# **What Is KEDA**

**KEDA** is a [Kubernetes](https://kubernetes.io/)-based Event Driven Autoscaler. With KEDA, you can drive the scaling of any container in Kubernetes based on the number of events needing to be processed.

**KEDA** is a single-purpose and lightweight component that can be added to any Kubernetes cluster. KEDA works alongside standard Kubernetes components like the [Horizontal Pod Autoscaler](https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/) and can extend functionality without overwriting or duplication. In this segment, we will guide you through explicitly mapping the apps you wish to use on an event-driven scale, while allowing other apps to continue functioning with KEDA. KEDA is a flexible and secure option for running alongside Kubernetes applications or frameworks.

## **How KEDA works**

* **Event Sources:**

Define event sources that KEDA can monitor. These sources could be queues, message brokers, databases, or any system that emits events.

* **ScaledObjects:**

Create ScaledObjects in your Kubernetes cluster, specifying the scaling rules and target workloads. These ScaledObjects define how KEDA should respond to events from specific sources.

* **KEDA Controller:**

The KEDA controller runs in your Kubernetes cluster and continuously monitors the configured event sources. When events occur, the controller updates the associated ScaledObjects.

* **Horizontal Pod Autoscaler (HPA):**

KEDA interacts with Kubernetes' native HPA to adjust the number of pods based on the scaling rules defined in the ScaledObjects. This results in dynamic scaling of your application workloads.

### **Key Features of KEDA:**

### **Event-Driven Scaling:**

### KEDA allows you to define custom metrics based on external event sources. When these events occur, KEDA triggers the scaling of your Kubernetes workloads, enabling a more reactive and event-driven approach to autoscaling.

## **Broad Event Source Support:**

## KEDA supports a variety of event sources, including Azure Queue, Apache Kafka, RabbitMQ, AWS CloudWatch, and more. This flexibility allows you to integrate with different event producers and consume events from various systems.

## **Fine-Grained Scaling:**

## With KEDA, you can define scaling rules based on specific metrics associated with your event source. This enables fine-grained control over how your application scales in response to different types of events.

## **Extends Kubernetes HPA:**

## KEDA extends the functionality of Kubernetes' Horizontal Pod Autoscaler. It integrates seamlessly with Kubernetes deployments, stateful sets, and other workload types, leveraging existing Kubernetes concepts and making them easy to adopt.

## **Efficient Resource Utilization:**

## By dynamically adjusting the number of pods based on events, KEDA helps optimize resource utilization. Pods can scale up or down based on demand, ensuring that your application efficiently uses resources during peak and idle times.

## **Use Cases for KEDA:**

## **Microservices Architecture:**

## KEDA is beneficial in microservices architectures where different components may have varying processing demands based on events like message queue depth, HTTP requests, etc.

## **Serverless Architectures:**

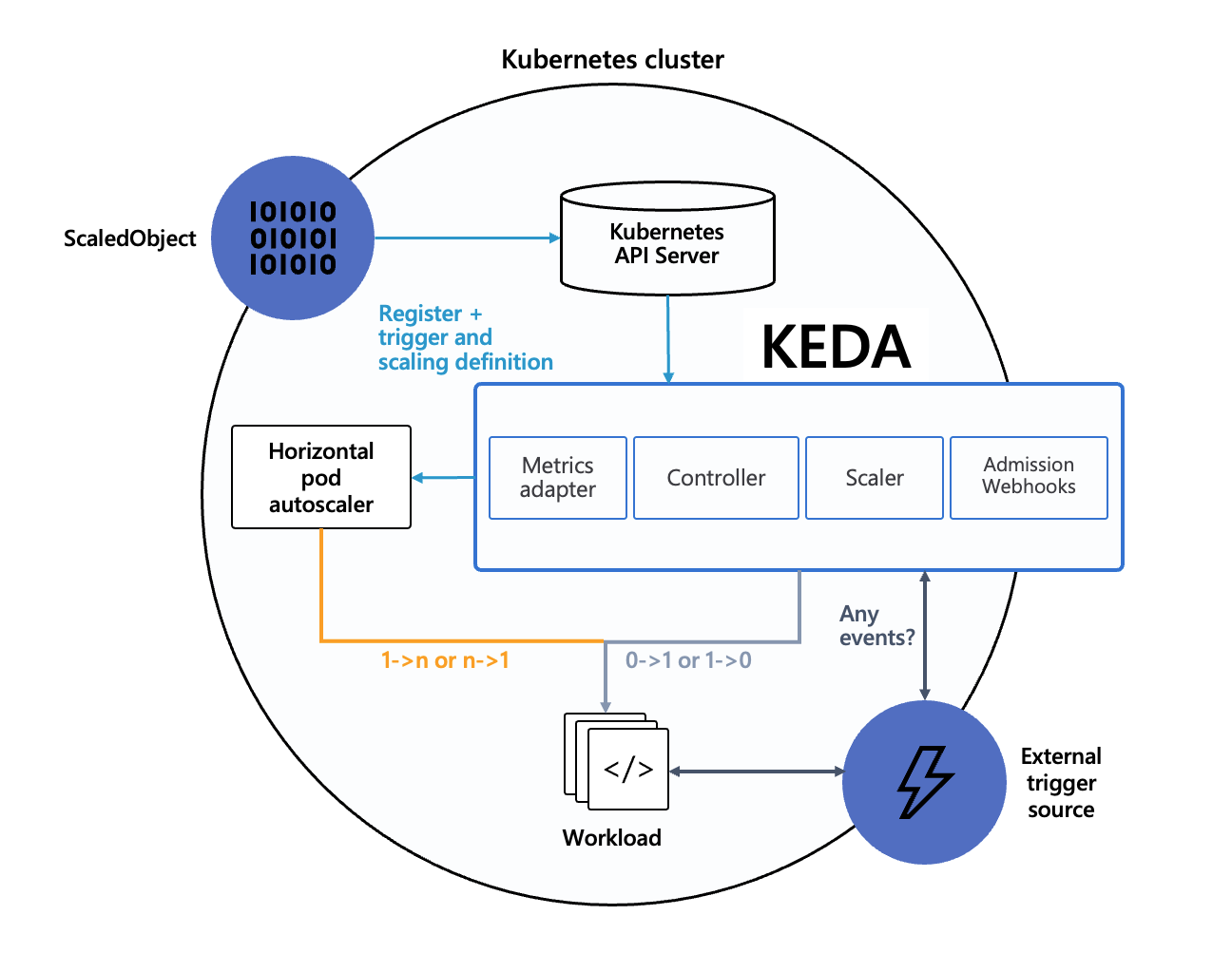
KEDA can be used with serverless frameworks to efficiently scale application components based on incoming events, providing a more cost-effective and responsive solution.

## **Event-Driven Workloads:**

## Applications that heavily rely on event-driven patterns, such as processing messages from queues, benefit from KEDA by ensuring that the number of processing pods aligns with the incoming event rate.

## **Architecture**

The diagram below shows how KEDA works in conjunction with the Kubernetes Horizontal Pod Autoscaler, external event sources, and Kubernetes’ [etcd](https://etcd.io/) data store:



### **Event sources and scalers**

KEDA has a wide range of [**scalers**](https://keda.sh/scalers) that detect whether a deployment should be activated or deactivated and feed custom metrics for a specific event source.

The following scalers are available:

* [**ActiveMQ**](https://keda.sh/docs/2.12/scalers/activemq/)
* [**ActiveMQ Artemis**](https://keda.sh/docs/2.12/scalers/artemis/)
* [**Apache Kafka**](https://keda.sh/docs/2.12/scalers/apache-kafka/)
* [**Apache Kafka (Experimental)**](https://keda.sh/docs/2.12/scalers/apache-kafka-go/)
* [**Apache Pulsar**](https://keda.sh/docs/2.12/scalers/pulsar/)
* [**ArangoDB**](https://keda.sh/docs/2.12/scalers/arangodb/)
* [**AWS CloudWatch**](https://keda.sh/docs/2.12/scalers/aws-cloudwatch/)
* [**AWS DynamoDB**](https://keda.sh/docs/2.12/scalers/aws-dynamodb/)
* [**AWS DynamoDB Streams**](https://keda.sh/docs/2.12/scalers/aws-dynamodb-streams/)
* [**AWS Kinesis Stream**](https://keda.sh/docs/2.12/scalers/aws-kinesis/)
* [**AWS SQS Queue**](https://keda.sh/docs/2.12/scalers/aws-sqs/)
* [**Azure Application Insights**](https://keda.sh/docs/2.12/scalers/azure-app-insights/)
* [**Azure Blob Storage**](https://keda.sh/docs/2.12/scalers/azure-storage-blob/)
* [**Azure Data Explorer**](https://keda.sh/docs/2.12/scalers/azure-data-explorer/)
* [**Azure Event Hubs**](https://keda.sh/docs/2.12/scalers/azure-event-hub/)
* [**Azure Log Analytics**](https://keda.sh/docs/2.12/scalers/azure-log-analytics/)
* [**Azure Monitor**](https://keda.sh/docs/2.12/scalers/azure-monitor/)
* [**Azure Pipelines**](https://keda.sh/docs/2.12/scalers/azure-pipelines/)
* [**Azure Service Bus**](https://keda.sh/docs/2.12/scalers/azure-service-bus/)
* [**Azure Storage Queue**](https://keda.sh/docs/2.12/scalers/azure-storage-queue/)
* [**Cassandra**](https://keda.sh/docs/2.12/scalers/cassandra/)
* [**CouchDB**](https://keda.sh/docs/2.12/scalers/couchdb/)
* [**CPU**](https://keda.sh/docs/2.12/scalers/cpu/)
* [**Cron**](https://keda.sh/docs/2.12/scalers/cron/)
* [**Datadog**](https://keda.sh/docs/2.12/scalers/datadog/)
* [**Elasticsearch**](https://keda.sh/docs/2.12/scalers/elasticsearch/)
* [**Etcd**](https://keda.sh/docs/2.12/scalers/etcd/)
* [**External**](https://keda.sh/docs/2.12/scalers/external/)
* [**External Push**](https://keda.sh/docs/2.12/scalers/external-push/)
* [**Github Runner Scaler**](https://keda.sh/docs/2.12/scalers/github-runner/)
* [**Google Cloud Platform Stackdriver**](https://keda.sh/docs/2.12/scalers/gcp-stackdriver/)
* [**Google Cloud Platform Storage**](https://keda.sh/docs/2.12/scalers/gcp-storage/)
* [**Google Cloud Platform‎ Cloud Tasks**](https://keda.sh/docs/2.12/scalers/gcp-cloud-tasks/)
* [**Google Cloud Platform‎ Pub/Sub**](https://keda.sh/docs/2.12/scalers/gcp-pub-sub/)
* [**Graphite**](https://keda.sh/docs/2.12/scalers/graphite/)
* [**Huawei Cloudeye**](https://keda.sh/docs/2.12/scalers/huawei-cloudeye/)
* [**IBM MQ**](https://keda.sh/docs/2.12/scalers/ibm-mq/)
* [**InfluxDB**](https://keda.sh/docs/2.12/scalers/influxdb/)
* [**Kubernetes Workload**](https://keda.sh/docs/2.12/scalers/kubernetes-workload/)
* [**Liiklus Topic**](https://keda.sh/docs/2.12/scalers/liiklus-topic/)
* [**Loki**](https://keda.sh/docs/2.12/scalers/loki/)
* [**Memory**](https://keda.sh/docs/2.12/scalers/memory/)
* [**Metrics API**](https://keda.sh/docs/2.12/scalers/metrics-api/)
* [**MongoDB**](https://keda.sh/docs/2.12/scalers/mongodb/)
* [**MSSQL**](https://keda.sh/docs/2.12/scalers/mssql/)
* [**MySQL**](https://keda.sh/docs/2.12/scalers/mysql/)
* [**NATS JetStream**](https://keda.sh/docs/2.12/scalers/nats-jetstream/)
* [**NATS Streaming**](https://keda.sh/docs/2.12/scalers/nats-streaming/)
* [**New Relic**](https://keda.sh/docs/2.12/scalers/new-relic/)
* [**OpenStack Metric**](https://keda.sh/docs/2.12/scalers/openstack-metric/)
* [**OpenStack Swift**](https://keda.sh/docs/2.12/scalers/openstack-swift/)
* [**PostgreSQL**](https://keda.sh/docs/2.12/scalers/postgresql/)
* [**Predictkube**](https://keda.sh/docs/2.12/scalers/predictkube/)
* [**Prometheus**](https://keda.sh/docs/2.12/scalers/prometheus/)
* [**RabbitMQ Queue**](https://keda.sh/docs/2.12/scalers/rabbitmq-queue/)
* [**Redis Lists**](https://keda.sh/docs/2.12/scalers/redis-lists/)
* [**Redis Lists (supports Redis Cluster)**](https://keda.sh/docs/2.12/scalers/redis-cluster-lists/)
* [**Redis Lists (supports Redis Sentinel)**](https://keda.sh/docs/2.12/scalers/redis-sentinel-lists/)
* [**Redis Streams**](https://keda.sh/docs/2.12/scalers/redis-streams/)
* [**Redis Streams (supports Redis Cluster)**](https://keda.sh/docs/2.12/scalers/redis-cluster-streams/)
* [**Redis Streams (supports Redis Sentinel)**](https://keda.sh/docs/2.12/scalers/redis-sentinel-streams/)
* [**Selenium Grid Scaler**](https://keda.sh/docs/2.12/scalers/selenium-grid-scaler/)
* [**Solace PubSub+ Event Broker**](https://keda.sh/docs/2.12/scalers/solace-pub-sub/)
* [**Solr**](https://keda.sh/docs/2.12/scalers/solr/)

### **Custom Resources (CRD):**

When you install KEDA, it creates four custom resources:

* scaledobjects.keda.sh
* scaledjobs.keda.sh
* triggerauthentications.keda.sh
* clustertriggerauthentications.keda.sh

These custom resources enable you to map an event source (and the authentication to that event source) to a Deployment, StatefulSet, Custom Resource or Job for scaling.

* ScaledObjects represent the desired mapping between an event source (e.g. Rabbit MQ) and the Kubernetes Deployment, StatefulSet or any Custom Resource that defines /scale subresource.
* ScaledJobs represent the mapping between the event source and Kubernetes Job.
* ScaledObject/ScaledJob may also reference a TriggerAuthentication or ClusterTriggerAuthentication, which contains the authentication configuration or secrets used to monitor the event source.

**Deploying KEDA With Helm**

### **Install**

**Deploying KEDA with Helm is very simple:**

* **Add Helm repo**

**helm repo add kedacore** [**https://kedacore.github.io/charts**](https://kedacore.github.io/charts)

* **Update Helm repo**

**helm repo update**

* **Install the Keda Helm chart**

**helm install keda kedacore/keda --namespace keda --create-namespace**

* **To deploy the CRDs separately from the Helm chart, use the keda-2.xx.x-crds.yaml file provided on the** [**GitHub releases**](https://github.com/kedacore/keda/releases) **page.**