bq command-line tool reference

cloud.google.com/bigquery/docs/reference/bq-cli-reference

This document describes the syntax, commands, flags, and arguments for `bq`, the BigQuery command-line tool. It is intended for users who are familiar with BigQuery, but want to know how to use a particular `bq` command-line tool command. For general information about how to use the `bq` command-line tool, see Using the bq command-line tool.

Synopsis

The `bq` command-line tool uses the following format:

```
bq COMMAND [FLAGS] [ARGUMENTS]
```

Some flags can be used with multiple `bq` command-line tool commands; these flags are described in the Global flags section.

Other flags are command-specific; they can only be used with a particular `bq` command-line tool command. The command-specific flags are described in the command sections.

Boolean flags

Some `bq` command-line tool flags are *boolean*; you can set the flag's value to either `true` or `false`. The `bq` command-line tool accepts the following formats for setting boolean flags.

Value	Format	Example
`true`	`--FLAGNAME=true`	`--debug=true`
`true`	`--FLAGNAME`	`--debug`
`false`	`--FLAGNAME=false`	`--debug=false`
`false`	`--noFLAGNAME`	`--nodebug`

This document uses the `--FLAGNAME=VALUE` format for boolean flags.

All boolean flags are optional; if a boolean flag is not present, then BigQuery uses the flag's default value.

Specifying values for flags

When you specify a value for a flag, the equals sign `=` is optional. For example, the following two commands are equivalent:

```
bq ls --format prettyjson=myDataset
bq ls --format=prettyjson myDataset
```

This document uses the equals sign for clarity.

Online help

Documentation is available in the `bq` command-line tool, as follows:

Description	Help command format	Example
List of all commands with examples	`bq help`	`bq help`
Description of global flags	`bq --help`	`bq --help`
Description of a particular command	`bq help COMMAND`	`bq help mk`

Resource specification

The format for specifying a resource depends on the context; in some cases the separator between the project and dataset is a colon (`:`) and in some cases, it is a period (`.`). The following table describes how to specify a BigQuery table in different contexts.

Context	Format	Example
`bq` command-line tool	`PROJECT:DATASET.TABLE`	`myProject:myDataset.myTable`
Standard SQL query	`PROJECT.DATASET.TABLE`	`myProject.myDataset.myTable`
Legacy SQL query	`PROJECT:DATASET.TABLE`	`myProject:myDataset.myTable`

If you don't specify a project, then BigQuery uses the current project. For example, if the current project is `myProject` , then BigQuery interprets `myDataset.myTable` as `myProject:myDataset.myTable` (or `myProject.myDataset.myTable`).

Some resource identifiers must be quoted using back ticks (`). If your resource identifier begins with a letter or underscore character, and contains only characters that are letters, numbers, and underscores, then you don't need to quote it. However, if your resource identifier contains other types of characters, or reserved keywords, you need to surround the identifier (or the part of the identifier with the special characters or reserved keywords) with back ticks. For more information, see Identifiers.

Global flags

You can use the following flags with any `bq` command, where applicable:

`--api=ENDPOINT`
Specifies the API endpoint to call. The default value is `https://www.googleapis.com` .

`--api_version=VERSION`
Specifies the API version to use. The default is `v2`.

`--apilog=FILE`
Logs all API requests and responses to the file specified by `FILE`. Possible values are the following:

- the path to a file - logs to the specified file
- `stdout` - logs to standard output
- `stderr` - logs to standard error
- `false` - API requests and responses are not logged (default)

`--bigqueryrc=PATH`
Specifies the path to the `bq` command-line tool configuration file. If you don't specify the `--bigqueryrc` flag, then the command uses the `BIGQUERYRC` environment variable. If the environment variable is not set, then `$HOME/.bigqueryrc` is used. If that file does not exist, then `~/.bigqueryrc` is used. For more information, see Setting default values for command-line flags.

`--ca_certificates_file=PATH`
Specifies the location of your Certificate Authority Service (CA) file.

`--dataset_id=DATASET_ID`
Specifies the default dataset to use with the command. This flag is ignored when not applicable. You can specify the `DATASET_ID` argument using the format `PROJECT:DATASET` or `DATASET`. If the `PROJECT` part is missing, then the default project is used. You can override the default project setting by specifying the `--project_id` flag.

`--debug_mode={true|false}`
If set to `true`, shows tracebacks on Python exceptions. The default value is `false`.

`--disable_ssl_validation={true|false}`
If set to `true`, enables HTTPS certificate validation. The default value is `false`.

`--discovery_file=PATH`
Specifies the JSON file to read for discovery.

`--enable_gdrive={true|false}`
If set to `false`, requests a new OAuth token without Drive scope. The default value is `true`; requests a new OAuth token with Drive scope.

`--fingerprint_job_id={true|false}`
To use a job ID that is derived from a fingerprint of the job configuration, set to `true`. This prevents the same job from running multiple times accidentally. The default value is `false`.

`--format=FORMAT`
Specifies the format of the command's output. Use one of the following values:

- `pretty` : formatted table output
- `sparse` : simpler table output
- `prettyjson` : easy-to-read JSON format
- `json` : maximally compact JSON
- `csv` : csv format with header

`pretty` , `sparse` , and `prettyjson` are intended to be human-readable. `json` and `csv` are intended to be used by another program. If `none` is specified, then the command produces no output. If the `--format` flag is absent, then an appropriate output format is chosen based on the command.

`--headless={true|false}`
To run the `bq` session without user interaction, set to `true` . For example, `debug_mode` does not break into the debugger, and the frequency of informational printing is lowered. The default value is `false` .

`--httplib2_debuglevel=DEBUG_LEVEL`
Specifies whether to show HTTP debugging information. If `DEBUG_LEVEL` is greater than `0` , then the command logs HTTP server requests and responses to stderr, in addition to error messages. If `DEBUG_LEVEL` is not > 0, or if the `--httplib2_debuglevel` flag is not used, then only error messages are provided.

For example:

```
--httplib2_debuglevel=1
```

Note: Multi-level debugging is not supported for this flag, so you can set `DEBUG_LEVEL` to any positive number.

`--job_id=JOB_ID`
Specifies a job identifier for a new job. This flag applies only to commands that create jobs: `cp` , `extract` , `load` , and `query` . If you don't use the `--job_id` flag, then the commands generate a unique job identifier. For more information, see Running jobs programmatically.

`--job_property=KEY:VALUE`
A key-value pair to include in the properties field of the job configuration. Repeat this flag to specify additional properties.

`--location=LOCATION`
A string corresponding to a region or multi-region location. The location flag is required for the `bq cancel` command and for the `bq show` command when you use the `--jobs` flag to show information about jobs. The location flag is optional for the following commands:

All other commands ignore the `--location` flag.

Note: The `--location` flag was introduced in bq version 2.0.29. To verify the version of the `bq` command-line tool, enter `bq version` .

`--max_rows_per_request=MAX_ROWS`
An integer that specifies the maximum number of rows to return per read.

`--project_id=PROJECT`
Specifies the project to use for commands.

`--proxy_address=PROXY`
Specifies the name or IP address of the proxy host to use for connecting to Google Cloud.

`--proxy_password=PASSWORD`
Specifies the password to use when authenticating with the proxy host.

`--proxy_port=PORT`
Specifies the port number to use to connect to the proxy host.

`--proxy_username=USERNAME`
Specifies the username to use when authenticating with the proxy host.

`--quiet={true|false}` **or** `-q={true|false}`
To suppress status updates while jobs are running, set to `true`. The default value is `false`.

`--synchronous_mode={true|false}` **or** `-sync={true|false}`
To create the job and immediately return, with a successful completion status as the error code, set to `false`. If set to `true`, then the command waits for the job to complete before returning, and returns the job completion status as the error code. The default value is `true`.

`--trace=token:TOKEN`
Specifies a tracing token to include in API requests.

Deprecated global flags

.

Commands

The following sections describe the `bq` command-line tool commands, along with their command-specific flags and arguments.

`bq add-iam-policy-binding`

Use the `bq add-iam-policy-binding` command to retrieve the Identity and Access Management (IAM) policy for a table or view and add a binding to the policy, in one step.

This command is an alternative to the following three-step process:

1. Using the `bq get-iam-policy` command to retrieve the policy file (in JSON format).
2. Editing the policy file.

3. Using the `bq set-iam-policy` command to update the policy with a new binding.

Synopsis

```
bq add-iam-policy-binding [FLAGS] --member=MEMBER_TYPE:MEMBER --role=ROLE
  [--table] RESOURCE
```

Example

```
bq add-iam-policy-binding --member=user:myAccount@gmail.com \
  --role=roles/bigquery.dataViewer myDataset.myTable
```

Flags and arguments

The `bq add-iam-policy-binding` command uses the following flags and arguments:

`--member=MEMBER_TYPE:MEMBER`
Required. Use the `--member` flag to specify the member part of the IAM policy binding. The `--member` flag is required along with the `--role` flag. One combination of `--member` and `--role` flags equals one binding.

The *`MEMBER_TYPE`* value specifies the type of member in the IAM policy binding. Use one of the following values:

- `user`
- `serviceAccount`
- `group`
- `domain`

The *`MEMBER`* value specifies the email address or domain of the member in the IAM policy binding.

`--role=ROLE`
Required. Specifies the role part of the IAM policy binding. The `--role` flag is required along with the `--member` flag. One combination of `--member` and `--role` flags equals one binding.

`--table={true|false}`
To return an error if the *`RESOURCE`* argument is not a table or view identifier, set the `--table` flag to `true`. The default value is `false`. This flag is supported for consistency with other commands.

RESOURCE
The table or view whose policy you want to add to.

For more information, see the IAM policy reference.

`bq cancel`

Use the `bq cancel` command to cancel BigQuery jobs.

Synopsis

```
bq [--synchronous_mode=false] cancel JOB_ID
```

Examples

```
bq cancel bqjob_12345
```

```
bq --synchronous_mode=false cancel bqjob_12345
```

Flags and arguments

The `bq cancel` command uses the following flags and arguments:

`--synchronous_mode=false`
If you don't want to wait for the `bq cancel` command to complete ,set the global `--synchronous_mode` flag to `false`. The default is `true`.

`JOB_ID`
The job you want to cancel.

For more information about using the `bq cancel` command, see Managing jobs.

`bq cp`

Use the `cp` command to copy tables, create table snapshots (Preview), or restore table snapshots (Preview).

Synopsis

```
bq cp [FLAGS] SOURCE_TABLE DESTINATION_TABLE
```

Example

```
bq cp myDataset.myTable myDataset.myTableCopy
```

Flags and arguments

The `bq cp` command uses the following flags and arguments:

`--append_table={true|false}` **or** **`-a={true|false}`**
To append a table to an existing table, set to `true`. The default value is `false`.

`--destination_kms_key=KEY`
Specifies a Cloud KMS key resource ID for encrypting the destination table data.

For example:

```
--
destination_kms_key=projects/myProject/locations/global/keyRings/myKeyRing/cryptoKe
```

`--expiration=SECONDS` **(Preview)**

The number of seconds until a table snapshot expires. If not included, the table snapshot expiration is set to the default expiration of the dataset containing the new table snapshot. Use with the `--snapshot` flag.

`--force={true|false}` or `-f={true|false}`
To overwrite the destination table, if it exists, without prompting, set to `true`. The default value is `false`; if the destination table exists, then the command prompts for confirmation before overwriting.

`--no_clobber={true|false}` or `-n={true|false}`
To disallow overwriting the destination table, if it exists, set to `true`. The default value is `false`; if the destination table exists, then it is overwritten.

`--restore` (Preview)
Creates a standard table from a table snapshot. The *`SOURCE_TABLE`* argument must specify a table snapshot.

`--snapshot` (Preview)
Creates a table snapshot of the standard table that's specified in the *`SOURCE_TABLE`* argument. Requires the `--no_clobber` flag.

`SOURCE_TABLE`
The table that you want to copy.

`DESTINATION_TABLE`
The table that you want to copy to.

For more information about using the `cp` command, see the following:

`bq extract`

Use the `bq extract` command to export table data to Cloud Storage.

Synopsis

```
bq extract [FLAGS] RESOURCE DESTINATION
```

Examples

```
bq extract --compression=GZIP --destination_format=CSV --field_delimiter=tab \
    --print_header=false myDataset.myTable gs://my-bucket/myFile.csv.gzip

bq extract --destination_format=CSV --field_delimiter='|' myDataset.myTable \
  gs://myBucket/myFile.csv
```

Flags and arguments

The `bq extract` command uses the following flags and arguments:

`--compression=`*`COMPRESSION_TYPE`*

Specifies the type of compression to use for exported files. Possible values are the following:

- `GZIP`
- `DEFLATE`
- `SNAPPY`
- `NONE`

The default value is `NONE`.

For information about which formats are supported for each compression type, see Export formats and compression types.

`--destination_format=`*`FORMAT`*
Specifies the format for the exported data. Possible values are the following:

- `CSV`
- `NEWLINE_DELIMITED_JSON`
- `AVRO`
- `PARQUET` (Preview)

The default value is `CSV`.

`--field_delimiter=`*`DELIMITER`*
For CSV exports, specifies the character that marks the boundary between columns in the output file. The delimiter can be any ISO-8859-1 single-byte character. You can use `\t` or `tab` to specify tab delimiters.

`--print_header={true|false}`
To suppress printing header rows for formats that have headers, set to `false`. The default is `true`; header rows are included.

RESOURCE
The table that you are exporting from.

DESTINATION
The storage location that receives the exported data.

For more information about using the `bq extract` command, see Exporting table data.

`bq get-iam-policy`

Use the `bq get-iam-policy` command to retrieve the IAM policy for a resource and print it to `stdout`. The resource can be a table or a view. The policy is in JSON format.

Synopsis

```
bq get-iam-policy [FLAGS] RESOURCE
```

Example

```
bq get-iam-policy myDataset.myTable
```

Flags and arguments

The `bq get-iam-policy` command uses the following flags and arguments:

`--table={true|false}` or `--t={true|false}`
To return an error if *`RESOURCE`* is not a table or view identifier, set the `--table` flag to `true`. The default value is `false`. This flag is supported for consistency with other commands.

`RESOURCE`
The table or view whose policy you want to get.

For more information about the `bq get-iam-policy` command, see Introduction to table access controls.

`bq head`

Use the `bq head` command to display the specified rows and columns of a table. By default, it displays all columns of the first 100 rows.

Synopsis

```
bq head [FLAGS] [TABLE]
```

Example

```
bq head --max_rows=10 --start_row=50 --selected_fields=field1,field3 \
  myDataset.myTable
```

Flags and arguments

The `bq head` command uses the following flags and arguments:

`--job=JOB or -j=JOB`
To read the results of a query job, specify this flag with a valid job ID.

`--max_rows=MAX or -n=MAX`
An integer that indicates the maximum number of rows to print when showing table data. The default value is `100`.

`--selected_fields=COLUMN_NAMES or -c=COLUMN_NAMES`
A comma-separated list that specifies a subset of fields (including nested and repeated fields) to return when showing table data. If this flag is not specified, then all columns are returned.

`--start_row=START_ROW or -s=START_ROW`
An integer that specifies the number of rows to skip before showing table data. The default value is `0`; the table data starts at the first row.

`--table={true|false}` or `-t={true|false}`

To return an error if the command argument is not a table or view, set to `true`. The default value is `false`. This flag is supported for consistency with other commands.

TABLE
The table whose data you want to retrieve.

For more information about using the `bq head` command, see Managing table data.

bq help

Use the `bq help` command to display `bq` command-line tool documentation within the tool.

Synopsis

```
bq help [COMMAND]
```

Flags and arguments

The `bq help` command uses the following flags and arguments:

COMMAND
Specifies a particular `bq` command-line tool command that you want to get online help for.

bq insert

Use the `bq insert` command to insert rows of newline-delimited, JSON-formatted data into a table from a file using the streaming buffer. Data types are converted to match the column types of the destination table. This command is intended for testing purposes only. To stream data into BigQuery, use the `insertAll` API method.

Synopsis

```
bq insert [FLAGS] TABLE FILE
```

Examples

```
bq insert --ignore_unknown_values --template_suffix=_insert myDataset.myTable
/tmp/myData.json

echo '{"a":1, "b":2}' | bq insert myDataset.myTable
```

Flags and arguments

The `bq insert` command uses the following flags and arguments:

`--ignore_unknown_values={true|false}` or `-i={true|false}`
When set to `true`, BigQuery ignores any key-value pairs that do not match the table's schema, and inserts the row with the data that does match the schema. When set to `false`, rows with data that does not match the table's schema are not inserted. The default is `false`.

`--skip_invalid_rows={true|false}` or `-s={true|false}`
When set to `true`, BigQuery attempts to insert any valid rows, even if invalid rows are present. When set to `false`, the command fails if any invalid rows are present. The default is `false`.

`--template_suffix=SUFFIX or -x=SUFFIX`
When specified, treat the destination table *TABLE* as a base template, and insert the rows into an instance table named `{destination}{templateSuffix}`. BigQuery creates the instance table using the schema of the base template.

`TABLE`
The table that you want to insert data into.

`FILE`
The file containing the data that you want to insert.

For more information about using the `bq insert` command, see Streaming data into BigQuery.

`bq load`

Use the `bq load` command to load data into a table.

Synopsis

```
bq load [FLAGS] DESTINATION_TABLE SOURCE_DATA [SCHEMA]
```

Example

```
bq load myDataset.newTable gs://mybucket/info.csv ./info_schema.json
```

Flags and arguments

The `bq load` command uses the following flags and arguments:

`--allow_jagged_row={true|false}`
To allow missing trailing optional columns in CSV data, set to `true`.

`--allow_quoted_newlines={true|false}`
To allow quoted newlines in CSV data, set to `true`.

`--autodetect={true|false}`
To enable schema auto-detection for CSV and JSON data, set to `true`. The default is `false`. If `--autodetect` is `false`, and no schema is specified by using the `--schema` flag, and the destination table exists, then the schema of the destination table is used.

`--clustering_fields=COLUMNS`
A comma-separated list of up to four column names that specifies the fields to use for table clustering.

`--destination_kms_key=KEY`
Specifies a Cloud KMS key resource ID for encrypting the destination table data.

`--encoding=ENCODING_TYPE or -E=ENCODING_TYPE`
The character encoding used in the data. Use one of the following values:

- `ISO-8859-1` (also known as Latin-1)
- `UTF-8`

`--field_delimiter=DELIMITER or -F=DELIMITER`
Specifies the character that marks the boundary between columns in the data. The delimiter can be any ISO-8859-1 single-byte character. You can use either `\t` or `tab` to specify tab delimiters.

`--ignore_unknown_values={true|false}`
When set to `true`, for CSV and JSON files, rows with extra column values that do not match the table schema are ignored and are not loaded. Similarly, for Avro, Parquet and ORC files, fields in the file schema that do not exist in the table schema are ignored and are not loaded.

`--json_extension=JSON_TYPE`
Specifies the type of JSON file to load. Applies only to JSON files. Possible values are the following:

`GEOJSON` - newline-delimited GeoJSON file

To use this flag, the `--source_format` flag must be set to `NEWLINE_DELIMITED_JSON`.

For more information, see Loading newline-delimited GeoJSON files.

`--max_bad_records=MAX`
An integer that specifies the maximum number of bad records allowed before the entire job fails. The default value is `0`. At most, five errors of any type are returned regardless of the `--max_bad_records` value. This flag applies for loading CSV, JSON, and Sheets data only.

`--null_marker=STRING`
An optional custom string that represents a `NULL` value in CSV data.

`--projection_fields=PROPERTY_NAMES`
If you set `--source_format` to `DATASTORE_BACKUP`, then this flag indicates which entity properties to load from a Datastore export. Specify the property names in a comma-separated list. Property names are case sensitive and must refer to top-level properties. You can also use this flag with Firestore exports.

`--quote=CHARACTER`
Specifies a quote character to surround fields in CSV data. The `CHARACTER` argument can be any one-byte character. The default value is double quote (`"`). To specify that there is no quote character, use an empty string `""`.

`--replace={true|false}`
To erase any existing data and schema when new data is loaded, set to `true`. Any Cloud KMS key is also removed, unless you specify the `--destination_kms_key` flag. The default value is `false`.

Note: You can use the `TRUNCATE TABLE` statement to remove all rows from a table without deleting the schema.

`--schema={SCHEMA_FILE|SCHEMA}`
Specifies either the path to a local JSON schema file or a comma-separated list of column definitions in the form `FIELD:DATA_TYPE, FIELD:DATA_TYPE`, and so on. If you use a schema file, then do not give it an extension.

For example:

```
--schema=/tmp/tabledef

--schema=Region:STRING,Quarter:STRING,Total_sales:INTEGER
```

If no schema is specified, and `--autodetect` is `false`, and the destination table exists, then the schema of the destination table is used.

`--schema_update_option=OPTION`
When appending data to a table (in a load job or a query job), or when overwriting a table partition, specifies how to update the schema of the destination table. Use one of the following values:

- `ALLOW_FIELD_ADDITION`: Allow new fields to be added
- `ALLOW_FIELD_RELAXATION`: Allow relaxing `REQUIRED` fields to `NULLABLE`

Repeat this flag to specify multiple schema update options.

`--skip_leading_rows=NUMBER_OF_ROWS`
An integer that specifies the number of rows to skip at the beginning of the source file. The default is `0`.

`--source_format=FORMAT`
The format of the source data. Use one of the following values:

- `CSV`
- `NEWLINE_DELIMITED_JSON`
- `AVRO`
- `DATASTORE_BACKUP`
- `PARQUET`
- `ORC`

`--time_partitioning_expiration=SECONDS`
An integer that specifies (in seconds) when a time-based partition should be deleted. The expiration time evaluates to the partition's UTC date plus the integer value. A negative number indicates no expiration.

`--time_partitioning_field=COLUMN_NAME`
Specifies the field that determines how to create a time-based partition. If time-based partitioning is enabled without this value, then the table is partitioned based on the load time.

`--time_partitioning_type=INTERVAL`
Enables time-based partitioning on a table and sets the partition type. Use one of the following values:

- `DAY`
- `HOUR`
- `MONTH`
- `YEAR`

The default partition type for time-based partitioning is `DAY`.

`--use_avro_logical_types={true|false}`
If the `--source_format` flag is set to `AVRO`, then set this flag to `true` to convert logical types into their corresponding types (such as `TIMESTAMP`) instead of only using their raw types (such as `INTEGER`).

`--decimal_target_types=DECIMAL_TYPE`
Determines how to convert a `Decimal` logical type. Equivalent to JobConfigurationLoad.decimalTargetTypes. Repeat this flag to specify multiple target types.

`--parquet_enum_as_string={true|false}`
If the `--source_format` flag is set to `PARQUET`, and you want BigQuery to infer Parquet `ENUM` logical types as `STRING` values, then set this flag to `true`. The default is `false`.

`--parquet_enable_list_inference={true|false}`
If the `--source_format` flag is set to `PARQUET`, then this flag indicates whether to use schema inference for Parquet `LIST` logical types.

`DESTINATION_TABLE`
The table that you want to load data into.

`SOURCE_DATA`
The Cloud Storage URI of the file that contains the data that you want to load.

`SCHEMA`
The schema for the destination table.

For more information about loading data from Cloud Storage using the `bq load` command, see the following:

- Loading Avro data
- Loading CSV data
- Loading JSON data
- Loading ORC data
- Loading Parquet data
- Loading data from Datastore exports
- Loading data from Firestore exports

For more information about loading data from a local source using the `bq load` command, see the following:

Loading data from local files.

`bq ls`

Use the `bq ls` command to list objects in a collection.

Synopsis

```
bq ls [FLAGS] [RESOURCE]
```

Example

```
bq ls myDataset
```

Flags and arguments

The `bq ls` command uses the following flags and arguments:

`--all={true|false}` or **`-a={true|false}`**
To show all results, set to `true`. Shows jobs from all users or all datasets, including hidden ones. This flag is not needed when listing transfer configurations or transfer runs. The default value is `false`.

`--capacity_commitment={true|false}`
To list capacity commitments, set to `true`. The default value is `false`.

`--datasets={true|false}` or **`-d={true|false}`**
To list datasets, set to `true`. The default value is `false`.

`--filter="FILTER"`
Lists datasets that match the `FILTER` argument, which consists of one or more space-separated triples in the format `labels.KEY:VALUE`. If more than one triple is provided, then the command only returns datasets matching *all* of the triples (i.e., the command uses the `AND` logical operator, not `OR`). If you want to specify more than one triple, then surround the `FILTER` value with quotation marks.

- To filter based on dataset labels, use the keys and values that you applied to your datasets.

  For example:

  ```
  --filter "labels.department:marketing labels.team:sales"
  ```

- To filter based on transfer configurations, use `dataSourceIds` as the key, and one of the following data sources as the value:

- `amazon_s3` - Amazon S3 data transfer
- `dcm_dt` - Campaign Manager data transfer
- `google_cloud_storage` - Cloud Storage data transfer

- `cross_region_copy` - Dataset copy
- `dfp_dt` - Google Ad Manager data transfer
- `adwords` - Google Ads data transfer
- `merchant_center` - Google Merchant Center data transfer
- `play` - Google Play data transfer
- `doubleclick_search` - Search Ads 360 data transfer
- `youtube_channel` - YouTube Channel data transfer
- `youtube_content_owner` - YouTube Content Owner data transfer
- `redshift` - Amazon Redshift migration
- `on_premises` - Teradata migration

For example:

```
--filter labels.dataSourceIds:dcm_dt
```

To filter based on transfer runs, use `states` as the key, and one of the following transfer states as the value:

- `SUCCEEDED`
- `FAILED`
- `PENDING`
- `RUNNING`
- `CANCELLED`

For example:

```
--filter labels.states:FAILED
```

`--jobs={true|false}` or `-j={true|false}`
To list jobs, set to `true`. The default value is `false`. By default, you are limited to 100,000 results.

`--max_creation_time=MAX_CREATION_TIME_MS`
An integer that represents a timestamp in milliseconds. When specified with the `--jobs` flag, this flag lists only the jobs created before the timestamp.

`--max_results=MAX_RESULTS or -n=MAX_RESULTS`
An integer indicating the maximum number of results. The default value is 50.

-- `min_creation_time=MIN_CREATION_TIME_MS`
An integer that represents a timestamp in milliseconds. When specified with the `--jobs` flag, this flag lists only the jobs created after the timestamp.

`--message_type=messageTypes:MESSAGE_TYPE`
To list only transfer run log messages of a particular type, specify `messageTypes:MESSAGE_TYPE`. Possible values are the following:

- `INFO`
- `WARNING`
- `ERROR`

`--models={true|false}` or **`-m={true|false}`**
To list BigQuery ML models, set to `true`. The default value is `false`.

`--page_token=TOKEN` or **`-k=TOKEN`**
Lists items starting from the specified page token.

`--projects={true|false}` or **`-p={true|false}`**
To show all projects, set to `true`. The default value is `false`.

`--reservation={true|false}`
To list all reservations for a given project and location, set to `true`. The default value is `false`. Use with the `--project_id` and `--location` flags.

For example:

```
bq ls --reservation=true --project_id=myProject --location=us
```

`--reservation_assignment={true|false}`
To list all reservation assignments for a given project and location, set to `true`. The default value is `false`. Use with the `--project_id` and `--location` flags.

`--row_access_policies`
When specified, lists all the row-level access policies on a table. Row-level access policies are used for row-level security. You must supply the table name in the format `dataset.table`.

`--run_attempt=RUN_ATTEMPT`
Use with the `--transfer_run` flag. To list all run attempts for the specified transfer run, set to `RUN_ATTEMPT_UNSPECIFIED`. To list only the latest run attempt, set to `LATEST`. The default is `LATEST`.

`--transfer_config={true|false}`
To list transfer configurations in the specified project and location, set to `true`. Use with the `--transfer_location` and `--project_id` flag. The default value is `false`.

`--transfer_location=LOCATION`
List transfer configurations in the specified location. You set the transfer location when the transfer is created.

`--transfer_log={true|false}`
Use with the `--transfer_run` flag. To list transfer log messages for the specified transfer run, set to `true`. The default value is `false`.

`--transfer_run={true|false}`
Lists the transfer runs for the specified transfer configuration.

For example:

```
bq ls --transfer_run=true projects/myProject/locations/us/transferConfigs/12345
```

`RESOURCE`

The collection whose objects that you want to list. The resource can be a dataset, project, reservation, or transfer configuration.

For more information about using the `bq ls` command, see the following:

- Managing jobs
- Listing datasets in a project
- Creating and using tables
- Listing views in a dataset
- Working with transfers
- Working with Reservations
- Listing table snapshots in a dataset

`bq mk`

Use the `bq mk` command to create a BigQuery resource.

Synopsis

```
bq mk TYPE_FLAG [OTHER FLAGS] [ARGS]
```

Flags and arguments

The `bq mk` command takes a *type* flag that specifies the type of resource to create, and additional flags that depend on the resource type.

`TYPE_FLAG` **:** Set one of the following flags to `true`. Your selection specifies the type of resource to create.

- `--capacity_commitment`: Purchase a capacity commitment.
- `--connection`: Create a connection.
- `--dataset` or `-d` : Create a dataset.
- `--materialized_view`: Create a materialized view.
- `--reservation`: Create a reservation.
- `--reservation_assignment`. Assign a folder, project, or organization to a reservation.
- `--table` or `-t` : Create a table.
- `--transfer_config`: Create a transfer configuration.
- `--transfer_run`: Create a transfer run for a time range.
- `--view`: Create a view.

The `bq mk` command supports the following flag for all types of resources:

`--force={true|false}` or `-f={true|false}`
To ignore errors if a resource with the same name already exists, set to `true`. If the resource already exists, then the exit code is 0, but setting this flag to `true` does not cause the `bq mk` command to overwrite the resource. The default value is `false`.

The `bq mk` command supports additional flags, depending on the type of resource you are creating, as described in the following sections.

`bq mk --capacity_commitment`

Purchase a capacity commitment. The following flags are supported:

`--location=LOCATION`
Specifies the location of the project.

`--plan=PLAN_TYPE`
Specifies the plan type. One of the following:

- `FLEX`
- `MONTHLY`
- `ANNUAL`

`--project_id=PROJECT_ID`
Specifies the project that administers the slots.

`--slots=NUMBER_OF_SLOTS`
Specifies the number of slots to purchase.

For more information, see Working with commitments.

`bq mk --connection`

Creates a connection. The following flags are supported:

`--connection_type=CONNECTION_TYPE`
The type of the connection, for example `CLOUD_SQL` for Cloud SQL connections.

`--properties=PROPERTIES`
Connection specific parameters in JSON format. `instanceId`, `database` and `type` must be specified.

`--connection_credential=CONNECTION_CREDENTIAL`
The credentials of the connection in JSON format. `username` and `password` must be specified.

`--project_id=PROJECT_ID`
Specifies the ID of the project that the connection belongs to.

`--location=LOCATION`
Specifies the location that the connection will be stored.

`--display_name=DISPLAY_NAME`
Specifies an optional friendly name for the connection.

`--description=DESCRIPTION`

Specifies an optional description of the connection.

`CONNECTION_ID`
Specifies an optional connection id for the connection. If a connection id is not provided a unique id is automatically generated. The connection id can contain letters, numbers and underscores.

For more information, see Creating connections.

`bq mk --dataset`

Creates a dataset. The following flags are supported:

`--default_kms_key=KEY`
Specifies the default Cloud KMS key resource ID for encrypting the table data in a dataset if no explicit key is provided during table creation or query.

`--default_partition_expiration=SECONDS`
An integer that specifies the default expiration time, in seconds, for all partitions in newly-created partitioned tables in the dataset. A partition's expiration time is set to the partition's UTC date plus the integer value. If this property is set, then its value overrides the dataset-level default table expiration if it exists. If you supply the `--time_partitioning_expiration` flag when you create or update a partitioned table, then the table-level partition expiration takes precedence over the dataset-level default partition expiration.

`--default_table_expiration=SECONDS`
An integer that specifies the default lifetime, in seconds, for newly created tables in a dataset. The expiration time is set to the current UTC time plus this value.

`--description=DESCRIPTION`
Specifies the description of the dataset.

`--label=KEY:VALUE`
Specifies a label for the dataset. Repeat this flag to specify multiple labels.

`--location=LOCATION` **or** `--data_location=LOCATION`
Specifies the location of the dataset. Prefer the `--location` flag; the `--data_location` flag is a legacy flag.

For more information, see Creating datasets.

`bq mk --materialized_view`

Creates a materialized view. The following flags are supported:

`--enable_refresh={true|false}`
To disable automatic refresh for a materialized view, set to `false`. The default when creating a materialized view is `true`.

`--refresh_interval_ms=MILLISECONDS`
Specifies the number of milliseconds for the refresh interval of a materialized view. If this flag is not specified, then the default refresh interval for a materialized view that has

refresh enabled is 1,800,000 milliseconds, which is 30 minutes.

For more information, see Creating and using materialized views.

`bq mk --reservation`

Creates a reservation with dedicated slots. The following flags are supported:

`--ignore_idle_slots={true|false}`
To restrict jobs running in this reservation to only use slots allocated to the reservation, set to `true`. The default value is `false`; jobs in this reservation can use idle slots from other reservations, or slots that are not allocated to any reservation. For more information, see Idle slots.

`--location=LOCATION`
Specifies the location of the project.

`--project_id=PROJECT_ID`
Specifies the project that owns the reservation.

`--slots=NUMBER_OF_SLOTS`
Specifies the number of slots to allocate to this reservation.

For more information, see Working with reservations.

`bq mk --reservation_assignment`

Assigns a project, folder, or organization to a reservation. The following flags are supported:

`--assignee_id=ASSIGNEE_ID`
Specifies the ID of the folder, organization, or project.

`--assignee_type=ASSIGNEE_TYPE`
Specifies the type of entity to assign to the reservation. One of the following:

- `FOLDER`
- `ORGANIZATION`
- `PROJECT`

`--job_type=JOB_TYPE`
Specifies the type of job to assign to the reservation. One of the following:

- `QUERY`
- `PIPELINE`
- `ML_EXTERNAL`

`--location=LOCATION`
Specifies the location of the project.

`--project_id=PROJECT_ID`
Specifies the project that owns the reservation.

`--reservation_id=RESERVATION_ID`
Specifies the ID of the reservation.

For more information, see Working with assignments.

`bq mk --table`

Creates a table. The following flags are supported:

`--clustering_fields=COLUMNS`
A comma-separated list of up to four column names that specifies the fields to use for table clustering. If specified with partitioning, then the table is first partitioned, and then each partition is clustered using the supplied columns.

`--description=DESCRIPTION`
Specifies the description of the table.

`--destination_kms_key=KEY`
Specifies a Cloud KMS key resource ID for encrypting the destination table data.

`--expiration=SECONDS`
Specifies the lifetime for the table. If `SECONDS` is `0`, then the table doesn't expire. If you don't specify the `--expiration` flag, then BigQuery creates the table with the dataset's default table lifetime.

`--external_table_definition={PATH_TO_FILE|DEFINITION}`
Specifies a table definition for creating an external table. The value can be either a path to a file containing a table definition file (`PATH_TO_FILE`) or an inline table definition (`DEFINITION`).

- The format for the `DEFINITION` field is `SCHEMA@FORMAT=URI` .
- The format for the `SCHEMA` value is a comma-separated list of column definitions in the form `FIELD:DATA_TYPE, FIELD:DATA_TYPE` , and so on. You can omit the `SCHEMA` value if the data format is self-describing (such as Avro) or if you are using schema auto-detection. .
- The `FORMAT` value specifies the data format, such as `CSV` , `AVRO` , or `PARQUET` .

If you specify a table definition file, then do not give it an extension.

For example:

```
--external_table_definition=/tmp/tabledef

--
external_table_definition=Region:STRING,Quarter:STRING,Total_sales:INTEGER@CSV=gs:/
```

`--label=KEY:VALUE`
Specifies a label for the table. Repeat this flag to specify multiple labels.

`--range_partitioning=COLUMN_NAME,START,END,INTERVAL`
Specifies options for an integer-range partition, as follows:

- `column_name` is the column used to create the integer range partitions.
- `start` is the start of range partitioning, inclusive.
- `end` is the end of range partitioning, exclusive.

- `interval` is the width of each range within the partition.

For example:

```
--range_partitioning=customer_id,0,10000,100
```

`--require_partition_filter={true|false}`
To require a partition filter for queries over the supplied table, set to `true`. This flag only applies to partitioned tables. The default value is `false`.

`--schema={`*`SCHEMA_FILE|SCHEMA`*`}`
Specifies either the path to a local JSON schema file or a comma-separated list of column definitions in the form *`FIELD:DATA_TYPE, FIELD:DATA_TYPE`*, and so on. If you use a schema file, then do not give it an extension.

Examples:

```
--schema=/tmp/tabledef
```

```
--schema=Region:STRING,Quarter:STRING,Total_sales:INTEGER
```

`--time_partitioning_expiration=`*`SECONDS`*
An integer that specifies (in seconds) when a time-based partition should be deleted. The expiration time evaluates to the partition's UTC date plus the integer value. A negative number indicates no expiration.

`--time_partitioning_field=`*`COLUMN_NAME`*
Specifies the field used to determine how to create a time-based partition. If time-based partitioning is enabled without this value, then the table is partitioned based on the load time.

`--time_partitioning_type=`*`INTERVAL`*
Enables time-based partitioning on a table and sets the partition type. Use one of the following values:

- `DAY`
- `HOUR`
- `MONTH`
- `YEAR`

`--use_avro_logical_types={true|false}`
If the *`FORMAT`* part of the `--external_table_definition` flag is set to `AVRO`, then this flag specifies whether to convert logical types into their corresponding types (such as `TIMESTAMP`) instead of only using their raw types (such as `INTEGER`).

`--parquet_enable_list_inference={true|false}`
If the *`FORMAT`* part of the `--external_table_definition` flag is set to `PARQUET`, then this flag specifies whether to use schema inference for Parquet `LIST` logical types.

`--parquet_enum_as_string={true|false}`

If the `FORMAT` part of the `--external_table_definition` flag is set to `PARQUET`, then this flag specifies whether to infer Parquet `ENUM` logical types as `STRING` values.

For more information, see Creating and using tables.

`bq mk --transfer_config`

Creates a transfer configuation. The following flags are supported:

`--data_source=DATA_SOURCE`
Specifies the data source. Required when creating a transfer configuration. Use one of the following values:

- `amazon_s3` - Amazon S3 data transfer
- `dcm_dt` - Campaign Manager data transfer
- `google_cloud_storage` - Cloud Storage data transfer
- `cross_region_copy` - Dataset copy
- `dfp_dt` - Google Ad Manager data transfer
- `adwords` - Google Ads data transfer
- `merchant_center` - Google Merchant Center data transfer
- `play` - Google Play data transfer
- `doubleclick_search` - Search Ads 360 data transfer
- `youtube_channel` - YouTube Channel data transfer
- `youtube_content_owner` - YouTube Content Owner data transfer
- `redshift` - Amazon Redshift migration
- `on_premises` - Teradata migration

Note: The `redshift` and `on_premises` values are for data migrations; before you use the `bq mk --transfer_config` command with these values, consult the linked documentation from the preceding list.

`--display_name=DISPLAY_NAME`
Specifies the display name for the transfer configuration.

`--params={"PARAMETER":"VALUE"}` or **`-p={"PARAMETER":"VALUE"}`**
Specifies the parameters for the transfer configuration in JSON format. The parameters vary depending on the data source. For more information, see Introduction to BigQuery Data Transfer Service.

`--refresh_window_days=DAYS`
An integer that specifies the refresh window for the transfer configuration in days. The default value is `0`.

`--target_dataset=DATASET`
Specifies the target dataset for the transfer configuration.

For information about using the `bq mk` command with the BigQuery Data Transfer Service, see the following:

- Setting up an Amazon S3 transfer
- Setting up a Campaign Manager transfer

- Setting up a Cloud Storage transfer
- Setting up a Google Ad Manager transfer
- Setting up a Google Ads transfer
- Setting up a Google Merchant Center transfer (beta)
- Setting up a Google Play transfer
- Setting up a Search Ads 360 transfer (beta)
- Setting up a YouTube Channel transfer
- Setting up a YouTube Content Owner transfer
- Migrating data from Amazon Redshift
- Migrating data from Teradata

`bq mk --transfer_run`

Creates a data transfer run at the specified time or time range using the specified data transfer configuration.

Synopsis

```
bq mk --transfer_run [--run_time=RUN_TIME | --start_time=START_TIME --
end_time=END_TIME] CONFIG
```

The following flags are supported:

`--run_time=RUN_TIME`
A timestamp that specifies the time to schedule the data transfer run.

`--start_time=START_TIME`
A timestamp that specifies the start time for a range of data transfer runs.

`--end_time=END_TIME`
A timestamp that specifies the end time for a range of data transfer runs.

The format for the timestamps is RFC3339 UTC "Zulu".

The `CONFIG` argument specifies a preexisting data transfer configuration.

Examples

```
bq mk --transfer_run \
      --run_time=2021-01-20T17:00:00.00Z \
      projects/p/locations/l/transferConfigs/c

bq mk --transfer_run \
      --start_time=2020-12-19T16:39:57-08:00 \
      --end_time=2020-12-19T20:39:57-08:00 \
      projects/p/locations/l/transferConfigs/c
```

`bq mk --view`

Creates a view. The following flags are supported:

`--description=DESCRIPTION`
Specifies the description of the view.

`--expiration=SECONDS`

Specifies the lifetime for the view. If *SECONDS* is `0`, then the view doesn't expire. If you don't specify the `--expiration` flag, then BigQuery creates the view with the dataset's default table lifetime.

`--label=KEY:VALUE`
Specifies a label for the view. Repeat this flag to specify multiple labels.

`--use_legacy_sql={true|false}`
Set to `false` to use a Standard SQL query to create a view. The default value is `true`; uses legacy SQL.

`--view_udf_resource=FILE`
Specifies the Cloud Storage URI or the path to a local code file that is loaded and evaluated immediately as a user-defined function resource used by a view's SQL query. Repeat this flag to specify multiple files.

For more information, see Creating views.

`bq mkdef`

Use the `bq mkdef` command to create a table definition in JSON format for data stored in Cloud Storage or Drive.

Synopsis

```
bq mkdef [FLAGS] URI [ > FILE ]
```

Flags and arguments

The `bq mkdef` command uses the following flags and arguments:

`--autodetect={true|false}`
Specifies whether to use schema auto-detection for CSV and JSON data. The default is `false`.

`--ignore_unknown_values={true|false}` or **`-i={true|false}`**
Specifies whether to ignore any values in a row that are not present in the schema. The default is `false`.

`--parquet_enable_list_inference={true|false}`
If `source_format` is set to `PARQUET`, then this flag specifies whether to use schema inference for Parquet `LIST` logical types. The default is `false`.

`--parquet_enum_as_string={true|false}`
If `source_format` is set to `PARQUET`, then this flag specifies whether to infer Parquet `ENUM` logical types as `STRING` values. The default is `false`.

`--source_format=FORMAT`
Specifies the format of the source data. Use one of the following values:

- `AVRO`
- `CSV`
- `DATASTORE_BACKUP`

- `GOOGLE_SHEETS`
- `NEWLINE_DELIMITED_JSON`
- `ORC`
- `PARQUET`

The default value is `CSV`.

`--use_avro_logical_types={true|false}`
If the `--source_format` flag is set to `AVRO`, then this flag specifies whether to convert logical types into their corresponding types (such as `TIMESTAMP`) instead of only using their raw types (such as `INTEGER`). The default is `false`.

For more information about using the `bq mkdef` command, see Creating a table definition file for an external data source.

`bq partition`

Use the `bq partition` command to convert a group of tables with time-unit suffixes, such as tables ending in *`YYYYMMDD`* for date partitioning, into partitioned tables.

Synopsis

```
bq partition [FLAGS] SOURCE_TABLE_BASE_NAME PARTITION_TABLE
```

Flags and arguments

The `bq partition` command uses the following flags and arguments:

`--no_clobber={true|false}` **or** **`-n={true|false}`**
To disallow overwriting an existing partition, set to `true`. The default value is `false`; if the partition exists, then it is overwritten.

`--time_partitioning_expiration=SECONDS`
An integer that specifies (in seconds) when a time-based partition should be deleted. The expiration time evaluates to the partition's UTC date plus the integer value. A negative number indicates no expiration.

`--time_partitioning_type=INTERVAL`
Specifies the partition type. The following table provides the possible values for the *`INTERVAL`* flag and the expected time-unit-suffix format for each:

`INTERVAL`	Suffix
`HOUR`	*`YYYYMMDDHH`*
`DAY`	*`YYYYMMDD`*
`MONTH`	*`YYYYMM`*
`YEAR`	*`YYYY`*

`SOURCE_TABLE_BASE_NAME`

The base name of the group of tables with time-unit suffixes.

PARTITION_TABLE
The name of the destination partitioned table.

For more information about using the `bq partition` command, see Converting date-sharded tables into ingestion-time partitioned tables.

bq query

Use the `bq query` command to create a query job that runs the specified SQL query.

Synopsis

```
bq query [FLAGS] 'QUERY'
```

Flags and arguments

The `bq query` command uses the following flags and arguments:

--allow_large_results={true|false}
To enable large destination table sizes for legacy SQL queries, set to `true`. The default value is `false`.

--append_table={true|false}
To append data to a destination table, set to `true`. The default value is `false`.

--batch={true|false}
To run the query in batch mode, set to `true`. The default value is `false`.

--clustering_fields=*COLUMNS*
A comma-separated list of up to four column names that specifies fields to use to cluster the destination table in a query. If specified with partitioning, then the table is first partitioned, and then each partition is clustered using the supplied columns.

--destination_kms_key=*KEY*
Specifies a Cloud KMS key resource ID for encrypting the destination table data.

--destination_schema={*PATH_TO_FILE*|*SCHEMA*}
The path to a local JSON schema file or a comma-separated list of column definitions in the form *FIELD*:*DATA_TYPE*, *FIELD*:*DATA_TYPE*, and so on.

--destination_table=*TABLE*
When specified, the query results are saved to `TABLE`. Specify `TABLE` in the following format: `PROJECT`:`DATASET`.`TABLE`. If `PROJECT` is not specified, then the current project is assumed. If the `--destination_table` flag is not specified, then the query results are saved to a temporary table.

Examples:

```
--destination_table myProject:myDataset.myTable
```

```
--destination_table myDataset.myTable
```

`--dry_run={true|false}`
When specified, the query is validated but not run.

`--external_table_definition={TABLE::PATH_TO_FILE|TABLE::DEFINITION}`
Specifies the table name and table definition for an external table query. The table definition can be a path to a local JSON schema file or an inline table definition. The format for supplying the inline table definition is `SCHEMA@SOURCE_FORMAT=CLOUD_STORAGE_URI`. The format for the `SCHEMA` value is a comma-separated list of column definitions in the form `FIELD:DATA_TYPE, FIELD:DATA_TYPE`, and so on. If you use a table definition file, then do not give it an extension.

For example:

```
--external_table_definition=myTable::/tmp/tabledef

--
external_table_definition=myTable::Region:STRING,Quarter:STRING,Total_sales:INTEGER
```

Repeat this flag to query multiple tables.

`--flatten_results={true|false}`
To disallow flattening nested and repeated fields in the results for legacy SQL queries, set to `false`. The default value is `true`.

`--label=KEY:VALUE`
Specifies a label for the query job. Repeat this flag to specify multiple labels.

`--max_rows=MAX_ROWS` or `-n=MAX_ROWS`
An integer specifying the number of rows to return in the query results. The default value is `100`.

`--maximum_bytes_billed=MAX_BYTES`
An integer that limits the bytes billed for the query. If the query goes beyond the limit, then the query fails (without incurring a charge). If this flag is not specified, then the bytes billed is set to the project default.

`--min_completion_ratio=RATIO`
[Experimental] A number from 0 through 1.0 that specifies the minimum fraction of data that must be scanned before a query returns. If the flag is not specified, then the default server value `1.0` is used.

`--parameter={PATH_TO_FILE|PARAMETER}`
Either a JSON file containing a list of query parameters, or a query parameter in the form `NAME:TYPE:VALUE`. An empty name creates a positional parameter. If `TYPE` is omitted, then the `STRING` type is assumed. `NULL` specifies a null value. Repeat this flag to specify multiple parameters.

For example:

```
--parameter=/tmp/queryParams

--parameter=Name::Oscar

--parameter=Count:INTEGER:42
```

`--range_partitioning=COLUMN_NAME,START,END,INTERVAL`
Use with the `--destination_table` flag. Specifies options for integer-range partitioning in the destination table. The value is a comma-separated list of the form `column_name,start,end,interval`, where

- `column_name` is the column used to create the integer range partitions.
- `start` is the start of range partitioning, inclusive.
- `end` is the end of range partitioning, exclusive.
- `interval` is the width of each range within the partition.

For example:

```
--range_partitioning=customer_id,0,10000,100
```

`--replace={true|false}`
To overwrite the destination table with the query results, set to `true`. Any existing data and schema are erased. Any Cloud KMS key is also removed, unless you specify the `--destination_kms_key` flag. The default value is `false`.

Note: To remove all rows from a table without deleting the schema, use the `TRUNCATE TABLE` statement.

`--require_cache={true|false}`
If specified, then run the query only if results can be retrieved from the cache.

`--require_partition_filter={true|false}`
If specified, then a partition filter is required for queries over the supplied table. This flag can only be used with a partitioned table.

`--rpc={true|false}`
To use the RPC-style query API instead of the REST API `jobs.insert` method, set to `true`. The default value is `false`.

`--schedule="SCHEDULE"`
Makes a query a recurring scheduled query. A schedule for how often the query should run is required.

Examples:

```
--schedule="every 24 hours"

--schedule="every 3 hours"
```

For a description of the schedule syntax, see Formatting the schedule.

`--schema_update_option=OPTION`

When appending data to a table in a load job or a query job, or when overwriting a table partition, specifies how to update the schema of the destination table. Use one of the following values:

- `ALLOW_FIELD_ADDITION` : Allow new fields to be added.
- `ALLOW_FIELD_RELAXATION` : Allow relaxing `REQUIRED` fields to `NULLABLE` .

Repeat this flag to specify multiple schema update options.

`--start_row=ROW_NUMBER` **or** **`-s=ROW_NUMBER`**
An integer that specifies the first row to return in the query result. The default value is `0` .

`--target_dataset=DATASET`
When specified with `--schedule` , updates the target dataset for a scheduled query. The query must be DDL or DML.

`--time_partitioning_expiration=SECONDS`
Use with the `--destination_table` flag. An integer that specifies (in seconds) when a time-based partition should be deleted. The expiration time evaluates to the partition's UTC date plus the integer value. A negative number indicates no expiration.

`--time_partitioning_field=COLUMN_NAME`
Use with the `--destination_table` flag. Specifies the partitioning column for time-based partitioning. If time-based partitioning is enabled without this value, then the table is partitioned based on the ingestion time.

`--time_partitioning_type=INTERVAL`
Use with the `--destination_table` flag. Specifies the partition type for the destination table. Use one of the following values:

- `DAY`
- `HOUR`
- `MONTH`
- `YEAR`

`--udf_resource=FILE`
This flag applies only to legacy SQL queries. Specifies the Cloud Storage URI or the path to a local file containing a user-defined function resource to be used by a legacy SQL query. Repeat this flag to specify multiple files.

`--use_cache={true|false}`
To disallow caching query results, set to `false` . The default value is `true` .

`--use_legacy_sql={true|false}`
To run a Standard SQL query, set to `false` . The default value is `true` ; the command uses legacy SQL.

`QUERY`
The query that you want to run.

For more information about using the `bq query` command, see Running interactive and batch queries.

`bq remove-iam-policy-binding`

Use the `bq remove-iam-policy-binding` command to retrieve the IAM policy for a resource and remove a binding from the policy, in one step. The resource can be a table or a view.

This command is an alternative to the following three-step process:

1. Using the `bq get-iam-policy` command to retrieve the policy file (in JSON format).
2. Editing the policy file.
3. Using the `bq set-iam-policy` command to update the policy without the binding.

Synopsis

```
bq remove-iam-policy-binding FLAGS --member=MEMBER_TYPE:MEMBER --role=ROLE RESOURCE
```

Flags and arguments

The `bq remove-iam-policy-binding` command uses the following flags and arguments:

`--member=`*`MEMBER_TYPE:MEMBER`*
Required. Use the `--member` flag to specify the member part of the IAM policy binding. The `--member` flag is required along with the `--role` flag. One combination of `--member` and `--role` flags equals one binding.

The *`MEMBER_TYPE`* value specifies the type of member in the IAM policy binding. Use one of the following values:

- `user`
- `serviceAccount`
- `group`
- `domain`

The *`MEMBER`* value specifies the email address or domain of the member in the IAM policy binding.

`--role=`*`ROLE`*
Required. Specifies the role part of the IAM policy binding. The `--role` flag is required along with the `--member` flag. One combination of `--member` and `--role` flags equals one binding.

`--table={true|false}` **or** **`-t={true|false}`**
Optional. To remove a binding from the IAM policy of a table or view, set to `true`. The default value is `false`.

`RESOURCE` is the table or view whose policy binding you want to remove.

For more information, see the IAM policy reference.

`bq rm`

Use the `bq rm` command to delete a BigQuery resource.

Synopsis

```
bq rm [FLAGS] RESOURCE
```

Flags and arguments

The `bq rm` command uses the following flags and arguments:

`--capacity_commitment={false|true}`
To delete a capacity commitment, set to `true`. The default value is `false`.

`--dataset={true|false}` or **`-d={true|false}`**
To delete a dataset, set to `true`. The default value is `false`.

`--force={true|false}` or **`-f={true|false}`**
To delete a resource without prompting, set to `true`. The default value is `false`.

`--model={true|false}` or **`-m={true|false}`**
To delete a BigQuery ML model, set to `true`. The default is `false`.

`--recursive={true|false}` or **`-r{true|false}`**
To delete a dataset and any tables, table data, or models in it, set to `true`. The default value is `false`.

`--reservation={true|false}`
To delete a reservation, set to `true`. The default value is `false`.

`--reservation_assignment={true|false}`
To delete a reservation assignment, set to `true`. The default value is `false`.

`--table={true|false}` or **`-t={true|false}`**
To delete a table, set to `true`. The default value is `false`.

`--transfer_config={true|false}`
To delete a transfer configuration, set to `true`. The default value is `false`.

`RESOURCE`
The resource that you want to remove.

For more information about using the `bq rm` command, see the following:

`bq set-iam-policy`

Use the `bq set-iam-policy` command to specify or update the IAM policy for a resource. The resource can be a table or a view. After setting the policy, the new policy is printed to `stdout`. The policy is in JSON format.

The `etag` field in the updated policy must match the `etag` value of the current policy, otherwise the update fails. This feature prevents concurrent updates.

You can obtain the current policy and `etag` value for a resource by using the `bq get-iam-policy` command.

Synopsis

```
bq set-iam-policy [FLAGS] RESOURCE FILE_NAME
```

Flags and arguments

The `bq set-iam-policy` command uses the following flags and arguments.

`--table={true|false}` or `-t={true|false}`
Optional. To set the IAM policy of a table or view, set to `true`. The default value is `false`.

RESOURCE is the table or view whose policy you want to update.

FILE_NAME is the name of a file containing the policy in JSON format.

For more information about the `bq set-iam-policy` command, with examples, see Introduction to table access controls.

`bq show`

Use the `bq show` command to display information about a resource.

Synopsis

```
bq show [FLAGS] [RESOURCE]
```

Flags and arguments

The `bq show` command uses the following flags and arguments:

`--assignee_id=ASSIGNEE`
When used with the `--reservation_assignment` flag, specifies the ID of a folder, organization, or project. Use the `--assignee_type` flag to specify which type of assignee to show.

`--assignee_type=TYPE`
When used with the `--reservation_assignment` flag, specifies the type of entity to show. Use one of the following values:

- `FOLDER`
- `ORGANIZATION`
- `PROJECT`

`--connection={true|false}`

To show information about a connection, set to `true`. The default value is `false`. For more information, see Viewing a connection resource.

`--dataset={true|false}` or **`-d={true|false}`**
To show information about a dataset, set to `true`. The default value is `false`.

`--encryption_service_account={true|false}`
To show the encryption service account for a project, if it exists, or create one if it doesn't exist, set to `true`. The default value is `false`. Use with the `--project_id` flag.

`--job={true|false}` or **`-j={true|false}`**
To show information about a job, set to `true`. The default value is `false`.

`--job_type=JOB_TYPE`
When used with the `--reservation_assignment` flag, specifies the job type of the reservation assignments you want to show. Use one of the following values:

- `QUERY`
- `PIPELINE`
- `ML_EXTERNAL`

`--model={true|false}` or **`-m={true|false}`**
To show information about a BigQuery ML model, set to `true`. The default value is `false`.

`--reservation={true|false}`
To show information about a reservation, set to `true`. The default value is `false`.

`--reservation_assignment={true|false}`
When set to `true`, the command displays reservation assignments for a specified folder, organization, or project. The command displays the target resource's explicit assignments, if any; otherwise, displays assignments inherited from parent resources. For example, a project might inherit assignments from its parent folder. When using this flag, the `--job_type`, `--assignee_type`, and `--assignee_id` flags apply. The default value is `false`.

`--schema={true|false}`
To display only the table's schema, set to `true`. The default value is `false`.

`--transfer_config={true|false}`
To display information about a transfer configuration, set to `true`. The default value is `false`.

`--transfer_run={true|false}`
To display information about a transfer run, set to `true`. The default value is `false`.

`--view={true|false}`
To display information about a view, set to `true`. The default value is `false`.

`RESOURCE`
The resource whose information you want to show.

For more information about using the `bq show` command, see the following:

`bq update`

Use the `bq update` command to change a resource.

Synopsis

```
bq update [FLAGS] [RESOURCE]
```

Flags and arguments

The `bq update` command uses the following flags and arguments:

`--capacity_commitment={true|false}`
To update a capacity commitment, set to `true`. The default value is `false`. Use this flag with the `--merge`, `--plan`, `--renewal_plan`, `--split`, and `--slots` flags.

`--clear_label=KEY:VALUE`
Removes a label from the resource. Use the format `KEY:VALUE` to specify the label to remove. Repeat this flag to remove multiple labels.

`--clustering_fields=COLUMNS`
Updates a table's clustering specification. The *COLUMNS* value is a comma-separated list of column names to use for clustering. To remove the clustering, set *COLUMNS* to `""` (the empty string). For more information, see Modifying clustering specification.

`--dataset={true|false}` or **`-d={true|false}`**
To update a dataset, set to `true`. The default value is `false`.

`--default_kms_key=KEY`
Specifies the default Cloud KMS key resource ID for encrypting table data in a dataset. The default key is used if no explicit key is provided for a table creation or a query.

`--default_partition_expiration=SECONDS`
An integer that specifies the default expiration time, in seconds, for all partitions in newly created partitioned tables in the dataset. This flag has no minimum value.

A partition's expiration time is set to the partition's UTC date plus the integer value. If this property is set, then it overrides the dataset-level default table expiration if it exists. If you supply the `--time_partitioning_expiration` flag when you create or update a partitioned table, then the table-level partition expiration takes precedence over the dataset-level default partition expiration. Specify `0` to remove an existing expiration.

`--default_table_expiration=SECONDS`
An integer that updates the default lifetime, in seconds, for newly created tables in a dataset. The expiration time is set to the current UTC time plus this value. Specify `0` to remove the existing expiration.

`--description=DESCRIPTION`
Updates the description of a dataset, table, table snapshot (Preview), model, or view.

`--destination_reservation_id=RESERVATION_ID`
When used with the `--reservation_assignment` flag, moves an existing reservation assignment to the specified reservation. The value is the ID of the destination reservation. For more information, see Move an assignment to a different reservation.

`--display_name=DISPLAY_NAME`
Updates the display name for a transfer configuration.

`--etag=ETAG`
Acts as a filter; updates the resource only if the resource has an ETag that matches the string specified in the `ETAG` argument.

`--expiration SECONDS`
To update the expiration for the table, model, table snapshot (Preview), or view, include this flag. Replace `SECONDS` with the number of seconds from the update time to the expiration time. To remove the expiration for a table, model, table snapshot (Preview), or view, set the `SECONDS` argument to 0.

`--external_table_definition={TABLE::PATH_TO_FILE|TABLE::DEFINITION}`
Updates an external table with the specified table definition. The table definition can be a path to a local JSON table definition file or an inline table definition in the format `SCHEMA@SOURCE_FORMAT=CLOUD_STORAGE_URI`. The `SCHEMA` value is a comma-separated list of column definitions in the form `FIELD:DATA_TYPE, FIELD:DATA_TYPE`, and so on. If you use a table definition file, then do not give it an extension.

For example:

```
--external_table_definition=myTable::/tmp/tabledef

--
external_table_definition=myTable::Region:STRING,Quarter:STRING,Total_sales:INTEGER
```

`--ignore_idle_slots={true|false}`
Use with the `--reservation` flag. To restrict jobs running in the specified reservation to only use slots allocated to that reservation, set to `true`. The default value is `false`; jobs in the specified reservation can use idle slots from other reservations, or slots that are not allocated to any reservation. For more information, see Idle slots.

`--merge={true|false}`
Use with the `--capacity_commitment` flag. Set to `true` to merge two capacity commitments. The default value is `false`. For more information, see Merge two commitments.

`--model={true|false}` or `-m={true|false}`
To update metadata for a BigQuery ML model, set to `true`. The default value is `false`.

`--params={"PARAMETER":"VALUE"} or -p={"PARAMETER":"VALUE"}`
Updates parameters for a transfer configuration. The parameters vary depending on the data source. For more information, see Introduction to BigQuery Data Transfer Service.

`--plan=PLAN`
When used with the `--capacity_commitment` flag, converts a capacity commitment to a longer-duration commitment plan. One of the following:

- `FLEX`
- `MONTHLY`
- `ANNUAL`

`--refresh_window_days=`*`DAYS`*
An integer that specifies an updated refresh window (in days) for a transfer configuration.

`--renewal_plan=`*`PLAN`*
When used with the `--capacity_commitment` flag, specifies the renewal plan for an existing capacity commitment. One of the following:

- `FLEX`
- `MONTHLY`
- `ANNUAL`

`--reservation={true|false}`
Specifies whether to update a reservation. The default value is `false`.

`--reservation_assignment={true|false}`
Specifies whether to update a reservation assignment. The default value is `false`.

`--schema={`*`SCHEMA_FILE|SCHEMA`*`}`
Specifies either the path to a local JSON schema file or a comma-separated list of column definitions in the form *`FIELD:DATA_TYPE, FIELD:DATA_TYPE`*, and so on. If you use a schema file, then do not give it an extension.

For example:

```
--schema=/tmp/tabledef

--schema=Region:STRING,Quarter:STRING,Total_sales:INTEGER
```

`--set_label=`*`KEY:VALUE`*
Specifies a label to update. To update multiple labels, repeat this flag.

`--slots=`*`NUMBER_OF_SLOTS`*
When used with the `--capacity_commitment` and `--split` flags, specifies the number of slots to split from an existing capacity commitment into a new commitment. When used with the `--reservation` flag, updates the number of slots in a reservation.

`--source=`*`FILE`*
The path to a local JSON file containing a payload used to update a resource. For example, you can use this flag to specify a JSON file that contains a dataset resource with an updated `access` property. The file is used to overwrite the dataset's access controls.

`--split={true|false}`
When used with the `--capacity_commitment` flag, specifies whether to split an existing capacity commitment. The default value is `false`. For more information, see Split a commitment.

`--table={true|false}` or **`-t={true|false}`**
Specifies whether to update a table. The default value is `false`.

`--target_dataset=DATASET`
When specified, updates the target dataset for a transfer configuration.

`--time_partitioning_expiration=SECONDS`
An integer that updates (in seconds) when a time-based partition should be deleted. The expiration time evaluates to the partition's UTC date plus the integer value. A negative number indicates no expiration.

`--time_partitioning_field=COLUMN_NAME`
Updates the field used to determine how to create a time-based partition. If time-based partitioning is enabled without this value, then the table is partitioned based on the load time.

`--time_partitioning_type=INTERVAL`
Specifies the partitioning type. Use one of the following values:

- `DAY`
- `HOUR`
- `MONTH`
- `YEAR`

You can't change the partitioning type of an existing table.

`--transfer_config={true|false}`
Specifies whether to update a transfer configuration. The default value is `false`.

`--update_credentials={true|false}`
Specifies whether to update the transfer configuration credentials. The default value is `false`.

`--use_legacy_sql={true|false}`
Set to `false` to update the SQL query for a view from legacy SQL to Standard SQL. The default value is `true`; the query uses legacy SQL.

`--view=QUERY`
When specified, updates the SQL query for a view.

`--view_udf_resource=FILE`
Updates the Cloud Storage URI or the path to a local code file that is loaded and evaluated immediately as a user-defined function resource in a view's SQL query. Repeat this flag to specify multiple files.

`RESOURCE`
The resource that you want to update.

For more information about using the `bq update` command, see the following:

`bq version`

Use the `bq version` command to display the version number of your `bq` command-line tool.

Synopsis

```
bq version
```

Note: You can see the version number of all components in your Cloud SDK installation by using the `gcloud version` command.

`bq wait`

Use the `bq wait` command to wait a specified number of seconds for a job to finish. If a job isn't specified, then the command waits for the current job to finish.

Synopsis

```
bq wait [FLAGS] [JOB] [SECONDS]
```

Examples

```
bq wait

bq wait --wait_for_status=RUNNING 12345 100
```

Flags and arguments

The `bq wait` command uses the following flags and arguments:

`--fail_on_error={true|false}`
To return success if the job completed during the wait time, even if the job failed, set to `false`. The default value is `true`; after the wait time elapses, the command exits with an error if the job is still running, or if the job completed but failed.

`--wait_for_status=STATUS`
When specified, waits for a particular job status before exiting. Use one of the following values:

- `PENDING`
- `RUNNING`
- `DONE`

The default value is `DONE`.

`JOB`
Specifies the job to wait for. You can use the `bq ls --jobs myProject` command to find a job identifier.

`SECONDS`

Specifies the maximum number of seconds to wait until the job is finished. If you enter `0`, then the command polls for job completion and returns immediately. If you do not specify an integer value, then the command waits until the job is finished.

Was this helpful?