Autoscaling Container Apps

Memi Lavi
www.memilavi.com

Autoscaling Container Apps

- Container apps can be scaled automatically
- Scaling is per revision
- Done by adding or removing replicas for the revision
- Ensures there's enough compute power to handle incoming traffic
- Scaling can be configured based on various parameters
- Can scale to zero – no cost

Autoscaling Components

- Scaling is a combination of:

Limits

Rules

Behaviors

Scaling Limits

- Set the minimum and maximum number of replicas per revision

Limit	Default value	Min value	Max value
Min replicas / revision	0	0	300
Max replicas / revision	10	1	300

Scaling Rules

- Define the trigger for the scaling
- Three categories of triggers:

HTTP

- Sets threshold of concurrent HTTP requests
- When threshold is reached a new replica is added
- Not supported with Container Apps jobs

TCP

- Sets threshold of concurrent TCP connections
- When threshold is reached a new replica is added
- Not supported with Container Apps jobs
- Configured using Azure CLI or ARM

Custom

- KEDA-scaler based
- Queries other services for triggering scaling
- Example: Scaling is based on messages in Service Bus
- Portal support for Storage Queue scaling

Scaling Behavior

- Combines limits and rules
- Sets how rules are evaluated and how scale is executed
  - ie. Polling interval, scale up steps etc.
- Usually should not be modified

Configuring Scaling

- Done using the portal, CLI or ARM template
- Adding scaling configuration creates a new revision

Scaling with KEDA

- Stands for Kubernetes Event-Driven Autoscaler
- Scales Kubernetes deployments based on events collected from various event sources
- Remember: Container Apps are based on Kubernetes
- Not related to Azure, integrated into Container Apps

KEDA Concepts

Event Source

- External source monitored by KEDA
  - Examples: Azure Storage Queue, ActiveMQ, Memory, etc.
- Provides metrics for KEDA to process
- Metrics are sent to Scaler

KEDA Concepts

Scalers

- Receive metrics from event source
- Detects whether a deployment should be scaled
- Currently there are more than 60 types of scalers

KEDA Concepts

ScaledObject / ScaledJob

- Specification for describing how KEDA should scale deployments and which trigger to use

```yaml
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: kafka-scaledobject
  namespace: default
spec:
  scaleTargetRef:
    name: azure-functions-deployment
  pollingInterval: 30
  triggers:
  - type: kafka
    metadata:
      bootstrapServers: localhost:9092
      consumerGroup: my-group       # Make sure that this consumer group name
      topic: test-topic
      # Optional
      lagThreshold: "50"
      offsetResetPolicy: latest
```

Using KEDA with Container Apps

- KEDA scalers can be used in Container Apps
- Using Custom scale rules
- Specifications are entered as metadata fields
- Authentication to resources is done using secrets
  - ie. Connection strings