

[Solutions](https://cloud.google.com/solutions/) (<https://cloud.google.com/solutions/>) [Solutions](#)

Authenticating end users of Cloud Run for Anthos services using Istio and Identity Platform

This tutorial shows how to authenticate end users of applications deployed to [Cloud Run for Anthos on Google Cloud](https://cloud.google.com/run/) (<https://cloud.google.com/run/>) using [Istio authentication policies](https://istio.io/docs/concepts/security/#authentication-policies) (<https://istio.io/docs/concepts/security/#authentication-policies>) and [Identity Platform](https://cloud.google.com/identity-platform/) (<https://cloud.google.com/identity-platform/>). Using Istio to authenticate means that authentication logic doesn't need to be part of the application code. This separation lets different teams be responsible for application code and authentication policy, and authentication policies can apply across multiple applications or services.

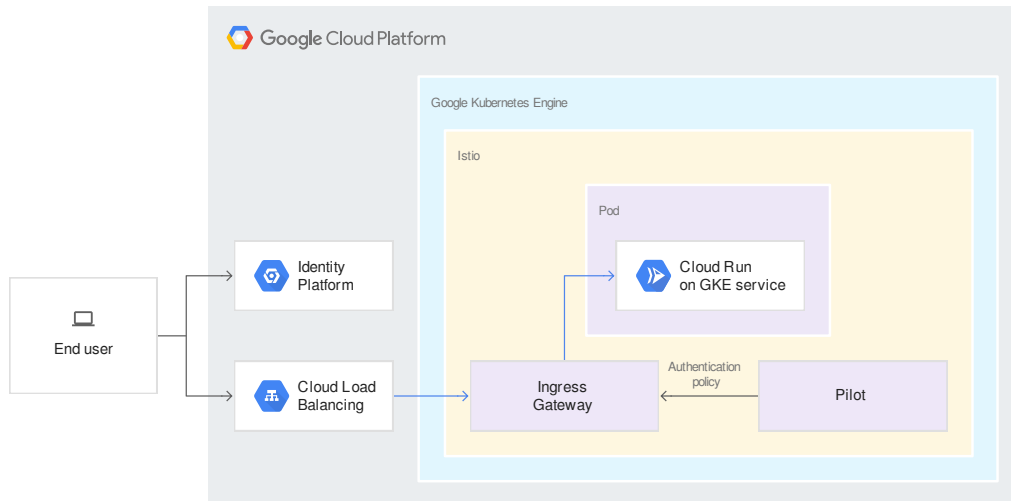
Introduction

[Cloud Run for Anthos](https://cloud.google.com/run/) (<https://cloud.google.com/run/>) provides a developer-focused experience for deploying and serving applications and functions running on GKE. It offers development teams a serverless experience, supporting autoscaling based on demand, routing and traffic management for blue-green deployments, and more. Cloud Run for Anthos builds on the open source projects Istio and Knative, and it integrates with Google Cloud products such as Stackdriver.

Istio can authenticate incoming requests by validating JSON Web Tokens (JWT) according to [authentication policies](https://istio.io/docs/concepts/security/#authentication-policies) (<https://istio.io/docs/concepts/security/#authentication-policies>). The authentication policies can apply to all services in a namespace, or to specific named services. A policy can include and exclude specific HTTP request paths, for example, to allow unauthenticated access to public website assets and health check endpoints. In this tutorial,

the [Istio ingress gateway](https://istio.io/docs/concepts/traffic-management/#ingress-and-egress). (https://istio.io/docs/concepts/traffic-management/#ingress-and-egress) enforces the authentication policy.

The following diagram shows the authentication flow that this tutorial is based on.



This tutorial uses Identity Platform to sign in end users, but you can adapt the solution to other providers that support [OpenID Connect](https://openid.net/connect/) (https://openid.net/connect/), such as [Google Sign-In](https://developers.google.com/identity/) (https://developers.google.com/identity/), [Firebase Authentication](https://firebase.google.com/docs/auth/) (https://firebase.google.com/docs/auth/), third-party offerings such as [Auth0](https://auth0.com/) (https://auth0.com/), [Gluu](https://www.gluu.org/) (https://www.gluu.org/), [Okta](https://www.okta.com/) (https://www.okta.com/), [Ping Identity](https://www.pingidentity.com/) (https://www.pingidentity.com/), or your own deployment of an [OpenID Connect implementation](https://openid.net/developers/certified/) (https://openid.net/developers/certified/).

Objectives

- Create a GKE cluster with the Cloud Run add-on.
- Set up Identity Platform.
- Deploy a sample application that consists of a public frontend and backend API.
- Add an authentication policy for the backend API.
- Verify authentication.

Costs

This tutorial uses the following billable components of Google Cloud:

- [Cloud Build](https://cloud.google.com/cloud-build/pricing) (https://cloud.google.com/cloud-build/pricing)
- [Compute Engine](https://cloud.google.com/compute/pricing) (https://cloud.google.com/compute/pricing)
- [Cloud Run for Anthos on Google Cloud](https://cloud.google.com/run/pricing#cloudrun-gke-pricing) (https://cloud.google.com/run/pricing#cloudrun-gke-pricing)
- [Container Registry](https://cloud.google.com/container-registry/pricing) (https://cloud.google.com/container-registry/pricing)
- [Identity Platform](https://cloud.google.com/identity-platform/pricing) (https://cloud.google.com/identity-platform/pricing)
- [Stackdriver](https://cloud.google.com/stackdriver/pricing) (https://cloud.google.com/stackdriver/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, or if you don't have one, [sign up for a new account](https://accounts.google.com/SignUp) (https://accounts.google.com/SignUp).
2. In the Cloud Console, go to the project selector page.

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

3. Select or create a Google Cloud project.
4. 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).
5. Enable the Cloud Build, Cloud Run, Container Analysis, and Google Kubernetes Engine APIs, and the Cloud APIs.

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

Initializing the environment

In this section, you set environment variables and `gcloud` defaults that you use later in the tutorial.

1. In the Cloud Console, from the **Select a project** drop-down, select the project you want to use.
2. Open Cloud Shell:

[GO TO CLOUD SHELL \(HTTPS://CONSOLE.CLOUD.GOOGLE.COM/?CLOUDSHELL=TRUE\)](https://console.cloud.google.com/?cloudshell=true)

You use Cloud Shell to run all the commands in this tutorial.

3. Define environment variables and `gcloud` defaults for the Compute Engine zone and GKE cluster name you want to use for this tutorial:

```
ZONE=us-central1-c
CLUSTER=cloud-run-gke-auth-tutorial

gcloud config set compute/zone $ZONE
gcloud config set run/cluster $CLUSTER
gcloud config set run/cluster_location $ZONE
```

You can change the zone and the cluster name to suit your needs.

Create GKE cluster with Cloud Run enabled

- Create a GKE cluster with the Cloud Run add-on:

```
gcloud beta container clusters create $CLUSTER \
  --addons HorizontalPodAutoscaling,HttpLoadBalancing,CloudRun \
  --enable-ip-alias \
  --enable-stackdriver-kubernetes \
  --machine-type n1-standard-2
```

Find the public IP address

Cloud Run for Anthos on Google Cloud exposes external services on the public IP address of the Istio ingress gateway.

1. Check the status of creating an external IP address for the `istio-ingress` Kubernetes Service:

```
kubectl get services istio-ingress -n gke-system --watch
```

Wait until the `EXTERNAL-IP` value changes from `<pending>` to an IP address. If you get a `NotFound` error, wait a minute and run the command again. To stop waiting, press `Control+C`.

2. Create an environment variable to store the public IP address of the Istio ingress gateway:

```
export EXTERNAL_IP=$(kubectl get services istio-ingress \
  --namespace gke-system \
  --output jsonpath='{.status.loadBalancer.ingress[0].ip}')
```

3. Display the public IP address of the Istio ingress gateway. You need this address later.

```
echo $EXTERNAL_IP
```

Note: In this tutorial, you access the services using an IP address, and using unencrypted HTTP. For a production setup, we recommend that you do both of the following:

- Create a DNS A record for your domain name and set its value to the public IP address of the Istio ingress gateway. If you do so, replace `$EXTERNAL_IP` in the rest of this tutorial with the domain name of the DNS A record. If you use Cloud DNS, follow the [Quickstart guide](https://cloud.google.com/dns/docs/quickstart#create_a_new_record) (https://cloud.google.com/dns/docs/quickstart#create_a_new_record). If you use another provider, refer to its documentation.
- [Enable HTTPS for Cloud Run for Anthos on Google Cloud](https://cloud.google.com/run/docs/gke/enabling-cluster-https) (<https://cloud.google.com/run/docs/gke/enabling-cluster-https>) services.

Set up Identity Platform

1. Leave Cloud Shell open and visit Google Cloud Marketplace to enable Identity Platform in a new web browser window.

[GO TO IDENTITY PLATFORM ON GOOGLE CLOUD MARKETPLACE](https://console.cloud.google.com/marketplace/details/googlecloudplatform/identity-platform) ([HTTPS://CONSOLE.CLOUD.GOOG](https://console.cloud.google.com/marketplace/details/googlecloudplatform/identity-platform)

2. From the **Select a project** drop-down, select the Google Cloud project where you want to set up Identity Platform. You can set up the GKE cluster and Identity Platform in separate Google Cloud projects. For simplicity, use the same project in this tutorial.
3. Click **Enable Identity Platform**.

Note: If you already have a Firebase project associated with your Google Cloud project, you might see a dialog box asking you to upgrade your Firebase project. Click **Upgrade** to use Identity Platform. If you want to use [Firebase Authentication](https://firebase.google.com/docs/auth/) (<https://firebase.google.com/docs/auth/>) instead of Identity Platform, click **Cancel** and complete the rest of this section in the [Firebase console](https://console.firebase.google.com/) (<https://console.firebase.google.com/>).

You are now on the **Identity Platform > Providers** page in the Cloud Console.

4. On the **Providers** page, click **Add a provider**.
5. From the **Select a provider** drop-down, scroll down and select **Email / Password**. For your own application, you can choose the providers you want to enable.
6. Ensure that **Enabled** is selected.
7. Clear **Allow passwordless login**.
8. Click **Save**.
9. Navigate to the **Identity Platform > Settings** page.
10. Click on the **Security** tab.
11. Click **Add domain**. This opens the **Add authorized domain** dialog.
12. In the **Domain** box, enter the public IP address of the Istio ingress gateway from the previous section (`$EXTERNAL_IP`).
13. Click **Add**. This closes the dialog. The IP address you entered is in the **Authorized Domains** table.
14. On the **Identity Platform > Settings** page, click **Save**.

Create a test user

1. In the Cloud Console, navigate to the **Identity Platform > Users** page.
2. Click **Add user** to add a test user for this tutorial. This opens the **Add user** dialog.

3. In the **Email** box, enter the email address of an end user. For testing, this email address doesn't have to be real. For this tutorial, use `user@example.com`.
4. In the **Password** box, enter a password for the test user. Remember this password, because you need it later.
5. Click **Add** to finish adding the user. This closes the dialog and takes you back to the **Identity Platform > Users** page.

Create a sample application

Deploy a sample application that has two services. One service is a public frontend user interface; the other service is a backend API.

1. In the **Identity Platform > Users** page, click the **Application setup details** link on the right side of the window. This opens the **Configure your application** dialog.
2. Highlight and copy the value of **apiKey** to your clipboard (`Control+C` on Chrome OS/Linux/Windows, `Cmd+C` on macOS).
3. Click **Close** to close the **Configure your application** dialog.
4. In Cloud Shell, create an environment variable to store **apiKey**, where **api-key** is the **apiKey** from the **Configure your application** dialog:

```
export AUTH_APIKEY=api-key
```

5. Create an environment variable for **authDomain**:

```
export AUTH_DOMAIN=$GOOGLE_CLOUD_PROJECT.firebaseio.com
```

6. Clone the Cloud Run samples repository from GitHub:

```
git clone https://github.com/GoogleCloudPlatform/cloud-run-samples.git
```

7. Switch to the directory containing the files for this tutorial:

```
cd cloud-run-samples/identity-platform/gke
```

8. Substitute the Identity Platform variables in the frontend JavaScript file:

```
envsubst < frontend/index.template.js > frontend/index.js
```



Deploy the sample application

1. Use Cloud Build to create two container images for the sample application, one for the frontend and one for the backend:

```
gcloud builds submit -t gcr.io/$GOOGLE_CLOUD_PROJECT/cloud-run-gke-auth-frontend
gcloud builds submit -t gcr.io/$GOOGLE_CLOUD_PROJECT/cloud-run-gke-auth-backend
```



Cloud Build stores the images in Container Registry.

2. In the GKE cluster, create two namespaces called **public** and **api**:

```
kubectl create namespace public
kubectl create namespace api
```



3. Deploy the frontend container image to Cloud Run for Anthos on Google Cloud as a service in the **public** namespace:

```
gcloud run deploy frontend \
  --namespace public \
  --image gcr.io/$GOOGLE_CLOUD_PROJECT/cloud-run-gke-auth-frontend \
  --platform gke
```



4. Deploy the backend container image to Cloud Run for Anthos on Google Cloud as a service in the **api** namespace:

```
gcloud run deploy backend \
  --namespace api \
  --image gcr.io/$GOOGLE_CLOUD_PROJECT/cloud-run-gke-auth-backend \
  --platform gke
```



5. Create an Istio virtual service

(<https://istio.io/docs/reference/config/networking/v1alpha3/virtual-service/>) that routes requests by URI path:


```
kubectl apply -f istio/virtualservice.yaml
```

This virtual service routes requests where the URI path starts with `/api/` to the backend API, and routes all other requests to the frontend user interface.

[identity-platform/gke/istio/virtualservice.yaml](https://github.com/GoogleCloudPlatform/cloud-run-samples/blob/master/identity-platform/gke/istio/virtualservice.yaml)

(<https://github.com/GoogleCloudPlatform/cloud-run-samples/blob/master/identity-platform/gke/istio/virtualservice.yaml>)

FORM/CLOUD-RUN-SAMPLES/BLOB/MASTER/IDENTITY-PLATFORM/GKE/ISTIO/VIRTUALSERVICE.YAML)

```
apiVersion: networking.istio.io/v1alpha3
kind: VirtualService
metadata:
  name: cloud-run-gke-auth
spec:
  gateways:
  - gke-system-gateway.knative-serving.svc.cluster.local
  hosts:
  - "*"
  http:
  - match:
    - uri:
        prefix: "/api/"
      rewrite:
        authority: backend.api.svc.cluster.local
        route:
        - destination:
            host: cluster-local-gateway.gke-system.svc.cluster.local
  - match:
    - uri:
        prefix: "/"
      rewrite:
        authority: frontend.public.svc.cluster.local
        route:
        - destination:
            host: cluster-local-gateway.gke-system.svc.cluster.local
```

6. Verify that unauthenticated requests to the backend API succeed:

```
curl -si $EXTERNAL_IP/api/secure.json | head -n1
```

If the output is not `HTTP/1.1 200 OK`, wait a minute and try again.

Add an Istio authentication policy

1. Create an Istio authentication policy:

```
envsubst < istio/authenticationpolicy.template.yaml | kubectl apply -f -
```

[identity-platform/gke/istio/authenticationpolicy.template.yaml](https://github.com/GoogleCloudPlatform/cloud-run-samples/blob/master/identity-platform/gke/istio/authenticationpolicy.template.yaml)

(<https://github.com/GoogleCloudPlatform/cloud-run-samples/blob/master/identity-platform/gke/istio/authenticationpolicy.template.yaml>)

```
-SAMPLES/BLOB/MASTER/IDENTITY-PLATFORM/GKE/ISTIO/AUTHENTICATIONPOLICY.TEMPLATE.YAML)
```

```
apiVersion: authentication.istio.io/v1alpha1
kind: Policy
metadata:
  name: api-origin-auth
  namespace: gke-system
spec:
  targets:
  - name: istio-ingress
    ports:
    - number: 80
    - number: 443
  origins:
  - jwt:
      issuer: "https://securetoken.google.com/$GOOGLE_CLOUD_PROJECT"
      audiences:
      - "$GOOGLE_CLOUD_PROJECT"
      jwksUri: "https://www.googleapis.com/service_accounts/v1/jwk/securetoken@
      trigger_rules:
      - excluded_paths:
        - exact: /api/healthz
        included_paths:
        - prefix: /api/
    principalBinding: USE_ORIGIN
```

This policy authenticates requests where the URI path starts with `/api/`, except the path `/api/healthz`. Because of the routing rules in the Istio virtual service deployed in the previous section, this policy authenticates requests to the backend API.

Note: The attributes in this policy are correct for Identity Platform and Firebase Authentication. If you use a different identity management solution, adjust the values according to the provider's

instructions.

2. It can take a moment for the policy to take effect. Run the following command and wait until you see `HTTP/1.1 401 Unauthorized` printed to the console:

```
while sleep 2; do
  curl -si $EXTERNAL_IP/api/secure.json | head -n1
done
```



Initially, you might see the output alternate between `HTTP/1.1 200 OK` and `HTTP/1.1 401 Unauthorized`. This is due to eventual consistency in [Istio](https://istio.io/docs/concepts/security/#updating-authentication-policies) (<https://istio.io/docs/concepts/security/#updating-authentication-policies>) and [Envoy Proxy](https://blog.envoyproxy.io/embracing-eventual-consistency-in-soa-networking-32a5ee5d443d) (<https://blog.envoyproxy.io/embracing-eventual-consistency-in-soa-networking-32a5ee5d443d>).

When you see only `HTTP/1.1 401 Unauthorized` being printed to the console, press `Control+C` to stop waiting.

Test the solution

1. Open a browser window to the address `http://$EXTERNAL_IP/api/secure.json`, where `$EXTERNAL_IP` is the public IP address of the Istio ingress gateway you found in the section [Find the public IP address](#) (`#find_the_public_ip_address`).

This is a request directly to the backend API. The browser window shows **Origin authentication failed** because the request didn't include credentials that satisfied the Istio authentication policy.

2. Open a browser window to the address `$EXTERNAL_IP`. You should see a sign-in form.
3. Sign in as the test user you created in the section [Create a test user](#) (`#create_a_test_user`).

The page shows the test user email address and the text **The secret message is: Hello World**. It can take a moment for the message to appear.

The browser fetches the message with an HTTP request to the backend API (`/api/secure.json`), using a token obtained from Identity Platform when signing in. Inspect the file `frontend/index.js` to view the implementation, and refer to the [FirebaseUI library](https://github.com/firebase/firebaseui-web#using-firebaseui-for-authentication) (<https://github.com/firebase/firebaseui-web#using-firebaseui-for-authentication>) for further documentation.

Troubleshooting

If you run into problems with this tutorial, we recommend that you review these documents:

- [Cloud Run for Anthos on Google Cloud troubleshooting](https://cloud.google.com/run/docs/gke/troubleshooting) (https://cloud.google.com/run/docs/gke/troubleshooting)
- [GKE troubleshooting](https://cloud.google.com/kubernetes-engine/docs/troubleshooting) (https://cloud.google.com/kubernetes-engine/docs/troubleshooting)
- [Istio Operations Guide](https://istio.io/help/ops/) (https://istio.io/help/ops/)
- [Troubleshooting Kubernetes clusters](https://kubernetes.io/docs/tasks/debug-application-cluster/debug-cluster/) (https://kubernetes.io/docs/tasks/debug-application-cluster/debug-cluster/)

Cleaning up

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


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 resources

If you want to keep the Google Cloud project you used in this tutorial, delete the individual resources:

1. Delete the GKE cluster:

```
gcloud container clusters delete $CLUSTER --async --quiet
```

2. Delete the sample application container images in Container Registry:

```
gcloud container images delete gcr.io/$G00GLE_CLOUD_PROJECT/cloud-run-gke-auth-  
--force-delete-tags --quiet
```

```
gcloud container images delete gcr.io/$G00GLE_CLOUD_PROJECT/cloud-run-gke-auth-  
--force-delete-tags --quiet
```

3. Delete the **Email / Password** identity provider in Identity Platform.

- a. In the Cloud Console, go to the **Identity Platform > Providers** page:

[GO TO THE PROVIDERS PAGE \(HTTPS://CONSOLE.CLOUD.GOOGLE.COM/CUSTOMER-IDENTITY/PROVIDERS\)](https://console.cloud.google.com/customer-identity/providers)


- b. Find the **Email / Password** identity provider in the table of providers and click .

- c. In the dialog that appears, click **Delete** to confirm.

4. Delete the test user.

- a. Go to the **Identity Platform > Users** page:

[GO TO THE USERS PAGE \(HTTPS://CONSOLE.CLOUD.GOOGLE.COM/CUSTOMER-IDENTITY/USERS\)](https://console.cloud.google.com/customer-identity/users)


- b. Find **user@example.com** in the table of users and click .

- c. In the dialog that appears, click **Delete** to confirm.

5. Delete the authorized domain.

- a. Go to the **Identity Platform > Settings** page:

[GO TO THE SETTINGS PAGE \(HTTPS://CONSOLE.CLOUD.GOOGLE.COM/CUSTOMER-IDENTITY/SETTINGS\)](https://console.cloud.google.com/customer-identity/settings)

- b. In the table of authorized domains, find the IP address you added in the section [Set up Identity Platform \(#set_up_identity_platform\)](#) (**\$EXTERNAL_IP**) and click .

- c. Click **Save**.

What's next

- Explore other [Cloud Run demos, tutorials, and samples](https://cloud.google.com/run/docs) (<https://cloud.google.com/run/docs>).
- Explore [other ways to authenticate developers, services, and users](https://cloud.google.com/run/docs/securing/authenticating) (<https://cloud.google.com/run/docs/securing/authenticating>) of applications deployed to Cloud Run.
- Learn how to [use a custom domain for services deployed to Cloud Run](https://cloud.google.com/run/docs/mapping-custom-domains) (<https://cloud.google.com/run/docs/mapping-custom-domains>).
- [Enable HTTPS on a Cloud Run for Anthos on Google Cloud cluster](https://cloud.google.com/run/docs/gke/enabling-cluster-https) (<https://cloud.google.com/run/docs/gke/enabling-cluster-https>).
- [Configure custom claims, such as roles](https://cloud.google.com/identity-platform/docs/how-to-configure-custom-claims) (<https://cloud.google.com/identity-platform/docs/how-to-configure-custom-claims>), on end users in Identity Platform.
- Explore [Knative](https://www.knative.dev/) (<https://www.knative.dev/>), the open source project that underpins Cloud Run.
- 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>).

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 November 26, 2019.