

The operator antipattern

Kubernetes Community Days London 2024

Gerald Schmidt



Cindy Sridharan

@copyconstruct · [Follow](#)



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



596



Reply

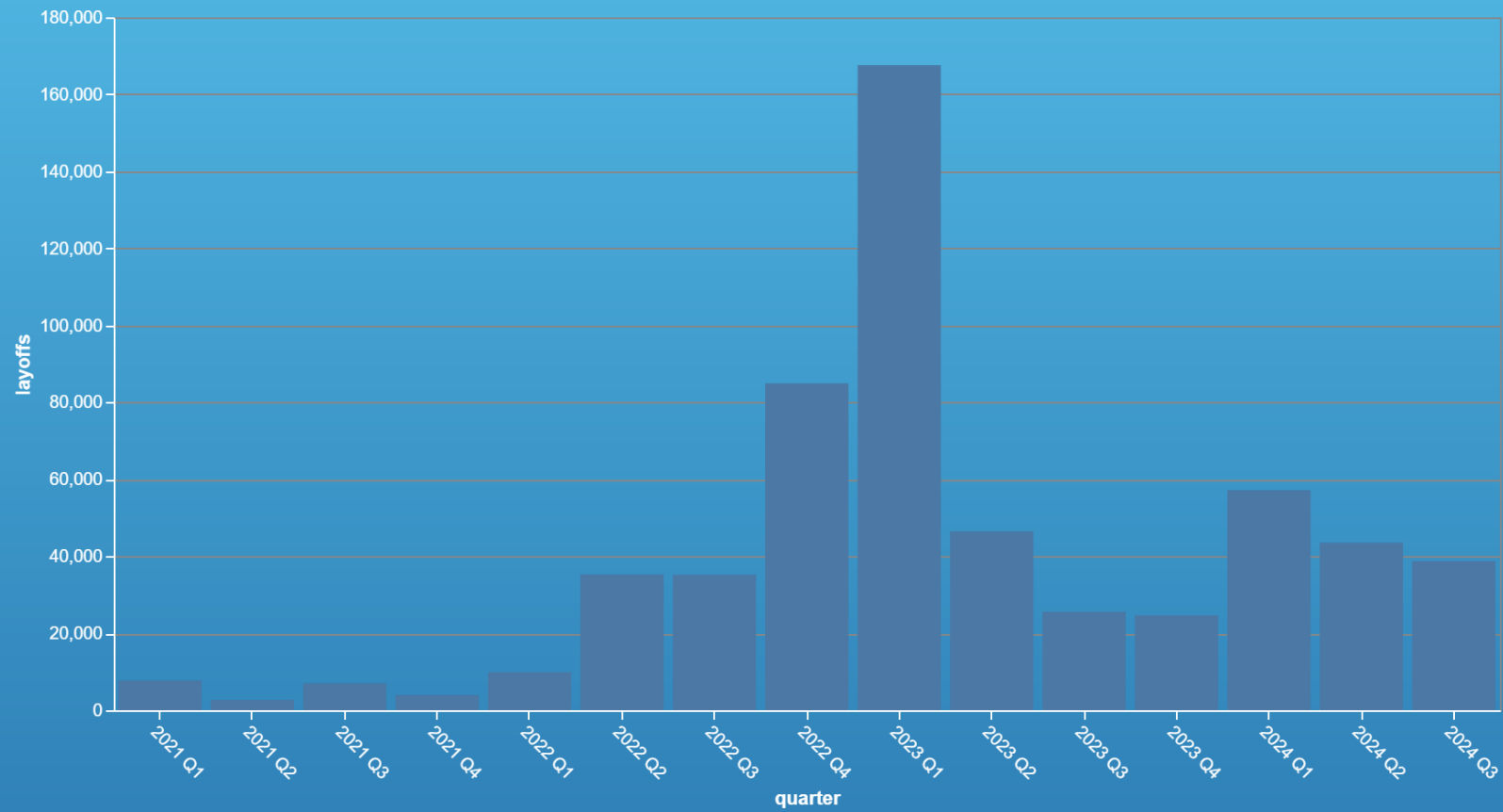


Copy link

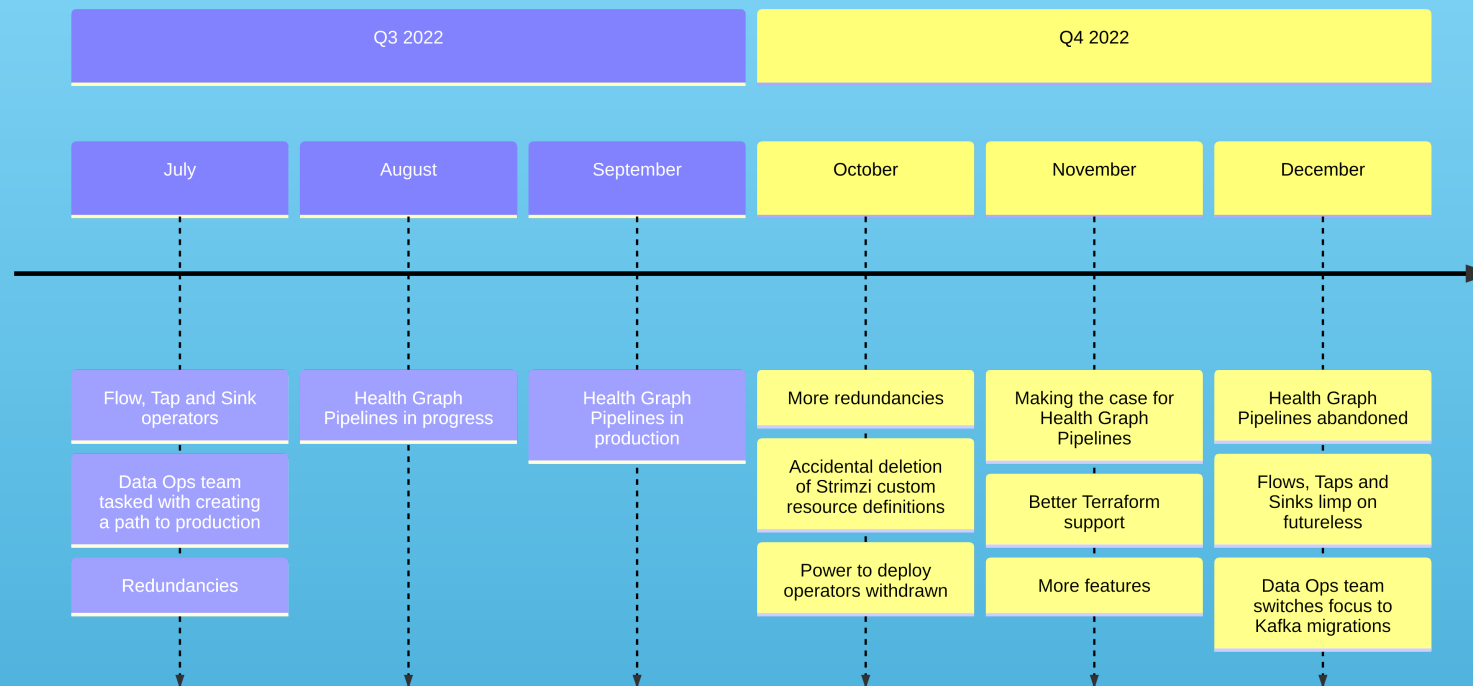
[Read 17 replies](#)

Classic Sridharan:

[Testing Microservices, the sane way \(2017\)](#) | [Testing in Production, the safe way \(2018\)](#) | [Testing in Production: the hard parts \(2019\)](#)



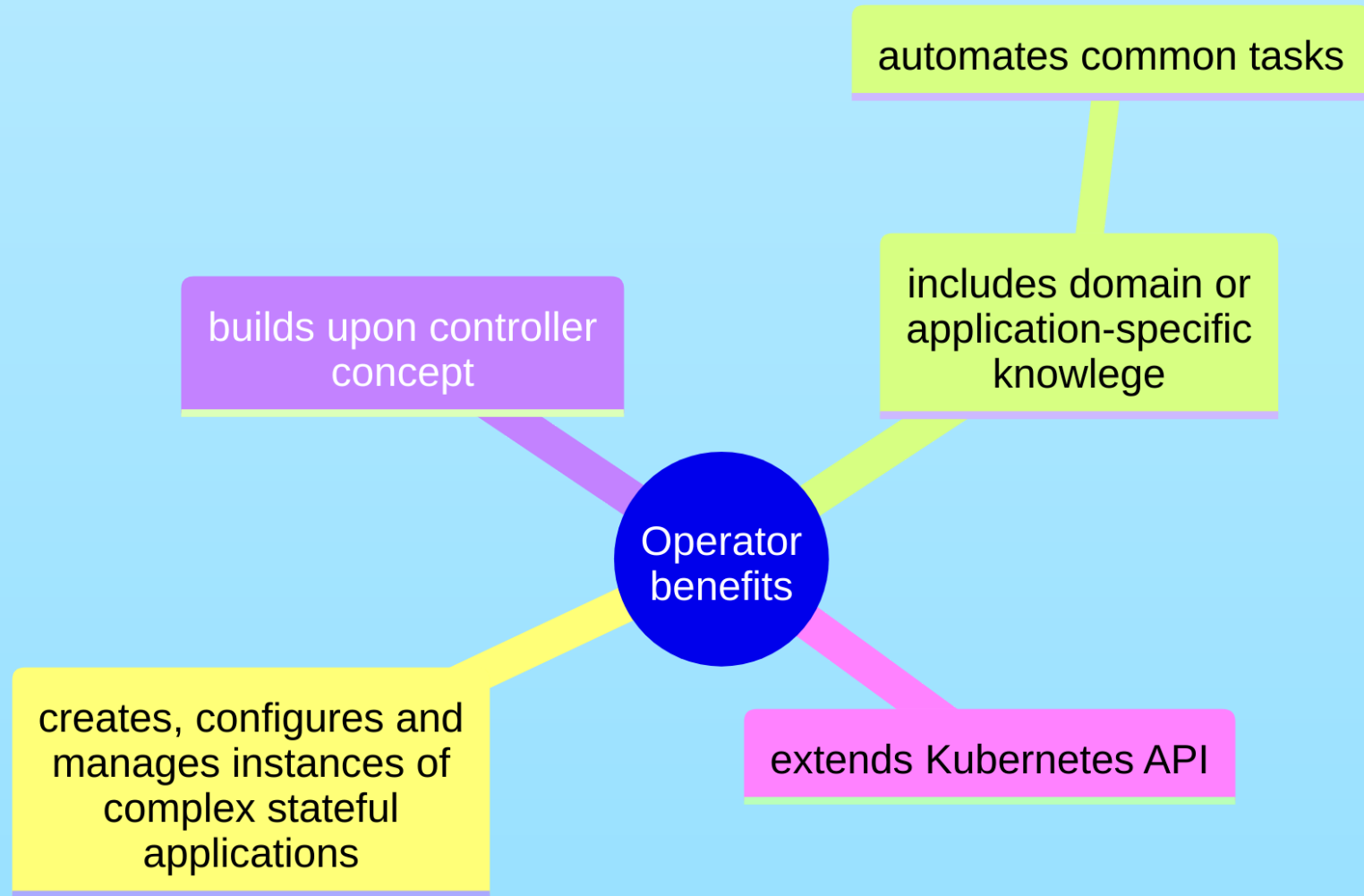
Source: layoffs.fyi



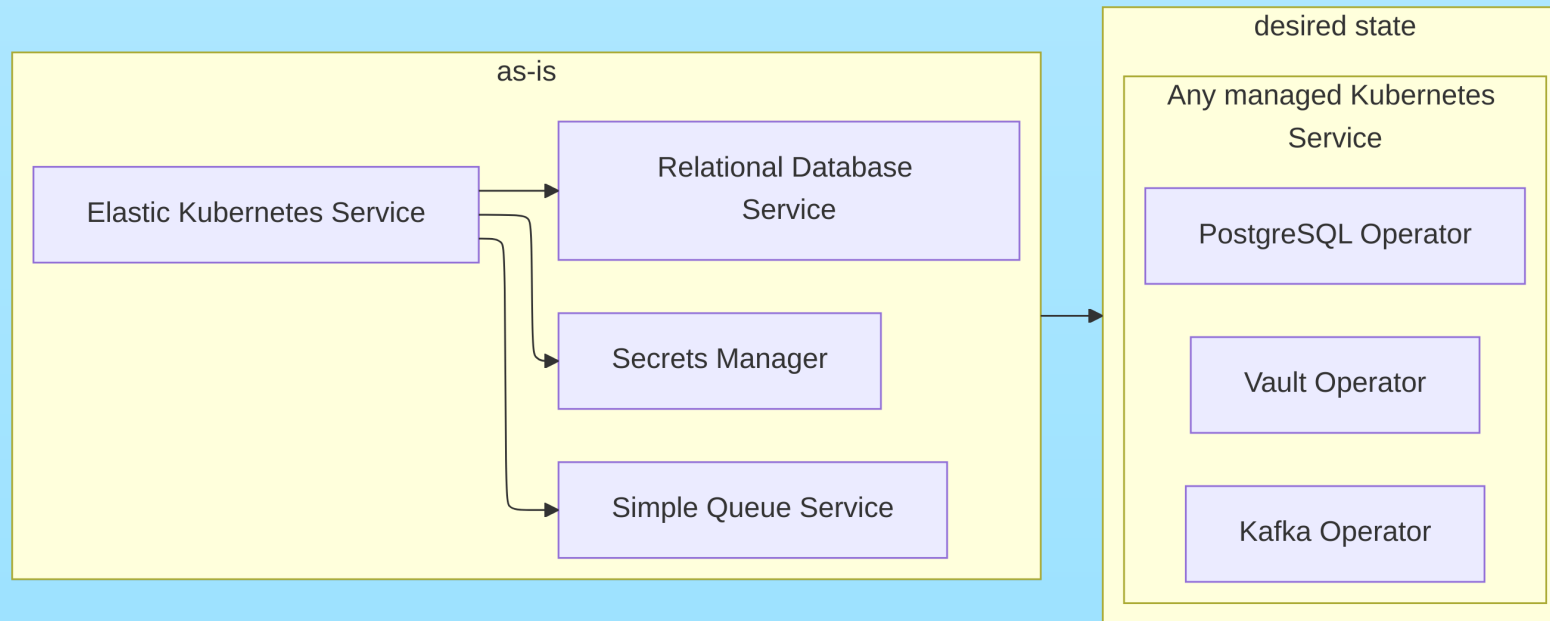
The premise

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.

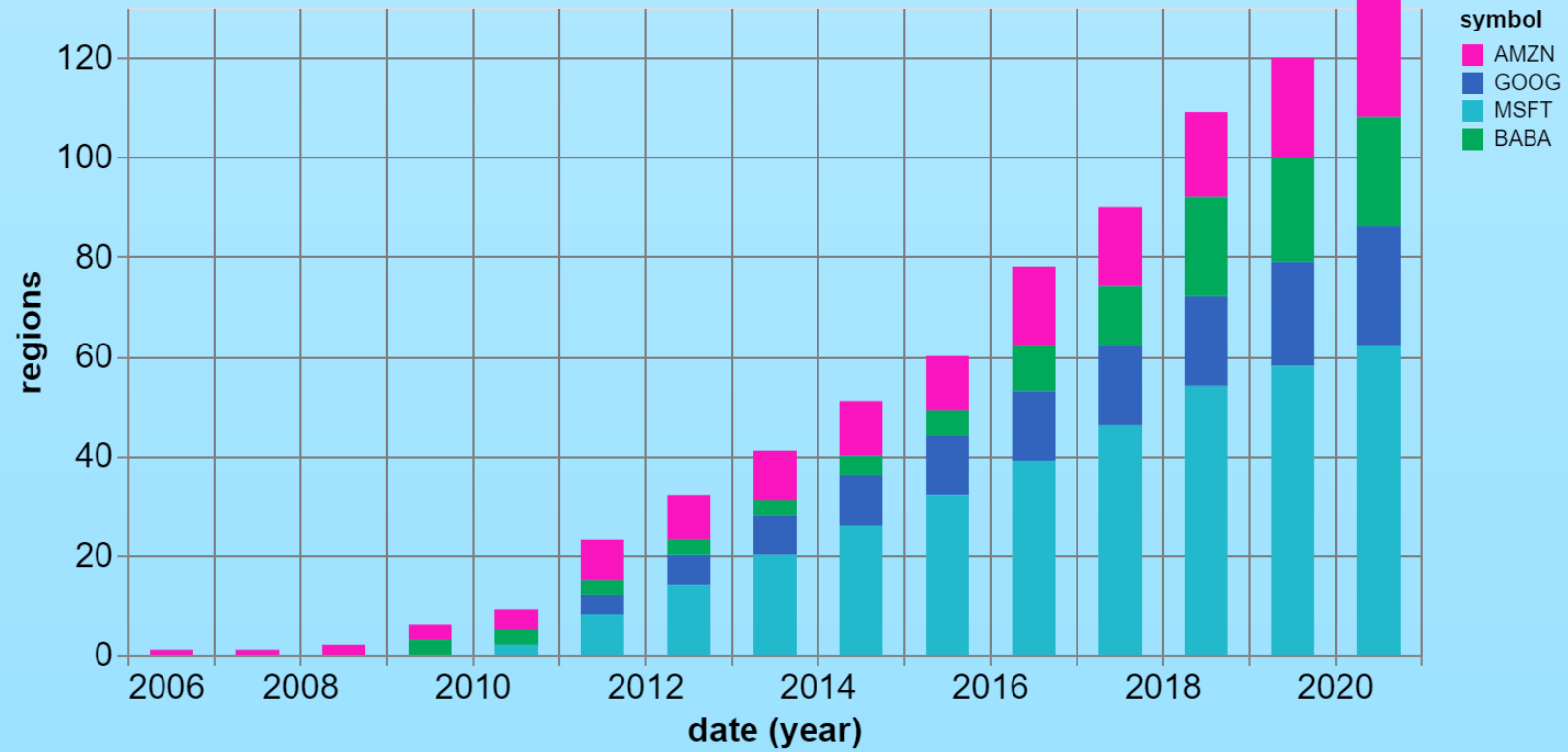
Brandon Philips, [Introducing Operators: Putting Operational Knowledge into Software](#) (2016)



Where we thought we were going



Why did Amazon's competitors cheer us on?

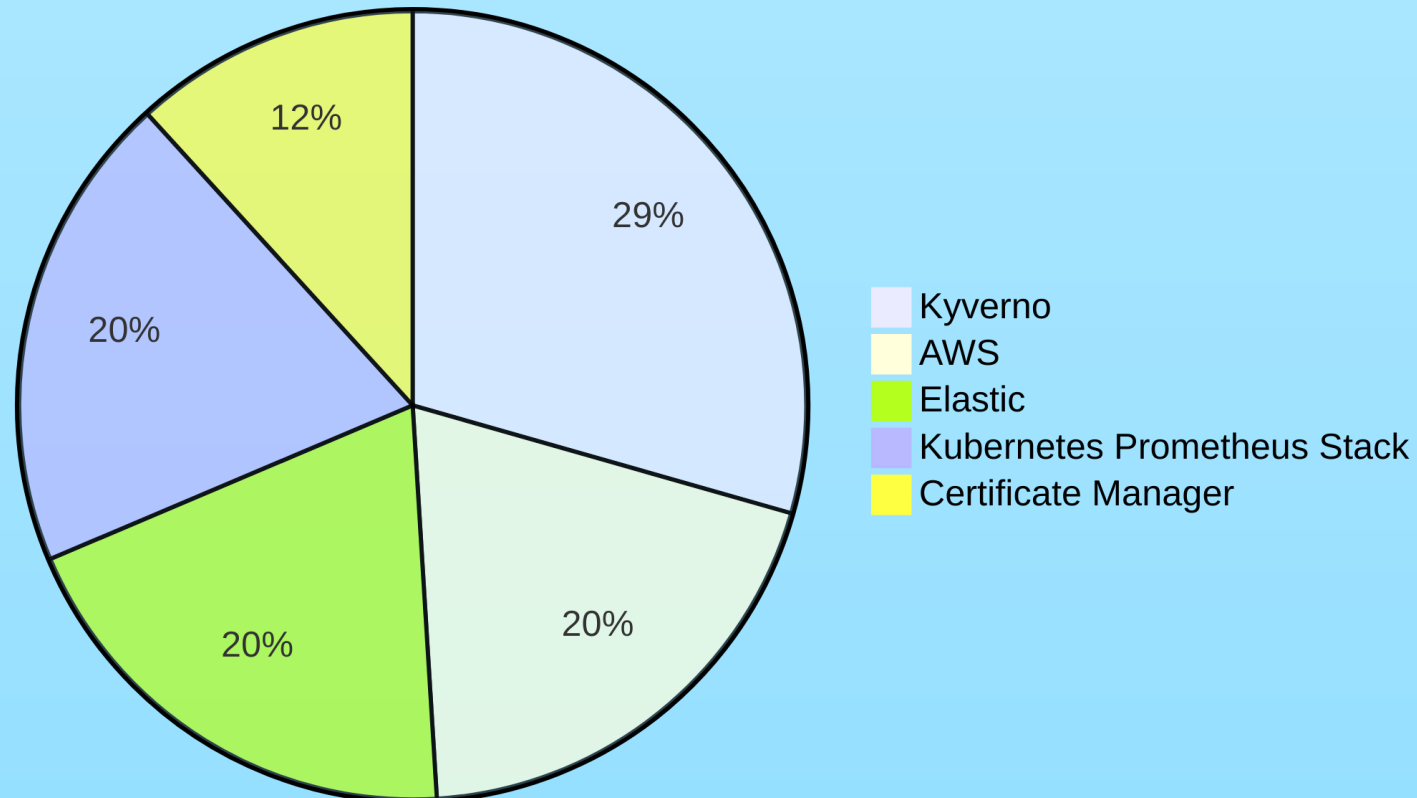


What a four-year head start buys you

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, PostgreSQL and Prometheus. But with the honourable exception of Prometheus, these aren't the operators that are running in most clusters today.



Operators did not deliver on the promise of in-cluster databases that magically reach the resilience of managed database offerings such as Amazon's relational database service, Google's Cloud SQL or Microsoft's Azure SQL Database.

Weighting

The benefits of operators centre on the controller.

The load bearing part is the control loop.

The custom resource component is to a meaningful extent syntactic sugar.

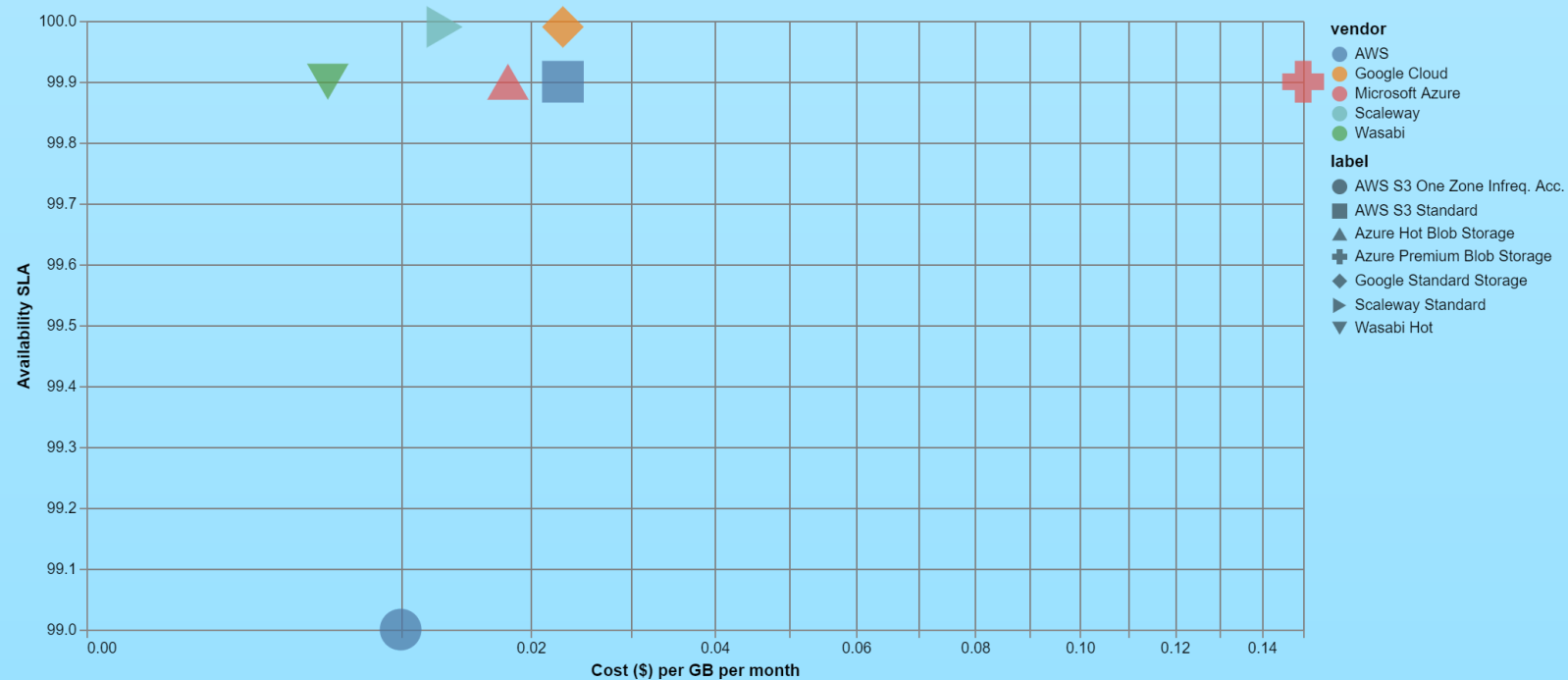
At the same time it is responsible for most of the problematic side-effects of operators: custom resource definitions are cluster-level objects, which brings with it a whole set of permission issues. Versioning is another problematic aspect that we'll come back to.

Reason #1 why the failure of operators didn't matter

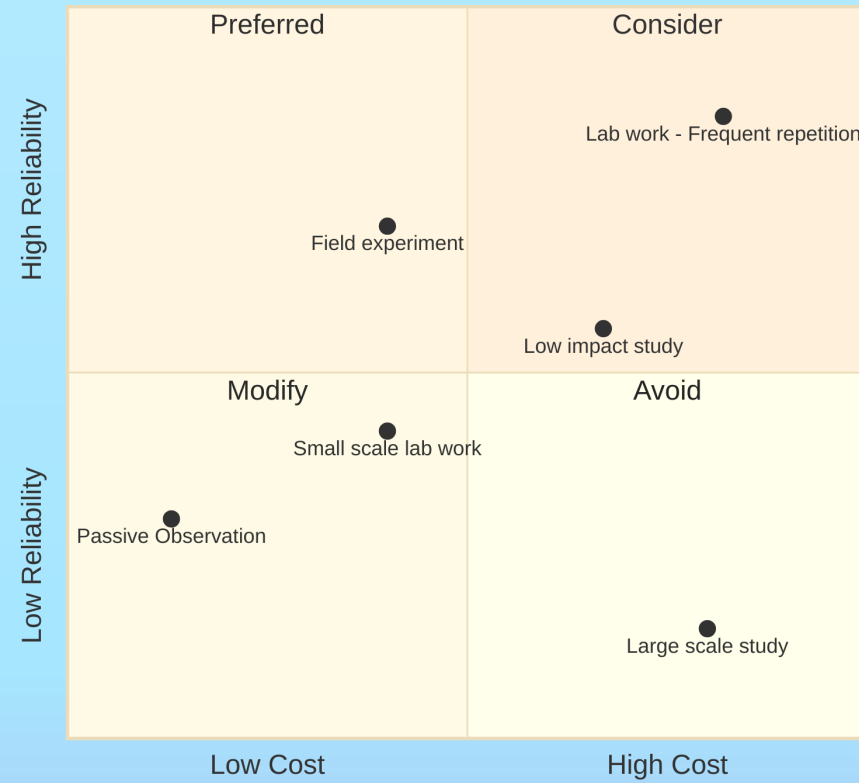
Amazon discovered to their cost that they now had too many services to look after.
The moat had become so deep it became very hard to grow dynamically.

Reason #2 why the failure of operators didn't matter

It dawned on us as practitioners that Kubernetes never had a stateful application problem; it had a persistent volume problem. One by one, applications are switching from block to object storage. Everybody already supports that.



Cost and Results of experiments



Prometheus without operator

```
1 apiVersion: v1
2 kind: Service
3 metadata:
4   annotations:
5     prometheus.io/port: "2112"
6     prometheus.io/scrape: "true"
```

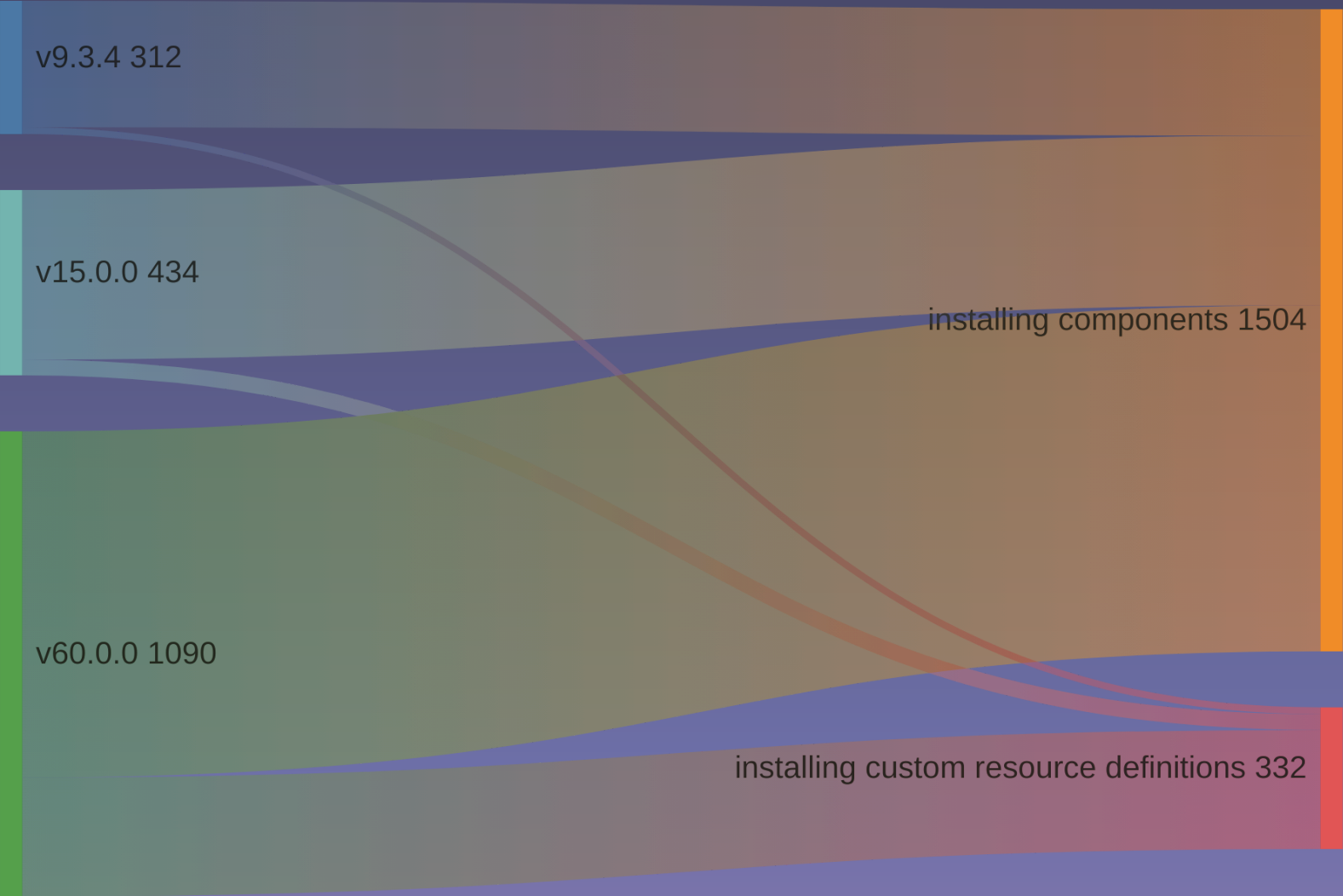

Prometheus with operator

```
1 $ kubectl get crd -o custom-columns=NAME:.metadata.name \  
2   | grep "monitoring\.coreos\  
3 alertmanagerconfigs.monitoring.coreos.com  
4 alertmanagers.monitoring.coreos.com  
5 podmonitors.monitoring.coreos.com  
6 probes.monitoring.coreos.com  
7 prometheusagents.monitoring.coreos.com  
8 prometheuses.monitoring.coreos.com  
9 prometheusrules.monitoring.coreos.com  
10 scrapeconfigs.monitoring.coreos.com  
11 servicemonitors.monitoring.coreos.com  
12 thanosrulers.monitoring.coreos.com
```

Kube Prometheus stack

```
1 FILE=charts/kube-prometheus-stack/README.md
2 for TAG in "9.3.4" "15.0.0" "60.0.0"; do
3     git checkout "kube-prometheus-stack-${TAG}" 2>/dev/null
4     echo "# ${TAG}"
5     head -n -100 "${FILE}" | wc -l charts/kube-prometheus-stack/README.md
6     grep -v "\(^kubect1 .*crd\|CRD\)\" "${FILE}" | wc -l
7 done
```

Kube-prometheus-operator



Antipattern 1: operators in developer workflows

Flow operator

Antipattern 2: tight coupling with external resources

Strimzi

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.

See AWS Controllers for Kubernetes. (Still on alpha.)

Antipattern 4: overpromising

The association of operators with complex *stateful* applications has not displaced managed databases such as the Relational Database Service. The CRD that allows me to create a VectorDatabase resource does not magically make it a good, fault-tolerant. A backup method is helpful and appreciated, but it does not rival a mature point-in-time recovery facility.

Whisper it: Kubernetes does not have a 'stateful workload' problem. It has a persistent volume problem. The solution is object storage, and the challenge is working around the limitations of object storage when it comes to read and write speed.

[See Object storage for stateful applications on Kubernetes](#)

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.

Controller revival

Grafana has bucked the trend of CRD sprawl.




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.

Grafana dashboards are JSON objects following a well-established structure.

Teams store dashboards they wish to keep in a folder:

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

 gerald1248/operator-antipattern-slides
 www.linkedin.com/in/gerald1248
 03spirit