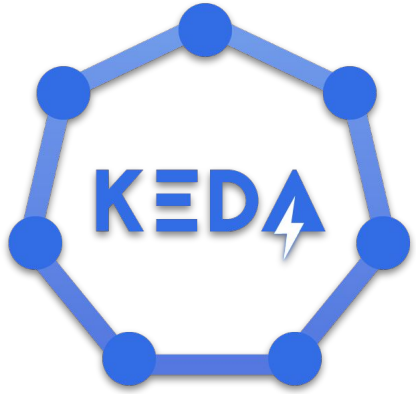
KubeCon CloudNativeCon Europe 2022

Kubernetes Event-driven Autoscaling with KEDA

Jorge Turrado, SRE @ Docplanner Tech
Zbyněk Roubalík, Principal Software Engineer @ Red Hat

Who we are?

- Jorge Turrado
  - SRE @ Docplanner Tech
  - KEDA Maintainer, Microsoft MVP
  - https://github.com/JorTurFer
  - https://twitter.com/JorgeTurrado
  - https://www.linkedin.com/in/jorge-turrado-ferrero-64b70371/
- Zbyněk Roubalík
  - Principal Software Engineer @ Red Hat
  - KEDA Maintainer, Knative (Functions), Microsoft MVP
  - https://github.com/zroubalik
  - https://twitter.com/zroubalik
  - https://www.linkedin.com/in/zbynek-roubalik/



Agenda

1. What is KEDA?
2. KEDA project and community
3. KEDA concepts and architecture
4. Demo!
5. Future development
6. Q&A

What is KEDA?

Kubernetes **E**vent **D**riven **A**utoscaling dead simple !

What is KEDA?

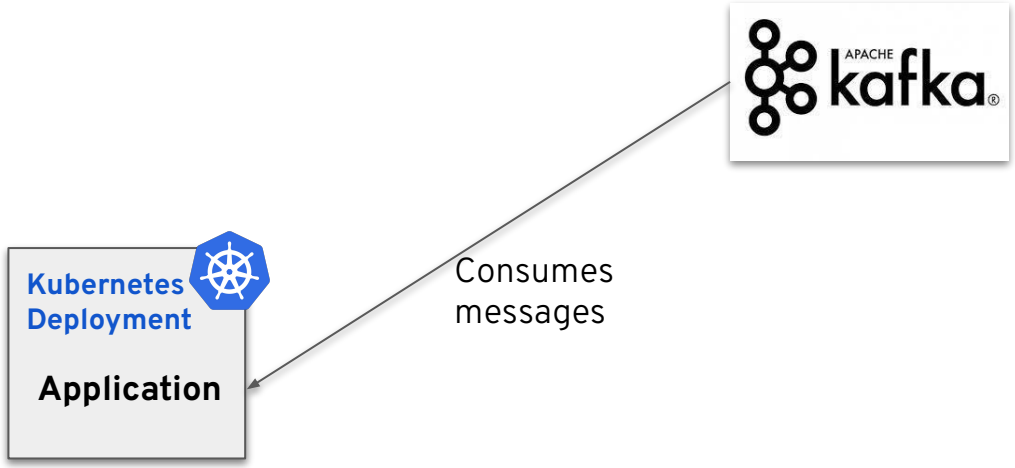
Kubernetes **E**vent **D**riven **A**utoscaling dead simple !

What is KEDA?

Example:

Application consuming messages from Kafka topic

- Application is deployed as standard Kubernetes Deployment
- Can be autoscaled only via standard k8s HPA: CPU & Memory
- No event-driven autoscaling

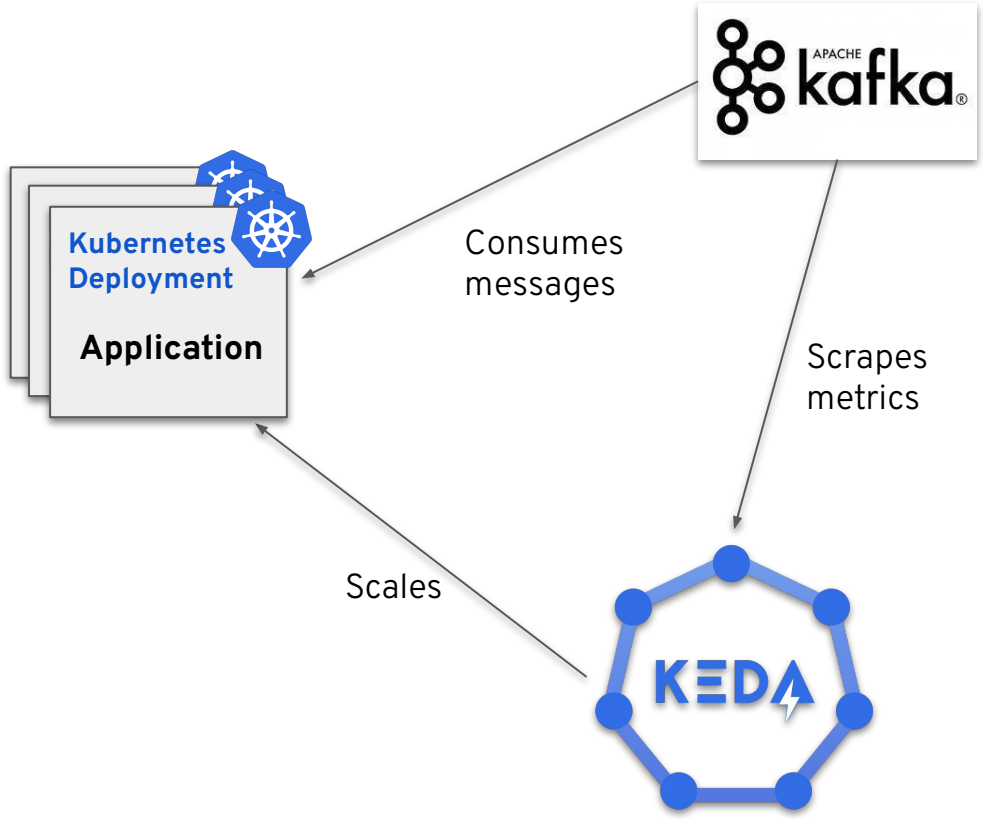


What is KEDA?

Example:

Application redesigned to utilize KEDA

- Application remains the same and is being deployed the same way
- Event-driven autoscaling enabled through KEDA



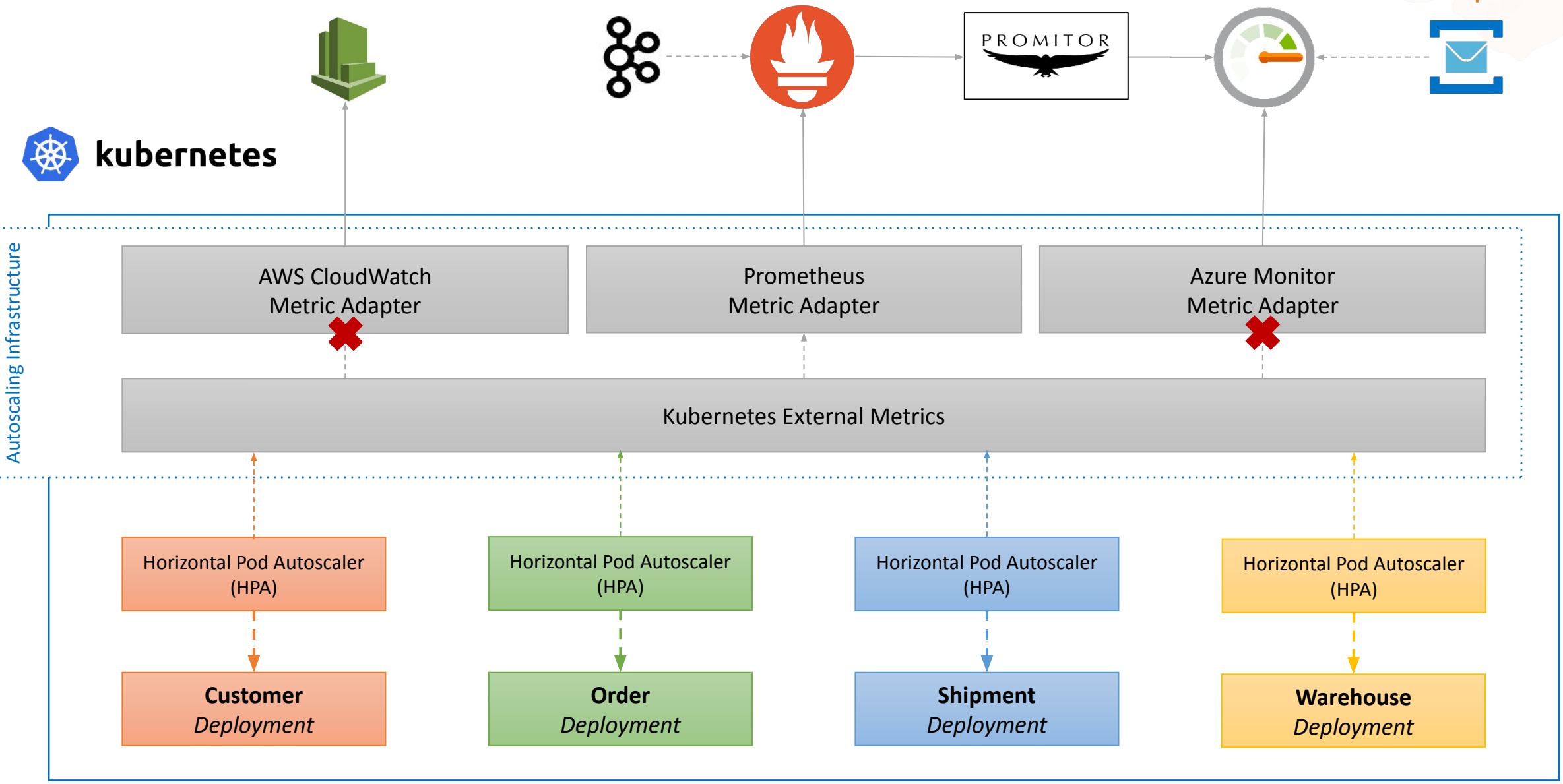
What is KEDA?

“We are using Kubernetes as a platform for the departments/projects to run their applications. The billing depends on the consumed resources and so, as you might imagine, some of them would like to scale down if there’s no work to do.

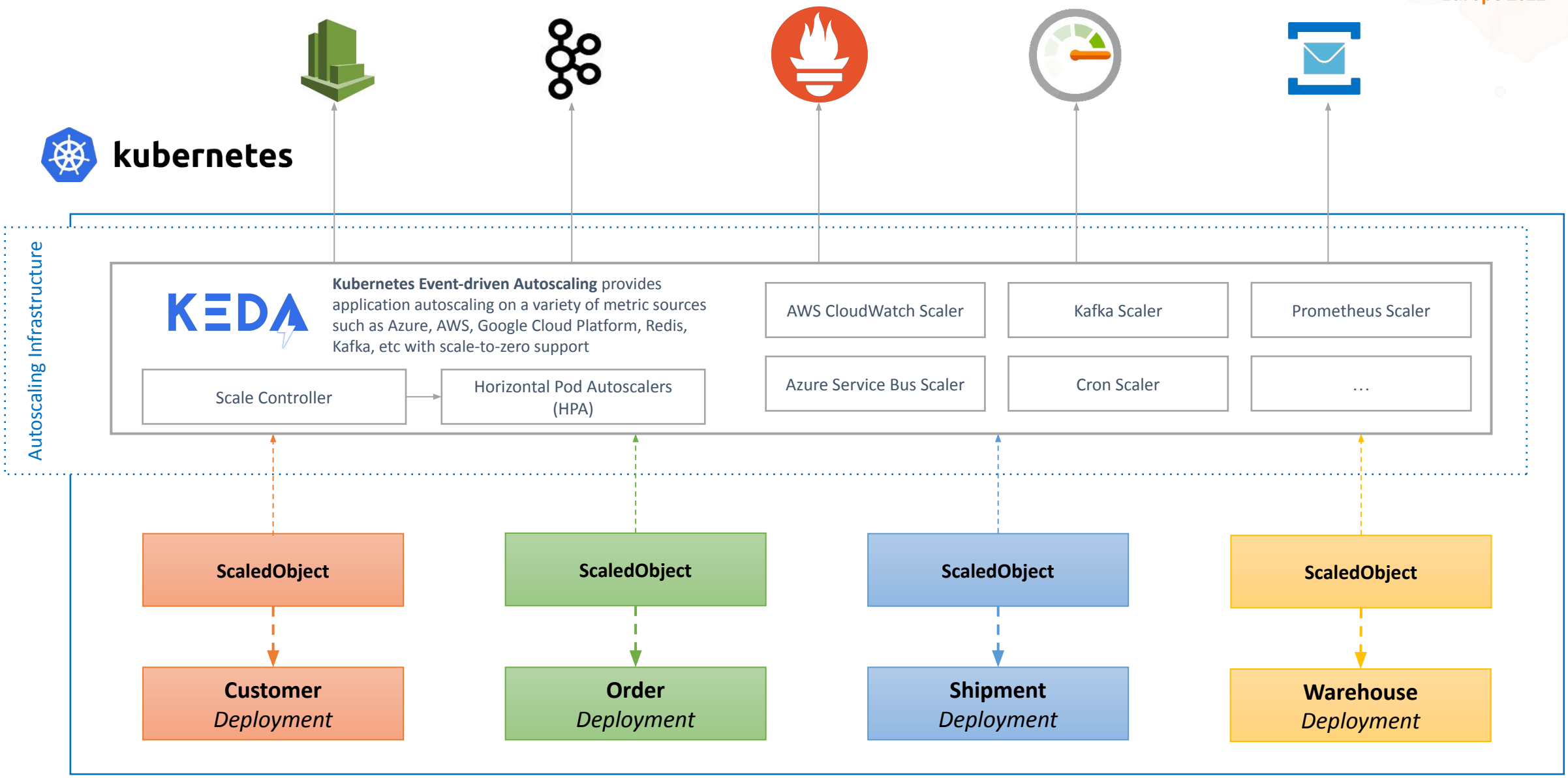
Until now we only supported the default CPU/Mem based scaling,
because scaling based on **custom metrics introduces complexity we didn’t want to maintain**.

But **with KEDA this seems a bit different**.”

What is KEDA?


kubernetes
PROMITOR
Autoscaling Infrastructure
AWS CloudWatch Metric Adapter
Prometheus Metric Adapter
Azure Monitor Metric Adapter
Kubernetes External Metrics
Horizontal Pod Autoscaler (HPA)
Horizontal Pod Autoscaler (HPA)
Horizontal Pod Autoscaler (HPA)
Horizontal Pod Autoscaler (HPA)
Customer Deployment
Order Deployment
Shipment Deployment
Warehouse Deployment

What is KEDA?


kubernetes
Autoscaling Infrastructure
KEDA
Kubernetes Event-driven Autoscaling provides application autoscaling on a variety of metric sources such as Azure, AWS, Google Cloud Platform, Redis, Kafka, etc with scale-to-zero support
Scale Controller
Horizontal Pod Autoscalers (HPA)
AWS CloudWatch Scaler
Kafka Scaler
Prometheus Scaler
Azure Service Bus Scaler
Cron Scaler
…
ScaledObject
ScaledObject
ScaledObject
ScaledObject
Customer Deployment
Order Deployment
Shipment Deployment
Warehouse Deployment

KEDA project and community

- Project aims to make **K**ubernetes **E**vent **D**riven **A**utoscaling dead simple
- Started as a partnership between Red Hat and Microsoft (Feb 2019)
- Donated into CNCF as a Sandbox project (Mar 2020)
- KEDA 2.0 brought major redesign (Nov 2020)
- Promoted to **CNCF Incubation** project (Aug 2021)
- KEDA **2.7** has been recently released (May 2022)
- KEDA releases ~ every 3 months (2.8 -> Aug 2022)
- https://keda.sh

KEDA project and community

Cor

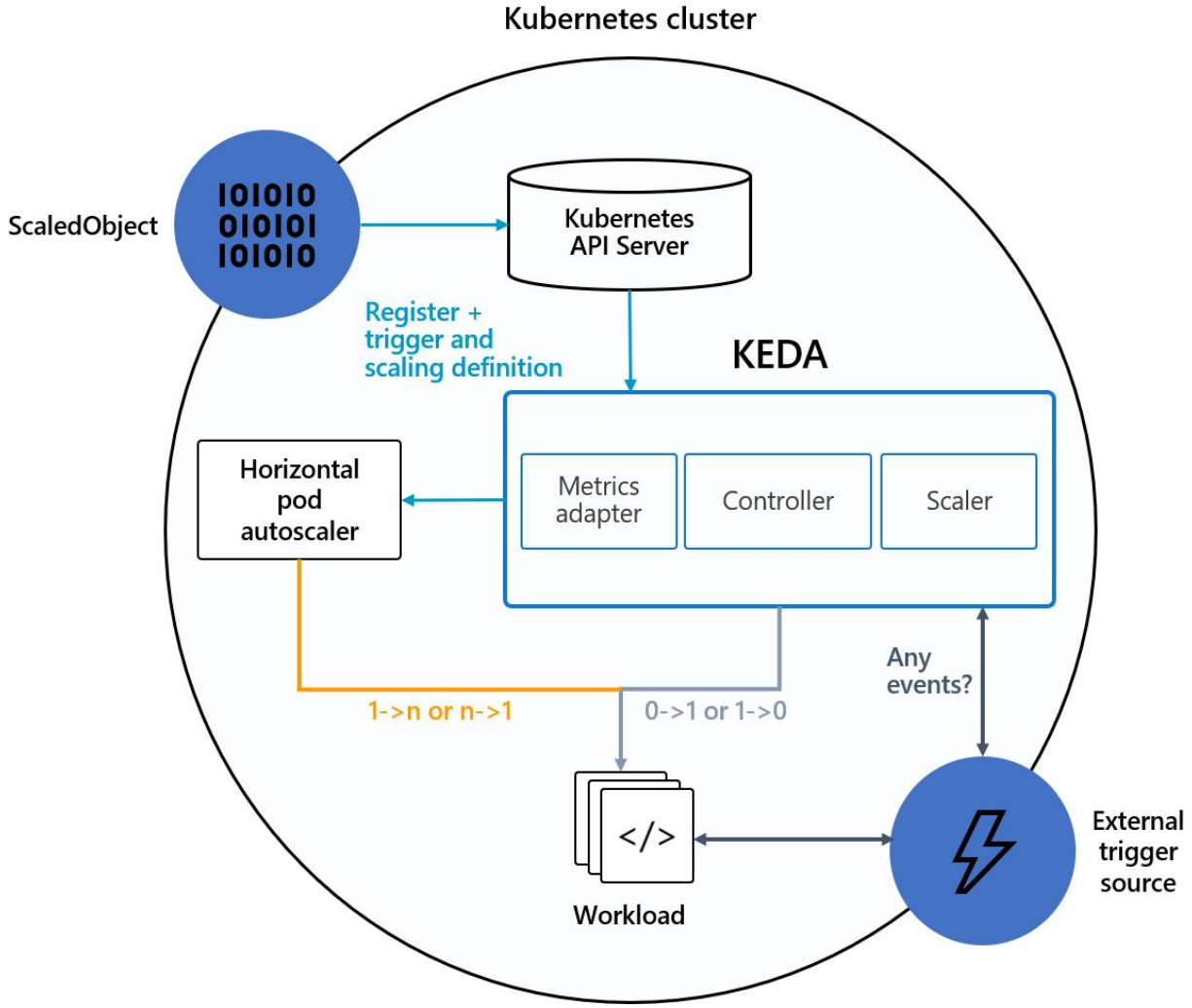
- 4.9k stars on GitHub
- ~190 contributors, incl.
  - Red Hat
  - Docplanner Tech
  - Microsoft
  - Codit
  - IBM
- Bi-weekly community standups
- https://keda.sh/community

KEDA concepts and architecture

- Automatically scale Kubernetes Deployments, Jobs & Custom Resources
- Provides **50+** built-in scalers, but users can build own external scalers
  - Kafka, Prometheus, RabbitMQ, AWS services, Azure Services,...
- Scale resources based on **events** in the target scalers, eg. messages in Kafka topic
- Save resources by **scale to 0**
- KEDA **does not** manipulate the data, just scales the workload
- Installation through Helm or OLM Operator
- ARM support

KEDA concepts and architecture

- KEDA is built on top of Kubernetes
- Use **ScaledObject/ScaledJob** to define scaling metadata
- Manages workloads to scale to 0
- Registers itself as Kubernetes Metric Adapter
- Provides metrics for Horizontal Pod Autoscaler (HPA)



KEDA concepts and architecture

ScaledObject

- Can target **Deployment**, **StatefulSet** or **Custom Resource** with **/scale**
- **Multiple scalers** can be defined as triggers for the target workload
- User can specify **HPA related settings** to tweak the scaling behavior

```yaml
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: example-so
spec:
  scaleTargetRef:
        name: example-deployment
  minReplicaCount: 0
  maxReplicaCount: 100
  triggers:
  - type: kafka
    metadata:
        bootstrapServers: kafka.svc:9092
        consumerGroup: my-group
        topic: test-topic
        lagThreshold: '5'
```

KEDA concepts and architecture

ScaleJob

- Schedule **Kubernetes Job** based on events
- Useful option to handle **processing long running executions**

```yaml
apiVersion: keda.sh/v1alpha1
kind: ScaledJob
metadata:
  name: example-sj
spec:
  jobTargetRef:
      ... # standard k8s Job definition

  maxReplicaCount: 100
  triggers:
  - type: kafka
    metadata:
      bootstrapServers: kafka.svc:9092
      consumerGroup: my-group
      topic: test-topic
      lagThreshold: '5'
```

Advanced features

- Ability to specify **Fallback replicas count** - in case of problems
- Users can still tweak HPA settings if they want to (**scaling behavior**)
- Ability to **Pause autoscaling** (new in 2.7)
- KEDA exposes Prometheus metrics
- Users can extend KEDA implementing **External scalers** via gRPC interface or **Metrics API scalers** via Rest API.
  - KEDA HTTP Add-on
- https://keda.sh/docs/2.7/concepts/scaling-deployments/#scaledobject-spec

Production-grade authentication

- Typical security concerns:
  - Re-use secrets from scaled target – No separate identities
  - Duplication of secrets – Harder to manage & rotate
- Re-use trigger authentication across ScaledObject/ScaledJobs with **TriggerAuthentication** (namespaced) or **ClusterTriggerAuthentication**
- Provides out-of-the-box integration with sources such as:
  - Environment variables (on scale target)
  - Kubernetes secrets
  - Pod Identity (“No secret authentication” - Azure / AWS)
  - HashiCorp Vault
  - Azure Key Vault

KEDA concepts and architecture

KEDA vs Prometheus Adapter

```yaml
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: example-so
spec:
  scaleTargetRef:
        name: example-deployment
  minReplicaCount: 0
  maxReplicaCount: 100
  triggers:
  - type: kafka
    metadata:
        bootstrapServers: kafka.svc:9092
        consumerGroup: my-group
        topic: test-topic
        lagThreshold: '5'
```

```yaml
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: web-prometheus-adapter
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: web
  minReplicas: 1
  maxReplicas: 50
  metrics:
  - type: Object
    object:
      metric:
        name: http_requests_per_second_per_pod
      describedObject:
        apiVersion: v1
        kind: Service
        name: web-prometheus-adapter-service
      target:
        type: Value
        value: 25
```

```yaml
rules:
  default: false
  custom:
  - seriesQuery: 'http_requests_received_total'
    resources:
      overrides:
        namespace: {resource: "namespace"}
        service: {resource: "service"}
    name:
      matches: ""
      as: "http_requests_per_second_per_pod"
    metricsQuery:
'max(irate(<<.Series>>{<<.LabelMatchers>>}[1m]))
by (<<.GroupBy>>)'
```

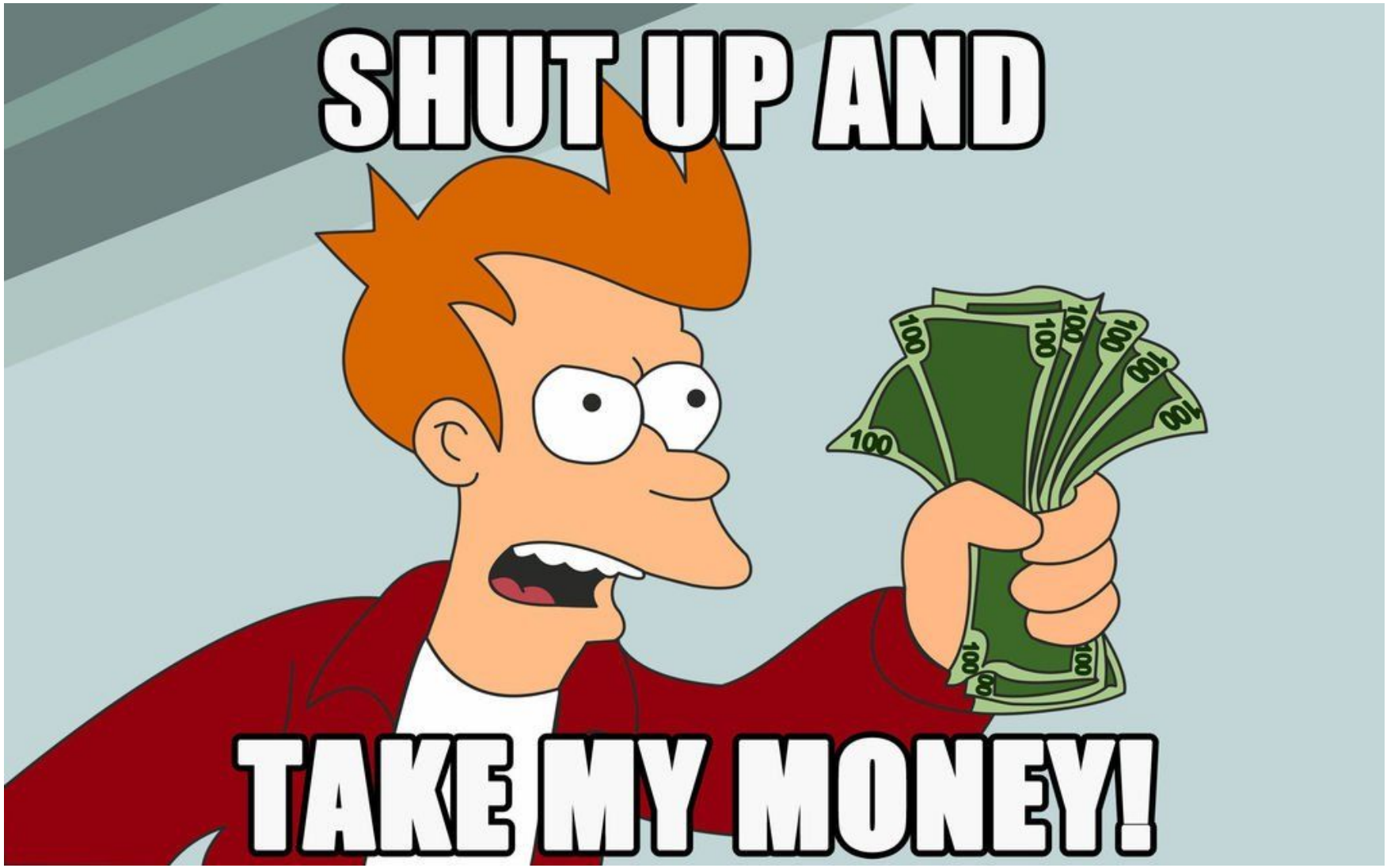
Demo time!

https://github.com/kedacore/sample-go-rabbitmq

Future development

- Cache metrics values in KEDA Metrics Server
- Multiple KEDA installations per cluster
- CloudEvents integration
- Open interface for Predictive autoscaling
- Smoother autoscaling (apply AI/ML model to incoming metrics)
- Suggest (and contribute :-)) what would you like to see in KEDA

Questions?


SHUT UP AND
TAKE MY MONEY!