Autoscaling

What is Autoscaling?

In computing; autoscaling is when systems without manual intervention adjust capacity (eg. amount of CPU, Ram) to meet the demand (traffic from users) by adding or removing resources commonly triggered by events.

Pod-Based Scaling

Horizontal Pod Scaling (*HorizontalPodAutoscaler*)
Add more pods to meet the demand

Vertical Pod Scaling (*VerticalPodAutoscaler*)
Right-size pods for the optimal CPU and memory resources

Node-based scaling

Cluster Auto Scaling *(Cluster Autoscaler or Karpenter)*
Add or remove Nodes (compute servers) based on demand

Cluster API

Declarative APIs and tooling to simplify provisioning, upgrading, and operating multiple Kubernetes clusters. Cluster API can be extended to support any infrastructure (AWS, Azure, vSphere, etc.), bootstrap or control plane (kubeadm is built-in) provider.

KubeCTL Scale vs Autoscale

The **scale** command it used to:

- update the amount of replicas in the state of deployment object
- perform a deploy

```
kubectl scale --replicas=3 deploy/my-app
```

The **autoscale** command is used to create a **HorizontalPodAutoscaler**

```
kubectl autoscale rc foo --min=1 --max=5 --cpu-percent=80
```

KEDA

Kubernetes Event-driven Autoscaling (KEDA) allows you scale based on event data.

KEDA has a wide range of scalers

- ActiveMQ
- Apache Kafka
- AWS CloudWatch
- AWS Kinesis Stream
- AWS SQS Queue
- Azure Application Insights
- Azure Blob Storage
- Azure Event Hubs
- Azure Log Analytics
- Azure Monitor
- Azure Pipelines
- Azure Service Bus
- Azure Storage Queue
- Cassandra
- CPU
- Cron
- Datadog
- Elasticsearch
- External
- External Push
- GCP Pub/Sub
- Graphite
- Hauwei Cloudeye
- IBM MQ
- InfluxDB
- Kubernetes Workload
- Liklus Topic
- Memory
- Metrics API
- MongoDB
- MSSQL
- MySQL
- NATS Streaming

New Relic
OpenStack Metric
OpenStack Swift
PostgreSQL
Predictkube
Prometheus
RabbitMQ Queue
Redis Lists
Redis Streams
Selenium Grid Scalars
Solace PubSub + Event Broker