Cross Origin Resource Sharing (CORS) is a mechanism for allowing interactions between resources from different origins, something that is normally prohibited in order to prevent malicious behavior. Use this topic to learn how to configure CORS on a Cloud Storage bucket.

For more information about Cloud Storage CORS support, see Cross-Origin Resource Sharing (CORS) (/storage/docs/cross-origin).

You set CORS configuration on a bucket by specifying information, such as HTTP methods and originating domains, that identify the types of requests it will accept. You can use the gsutil command-line tool (/storage/docs/gsutil), the XML API (/storage/docs/xml-api/overview), the JSON API (/storage/docs/json_api/), or the client libraries for Cloud Storage (/storage/docs/reference/libraries) to set CORS configuration on a bucket.

The following example illustrates how to configure CORS on the bucket named `example-bucket`. The example sets the CORS configuration as follows:

- Allow requests that originate from `example.appspot.com`.
- Allow requests that use the `GET`, `HEAD`, or `DELETE` HTTP methods.
- Allow the `Content-Type` response header to be shared across origins.
- For preflighted requests, allow the browser to make requests for 3600 seconds (1 hour) before it must repeat the preflight request.

If you run into unexpected behavior when accessing Cloud Storage buckets from a different origin, try the following steps:

1. Use `gsutil cors get` on the target bucket to get its CORS configuration. If you have multiple CORS configuration entries, make sure that as you go through the following steps that the request values map to values in the same single CORS configuration entry.
2. Review a request and response using the tool of your choice. In a Chrome browser, you can use the standard developer tools to see this information:
    a. Click the Chrome menu 

on the browser toolbar.

b. Select **More Tools > Developer Tools**.

c. Click the **Network** tab.

d. From your application or command line, send the request.

e. In the pane displaying the network activity, locate the request.

f. In the **Name** column, click the name corresponding to the request.

g. Click the **Headers** tab to see the response headers, or the **Response** tab to see the content of the response.

If you're not seeing a request and response, it is possible that your browser has cached an earlier failed preflight request attempt. Clearing your browser's cache should also clear the preflight cache. If it doesn't, set the `MaxAgeSec` value in your CORS configuration to a lower value (the default value is 1800 (30 minutes) if not specified), wait for however long the old `MaxAgeSec` was, then try the request again. This performs a new preflight request, which fetches the new CORS configuration and purges the cache entries. Once you have debugged your problem, raise `MaxAgeSec` back to a higher value, to reduce the preflight traffic to your bucket.

3. Ensure that the request has an `Origin` header and that the header value matches at least one of the `Origins` values in the bucket's CORS configuration. Note that the scheme, host, and port of the values must match exactly. Some examples of acceptable matches are as follows:

   - `http://origin.example.com` matches `http://origin.example.com:80` (because 80 is the default HTTP port), but does not match `https://origin.example.com`, `http://origin.example.com:8080`, `http://origin.example.com:5151`, or `http://sub.origin.example.com`.
   - `https://example.com:443` matches `https://example.com` but not `http://example.com` or `http://example.com:443`.
   - `http://localhost:8080` only matches exactly `http://localhost:8080`, not `http://localhost:5555` or `http://localhost.example.com:8080`.

4. Ensure that the HTTP method of the request (if this is a simple request), or the method specified in `Access-Control-Request-Method` (if this a preflight request), matches at least one of the `Methods` values in the bucket's CORS configuration.

5. If this is a preflight request, see if it includes one or more `Access-Control-Request-Header` headers. If so, then ensure that each `Access-Control-Request-Header` value matches a `ResponseHeader` value in the bucket's CORS configuration. All headers named in the `Access-Control-Request-Header` must be in the CORS configuration for the preflight request to succeed and include CORS headers in the response.