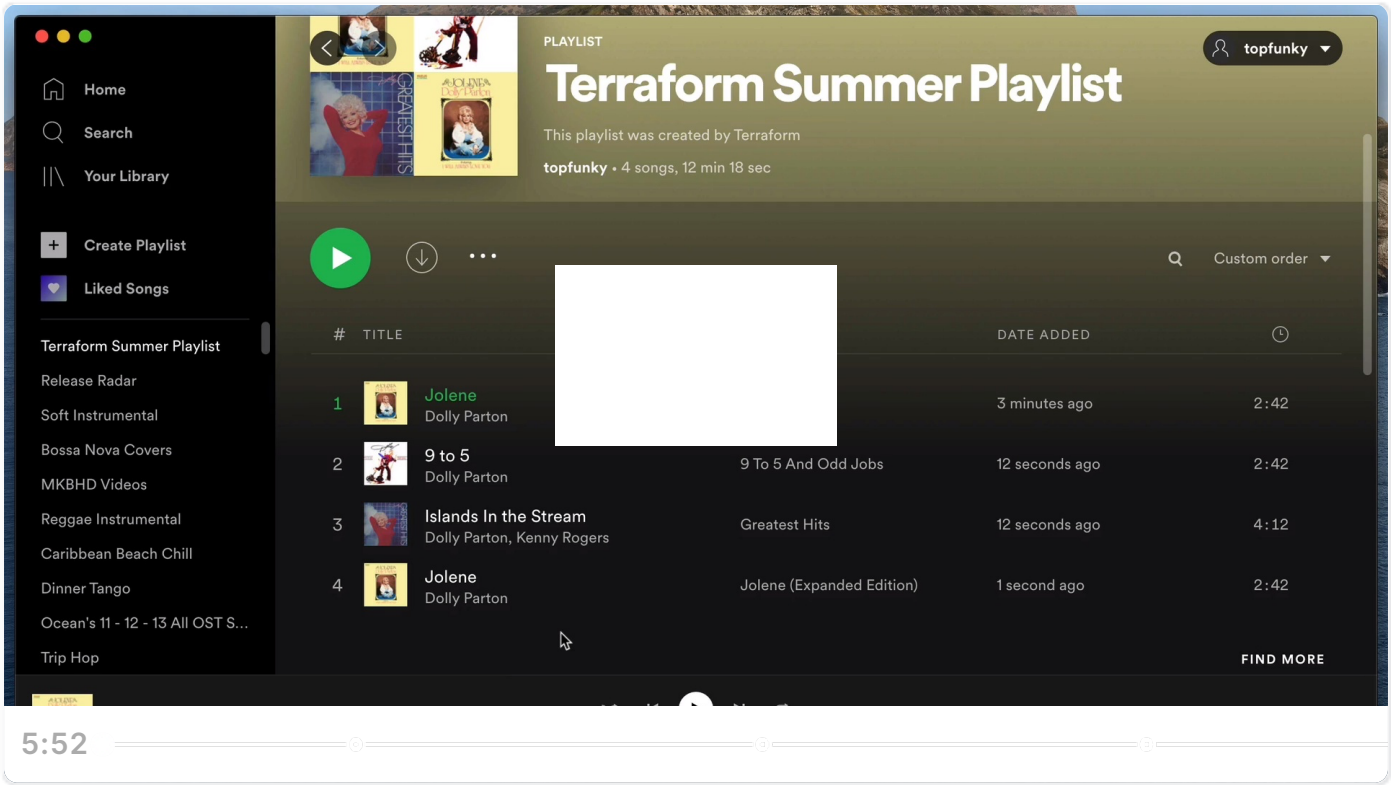
Create a Spotify playlist with Terraform

15min |

Reference this often? Create an account to bookmark tutorials.



Terraform manages infrastructure on cloud computing providers such as AWS, Azure, and GCP. But, it can also manage resources in hundreds of other services,

including the music service Spotify.

In this tutorial, you will use a Terraform data source to search Spotify for an artist, album, or song, and use that data to build a playlist.

Prerequisites

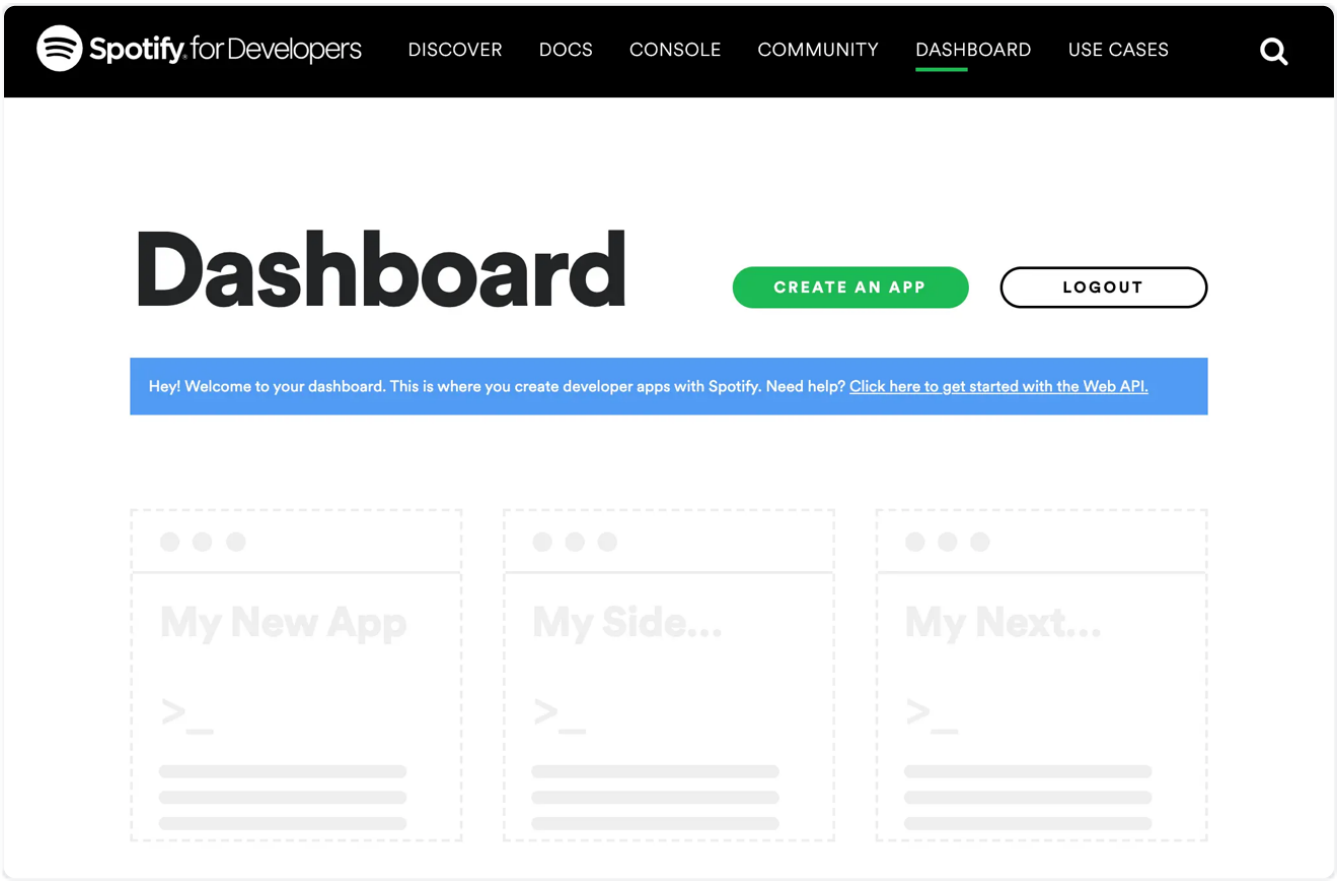
To complete this tutorial, you will need:

- Terraform **version 1.0+**
- Docker Desktop
- Spotify account with developer access

Create Spotify developer app

Before you can use Terraform with Spotify, you need to create a Spotify developer app and run Spotify's authorization proxy server.

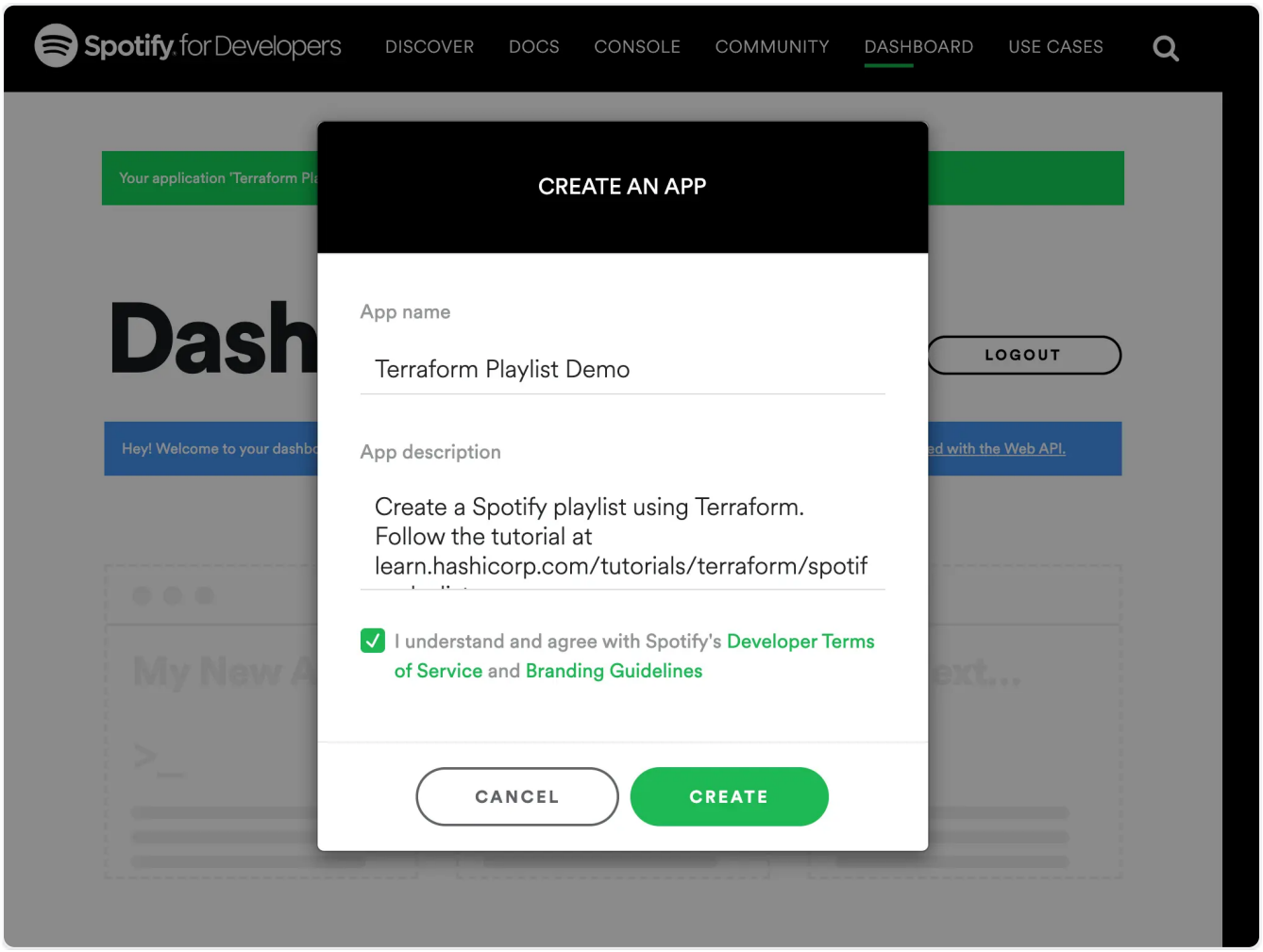
Login to the Spotify developer dashboard.



Click the green **Create an app** button.

Fill out the name and description according to the table below, check the box to agree to the terms of services, then click **Create**.

Name	Description
Terraform Playlist Demo	Create a Spotify playlist using Terraform. Follow the tutorial at learn.hashicorp.com/tutorials/terraform/spotify-playlist



Once Spotify creates the application, find and click the green **Edit Settings** button on the top right side.

Copy the URI below into the **Redirect URI** field and click **Add** so that Spotify can find its authorization application locally on port `27228` at the correct path. Scroll to the bottom of the form and click **Save**.

```
http://localhost:27228/spotify_callback
```

Run authorization server

Now that you created the Spotify app, you are ready to configure and start the authorization proxy server, which allows Terraform to interact with Spotify.

Return to your terminal and set the redirect URI as an environment variable, instructing the authorization proxy server to serve your Spotify access tokens on port `27228`.

```
$ export SPOTIFY_CLIENT_REDIRECT_URI=http://localhost:27228/spotify_ca
```

Next, create a file called `.env` with the following contents to store your Spotify application's client ID and secret.

```
SPOTIFY_CLIENT_ID=
SPOTIFY_CLIENT_SECRET=
```

Copy the **Client ID** from the Spotify app page underneath your app's title and description, and paste it into `.env` as your `SPOTIFY_CLIENT_ID`.

Click **Show client secret** and copy the value displayed into `.env` as your `SPOTIFY_CLIENT_SECRET`.

Make sure Docker Desktop is running, and start the server. It will run in your terminal's foreground.

```
$ docker run --rm -it -p 27228:27228 --env-file ./.env ghcr.io/conradl
Unable to find image 'ghcr.io/conradludgate/spotify-auth-proxy:latest' loc
latest: Pulling from conradludgate/spotify-auth-proxy
5843afab3874: Pull complete
b244520335f6: Pull complete
Digest: sha256:c738f59a734ac17812aae5032cfc6f799e03c1f09d9146edb9c2836bc58
Status: Downloaded newer image for ghcr.io/conradludgate/spotify-auth-proxy
APIKey: xxxxxx...
Token:  xxxxxx...
Auth:   http://localhost:27228/authorize?token=xxxxxx...
```

Visit the authorization server's URL by visiting the link that your terminal output lists after `Auth:`.

The server will redirect you to Spotify to authenticate. After authenticating, the server will display `Authorization successful`, indicating that the Terraform provider can use the server to retrieve access tokens.

Leave the server running.

Clone example repository

Clone the [example Terraform configuration] for this tutorial. It contains a complete Terraform configuration that searches for songs by Dolly Parton, and creates a playlist out of them.

```
$ git clone https://github.com/hashicorp-education/learn-terraform-spo
```

Change into the directory.

```
$ cd learn-terraform-spotify
```

Explore the configuration

Open `main.tf`. This file contains the Terraform configuration that searches Spotify and creates the playlist. The first two configuration blocks in the file:

- configure Terraform itself and specify the community provider that Terraform uses to communicate with Spotify.
- configure the Spotify provider with the key you set as a variable.

```
terraform {
  required_providers {
    spotify = {
      version = "~> 0.1.5"
      source  = "conradludgate/spotify"
    }
  }
}

provider "spotify" {
  api_key = var.spotify_api_key
}
```

The next block defines a Terraform data source to search the Spotify provider for Dolly Parton songs.

```
data "spotify_search_track" "by_artist" {
  artists = ["Dolly Parton"]
  #  album = "Jolene"
  #  name  = "Early Morning Breeze"
}
```

The next block uses a Terraform resource to create a playlist from the first three songs that match the search in the data source block.

```
resource "spotify_playlist" "playlist" {
  name        = "Terraform Summer Playlist"
  description = "This playlist was created by Terraform"
  public      = true

  tracks = [
    data.spotify_search_track.by_artist.tracks[0].id,
    data.spotify_search_track.by_artist.tracks[1].id,
    data.spotify_search_track.by_artist.tracks[2].id,
  ]
}
```

Open `outputs.tf`, which defines an output value for the URL of the playlist.

```
output "playlist_url" {
  value       = "https://open.spotify.com/playlist/${spotify_playlist.playl
  description = "Visit this URL in your browser to listen to the playlist"
}
```

Set the API key

Rename the `terraform.tfvars.example` file `terraform.tfvars` so that Terraform can detect the file.

```
$ mv terraform.tfvars.example terraform.tfvars
```

The `.gitignore` file in this repository excludes files with the `.tfvars` extension from version control to prevent you from accidentally committing your credentials.

> **Warning**
> Never commit sensitive values to version control.

Find the terminal window where the Spotify authorization proxy server is running and copy the `APIKey` from its output.

Open `terraform.tfvars`, and replace `...` with the key from the proxy, so that Terraform can authenticate with Spotify. Save the file.

```
spotify_api_key = "..."
```

This variable is declared for you in `variables.tf`.

```
variable "spotify_api_key" {
  type        = string
  description = "Set this as the APIKey that the authorization proxy serve
}
```

Install the Spotify provider

In your terminal, initialize Terraform, which will install the Spotify provider.

```
$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding conradludgate/spotify versions matching "~> 0.1.5"...
- Installing conradludgate/spotify v0.1.5...
- Installed conradludgate/spotify v0.1.5 (self-signed, key ID B4E4E68AFAC5

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it h
https://www.terraform.io/docs/cli/plugins/signing.html

Terraform has created a lock file .terraform.lock.hcl to record the provid
selections it made above. Include this file in your version control reposi
so that Terraform can guarantee to make the same selections by default whe
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to
any changes that are required for your infrastructure. All Terraform comma
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget,
commands will detect it and remind you to do so if necessary.
```

Create the playlist

Now you are ready to create your playlist. Apply your Terraform configuration.

Terraform will show you the changes it plans to make and prompt for your approval.

```
$ terraform apply

Terraform used the selected providers to generate the following execution p
indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # spotify_playlist.playlist will be created
  + resource "spotify_playlist" "playlist" {
      + description = "This playlist was created by Terraform"
      + id          = (known after apply)
      + name        = "Terraform Summer Playlist"
      + public      = true
      + snapshot_id = (known after apply)
      + tracks      = [
          + "2SpEHTbUuebeLkgs9QB7Ue",
          + "4w3tQBXhn5345eUXDGBWZG",
          + "6dnco8haegnJYtylV26cBq",
        ]
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + playlist_url = (known after apply)

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value:
```

Confirm the apply with a `yes`, and Terraform will create your playlist.

```
  Enter a value: yes

spotify_playlist.playlist: Creating...
spotify_playlist.playlist: Creation complete after 1s [id=40bGNifvqzwjO8gHl

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

playlist_url = "https://open.spotify.com/playlist/40bGNifvqzwjO8gHDvhbB3"
```

Listen to your playlist

Open the playlist URL returned in the Terraform output and enjoy your playlist!

Customize and share!