Running workloads with GKE Sandbox

Beta

This product or feature is in a pre-release state and might change or have limited support. For more information, see the product launch stages (https://cloud.google.com/products/#product-launch-stages).

This page describes how to use GKE Sandbox (https://cloud.google.com/kubernetes-engine/docs/concepts/sandbox-pods) to protect the host kernel on your nodes when containers in the Pod execute unknown or untrusted code, or need extra isolation from the node.

Enabling GKE Sandbox

You can enable GKE Sandbox on a new cluster or an existing cluster.

Before you begin

To prepare for this task, perform the following steps:

- Ensure that you have enabled the Google Kubernetes Engine API.

  ENABLE GOOGLE KUBERNETES ENGINE API (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/APIS/LIBRA

- Ensure that you have installed the Cloud SDK (https://cloud.google.com/sdk/downloads).
- Set your default project ID (https://support.google.com/cloud/answer/6158840):

```
gcloud config set project [PROJECT_ID]
```

- If you are working with zonal clusters, set your default compute zone (https://cloud.google.com/compute/docs/zones#available):

```
gcloud config set compute/zone [COMPUTE_ZONE]
```

- If you are working with regional clusters, set your default compute region (https://cloud.google.com/compute/docs/zones#available):

```
gcloud config set compute/region [COMPUTE_REGION]
```

- Update `gcloud` to the latest version:

```
gcloud components update
```

★ **Note:** You can override these default settings in `gcloud` commands using the `--project`, `--zone`, and `--region` operational flags.

- GKE Sandbox requires GKE v1.12.7-gke.17 or higher, or v1.13.5-gke.15 or higher, for the cluster master and nodes.
- Ensure that the `gcloud` command is version 243.0.0 or higher.

On a new cluster

To enable GKE Sandbox, you configure a node pool. The default node pool (the first node pool in your cluster, created when the cluster is created) cannot use GKE Sandbox. To enable GKE Sandbox during cluster creation, you must add a second node pool when you create the cluster.

CONSOLE GCLOUD

To view your clusters, visit the Google Kubernetes Engine menu in Cloud Console.

1. Visit the Google Kubernetes Engine menu in Cloud Console.

   VISIT THE GOOGLE KUBERNETES ENGINE MENU (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/KUBEF

2. Click **Create cluster**.
3. Choose the **Standard cluster** template or choose an appropriate template (#cluster-templates) for your workload.
4. **Optional but recommended**: Enable Stackdriver Logging and Stackdriver Monitoring, so that gVisor messages are logged.
5. Click **Add node pool**.
6. Configure the node pool according to your requirements. Click **More node pool options** for the node pool. Configure these settings:

- For the node version, select v1.12.6-gke.8 or higher.
- For the node image, select **Container-Optimized OS with Containerd (cos_containerd) (beta)**.
- Enable **Enable sandbox with gVisor (beta)**.
- If the nodes in the node pool use more than a single vCPU, click **Add Label**. Set the key to `cloud.google.com/gke-smt-disabled` and the value to `true`. Next, follow the instructions for disabling Hyper-Threading in the security bulletin (https://cloud.google.com/kubernetes-engine/docs/security-bulletins#may-14-2019).

Configure other node pool settings as required.

7. Save the node pool settings and continue configuring your cluster.

The `gvisor` RuntimeClass is instantiated during node creation, before any workloads are scheduled onto the node. You can check for the existence of the `gvisor` RuntimeClass using the following command:

```
kubectl get runtimeclasses
```

```
NAME     AGE
gvisor   19s
```

Note: For clusters running versions prior to v1.14.3, you also need to wait for the `gvisor-admission-webhook-config` to be instantiated on the node. To check, use the following command:

```
kubectl get mutatingwebhookconfiguration gvisor-admission-webhook-config
```

```
NAME                              CREATED AT
gvisor-admission-webhook-config   2019-04-19T20:52:25Z
```

On an existing cluster

You can enable GKE Sandbox on an existing cluster by adding a new node pool and enabling the feature for that node pool, or by modifying an existing non-default node pool.

CONSOLE GCLOUD

1. Visit the Google Kubernetes Engine menu in Cloud Console.

VISIT THE GOOGLE KUBERNETES ENGINE MENU (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/KUBEF

2. Click the cluster's Edit button, which looks like a pencil.
3. If necessary, add an additional node pool by clicking **Add node pool**. To edit an existing node pool, click the node pool's Edit button. Do not enable **Sandbox with gVisor (beta)** on the default node pool.
4. Enable **Sandbox with gVisor (beta)**, then click **Done**.
5. If necessary, make additional configuration changes to the cluster, then click **Save**.

The `gvisor` RuntimeClass is instantiated during node creation, before any workloads are scheduled onto the node. You can check for the existence of the `gvisor` RuntimeClass using the following command:

```
kubectl get runtimeclasses
```

```
NAME     AGE
gvisor   19s
```

Note: For clusters running versions prior to v1.14.3, you also need to wait for the `gvisor-admission-webhook-config` to be instantiated on the node. To check, use the following command:

```
kubectl get mutatingwebhookconfiguration gvisor-admission-webhook-config
```

```
NAME                              CREATED AT
gvisor-admission-webhook-config   2019-04-19T20:52:25Z
```

Optional: Enable Stackdriver Logging and Stackdriver Monitoring

It is optional but recommended that you enable Stackdriver Logging and Stackdriver Monitoring on the cluster, so that gVisor messages are logged. You must use Google Cloud Console to enable these features on an existing cluster.

1. Visit the Google Kubernetes Engine menu in Cloud Console.

   VISIT THE GOOGLE KUBERNETES ENGINE MENU (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/KUBERN

2. Click the cluster's Edit button, which looks like a pencil.

3. Enable Stackdriver Logging and Stackdriver Monitoring.
4. If necessary, make additional configuration changes to the cluster, then click **Save**.

Working with GKE Sandbox

Running an application in a sandbox

To force a Deployment to run on a node with GKE Sandbox enabled, set its `spec.template.spec.runtimeClassName` to `gvisor`, as shown by this manifest for a Deployment:

```
# httpd.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: httpd
  labels:
    app: httpd
spec:
  replicas: 1
  selector:
    matchLabels:
      app: httpd
  template:
    metadata:
      labels:
        app: httpd
    spec:
      runtimeClassName: gvisor
      containers:
      - name: httpd
        image: httpd
```

To create the Deployment, use the `kubectl create` command:

```
kubectl create -f httpd.yaml
```

The Pod is deployed to a node in a node pool with GKE Sandbox enabled. To verify this, use the `kubectl get pods` command to find the node where the Pod is deployed:

```
kubectl get pods
```

```
NAME                     READY   STATUS    RESTARTS   AGE
httpd-db5899bc9-dk7lk    1/1     Running   0          24s
```

Find the name of the Pod in the output, then run the following command to check its value for RuntimeClass:

```
kubectl get pods [NAME-OF-POD] -o jsonpath='{.spec.runtimeClassName}'
```

```
gvisor
```

Alternatively, you can list the RuntimeClass of each Pod, and look for the ones where it is set to `gvisor`:

```
kubectl get pods -o jsonpath=$'{range .items[*]}{.metadata.name}: {.spec.runtimeClas
```

```
[NAME-OF-POD]: gvisor
```

This method of verifying that the Pod is running in a sandbox is trustworthy because it does not rely on any data within the sandbox itself. Anything reported from within the sandbox is untrustworthy, because it could be defective or malicious.

Running a regular Pod along with sandboxed Pods

After enabling GKE Sandbox on a node pool, you can run trusted applications on those nodes without using a sandbox by using node taints and tolerations. These Pods are referred to as "regular Pods" to distinguish them from sandboxed Pods.

Regular Pods, just like sandboxed Pods, are prevented from accessing other Google Cloud services or cluster metadata. This prevention is part of the node's configuration. If your regular Pods or sandboxed Pods require access to Google Cloud services, use Workload Identity (https://cloud.google.com/kubernetes-engine/docs/how-to/workload-identity).

Running untrusted code on the same nodes as critical system services comes with potential risk, even if the untrusted code is running in a sandbox. Consider these risks when designing your applications.

GKE Sandbox adds the following label and taint to nodes that can run sandboxed Pods:

```
labels:
  sandbox.gke.io: gvisor
```

```
taints:
- effect: NoSchedule
  key: sandbox.gke.io
  value: gvisor
```

In addition to any node affinity and toleration settings in your Pod manifest, GKE Sandbox applies the following node affinity and toleration to all Pods with `RuntimeClass` set to `gvisor`:

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: sandbox.gke.io/runtime
          operator: In
          values:
          - gvisor
```

```
tolerations:
  - effect: NoSchedule
    key: sandbox.gke.io/runtime
    operator: Equal
    value: gvisor
```

To schedule a regular Pod on a node with GKE Sandbox enabled, manually apply the node affinity and toleration above in your Pod manifest.

- If your pod **can** run on nodes with GKE Sandbox enabled, add the toleration.
- If your pod **must** run on nodes with GKE Sandbox enabled, add both the node affinity and toleration.

For example, the following manifest modifies the manifest used in Running an application in a sandbox (#sandboxed-application) so that it runs as a regular Pod on a node with sandboxed Pods, by removing the runtimeClass and adding both the taint and toleration above.

```
# httpd-no-sandbox.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: httpd-no-sandbox
  labels:
    app: httpd
spec:
  replicas: 1
  selector:
    matchLabels:
      app: httpd
  template:
    metadata:
      labels:
        app: httpd
    spec:
      containers:
      - name: httpd
        image: httpd
      affinity:
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
            - matchExpressions:
              - key: sandbox.gke.io/runtime
                operator: In
                values:
                - gvisor
      tolerations:
        - effect: NoSchedule
          key: sandbox.gke.io/runtime
          operator: Equal
          value: gvisor
```

First, verify that the Deployment is not running in a sandbox:

```
kubectl get pods -o jsonpath=$'{range .items[*]}{.metadata.name}: {.spec.runtimeClas
```

```
httpd-db5899bc9-dk7lk: gvisor
httpd-no-sandbox-5bf87996c6-cfmmd:
```

The `httpd` Deployment created earlier is running in a sandbox, because its runtimeClass is `gvisor`. The `httpd-no-sandbox` Deployment has no value for runtimeClass, so it is not running in a sandbox.

Next, verify that the non-sandboxed Deployment is running on a node with GKE Sandbox by running the following command:

```
kubectl get pod -o jsonpath=$'{range .items[*]}{.metadata.name}: {.spec.nodeName}\n{
```

The name of the node pool is embedded in the value of `nodeName`. Verify that the Pod is running on a node in a node pool with GKE Sandbox enabled.

Note: If the regular Pod is unschedulable, verify that the taint and/or toleration is set correctly in the Pod manifest.

Verifying metadata protection

To validate the assertion that metadata is protected from nodes that can run sandboxed Pods, you can run a test:

1. Create a sandboxed Deployment from the following manifest, using `kubectl apply -f`. It uses the `fedora` image, which includes the `curl` command. The Pod runs the `/bin/sleep` command to ensure that the Deployment runs for 10000 seconds.

```
# sandbox-metadata-test.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: fedora
  labels:
    app: fedora
spec:
  replicas: 1
  selector:
    matchLabels:
      app: fedora
  template:
    metadata:
      labels:
```

```
        app: fedora
    spec:
      runtimeClassName: gvisor
      containers:
      - name: fedora
        image: fedora
        command: ["/bin/sleep","10000"]
```

2. Get the name of the Pod using `kubectl get pods`, then use `kubectl exec` to connect to the Pod interactively.

```
kubectl exec -it [POD-NAME] /bin/sh
```

   You are connected to a container running in the Pod, in a `/bin/sh` session.

3. Within the interactive session, attempt to access a URL that returns cluster metadata:

```
curl -s "http://metadata.google.internal/computeMetadata/v1/instance/attributes
```

   The command hangs and eventually times out, because the packets are silently dropped.

4. Press **Ctrl+C** to terminate the `curl` command, and type `exit` to disconnect from the Pod.

5. Remove the `RuntimeClass` line from the YAML manifest and redeploy the Pod using `kubectl apply -f [FILENAME]`. The sandboxed Pod is terminated and recreated on a node without GKE Sandbox.

6. Get the new Pod name, connect to it using `kubectl exec`, and run the `curl` command again. This time, results are returned. This example output is truncated.

```
ALLOCATE_NODE_CIDRS: "true"
API_SERVER_TEST_LOG_LEVEL: --v=3
AUTOSCALER_ENV_VARS: kube_reserved=cpu=60m,memory=960Mi,ephemeral-storage=41Gi;
...
```

   Type `exit` to disconnect from the Pod.

7. Remove the deployment:

```
kubectl delete deployment fedora
```

Disabling GKE Sandbox

It isn't currently possible to update a node pool to disable GKE Sandbox. To disable GKE Sandbox on an existing node pool, you can do **one** of the following:

- Delete the previously-sandboxed Pods. Otherwise, after you disable GKE Sandbox, those Pods run as regular Pods if no available nodes have GKE Sandbox enabled. Then delete the node pool (https://cloud.google.com/kubernetes-engine/docs/how-to/node-pools#deleting_a_node_pool) where GKE Sandbox was enabled, **or**
- Resize the node pool to zero nodes (https://cloud.google.com/kubernetes-engine/docs/how-to/node-pools#resizing_a_node_pool), **or**
- Recreate the Pods without specifying a value for the RuntimeClassName.

What's next

- Learn more about managing node pools (https://cloud.google.com/kubernetes-engine/docs/how-to/node-pools).
- Read the security overview (https://cloud.google.com/kubernetes-engine/docs/concepts/security-overview).

Last updated December 4, 2019.