appvia

Self-service of Cloud Resources for Kubernetes Applications

About Me

Lewis Marshall
SRE, Tech Evangelist, Appvia

28 years of development and operations...

- x86 Assembly
- Golang, Kubernetes and Cloud

phoenix technologies

DK

Pfizer

London Stock Exchange

Home Office

Appvia.io

Self-service of Cloud Resources for Kubernetes Applications

Lewis Marshall, Tech Evangelist, SRE, Kore Developer

9th June, 2020

Agenda

Self-service of Cloud Resources for Kubernetes Applications

Intro - Why it isn’t easy!

What we learnt from doing self-service

The industry is moving

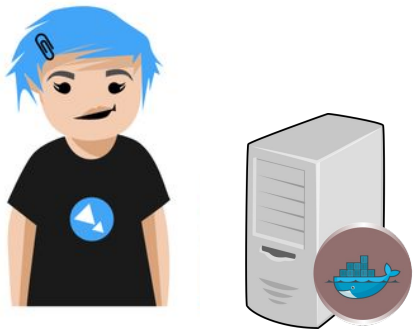
- Lots of products in the industry around cloud infrastructure
- Not many are developer focused
- Developer focused solutions seem to be being replaced with:
  - Operations heavy solutions
  - Domain / Cloud specific solutions
- We had to implement a custom solution around the industry to get a good outcome

Problem and Developer Experience

How it is for a Developer

Developer Mindset

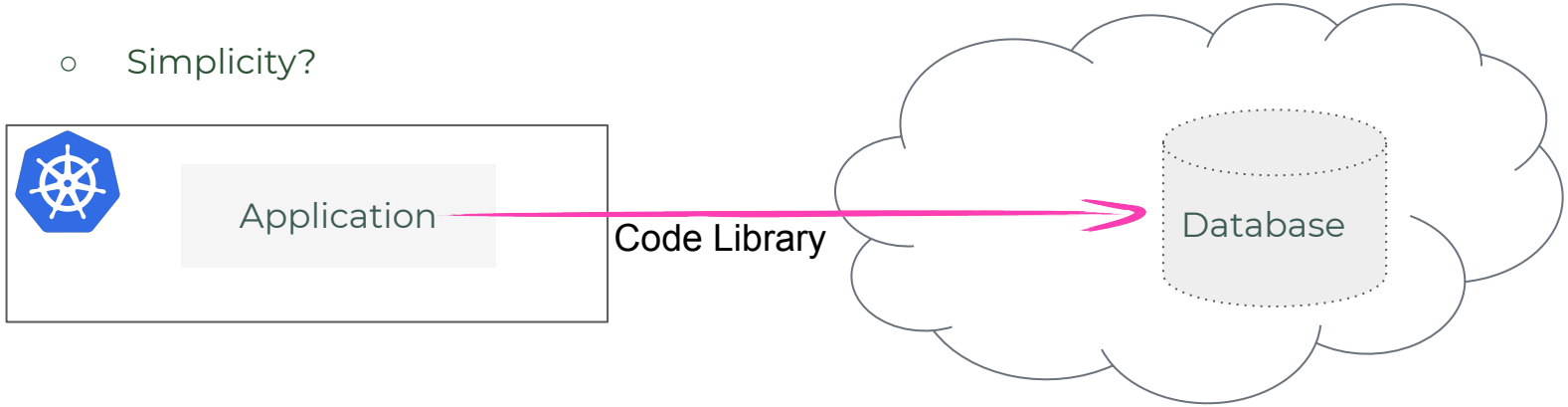
- Local container focused development
- Fast to iterate and test when local
- Dependencies are libraries and containers and not directly cloud
- Velocity drops as cloud services get introduced
- Cloud consumption is brokered and gated and requires specialist domain knowledge



What the developer need is

Applications consume cloud services via libraries

- Application in a Kubernetes Cluster
- Cloud provider managed services:
  - Operational overhead removed from team
  - Reliability
  - Simplicity?

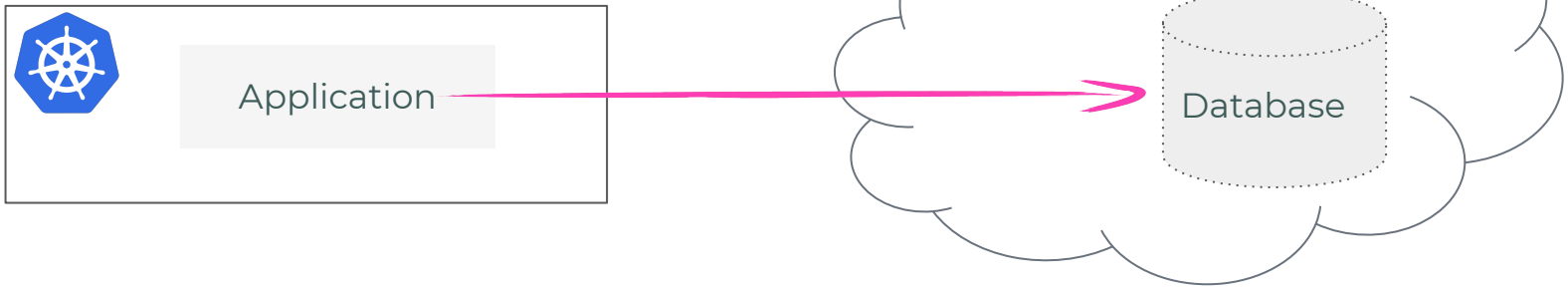


What’s the Problem?

Provisioning Applications and Dependant Services

Developer

Operations



What’s the Problem?

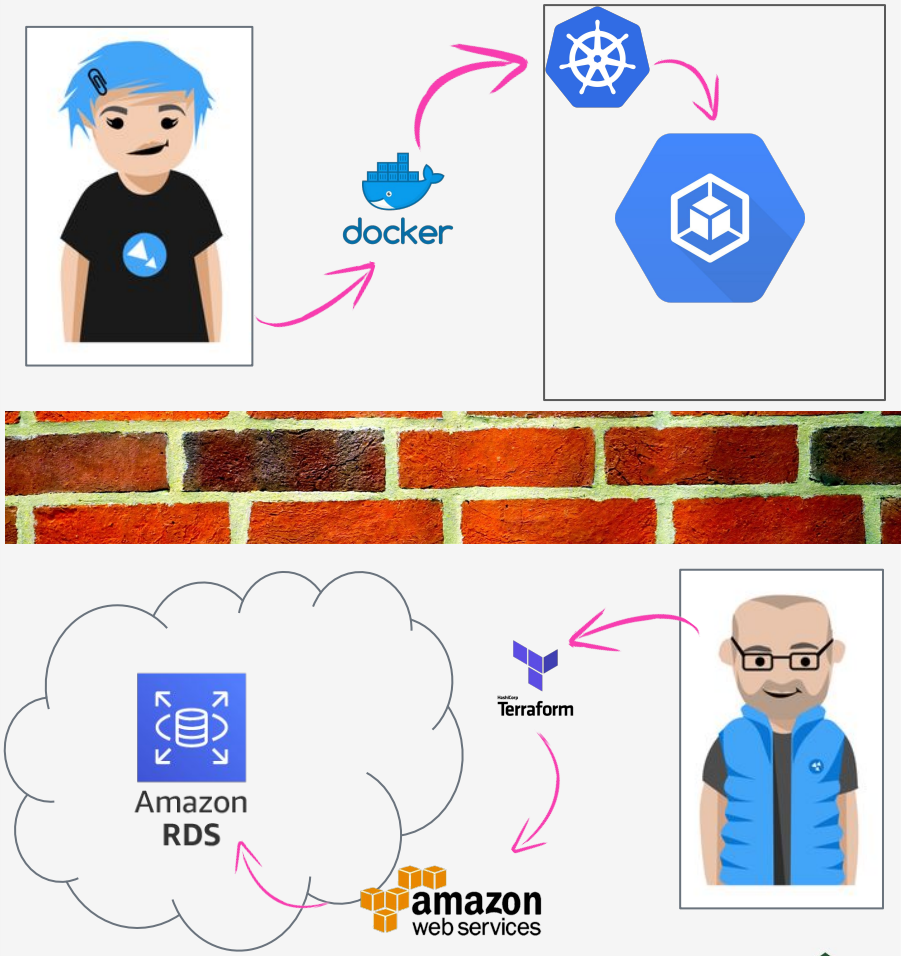
Delivery Tools

Developers

- Create Containerized Applications
- Deliver to Kubernetes

Operations

- Automation Tools
- Deliver Cloud Resources



Developer Self-Service

Why Self Service?

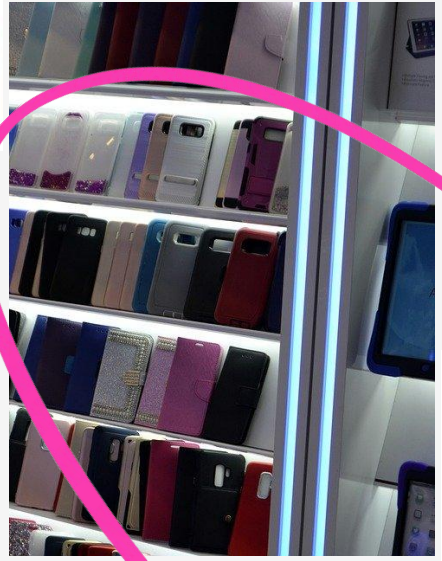
Cloud Services

- Reduce lead times
  - No separate team
  - No manual intervention
  - Nothing to approve
- Reliability of updates
  - Application
  - Cloud
- Enables an Agile process
  - Reduces cost

Self Service

Limiting Risk with Informed Choice

- Best Practice
  - Simplify Choice e.g. Backups, Cost, Encryption...
- Cost
  - Trust Staff
  - Reduction with Agility and Reliability
- Security
  - Enables Best Practice
  - Products over Human Error



Developer Self Service

Ideal Flow per Application Instance

Service Description Name & Plan

1. Request

?

Application

2. Access Configuration E.g. endpoint & credentials

Any Cloud Provider

Self Service In Practice

Industry Assumptions

Kubernetes Resources

- Native Kubernetes
  - Documentation
  - Resource handling
  - Application domain
- Extended Kubernetes
  - Not simple
  - Domain specific
    - Cloud / Ops knowledge
    - What's the Dev benefit?

Reconcile Intended State

Custom Resources for Cloud Services

A Cloud Resource

A Kubernetes resource like any other:

```yaml
apiVersion: redis.cnrm.cloud.google.com/v1beta1
kind: RedisInstance
metadata:
 labels:
   label-one: "value-one"
 name: cache-appfe
spec:
 displayName: Sample Redis Instance
 region: us-central1
 tier: BASIC
 memorySizeGb: 16
```

```yaml
apiVersion: service-operator.aws/v1alpha1
kind: ElastiCache
metadata:
 name: cache-appfe
spec:
 cacheSubnetGroupName: "loadtest-cluster-k8s"
 vpcSecurityGroupIds: "sg-0581b94aa3c0db58c"
 autoMinorVersionUpgrade: true
 engine: redis
 engineVersion: 5.0.0
 cacheNodeType: "cache.m4.large"
```

Use familiar tools and deployment systems

```
# find configured databases
kubectl get RedisInstance
```

```
# manually change some settings
kubectl edit RedisInstance/cache-appfe
```

Scaling custom resources

Unique specs for each service

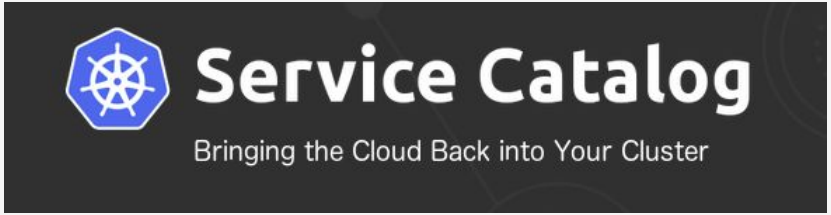
- Different specs for each cloud
- Domain knowledge expected on each service
- No consistency between cloud providers for the Developer
- No guidance on security and best practice
- No high level abstraction

Service Catalog

Open Service Broker API on Kubernetes

Provides:

- Reuse Service Brokers from CloudFoundry
- Provides Custom Resources



kubernetes-sigs/service-catalog

svc-cat.io

Production Ready **Amber**

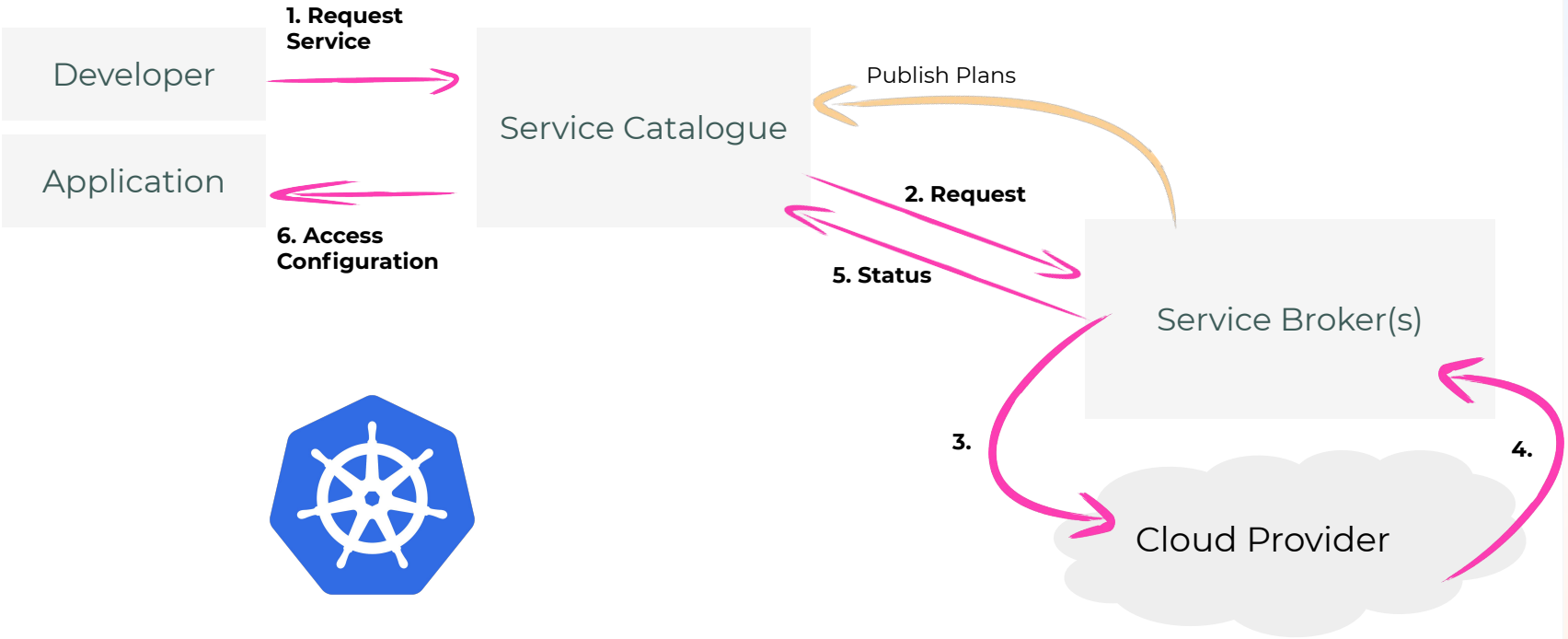
Service Broker Plans

Simplify delivery of Cloud Services

- Provide default parameters for services
  - Best Practice
- What is published in a system
  - Vetted Services

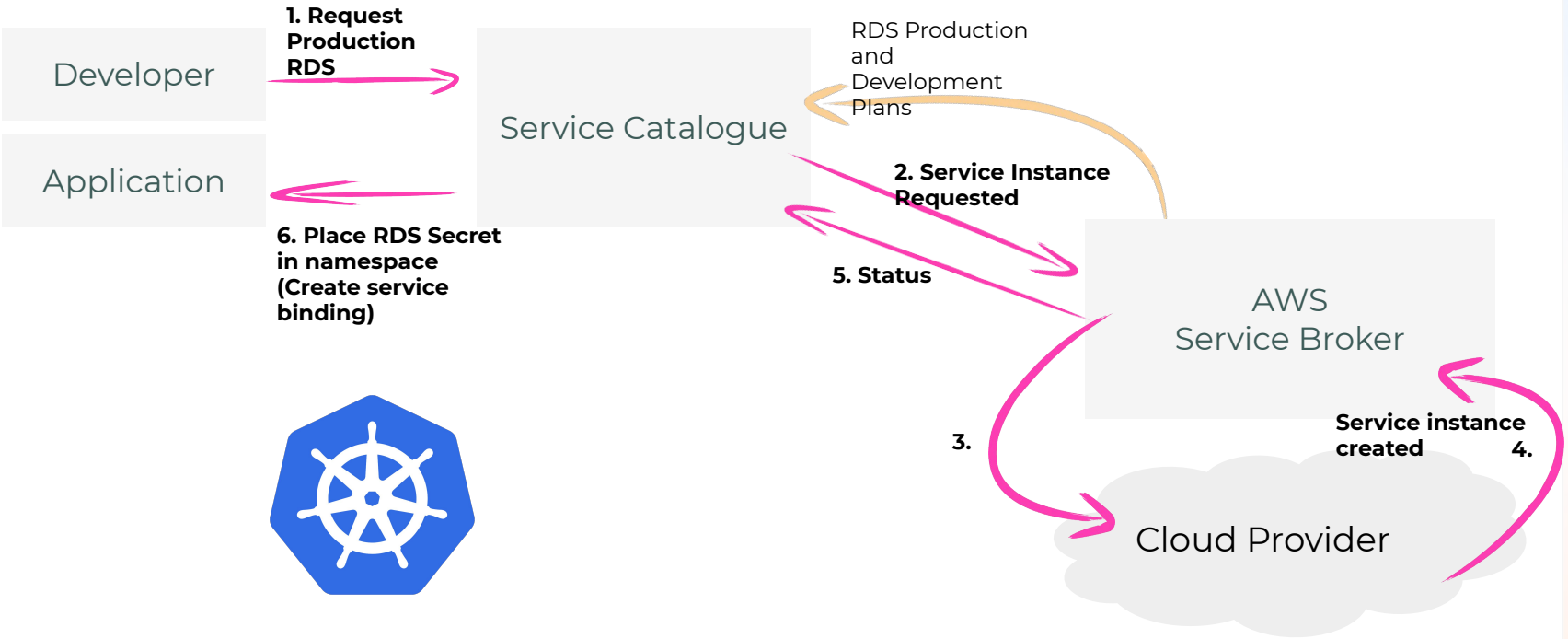
Open Service Broker API

Request Flow



Open Service Broker API

Request Flow Concrete Example

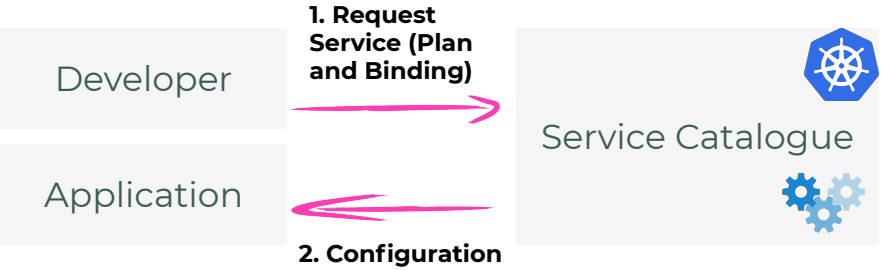


Demo of Custom Resources

Demo of Cloud Resources

Service Catalog

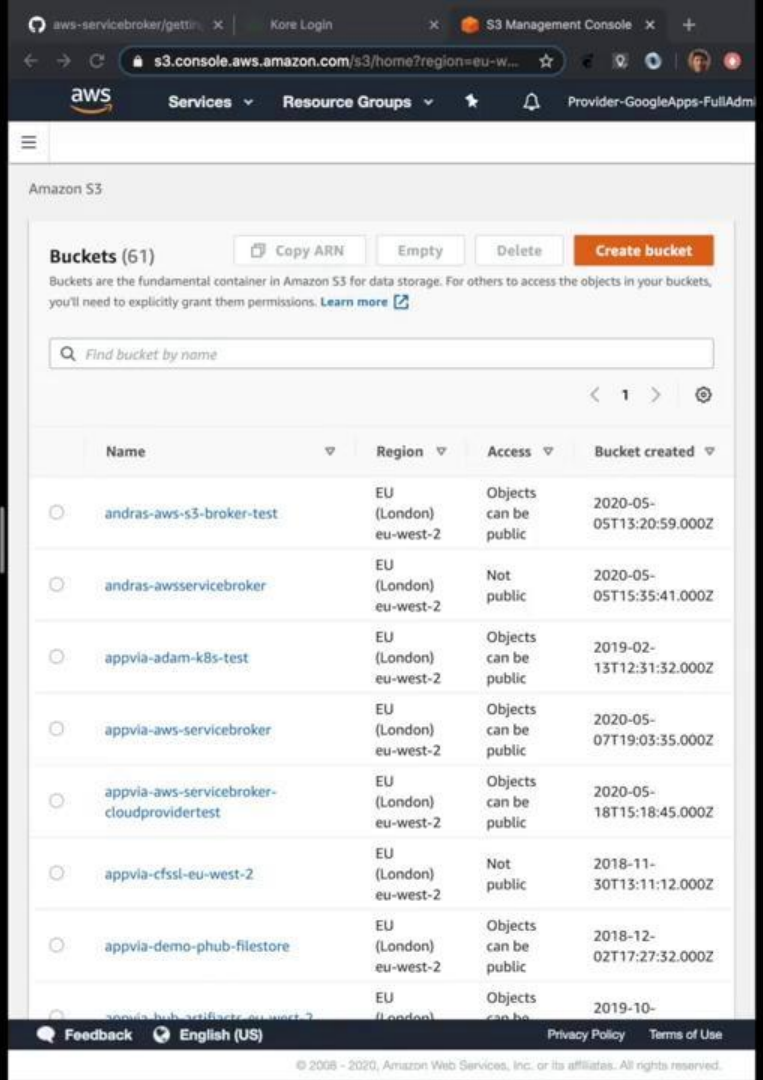
- Use of Plans
- Consume Application Configuration




```yaml
apiVersion: servicecatalog.k8s.io/v1beta1
kind: ServiceInstance
metadata:
 name: my-latest-bucket
spec:
 clusterServiceClassExternalName: s3
 clusterServicePlanExternalName: production
 parameters:
---
apiVersion: servicecatalog.k8s.io/v1beta1
kind: ServiceBinding
metadata:
 name: my-latest-bucket
spec:
 instanceRef:
   name: my-latest-bucket
 secretName: my-latest-bucket
```

```
            secretKeyRef:
              name: my-latest-bucket
              key: S3_REGION
        - name: AWS_ACCESS_KEY_ID
          valueFrom:
            secretKeyRef:
              name: my-latest-bucket
              key: S3_AWS_ACCESS_KEY_ID
        - name: AWS_SECRET_ACCESS_KEY
          valueFrom:
            secretKeyRef:
              name: my-latest-bucket
              key: S3_AWS_SECRET_ACCESS_KEY
        image: mikesir87/aws-cli
        imagePullPolicy: IfNotPresent
        name: aws-cli
[lewismarshall@lewiss-mbp aws]$ kubectl get secrets
NAME                  TYPE                                  DATA   AGE
default-token-4zxcf   kubernetes.io/service-account-token   3      3h43m
my-latest-bucket      Opaque                                6      63m
[lewismarshall@lewiss-mbp aws]$ kubectl get secrets my-latest-bucket -o yaml | grep BUCKET
  BUCKET_ARN: YXJuOmF3czpzMzo6OmF3cy1zZXJ2aWNlLWJyb2tlci1zMy01ZmExYjM2ZS05OWMtczNidWNrZXRyZXR
haW4tMTU3ZjJ4MDB2YnFvMA==
  BUCKET_NAME: YXdzLXNlcnZpY2UtYnJva2VyLXMzLTVmYTFiMzZlLTk5Yy1zM2J1Y2tldHJldGFpbi0xNTdmMngwMH
ZicW8w
  LOGGING_BUCKET_NAME: YXdzLXNlcnZpY2UtYnJva2VyLXMzLTVmYTFiMzZlLTk5Y2YtbG9nZ2luZ2J1Y2tldC1ycW
l5ZDM0c2E5am4=
[lewismarshall@lewiss-mbp aws]$ kubectl apply -f awscli-deploy.yaml
deployment.apps/aws-cli created
[lewismarshall@lewiss-mbp aws]$ kubectl get po
NAME                      READY   STATUS    RESTARTS   AGE
aws-cli-b9b8bb5db-dqpxv   1/1     Running   0          6s
[lewismarshall@lewiss-mbp aws]$ kubectl exec -it aws-cli-b9b8bb5db-dqpxv bash
root@aws-cli-b9b8bb5db-dqpxv:/aws# aws s3 ls ${BUCKET_NAME} [lewismarshall@lewiss-mbp aws]$
[lewismarshall@lewiss-mbp aws]$ kubectl exec -it aws-cli-b9b8bb5db-dqpxv bash
root@aws-cli-b9b8bb5db-dqpxv:/aws# aws s3 ls s3://${BUCKET_NAME}
root@aws-cli-b9b8bb5db-dqpxv:/aws# aws s3 cp /etc/resolv.conf s3://${BUCKET_NAME}/
upload: ../etc/resolv.conf to s3://aws-service-broker-s3-5fa1b36e-99c-s3bucketretain-157f2x00
vbqo0/resolv.conf
root@aws-cli-b9b8bb5db-dqpxv:/aws# aws s3 ls /etc/resolv.conf s3://${BUCKET_NAME}/

Unknown options: s3://aws-service-broker-s3-5fa1b36e-99c-s3bucketretain-157f2x00vbqo0/
root@aws-cli-b9b8bb5db-dqpxv:/aws# aws s3 ls s3://${BUCKET_NAME}/
2020-05-19 14:01:21        131 resolv.conf
root@aws-cli-b9b8bb5db-dqpxv:/aws#
```



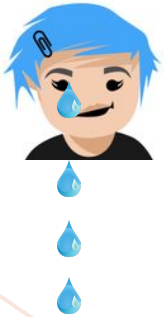
Industry Summary

Product Comparison

Product	Application Services	Time Investment	Production Ready	Developer Self Service
AWS Service Broker	22	**High - Cloudformation**	- Future	**Plans**
AWS Service Operator	0	NA	**- MVP**	Too early
Azure Open Service Broker	3		- No Recent Updates	**Plans**
GCP Service Broker	3	**High**	- Not supported direction	**Plans**
GCP Config Connector	7	**High**	- PSP Required	- **Docs** - **Infrastructure Focus,** - **Implied Config**
Crossplane	~4 / Cloud	**High**		**OK** - **Open App Model** - **Traits** - **More Infrastructure**
Terraform K8	All	**High**	- Alpha	Infrastructure Focus
Terraform Controller	All	**High**	- Experimental	Infrastructure Focus

Self Service For Developers?

Self Service For Developers



Cloud Vendors Direction

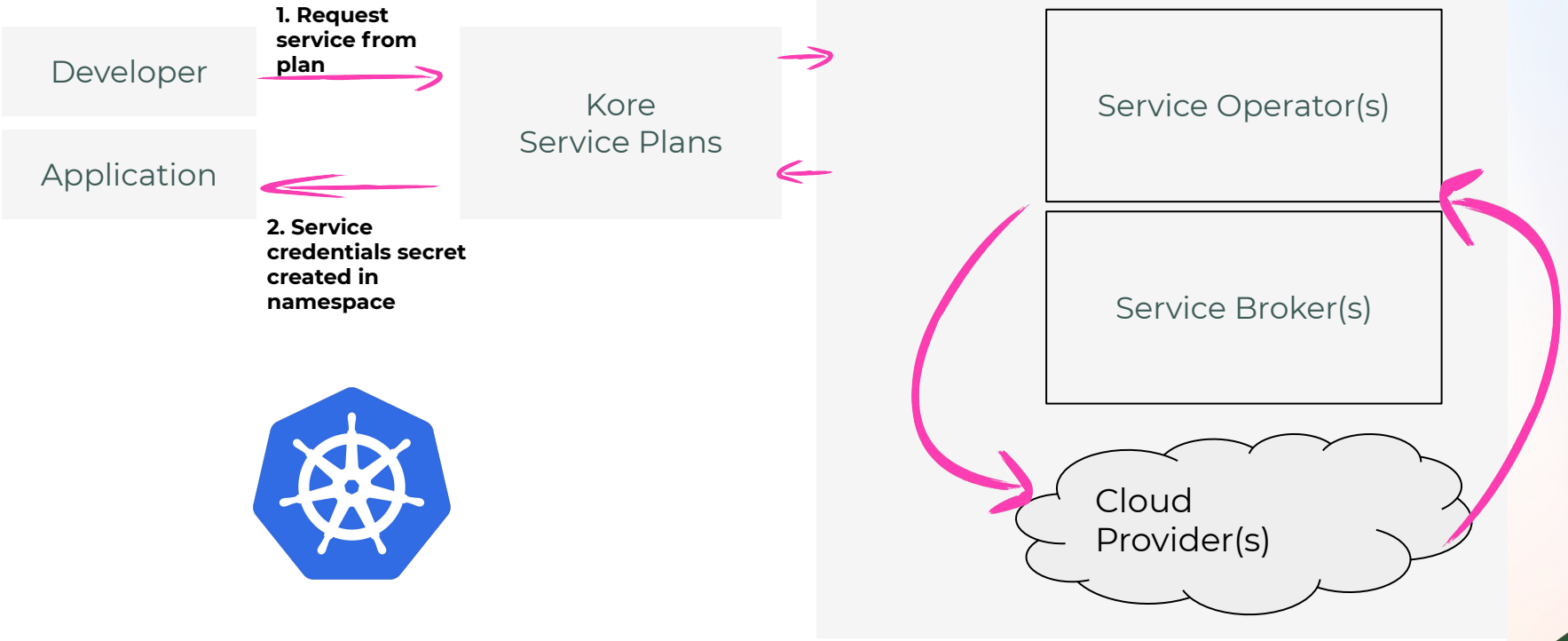
Motives

- Commercial
- Support many customers
  - Reliability at scale
  - Self support
- NOT multi-cloud
- ~~Save customers time and money?~~

Appvia Approach

Kore Architecture for Cloud Resource

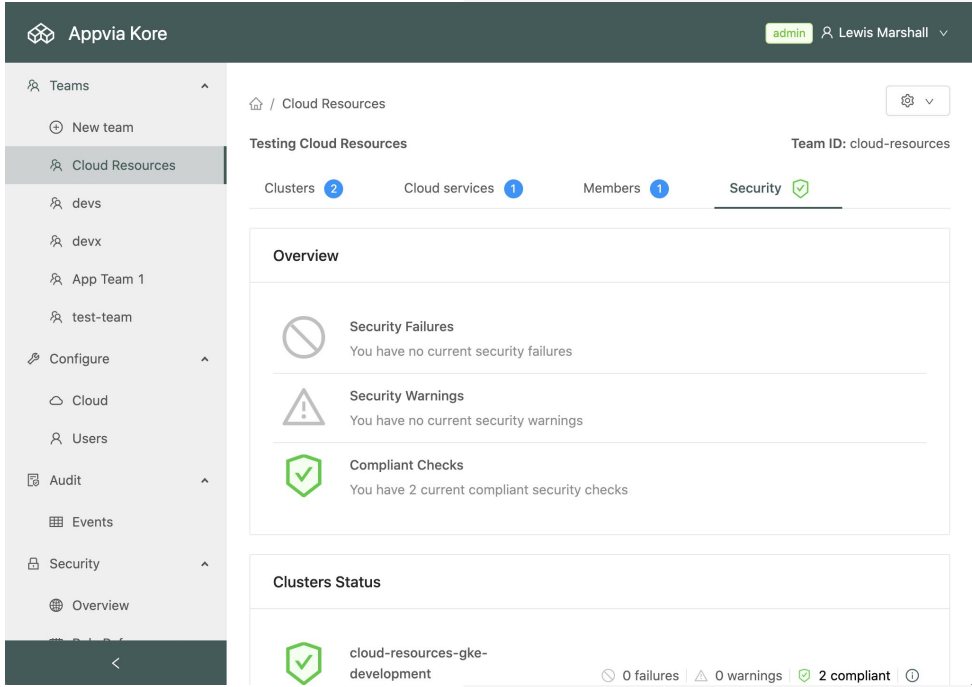
Kore



Demo Appvia Ideas

Demo Cloud Resources

- Self Service
  - Plans
  - Simplicity
- Running Application
  - Consuming Cloud Resources



Appvia Kore

admin Lewis Marshall

Teams
- New team
- A Team
- devs
- devx
- App Team 1
- test-team

Configure
- Cloud
- Users

Audit

Security
- Overview
- Rule Reference

Cluster access

1 Download

If you haven't already, download the CLI from https://github.com/appvia/kore/releases

2 Setup profile

Create a profile

```
kore profile configure api.qa.kore.appvia.io
```

Enter the Kore API URL as follows

```
https://api.qa.kore.appvia.io
```

3 Login

Login to the CLI

```
kore login
```

4 Setup access

Then, you can use the Kore CLI to setup access to your team's clusters

```
kore kubeconfig -t a-team
```

This will add local kubernetes configuration to allow you to use `kubectl` to talk to the provisioned cluster(s).

Summary

Summary

- Current Solutions for Operations
  - Enable Complexity
  - Not Agility
- Plans Provide
  - Simplicity for Developers
    - Agile
  - Best Practice
    - Oversight
    - Audit
    - Compliance

About Appvia

Kubernetes for Teams

5+ years Kubernetes experience

70% cost reduction

amazon web services | Partner Network

CONSULTING PARTNER

CLOUD NATIVE COMPUTING FOUNDATION

kubernetes CERTIFIED SERVICE PROVIDER

Google Cloud Partner

700+ Developers

500+ applications

Home Office

BANK OF ENGLAND

Cloud Resources with Kubernetes - Questions?

Contact Us

Lewis Marshall
SRE, Tech Evangelist, Appvia

info@appvia.io

appvia.io

appvia/kore