In addition to running gcloud CLI commands from the command line, you can also run them from scripts or other automations — for example, when using Jenkins to drive automation of Google Cloud Platform tasks.

Google Cloud SDK tools support two authorization (/sdk/gcloud/guide/authorizing) methods:

- User account authorization
- Service account authorization

User account authorization is recommended if you are running a script or other automation on a single machine.

To authorize access and perform other common Cloud SDK setup steps:

Service account authorization is recommended if you are deploying a script or other automation across machines in a production environment. It is also the recommended authorization method if you are running gcloud CLI commands on a Google Compute Engine virtual machine instance where all users have access to `root`.

To use service account authorization, use an existing service account or create a new one through the Google Cloud Platform Console (https://console.cloud.google.com/iam-admin/serviceaccounts). From the options column of the service accounts table, create and download the associated private key as a JSON-formatted key file.

To run the authorization, use `gcloud auth activate-service-account` (/sdk/gcloud/reference/auth/activate-service-account):

You can SSH into your VM instance by using `gcloud compute ssh` (/sdk/gcloud/reference/compute/ssh), which takes care of authentication. SSH configuration files can be configured using `gcloud compute` `config-ssh` (/sdk/gcloud/reference/compute/config-ssh/).

For detailed instructions regarding authorizing Cloud SDK tools, refer to this comprehensive guide (/sdk/docs/authorizing).

Some gcloud CLI commands are interactive, prompting users for confirmation of an operation or requesting additional input for an entered command.

In most cases, this is not desirable when running commands in a script or other automation. You can disable prompts from gcloud CLI commands by setting the `disable_prompts` (/sdk/gcloud/guide/properties) property in your configuration (/sdk/gcloud/guide/configurations) to `True` or by using the global `--quiet` (/sdk/gcloud/reference) or `-q` flag. Most interactive commands have default values when additional confirmation or input is required. If prompts are disabled, these default values are used.

For example:

Note the `--quiet` flag is inserted right at the front.

If you want a script or other automation to perform actions conditionally based on the output of a gcloud CLI command, observe the following:

- **Don't depend on messages printed to standard error.**

  These may change in future versions of the gcloud CLI and break your automation.

- **Don't depend on the raw output of messages printed to standard output.**

  The default output for any command may change in a future release. You can minimize the impact of those changes by using the `--format` flag (/sdk/gcloud/reference/topic/formats) to format the output with one of the following: –format=json|yaml|csv|text|list to specify values to be returned. Run $ gcloud topic formats for more options.

You can modify the default output from `--format` (/sdk/gcloud/reference/topic/formats) by using `projections` (/sdk/gcloud/reference/topic/projections). For increased granularity, use the `--filter` flag (/sdk/gcloud/reference/topic/filters) to return a subset of the values based on an expression. You can then script against those returned values.

Examples of formatting and filtering output can be found in the section below.

- **Do depend on command exit status.**

  If the exit status is not zero, an error occurred and the output may be incomplete unless the command documentation notes otherwise. For example, a command that creates multiple resources may only create a few, list them on the standard output, and then exit with a non-zero status. Alternatively, you can use the `show_structured_logs` property to parse error logs. Run $ gcloud config for more details.

To work through an interactive tutorial about using the filter and format flags instead, launch the tutorial using the following button:

OPEN IN GOOGLE CLOUD SHELL

(https://console.cloud.google.com/cloudshell/open?git_repo=https://github.com/GoogleCloudPlatform/cloud-shell-tutorials&page=editor&tutorial=cloudsdk/tutorial.md)

The following are examples of common uses of formatting and filtering with gcloud CLI commands:

List instances created in zone *us-central1-a*:

List in JSON format those projects where the labels match specific values (e.g. label.env is 'test' and label.version is alpha):

List projects with their creation date and time specified in the local timezone:

List projects that were created after a specific date in table format:

Note that in the last example, a projection on the key was used. The filter is applied on the createTime key after the date formatting is set.

List a nested table of the quotas of a region:

Print a flattened list of global quotas in CSV format:

List compute instance resources with box decorations and titles, sorted by name, in table format:

List the project authenticated user email address:

Using this functionality of format and filter, you can combine gcloud CLI commands into a script to easily extract embedded information.

If you were to list all the keys associated with all your projects' service accounts, you'd need to iterate over all your projects and for each project, get all the service accounts associated with it. For each service account, get all the keys. This can be accomplished as demonstrated below:

As a bash script:

Or as Windows PowerShell:

Oftentimes, you'll need to parse output for processing. For example, it'd be useful to write the service account information into an array and segregate values in the multi-valued CSV-formatted `serviceAccounts.scope()` field. The script below does just this:

For a step-by-step guide to building basic scripts with the gcloud CLI, refer to this beginner's guide to automating GCP tasks (https://cloud.google.com/blog/products/management-tools/scripting-with-gcloud-a-beginners-guide-to-automating-gcp-tasks)
.

More involved examples of the output configuring capabilities built into gcloud CLI's `filters` (/sdk/gcloud/reference/topic/filters), `formats` (/sdk/gcloud/reference/topic/formats), and `projections` (/sdk/gcloud/reference/topic/projections) flags can be found in this blog post about filtering and formatting (https://cloudplatform.googleblog.com/2016/06/filtering-and-formatting-fun-with.html).