About node pools

STANDARD (/KUBERNETES-ENGINE/DOCS/CONCEPTS/TYPES-OF-CLUSTERS)

This page explains how node pools work in Google Kubernetes Engine (GKE).

To learn how to manage node pools, see Adding and managing node pools (/kubernetes-engine/docs/how-to/node-pools).

Overview

A *node pool* is a group of nodes (/kubernetes-engine/docs/concepts/cluster-architecture#nodes) within a cluster that all have the same configuration. Node pools use a NodeConfig (/kubernetes-engine/docs/reference/rest/v1/NodeConfig) specification. Each node in the pool has a Kubernetes node label, `cloud.google.com/gke-nodepool`, which has the node pool's name as its value.

When you create a cluster (/kubernetes-engine/docs/concepts/cluster-architecture), the number of nodes and type of nodes that you specify are used to create the first node pool of the cluster. Then, you can add additional node pools of different sizes and types to your cluster. All nodes in any given node pool are identical to one another.

For example, you might create a node pool in your cluster with local SSDs (/kubernetes-engine/docs/concepts/local-ssd), a minimum CPU platform (/kubernetes-engine/docs/how-to/min-cpu-platform), Spot VMs (/kubernetes-engine/docs/concepts/spot-vms), a specific node image (/kubernetes-engine/docs/concepts/node-images), different machine types (/compute/docs/machine-types), or a more efficient virtual network interface (/kubernetes-engine/docs/how-to/using-gvnic).

Custom node pools are useful when you need to schedule Pods that require more resources than others, such as more memory or more local disk space. If you need more control of where Pods are scheduled, you can use node taints (/kubernetes-engine/docs/how-to/node-taints).

You can create, upgrade, and delete node pools (/kubernetes-engine/docs/how-to/node-pools) individually without affecting the whole cluster. You cannot configure a single node in a node pool; any configuration changes affect all nodes in the node pool.

You can resize node pools in a cluster by adding or removing nodes (/kubernetes-engine/docs/how-to/node-pools#resizing_a_node_pool).

By default, all new node pools run the same version of Kubernetes as the control plane. Existing node pools can be manually upgraded (/kubernetes-engine/docs/how-to/upgrading-a-container-cluster#manually_upgrading_your_nodes) or automatically upgraded (/kubernetes-engine/docs/concepts/node-auto-upgrade). You can also run multiple Kubernetes node versions on each node pool in your cluster, update each node pool independently, and target different node pools for specific deployments.

Deploying Services to specific node pools

When you define a Service (https://kubernetes.io/docs/concepts/services-networking/service/), you can indirectly control which node pool it is deployed into. The node pool is *not* dependent on the configuration of the Service, but on the configuration of the Pod (/kubernetes-engine/docs/concepts/pod).

- You can explicitly deploy a Pod to a specific node pool by setting a `nodeSelector` (https://kubernetes.io/docs/concepts/configuration/assign-pod-node/) in the Pod manifest. This forces a Pod to run only on nodes in that node pool. For an example see, Deploying a Pod to a specific node pool (/kubernetes-engine/docs/how-to/node-pools#deploy).
- You can specify resource requests for the containers (https://kubernetes.io/docs/concepts/configuration/manage-compute-resources-container/). The Pod only runs on nodes that satisfy the resource requests. For instance, if the Pod definition includes a container that requires four CPUs, the Service does not select Pods running on nodes with two CPUs.

Nodes in multi-zonal or regional clusters

If you created a multi-zonal (/kubernetes-engine/docs/concepts/types-of-clusters#multi-zonal_clusters) or regional (/kubernetes-engine/docs/concepts/regional-clusters) cluster, all of the node pools are replicated to those zones automatically. Any new node pool is automatically created in those zones. Similarly, any deletions delete those node pools from the additional zones as well.

Note that because of this multiplicative effect, this may consume more of your project's quota (/kubernetes-engine/quotas) for a specific region when creating node pools.

Deleting node pools

When you delete a node pool (/kubernetes-engine/docs/how-to/node-pools#deleting_a_node_pool), GKE drains all the nodes in the node pool. The draining process involves GKE evicting Pods on each node in the node pool. Each node in a node pool is drained by evicting Pods with an allotted graceful termination period of `MAX_POD`. `MAX_POD` is the maximum `terminationGracePeriodSeconds` (https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/#pod-termination) set on the Pods scheduled on the node, with a cap of one hour. `PodDisruptionBudget` (https://kubernetes.io/docs/tasks/run-application/configure-pdb/) settings are not honored during node pool deletion.

However, when a cluster is deleted, GKE does not follow this process of gracefully terminating the nodes by draining them. If the workloads running on a cluster must be gracefully terminated, use `kubectl drain` (https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#drain) to clean up the workloads before you delete the cluster.

To delete a node pool, see Delete a node pool (/kubernetes-engine/docs/how-to/node-pools#deleting_a_node_pool).

What's next

- Learn about the cluster architecture in GKE (/kubernetes-engine/docs/concepts/cluster-architecture).
- Learn how to add and manage node pools (/kubernetes-engine/docs/how-to/node-pools).
- Learn how to auto-provision nodes (/kubernetes-engine/docs/how-to/node-auto-provisioning).
- Learn about node taints, restrictions for Pods running in a given node pool (/kubernetes-engine/docs/how-to/node-taints).

Last updated 2023-01-24 UTC.