Terraform vs Crossplane: Building a self-service platform for developers

By: Lars Magelssen

Once upon a time
IaC was created









What should you use?

It depends

What is Crossplane?

How does Crossplane work?

Why use Crossplane?

How can I get a burger with fries
when I know nothing about animal husbandry
or growing potatoes?

You don't need to know this





sopra steria



Option 1: Order take out

- Make an order
- Delivery guy delivers order
- Burger restaurant marks order as completed



Someone spilled your coke



Option 2: Restaurant

- Make an order
- Waiter delivers the order
- The waiter stays in the room observing the tables

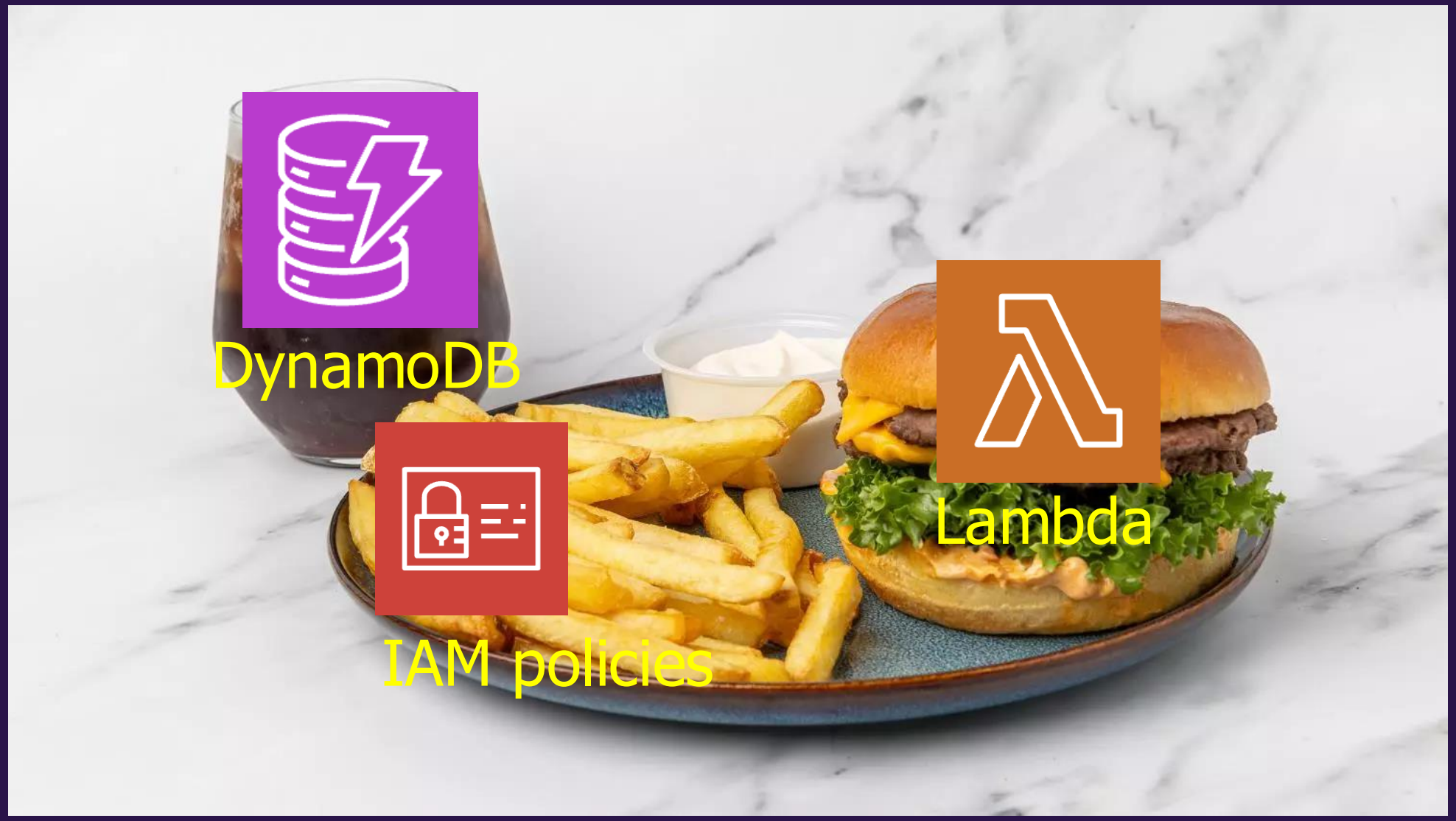


Someone spilled your coke





Cloud resources



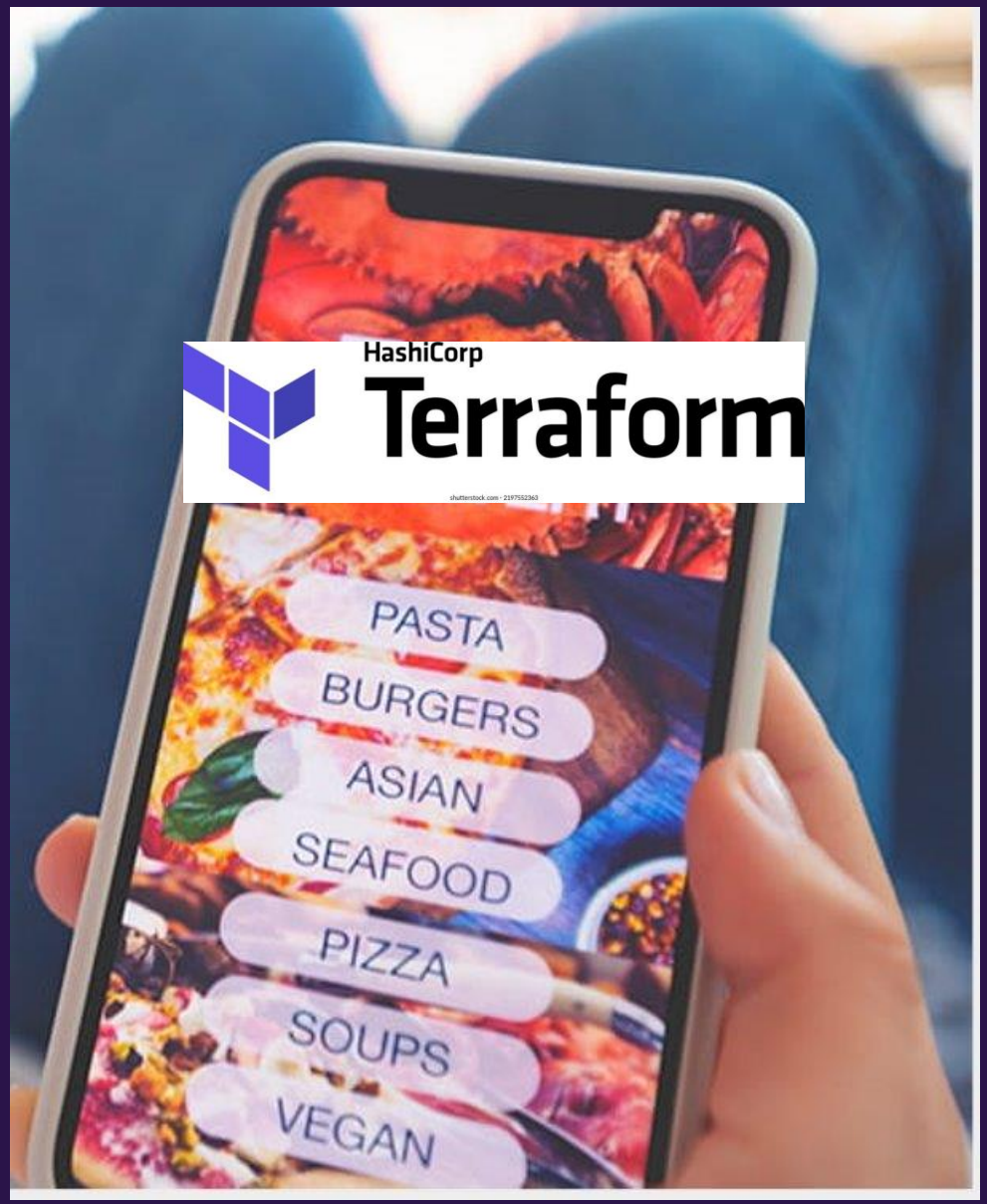
Option 1: Terraform

- Make order
- Deliver order
- Begin state mark as completed

A Terraform configuration

Run terraform apply main.tf

The state is stored in a state file



Someone manually deleted something = Drift



sopra steria

Option 2: Crossplane

- Make an order
- Wait delivers the order
- The waiter stays in the room observing the tables

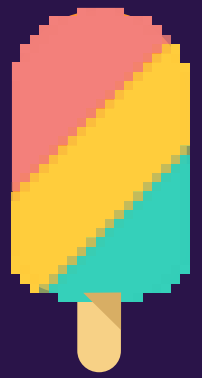
Apply manifest burger.yaml

Crossplane controller reconciling

control loop




DO YOU UNDERSTAND CROSSPLANE?
NOT YET
imgflip.com



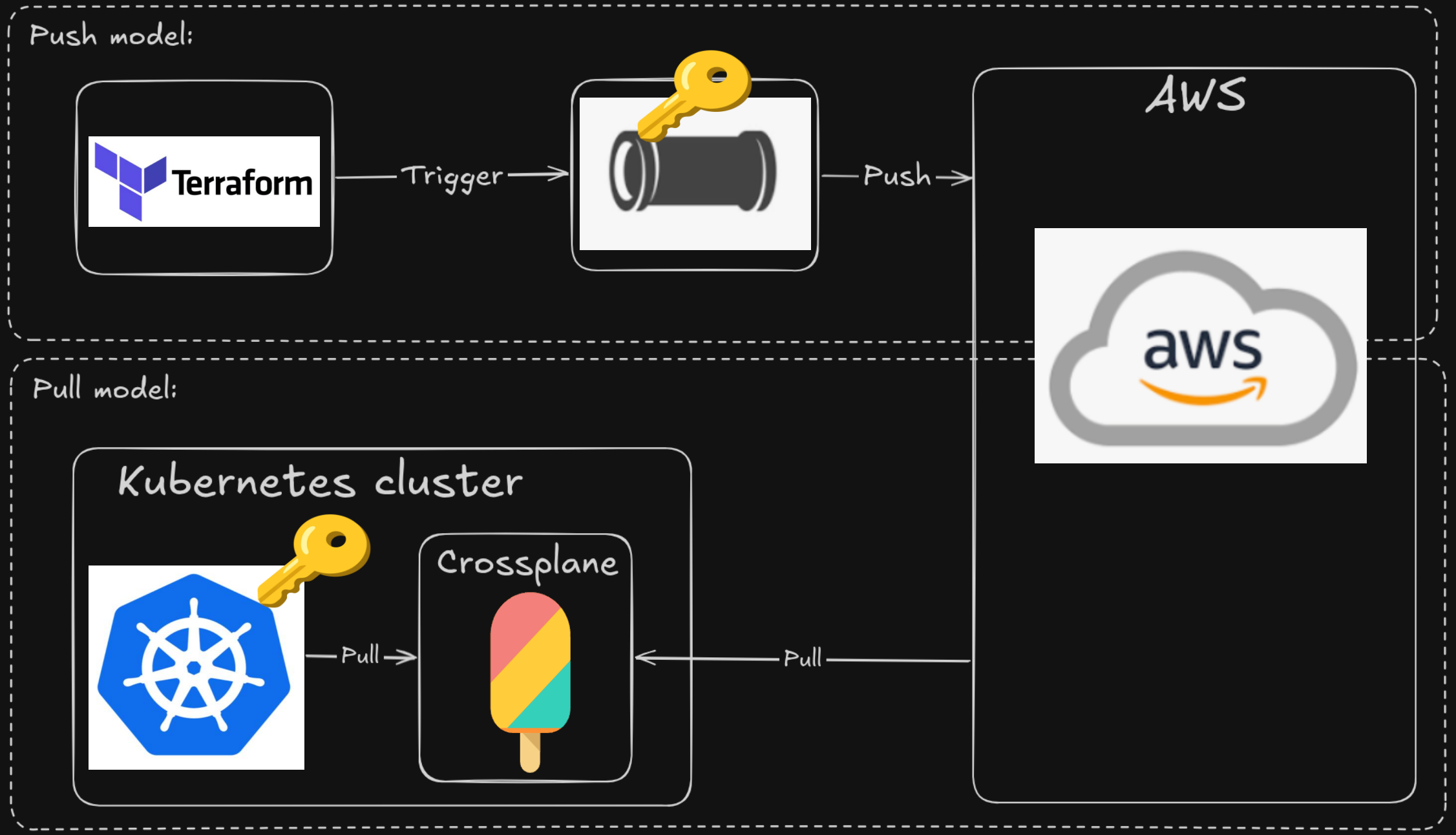
Crossplane

sopra steria



Crossplane

- Infrastructure as Code
- Runs in Kubernetes
- Pull model (not push)


Push model:
Terraform
Trigger
Push
AWS
aws
Pull model:
Kubernetes cluster
Crossplane
Pull
Pull



Crossplane

- Infrastructure as Code
- Runs in Kubernetes
- Pull model (not push)
- Control loop


Crossplane



Crossplane

- Infrastructure as Code
- Runs in Kubernetes
- Pull model (not push)
- Control loop
- Self-service platform



Crossplane

- Infrastructure as Code
- Runs in Kubernetes
- Pull model (not push)
- Control loop
- Self-service platform

- Who is the consumer?



sopra steria

- Who is the consumer?
- What resources are you provisioning?

Crossplane



Terraform

sopra steria

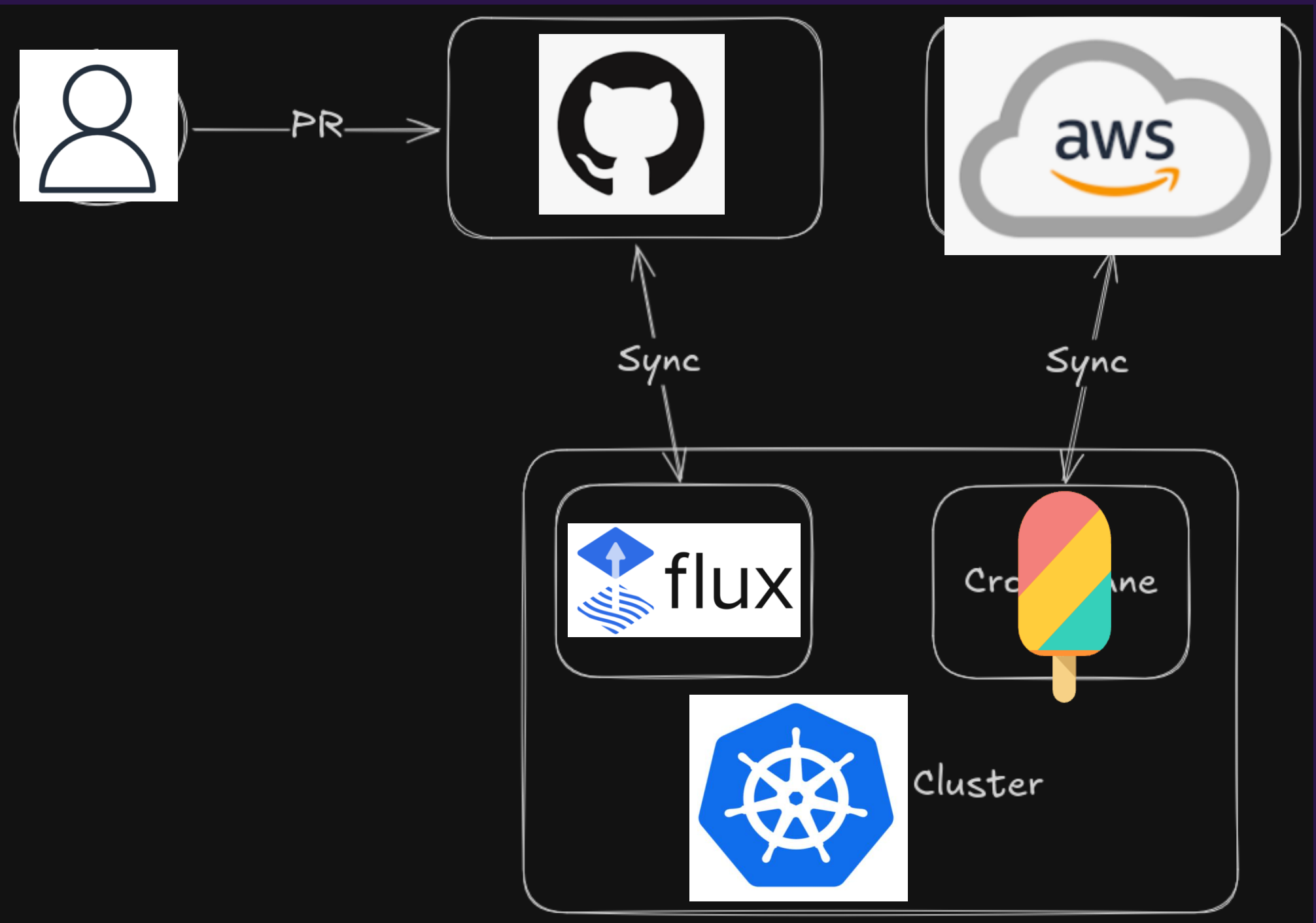
The developers perspective

```
{
   "AttributeDefinitions": [
      {
         "AttributeName": "string",
         "AttributeType": "string"
      }
   ],
   "BillingMode": "string",
   "DeletionProtectionEnabled": boolean,
   "GlobalSecondaryIndexes": [
      {
         "IndexName": "string",
         "KeySchema": [
            {
               "AttributeName": "string",
               "KeyType": "string"
            }
         ],
         "OnDemandThroughput": {
            "MaxReadRequestUnits": number,
            "MaxWriteRequestUnits": number
         },
         "Projection": {
            "NonKeyAttributes": [ "string" ],
            "ProjectionType": "string"
         },
         "ProvisionedThroughput": {
            "ReadCapacityUnits": number,
            "WriteCapacityUnits": number
         },
         "WarmThroughput": {
            "ReadUnitsPerSecond": number,
            "WriteUnitsPerSecond": number
         }
      }
   ],
   "GlobalTableSettingsReplicationMode": "string",
   "GlobalTableSourceArn": "string",
   "KeySchema": [
      {
         "AttributeName": "string",
         "KeyType": "string"
      }
   ],
   "LocalSecondaryIndexes": [
      {
         "IndexName": "string",
         "KeySchema": [
            {
               "AttributeName": "string",
               "KeyType": "string"
            }
         ],
         "Projection": {
            "NonKeyAttributes": [ "string" ],
            "ProjectionType": "string"
         }
      }
   ],
   "OnDemandThroughput": {
      "MaxReadRequestUnits": number,
      "MaxWriteRequestUnits": number
   },
   "ProvisionedThroughput": {
      "ReadCapacityUnits": number,
      "WriteCapacityUnits": number
   },
   "ResourcePolicy": "string",
   "SSESpecification": {
      "Enabled": boolean,
      "KMSMasterKeyId": "string",
      "SSEType": "string"
   },
   "StreamSpecification": {
      "StreamEnabled": boolean,
      "StreamViewType": "string"
   },
   "TableClass": "string",
   "TableName": "string",
   "Tags": [
      {
         "Key": "string",
         "Value": "string"
      }
   ],
   "WarmThroughput": {
      "ReadUnitsPerSecond": number,
      "WriteUnitsPerSecond": number
   }
}
```

```yaml
apiVersion: platform.example.com/v1alpha1
kind: DynamoDBTable
metadata:
  name: users
spec:
  tableName: users
  partitionKey:
    name: userId
    type: S
```

GitOps flow

1. Developer create Claim
2. Argo/Flux syncs the cluster
3. Crossplane reconciles creating the Composite Resource syncing against AWS
4. The cloud resources appear in AWS


PR
aws
Sync
Sync
flux
Cluster

Summary

- IaC in Kubernetes
- Not a full replacement
- Developers easily order resources





Thank you

Lars Magelssen

https://magelssen.io

Let's connect

(BTW, we're hiring!)

sopra steria