database
metadata:
  name: cool-db
spec:
  size: small
  engine: postgresql
```

XRD
Composite Resource Definition

GCP Composition

Configuration

Cloud SQL

SQL User

Global Address

provider-aws

Crossplane

Composite Resources

```yaml
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

Crossplane

Compositions

```yaml
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: {}
```

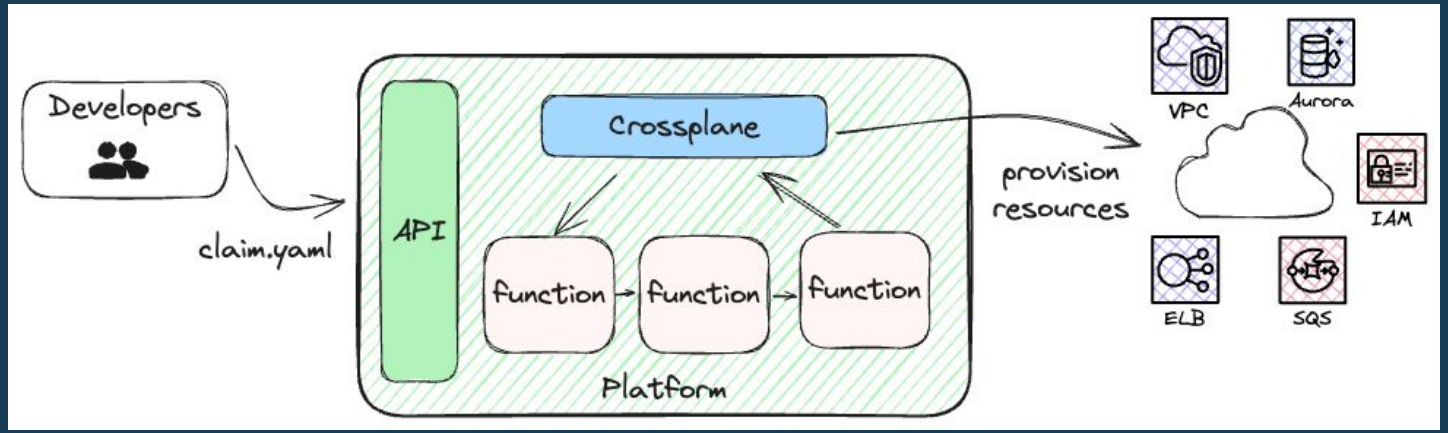
Then we define a Composition which implements the XRD

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
- Focus only on your unique logic
- Crossplane does the heavy lifting of resources CRUD, reconciling, finalizers, owner refs, etc.



How do I use Functions?

- You don't have to write any code to use functions
  - Community has built an ecosystem of "reusable functions"
  - Ready to use directly from your compositions
- Enables entire spectrum of no-code → low-code → full-code
  - Choose what you're comfortable with
  - Higher level config languages
  - Low level general purpose programming languages
  - 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 }}
```

Python

```
pipeline:
- step: compose-a-resource-with-python
  functionRef:
    name: function-python
  input:
    apiVersion: python.fn.crossplane.io/v1beta1
    kind: Script
    script: |
      from crossplane.function.proto.v1 import run_function_pb2 as fnv1

      def compose(req: fnv1.RunFunctionRequest, rsp: fnv1.RunFunctionResponse):
          rsp.desired.resources["bucket"].resource.update({
              "apiVersion": "s3.aws.upbound.io/v1beta2",
              "kind": "Bucket",
              "spec": {
                  "forProvider": {
                      "region": req.observed.composite.resource["spec"]["region"]
                  }
              },
          })
          rsp.desired.resources["bucket"].ready = True
```

KCL

```
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]
```

Crossplane

Thanks @Peefy!

CUE

```
// if a base ARN exusts, 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
                        },
                    ]
                })
            }
        }
    }
}
```

Crossplane

Thanks @gotwarlost!

PKL

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

Crossplane

Thanks @Avarei!

Full Code - General Purpose Programming

```go
// 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
}
```

Crossplane

⚡ v1.19 Lightning Tour ⚡

- Maturing Crossplane features and APIs
  - Usage to Beta
  - Claim Server Side Apply to Beta
  - API promotion policy and contributor guide
- HostNetwork scenarios - port configurability
- Automatic dependency management (downgrades)
- Private repos with Crossplane CLI

v1.20+ Roadmap 🛣️

- Roll out **change logs** feature across provider ecosystem
- Maturing Crossplane features and APIs
  - DeploymentRuntimeConfig
  - Realtime Compositions
- More high level **metrics** investments
- Clear **insight** into reconciliation of resources
- and of course, **Crossplane v2**...

Crossplane v2 preview is here!

- **Crossplane v2 is more useful, more intuitive, and less opinionated**
- Three major changes:
  - Composite resources are now namespaced
  - Managed resources are now namespaced
  - Composition supports any Kubernetes resource
- Crossplane v2 is better suited to building control planes for **applications**, not just infrastructure

Composite resources are namespaced

```yaml
apiVersion: example.crossplane.io/v1
kind: App
metadata:
  namespace: default
  name: my-app
spec:
  image: nginx
  crossplane:
    compositionRef:
      name: app-kcl
```

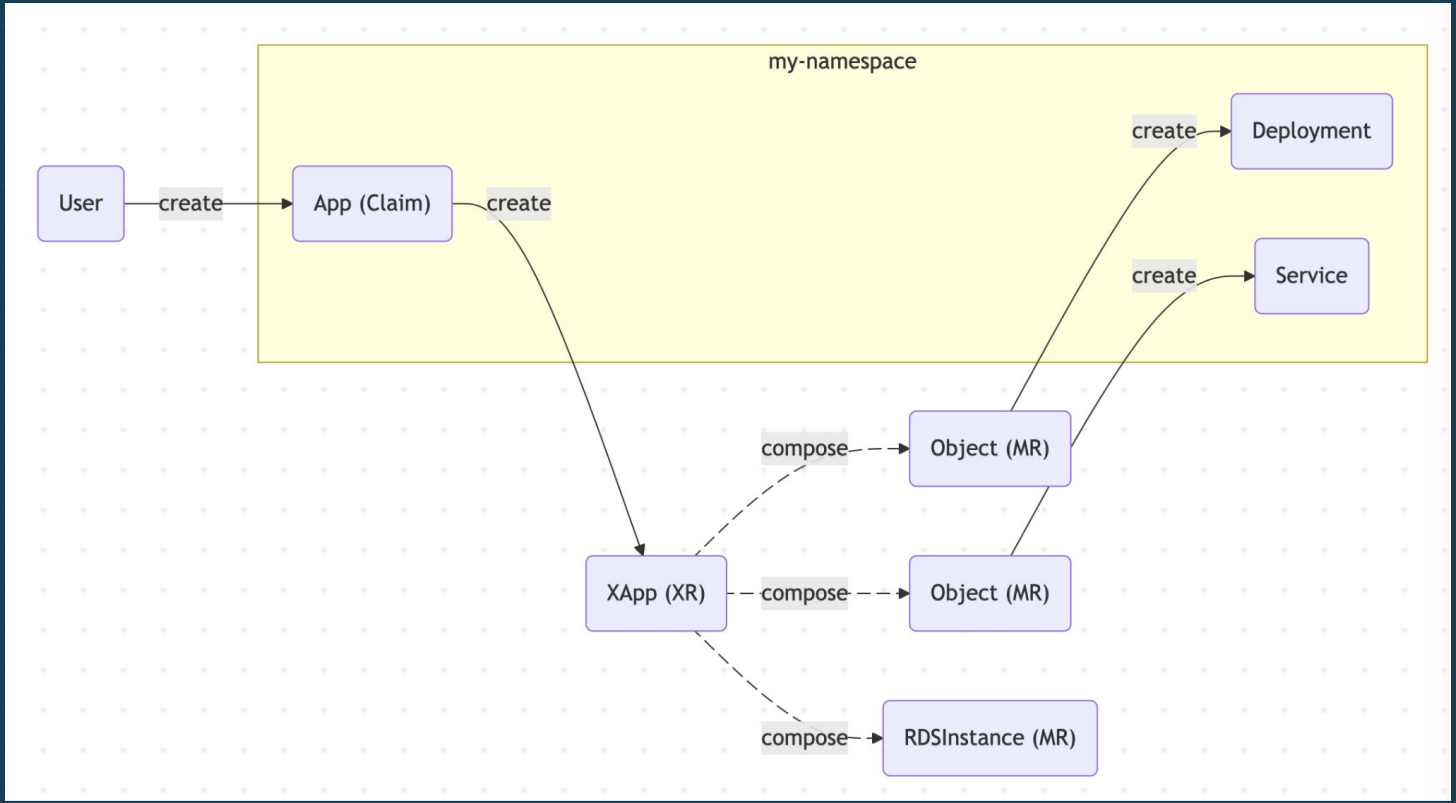
Most composite resources are namespaced

```yaml
apiVersion: apiextensions.crossplane.io/v2alpha1
kind: CompositeResourceDefinition
metadata:
  name: apps.example.crossplane.io
spec:
  scope: Namespaced
  group: example.crossplane.io
  names:
    kind: App
    plural: apps
  versions:
  - name: v1
  # Removed for brevity
```

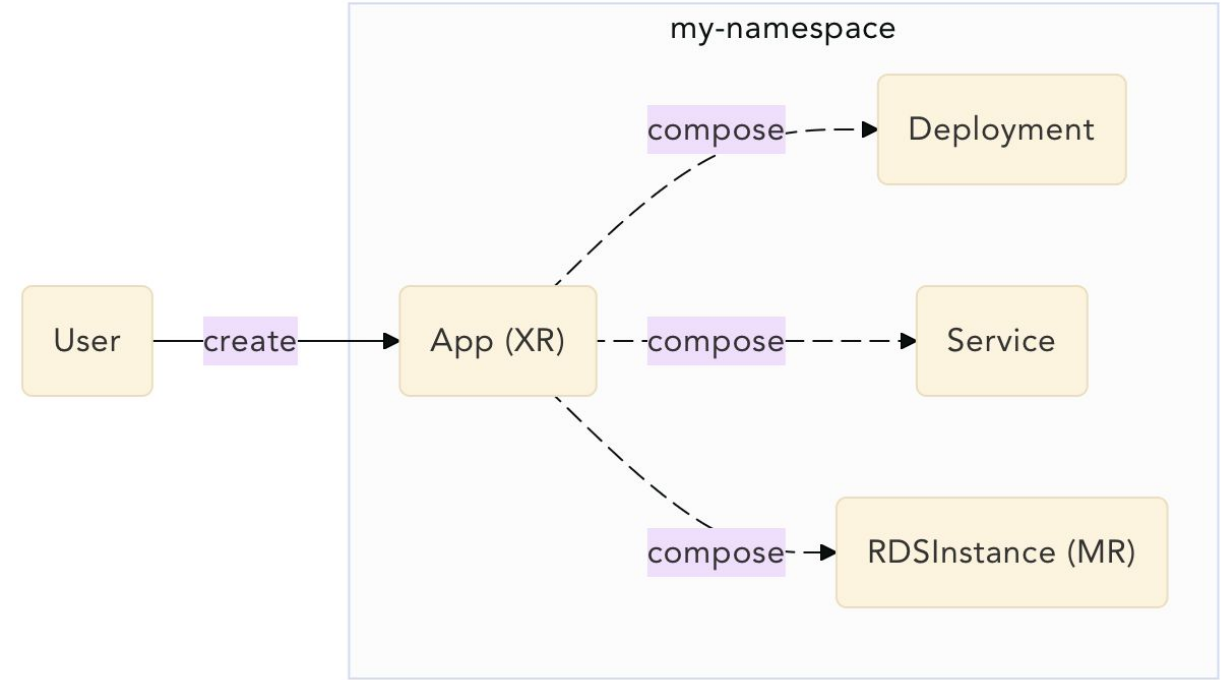
Managed resources are namespaced

```yaml
apiVersion: s3.aws.m.upbound.io/v1beta1
kind: Bucket
metadata:
  namespace: default
  name: my-bucket
spec:
  forProvider:
    region: us-east-2
```

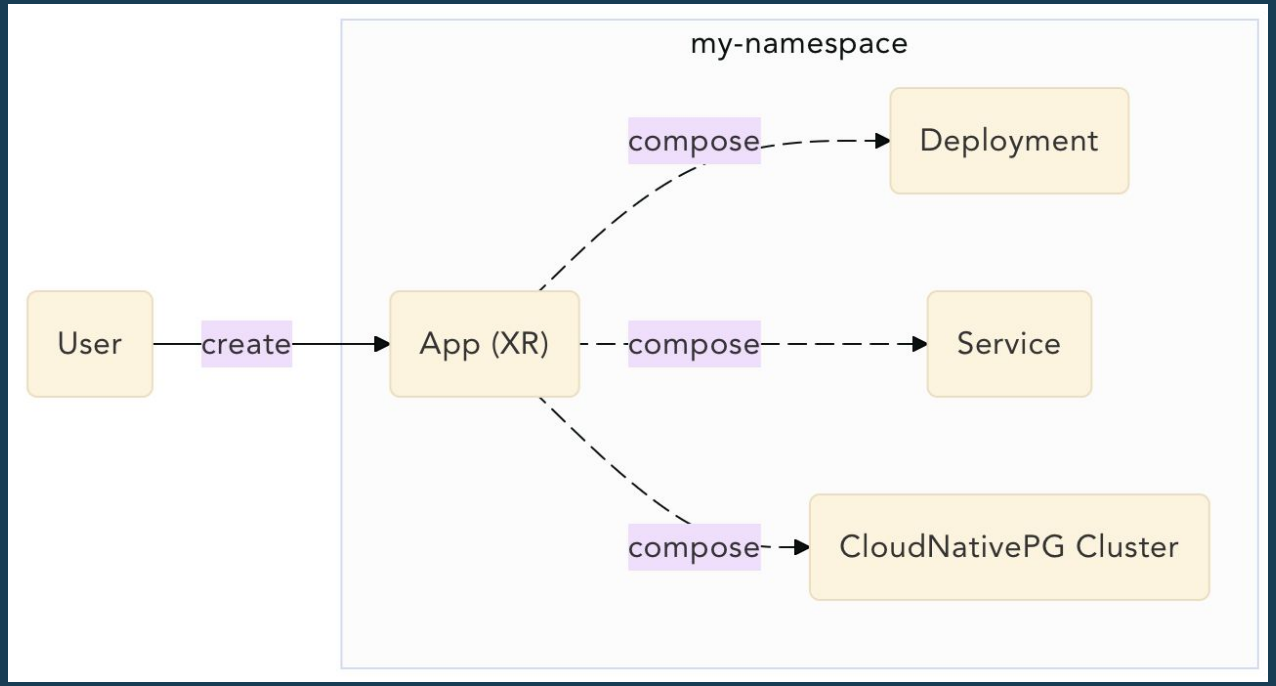
Compose any resource



Compose any resource



Compose any resource



Crossplane v2 is backward compatible with v1

- **Most people can upgrade from v1 to v2 without breaking changes**
- Special `scope: LegacyCluster` XRD creates a v1-style XR and claim
- Providers still support cluster scoped managed resources
- We removed some already-deprecated features in v2

Demo!

Try it today!

- Get started at https://docs.crossplane.io/v2.0-preview/

Community is everything

Contributors welcome

- Getting started guide now available!
  - contributing folder in Crossplane GitHub repo
- Code contributions in core and ecosystem
  - lots of functions and providers to help on!
- Docs contributions
  - share your expertise with others
- Good first issues, P0/P1 issues, roadmap
- Mentorship opportunities

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/
- Bluesky: https://bsky.app/profile/crossplane.io
- Twitter: https://twitter.com/crossplane_io
- Youtube: Crossplane Youtube

Try Crossplane v2 today!

- Get started at https://docs.crossplane.io/v2.0-preview/