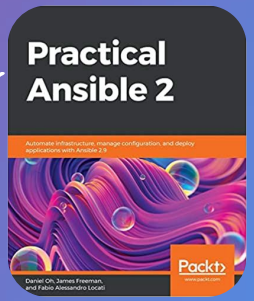

APACHECON
North America
Event-driven autoscaling through Apache Kafka Source, KEDA, and Knative Integration

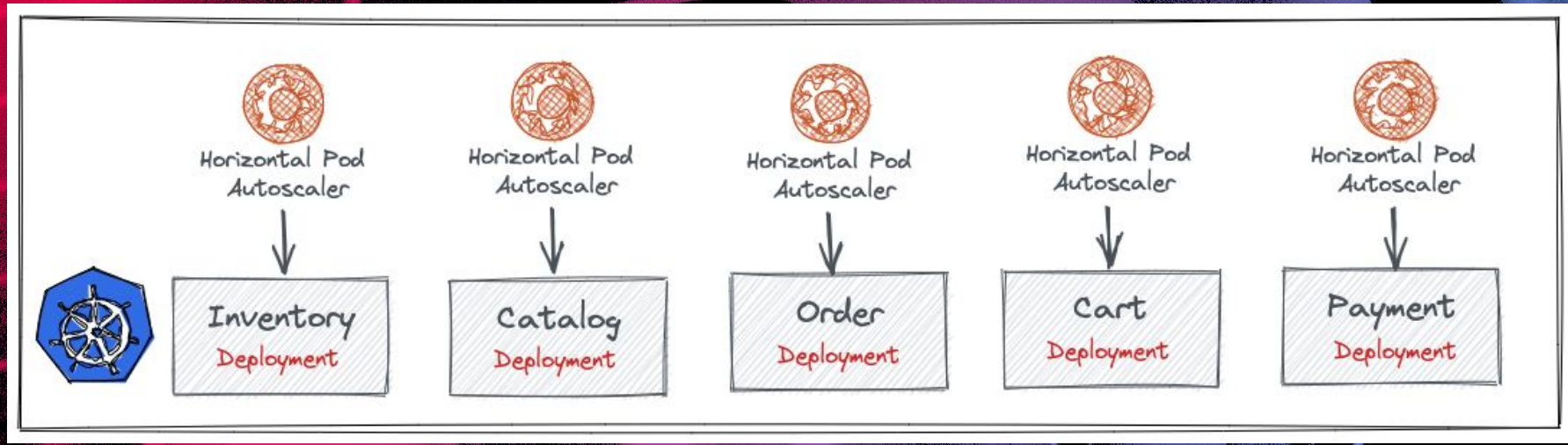
Daniel Oh

- Developer Advocate at Red Hat
  - Cloud Native Runtimes
  - Serverless, Service Mesh, and GitOps Practices
- CNCF Ambassador
- Advisory Board Member of Global Skill Development Council
- Opensource.com Correspondents
- Public Speaker & Published Author

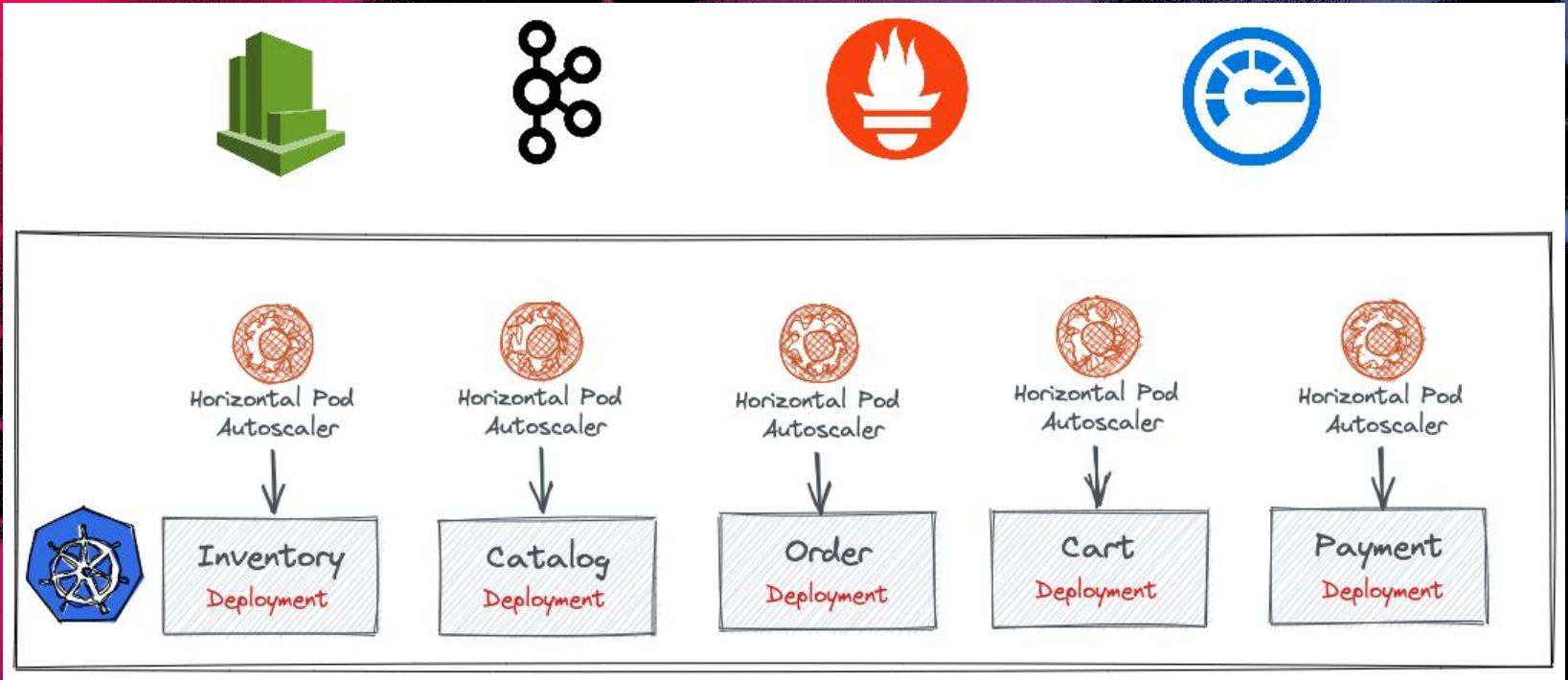
@danieloh30
bit.ly/danielohtv
danieloh30

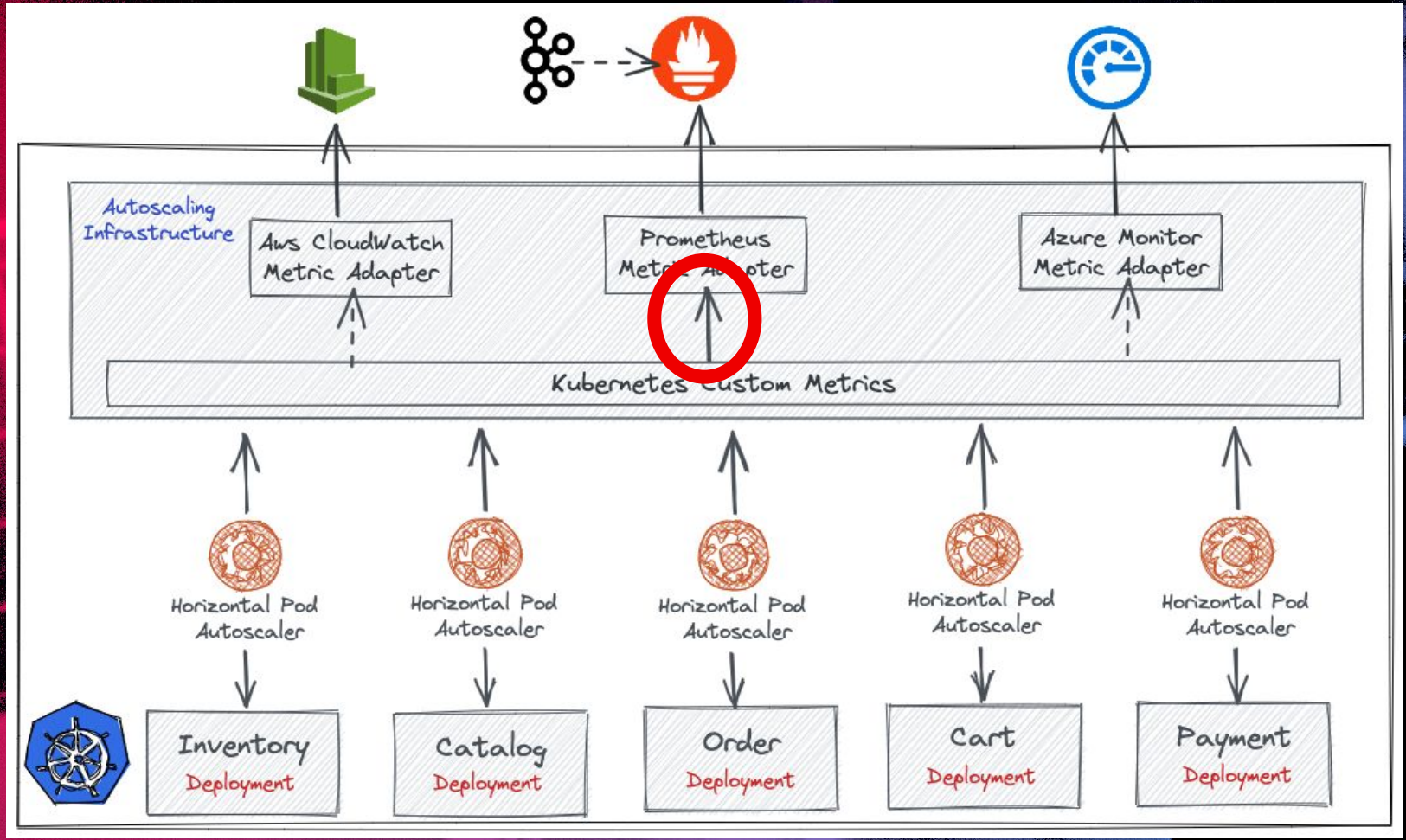


Autoscaling Architecture on Kubernetes



Autoscaling Architecture on Kubernetes w/ External Services




Autoscaling Infrastructure
Aws CloudWatch Metric Adapter
Prometheus Metric Adapter
Azure Monitor Metric Adapter
Kubernetes Custom Metrics
Horizontal Pod Autoscaler
Horizontal Pod Autoscaler
Horizontal Pod Autoscaler
Horizontal Pod Autoscaler
Horizontal Pod Autoscaler
Inventory Deployment
Catalog Deployment
Order Deployment
Cart Deployment
Payment Deployment


KEDA

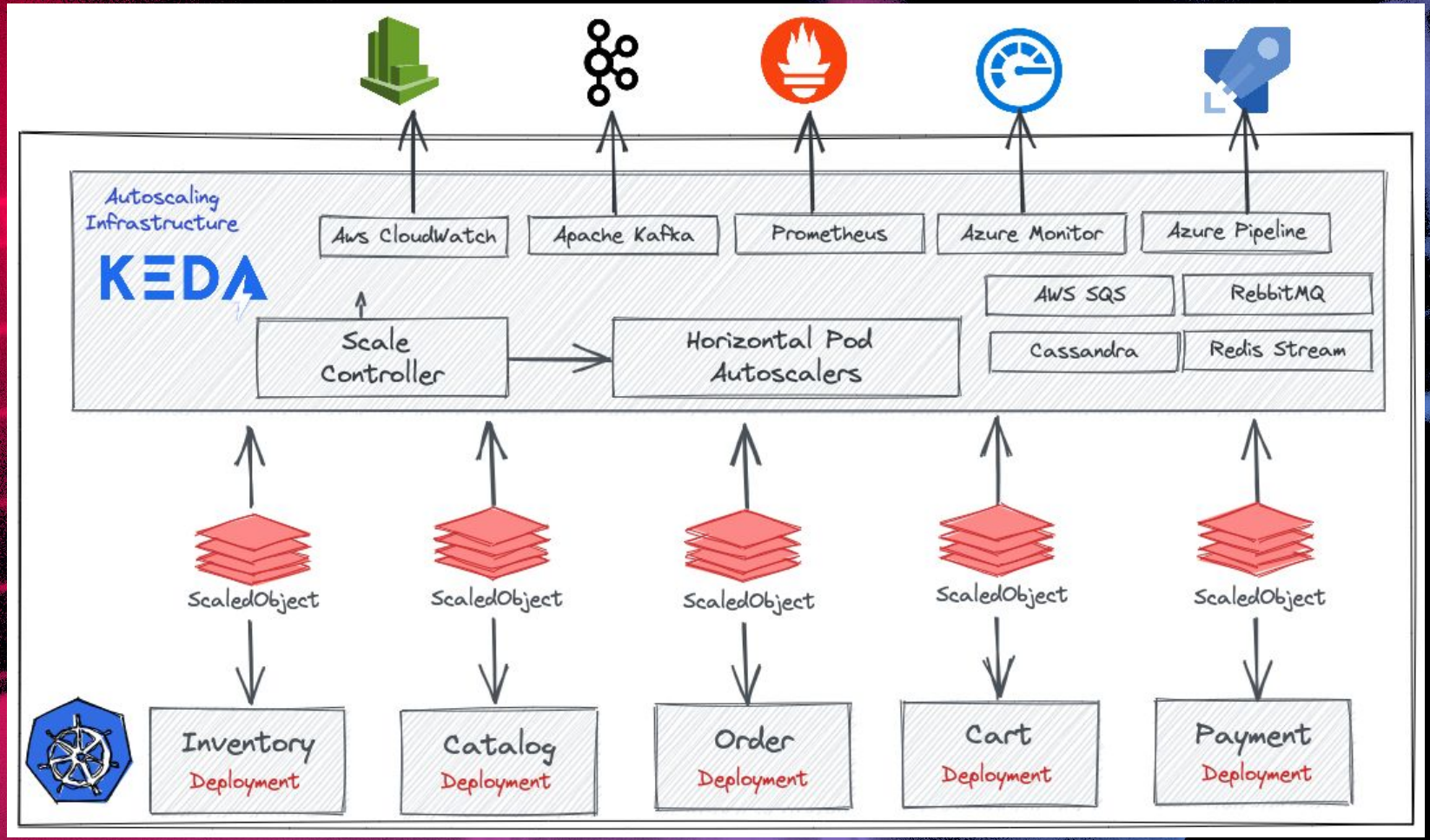
KEDA

- 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)
- Promoting to CNCF Incubation project (Aug 2021)
- KEDA **2.8** has been released recently (Sep 2022)
- https://keda.sh

https://keda.sh/docs/2.8/scalers/

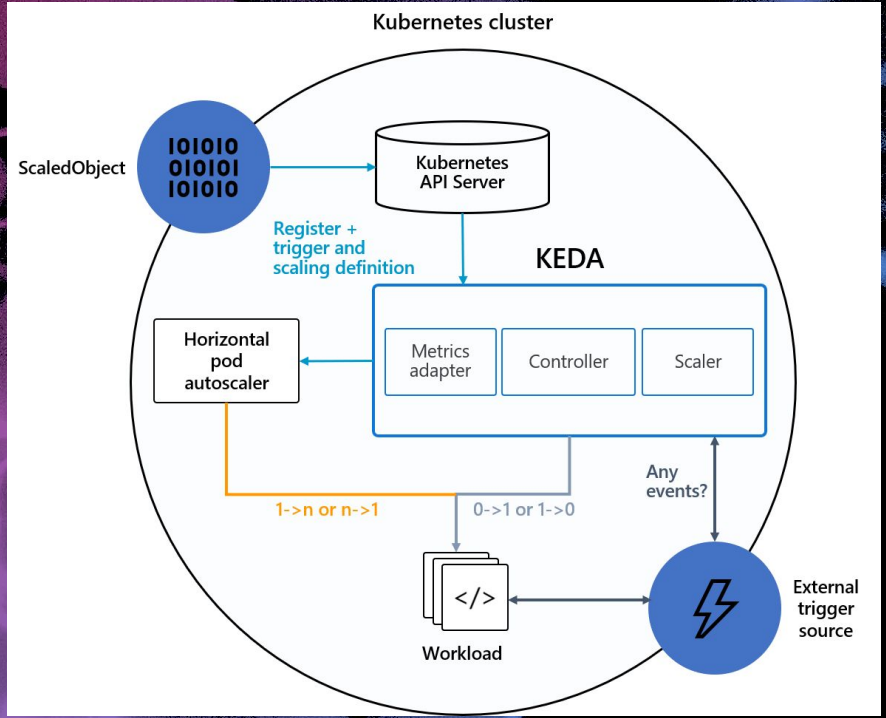
- Automatically scale Kubernetes Deployments, Jobs & Custom Resources
- Provides 56+ built-in scalers, but users can build own external scalers
- Kafka, Prometheus, RabbitMQ, AWS services, Azure Services,...
- Scale resources based on events in the target scalers, eg. messages in Kafka topic
- KEDA does not manipulate the data, just scales the workload
- Installation through OLM Operator or Helm

https://keda.sh/docs/2.8/scalers/


Autoscaling
Infrastructure
KEDA
Aws CloudWatch
Apache Kafka
Prometheus
Azure Monitor
Azure Pipeline
AWS SQS
RebbitMQ
Cassandra
Redis Stream
Scale
Controller
Horizontal Pod
Autoscalers
ScaledObject
ScaledObject
ScaledObject
ScaledObject
ScaledObject
Inventory
Deployment
Catalog
Deployment
Order
Deployment
Cart
Deployment
Payment
Deployment

How does KEDA works?

- KEDA is built on top of Kubernetes
- Use ScaledObject/ScaledJob to define scaling metadata
- Manages workloads to scale to 0
- Registers itself as k8s Metric Adapter
- Provides metrics for Horizontal Pod Autoscaler (HPA) to scale on



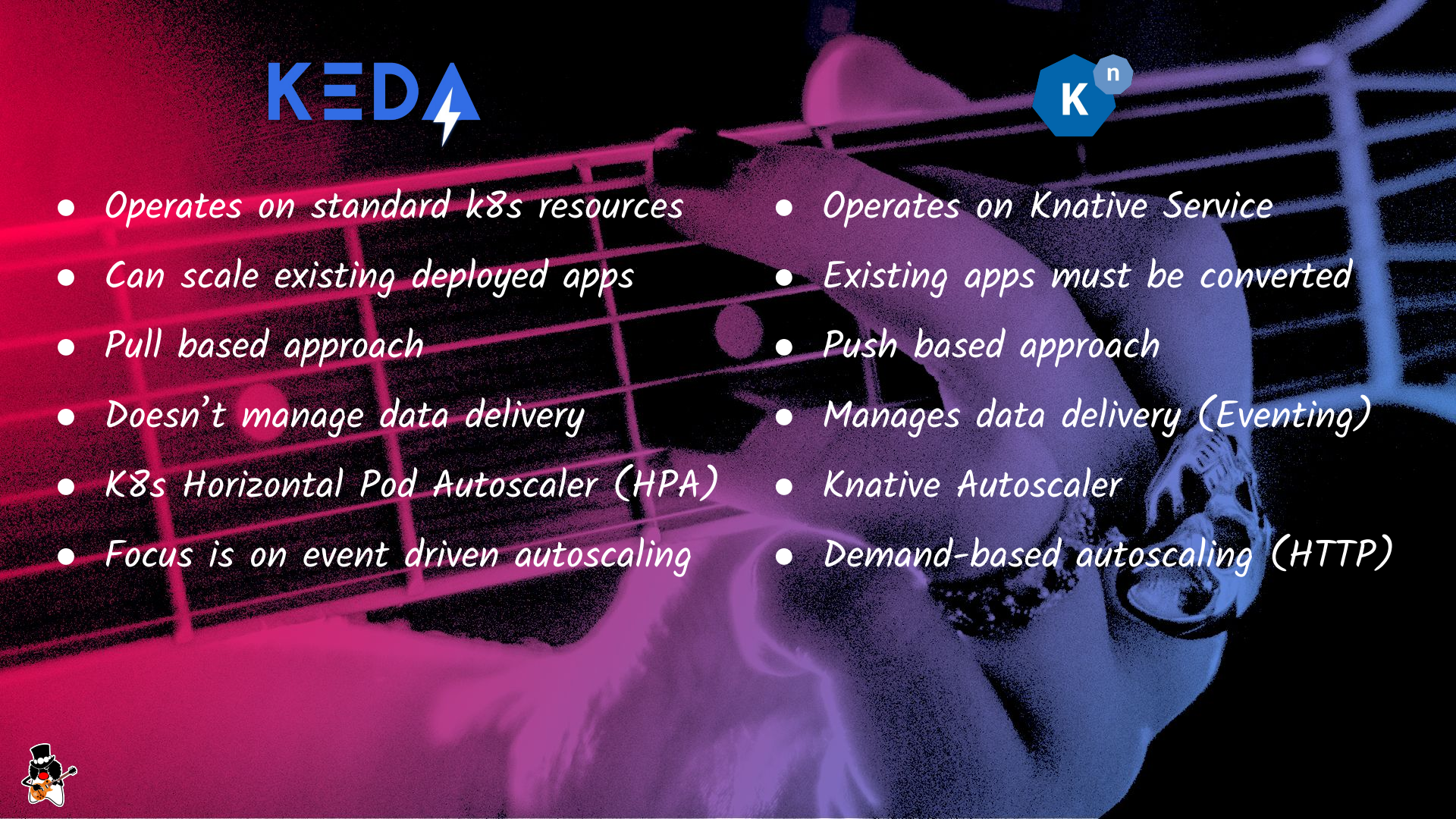
ScaledObject

- 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'
```


How about Serverless Autoscaling?


K
n

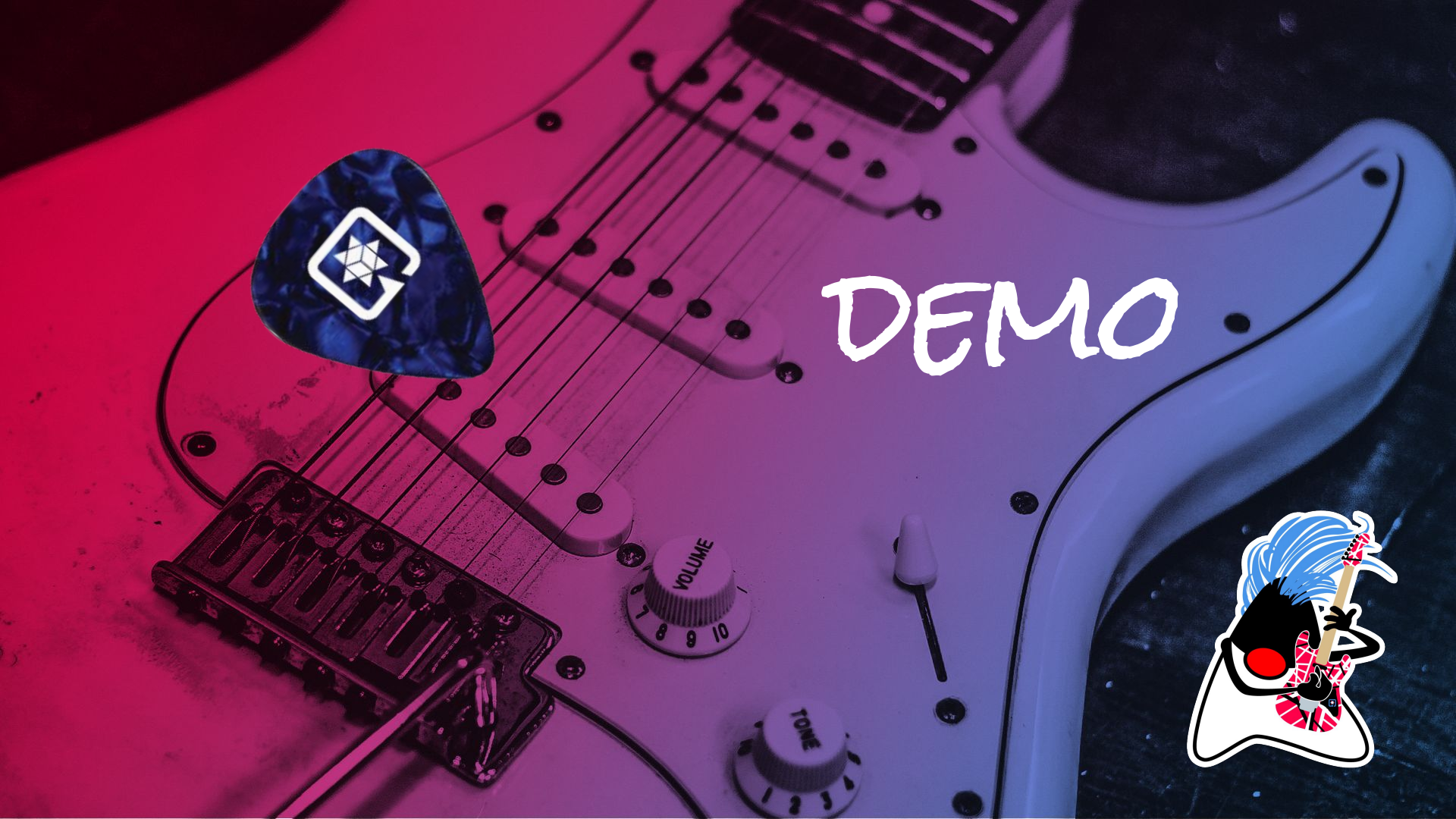

KEDA
K n
Operates on standard k8s resources
Can scale existing deployed apps
Pull based approach
Doesn't manage data delivery
K8s Horizontal Pod Autoscaler (HPA)
Focus is on event driven autoscaling
Operates on Knative Service
Existing apps must be converted
Push based approach
Manages data delivery (Eventing)
Knative Autoscaler
Demand-based autoscaling (HTTP)

Use Case #1

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

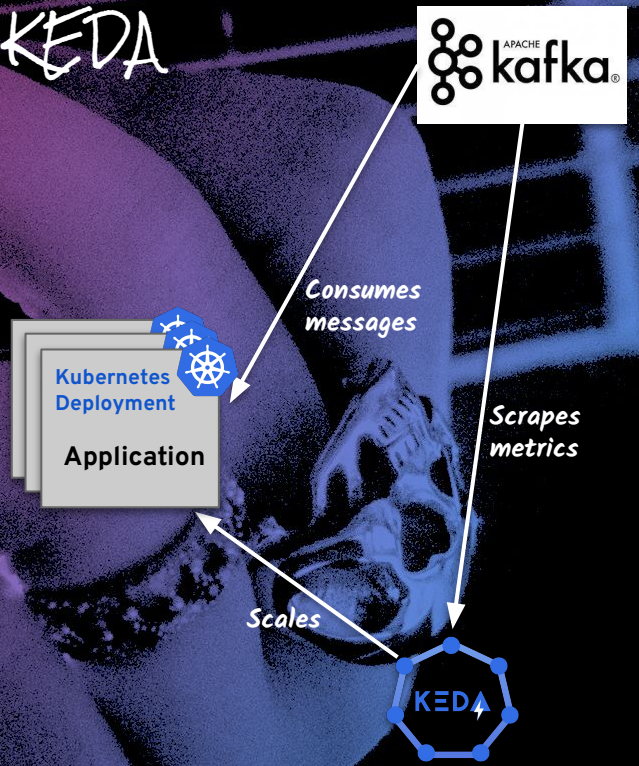
DEMO



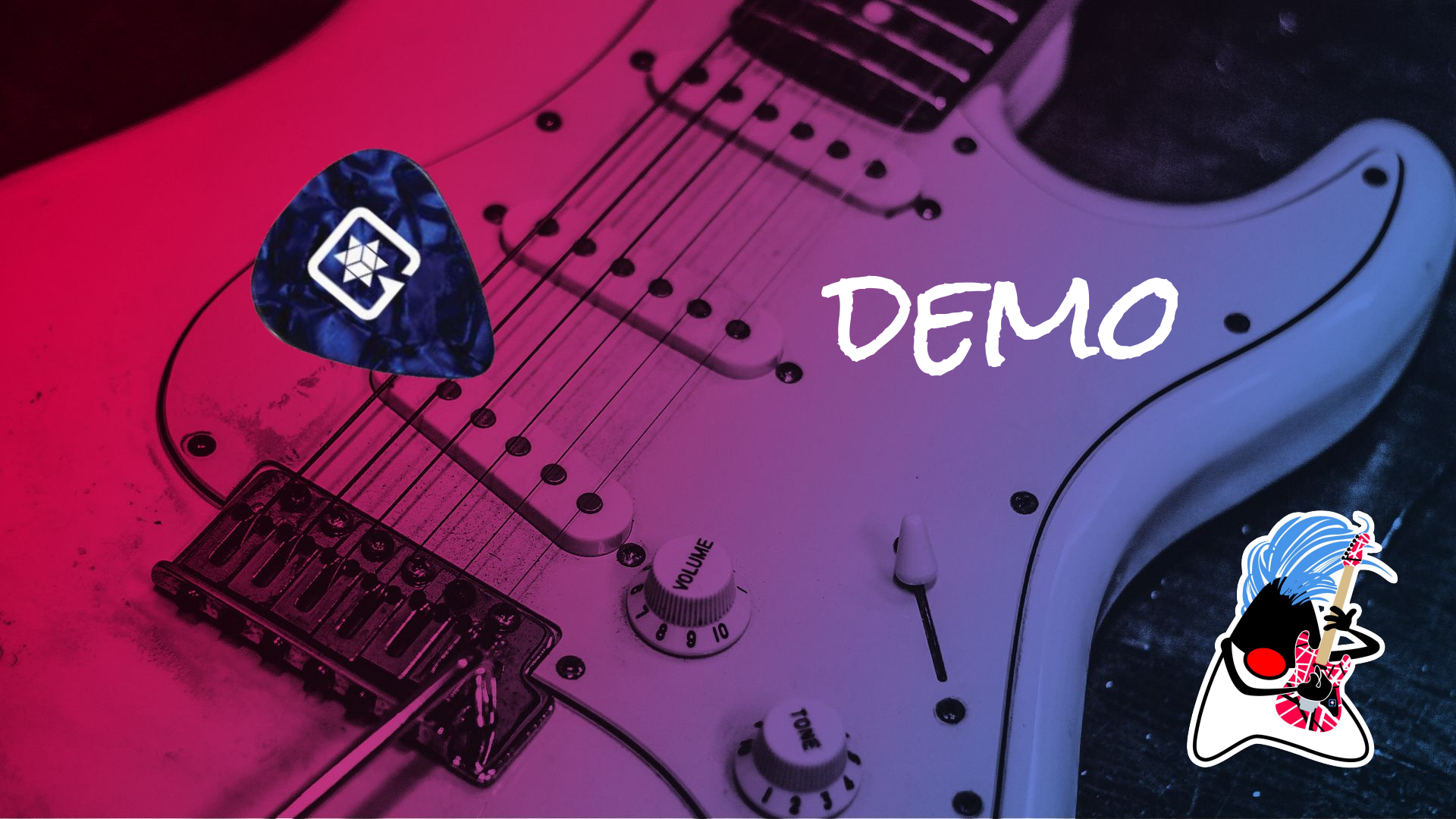
Use Case #2

Redesigned to utilize KEDA

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



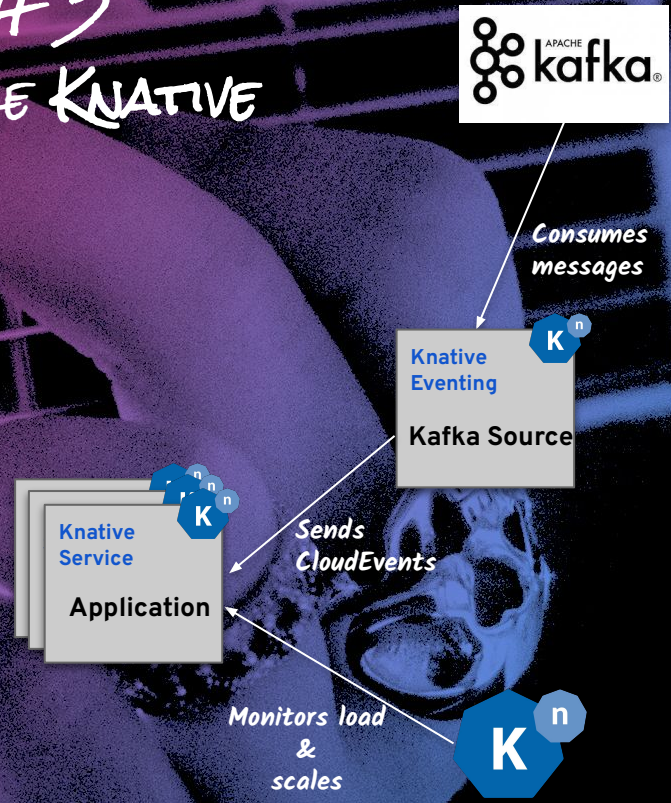
DEMO



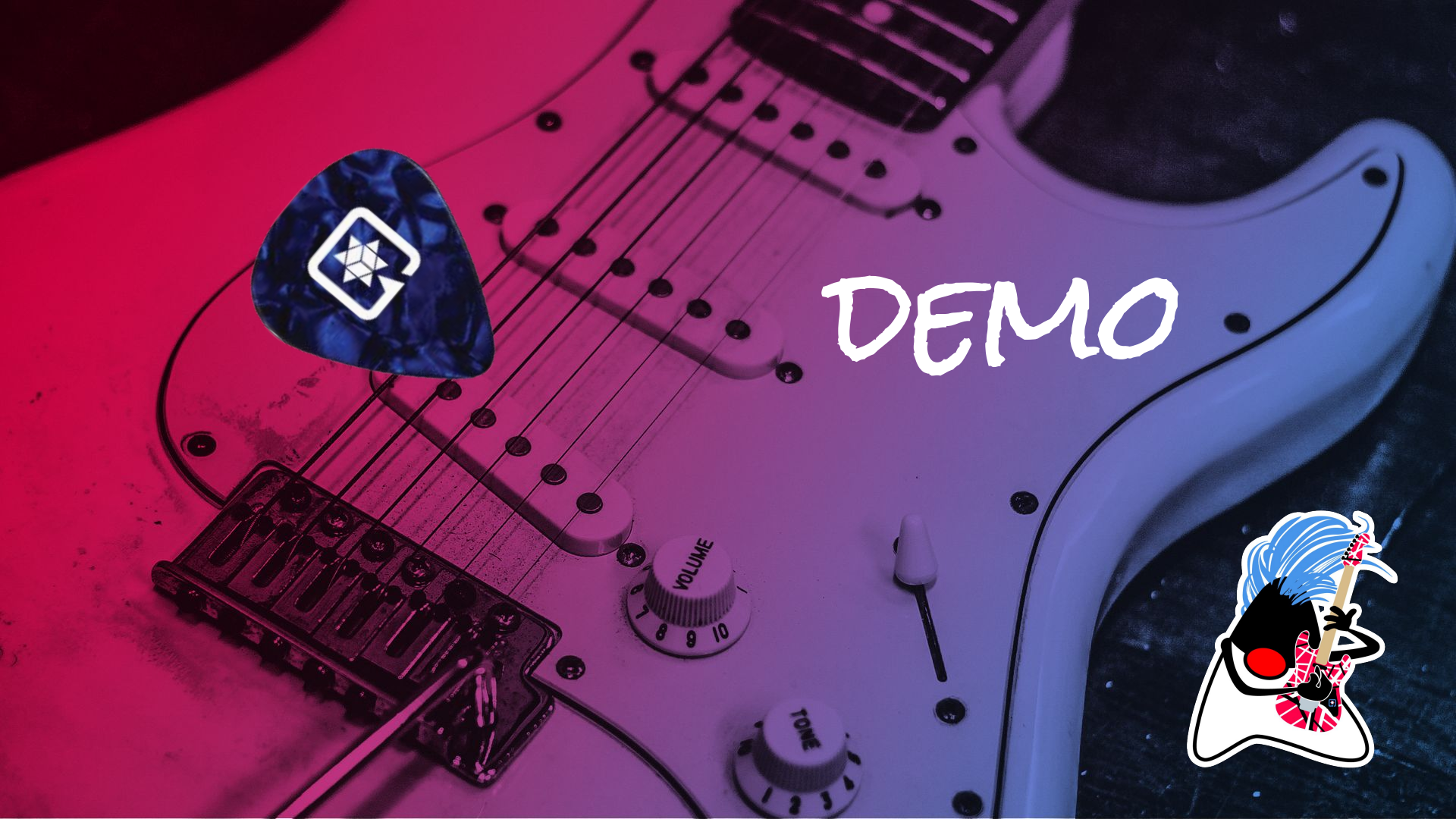
Use Case #3

Redesigned to utilize Knative

- Application needs to be rewritten from Kafka consumer to CloudEvents consumer
- Application needs to be redeployed as Knative Service
- Needs Knative Eventing Kafka Source
- Event-driven autoscaling enabled through Knative Autoscaler



DEMO




Event-driven
How about Serverless Autoscaling?

Integrate KEDA with Knative

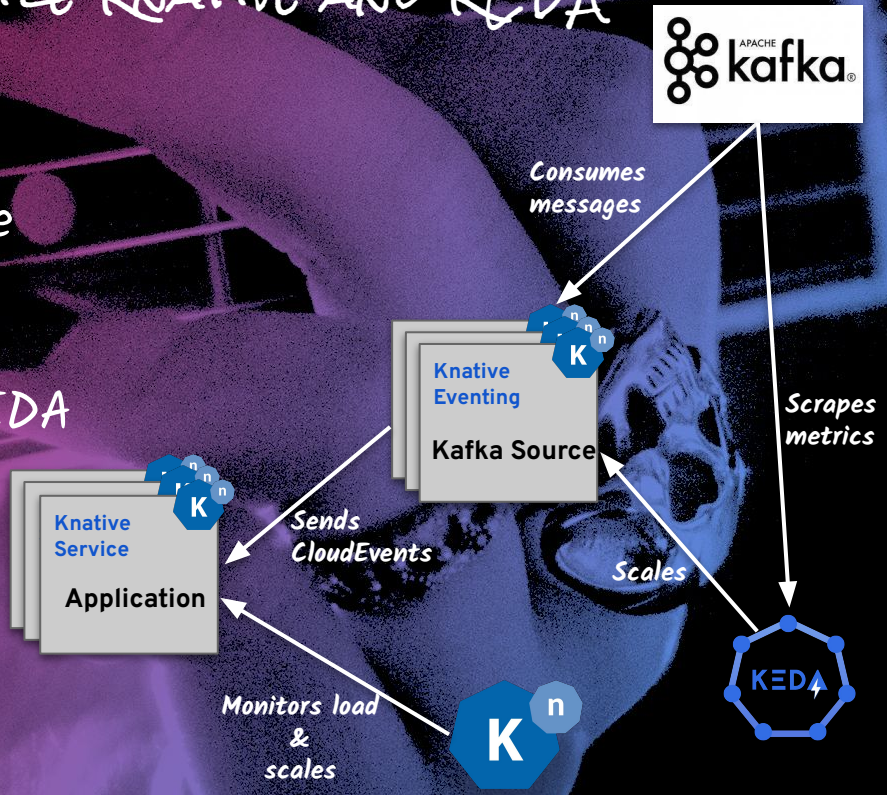
KEDA and Knative Integration

- KEDA can be used to autoscale Knative Eventing Infrastructure
  - Knative Eventing Sources, Channels
  - Autoscaling allows infrastructure to handle higher loads or save resources (by scaling to 0) when idle
- KEDA could be potentially used to scale Knative Service, in case users don't want to utilize Knative Eventing for event driven workloads (currently not implemented)

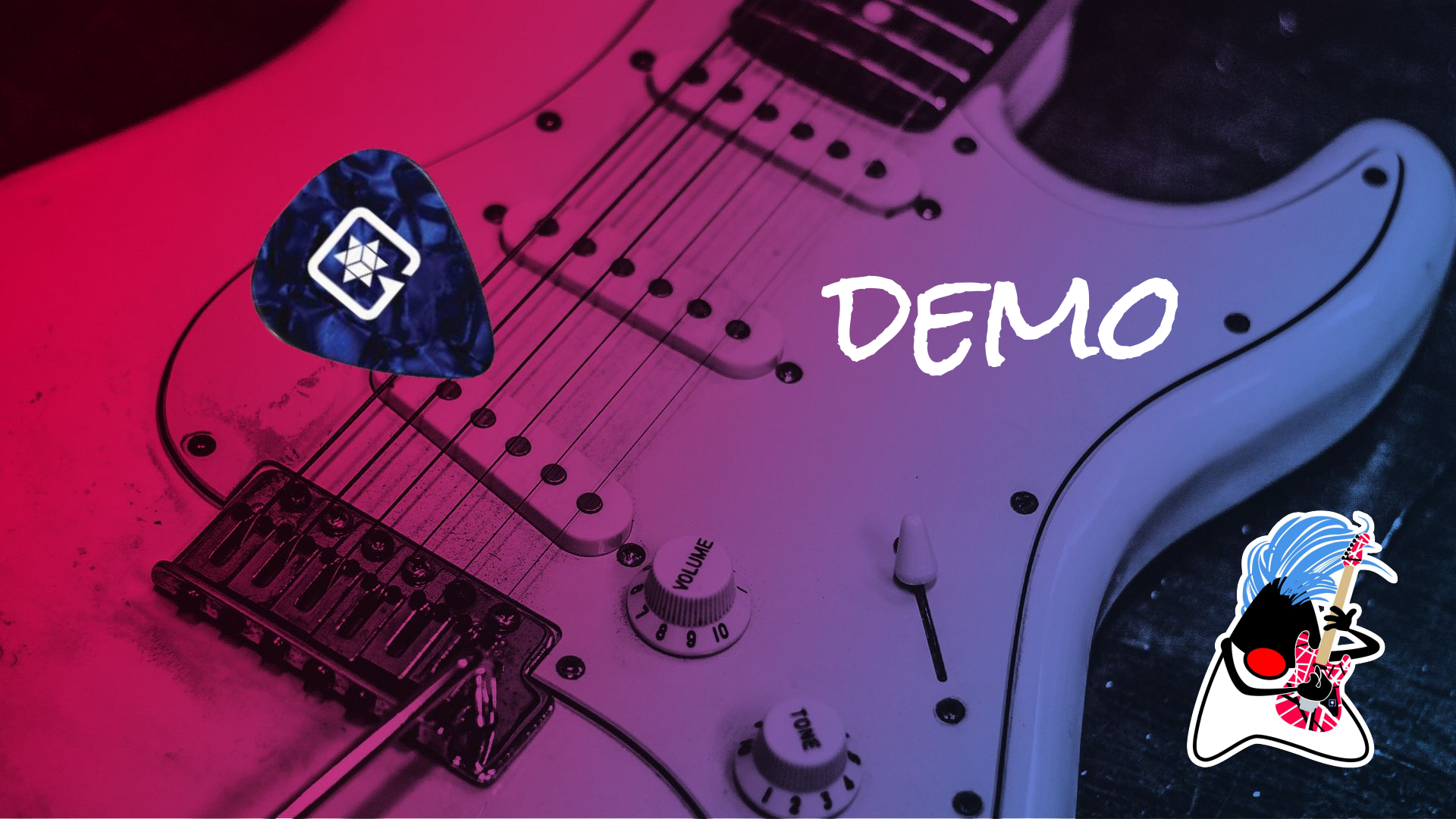
Use Case #4

Redesigned to utilize Knative and KEDA

- Application deployed as Knative Service and autoscaled by Knative
- Knative Eventing Infrastructure - Kafka Source is autoscaled by KEDA



DEMO



knative-sandbox/eventing-autoscaler-keda

- KafkaSource
- AWS SQS Source
- Redis Stream Source
- RabbitMQ Broker

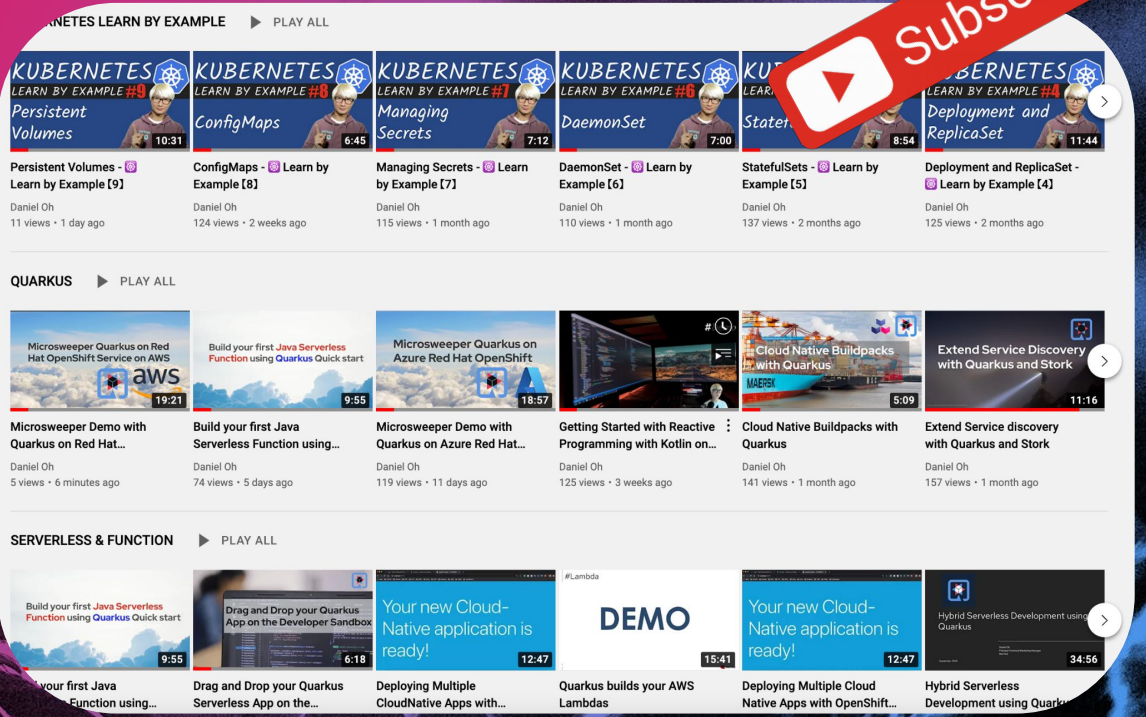
```yaml
metadata:
  annotations:
    autoscaling.knative.dev/class: keda.autoscaling.knative.dev
    autoscaling.knative.dev/minScale: "0"
    autoscaling.knative.dev/maxScale: "5"
    keda.autoscaling.knative.dev/pollingInterval: "30"
    keda.autoscaling.knative.dev/cooldownPeriod: "30"

    # Kafka Source
    keda.autoscaling.knative.dev/kafkaLagThreshold: "10"

    # AWS SQS Source
    keda.autoscaling.knative.dev/awsSqsQueueLength: "5"

    # Redis Stream Source
    keda.autoscaling.knative.dev/redisStreamPendingEntriesCount: "5"
```

https://github.com/knative-sandbox/eventing-autoscaler-keda

Takeaways

- KEDA - Kubernetes event driven autoscaling dead simple
- KEDA - pull model vs. Knative - push model
- KEDA - Standard Kubernetes resources vs. Knative Service
- KEDA can autoscale Knative Eventing Infrastructure
  - knative-sandbox/eventing-autoscaler-keda

bit.ly/danielohtv



Thank you!
Questions?