Introduction to Crossplane

Stéphane Di Cesare
Senior Engineer, Platform Experience

SRE Meetup Munich, 26.09.2023

DKB
Deutsche Kreditbank AG

Deutsche Kreditbank

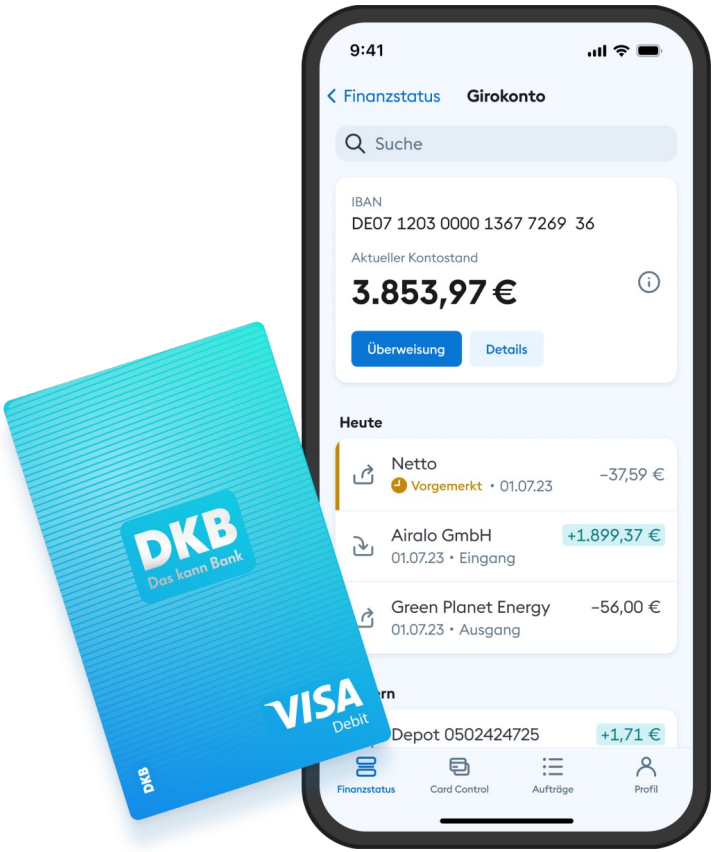
founded 1990

direct bank for private customers

- over 5 million customers
- over 5000 employees

investment bank for specific branches

- largest financer for renewable energy in Germany



We are hiring!

jobs.dkb.de

DKB
Deutsche Kreditbank AG

Why Crossplane at DKB?

our goal: promote **product team responsibility** -> “you build it, you run it”

... but product teams cannot bridge all required technical and bank-specific skills

Our platform is leveraging Crossplane to provide **platform building blocks** managed by the relevant teams

DKB
Deutsche Kreditbank AG

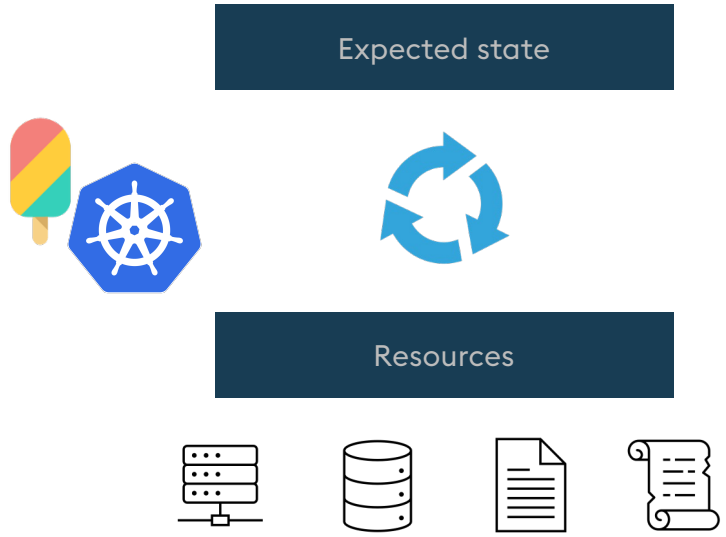
What is Crossplane?

Crossplane allows to run a resource control plane on top of Kubernetes.

Crossplane is an incubating CNCF project, mostly run by upbound.io.

Main features:

- Extension to the Kubernetes API
- Manage ***resources*** as code
- Automatically ***reconciles*** resources
- Provides a higher abstraction layer (***composite resources***), allowing better control of the relations between resources



DKB
Deutsche Kreditbank AG

Crossplane and Kubernetes
Kubernetes API

What is Kubernetes?

Most people know Kubernetes as a ***container orchestration system***

Kubernetes also defines an ***extensible API*** that can be used to define your platform

Crossplane extends the Kubernetes API to manage resources located inside or outside Kubernetes

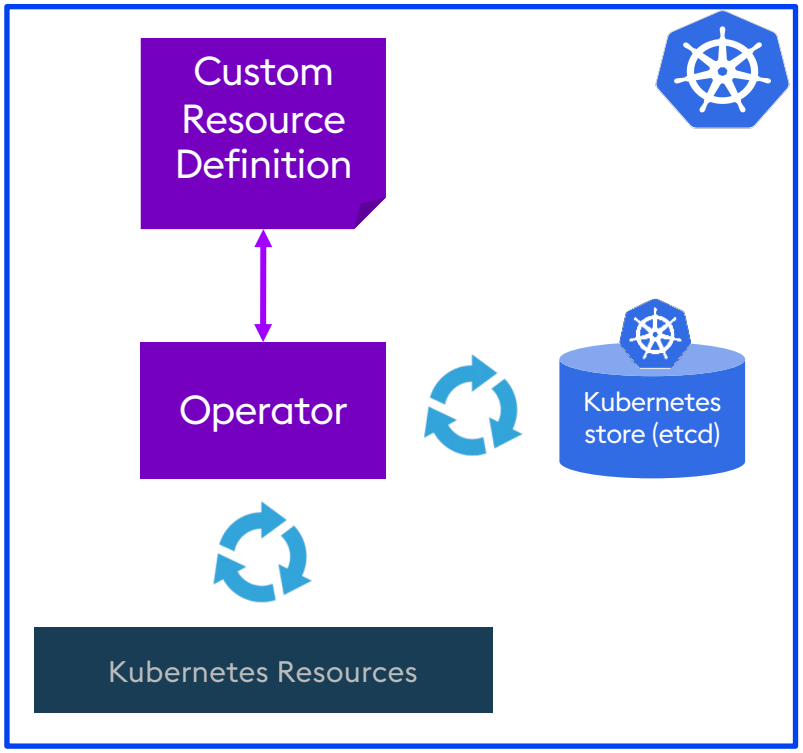
DKB
Deutsche Kreditbank AG

Crossplane and Kubernetes Operator pattern

Core Kubernetes objects are defined by YAML Resource Definitions.

Kubernetes allows to define custom objects through ***Custom Resource Definitions***.

CRDs are managed by ***Operators***, which are ensuring that resources are staying synchronized with their definition.



DKB
Deutsche Kreditbank AG

Crossplane – Managed Resources and Providers

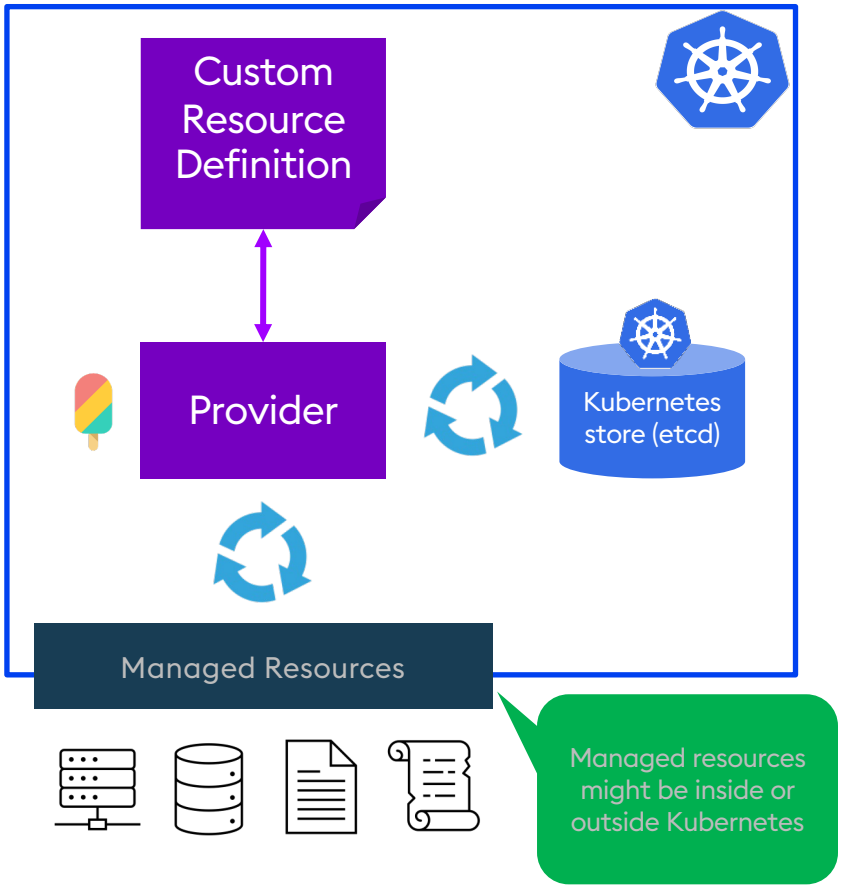
Crossplane allows to work with internal or external resources transparently by introducing the concept of ***Resource Providers***.

Resources under control of Crossplane are referred to as ***Managed Resources***.

Many open-source providers are available for different kinds of resources.

Crossplane providers can be generated from Terraform providers.

Custom providers can be created (in Go) for custom resources.



DKB
Deutsche Kreditbank AG

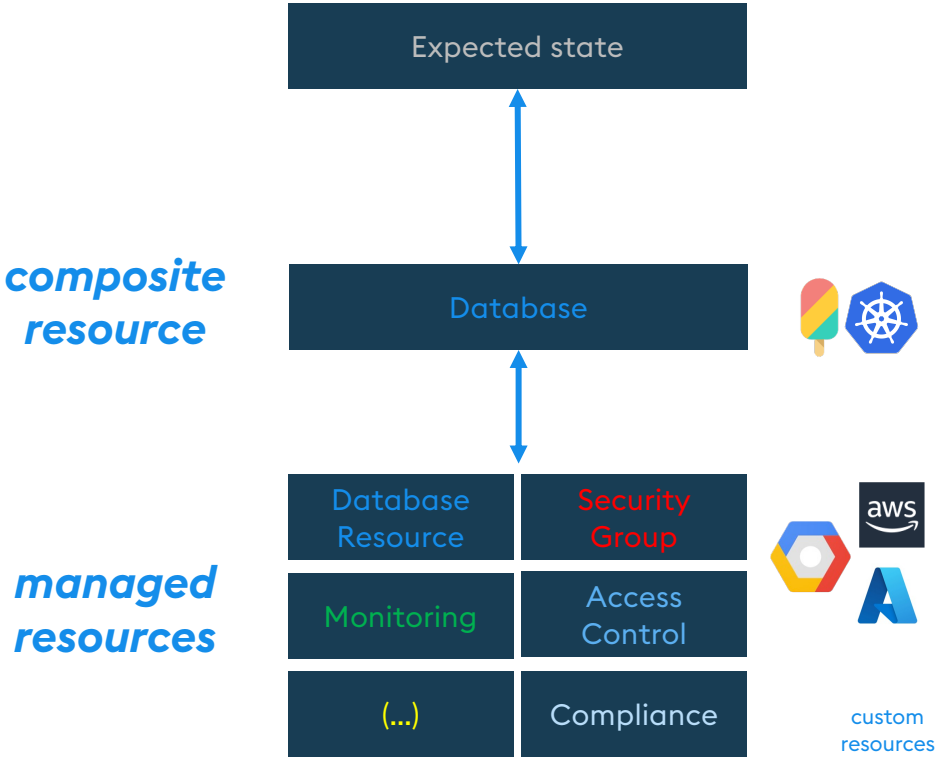
Crossplane – Composite resources

Managed resources can be aggregated into ***composite resources***.

This can be used to provide a better user experience, for example by simplifying the interface or enforcing compliance.

The API for a composite resource is defined by a ***Composite Resource Definition*** (XRD)

Its implementation is described in a ***Composition***.



DKB
Deutsche Kreditbank AG

Crossplane example

composite resource

Crossplane ***PostgreSQL claim***

```yaml
apiVersion: rds.aws.dkb.cloud/v1alpha1
kind: DBInstance
metadata:
  name: example-dbinstance
  namespace: example-db
  labels:
    controlling.dkb.cloud/cost-type: "854749"
    controlling.dkb.cloud/owner: dpt-test_AT_dkb.ag
    tags.dkb.cloud/account: "0000000001"
    tags.dkb.cloud/zone: "dev"
    tags.dkb.cloud/environment: "dev"
    tags.dkb.cloud/protection-requirement: "dev"
  annotations:
    kustomize.toolkit.fluxcd.io/prune: disabled
spec:
  forProvider:
    region: eu-central-1
    allocatedStorage: 20
    autoMinorVersionUpgrade: true
    autogeneratePassword: true
    backupRetentionPeriod: 14
    dbInstanceClass: db.t3.micro
    dbName: example
    engine: postgres
    preferredBackupWindow: "7:00-8:00"
    preferredMaintenanceWindow: "Sat:8:00-Sat:11:00"
    publiclyAccessible: false
    skipFinalSnapshot: true
    storageType: gp2
  writeConnectionSecretToRef:
    name: example-dbinstance-connection
  providerConfigRef:
    name: 0000000001-cat
```

(claim = request for a composite resource)

`kubectl apply`

managed resources

RDSInstance

```
Db:
  API Version:  database.aws.crosspla
  Kind:         RDSInstance
  Name:                              269j
  Status:
    At Provider:
      Db Instance Arn:     arn:aws:rd
      Db Instance Status:  configurin
      Db Parameter Groups:
        Db Parameter Group Name:  def
        Parameter Apply Status:   in-
      Db Resource Id:             db-
      Db Subnet Group:
        Db Subnet Group Description:
        Db Subnet Group Name:
        Subnet Group Status:
```

DBSubnetGroup

```
Cnp - Postgres - Dbsubnetgroup:
  API Version:  database.aws.crossplane.io/
  Kind:
  Name:                                -269j4
  Status:
    At Provider:
      Arn:     arn:aws:rds:eu-cent
      State:  Complete
      Subnets:
        Subnet ID:      subnet-7
        Subnet Status:  Active
        Subnet ID:      subnet-c
        Subnet Status:  Active
        Subnet ID:      subnet-4
        Subnet Status:  Active
      Vpc Id:           vpc-348d
    Conditions:
      Last Transition Time:  2021-10-07T09
      Reason:                Available
```

SecurityGroup

```
Sg:
  API Version:  ec2.aws.crossplane.io/v1beta1
  Kind:         SecurityGroup
  Name:                              -269j4
  Status:
    At Provider:
      Owner Id:           430
      Security Group ID:  sg-(              3c5
    Conditions:
      Last Transition Time:  2021-10-07T09:58:
      Reason:                Available
      Status:                True
      Type:                  Ready
      Last Transition Time:  2021-11-09T11:30:
      Reason:                ReconcileSuccess
      Status:                True
      Type:                  Synced
```

DKB
Deutsche Kreditbank AG

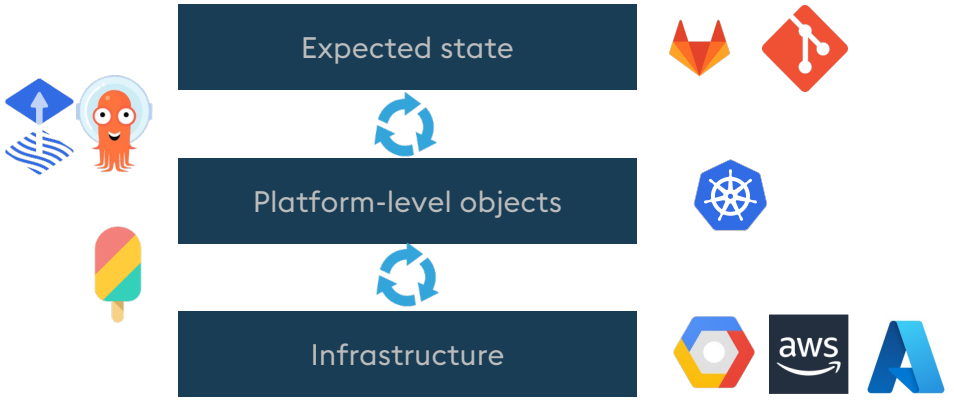
Crossplane and GitOps

The expected state is stored in a code repository, and synchronized using ***GitOps***.

GitOps principles:

- git repository describing the infrastructure is synchronized with the platform
- manual platform operations are discouraged

This can be implemented with ***Argo CD*** or ***Flux*** for example.



https://about.gitlab.com/topics/gitops/

DKB
Deutsche Kreditbank AG

DKB
Deutsche Kreditbank AG

Lessons learned
Team organization

Start small and focused

- Spend time directly with customer stakeholders in the beginning, rather than only with internal organization
- Start with a small platform engineering team and expand when needed

Aim for short feedback loops

- Start with an “officially experimental” platform (MVP, not prototype)
- Identify the best pilot users (easy to work with, provide good feedback)
- Clearly document the state of the platform features

Lessons learned
Communication with consumers

Explain the platform concepts well

- Crossplane involves a lot of abstraction that is not trivial for everyone, especially for developers not used to cloud-native.

Be ready to assist consumers technically

- Readily available developer information usually documents access to the cloud API, not Crossplane. Developers will need help to convert this to Crossplane.
- “Embed” engineers in the consumer teams if needed, or create a specific sub-team.

Define consumer application components well

- Clarify what are the important components and how their health can be assessed.
- Good monitoring + clear SLIs/SLOs are very useful to quickly identify issues when troubleshooting.

DKB
Deutsche Kreditbank AG

Lessons learned
GitOps and Operations

Separate application code and expected resources state

- use different directories or repos
- standardize the structure as much as possible
- simplifies operations and troubleshooting

Work on the main branch for expected resources state

- allows to troubleshoot effectively, especially when different people are involved
- allows for reviews of the environment without direct production access

Train Operations to work with Crossplane

- Operations are typically not used to work with GitOps and reconciliation
- Operations must understand very well how the infrastructure is defined, so that they can easily make modifications

DKB
Deutsche Kreditbank AG