

[Python](https://cloud.google.com/python/) (<https://cloud.google.com/python/>) [Guides](#)

Running the Python Bookshelf app on Google Kubernetes Engine

This tutorial shows how to run the [Python Bookshelf sample app](#) (<https://cloud.google.com/python/getting-started/tutorial-app>) on [Google Kubernetes Engine \(GKE\)](#) (<https://cloud.google.com/kubernetes-engine>). Follow this tutorial to containerize and deploy an existing Python web app to GKE. We recommend that you work through the Bookshelf app documentation as part of the tutorial for the [App Engine standard environment](#) (<https://cloud.google.com/python/getting-started/tutorial-app>).

Objectives

- Create a GKE cluster.
- Containerize a Python app.
- Create a replicated frontend for the Bookshelf app.
- Create a replicated backend for the Bookshelf app.
- Create a load-balanced service to route HTTP traffic to the Bookshelf app frontend.

Costs

This tutorial uses the following billable components of Google Cloud:

- [GKE](https://cloud.google.com/kubernetes-engine/pricing) (<https://cloud.google.com/kubernetes-engine/pricing>)
- [Compute Engine](https://cloud.google.com/compute/pricing) (<https://cloud.google.com/compute/pricing>)

- [Cloud Storage](https://cloud.google.com/storage/pricing) (https://cloud.google.com/storage/pricing)
- [Datastore](https://cloud.google.com/datastore/pricing) (https://cloud.google.com/datastore/pricing)
- [Pub/Sub](https://cloud.google.com/pubsub/pricing) (https://cloud.google.com/pubsub/pricing)

To generate a cost estimate based on your projected usage, use the [pricing calculator](https://cloud.google.com/products/calculator) (https://cloud.google.com/products/calculator). New Google Cloud users might be eligible for a [free trial](https://cloud.google.com/free-trial) (https://cloud.google.com/free-trial).

When you finish this tutorial, you can avoid continued billing by deleting the resources you created. For more information, see [Cleaning up](#) (#clean-up).

Before you begin

1. [Sign in](https://accounts.google.com/Login) (https://accounts.google.com/Login) to your Google Account.

If you don't already have one, [sign up for a new account](https://accounts.google.com/SignUp) (https://accounts.google.com/SignUp).

2. In the Cloud Console, on the project selector page, select or create a Google Cloud project.

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 THE PROJECT SELECTOR PAGE](https://console.cloud.google.com/projectselector) (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/PROJECTSELECTOR)

3. Make sure that billing is enabled for your Google Cloud project. [Learn how to confirm billing is enabled for your project](https://cloud.google.com/billing/docs/how-to/modify-project) (https://cloud.google.com/billing/docs/how-to/modify-project).
4. Enable the Datastore, GKE, Cloud Storage, and Pub/Sub APIs.

[ENABLE THE APIS](https://console.cloud.google.com/flows/enableapi?apiid=datastore) (HTTPS://CONSOLE.CLOUD.GOOGLE.COM/FLOWS/ENABLEAPI?APIID=DATASTORE)

5. [Install and initialize the Cloud SDK](https://cloud.google.com/sdk/docs/) (https://cloud.google.com/sdk/docs/).
6. Install [Docker](https://www.docker.com/) (https://www.docker.com/). Docker builds container images locally.
7. Install `kubectl`.

```
gcloud components install kubectl
```



Creating a GKE cluster

A GKE cluster is a managed set of Compute Engine virtual machines (VMs) that operate as a single GKE cluster. This tutorial needs a cluster with a minimum of two nodes, and these nodes need access to all Google APIs.

1. Create the cluster. Replace `[YOUR_GCP_ZONE]` with the [Google Cloud zone](#) (<https://cloud.google.com/compute/docs/regions-zones/#available>) where you want to host your cluster.

```
gcloud container clusters create bookshelf \  
  --scopes "cloud-platform" \  
  --num-nodes 2 \  
  --enable-basic-auth \  
  --issue-client-certificate \  
  --enable-ip-alias \  
  --zone [YOUR_GCP_ZONE]
```

2. Verify that you have access to the cluster. The following command lists the nodes in your container cluster and indicates that your container cluster is running and that you have access to it.

```
kubectl get nodes
```

You use the `kubectl` command to create resources in a GKE cluster. To learn more about `kubectl`, see [GKE cluster operations](#)

(<https://cloud.google.com/kubernetes-engine/docs/clusters/operations>). In general, you use `gcloud` to manage resources in your Google Cloud project, and you use `kubectl` to manage resources in your GKE cluster. A single project can have multiple clusters, which lets you have clusters made up of different machine types to satisfy different needs.

When you create a cluster with `gcloud`, authentication is set up automatically for `kubectl`. If you use the [Google Cloud Console](#) (<https://console.cloud.google.com/kubernetes>) to create clusters, you can set up authentication by using the `gcloud container clusters get-credentials` [command](#) (<https://cloud.google.com/sdk/gcloud/reference/container/clusters/get-credentials>).

Cloning the sample app

The sample app is available on GitHub at [GoogleCloudPlatform/getting-started-python](https://github.com/GoogleCloudPlatform/getting-started-python) (<https://github.com/GoogleCloudPlatform/getting-started-python>).

1. Clone the repository:

```
git clone https://github.com/GoogleCloudPlatform/getting-started-python.git
```

2. Go to the sample directory:

```
cd getting-started-python/optional-kubernetes-engine
```

3. Check out the `git` branch:

```
git checkout steps
```

Initializing Datastore

The Bookshelf app uses [Datastore](https://cloud.google.com/datastore) (<https://cloud.google.com/datastore>) to store book data. To initialize Datastore in your project for the first time, complete the following steps:

1. In the Cloud Console, open the **Datastore** page.

[GO TO THE DATASTORE PAGE \(HTTPS://CONSOLE.CLOUD.GOOGLE.COM/DATASTORE\)](https://console.cloud.google.com/datastore)

2. Select a region for your datastore.
3. Click **Continue** until you reach the **Create an Entity** page, and then close the window. The Bookshelf app is ready to create entities in Datastore.

Creating a Cloud Storage bucket

The Bookshelf app uses [Cloud Storage](https://cloud.google.com/storage) (<https://cloud.google.com/storage>) to store image files.

The following instructions detail how to create a Cloud Storage bucket. Buckets are the basic containers that hold your data in Cloud Storage.

Note: You can choose [any name](https://cloud.google.com/storage/docs/naming) (<https://cloud.google.com/storage/docs/naming>) for your Cloud Storage bucket. It's a good practice to give your bucket the same name as your project ID. Bucket names must be

unique across all of Google Cloud, so it's possible that you can't use your project ID as the bucket name.

1. In your terminal window, create a Cloud Storage bucket, where ***YOUR_BUCKET_NAME*** represents the name of your bucket:

```
gsutil mb gs://YOUR_BUCKET_NAME
```



2. To view uploaded images in the Bookshelf app, set the bucket's default access control list (ACL) to `public-read`:

```
gsutil defacl set public-read gs://YOUR_BUCKET_NAME
```



Configuring the app

1. Open the `config.py` file for editing and replace the following values:
 - Set the value of `[PROJECT_ID]` to your Google Cloud project ID.
 - Set the value of `[CLOUD_STORAGE_BUCKET]` to your Cloud Storage bucket name.
2. Save and close the `config.py` file.

Containerizing the app

The sample app includes a Dockerfile, which is used to create the app's Docker image. This Docker image runs the app on GKE.

[optional-kubernetes-engine/Dockerfile](https://github.com/GoogleCloudPlatform/getting-started-python/blob/optional-kubernetes-engine/Dockerfile)

(<https://github.com/GoogleCloudPlatform/getting-started-python/blob/optional-kubernetes-engine/Dockerfile>)

```
IDPLATFORM/GETTING-STARTED-PYTHON/BLOB/STEPS/OPTIONAL-KUBERNETES-ENGINE/DOCKERFILE)
```

```
# The Google App Engine python runtime is Debian Jessie with Python installed
# and various os-level packages to allow installation of popular Python
# libraries. The source is on github at:
# https://github.com/GoogleCloudPlatform/python-docker
FROM gcr.io/google-appengine/python
```



```
# Create a virtualenv for the application dependencies.
# If you want to use Python 2, add the -p python2.7 flag.
RUN virtualenv -p python3.4 /env

# Set virtualenv environment variables. This is equivalent to running
# source /env/bin/activate. This ensures the application is executed within
# the context of the virtualenv and will have access to its dependencies.
ENV VIRTUAL_ENV /env
ENV PATH /env/bin:$PATH

# Install dependencies.
ADD requirements.txt /app/requirements.txt
RUN pip install -r /app/requirements.txt

# Add application code.
ADD . /app

# Instead of using gunicorn directly, we'll use Honcho. Honcho is a python port
# of the Foreman process manager. $PROCESSES is set in the pod manifest
# to control which processes Honcho will start.
CMD honcho start -f /app/procfile $PROCESSES
```

The sample app also includes a `.dockerignore` file that lists file paths that aren't included in the resulting Docker container. Typically, this file includes build artifacts and local dependency installations.

[optional-kubernetes-engine/.dockerignore](https://github.com/GoogleCloudPlatform/getting-started-python/blob/optional-kubernetes-engine/.dockerignore)

(<https://github.com/GoogleCloudPlatform/getting-started-python/blob/optional-kubernetes-engine/.dockerignore>)

ATFORM/GETTING-STARTED-PYTHON/BLOB/STEPS/OPTIONAL-KUBERNETES-ENGINE/.DOCKERIGNORE)

```
__pycache__
*.pyc
*.pyo
*.pyd
.Python
env
pip-log.txt
pip-delete-this-directory.txt
.tox
.coverage
.coverage.*
.cache
nosetests.xml
```

```
coverage.xml
*,cover
*.log
.git
```

1. Build the app's Docker image:

```
docker build -t gcr.io/[YOUR_PROJECT_ID]/bookshelf .
```

2. Push the image to Container Registry (<https://cloud.google.com/container-registry>) so that your cluster can access the image:

```
gcloud docker -- push gcr.io/[YOUR_PROJECT_ID]/bookshelf
```

Deploying the Bookshelf frontend

The Bookshelf app has a frontend server that handles the web requests and a backend worker that processes books and adds information.

The cluster resources needed to run the frontend are defined in the `bookshelf-frontend.yaml` file. These resources are described as a GKE deployment (<http://kubernetes.io/docs/user-guide/deployments/>). You can use deployments to create and update a replica set and its associated pods.

[optional-kubernetes-engine/bookshelf-frontend.yaml](https://github.com/GoogleCloudPlatform/getting-started-python/blob/optional-kubernetes-engine/bookshelf-frontend.yaml)

(<https://github.com/GoogleCloudPlatform/getting-started-python/blob/optional-kubernetes-engine/bookshelf-frontend.yaml>)

TING-STARTED-PYTHON/BLOB/STEPS/OPTIONAL-KUBERNETES-ENGINE/BOOKSHELF-FRONTEND.YAML)

```
# Copyright 2015 Google Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
```

```
# See the License for the specific language governing permissions and
# limitations under the License

# This file configures the bookshelf application frontend. The frontend serves
# public web traffic.

apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: bookshelf-frontend
  labels:
    app: bookshelf
# The bookshelf frontend replica set ensures that at least 3
# instances of the bookshelf app are running on the cluster.
# For more info about Pods see:
# https://cloud.google.com/kubernetes-engine/docs/pods/
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: bookshelf
        tier: frontend
    spec:
      containers:
      - name: bookshelf-app
        # Replace [G_CLOUD_PROJECT] with your project ID or use `make template`.
        image: gcr.io/[G_CLOUD_PROJECT]/bookshelf
        # This setting makes nodes pull the docker image every time before
        # starting the pod. This is useful when debugging, but should be turned
        # off in production.
        imagePullPolicy: Always
        # The PROCESSES environment variable is used by Honcho in the
        # Dockerfile's CMD to control which processes are started. In this
        # case, only the bookshelf process is needed.
        env:
        - name: PROCESSES
          value: bookshelf
        # The bookshelf process listens on port 8080 for web traffic by default.
        ports:
        - name: http-server
          containerPort: 8080
```

1. In the `bookshelf-frontend.yaml` file, replace `[G_CLOUD_PROJECT]` with your project ID. Save and close the `bookshelf-frontend.yaml` file.

2. Deploy the resources to the cluster:

```
kubectl create -f bookshelf-frontend.yaml
```



3. Track the status of the deployment:

```
kubectl get deployments
```



After the deployment has the number of available pods that you want, the deployment is complete.

If you run into issues with the deployment, you can delete it and start over:

```
kubectl delete deployments bookshelf-frontend
```



4. After the deployment is complete, you can see the pods that the deployment created:

```
kubectl get pods
```



Deploying the Bookshelf backend

The Bookshelf backend is deployed the same way as the frontend.

[optional-kubernetes-engine/bookshelf-worker.yaml](https://github.com/GoogleCloudPlatform/getting-started-python/blob/STEPS/OPTIONAL-KUBERNETES-ENGINE/BOOKSHELF-WORKER.YAML)

(<https://github.com/GoogleCloudPlatform/getting-started-python/blob/STEPS/OPTIONAL-KUBERNETES-ENGINE/BOOKSHELF-WORKER.YAML>)

```
GETTING-STARTED-PYTHON/BLOB/STEPS/OPTIONAL-KUBERNETES-ENGINE/BOOKSHELF-WORKER.YAML)
```

```
# Copyright 2015 Google Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
```



```
# limitations under the License

# This file configures the bookshelf task worker. The worker is responsible
# for processing book requests and updating book information.

apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: bookshelf-worker
  labels:
    app: bookshelf
# The bookshelf worker replica set ensures that at least 2 instances of the
# bookshelf worker pod are running on the cluster.
# For more info about Pods see:
# https://cloud.google.com/kubernetes-engine/docs/pods/
spec:
  replicas: 2
  template:
    metadata:
      labels:
        app: bookshelf
        tier: worker
    spec:
      containers:
      - name: bookshelf-app
        # Replace [GCLLOUD_PROJECT] with your project ID or use `make template`.
        image: gcr.io/[GCLLOUD_PROJECT]/bookshelf
        # This setting makes nodes pull the docker image every time before
        # starting the pod. This is useful when debugging, but should be turned
        # off in production.
        imagePullPolicy: Always
        # The PROCESSES environment variable is used by Honcho in the
        # Dockerfile's CMD to control which processes are started. In this
        # case, only the worker process is needed.
        env:
        - name: PROCESSES
          value: worker
```

1. In the `bookshelf-worker.yaml` file, replace `[GCLLOUD_PROJECT]` with your project ID. Save and close the `bookshelf-worker.yaml` file.
2. Deploy the resources to the cluster:

```
kubectl create -f bookshelf-worker.yaml
```



3. Verify that the pods are running:

```
kubectl get pods
```



Creating the Bookshelf service

GKE services (<http://kubernetes.io/docs/user-guide/services/>) provide a single point of access to a set of pods. While it's possible to access a single pod, pods are ephemeral, and it's more convenient to address a set of pods with a single endpoint. In the Bookshelf app, the Bookshelf service lets you access the Bookshelf frontend pods from a single IP address. This service is defined in the `bookshelf-service.yaml` file.

[optional-kubernetes-engine/bookshelf-service.yaml](https://github.com/GoogleCloudPlatform/getting-started-python/blob/master/optional-kubernetes-engine/bookshelf-service.yaml)

(<https://github.com/GoogleCloudPlatform/getting-started-python/blob/master/optional-kubernetes-engine/bookshelf-service.yaml>)

GETTING-STARTED-PYTHON/BLOB/STEPS/OPTIONAL-KUBERNETES-ENGINE/BOOKSHELF-SERVICE.YAML)

```
# Copyright 2016 Google Inc.
#
# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at
#
#     http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License

# The bookshelf service provides a load-balancing proxy over the bookshelf
# frontend pods. By specifying the type as a 'LoadBalancer', Kubernetes Engine
# will create an external HTTP load balancer.
# For more information about Services see:
#   https://cloud.google.com/kubernetes-engine/docs/services/
# For more information about external HTTP load balancing see:
#   https://cloud.google.com/kubernetes-engine/docs/load-balancer
apiVersion: v1
kind: Service
```



```
metadata:  
  name: bookshelf-frontend  
  labels:  
    app: bookshelf  
    tier: frontend  
spec:  
  type: LoadBalancer  
  ports:  
  - port: 80  
    targetPort: http-server  
  selector:  
    app: bookshelf  
    tier: frontend
```

Notice that the pods and the service that uses the pods are separate. Kubernetes uses [labels](http://kubernetes.io/docs/user-guide/labels/) (<http://kubernetes.io/docs/user-guide/labels/>) to select the pods that a service addresses. By using labels, you can have a service that addresses pods from different replica sets and have multiple services that point to an individual pod.

1. Create the Bookshelf service:

```
kubectl create -f bookshelf-service.yaml
```

2. Get the service's external IP address:

```
kubectl describe service bookshelf
```

It might take up to a minute to allocate the IP address. The external IP address is listed under `LoadBalancer Ingress`.

Accessing the Bookshelf app

You've now deployed all the resources needed to run the Bookshelf app on GKE. To load the app in your browser and create books, use the external IP address from the previous step.

If you deployed the worker, the books are automatically updated with information from the Google Books API.

Cleaning up

To avoid incurring charges to your Google Cloud Platform account for the resources used in this tutorial:

Delete the project

The easiest way to eliminate billing is to delete the project that you created for the tutorial.

To delete the project:


Caution: Deleting a project has the following effects:

- **Everything in the project is deleted.** If you used an existing project for this tutorial, when you delete it, you also delete any other work you've done in the project.
- **Custom project IDs are lost.** When you created this project, you might have created a custom project ID that you want to use in the future. To preserve the URLs that use the project ID, such as an **appspot.com** URL, delete selected resources inside the project instead of deleting the whole project.

If you plan to explore multiple tutorials and quickstarts, reusing projects can help you avoid exceeding project quota limits.

1. In the Cloud Console, go to the **Manage resources** page.

[GO TO THE MANAGE RESOURCES PAGE \(HTTPS://CONSOLE.CLOUD.GOOGLE.COM/IAM-ADMIN/PROJ](https://console.cloud.google.com/iam-admin/projects)

2. In the project list, select the project you want to delete and click **Delete** .

3. In the dialog, type the project ID, and then click **Shut down** to delete the project.

Delete the cluster

Deleting the cluster removes all GKE and Compute Engine resources, but you need to manually remove any resources in Cloud Storage, Datastore, and Pub/Sub.

Delete the cluster by using the following command. Replace `[YOUR_PROJECT_ID]` and `[YOUR_GCP_ZONE]` with the project and zone you used when creating the cluster (`#creating_a_kubernetes_engine_cluster`).

```
gcloud container clusters delete bookshelf --project [YOUR_PROJECT_ID] --zone [YOUR_ZONE]
```

What's next

- Try out other Google Cloud Platform features for yourself. Have a look at our [tutorials](https://cloud.google.com/docs/tutorials) (<https://cloud.google.com/docs/tutorials>).
- Explore other [Google Cloud services](https://cloud.google.com/) (<https://cloud.google.com/>).

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 our [Site Policies](https://developers.google.com/terms/site-policies) (<https://developers.google.com/terms/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated December 5, 2019.