How to SSH into a Google Cloud VM

Credits:https://medium.com/@the.nick.miller/how-to-ssh-into-a-google-cloudvm-1a721fc77259

If you're coming to GCP from the land of AWS, one of the first things you'll discover while learning the platform is that there is no straightforward way to SSH into a machine from a local machine.

GCP encourages us to utilize their Cloud Shell, a web-based CLI and managed development environment. While it's a solid choice for basic terminal tasks, some of us may still prefer the familiarity and flexibility of our local environment when working on a project.

So, how are we expected SSH into a GCE instance remotely? Let's walk through it.

Prerequisites

- macOS/Linux Operating System — I don't use Windows. This guide will still provide a helpful method, but you'll likely have to tweak and troubleshoot some of the steps for Windows.

Install and authenticate with gcloud

Step 1: Installation

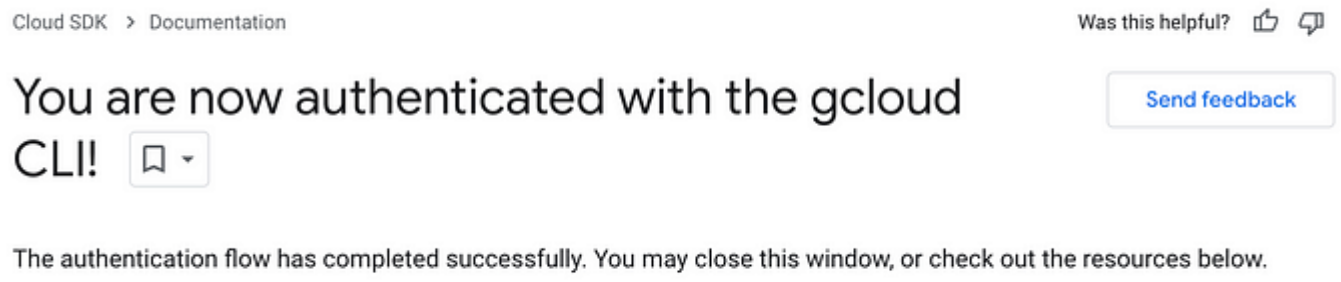
First, we will install Google's CLI tool that lets you interact with resources directly from your terminal. It's called gcloud.

Installing gcloud on your computer is easy when you use one of GCP's installers for MacOS or follow the package installation instructions for Linux. Google walks you through the installation here: https://cloud.google.com/sdk/docs/install

Step 2: Authentication

Next, run the `gcloud auth login` command. This will open up a Google login screen where you need to login to the account you're using for GCP. This will generate user credentials `gcloud` uses for authentication and authorization to GCP.

We will see this in our browser when we're finished authenticating in the browser:



First Method: gcloud SSH

If we need simple access to your instance via terminal, we can use `glcoud` to establish a connection:

```
gcloud compute ssh $INSTANCE_NAME --zone $ZONE
```

To access the instance, either define $INSTANCE_NAME and $ZONE variables in your terminal or replace them with the instance name and zone of your choice.

If we authenticate with a user who has permission to access your instance, `gcloud` automatically generates a new SSH key pair and adds the public key to the project's metadata. After `gcloud` generates the key, it will be stored on your local machine in
the `~/.ssh/google_compute_engine` (private key)
and `~/.ssh/google_compute_engine.pub` (public key) files.

Second Method: Use a gcloud script to configure SSH

However, a simple SSH is not the reason I wrote this article. For many use cases, you need to configure SSH connections on your machine so that they can be used by local tools. In my case, I was VS Code's **Remote — SSH** extension to develop on cloud instances as if I were on my local machine. VS Code has a large ecosystem of extensions and a clean editor interface (sorry Vim and Emacs aficionados) that I can use once that connection is established.

For **Remote — SSH** to work, I need to create an SSH configuration within `~/.ssh/configs` on my local machine that will be used to SSH into the remote instance. Configs will look something like this:

```
Host $PUBLIC_IP
  HostName $PUBLIC_IP
  IdentityFile $PRIVATE_KEY_PATH
  User $USERNAME
```

This bash script automates the collection of the information needed to create a config and adds the config to `~/.ssh/configs`:

```
#!/bin/bash

# Get instance name as an argument or use demo-instance as default
INSTANCE_NAME=${1:-demo-instance}
```

```
# Get zone as an argument or use us-west4-b as default
ZONE=${2:-us-west4-b}

# Location of Private Key created during authentication
PRIVATE_KEY_PATH=~/.ssh/google_compute_engine

# Check if the instance is running
INSTANCE_STATUS=$(gcloud compute instances describe $INSTANCE_NAME --zone=$ZONE
--format='get(status)')

# If the instance is not running, tell the user to start the instance first and
exit the script
if [ "$INSTANCE_STATUS" != "RUNNING" ]; then
  echo "Instance $INSTANCE_NAME is not running. Please start the instance
first."
  exit 1
fi

# Get the public IP address of the instance
PUBLIC_IP=$(gcloud compute instances describe $INSTANCE_NAME --zone=$ZONE --
format='get(networkInterfaces[0].accessConfigs[0].natIP)')

# Get the username of the instance
USERNAME=$(gcloud compute ssh $INSTANCE_NAME --zone $ZONE --dry-run | awk -F'[@
]' '{print $(NF-1)}')

# Ensure .ssh key is added to project metadata by logging in to the instance and
running a command
gcloud compute ssh $INSTANCE_NAME --zone $ZONE --command "echo 'Hello from
$INSTANCE_NAME.'"

echo "Public IP: $PUBLIC_IP"
echo "Username: $USERNAME"
echo "Identity file: $PRIVATE_KEY_PATH"

# Append config to .ssh/config
cat << EOF >> ~/.ssh/config

Host $PUBLIC_IP
    HostName $PUBLIC_IP
    IdentityFile $PRIVATE_KEY_PATH
    User $USERNAME
EOF

# ALTERNATIVE TO CONFIG CREATION: SSH into the instance
# ssh -i $PRIVATE_KEY_PATH $USERNAME@$PUBLIC_IP
```

To briefly describe what this script does:

- It takes the instance name and zone as command-line arguments, or uses default values if not provided.
- It retrieves the Public IP Address and Username of the instance using gcloud commands.

- It ensures that the SSH key is added to the project metadata by logging into the instance using `gcloud` and running a dummy command.
- It appends the SSH configuration for the instance to the user's `~/.ssh/config` file, allowing for easier future connections.

This script will make your life much easier if you frequently need to access GCE instances from your local machine.