
KubeCon
CloudNativeCon
North America 2023


KubeCon | CloudNativeCon

North America 2023

Exploring KEDA's Graduation and Advancements in Event-Driven Scaling

Zbynek Roubalik

CTO @ Kedify

About me

Zbynek Roubalik

Founder & CTO @ Kedify

- KEDA Maintainer
- Microsoft MVP

@zroubalik

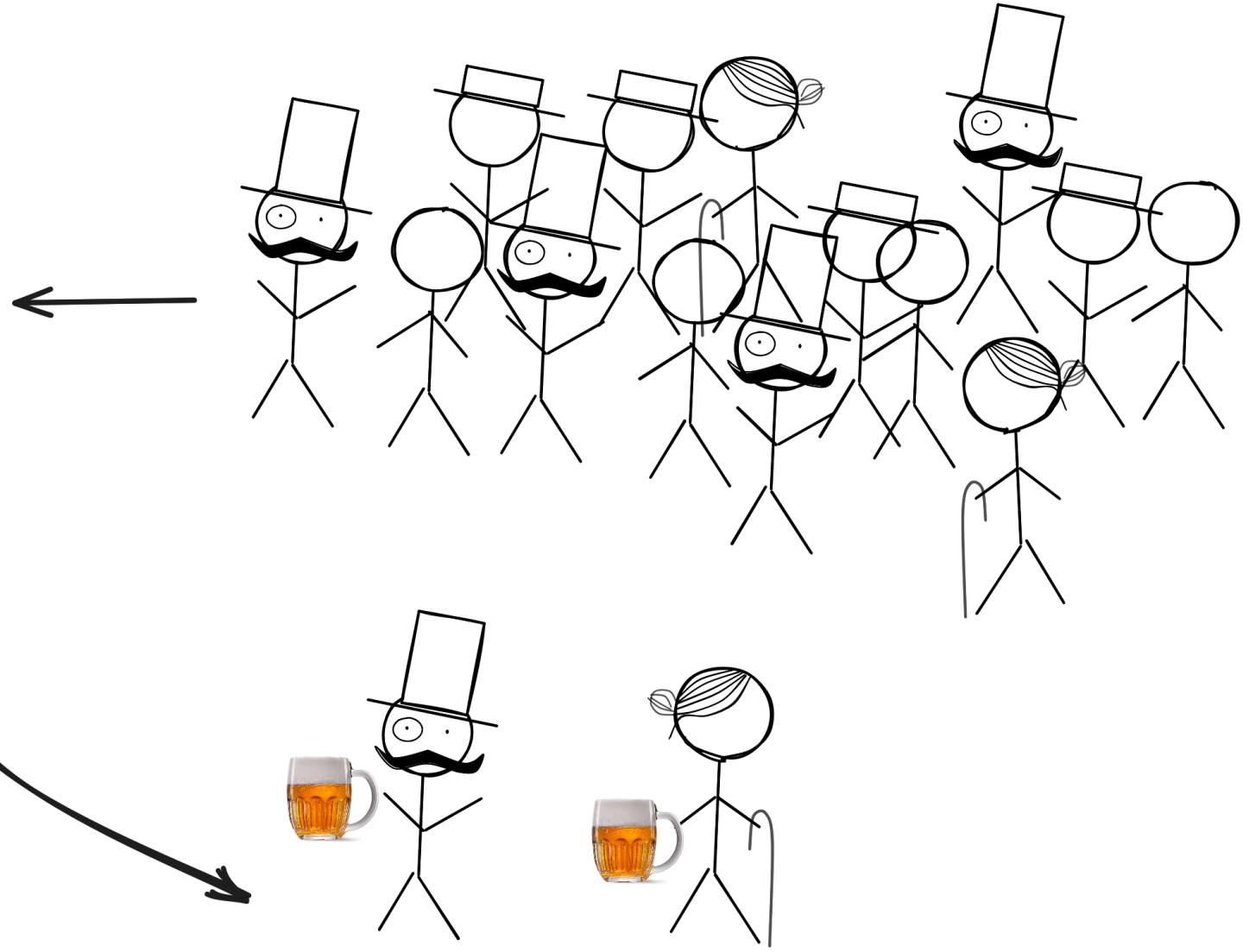
zbynek-roubalik

zroubalik

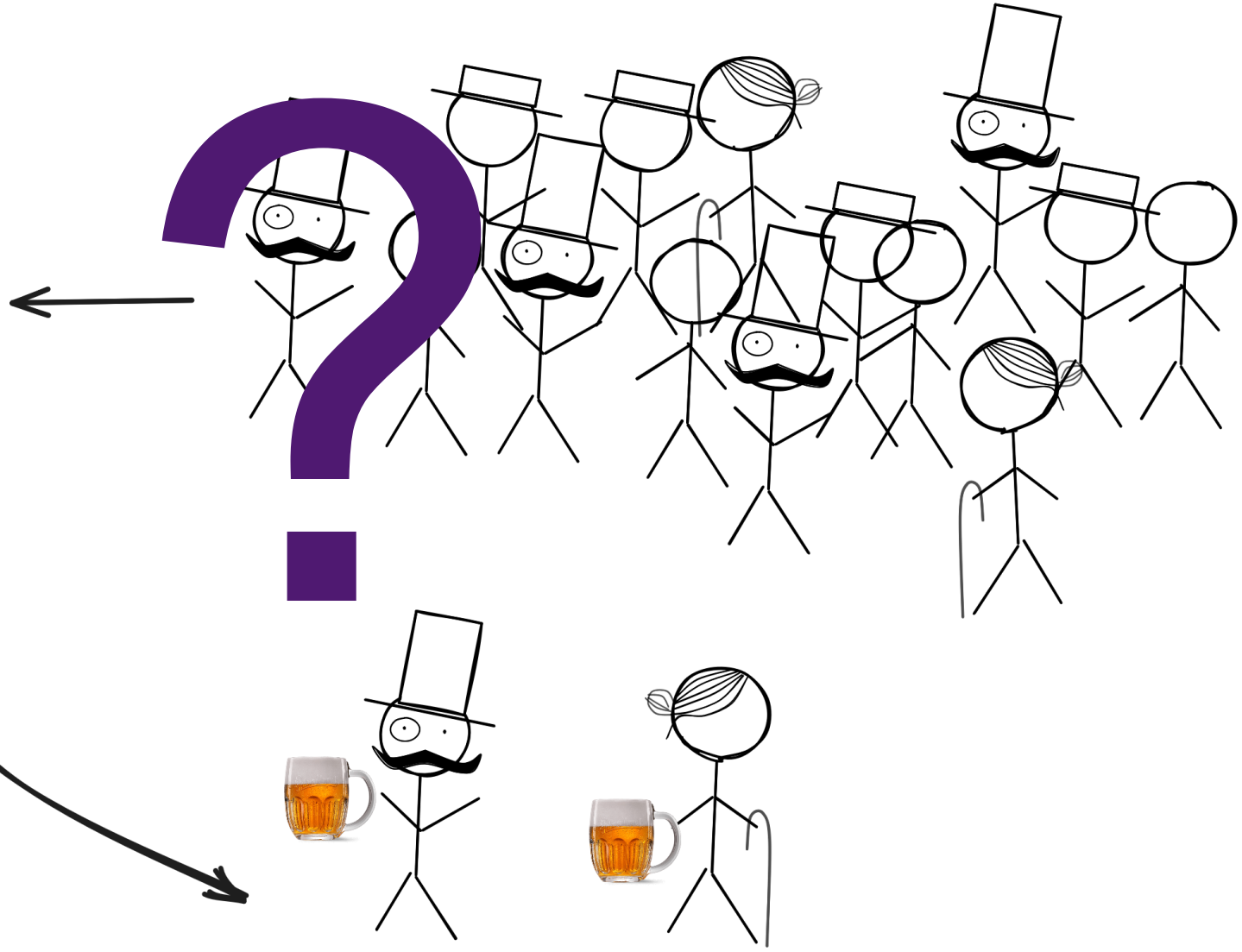
Agenda

- What is KEDA?
- New & advanced features
- Tips & best practices
- Demo
- Future
- Q & A

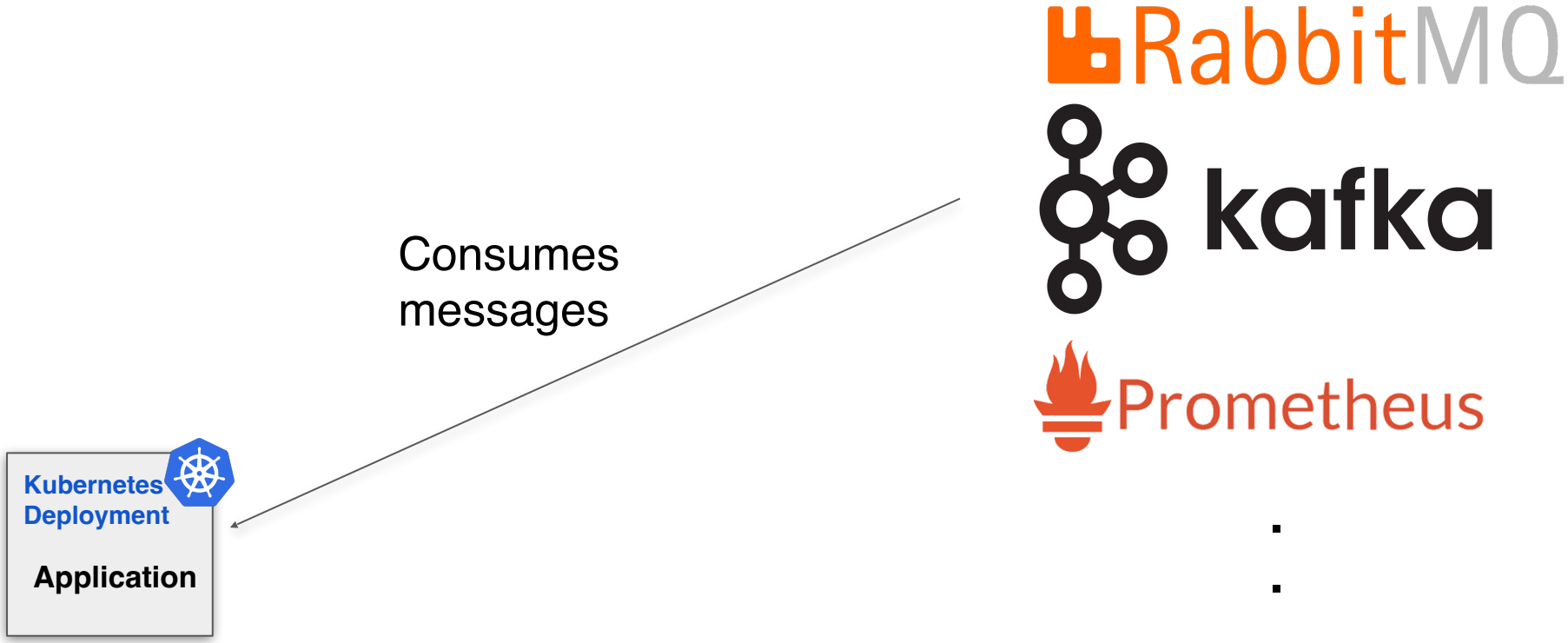
The Beer Problem



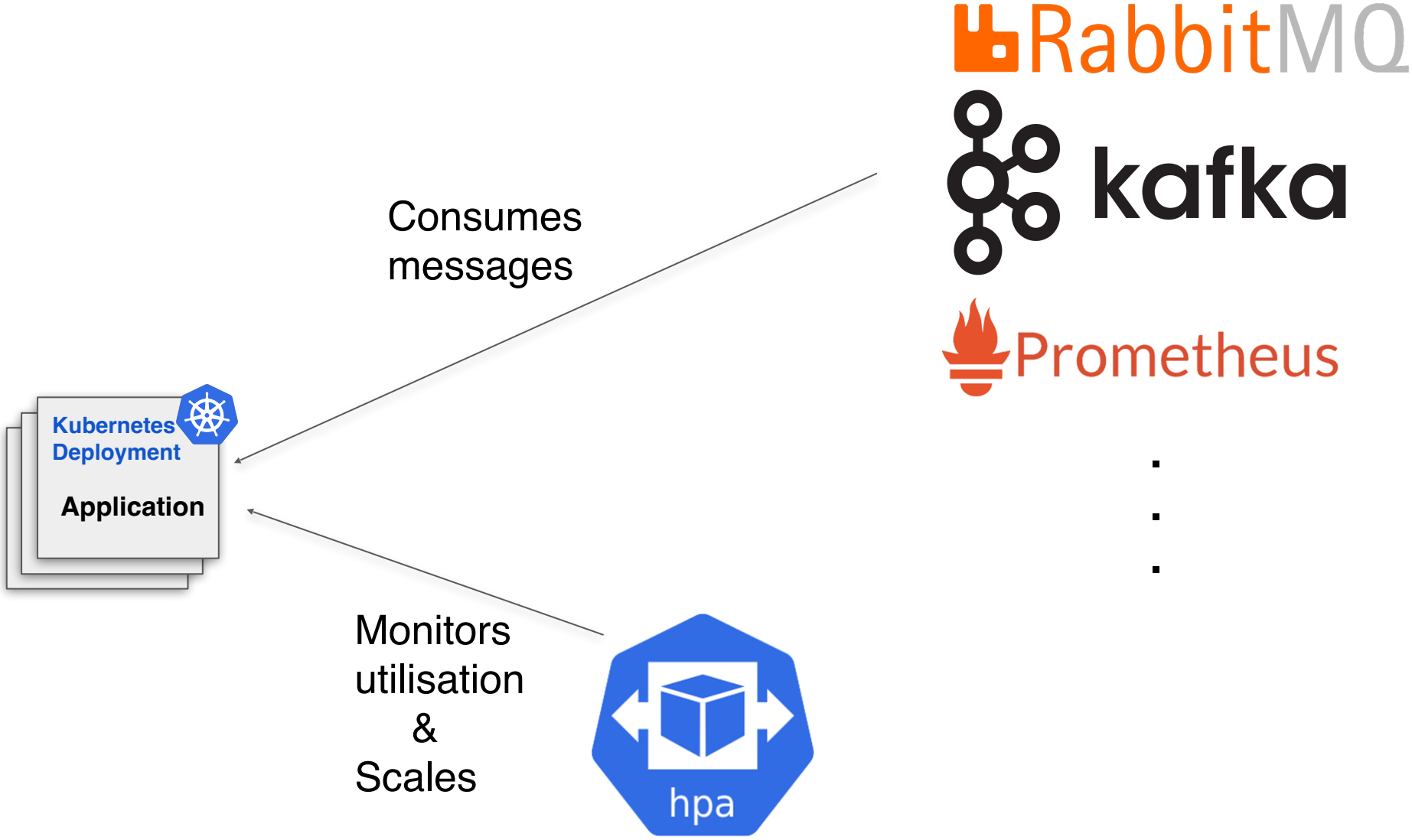
The Beer Problem



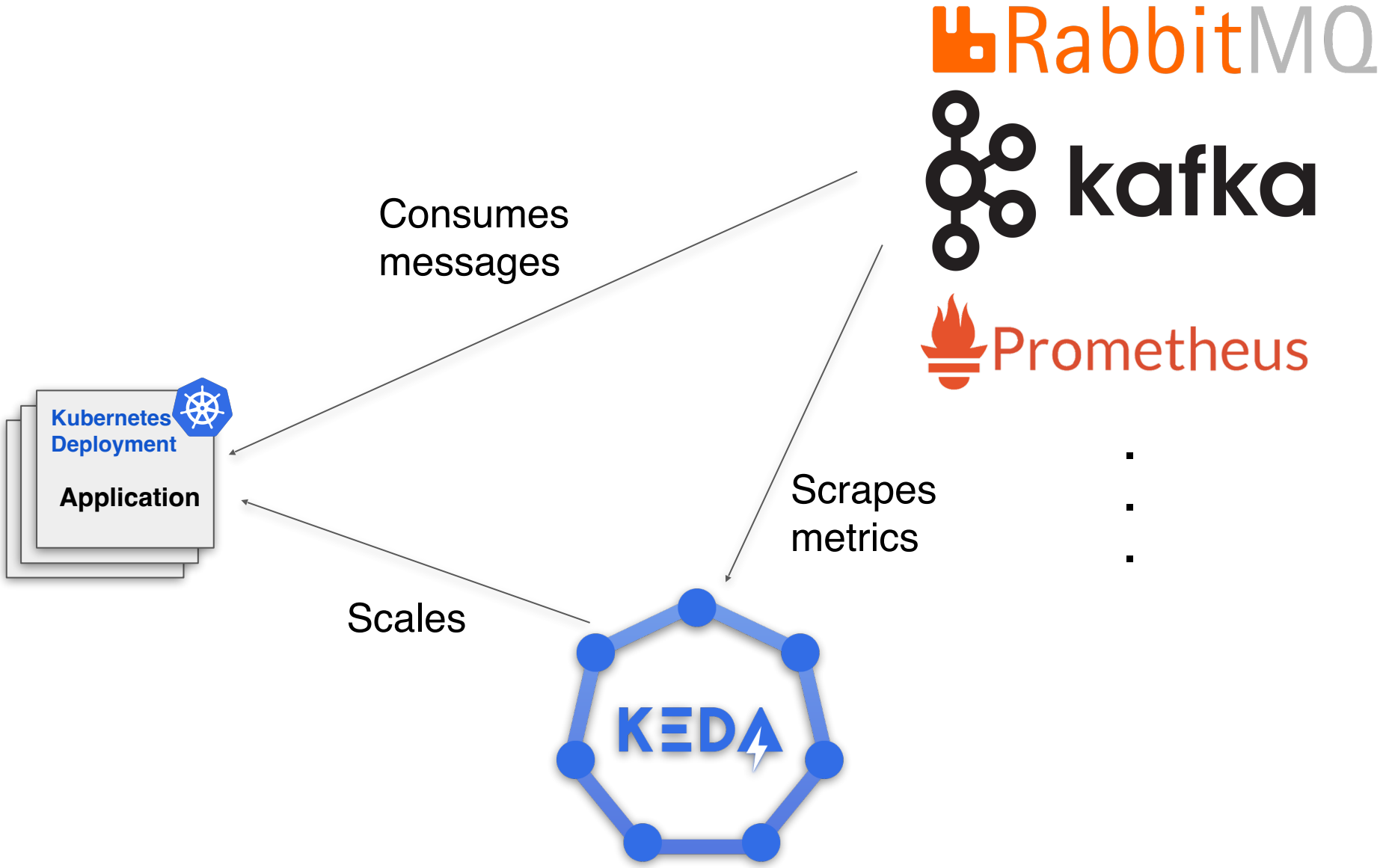
A problem


Consumes
messages
Kubernetes
Deployment
Application
RabbitMQ
kafka
Prometheus

An attempt to find a solution



The solution



Why autoscaling?

- Brings elasticity to your platform
- Ability to handle higher loads (peak hours)
- Save costs when apps are not needed

What is KEDA

- The Project aims to make **K**ubernetes **E**vent **D**riven **A**utoscaling dead simple
- Allows you to scale any deployment resource or job based on **events**, not only on CPU / Memory
- 65+ integrated event sources (Prometheus, RabbitMQ, Kafka, SQS, PostgreSQL, …..)
- https://keda.sh

KEDA is CNCF Graduated Project

KEDA Community

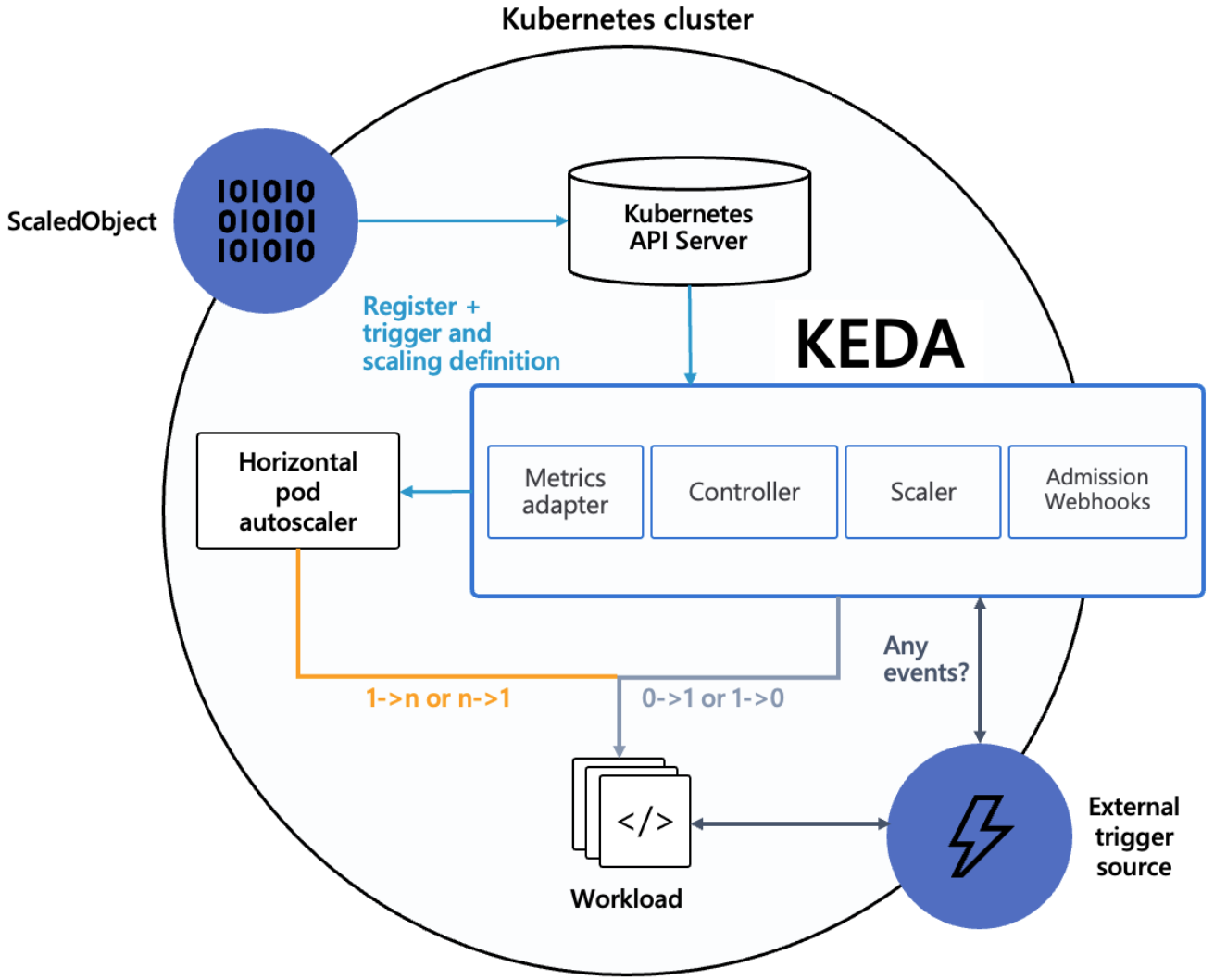
- 7k stars on GitHub
- ~320 contributors, incl.
  - Kedify
  - Microsoft
  - Red Hat
  - SCRM Lidl International Hub
  - Reddit

- KEDA Users survey:

Alibaba Cloud · AMPIFY · Apache Airflow · ASTRONOMER · baseten · BoldDesk by Syncfusion · Chattermill · CISCO · cloudtrade · Coralogix · cyera · devtron · Dig Security · Docplanner Tech · FedEx · fission · FORESIGHT · freepikcompany · Grafana Labs · Grab · HM Courts & Tribunals Service · intension · INTERCEPT · KENDAXA · KPMG · LeanIX · Meltwater · METAMAZE · Microsoft · nemlig.com · OpsWorks Co. · pollinate · PENINSULAR RESEARCH OPERATION · PURSFACTS · qonto · Red Hat · reddit · Roadwork · Schréder HYPERION · SCRM · SELDON · sharry · STILINGUE · SwissLife · Teamleader · vitc · WINGIE ENUYGUN · workiva · XBOX · zapier

KEDA architecture

- Built on top of Kubernetes
- **ScaledObject/ScaledJob** defines scaling metadata
- Manages workloads to scale to 0
- Publishes metrics HPA which makes most scaling decisions



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

```
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: example-so
spec:
  scaleTargetRef:
        name: example-deployment
  minReplicaCount: 0
  maxReplicaCount: 100
  triggers:
  - type: rabbitmq
     metadata:
        host: ”ampq://user:PASWORD@my-rabbit.com:5672”
        queueName: ”my-queue”
        queueLength: “5”
```

ScaledJob

- Schedule **Kubernetes Jobs** based on events

- Useful option to handle **processing long running executions**

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

  maxReplicaCount: 100
  triggers:
  - type: rabbitmq
     metadata:
        host: ”ampq://user:PASWORD@my-rabbit.com:5672”
        queueName: ”my-queue”
        queueLength: “5”
```

New & advanced features

- Certificate management
- Validation webhooks
- Prometheus metrics
- OpenTelemetry support
- Pausing of autoscaling
- Scaling modifiers for Multiple Scalers
- Changelog

Certificate management

- Certificates with for any internal communication
  - TLS1.3 encryption between components
  - CA validation for communications between API server and KEDA
  - Users can use their own certificates
    - Integration with cert-manager for helm chart
- Support to register custom CA in the trusted store
- Configurable minimal TLS version (with TLS1.2 as default)

Prometheus metrics

Operator

The KEDA Operator exposes Prometheus metrics which can be scraped on port `8080` at `/metrics`. The following metrics are being gathered:

- `keda_build_info` - Info metric, with static information about KEDA build like: version, git commit and Golang runtime info.
- `keda_scaler_active` - This metric marks whether the particular scaler is active (value == 1) or in-active (value == 0).
- `keda_scaler_metrics_value` - The current value for each scaler's metric that would be used by the HPA in computing the target average.
- `keda_scaler_metrics_latency` - The latency of retrieving current metric from each scaler.
- `keda_scaler_errors` - The number of errors that have occurred for each scaler.
- `keda_scaler_errors_total` - The total number of errors encountered for all scalers.
- `keda_scaled_object_errors` - The number of errors that have occurred for each ScaledObject.
- `keda_resource_totals` - Total number of KEDA custom resources per namespace for each custom resource type (CRD).
- `keda_trigger_totals` - Total number of triggers per trigger type.
- `keda_internal_scale_loop_latency` - Total deviation (in milliseconds) between the expected execution time and the actual execution time for the scaling loop. This latency could be produced due to accumulated scalers latencies or high load. This is an internal metric.
- Metrics exposed by the `Operator SDK` framework as explained here.

Admission Webhooks

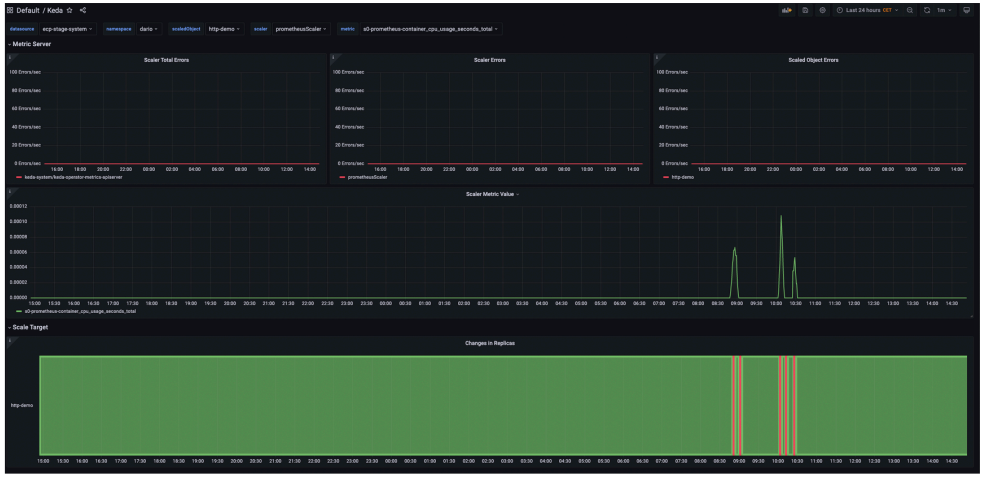
The KEDA Webhooks expose Prometheus metrics which can be scraped on port `8080` at `/metrics`. The following metrics are being gathered:

- `keda_webhook_scaled_object_validation_total` - The current value for scaled object validations.
- `keda_webhook_scaled_object_validation_errors` - The number of validation errors.

[Integrate with Prometheus | KEDA]

Premade Grafana dashboard

A premade Grafana dashboard is available to visualize metrics exposed by the KEDA Metrics Adapter.



OpenTelemetry support

Metric	Description
`keda.build.info`	Info metric, with static information about KEDA build like: version, git commit and Golang runtime info.
`keda.scaler.active`	This metric marks whether the particular scaler is active (value == 1) or in
`keda.scaler.metrics.value`	The current value for each scaler's metric that would be used by the HPA in computing the target average.
`keda.scaler.metrics.latency`	The latency of retrieving current metric from each scaler.
`keda.scaler.errors`	The number of errors that have occurred for each scaler.
`keda.scaler.errors.total`	The total number of errors encountered for all scalers.
`keda.scaled.object.errors`	The number of errors that have occurred for each ScaledObject.
`keda.resource.totals`	Total number of KEDA custom resources per namespace for each custom resource type (CRD).
`keda.trigger.totals`	Total number of triggers per trigger type.
`keda.internal.scale.loop.latency`	Total deviation (in milliseconds) between the expected execution time and the actual execution time for the scaling loop. This latency could be produced due to accumulated scalers latencies or high load. This is an internal metric.

OpenTelemetry Support | KEDA

Pausing of Autoscaling

- Autoscaling can be paused by annotating the ScaledObject
- Pausing to the current number of replicas
- Pausing to specific number of replicas
- ScaledJob pausing in development

https://keda.sh/docs/2.12/concepts/scaling-deployments/#pause-autoscaling

Scaling Modifiers

- When having multiple scalers or would like to modify the metric value
- Instead of using MAX (default for HPA) use a complex formula to decide the final metric
- Formula accepts mathematical and conditional statements

```
advanced:
  scalingModifiers:
    formula: "(trig_one + trig_two)/2"
    target: "2"
    activationTarget: "2"
    metricType: "AverageValue"
...
triggers:
  - type: kubernetes-workload
    name: trig_one
    metadata:
      podSelector: 'pod=workload-test'
  - type: metrics-api
    name: trig_two
    metadata:
      url: "https://mockbin.org/bin/33
      valueLocation: "tasks"
```

Scaling Modifiers for Multiple Scalers

- „We have the requirement to always be over provisioned by 3 pods. Our pod can process 3 messages at a time.“

- Formula: „trigger + 9“ = trigger + (3 pods x 3 msgs)

https://keda.sh/docs/latest/concepts/scaling-deployments/#scaling-modifiers-experimental

Best practices

- Use Fallback!
- Polling Interval & Metrics Caching
- HPA Scaling Behaviour
- Kubernetes Metrics

Use Fallback!

- Use **Fallback** feature to handle problems with scalers

- Helps keeping the workloads up

https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#configurable-scaling-behavior

```yaml
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: example-so
spec:
  scaleTargetRef:
        name: example-deployment
  minReplicaCount: 0
  maxReplicaCount: 100
  fallback:
      failureThreshold: 4
      replicas: 10
  triggers:
  - type: rabbitmq
    ...
```

Polling Interval & Metrics Caching

- **Polling Interval** is only relevant to 0<->1 scaling!
- Frequency of queries for 1<->N scaling is controller by HPA
  - default 15s, --horizontal-pod-autoscaler-sync-period
- Consider using **Metrics Caching** feature
  - metrics are scraped only each Polling Interval

https://keda.sh/docs/latest/concepts/scaling-deployments/#caching-metrics-experimental

HPA Scaling Behaviour

- **Stabilization window** prevents replica count flapping
- **Scaling policies** control the rate of change of replicas while scaling

https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#configurable-scaling-behavior

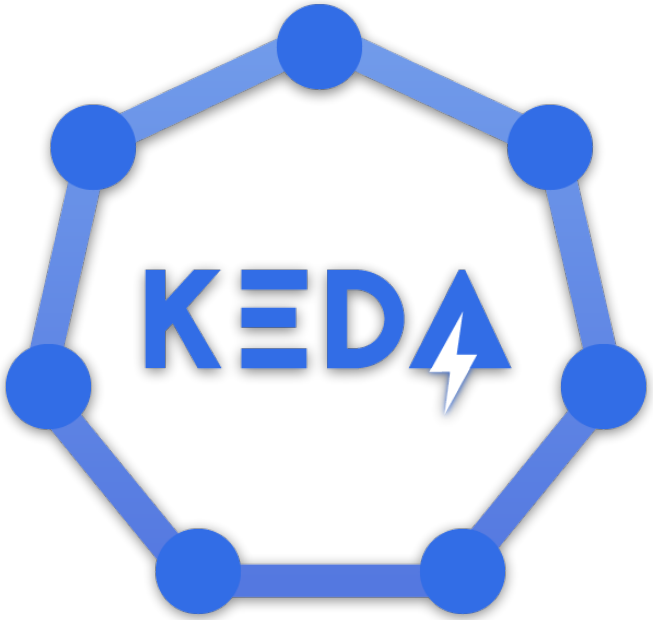
Kubernetes metrics

- Why the HPA reports metrics like **4800m/5**?
- Metric types:
  - Average
  - Value
  - Utilization

https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#algorithm-details

Demo

Demo time!



Future

- CloudEvents integration (almost there)
- Multi-tenant installations
- Open interface for Predictive autoscaling
- Global configuration
- Plugin management
- Carbon aware autoscaling
  - https://github.com/kedacore/keda/issues/4463

Session Feedback

Please scan the QR Code above
to leave feedback on this session

KEDA Users Survey

Please scan the QR Code above
if you are KEDA user to fill survey