

Quickstart: Deploy an app to a GKE cluster

Deploy an app to a GKE cluster

AUTOPILOT (HTTPS://CLOUD.GOOGLE.COM/KUBERNETES-ENGINE/DOCS/CONCEPTS/AUTOPILOT-OVERVIEW?AUTHUSER=5)

STANDARD (HTTPS://CLOUD.GOOGLE.COM/KUBERNETES-ENGINE/DOCS/CONCEPTS/TYPES-OF-CLUSTERS?AUTHUSER=5)

In this quickstart, you deploy a simple web server containerized application (<https://cloud.google.com/kubernetes-engine/docs/concepts/kubernetes-engine-overview?authuser=5#workloads>) to a Google Kubernetes Engine (GKE) cluster. You will learn how to create a cluster, and how to deploy the application to the cluster so that it can be accessed by users.

This quickstart assumes a basic understanding of Kubernetes (<https://kubernetes.io>).

Before you begin

Take the following steps to enable the Kubernetes Engine API:

1. In the Google Cloud console, on the project selector page, select or create a Google Cloud project

(<https://cloud.google.com/resource-manager/docs/creating-managing-projects?authuser=5>).

★ **Note:** If you don't plan to keep the resources that you create in this procedure, create a project instead of selecting an existing project. After you finish these steps, you can delete the project, removing all resources associated with the project.

Go to project selector (<https://console.cloud.google.com/projectselector2/home/dashboard?authuser=5>)

2. Make sure that billing is enabled for your Google Cloud project (<https://cloud.google.com/billing/docs/how-to/verify-billing-enabled?authuser=5#console>).

3. Enable the Artifact Registry and Google Kubernetes Engine APIs.

Enable the APIs (<https://console.cloud.google.com/flows/enableapi?apiid=artifactregistry.google>)

CLOUD SHELL

Terminal



Open Editor



In this tutorial you will use Cloud Shell (<https://cloud.google.com/shell/docs?authuser=5>), which is a shell environment for managing resources hosted on Google Cloud.

Cloud Shell comes preinstalled with the Google Cloud CLI (<https://cloud.google.com/sdk/gcloud?authuser=5>) and kubectl (<https://kubernetes.io/docs/reference/kubectl/>) command-line tool. The gcloud CLI provides the primary command-line interface for Google Cloud, and kubectl provides the primary command-line interface for running commands against Kubernetes clusters.

Launch Cloud Shell:


1. Go to the Google Cloud console.

Google Cloud console (<https://console.cloud.google.com/?authuser=5>)

2. From the upper-right corner of the console, click the **Activate Cloud Shell** button:



A Cloud Shell session opens inside a frame lower on the console. You use this shell to run `gcloud` and `kubectl` commands. Before you run commands, set your default project in the Google Cloud CLI using the following command:

```
gcloud config set project PROJECT_ID 
```

Replace *PROJECT_ID* with your project ID

(<https://support.google.com/cloud/answer/6158840?authuser=5>).

Create a GKE cluster

A cluster

(<https://cloud.google.com/kubernetes-engine/docs/concepts/cluster-architecture?authuser=5>) consists of at least one *cluster control plane* machine and multiple worker machines called *nodes*. Nodes are Compute Engine virtual machine (VM) instances (<https://cloud.google.com/compute/docs/instances?authuser=5>) that run the Kubernetes processes necessary to make them part of the cluster. You deploy applications to clusters, and the applications run on the nodes.

Create an Autopilot cluster named `hello-cluster`:

```
gcloud container clusters create-auto hello-cluster \
  --location=us-central1
```

Note: It might take several minutes to finish creating the cluster.

Get authentication credentials for the cluster

After creating your cluster, you need to get authentication credentials to interact with the cluster:

```
gcloud container clusters get-credentials hello-cluster \
  --location us-central1
```

This command configures `kubectl` to use the cluster you created.

Deploy an application to the cluster

Now that you have created a cluster, you can deploy a containerized application (<https://cloud.google.com/kubernetes-engine/docs/concepts/kubernetes-engine-overview?authuser=5#workloads>)

to it. For this quickstart, you can deploy our example web application, `hello-app`.

GKE uses Kubernetes objects to create and manage your cluster's resources. Kubernetes provides the Deployment (<https://kubernetes.io/docs/concepts/workloads/controllers/deployment/>) object for deploying stateless applications like web servers. Service (<https://cloud.google.com/kubernetes-engine/docs/concepts/service?authuser=5>) objects define rules and load balancing for accessing your application from the internet.

Create the Deployment

To run `hello-app` in your cluster, you need to deploy the application by running the following command:

```
kubectl create deployment hello-app
```

This Kubernetes command, `kubectl create deployment`

(<https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#create>), creates a Deployment named `hello-server`. The Deployment's `Pod` (<https://kubernetes.io/docs/concepts/workloads/pods>) runs the `hello-app` container image.

In this command:

- `--image` specifies a container image to deploy. In this case, the command pulls the example image from an [Artifact Registry](https://cloud.google.com/artifact-registry/docs?authuser=5) (<https://cloud.google.com/artifact-registry/docs?authuser=5>) repository, `us-docker.pkg.dev/google-samples/containers/gke/hello-app:1.0` indicates the specific image version to pull. If you don't specify a version, the image with the [default tag](https://cloud.google.com/architecture/best-practices-for-building-containers?authuser=5#properly_tag_your_images) (https://cloud.google.com/architecture/best-practices-for-building-containers?authuser=5#properly_tag_your_images) `latest` is used.

Expose the Deployment

After deploying the application, you need to expose it to the internet so that users can access it. You can expose your application by creating a Service, a Kubernetes resource that exposes your application to external traffic.

To expose your application, run the following `kubectl expose`

(<https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#expose>) command:

```
kubectl expose deployment hello-server \
  --type LoadBalancer \
  --port 80 \
  --target-port 8080
```

Passing in the `--type LoadBalancer` flag creates a Compute Engine load balancer for your container. The `--port` flag initializes public port 80 to the internet and the `--target-port` flag routes the traffic to port 8080 of the application.

Load balancers are billed per Compute Engine's [load balancer pricing](https://cloud.google.com/compute/pricing?authuser=5#lb)

(<https://cloud.google.com/compute/pricing?authuser=5#lb>).

1. Inspect the running Pods by using `kubectl get pods`

(<https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#get>):

```
kubectl get pods
```

You should see one `hello-server` Pod running on your cluster.

2. Inspect the `hello-server` Service by using `kubectl get service`

(<https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#get>):

```
kubectl get service hello-server
```

From this command's output, copy the Service's external IP address from the `EXTERNAL-IP` column.

★ **Note:** You might need to wait several minutes before the Service's external IP address populates. If the application's external IP is `<pending>`, run `kubectl get` again.

3. View the application from your web browser by using the external IP address with the exposed port:

```
http://EXTERNAL_IP
```

You have just deployed a containerized web application to GKE.

Clean up

To avoid incurring charges to your Google Cloud account for the resources used on this page, follow these steps.

1. Delete the application's Service by running `kubectl delete`

(<https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#delete>):

```
kubectl delete service hello
```

This command deletes the Compute Engine load balancer that you created when you exposed the Deployment.

2. Delete your cluster by running `gcloud container clusters delete`

(<https://cloud.google.com/sdk/gcloud/reference/container/clusters/delete?authuser=5>):

```
gcloud container clusters delete hello-cluster \
  --location us-central1
```

Optional: hello-app code review

`hello-app` is a simple web server application that consists of two files: `main.go` and a `Dockerfile`.

`hello-app` is packaged as a [Docker](https://docker.com/) (<https://docker.com/>) container image. Container images are stored in any Docker image registry, such as Artifact Registry. We host `hello-app` in a Artifact Registry repository at `us-docker.pkg.dev/google-samples/containers/gke/hello-app`.

[main.go](#)
`Dockerfile` (#dockerfile)
(#main.go)

`main.go` is a web server implementation written in the [Go programming language](https://golang.org) (<https://golang.org>). The server responds to any HTTP request with a "Hello, world!" message.

[quickstarts/hello-app/main.go](#)

(<https://github.com/GoogleCloudPlatform/kubernetes-engine-samples/blob/HEAD/quickstarts/hello-app/main.go>)

<https://github.com/GoogleCloudPlatform/kubernetes-engine-samples/blob/HEAD/quickstarts/hello-app/main.go>

```
package main

import (
    "fmt"
    "log"
    "net/http"
    "os"
```

```
func main() {
    // register hello function to handle all requests
    mux := http.NewServeMux()
    mux.HandleFunc("/", hello)

    // use PORT environment variable, or default to 8080
    port := os.Getenv("PORT")
    if port == "" {
        port = "8080"
    }

    // start the web server on port and accept requests
    log.Printf("Server listening on port %s", port)
    log.Fatal(http.ListenAndServe(":"+port, mux))
}

// hello responds to the request with a plain-text "Hello, world" message
func hello(w http.ResponseWriter, r *http.Request) {
    log.Printf("Serving request: %s", r.URL.Path)
    host, _ := os.Hostname()
    fmt.Fprintf(w, "Hello, world!\n")
    fmt.Fprintf(w, "Version: 1.0.0\n")
    fmt.Fprintf(w, "Hostname: %s\n", host)
}
```

What's next

- Learn more about [creating clusters](https://cloud.google.com/kubernetes-engine/docs/how-to/creating-a-container-cluster?authuser=5)
(<https://cloud.google.com/kubernetes-engine/docs/how-to/creating-a-container-cluster?authuser=5>)
- Learn more about [Kubernetes](http://kubernetes.io/) (<http://kubernetes.io/>).
- Read the [kubectl](https://kubernetes.io/docs/reference/kubectl/) reference documentation
(<https://kubernetes.io/docs/reference/kubectl/>).
- Learn how to [package, host, and deploy a simple web server application](https://cloud.google.com/kubernetes-engine/docs/tutorials/hello-app?authuser=5)
(<https://cloud.google.com/kubernetes-engine/docs/tutorials/hello-app?authuser=5>).
- [Create a Guestbook application with Redis and PHP](https://cloud.google.com/kubernetes-engine/docs/tutorials/guestbook?authuser=5)
(<https://cloud.google.com/kubernetes-engine/docs/tutorials/guestbook?authuser=5>).

- [Deploy WordPress on GKE with Persistent Disks and Cloud SQL](https://cloud.google.com/kubernetes-engine/docs/tutorials/persistent-disk?authuser=5) (<https://cloud.google.com/kubernetes-engine/docs/tutorials/persistent-disk?authuser=5>).
- [Deploy a Kubernetes application with Cloud Code for VS Code](https://cloud.google.com/code/docs/vscode/quickstart-local-dev?authuser=5) (<https://cloud.google.com/code/docs/vscode/quickstart-local-dev?authuser=5>) or [Cloud Code for IntelliJ](https://cloud.google.com/code/docs/intellij/deploy-kubernetes-app?authuser=5) (<https://cloud.google.com/code/docs/intellij/deploy-kubernetes-app?authuser=5>).

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (<https://www.apache.org/licenses/LICENSE-2.0>). For details, see the [Google Developers Site Policies](https://developers.google.com/site-policies?authuser=5) (<https://developers.google.com/site-policies?authuser=5>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2023-11-20 UTC.