

Overview

In this lab, you learn about two policies that can be used for traffic management: SpikeArrest and Quota.

Objectives

In this lab, you learn how to perform the following tasks:

- Rate-limit traffic by using a SpikeArrest policy.
- Limit the number of orders that are created allowed for an entity by using a Quota policy.
- Configure quota limits in an API product.

Setup

For each lab, you get a new Google Cloud project and set of resources for a fixed time at no cost.

1. Sign in to Qwiklabs using an **incognito window**.
2. Note the lab's access time (for example, **1:15:00**), and make sure you can finish within that time.
There is no pause feature. You can restart if needed, but you have to start at the beginning.
3. When ready, click **Start lab**.

- Note your lab credentials (**Username** and **Password**). You will use them to sign in to the Google Cloud Console.
- Click **Open Google Console**.
- Click **Use another account** and copy/paste credentials for **this** lab into the prompts.
If you use other credentials, you'll receive errors or **incur charges**.
- Accept the terms and skip the recovery resource page.

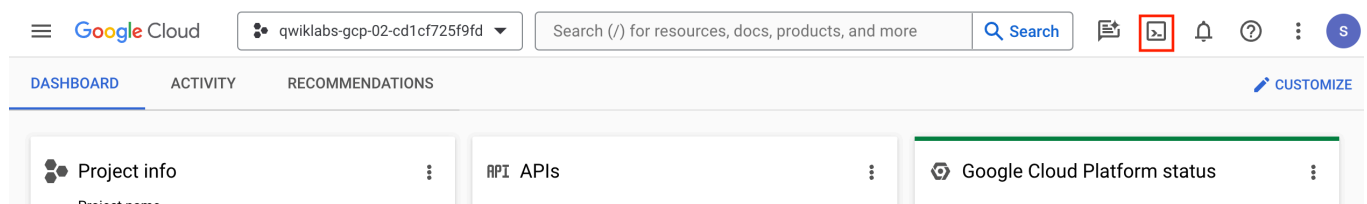
Note: Do not click **End Lab** unless you have finished the lab or want to restart it. This clears your work and removes the project.

Activate Google Cloud Shell

Google Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud.

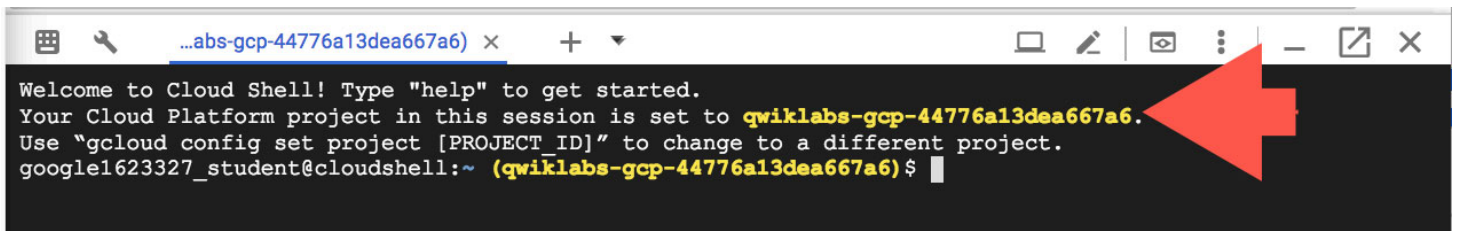
Google Cloud Shell provides command-line access to your Google Cloud resources.

- In Cloud console, on the top right toolbar, click the Open Cloud Shell button.



- Click **Continue**.

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your *PROJECT_ID*. For example:



```
...abs-gcp-44776a13dea667a6) x + ▾
Welcome to Cloud Shell! Type "help" to get started.
Your Cloud Platform project in this session is set to qwiklabs-gcp-44776a13dea667a6.
Use "gcloud config set project [PROJECT_ID]" to change to a different project.
google1623327_student@cloudshell:~ (qwiklabs-gcp-44776a13dea667a6) $
```

gcloud is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

- You can list the active account name with this command:

```
gcloud auth list
```

content_c

Output:

```
Credentialed accounts:
- @.com (active)
```

Example output:

```
Credentialed accounts:
- google1623327_student@qwiklabs.net
```

- You can list the project ID with this command:

```
gcloud config list project
```

content_c

Output:

```
[core]
project =
```

Example output:

[core]

project = qwiklabs-gcp-44776a13dea667a6

Note: Full documentation of **gcloud** is available in the gcloud CLI overview guide .

Preloaded assets

These assets have already been added to the Apigee organization:

- The **retail-v1** API proxy
- The oauth-v1 API proxy (for generating OAuth tokens)
- The backend-credentials shared flow (used by retail-v1)
- The TS-Retail target server in the eval environment (used by retail-v1)

These assets will be added to the Apigee organization as soon as the runtime is available:

- The API products, developer, and **developer app** (used by retail-v1)
- The ProductsKVM key value map in the eval environment (used by backend-credentials)
- The ProductsKVM key value map entries backendId and backendSecret


The **highlighted** items are used during this lab.

Note: Revision 1 of the retail-v1 proxy is marked as deployed, and is immutable. If you ever make a mistake in your proxy code that you can't recover from, you can select revision 1 and restart editing from there.

Task 1. Add a SpikeArrest policy

In this task, you add a SpikeArrest policy to protect against traffic surges.

Create the policy

1. In the Google Cloud console, on the **Navigation menu** () , select **Integration Services > Apigee > Proxy Development > API proxies**.
2. Select the **retail-v1** proxy.
3. Click the **Develop** tab.

You are modifying the version of the retail-v1 proxy that was created during Labs 1 through 10.

4. In the Navigator menu, click **Proxy endpoints > default > PreFlow**.
5. On the **Request PreFlow flow**, click **Add Policy Step (+)**.
6. In the **Add policy step** pane, select **Create new policy**, and then select **Traffic Management > Spike Arrest**.
7. Specify the following values:

Property	Value
Name	SA-SpikeArrest
Display name	SA-SpikeArrest

8. Click **Add**.
9. Click **Policies > SA-SpikeArrest**.
10. Replace the SpikeArrest configuration with:

```
<SpikeArrest continueOnError="false" enabled="true" name="SA-SpikeArrest">  
  <Rate>2pm</Rate>  
</SpikeArrest>
```

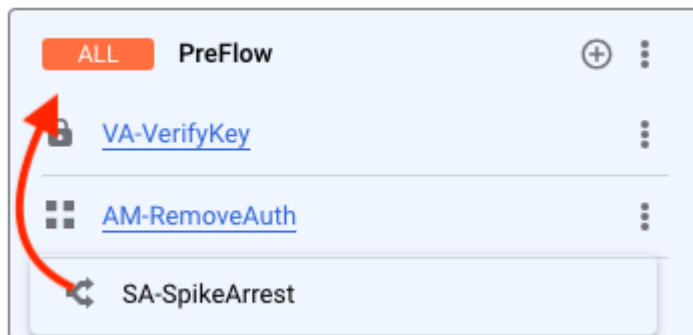
content_co

The rate of "2pm" indicates that requests will be allowed at a rate of only 2 per minute, or 1 per 30 seconds. This means that a request is allowed only if the previous request on the message processor was more than 30 seconds ago. The SpikeArrest rate is a sustained rate, and you might be able to make several requests in a row without being blocked.

There may be multiple message processors running your proxy.

Note: Two requests per minute is a very low SpikeArrest rate. This rate has been chosen only for ease of testing.

11. Drag the SpikeArrest policy to be first in the PreFlow.



12. Click **Save**, and then click **Save as New Revision**.

Task 2. Add a Quota policy

In this task you will add a Quota policy to limit the number of orders allowed for an app over a specific period of time.

1. In the Navigator pane, click **Proxy endpoints > default > createOrder**.
2. On the **Request createOrder flow**, click **Add Policy Step (+)**.
3. In the **Add policy step** pane, select **Create new policy**, and then select **Traffic Management > Quota**.
4. Specify the following values:

Property	Value
Name	Q-EnforceQuota
Display name	Q-EnforceQuota

5. Click **Add**.
6. Click **Policies > Q-EnforceQuota**.
7. Change the Quota configuration to:

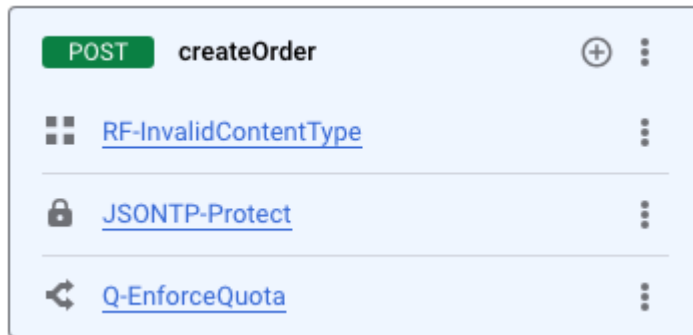
```
<Quota continueOnError="false" enabled="true" name="Q-EnforceQuota" type="calendar">
  <Identifier ref="client_id"/>
  <UseQuotaConfigInAPIProduct stepName="VA-VerifyKey">
    <DefaultConfig>
      <Allow>2</Allow>
      <Interval>1</Interval>
      <TimeUnit>hour</TimeUnit>
    </DefaultConfig>
  </UseQuotaConfigInAPIProduct>
  <Distributed>true</Distributed>
  <Synchronous>true</Synchronous>
  <StartTime>2021-01-01 00:00:00</StartTime>
</Quota>
```

content_co

You will be using the API product to specify the allowed rate. The **stepName** in the **UseQuotaConfigInAPIProduct** element specifies which step will determine the API product. When

an API key or OAuth token is validated, it can be associated with an app that is associated with an API product. Using these policy settings, the **VerifyAPIKey** step called **VA-VerifyKey** determines the API product. The VerifyAPIKey policy must run before the **Q-EnforceQuota** policy.

The quota policy is typically added after error checking is complete so that an error does not count against the quota. The **createOrder** request flow should now look like this:



The default configuration values specified in the Quota policy specify a maximum of 2 (Allow) transactions per 1 (Interval) month (TimeUnit). The default values will only be used if the values are not available, which should only happen if the quota settings are not set for the API product associated with the API key.

The **Distributed** element is set to true, so the Quota counter will be shared by all message processors.

8. Click **Save**, and then click **Save as New Revision**.
9. Click **Deploy**.
10. To specify that you want the new revision deployed to the eval environment, select **eval** as the **Environment**, and then click **Deploy**.

Check deployment status

A proxy that is deployed and ready to take traffic will show a green status.

The screenshot shows the 'Deployments' section of the Google Cloud Apigee console. A table lists the deployment status:

Status	Revision	Environment	Action
✓	1	eval	UNDEPLOY

A tooltip is displayed over the green checkmark status icon, showing:

- ✓ eval
- Status: Deployed
- Deployed on: 11/27/2023, 6:19:20 PM

Below the deployment table, there is a table for the proxy details:

Revision	Description
1	My retail API

When a proxy is marked as deployed but the runtime is not yet available and the environment is not yet attached, you may see a red warning sign. Hold the pointer over the **Status** icon to see the current status.

The screenshot shows the Google Cloud Apigee console interface. The left sidebar contains navigation options: Overview, Proxy development, API proxies (selected), Shared flows, Integrations, Offline debug, API monitor, Distribution, API product, Portals, and Developers.

The main content area shows the 'retail-v1' proxy configuration. The 'OVERVIEW' tab is active, displaying a 'Proxy summary' and a 'Deployments' section. The 'Deployments' section shows a red warning icon and '1 Disruption'.

Status	Revision	Environment	Action
!	1	eval	UNDEPLOY

A tooltip is displayed over the red warning icon, showing:

- ! eval
- Status: no instances are reporting status for this environment
- Deployed on: 11/27/2023, 6:19:20 PM

Below the deployment table, there is a table for the proxy details:

Revision	Description	Last modified
1	My retail API	November 27, 2023

If the proxy is deployed and shows as green, your proxy is ready for API traffic. If your proxy is not deployed because there are no runtime pods, you can check the status of provisioning.

Check provisioning dashboard

1. In the Google Cloud Console, navigate to **Compute Engine > VM instances**.

2. To open the Lab Startup Tasks dashboard, click on the **External IP** for the **lab-startup** VM.

Compute Engine

Virtual machines

VM instances

Instance templates

Sole-tenant nodes

Machine images

TPUs

Committed use discounts

VM instances

CREATE INSTANCE

IMPORT VM

REFRESH

INSTANCES

OBSERVABILITY

INSTANCE SCHEDULES

VM instances

Filter

Enter property name or value

<input type="checkbox"/>	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
<input type="checkbox"/>	✓	apigee-proxy-07m8	us-central1-f		apigee-proxy-group	10.128.0.3 (nic0)	34.27.73.218 (nic0)	SSH ▾ ⋮
<input type="checkbox"/>	✓	apigee-proxy-dg85	us-central1-f		apigee-proxy-group	10.128.0.4 (nic0)	34.132.15.243 (nic0)	SSH ▾ ⋮
<input type="checkbox"/>	✓	apigee-proxy-gdnx	us-central1-f		apigee-proxy-group	10.128.0.5 (nic0)	35.188.25.205 (nic0)	SSH ▾ ⋮
<input type="checkbox"/>	✓	lab-startup	us-central1-f			10.128.0.2 (nic0)	34.28.102.86 (nic0)	SSH ▾ ⋮

3. If you see a redirect notice page, click the link to the external IP address.

A new browser window will open. Lab startup tasks are shown with their progress.

- *Create proxies, shared flows, target servers* should be complete when you first enter the lab, allowing you to use the Apigee console for tasks like proxy editing.
- *Create API products, developers, apps, KVMs, KVM data* indicates when the runtime is available and those assets may be saved.
- *Proxies handle API traffic* indicates when the eval environment has been attached to the runtime and the deployed proxies can take runtime traffic.

- Lab Startup Tasks -				
Progress	Time	State	Task	
	05:02	completed	Create proxies, shared flows, target servers (environment available)	
	30:49	completed	Create API products, developers, apps, KVMs, KVM data (runtime is available)	
	31:11	started	Proxies handle API traffic (environment attached to runtime)	
	03:34	completed	Provide access to lab	
	30:05	started	Full provisioning of Apigee org qwiklabs-gcp-02-d23d90c73c5a in us-west4	
	01:41	completed	Create Apigee load balancer at api-test-qwiklabs-gcp-02-d23d90c73c5a.apigee-api	
	00:14	completed	Connect load balancer to runtime instance	

In this case, you need to wait for *Proxies handle API traffic* to complete.

While you are waiting

While you wait, learn more about policies by reviewing:

- SpikeArrest policy
- Quota policy

Task 3. Test the spike arrest

In this task, you validate that request spikes are rejected.

Store the app's key in a shell variable

The API key may be retrieved directly from the app accessible on the **Publish > Apps** page. It can also be retrieved via Apigee API call.

- In **Cloud Shell**, run the following command:

```
export API_KEY=$(curl -q -s -H "Authorization: Bearer $(gcloud auth print-access-token)" -X GET "https://apigee.googleapis.com/v1/organizations/${GOOGLE_CLOUD_PROJECT}app" | jq --raw-output '.credentials[0].consumerKey'); echo "export API_KEY=${API_KEY}" >> ~/.profile; echo "API_KEY=${API_KEY}"
```

This command retrieves a Google Cloud access token for the logged-in user, sending it as a Bearer token to the Apigee API call. It retrieves the **retail-app** app details as a JSON response, which is parsed by **jq** to retrieve the app's key. That key is then put into the **API_KEY** environment variable, and the export command is concatenated onto the **.profile** file which runs automatically when starting a Cloud Shell tab.

Note: If you run the command and it shows **API_KEY=null**, the runtime instance is probably not yet available.

Make requests until you get a spike arrest failure

- In Cloud Shell, send this curl request until you get a spike arrest failure:

```
curl -X GET -H "apikey:${API_KEY}" "https://api-test-${GOOGLE_CLOUD_PROJECT}.apiservices.dev/retail/v1/categories"
```

| json_pp

Note: To quickly repeat a command in Cloud Shell, click the UP arrow key and then press RETURN or ENTER.

You should receive 200 responses until the SpikeArrest raises a fault. It may take several requests to make it fail because each message processor is separately tracking the time that traffic is received.

You should eventually receive a **429 Too Many Requests** error that looks like this:

```
{
  "fault": {
    "faultstring": "Spike arrest violation. Allowed rate :
MessageRate{messagesPerPeriod=2, periodInMicroseconds=600000000,
maxBurstMessageCount=1.0}",
    "detail": {
      "errorcode": "policies.ratelimit.SpikeArrestViolation"
    }
  }
}
```

Note: The variable "system.uuid" is unique for each message processor. This variable can be useful if you want to distinguish requests that are handled by different message processors.

Two requests per minute is a very low value. Now that you have confirmed that the SpikeArrest policy works, you will allow a much faster rate.

Update the SpikeArrest policy

1. On the Navigator pane, click **Policies > SA-SpikeArrest**.
2. In the policy configuration, replace the rate of 2pm with 100ps .

A rate of **100 per second** will make it difficult to accidentally cause a spike arrest.

3. Click **Save**, and then click **Save as New Revision**.

4. Click **Deploy**.

5. To specify that you want the new revision deployed to the eval environment, select **eval** as the **Environment**, and then click **Deploy**.

Wait for the updated proxy to be deployed.

Task 4. Test the quota

In this task, you validate that the quota properly limits requests.

Test default configuration

- In Cloud Shell, send this request until you get a quota failure:

```
curl -H "Content-Type: application/json" -H "apikey: ${API_KEY}" -X
POST "https://api-
test-${GOOGLE_CLOUD_PROJECT}.apiservices.dev/retail/v1/orders" -d \
'{
  "orderNumber": 342345,
  "lineItems": [
    { "productId": "ME089LLA", "quantity": 1 },
    { "productId": "MD388LLA", "quantity": 2 }
  ],
  "promisedDeliveryDate": "30 Oct 2020",
  "deliveryNotes": "If not home, please place inside backyard
gate",
```

content_co

```
"destination": {  
  "addressType": "home",  
  "address": {  
    "streetAddr1": "1 Main St."  
  }  
}  
}' | json_pp
```

You should quickly get a quota error that looks like this:

```
{  
  "fault": {  
    "faultstring": "Rate limit quota violation. Quota limit exceeded.  
Identifier : VEq0HFZwGvcDGuOYmIRxx6p2I6LFyB2BTLWCKJcECOt4j2yb",  
    "detail": {  
      "errorcode": "policies.ratelimit.QuotaViolation"  
    }  
  }  
}
```

Because there are no quota settings in the API product, the default configuration in the policy is 2 requests per 1 hour.

Add a quota to the operation in the API product

1. On the left navigation menu, click **Distribution > API Products**.
2. Select **retail (full access)**.
3. Click **Edit**.
4. In the **Operations** section, click the **Actions** menu icon (⋮) and select **Edit**.
5. Set the operation's quota to **5 requests every 1 minute**, and click **Save** to save the operation.
6. Click **Save** to save the API product.

Test API product configuration

1. In Cloud Shell, send this request several times:

```
curl -H "Content-Type: application/json" -H "apikey: ${API_KEY}"  
-X POST "https://api-  
test-${GOOGLE_CLOUD_PROJECT}.apiservices.dev/retail/v1/orders" -  
d \  
{  
  "orderNumber": 342345,  
  "lineItems": [  
    { "productId": "ME089LLA", "quantity": 1 },  
    { "productId": "MD388LLA", "quantity": 2 }  
  ],  
  "promisedDeliveryDate": "30 Oct 2020",  
  "deliveryNotes": "If not home, please place inside backyard  
gate",  
  "destination": {  
    "addressType": "home",  
    "address": {  
      "streetAddr1": "1 Main St."  
    }  
  }  
}
```

content_c

2. Back in the debug tool, select a request that was sent, and click the Quota policy.

When you select the Quota policy, variables set by the Quota policy are displayed. The policy should read the API quota settings from the variables that were populated by the VerifyAPIKey policy:

Variables (14)

Name ↑	Value	Access
client_id	*****	GET
verifyapikey.VA-VerifyKey.apiprduct.developer.quota.interval	1	GET
verifyapikey.VA-VerifyKey.apiprduct.developer.quota.limit	5	GET
verifyapikey.VA-VerifyKey.apiprduct.developer.quota.timeunit	minute	GET
verifyapikey.VA-VerifyKey.apiprduct.operation.graphql.name		GET
verifyapikey.VA-VerifyKey.apiprduct.operation.graphql.type		GET
verifyapikey.VA-VerifyKey.apiprduct.operation.methods	[DELETE, GET, PATCH, POST, PUT]	GET
verifyapikey.VA-VerifyKey.apiprduct.operation.resource	/**	GET
verifyapikey.VA-VerifyKey.apiprduct.operation.rpc.methods		GET
verifyapikey.VA-VerifyKey.apiprduct.operation.service		GET
verifyapikey.VA-VerifyKey.apiprduct.operation.source		GET
verifyapikey.VA-VerifyKey.apiprduct.quota.counter.scope		GET
verifyapikey.VA-VerifyKey.apiprduct.quota.counter.scope		GET
verifyapikey.VA-VerifyKey.apiprduct.quota.counter.scope		GET

Note: These quota variables may remain empty for a short time after you change the API configuration. API product information is cached for a few minutes when it is read from the runtime data store. The updated values should be used when the API product information has expired from the cache. If you do not see the variables, send requests until you can see them.

The **Identifier** element is set to the **client_id** variable, which is the API key that was verified. This variable was set by the VerifyAPIKey policy. Each app has a different API key, and therefore each app will be tracked separately.

You should be able to send at least 5 successful requests before being rejected. If it took more requests than you expected to cause the quota violation, it may be because the quota is reset every 1 minute, indicated by the **Interval** and **TimeUnit** that were set in the API product operation.

Congratulations!

In this lab, you learned how to use the SpikeArrest and Quota policies and how to associate quota settings with an API product.

End your lab