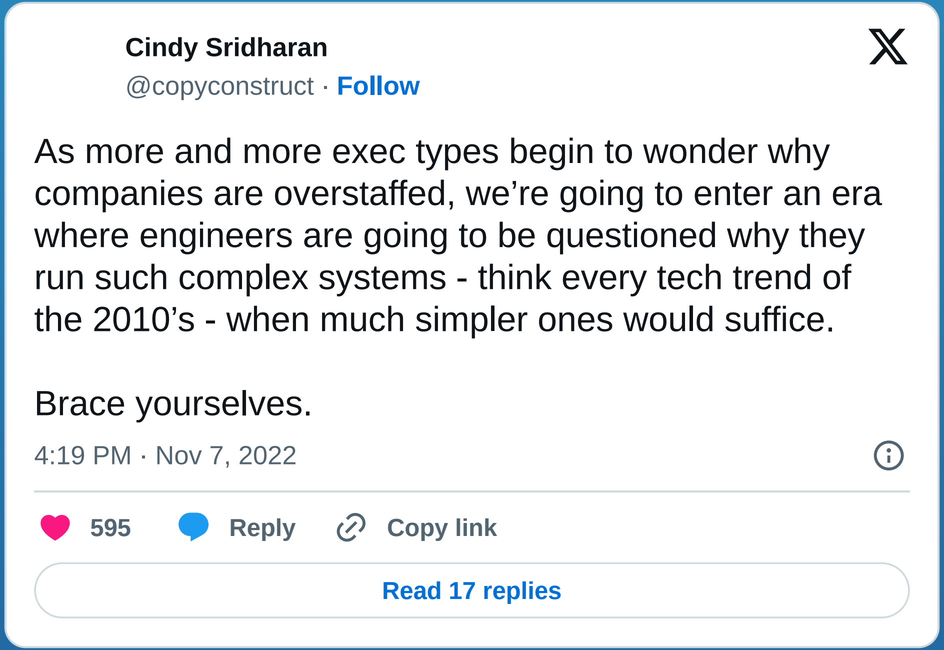
The operator antipattern

Kubernetes Community Days London 2024

Gerald Schmidt

DS Smith



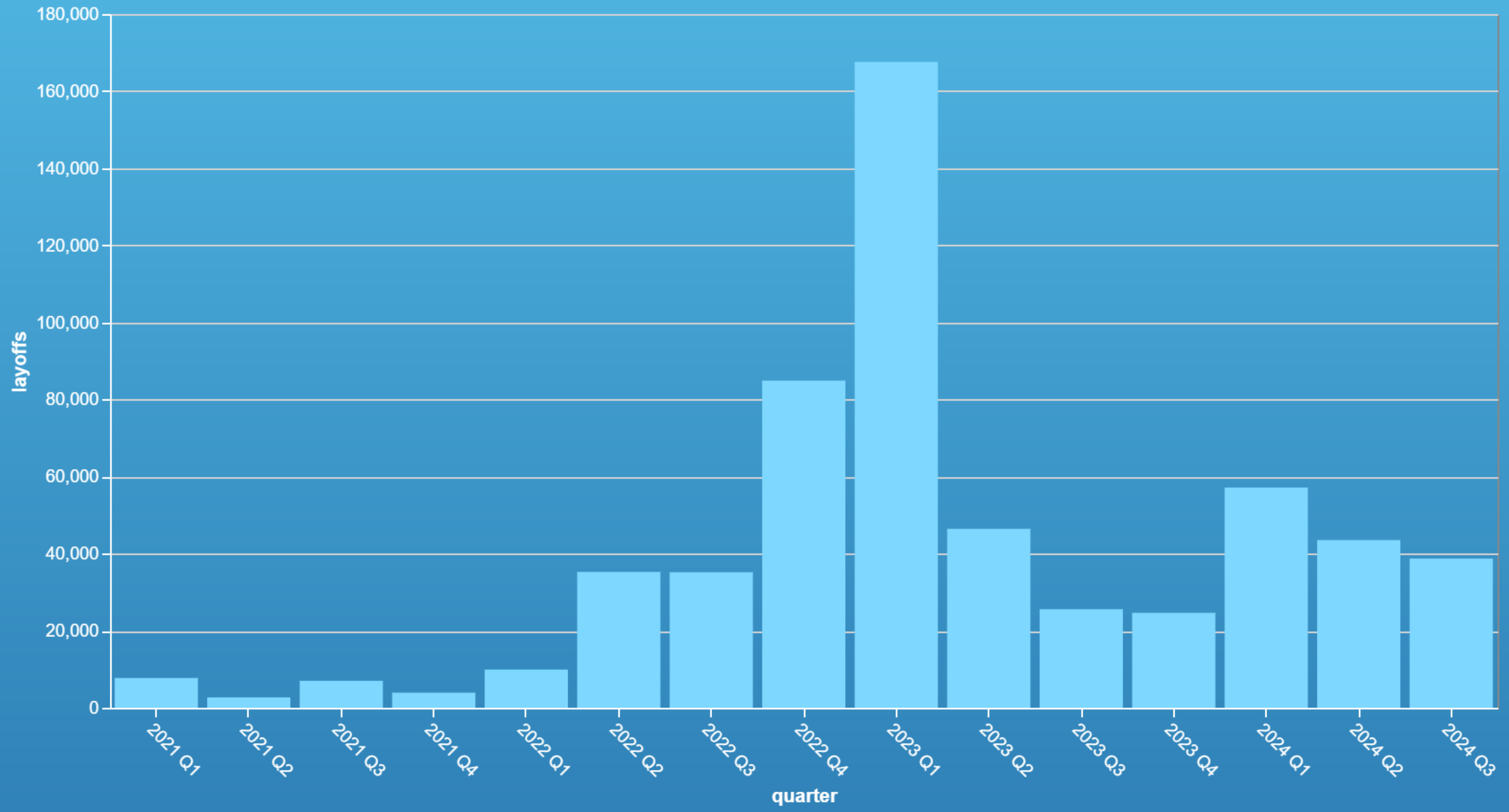
Cindy Sridharan
@copyconstruct ·

As more and more exec types begin to wonder why companies are overstaffed, we're going to enter an era where engineers are going to be questioned why they run such complex systems - think every tech trend of the 2010's - when much simpler ones would suffice.

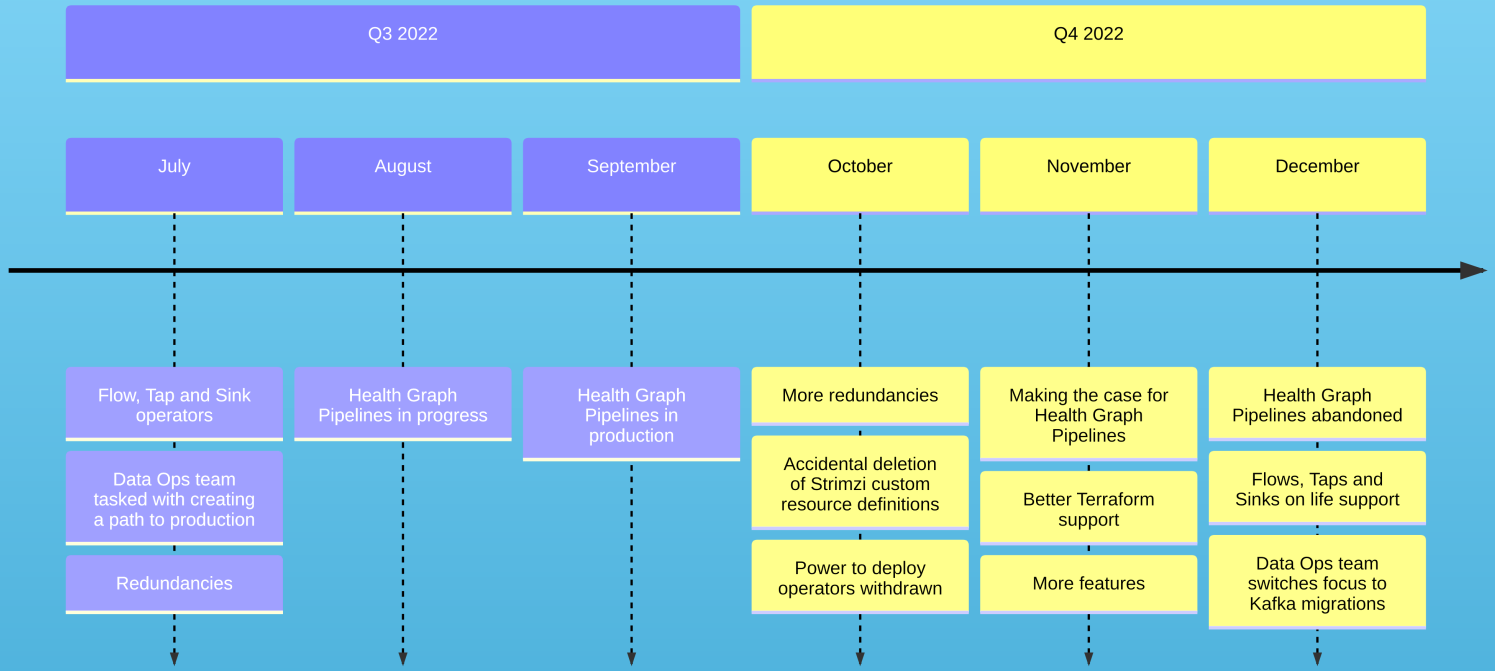
Brace yourselves.

4:19 PM · Nov 7, 2022

Classic Sridharan:
Testing Microservices, the sane way (2017) | Testing in Production, the safe way (2018) | Testing in Production: the hard parts (2019)



Source: layoffs.fyi



Q3 2022

July

Flow, Tap and Sink operators

Data Ops team tasked with creating a path to production

Redundancies

August

Health Graph Pipelines in progress

September

Health Graph Pipelines in production

Q4 2022

October

More redundancies

Accidental deletion of Strimzi custom resource definitions

Power to deploy operators withdrawn

November

Making the case for Health Graph Pipelines

Better Terraform support

More features

December

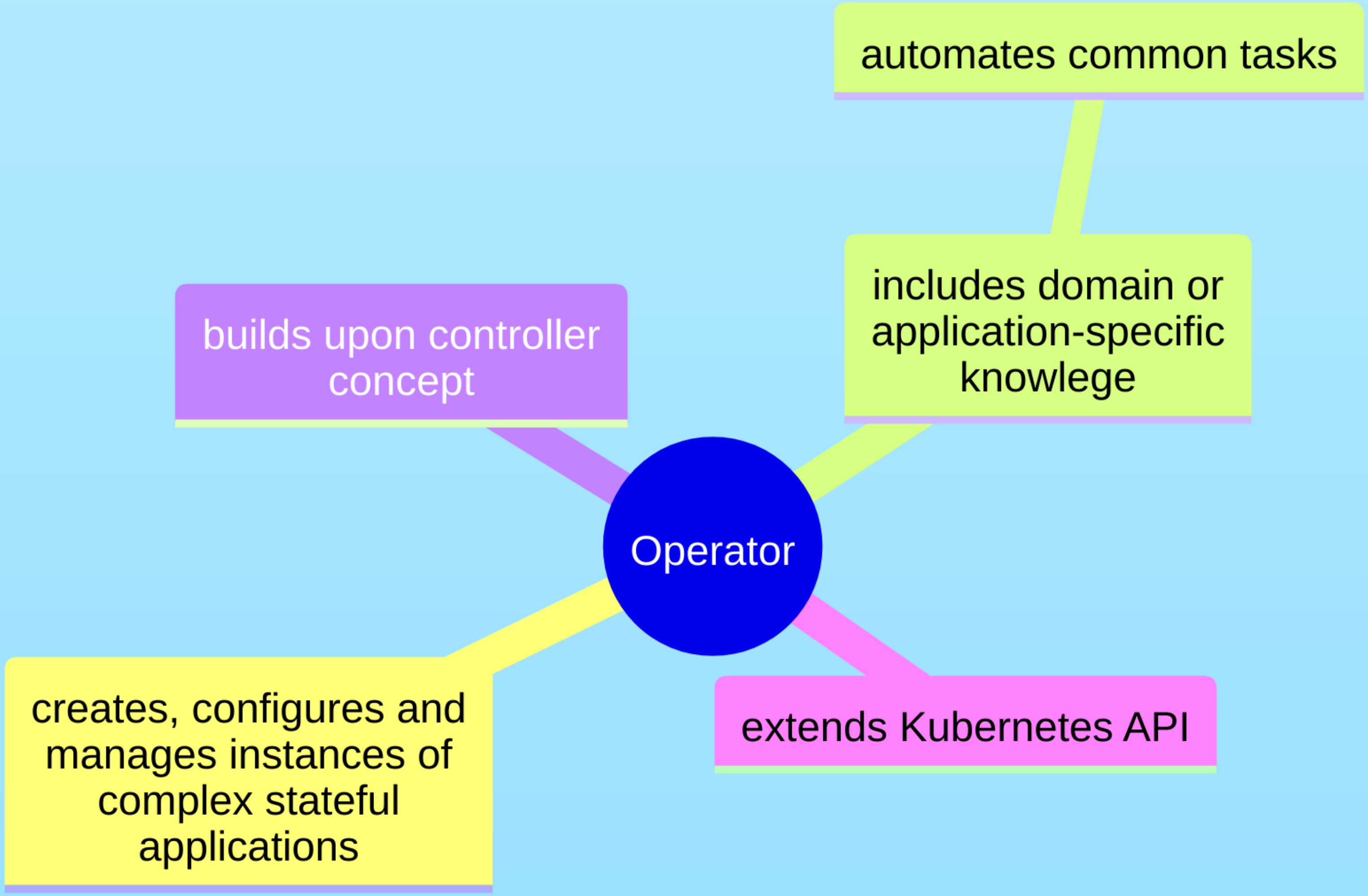
Health Graph Pipelines abandoned

Flows, Taps and Sinks on life support

Data Ops team switches focus to Kafka migrations

> An Operator is an application-specific controller that extends the Kubernetes API to create, configure, and manage instances of complex stateful applications on behalf of a Kubernetes user. It builds upon the basic Kubernetes resource and controller concepts but includes domain or application-specific knowledge to automate common tasks.

Brandon Philips, Introducing Operators: Putting Operational Knowledge into Software (2016)



Operator

builds upon controller concept

includes domain or application-specific knowlege

automates common tasks

extends Kubernetes API

creates, configures and manages instances of complex stateful applications

Great expectations ①

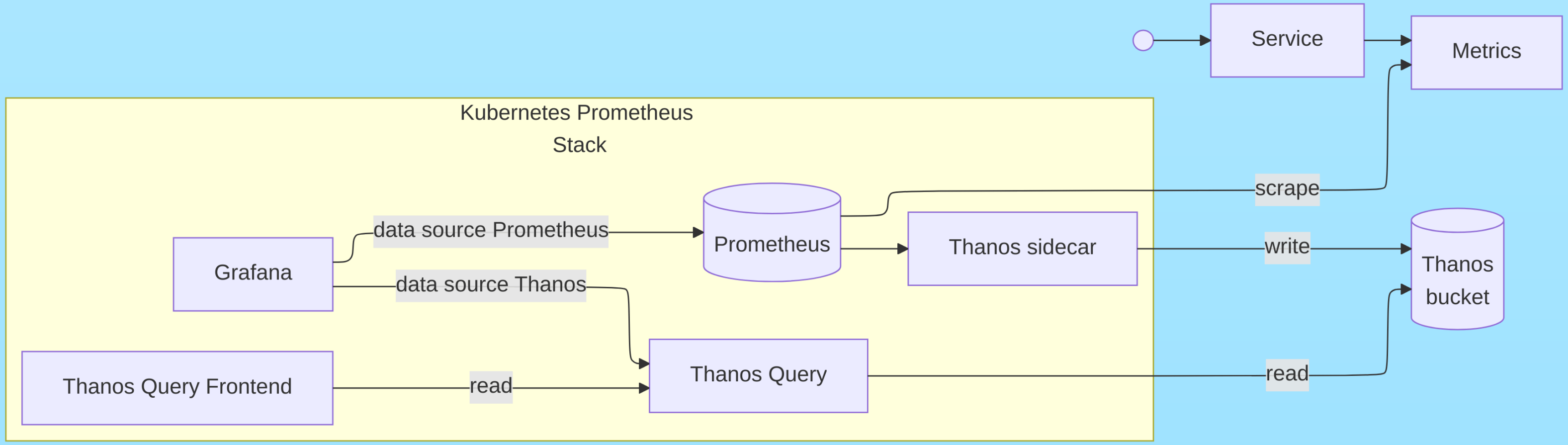
Operators promised to solve the problem of stateful applications on Kubernetes.

One issue is that operators couldn’t and didn’t do any such thing. Calling a resource ‘Database’ and adding a backup task doesn’t magically achieve managed service level availability and durability.

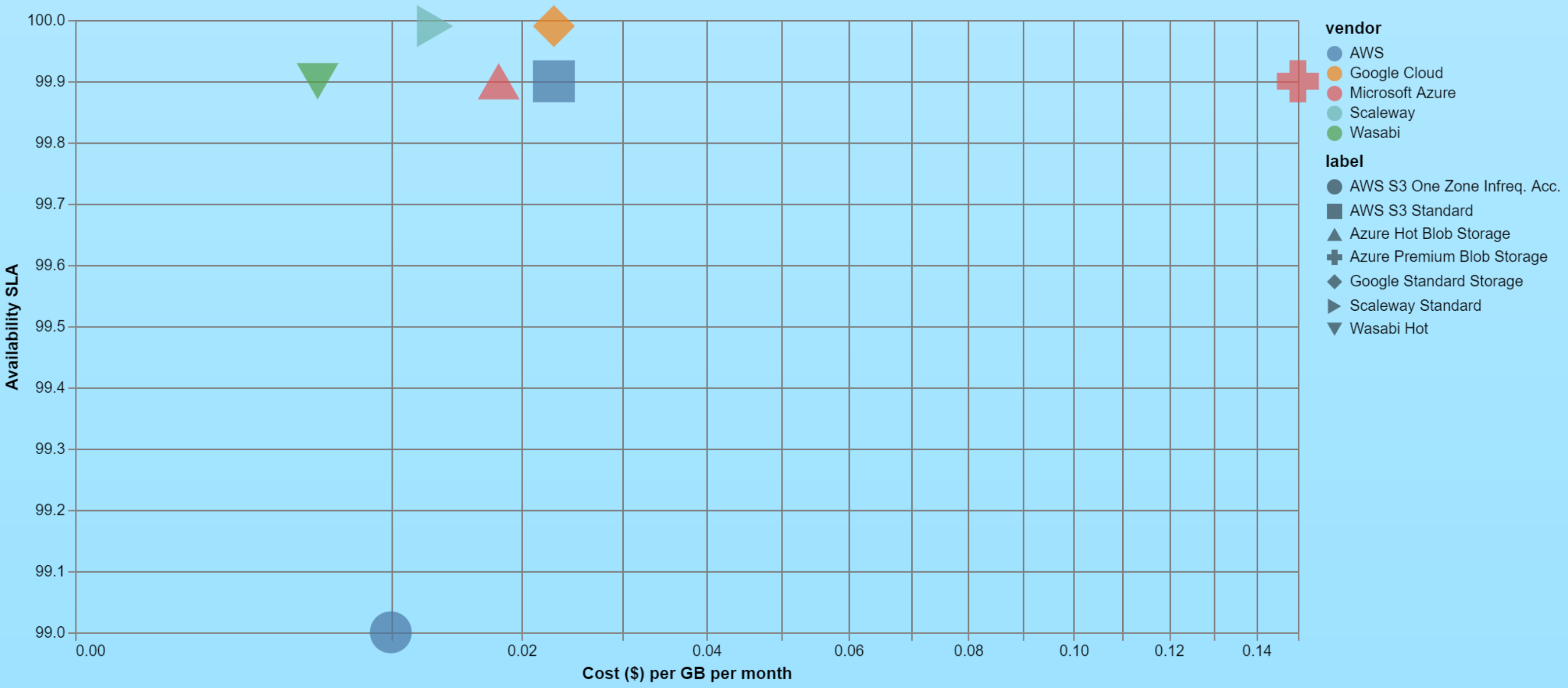
Another is that arguably Kubernetes never had a stateful application problem; it had a persistent volume problem.

See Object storage for stateful applications on Kubernetes (2022).

Enter Thanos and WarpStream stage left and right



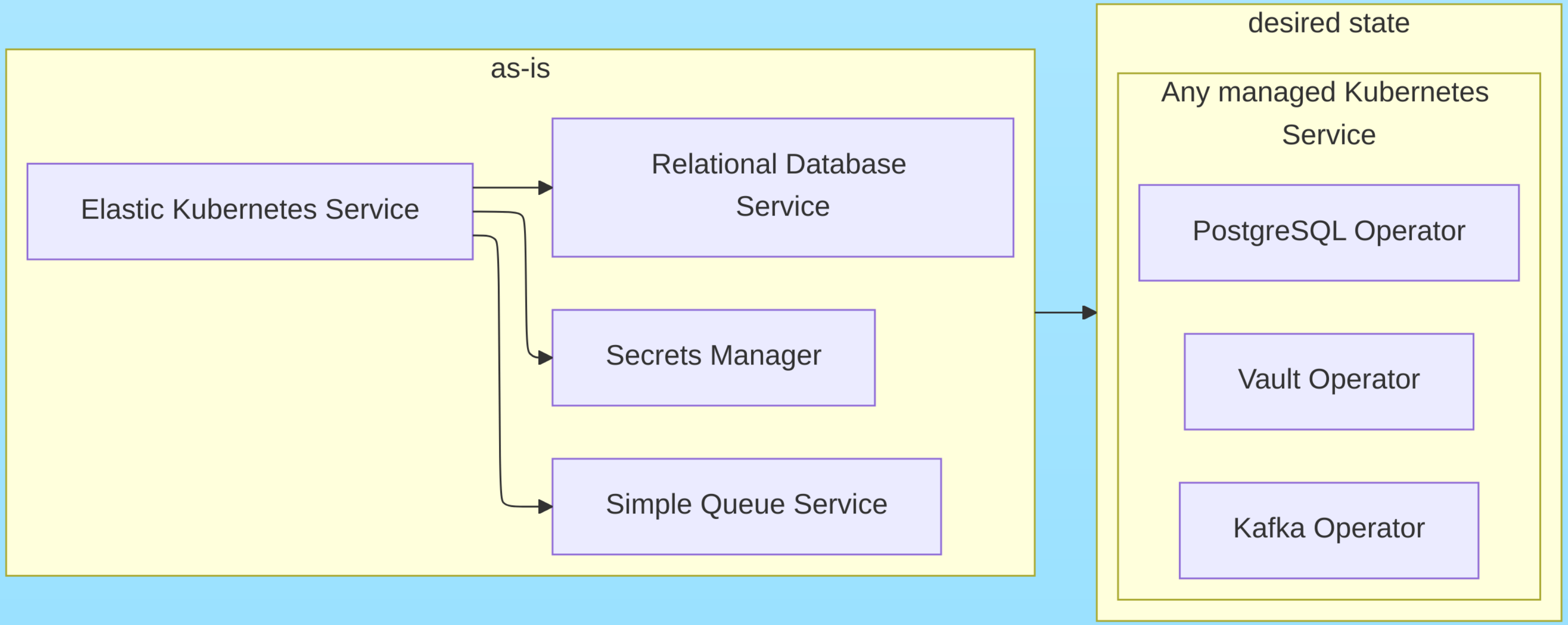
Waiting for COSI



container-object-storage-interface.github.io

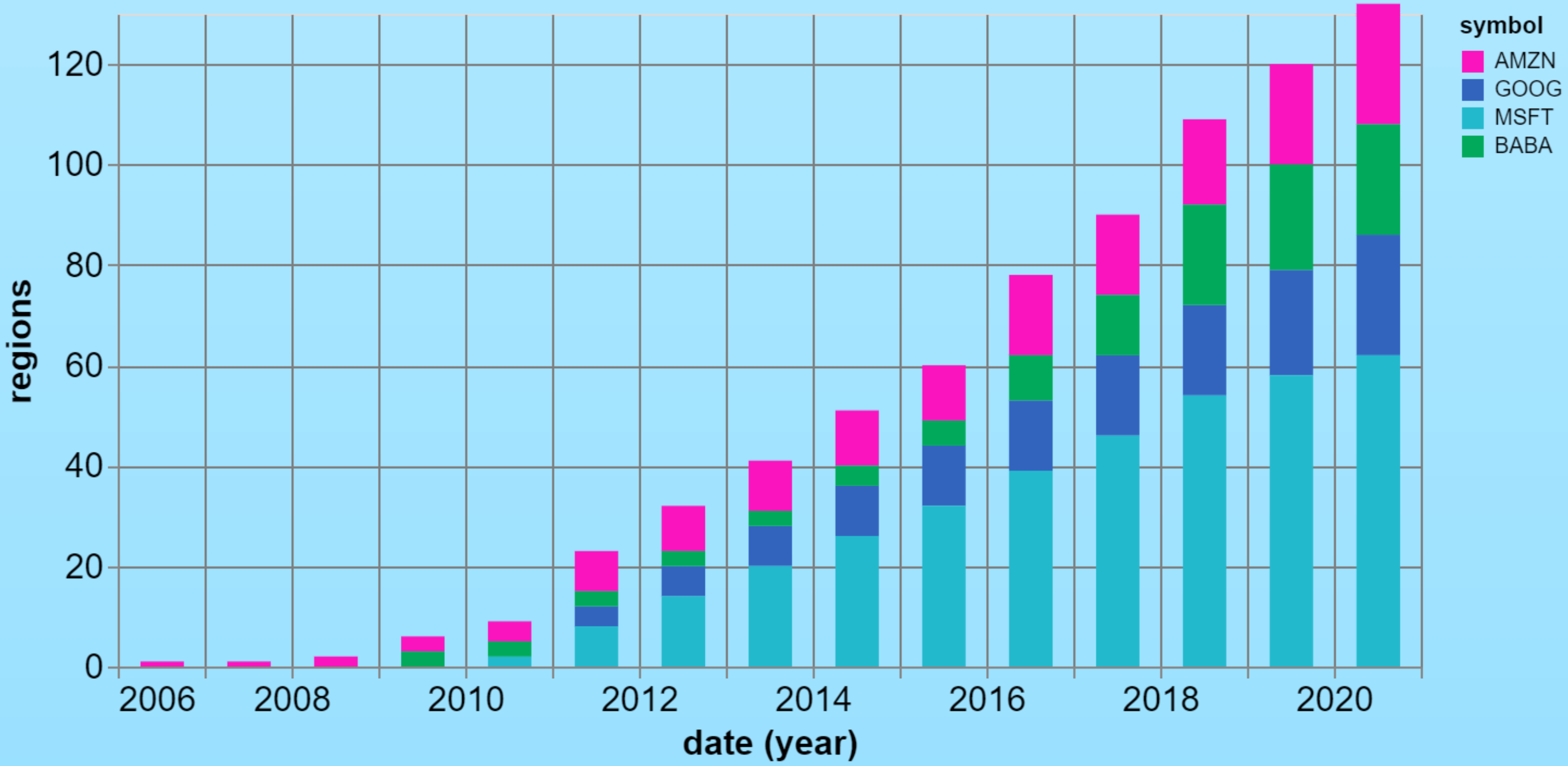
Great expectations ②

Many of us imagined that operators would help us move from managed and mostly proprietary services to portable Kubernetes environments teeming with open source operators.



There was another group cheering us on

That group is best described as everyone except Amazon.

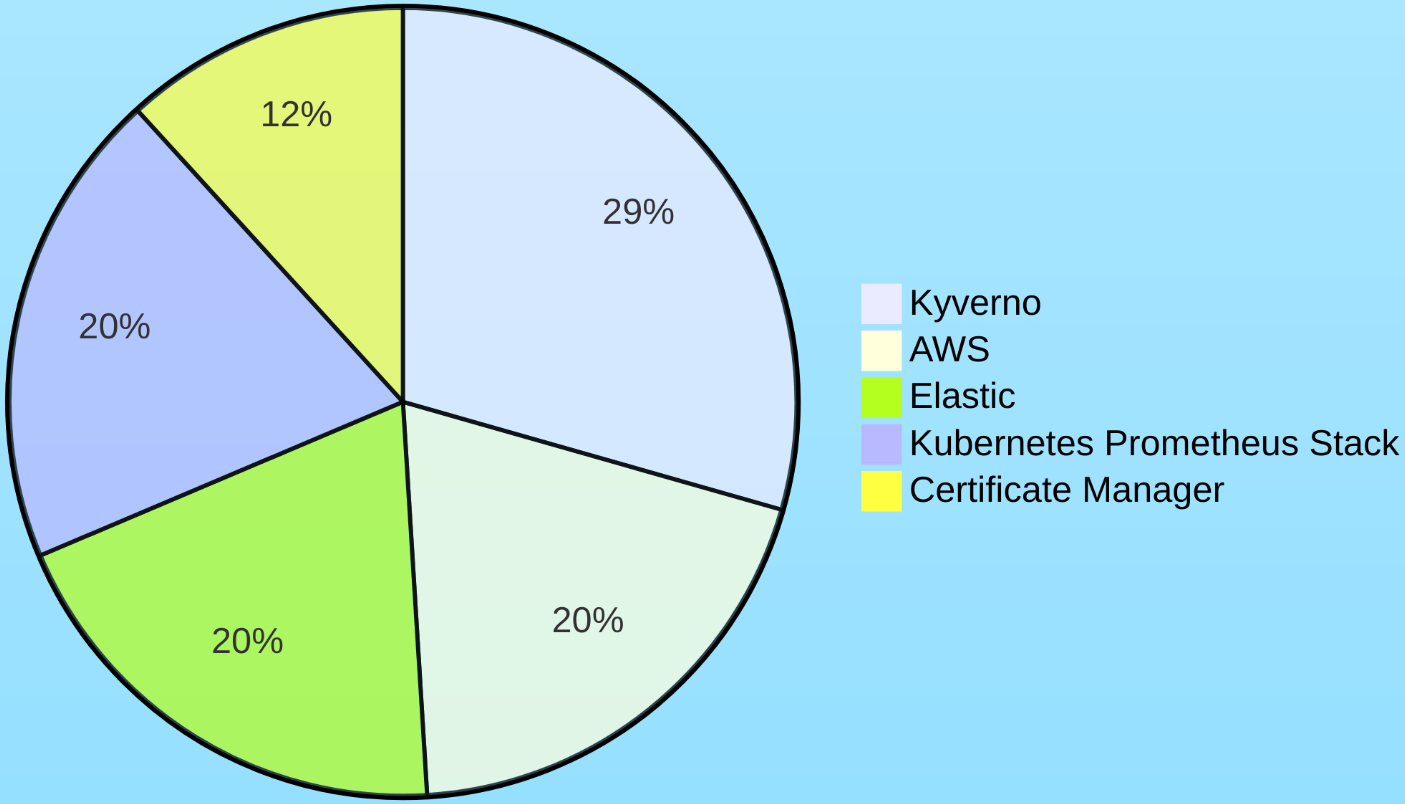


What a three-year head start buys you

Alexa for Business, Amazon AppFlow, Amazon Augmented AI, Amazon Braket, Amazon Chime, Amazon CodeGuru, Amazon Comprehend, Amazon Connect, Amazon DocumentDB, Amazon EventBridge, Amazon Forecast, Amazon Fraud Detector, Amazon GameLift, Amazon Honeycode, Amazon Interactive Video Service, Amazon Kendra, Amazon Keyspaces, Amazon Lex, Amazon Macie, Amazon Managed Blockchain, Amazon MQ, Amazon Personalize, Amazon Polly, Amazon QLDB, Amazon Redshift, Amazon Rekognition, Amazon SageMaker, Amazon Sumerian, Amazon Textract, Amazon Transcribe, Amazon Translate, API Gateway, Application Discovery Service, AppStream 2.0, Artifact, Athena, AWS Amplify, AWS App Mesh, AWS AppConfig, AWS AppSync, AWS Auto Scaling, AWS Backup, AWS Budgets, AWS Chatbot, AWS Cloud Map, AWS Compute Optimizer, AWS Cost Explorer, AWS Data Exchange, AWS DeepComposer, AWS DeepLens, AWS DeepRacer, AWS Firewall Manager, AWS Glue, AWS IQ, AWS Lake Formation, AWS License Manager, AWS Marketplace Subscriptions, AWS Migration Hub, AWS Organizations, AWS Outposts, AWS RoboMaker, AWS Single Sign-On, AWS Snow Family, AWS Transfer Family, AWS Well-Architected Tool, Batch, Certificate Manager, Cloud9, CloudFormation, CloudFront, CloudHSM, CloudSearch, CloudTrail, CloudWatch, CodeArtifact, CodeBuild, CodeCommit, CodeDeploy, CodePipeline, CodeStar, Cognito, Config, Control Tower, Data Pipeline, Database Migration Service, DataSync, Detective, Device Farm, Direct Connect, Directory Service, DynamoDB, EC2, EC2 Image Builder, EFS, Elastic Beanstalk, Elastic Container Registry, Elastic Container Service, Elastic Kubernetes Service, Elastic Transcoder, ElastiCache, Elasticsearch Service, Elemental Appliances & Software, EMR, FreeRTOS, FSx, Global Accelerator, Ground Station, GuardDuty, IAM, Inspector, IoT 1-Click, IoT Analytics, IoT Core, IoT Device Defender, IoT Device Management, IoT Events, IoT Greengrass, IoT SiteWise, IoT Things Graph, Key Management Service, Kinesis, Kinesis Video Streams, Lambda, Launch Wizard, Lightsail, Managed Services, MediaConnect, MediaConvert, MediaLive, MediaPackage, MediaStore, MediaTailor, Mobile Hub, MSK, Neptune, OpsWorks, Personal Health Dashboard, Pinpoint, QuickSight, RDS, Resource Access Manager, Route 53, S3, S3 Glacier, Secrets Manager, Security Hub, Server Migration Service, Serverless Application Repository, Service Catalog, Simple Email Service, Simple Notification Service, Simple Queue Service, Step Functions, Storage Gateway, Support, SWF, Systems Manager, Trusted Advisor, VPC, WAF & Shield, WorkDocs, WorkLink, WorkMail, WorkSpaces, X-Ray

Operators didn’t end up playing the role we had envisaged

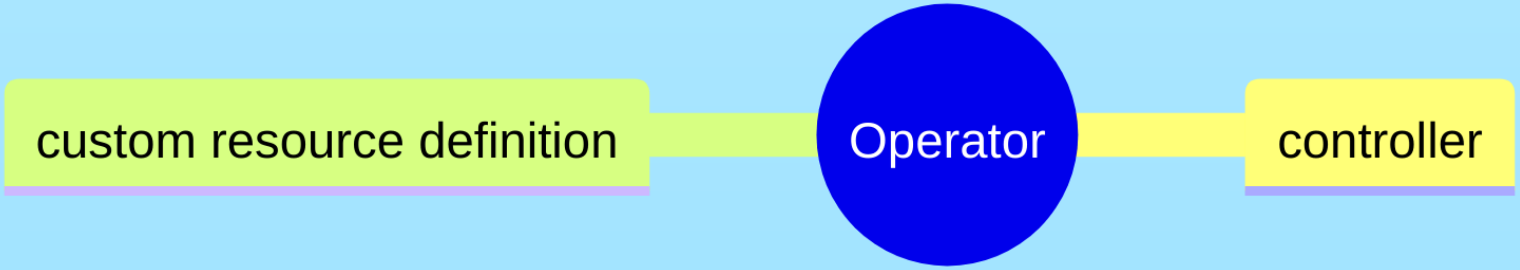
Operators were created for data-intensive applications such as Etcd and PostgreSQL. But with the honourable exception of Prometheus, these aren’t the operators that are running in most clusters today.



None of this mattered, why?

Amazon seemingly unassailable service advantage proved illusory, the moat so deep Amazon found it harder and harder to innovate.

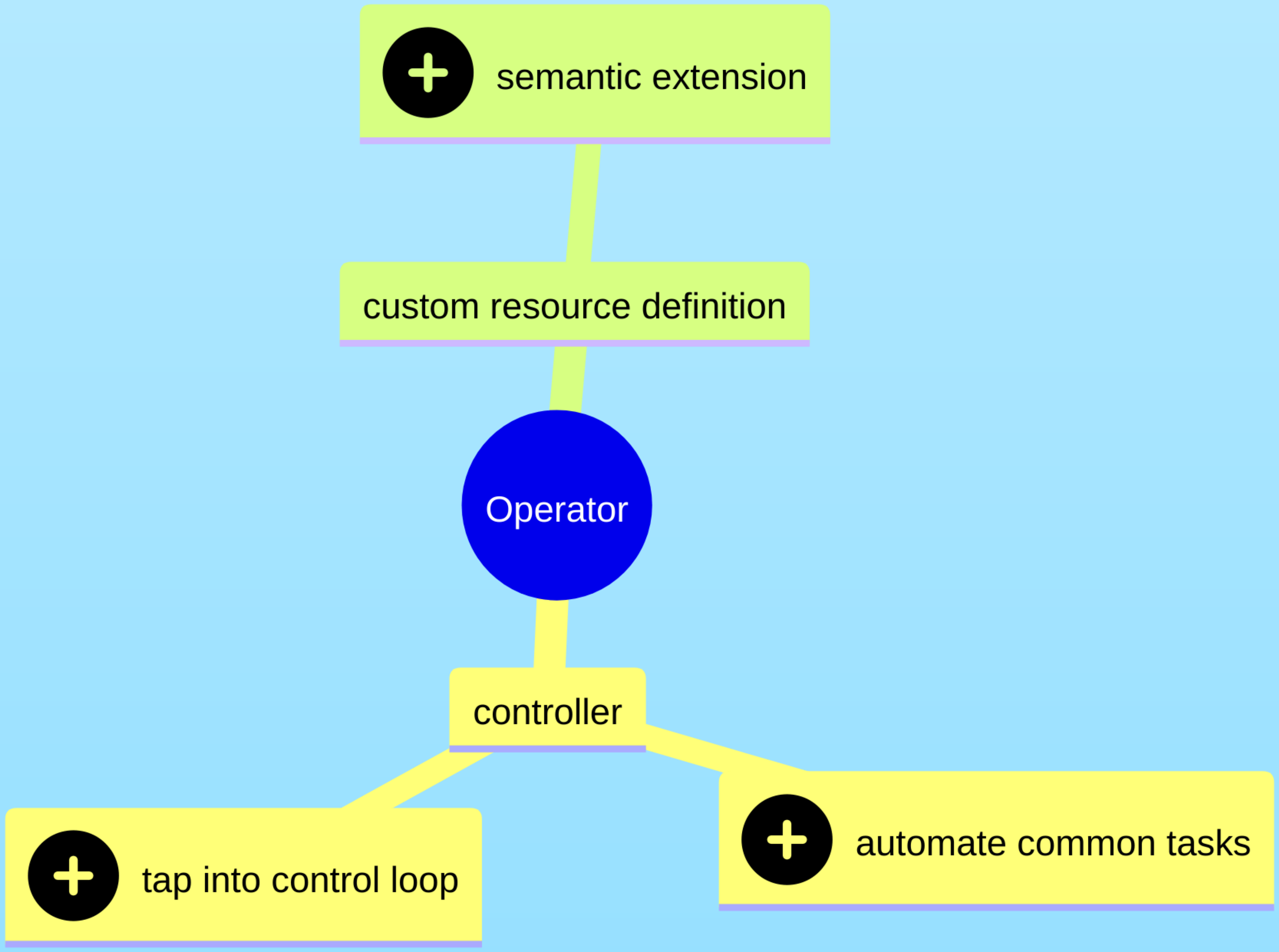
We kept using managed services as before and every self-respecting open source project added custom resource definitions.



custom resource definition

Operator

controller



semantic extension

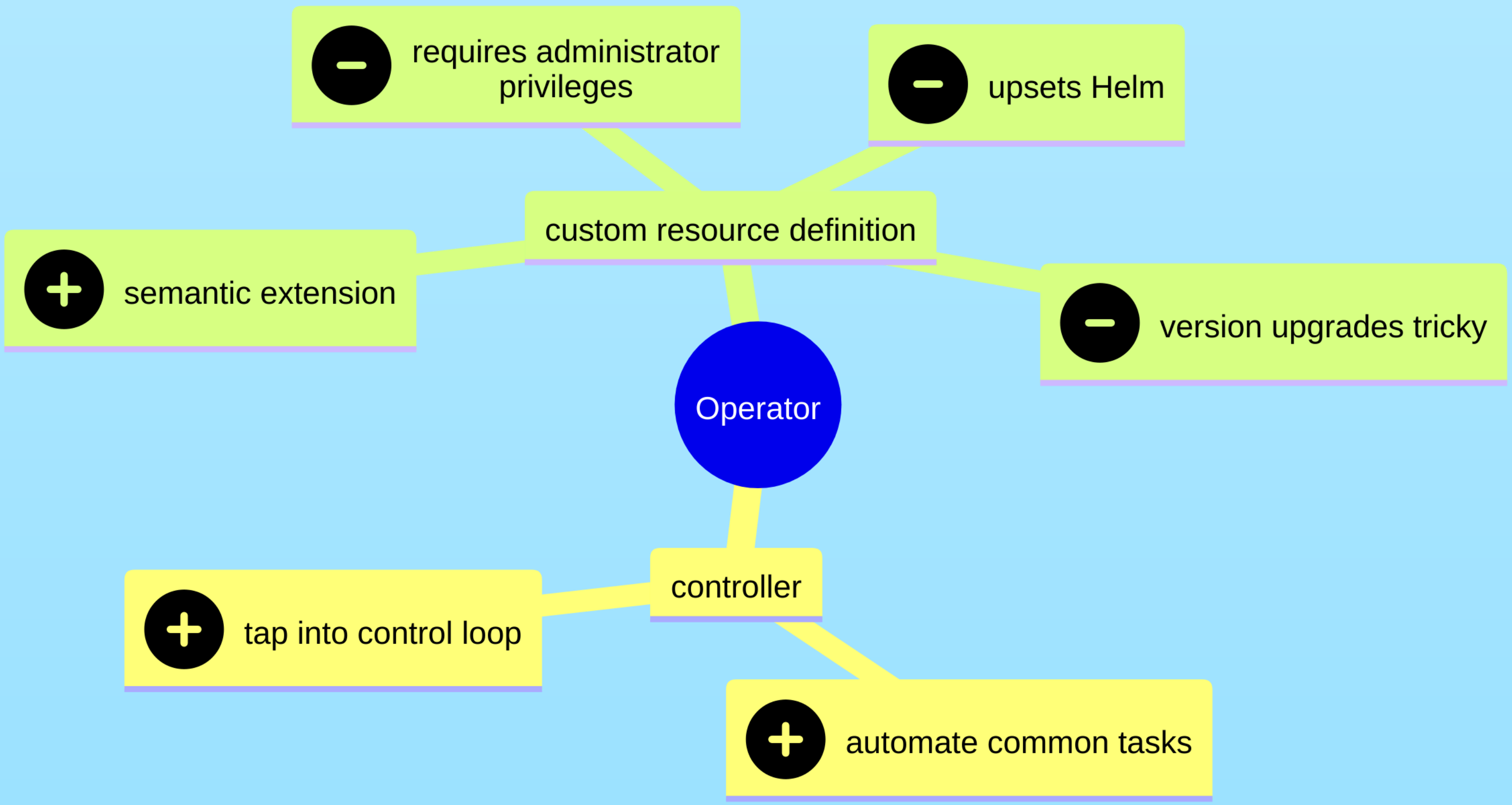
custom resource definition

Operator

controller

tap into control loop

automate common tasks



Operator

custom resource definition

− requires administrator privileges

− upsets Helm

+ semantic extension

− version upgrades tricky

controller

+ tap into control loop

+ automate common tasks

Nothing is simple about writing a CRD.

Adam Jacob, Kubernetes is an anti-platform, Ship It (18 October 2024)

What about the user experience?

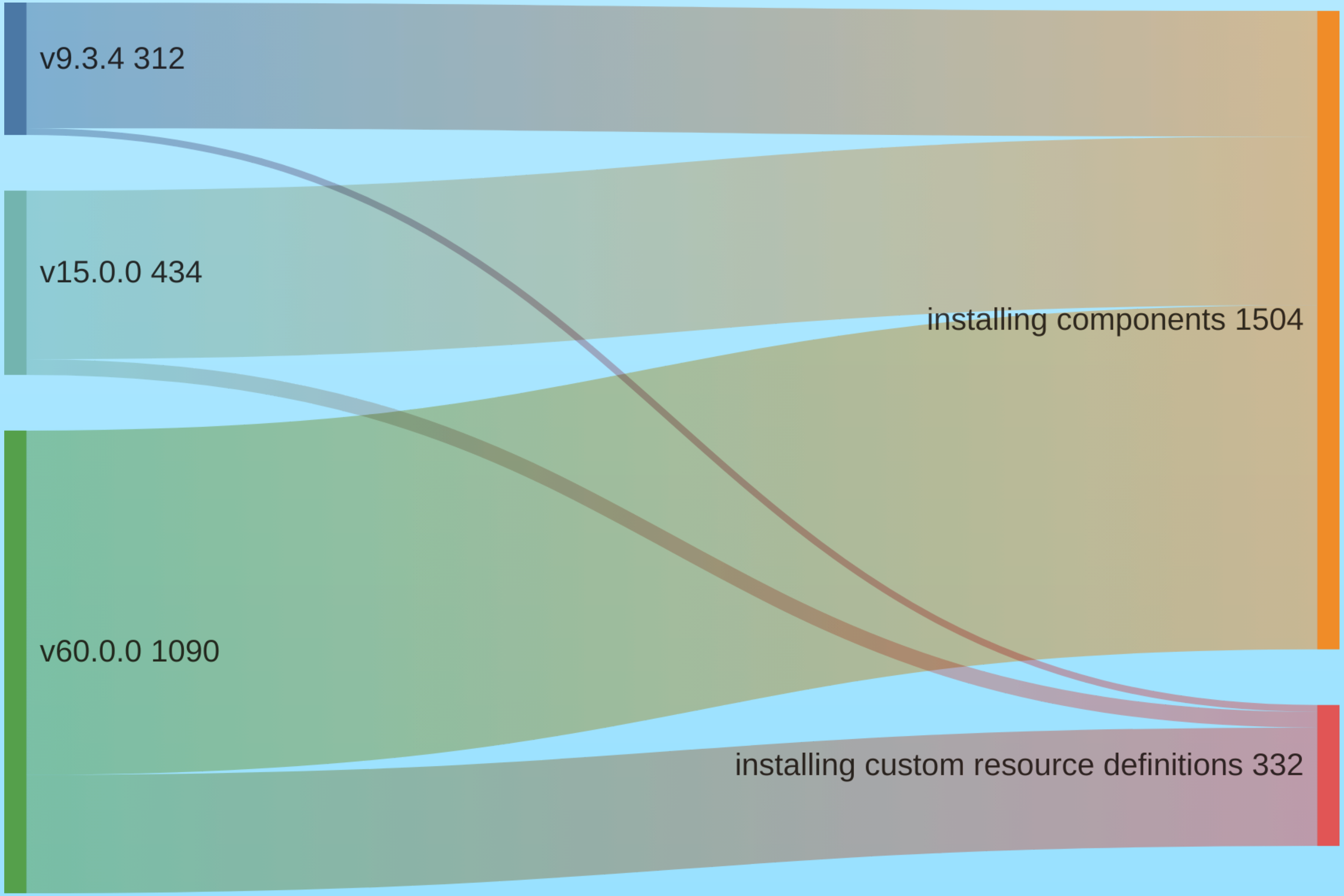
Let’s take a look at the Kubernetes Prometheus Stack’s README.

```
### From 64.x to 65.x

This version upgrades Prometheus-Operator to v0.77.1

Run these commands to update the CRDs before applying the upgrade.

kubectl apply ... -f https://.../alertmanagerconfigs.yaml
kubectl apply ... -f https://.../alertmanagers.yaml
kubectl apply ... -f https://.../podmonitors.yaml
kubectl apply ... -f https://.../probes.yaml
kubectl apply ... -f https://.../prometheusagents.yaml
kubectl apply ... -f https://.../prometheuses.yaml
kubectl apply ... -f https://.../prometheusrules.yaml
kubectl apply ... -f https://.../scrapeconfigs.yaml
...
```

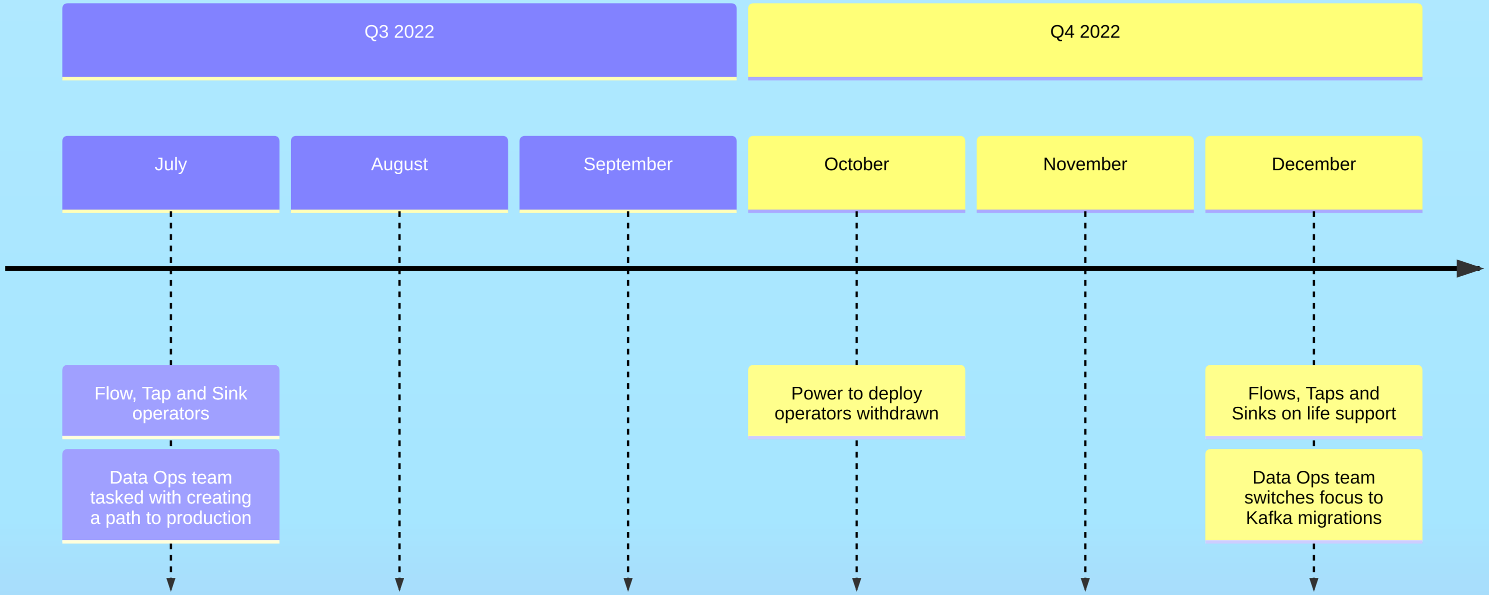

v9.3.4 312
v15.0.0 434
v60.0.0 1090
installing components 1504
installing custom resource definitions 332

Services before ServiceMonitor

```yaml
apiVersion: v1
kind: Service
metadata:
  annotations:
    prometheus.io/port: "2112"
    prometheus.io/scrape: "true"
```

Antipattern ① – operators in developer workflows

Consider flow operator with its Flow, Tap and Sink resources.



Q3 2022

Q4 2022

July

August

September

October

November

December

Flow, Tap and Sink operators

Data Ops team tasked with creating a path to production

Power to deploy operators withdrawn

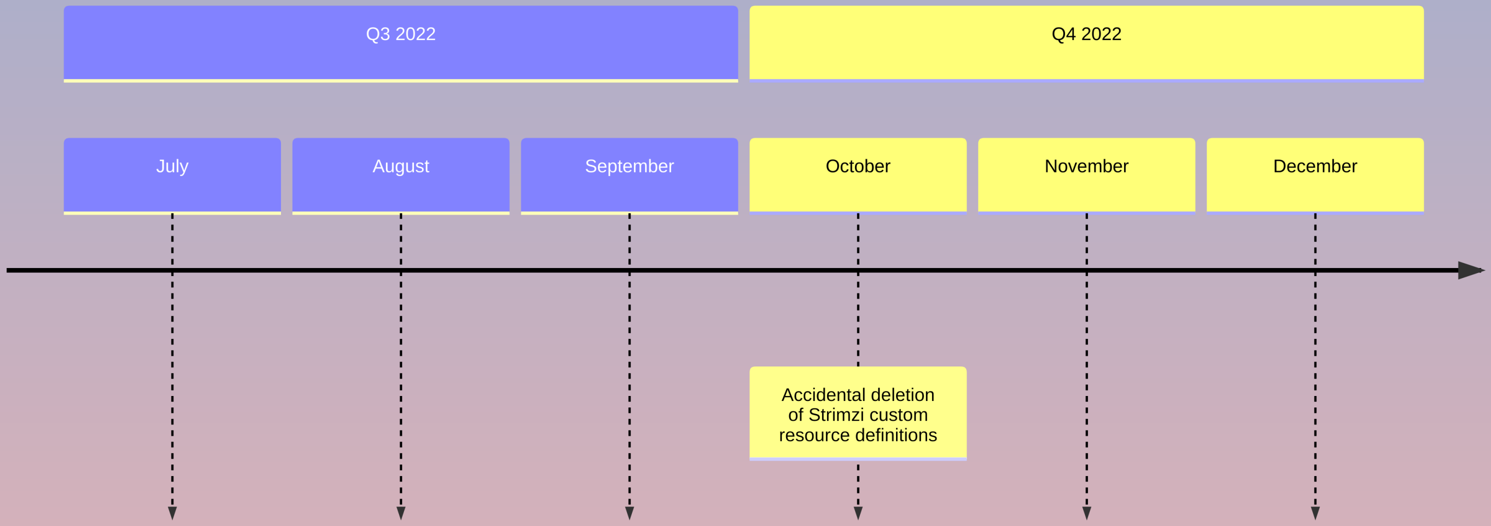
Flows, Taps and Sinks on life support

Data Ops team switches focus to Kafka migrations

Antipattern ② – tight coupling with external resources

Consider Strimzi.

Deletion protection is a bandaid but one that takes away the central value proposition, which is to manage the external resource.



Antipattern ③ – versioning trouble

Incrementing CRD versions is a serious matter.

Is the old version still served? Have we provided a conversion webhook?

So far from reducing complexity, we are introducing new error conditions, failure modes and edge cases.

Exhibit A is the excellent AWS Controllers for Kubernetes, still on v1alpha1.

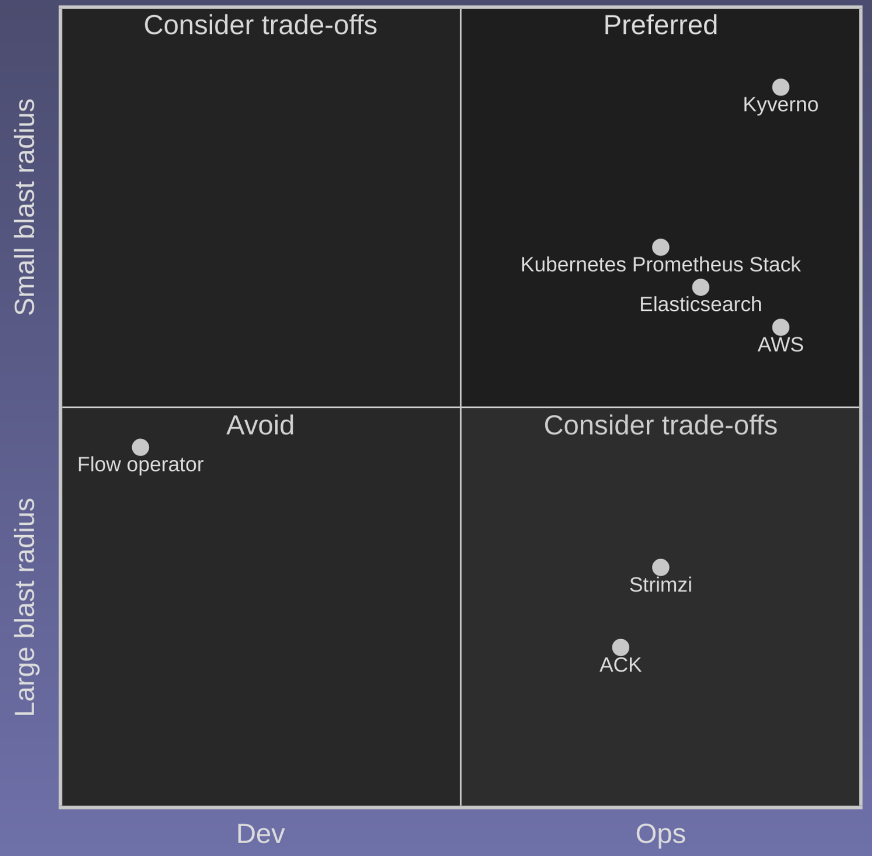
```yaml
apiVersion: s3.services.k8s.aws/v1alpha1
kind: Bucket
```

Which operators get everything right?

In my view Kyverno wins first prize for an implementation that feels as if it should be an in-tree policy engine.

Policy violations create detailed events and the new resources (Policy, ClusterPolicy) fit well into the existing set of resources.

```
$ kubectl get events --sort-by='{.lastTimestamp}'
TYPE    REASON          MESSAGE
Warning PolicyViolation policy require-ro-rootfs/validate-
                        readOnlyRootFilesystem fail: validation
                        error: Root filesystem must be read-only.
                        rule validate-readOnlyRootFilesystem failed at
                        path /spec/template/spec/containers/0/
                        securityContext/readOnlyRootFilesystem/
```


Consider trade-offs
Preferred
Kyverno
Kubernetes Prometheus Stack
Elasticsearch
AWS
Avoid
Flow operator
Consider trade-offs
Strimzi
ACK
Small blast radius
Large blast radius
Dev
Ops

The controller revival is overdue

Grafana has bucked the trend. To load a dashboard on startup, Grafana seeks out ConfigMaps that have label `grafana_dashboard` set to value `1`.

There is no need for a GrafanaDashboard CRD.

Teams commit dashboards they wish to keep to version control:

```
for DASHBOARD in \
  $(ls kube-prometheus-stack/dashboards/*.json)
do
  CONFIGMAP=$(basename "${DASHBOARD}" | cut -d'.' -f1)
  kubectl create configmap "${CONFIGMAP}" \
    -n monitoring \
    --dry-run=client \
    --from-file="${DASHBOARD}" -o yaml | \
    kubectl apply -f -
  kubectl label configmap "${CONFIGMAP}" \
    -n monitoring \
    --overwrite grafana_dashboard="1"
done
```

Decision time

When you are about to create a custom resource definition, avoid asking yourself:

Would classes and methods improve my application?

Instead, ask yourself:

Should I create a domain-specific language for my application?

If the answer is no or maybe, try a controller paired with JSON stored in ConfigMaps.

gerald1248/operator-antipattern-slides

in www.linkedin.com/in/gerald1248

03spirit