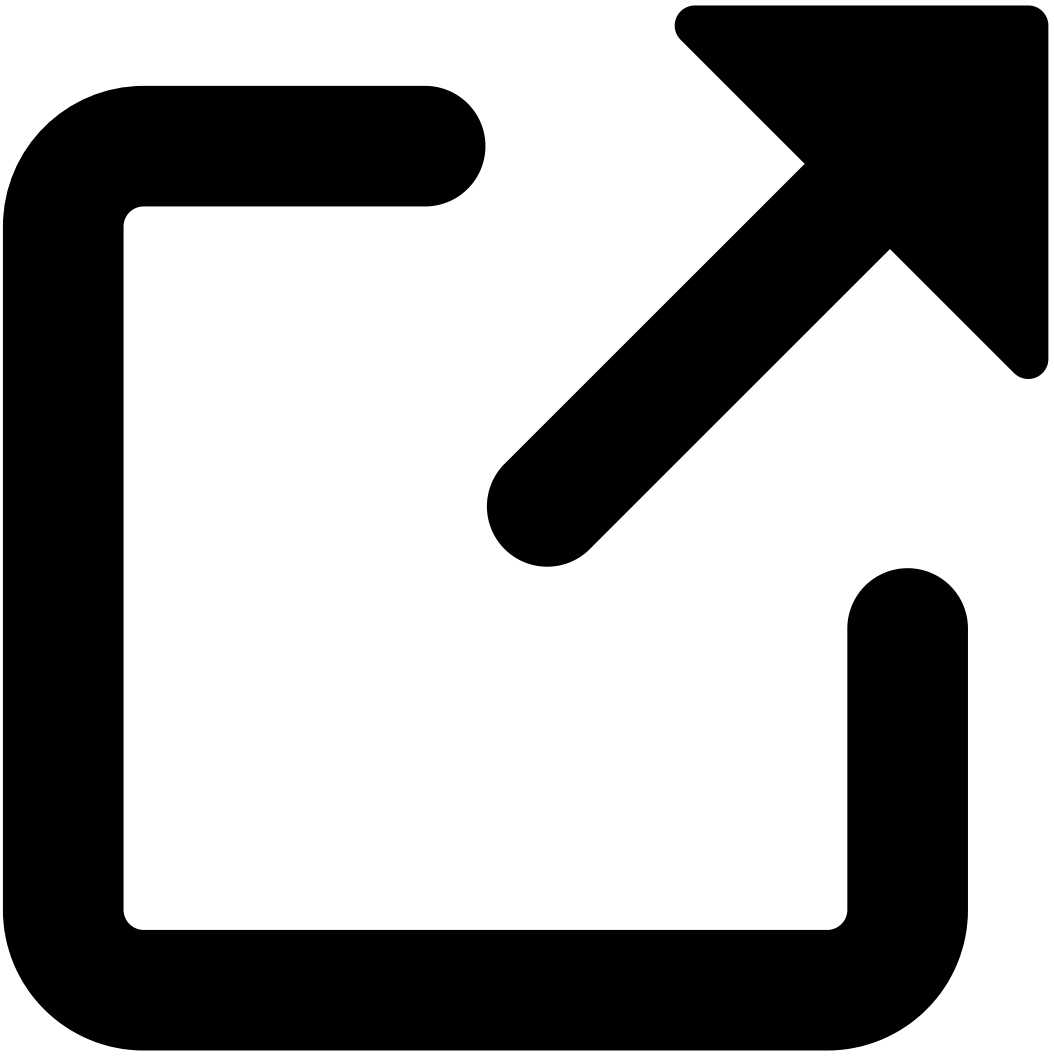
Deploy or upgrade JIT Access

Archived: 2026-04-05 17:37:23 UTC

This article describes how you can deploy the Just-In-Time Access application (or JIT Access for short) to App Engine or Cloud Run.

Deploying the Just-In-Time Access application to Cloud Run requires a more complex configuration than deploying the application to App Engine. We therefore recommend that you use App Engine unless you're deploying in a region



that doesn't support App Engine, or if you can't use App Engine for other reasons.

This section assumes that you are an administrator.

Configure your Google Cloud project

1. In the Cloud Console, switch to your project and enable required APIs:

   Enable the Cloud Asset Inventory, Resource Manager, Identity-Aware Proxy, Container Registry, Cloud Build, Identity and Access Management, and Directory APIs.

   Enable the APIs

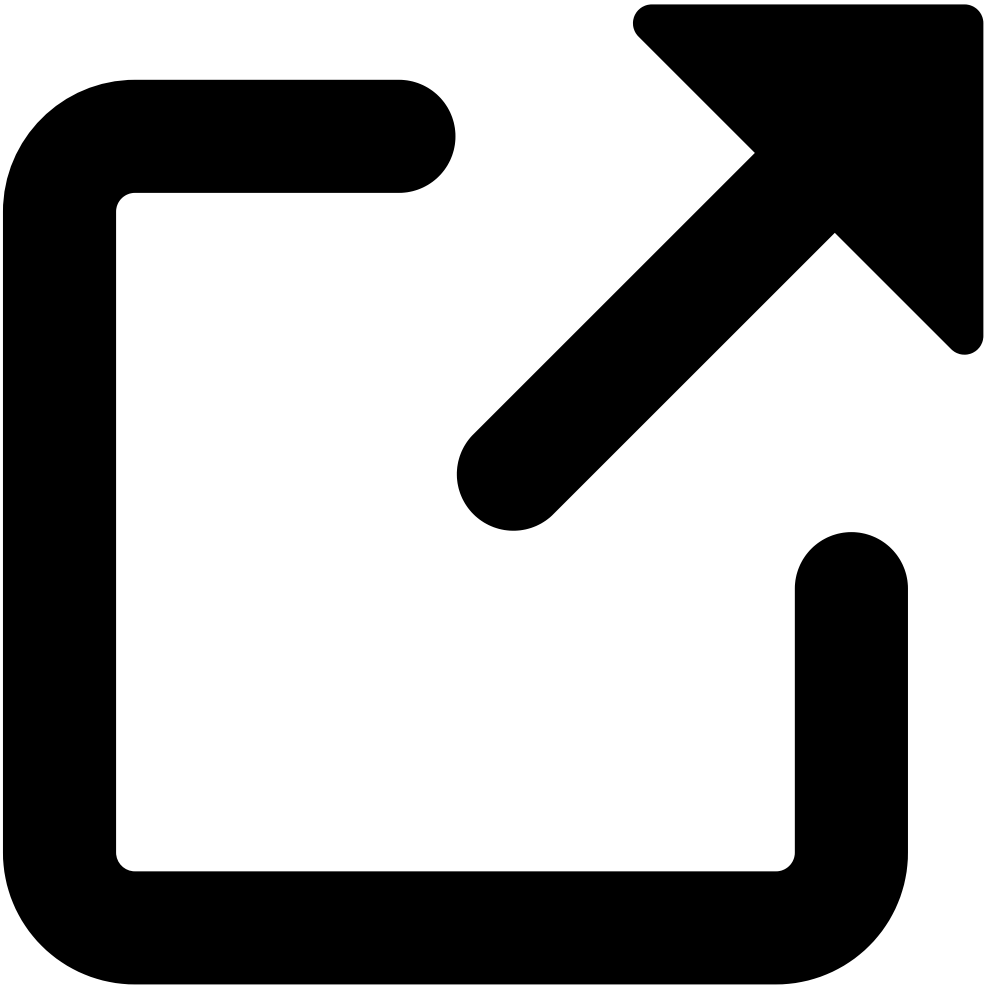
   Enable the Cloud Asset Inventory, Resource Manager, Identity-Aware Proxy, Container Registry, Cloud Run, Compute Engine, Identity and Access Management, and Directory APIs.

   Enable the APIs

2. Open Cloud Shell.

   Open Cloud Shell

3. Set an environment variable to contain your [project ID]



:

```
gcloud config set project PROJECT_ID
```

Replace *PROJECT_ID* with the ID of your project.

4. Create a service account for the Just-in-Time Access application:

```
SERVICE_ACCOUNT=$(gcloud iam service-accounts create jitaccess --display-name "Just-In-Time Access" --f
```

5. Allow the application to create tokens using its service account by granting it the **Service Account Token Creator** role (`roles/iam.serviceAccountTokenCreator`):

```
gcloud iam service-accounts add-iam-policy-binding $SERVICE_ACCOUNT \
 --member "serviceAccount:$SERVICE_ACCOUNT" \
 --role "roles/iam.serviceAccountTokenCreator"
```

The application uses the permission to create tokens to access the Directory API and, optionally, to handle multi-party approval workflows.

Grant the Just-in-Time Access application permission to manage IAM bindings

You now grant the **Project IAM Admin** role to the application's service account. This role lets the Just-In-Time Access application create temporary IAM bindings when it must grant just-in-time access.

Because the **Project IAM Admin** role is highly privileged, you must limit access to the application's service account and to the project that contains it.

Use the following guidelines:

- Limit the number of users that can access the project, and avoid granting any user the **Owner** or **Editor** role.
- Limit the number of users that can impersonate the service account. The users who should be able to do this impersonation include those who have the **Service Account User** role or the **Service Account Token Creator** role.

To grant the **Project IAM Admin** role to the service account, do the following:

1. Grant the **Project IAM Admin** role (`roles/resourcemanager.projectIamAdmin`) and **Cloud Asset Viewer** role (`roles/cloudasset.viewer`) to the part of your resource hierarchy that you want to manage just-in-time privileged access for:

   ```
    SCOPE_ID=RESOURCE_PROJECT_ID
   SCOPE_TYPE=projects

   gcloud projects add-iam-policy-binding $SCOPE_ID \
       --member "serviceAccount:$SERVICE_ACCOUNT" \
       --role "roles/resourcemanager.projectIamAdmin" \
       --condition None

   gcloud projects add-iam-policy-binding $SCOPE_ID \
       --member "serviceAccount:$SERVICE_ACCOUNT" \
       --role "roles/cloudasset.viewer" \
       --condition None
   ```

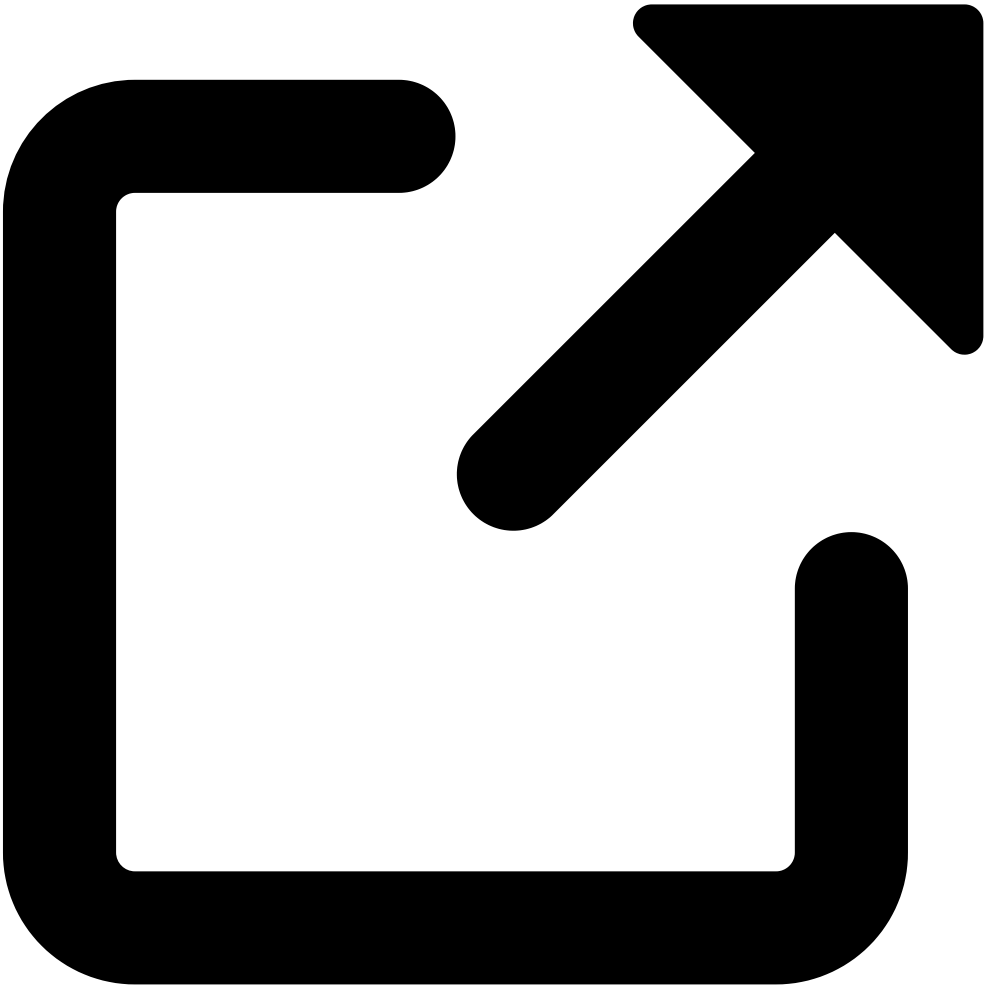
   Replace *RESOURCE_PROJECT_ID* with the ID of the Google Cloud project that you want to manage access for. This project is a different one than the one you're deploying Just-In-Time Access to.

```
 SCOPE_ID=RESOURCE_FOLDER_ID
SCOPE_TYPE=folders

gcloud resource-manager folders add-iam-policy-binding $SCOPE_ID \
    --member "serviceAccount:$SERVICE_ACCOUNT" \
    --role "roles/resourcemanager.projectIamAdmin" \
    --condition None
gcloud resource-manager folders add-iam-policy-binding $SCOPE_ID \
    --member "serviceAccount:$SERVICE_ACCOUNT" \
    --role "roles/cloudasset.viewer" \
    --condition None
```

Replace *RESOURCE_FOLDER_ID* with the ID of the folder that contains the projects that you want to manage access for.

```
 SCOPE_ID=ORGANIZATION_ID
SCOPE_TYPE=organizations

gcloud organizations add-iam-policy-binding $SCOPE_ID \
    --member "serviceAccount:$SERVICE_ACCOUNT" \
    --role "roles/resourcemanager.projectIamAdmin" \
    --condition None
gcloud organizations add-iam-policy-binding $SCOPE_ID \
    --member "serviceAccount:$SERVICE_ACCOUNT" \
    --role "roles/cloudasset.viewer" \
    --condition None
```

Replace *ORGANIZATION_ID* with the ID of your organization



.

Grant access to allow the application to resolve group memberships

The Just-In-Time Access application lets you grant eligible access to a specific user or to an entire group. To evaluate group memberships, the application must be allowed to read group membership information from your Cloud Identity or Google Workspace account.

To grant the application's service account access permission to read group memberships, do the following:

1. Open the Admin Console and sign in as a super-admin user.
2. Go to **Account > Admin Roles**:

   Go to Admin Roles

3. Click **Groups Reader > Admins**.
4. Click **Assign service accounts**.
5. Enter the following email address:

   ```
   jitaccess@PROJECT_ID.iam.gserviceaccount.com
   ```

   Replace *PROJECT_ID* with the ID of your Google Cloud project.
6. Click **Add**.
7. Click **Assign role**.

Look up your Cloud Identity or Google Workspace account's customer ID

To evaluate group memberships using the Directory API, the Just-In-Time Access application needs your Cloud Identity or Google Workspace account's customer ID. To look up this ID, do the following:

1. In the Admin Console, go to **Account > Account settings**:

   Go to Account settings
2. Copy your account's customer ID. The customer ID starts with `C`.

   You need the customer ID in a later step.
3. Close the Admin Console.

Deploy the application

You're now ready to deploy the Just-In-Time Access application to App Engine or Cloud Run.

To deploy the Just-In-Time Access application to App Engine, you perform the following steps.

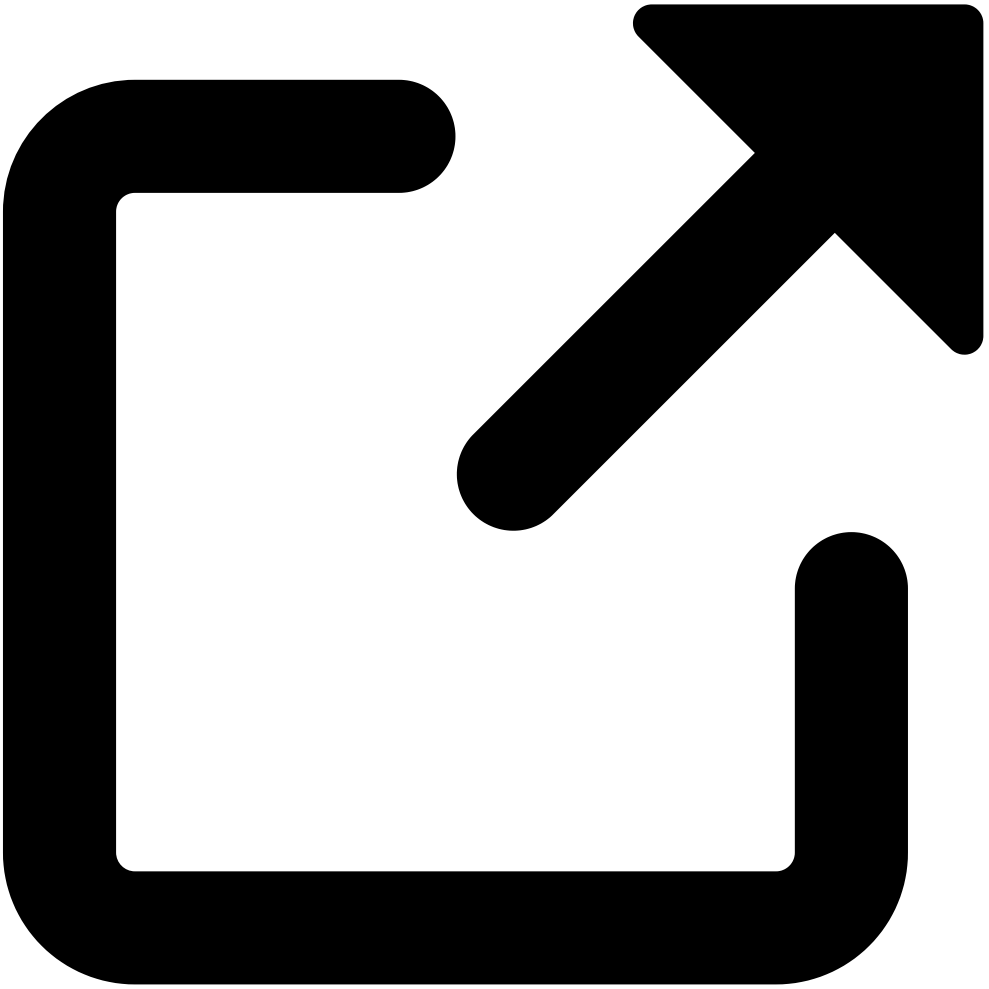
1. In Cloud Shell, set an environment variable to contain your Cloud Identity or Google Workspace account's customer ID:

   ```
   ACCOUNT_CUSTOMER_ID=CUSTOMER_ID
   ```

   Replace *CUSTOMER_ID* with the customer ID you looked up before.
2. Create an App Engine application:

   ```
   gcloud app create --region LOCATION
   ```

Replace *LOCATION* with a [supported App Engine location]



.

3. Grant the App Engine default service account the **Artifact Registry Create-on-push Writer** (`roles/artifactregistry.createOnPushWriter`) and **Storage Admin** (`roles/storage.admin`) roles to allow App Engine to useArtifact Registry.

```
 PROJECT_ID=$(gcloud config get-value core/project)
gcloud projects add-iam-policy-binding $PROJECT_ID \
    --member "serviceAccount:$PROJECT_ID@appspot.gserviceaccount.com" \
    --role "roles/artifactregistry.createOnPushWriter" \
    --condition None
gcloud projects add-iam-policy-binding $PROJECT_ID\
    --member "serviceAccount:$PROJECT_ID@appspot.gserviceaccount.com" \
    --role "roles/storage.admin" \
    --condition None
```

4. Clone the GitHub repository and switch to the `latest` branch:

```
 git clone https://github.com/GoogleCloudPlatform/jit-groups.git
cd jit-access/sources
git checkout latest
```

5. Create a configuration file for the Just-In-Time Access application:

```
 cat << EOF > app.yaml

runtime: java17
instance_class: F2
service_account: $SERVICE_ACCOUNT
env_variables:
    RESOURCE_SCOPE: $SCOPE_TYPE/$SCOPE_ID
    RESOURCE_CATALOG: AssetInventory
    RESOURCE_CUSTOMER_ID: $ACCOUNT_CUSTOMER_ID
    ACTIVATION_TIMEOUT: 60
    JUSTIFICATION_HINT: "Bug or case number"
    JUSTIFICATION_PATTERN: ".*"
EOF
```

   In this configuration file, you can customize the values of the variables. For a list of settings, see the Configuration options page in the associated GitHub repository.

6. Deploy the application:

```
 gcloud app deploy --appyaml app.yaml
```

   Take note of the `target url` in the output. This will be the public URL of the Just-in-Time Access application.

   If you see the error message `NOT_FOUND: Unable to retrieve P4SA`, retry the command.

To deploy the Just-In-Time Access application to Cloud Run, you perform the following steps.

1. In Cloud Shell, set an environment variable to contain your Cloud Identity or Google Workspace account's customer ID:

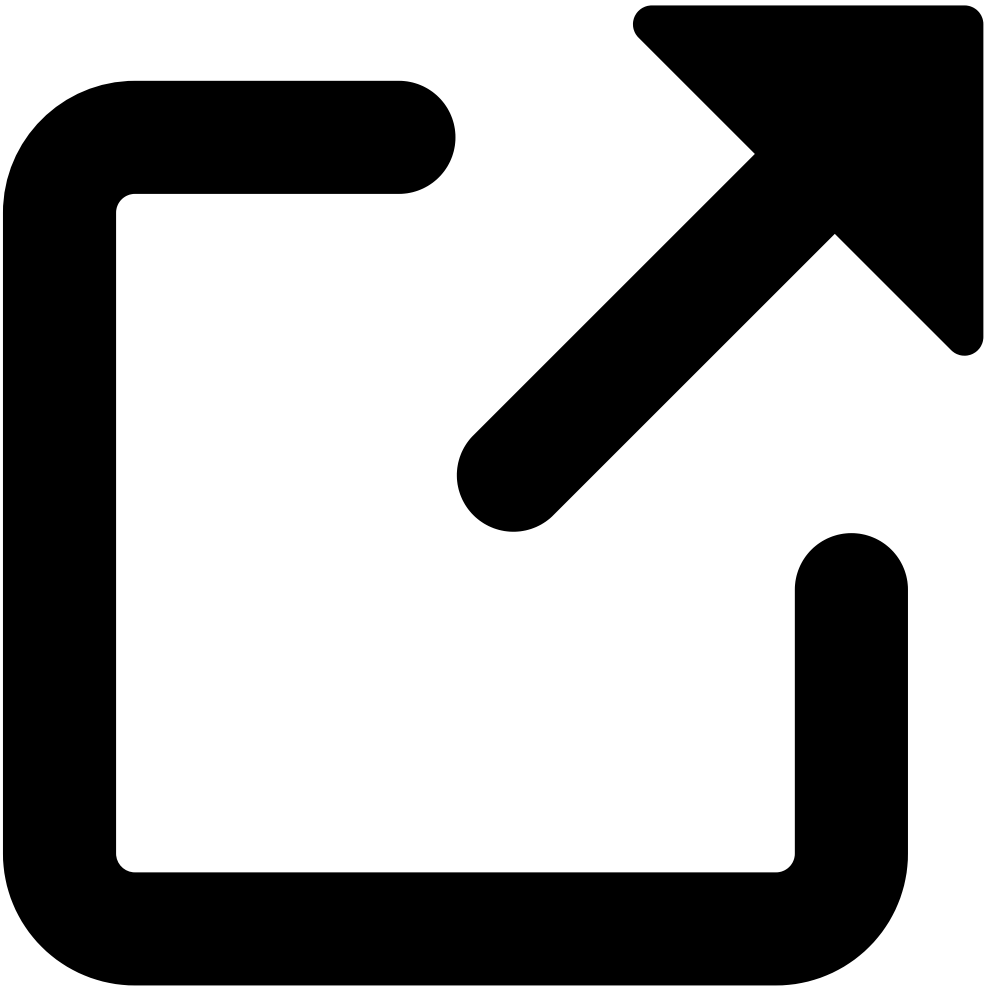
```
 ACCOUNT_CUSTOMER_ID=CUSTOMER_ID
```

   Replace *CUSTOMER_ID* with the customer ID you looked up before.

2. Select a region to deploy to:

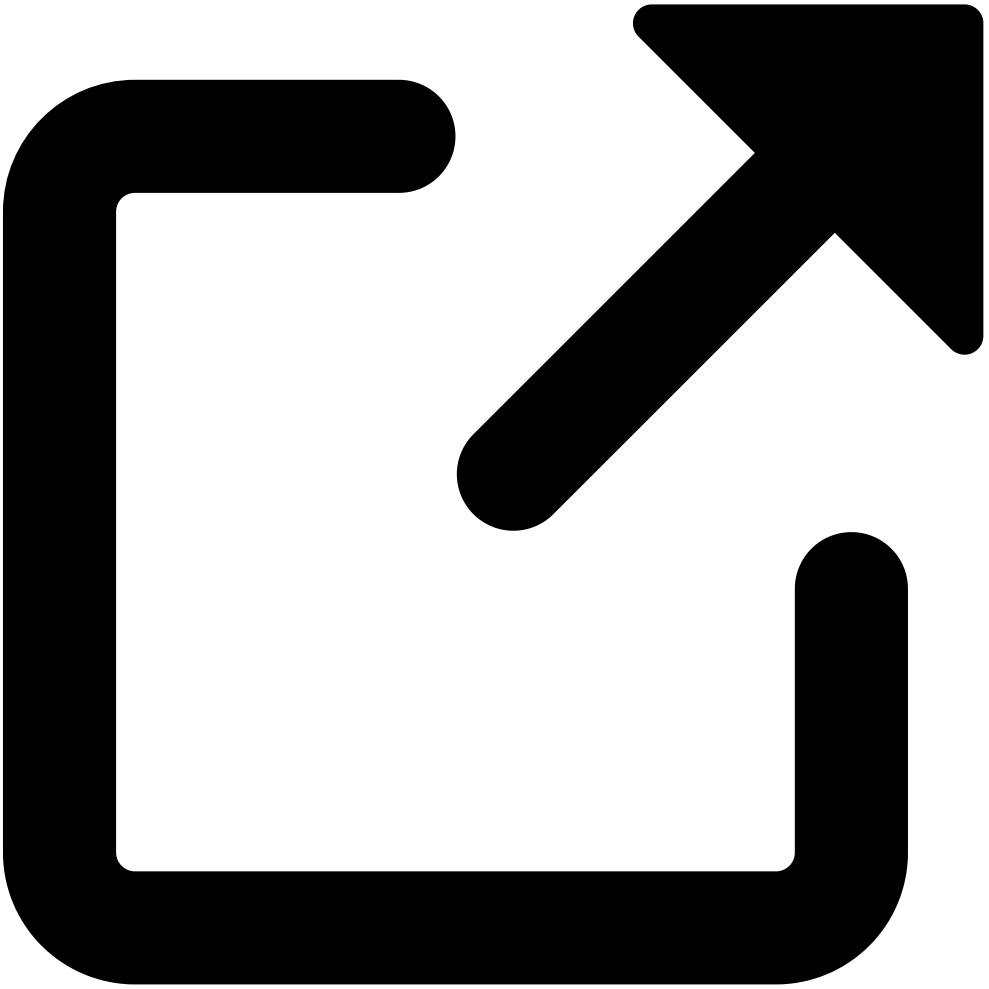
```
gcloud config set run/region REGION
```

Replace *REGION* with a region that supports Cloud Run



.

3. Create a backend service



:

```
gcloud compute backend-services create jitaccess-backend \
 --load-balancing-scheme=EXTERNAL \
 --global
```

You later use this backend service to configure a load balancer and Identity-Aware Proxy.

4. Clone the GitHub repository and switch to the `latest` branch:

```
 git clone https://github.com/GoogleCloudPlatform/jit-groups.git
cd jit-access/sources
git checkout latest
```

5. Build the application and push the container image to {{registry_name_short}}:

```
 PROJECT_ID=$(gcloud config get-value core/project)

docker build -t gcr.io/$PROJECT_ID/jitaccess:latest .
docker push gcr.io/$PROJECT_ID/jitaccess:latest
```

6. Create a configuration file for the Just-In-Time Access application:

```
 PROJECT_NUMBER=$(gcloud projects describe $PROJECT_ID --format 'value(projectNumber)')
REGION=$(gcloud config get-value run/region)
IAP_BACKEND_SERVICE_ID=$(gcloud compute backend-services describe jitaccess-backend --global --format 'va

cat << EOF > app.yaml

apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: jitaccess
  namespace: $PROJECT_NUMBER
  labels:
    cloud.googleapis.com/location: $REGION
  annotations:
    run.googleapis.com/ingress: internal-and-cloud-load-balancing
spec:
  template:
    spec:
      serviceAccountName: $SERVICE_ACCOUNT
      containers:
      - image: gcr.io/$PROJECT_ID/jitaccess:latest
        env:
        - name: RESOURCE_SCOPE
          value: "$SCOPE_TYPE/$SCOPE_ID"
        - name: RESOURCE_CATALOG
          value: "AssetInventory"
        - name: RESOURCE_CUSTOMER_ID
          value: "$ACCOUNT_CUSTOMER_ID"
        - name: ACTIVATION_TIMEOUT
          value: "60"
        - name: JUSTIFICATION_HINT
          value: "Bug or case number"
        - name: JUSTIFICATION_PATTERN
          value: ".*"
        - name: IAP_BACKEND_SERVICE_ID
```

```
          value: "$IAP_BACKEND_SERVICE_ID"
EOF
```

In this configuration file, you can customize the values of the variables. For a list of settings, see Configuration options.

7. Deploy the application:

```
gcloud run services replace app.yaml
```

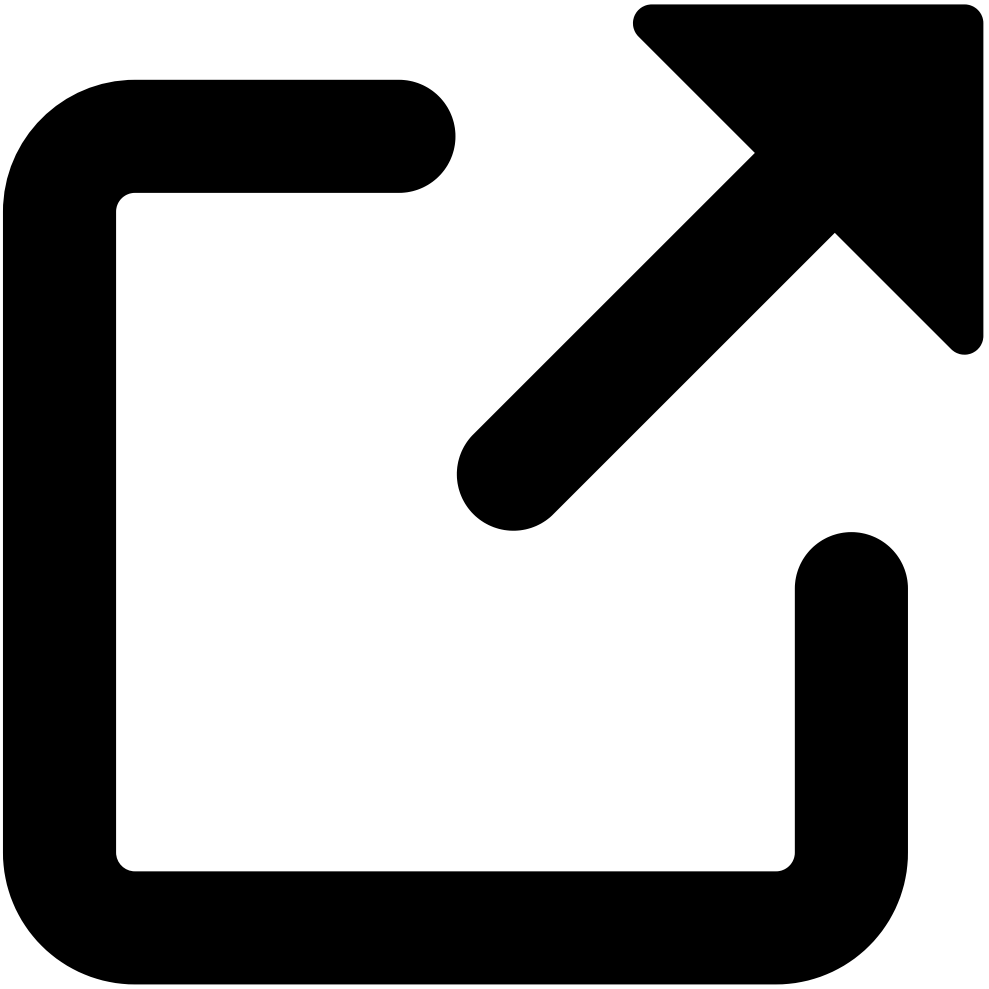
Configure a load balancer

You now configure a load balancer for the Just-In-Time Access application.

App Engine automatically configures the load balancer for you.

Configure a HTTPS load balancer for your Cloud Run service:

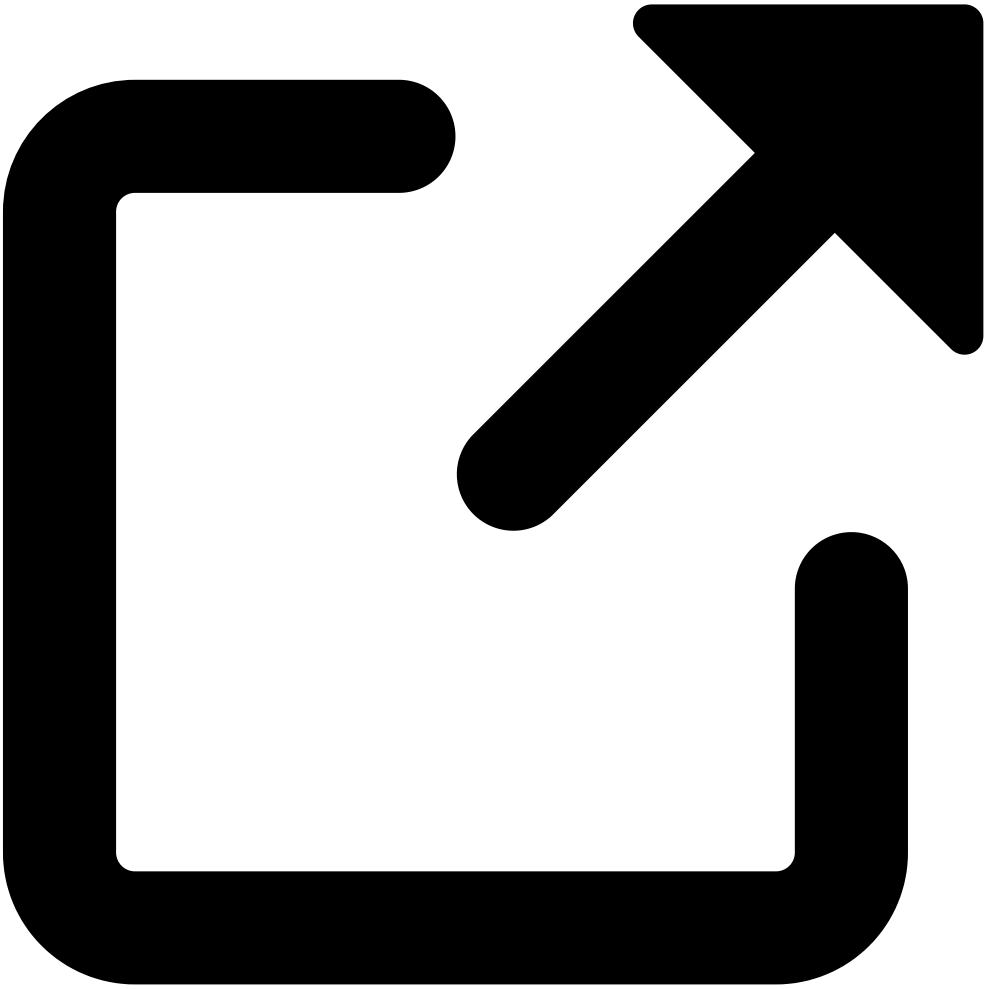
1. Reserve a static external IP address



for the load balancer:

```
gcloud compute addresses create jitaccess-ip --global
```

2. Create a managed SSL certificate



for the load balancer:

```
gcloud compute ssl-certificates create jitaccess \
 --domains <var>PUBLIC_FQDN</var> \
 --global
```

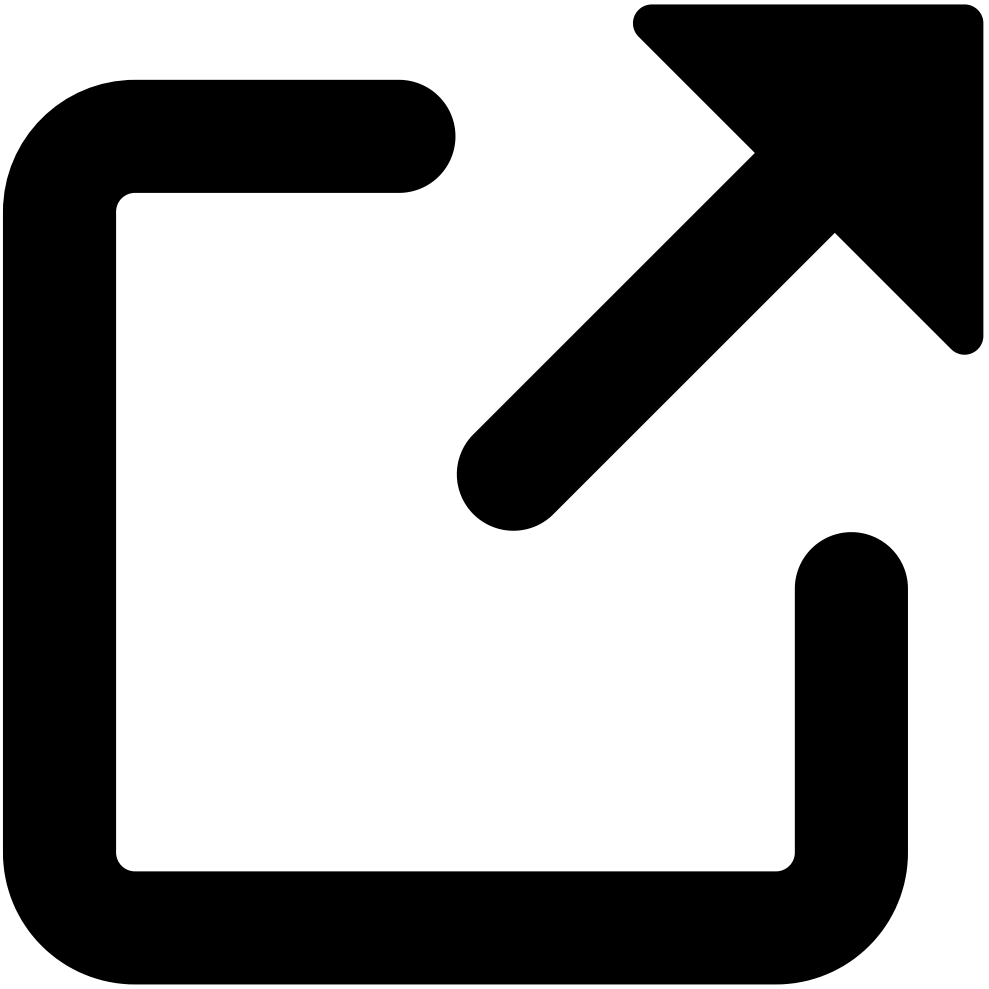
where *PUBLIC_FQDN* is the public, fully qualified domain name (FQDN) that you want to use, for example `jitaccess.example.com`.

3. Look up the IP address of the load balancer:

```
gcloud compute addresses describe jitaccess-ip \
 --global \
 --format=value\(address\)
```

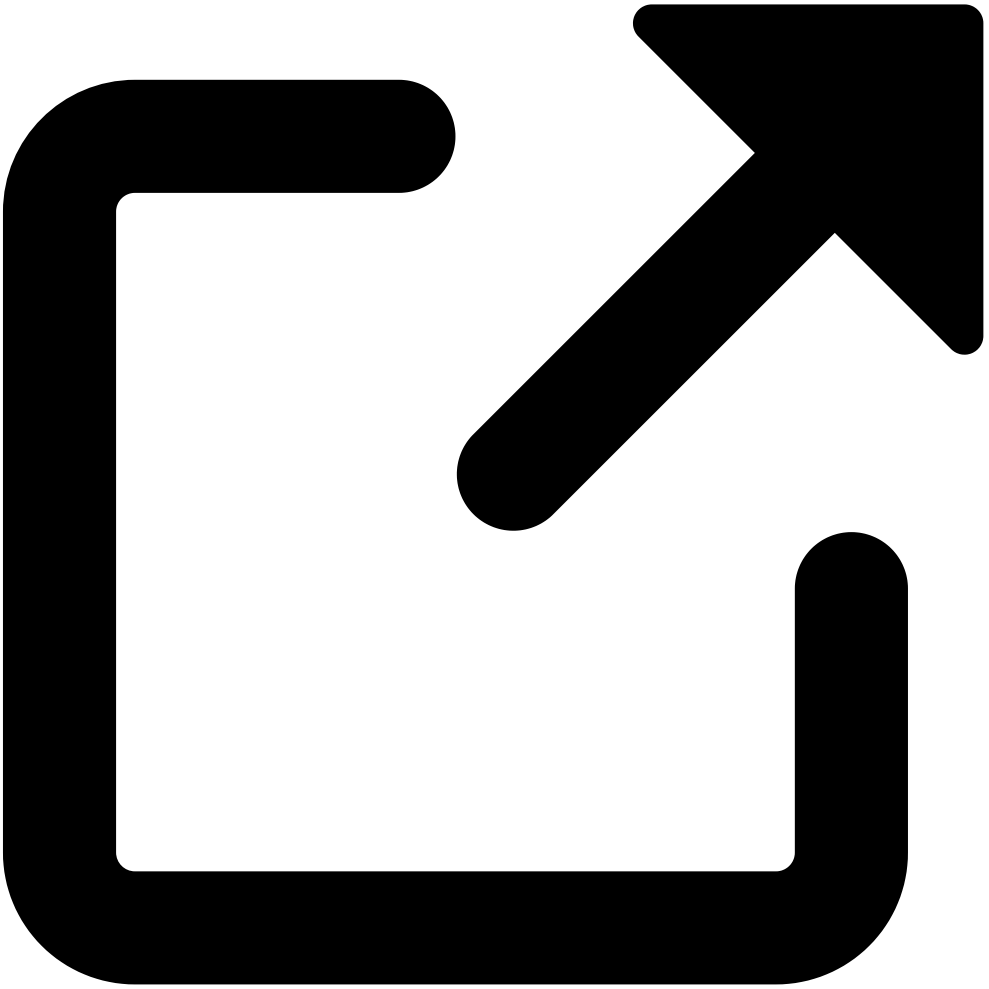
4. Create a DNS `A` record in your public DNS zone that points to the IP address of the load balancer. The fully qualified name of the DNS record must match the name that you used for the SSL certificate.

   Note: It can take multiple minutes or hours for the new DNS record to propagate. During this time, the managed SSL certificate can't be used. For details, see Troubleshooting Google-managed certificates



.

5. Create a [serverless network endpoint group]



for the Cloud Run service and connect it to the backend service:

```
 gcloud compute network-endpoint-groups create jitaccess \
  --region $(gcloud config get-value run/region) \
  --network-endpoint-type=serverless  \
  --cloud-run-service jitaccess
gcloud compute backend-services add-backend jitaccess-backend \
  --global \
  --network-endpoint-group jitaccess \
  --network-endpoint-group-region $(gcloud config get-value run/region)
```

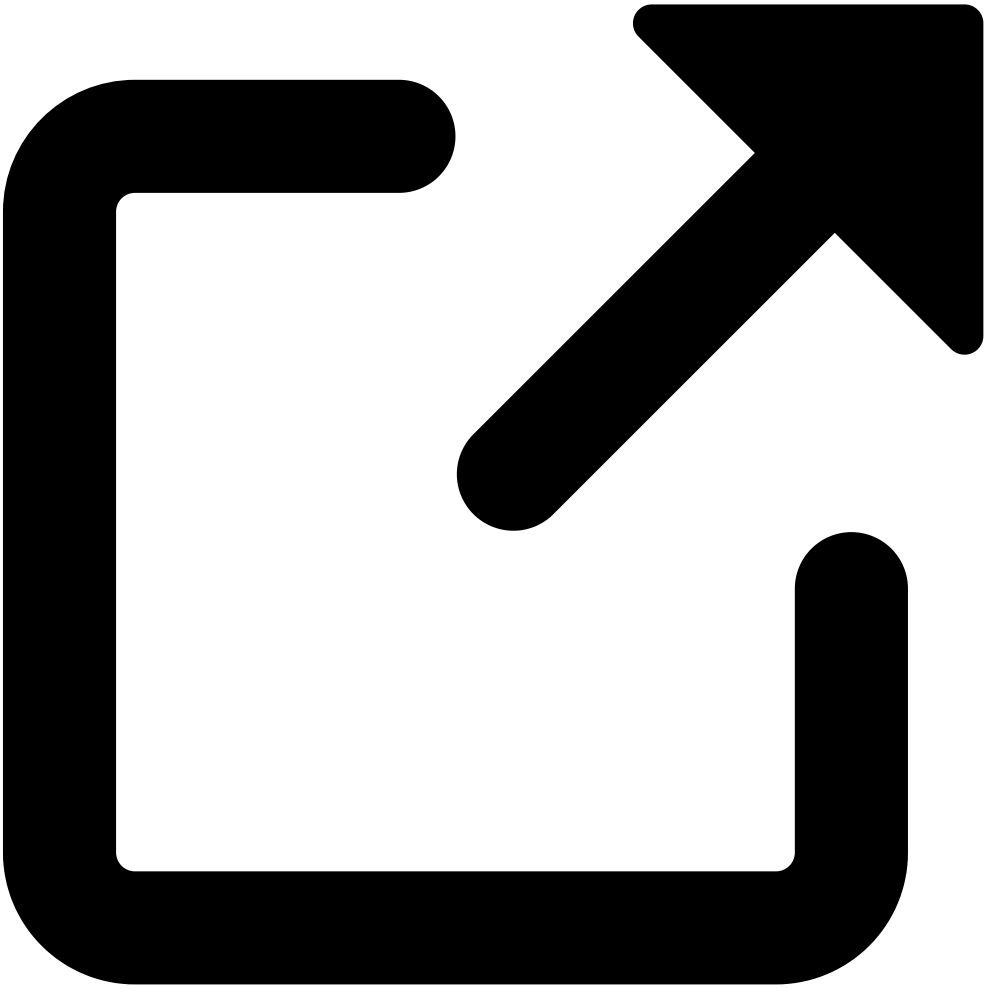
6. Create a load balancer frontend that uses the external IP address and forwards traffic to the backend service:

```
 gcloud compute url-maps create jitaccess \
  --default-service jitaccess-backend
gcloud compute target-https-proxies create jitaccess-proxy \
  --ssl-certificates jitaccess \
  --url-map jitaccess
gcloud compute forwarding-rules create jitaccess-https \
  --load-balancing-scheme EXTERNAL \
  --address jitaccess-ip \
  --target-https-proxy jitaccess-proxy \
  --global \
  --ports=443
```

Configure Identity-Aware Proxy

You now configure Identity-Aware Proxy for the Just-In-Time Access application.

1. In Cloud Shell, configure an OAuth consent screen



:

```
gcloud iap oauth-brands create \
   --application_title "Just-In-Time Access" \
   --support_email=$(gcloud config get core/account)
```

2. In the Cloud Console, go to **Security** > **Identity-Aware Proxy**.

   Go to Identity-Aware Proxy

3. Set **IAP** to **enabled**.

You now must define which users are allowed to access the Just-In-Time Access application. You can grant access to individual users, to groups, or to an entire domain.

1. In the Cloud Console, go to **IAM & Admin > IAM**.

   Go to IAM

2. Click **Grant access** and then set the following values:

   1. In the principals list, select a user, group, or domain.
   2. In the role list, select **IAP-secured web app user**.

   The **IAP-secured web app user** role lets users open the Just-In-Time Access application, but the role doesn't provide them access to any additional resources yet.

3. Click **Save**.

It can take a few minutes for the role binding to take effect.

The Identity-Aware Proxy configuration is now complete.

To complete the Identity-Aware Proxy configuration, grant the **Cloud Run Invoker** role (`roles/run.invoker`) to the service agent that's used by Identity-Aware Proxy:

```
 PROJECT_NUMBER=$(gcloud projects list \
  --filter $(gcloud config get-value project) \
  --format "value(PROJECT_NUMBER)")

gcloud projects add-iam-policy-binding $(gcloud config get-value core/project) \
  --member "serviceAccount:service-$PROJECT_NUMBER@gcp-sa-iap.iam.gserviceaccount.com" \
  --role "roles/run.invoker"
```

Test Just-in-Time Access

You can now test the process of granting eligible access and the process of using the Just-In-Time Access application to activate eligible access.

Grant eligible access

To start, you grant eligible access to a second Cloud Identity or Google Workspace user.

1. In the Cloud Console, use the project list to select a project that's part of the resource hierarchy that's managed by the Just-In-Time Access application.
2. On the IAM page, click **Grant access**.
3. Enter the email address of your second Cloud Identity or Google Workspace user and select a role such as **Project > Browser**.
4. Click **Add condition**.
5. Enter a title such as `Eligible for JIT access` .
6. Select **Condition editor** and then enter the following CEL expression:

```
has({}.jitAccessConstraint)
```

7. Save your changes.

Activate access

Now you can switch users and request temporary access to a resource.

1. Open an incognito browser window and navigate to the URL of the Just-In-Time Access application that you noted earlier.
2. Sign in as the user that you've granted eligible access to.
3. In the Just-In-Time Access application, select a role and resource that you want to activate access for.
4. Enter a justification such as `testing` and then click **Request access**.

   On the next page, notice that your access has temporarily been activated.

Analyze logs

You can now switch back to your administrative user and review the log.

1. In the Cloud Console, go to **Logging > Logs Explorer**.

   Go to Cloud Logging

2. Set **Show query** to **enabled**.
3. Enter the following query:

```
labels.event="api.activateRole"
```

4. Click **Run query**.

   The output is similar to the following:

```
 {
"textPayload": "User EMAIL activated role 'ROLE' on '//cloudresourcemanager.googleapis.com/projects/PROJ
"severity": "INFO",
"labels": {
    "resource": "//cloudresourcemanager.googleapis.com/projects/PROJECT_ID",
    "event": "api.activateRole",
    "role": "ROLE",
    "clone_id": "00c6...",
    "user": "EMAIL",
    "justification": "testing",
    ...
 },
```

```
...
}
```

Notice that a log record has been created for each role you activated. The log record includes a set of labels that you can use to create custom filters.

Upgrade Just-in-Time Access

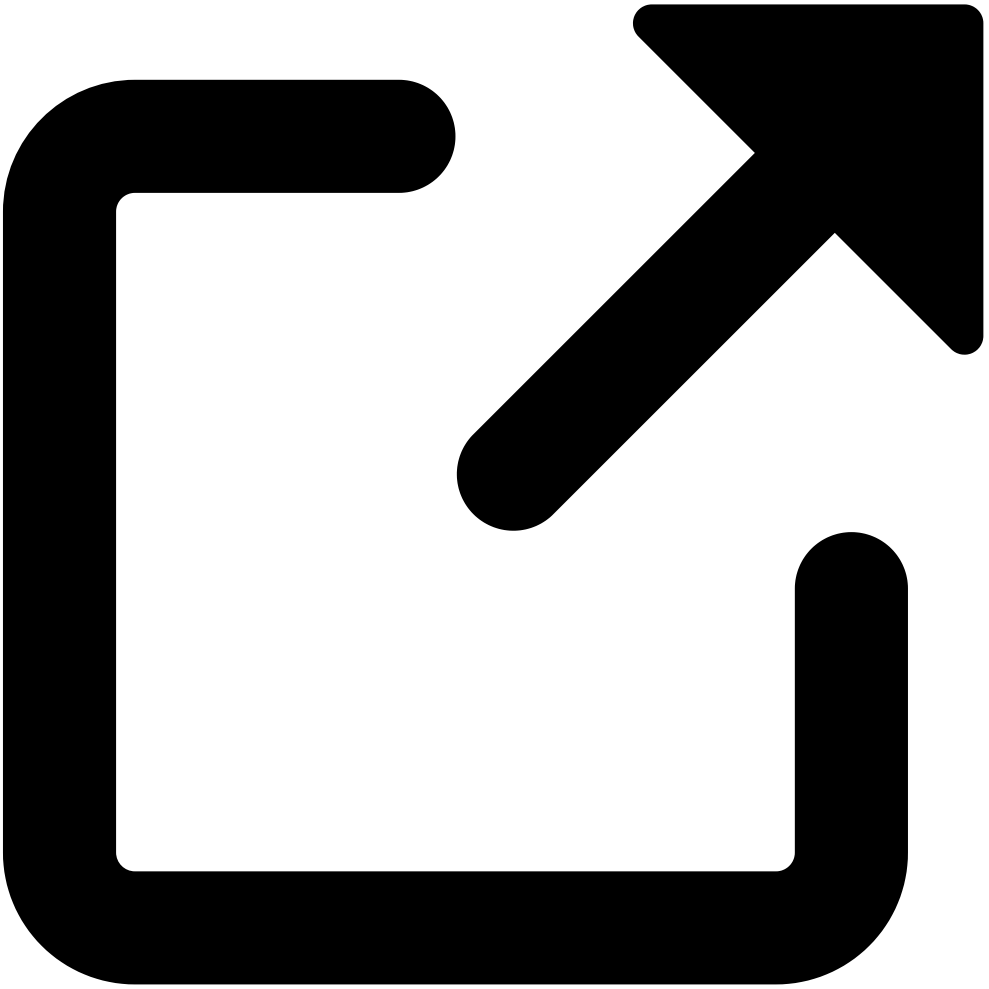
This section describes how you can upgrade an existing Just-In-Time Access deployment to use a newer version of the application, or to use a different configuration.

This section assumes that you are an administrator.

1. In the Cloud Console, switch to your project and then open Cloud Shell.

   Open Cloud Shell

2. Set an environment variable to contain your project ID



:

```
gcloud config set project PROJECT_ID
```

Replace *PROJECT_ID* with the ID of your project.

3. Clone the GitHub repository and switch to the `latest` branch:

```
 git clone https://github.com/GoogleCloudPlatform/jit-groups.git
cd jit-groups/sources
git checkout latest
```

4. Download the configuration file that you used previously to deploy the application and save it to a file `app.yaml`:

```
 APPENGINE_VERSION=$(gcloud app versions list --service default --hide-no-traffic --format "value(versio
APPENGINE_APPYAML_URL=$(gcloud app versions describe $APPENGINE_VERSION --service default --format "valu

curl -H "Authorization: Bearer $(gcloud auth print-access-token)" $APPENGINE_APPYAML_URL -o app.yaml
cat app.yaml
```

If downloading the file `app.yaml` fails, you can download your current configuration [in the Cloud Console].

```
 gcloud config set run/region REGION
gcloud run services describe jitaccess --format yaml > app.yaml
```

Replace *`REGION`* with the region that contains your existing Cloud Run deployment.

5. If you want to make changes to your configuration, edit the `app.yaml` file. For a list of settings, see the Configuration options page in the associated GitHub repository.

6. Deploy the application:

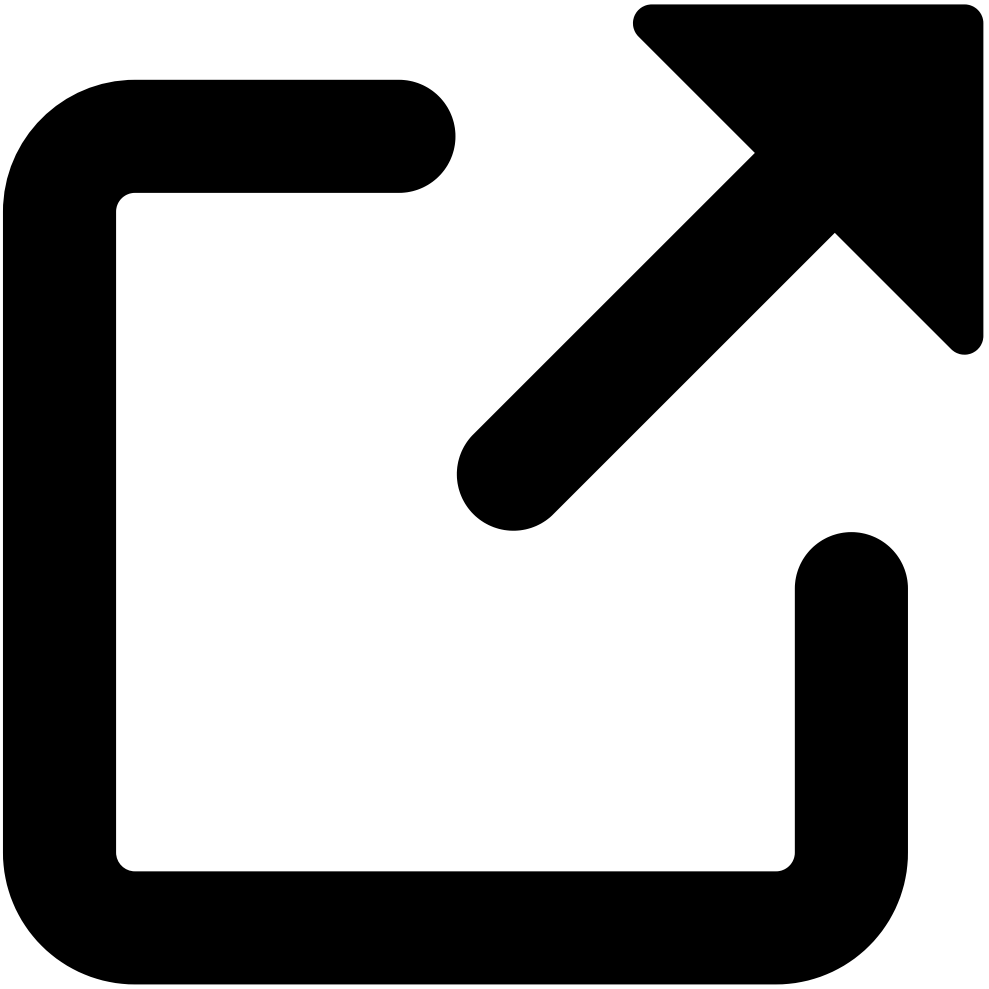
```
 sed -i 's/java11/java17/g' app.yaml
gcloud app deploy --appyaml app.yaml
```

```
 PROJECT_ID=$(gcloud config get-value core/project)

docker build -t gcr.io/$PROJECT_ID/jitaccess:latest .
docker push gcr.io/$PROJECT_ID/jitaccess:latest

IMAGE=$(docker inspect --format='&#123;{index .RepoDigests 0}}'  gcr.io/$PROJECT_ID/jitaccess)
sed -i "s|image:.*|image: $IMAGE|g" app.yaml

gcloud run services replace app.yaml
```

What's next

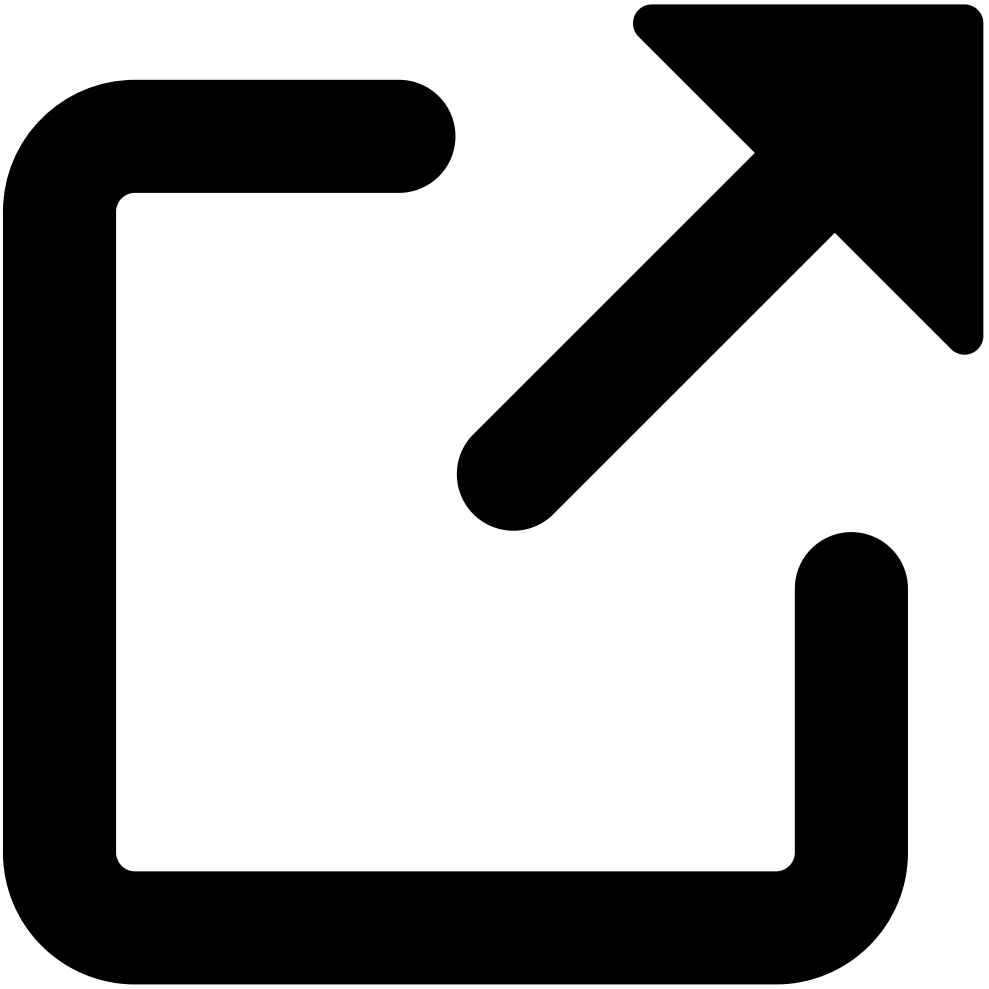
- Configure multi-party approval.

- Learn how you can use context-aware access to secure access to Just-In-Time Access



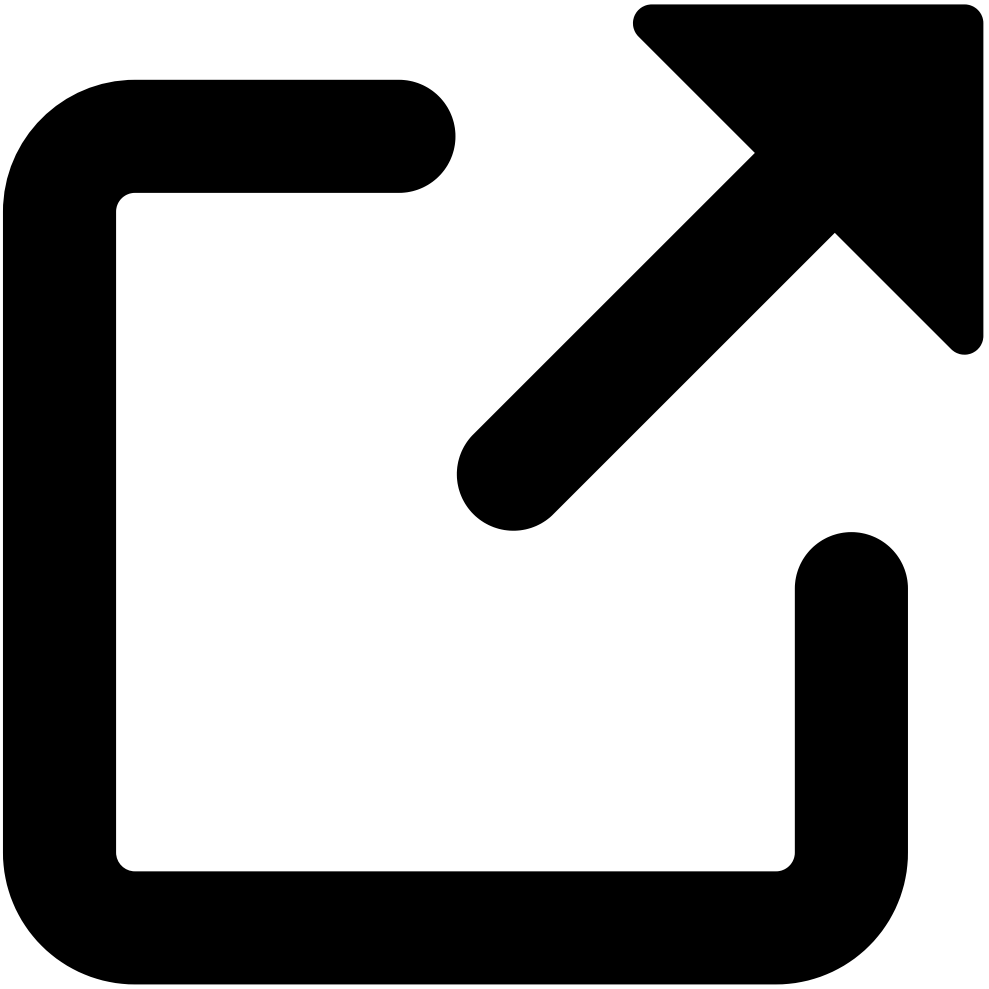
.

- Read more about IAM conditions



.

- Configure a custom domain for the Just-In-Time Access application



.

Source: https://cloud.google.com/architecture/manage-just-in-time-privileged-access-to-project