white duck

Scaling in the AKS Wonderland

Festive Tech Calendar 2023

Microsoft
Solutions Partner

Digital & App Innovation
Data & AI
Azure

Specialist
Migrate Enterprise Applications
to Microsoft Azure

Who we are

Philip Welz
(Senior DevOps & Kubernetes Engineer, Azure MVP)

+49 8031 230159-0

philip.welz@whiteduck.de

@philip_welz

www.linkedin.com/in/philip-welz

Nico Meisenzahl
(Head of DevOps Consulting & Operations, Cloud Solution Architect)

+49 8031 230159-0

nico.meisenzahl@whiteduck.de

@nmeisenzahl

www.linkedin.com/in/nicomeisenzahl

Agenda

- Kubernetes Scheduler
- Horizontal Pod Autoscaler (HPA)
- Vertical Pod Autoscaler (VPA)
- Kubernetes Event-driven Autoscaling (KEDA)
- Kubernetes Cluster Autoscaler (CA)
- Karpenter

Kubernetes Scheduler

- is one of the main components of the control plane, responsible for managing pod scheduling
- is responsible for resource allocation within a cluster
- schedulers are swappable (currently not with AKS)
- you can also have multiple (currently not with AKS)
- "kube-scheduler" is the default one
  - customizable and extendable scheduling in a two-step process (filtering & storing)
  - does not reschedule!
- https://kubernetes.io/docs/concepts/scheduling-eviction

Scheduling details

- resource quotas
- taints and tolerations
- selectors and affinity (node & inter-pod affinity and anti-affinity)
- pod distribution across zones
- disk dependencies/limitations

Horizontal Pod Autoscaler (HPA)

- scales the number of pod replicas based on CPU, memory or custom metrics
  - also supports multiple metrics
- relies on the metrics server (available by default with AKS)
- works with every scalable resource (Deployment, StatefulSet, …)
- supports scaling based on container metrics since 1.27

Vertical Pod Autoscaler (VPA)

- automatically adjusts the resource requests and limits of containers based on their actual usage
- helps increasing cluster efficiency
  - optimizing resource allocation, reducing resource waste
- consists out of “Recommender”, “Updater” and an Admission controller
- GA on AKS since Sep 2023
- tips & best practices
  - do use with pod disruption budget
  - don‘t use with HPA (based on CPU & memory metrics) or JVM-based workloads
  - don’t use in big clusters (scaling issues)

Kubernetes Event-driven Autoscaling (KEDA)

- scales based on events instead of predefined metrics
- supports
  - Deployments, StatefulSets and custom resources
  - 60+ scalers (and bring-your-own scalers)
  - scaling to/from zero
- integrates with HPA
- GA on AKS since Nov 2023
- examples: scales up from zero on
  - a message in a queue (ServiceBus, Redis, …)
  - a blob upload in a Storage Container
  - any Azure event (Event Hub)
  - a HTTP Get call or CI/CD pipeline queue

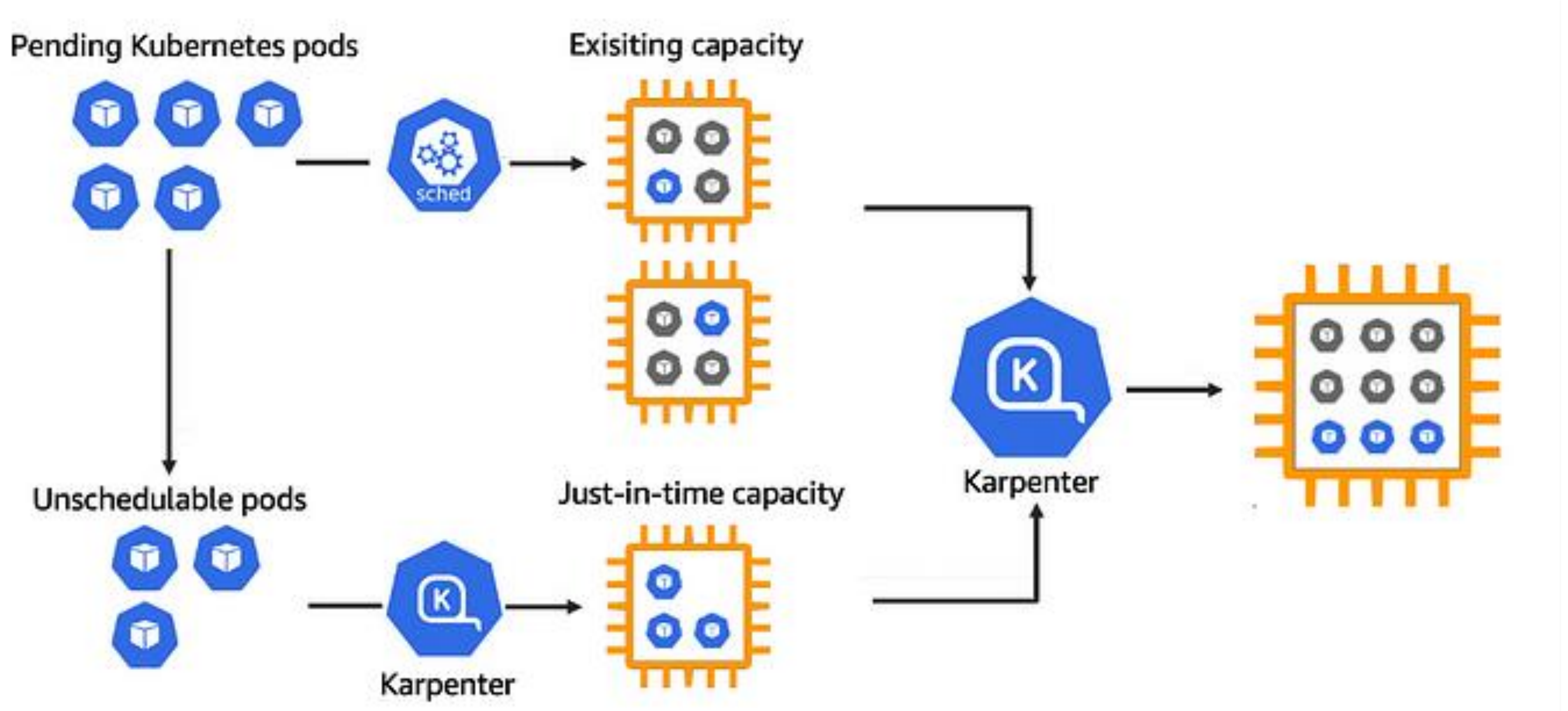
Kubernetes Cluster Autoscaler (CA)

- automatically resizes a cluster based on demand by scaling nodes up/down
- is key when using HPA, VPA, KEDA or any other autoscaler
- acts on
  - pods that failed to run in the cluster due to insufficient resources (pod pending)
  - nodes in the cluster that are underutilized, and pods can be rescheduled
- scales VMMS to and from zero on AKS

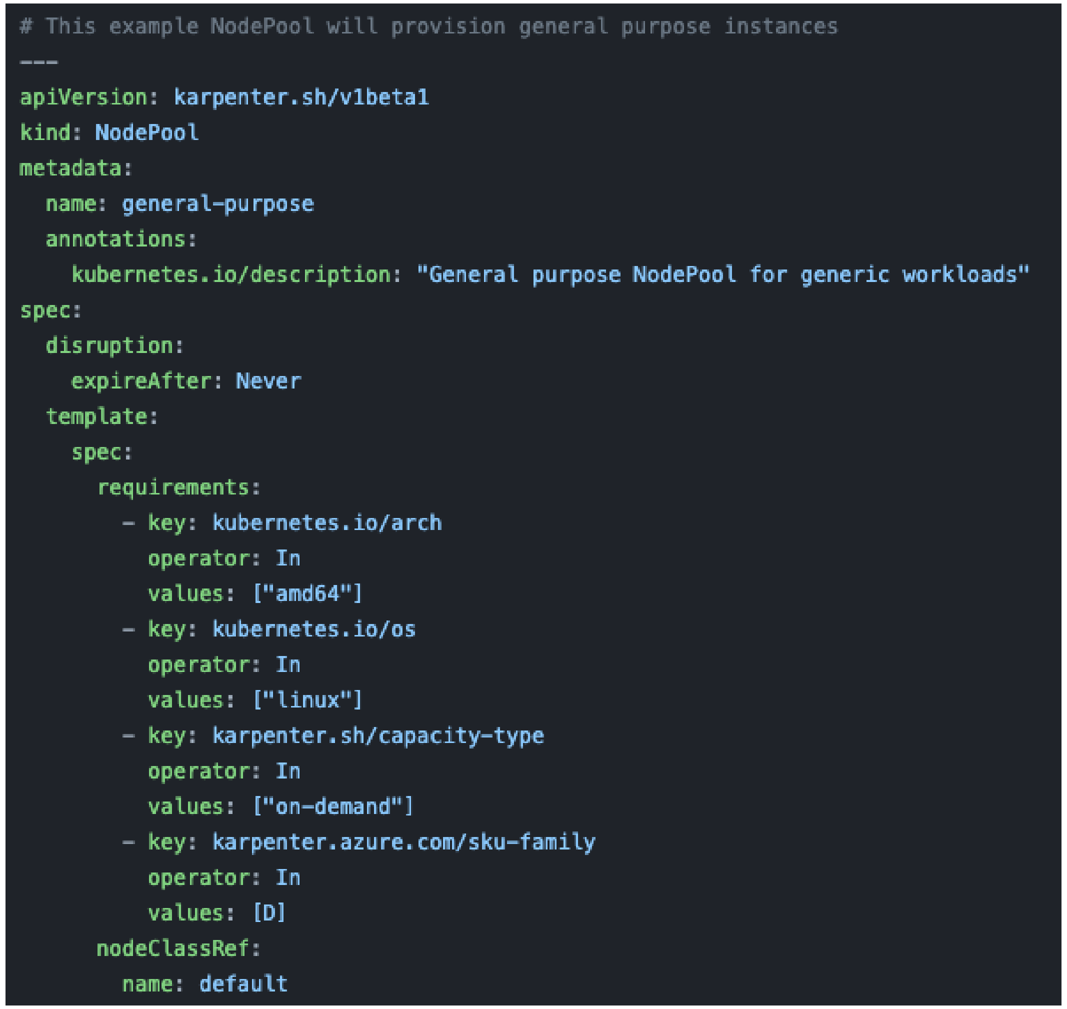
Karpenter

- offers a more granular approach than traditional cluster autoscalers
- scaling lifecycle
  - <u>watching</u> for pods that the scheduler has marked as unschedulable
  - <u>evaluating</u> scheduling constraints (requests, selectors, affinities, tolerations, …)
  - <u>provisioning</u> nodes that meet the requirements of the pods
    - does not rely on AKS node pools / VMMS
  - <u>removing</u> the nodes when the nodes are no longer needed
    - also reschedules pods
- available (alpha) on AKS since Nov 23 – early stage
  - no AKS-related docs
  - no node-upgrade support
  - …

Karpenter


Pending Kubernetes pods
sched
Exisiting capacity
Unschedulable pods
Karpenter
Just-in-time capacity
Karpenter

Karpenter



```yaml
# This example NodePool will provision general purpose instances
---
apiVersion: karpenter.sh/v1beta1
kind: NodePool
metadata:
  name: general-purpose
  annotations:
    kubernetes.io/description: "General purpose NodePool for generic workloads"
spec:
  disruption:
    expireAfter: Never
  template:
    spec:
      requirements:
        - key: kubernetes.io/arch
          operator: In
          values: ["amd64"]
        - key: kubernetes.io/os
          operator: In
          values: ["linux"]
        - key: karpenter.sh/capacity-type
          operator: In
          values: ["on-demand"]
        - key: karpenter.azure.com/sku-family
          operator: In
          values: [D]
      nodeClassRef:
        name: default
```

Getting started

- Slides & Demos
  - https://github.com/whiteducksoftware/aks-scaling-examples

Questions?

Philip Welz
(Senior DevOps & Kubernetes Engineer, Azure MVP)

+49 8031 230159-0
philip.welz@whiteduck.de
@philip_welz
www.linkedin.com/in/philip-welz

Nico Meisenzahl
(Head of DevOps Consulting & Operations, Cloud Solution Architect)

+49 8031 230159-0
nico.meisenzahl@whiteduck.de
@nmeisenzahl
www.linkedin.com/in/nicomeisenzahl

white duck

Thank you!