Use preemptible VMs to run fault-tolerant workloads

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

This page shows you how to use preemptible VMs in Google Kubernetes Engine (GKE).

Note: Spot VMs are the latest version of preemptible VMs. We recommend that you use either Spot Pods in Autopilot clusters (/kubernetes-engine/docs/how-to/autopilot-spot-pods) or Spot VMs in Standard clusters (/kubernetes-engine/docs/how-to/spot-vms) instead of preemptible VMs. GKE continues to support preemptible VMs, which now use the same pricing model as Spot VMs.

Overview

Preemptible VMs are Compute Engine VM instances (/compute/docs/instances/preemptible) that are priced lower than standard VMs and provide no guarantee of availability. Preemptible VMs offer similar functionality to Spot VMs (/compute/docs/instances/spot), but only last up to 24 hours after creation.

In some cases, a preemptible VM might last longer than 24 hours. This can occur when the new Compute Engine instance comes up too fast and Kubernetes doesn't recognize that a different Compute Engine VM was created. The underlying Compute Engine instance will have a maximum duration of 24 hours and follow the expected preemptible VM behavior (/compute/docs/instances/preemptible).

Note: Preemptible VMs share the same pricing model as Spot VMs. For more information, see Compute Engine Pricing (/compute/all-pricing#compute-engine-pricing).

Comparison to Spot VMs

Preemptible VMs share many similarities with Spot VMs, including the following:

- Terminated when Compute Engine requires the resources to run standard VMs.
- Useful for running stateless, batch, or fault-tolerant workloads.
- Lower pricing than standard VMs.

- On clusters running GKE version 1.20 and later, graceful node shutdown (#graceful-shutdown) is enabled by default.
- Works with the cluster autoscaler (/kubernetes-engine/docs/concepts/cluster-autoscaler) and node auto-provisioning (/kubernetes-engine/docs/how-to/node-auto-provisioning).

In contrast to Spot VMs, which have no maximum expiration time, preemptible VMs only last for up to 24 hours after creation.

You can enable preemptible VMs on new clusters and node pools, use `nodeSelector` or node affinity to control scheduling, and use taints and tolerations to avoid issues with system workloads when nodes are preempted.

Termination and graceful shutdown of preemptible VMs

When Compute Engine needs to reclaim the resources used by preemptible VMs, a preemption notice (/compute/docs/instances/preemptible#preemption) is sent to GKE. Preemptible VMs terminate 30 seconds after receiving a termination notice.

By default, clusters use graceful node shutdown (https://kubernetes.io/docs/concepts/cluster-administration/node-shutdown/#graceful-node-shutdown)
. The kubelet notices the termination notice and gracefully terminates Pods that are running on the node. If the Pods are part of a Deployment, the controller creates and schedules new Pods to replace the terminated Pods.

On a best-effort basis, the kubelet grants a graceful termination period of 15 seconds for non-system Pods, after which system Pods (with the `system-cluster-critical` or `system-node-critical` priorityClasses) have 15 seconds to gracefully terminate.

Note: Adjusting the values of **`terminationGracePeriodSeconds`**, **`shutdownGracePeriodCriticalPods`**, or **`shutdownGracePeriod`** in your Pod spec to more than the granted 15 seconds has no effect on the graceful termination of nodes that use preemptible VMs or Spot VMs.

During graceful node termination, the kubelet updates the status of the Pods, assigning a `Failed` phase and a `Terminated` reason to the terminated Pods.

Note: For clusters running version 1.27.2-gke.1800 or later, GKE automatically deletes Pods that were evicted during node termination. Use the following instructions to manually delete terminated Pods for earlier GKE versions.

When the number of terminated Pods reaches a threshold of 1000 for clusters with fewer than 100 nodes or 5000 for clusters with 100 nodes or more, garbage collection (https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/#pod-garbage-collection) cleans up the Pods.

You can also delete terminated Pods manually using the following commands:

```
kubectl get pods --all-namespaces | grep -i NodeShutdown | awk '{print $1, $
kubectl get pods --all-namespaces | grep -i Terminated | awk '{print $1, $2}
```

Modifications to Kubernetes behavior

Using preemptible VMs on GKE modifies some guarantees and constraints that Kubernetes provides, such as the following:

- GKE shuts down preemptible VMs without a grace period for Pods, 30 seconds after receiving a preemption notice from Compute Engine.
- Reclamation of preemptible VMs is involuntary and is not covered by the guarantees of `PodDisruptionBudgets` (https://kubernetes.io/docs/tasks/run-application/configure-pdb/). You might experience greater unavailability than your configured **`PodDisruptionBudget`**.

Limitations

- The kubelet graceful node shutdown feature (https://kubernetes.io/docs/concepts/architecture/nodes/#graceful-node-shutdown) is only enabled on clusters running GKE version 1.20 and later. For GKE versions prior to 1.20, you can use the Kubernetes on GCP Node Termination Event Handler (https://github.com/GoogleCloudPlatform/k8s-node-termination-handler) to gracefully terminate your Pods when preemptible VMs are terminated.
- Preemptible VMs do not support Windows Server node pools.

Create a cluster or node pool with preemptible VMs

You can use the Google Cloud CLI to create a cluster or node pool with preemptible VMs.

To create a cluster with preemptible VMs, run the following command:

```
gcloud container clusters create CLUSTER_NAME \
    --preemptible
```

Replace *CLUSTER_NAME* with the name of your new cluster.

To create a node pool with preemptible VMs, run the following command:

```
gcloud container node-pools create POOL_NAME \
    --cluster=CLUSTER_NAME \
    --preemptible
```

Replace *POOL_NAME* with the name of your new node pool.

Note: If you use the Google Cloud console to duplicate an existing cluster that uses preemptible VMs, you must enable Spot VMs for each node pool and configure your workloads to use Spot VMs instead of preemptible VMs. For instructions, refer to Use Spot VMs to run fault-tolerant workloads (/kubernetes-engine/docs/how-to/spot-vms).

Use nodeSelector to schedule Pods on preemptible VMs

GKE adds the `cloud.google.com/gke-preemptible=true` and `cloud.google.com/gke-provisioning=preemptible` (for nodes running GKE version 1.25.5-gke.2500 or later) labels to nodes that use preemptible VMs. You can use a `nodeSelector` in your deployments to tell GKE to schedule Pods onto preemptible VMs.

For example, the following Deployment filters for preemptible VMs using the `cloud.google.com/gke-preemptible` label:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello-app
  template:
    metadata:
      labels:
        app: hello-app
    spec:
      containers:
      - name: hello-app
        image: us-docker.pkg.dev/google-samples/containers/gke/hello-app:1.0
        resources:
          requests:
            cpu: 200m
      nodeSelector:
        cloud.google.com/gke-preemptible: "true"
```

Use node taints for preemptible VMs

You can taint nodes that use preemptible VMs so that GKE can only place Pods with the corresponding toleration on those nodes.

To add a node taint to a node pool that uses preemptible VMs, use the `--node-taints` flag when creating the node pool, similar to the following command:

```
gcloud container node-pools create POOL2_NAME \
    --cluster=CLUSTER_NAME \
    --node-taints=cloud.google.com/gke-preemptible="true":NoSchedule
```

Now, only Pods that tolerate the node taint are scheduled to the node.

To add the relevant toleration to your Pods, modify your deployments and add the following to your Pod specification:

```
tolerations:
- key: cloud.google.com/gke-preemptible
  operator: Equal
  value: "true"
  effect: NoSchedule
```

Node taints for GPU preemptible VMs

Preemptible VMs support using GPUs (/kubernetes-engine/docs/how-to/gpus). You should create at least one other node pool in your cluster that doesn't use preemptible VMs before adding a GPU node pool that uses preemptible VMs. Having a standard node pool ensures that GKE can safely place system components like DNS.

If you create a new cluster with GPU node pools that use preemptible VMs, or if you add a new GPU node pool that uses preemptible VMs to a cluster that does not already have a standard node pool, GKE does not automatically add the `nvidia.com/gpu=present:NoSchedule` taint to the nodes. GKE might schedule system Pods onto the preemptible VMs, which can lead to disruptions. This behavior also increases your resource consumption, because GPU nodes are more expensive than non-GPU nodes.

What's next

- Learn how to run a GKE application on Spot VMs with on-demand nodes as fallback (/blog/topics/developers-practitioners/running-gke-application-spot-nodes-demand-nodes-fallback) .
- Learn more about Spot VMs in GKE (/kubernetes-engine/docs/concepts/spot-vms).
- Learn about taints and tolerations (/kubernetes-engine/docs/how-to/node-taints).

Last updated 2024-09-10 UTC.