your Free Tier credits, the examples in this book should still cost you no more than a few dollars.

If you've never used AWS or Terraform before, don't worry; this tutorial is designed for novices to both technologies. I'll walk you through the following steps:

- Setting up your AWS account
- Installing Terraform
- Deploying a single server
- Deploying a single web server
- Deploying a configurable web server
- Deploying a cluster of web servers
- Deploying a load balancer
- Cleaning up

> **EXAMPLE CODE**
>
> As a reminder, you can find all of the code examples in the book on GitHub.

Setting Up Your AWS Account

If you don't already have an AWS account, head over to *https://aws.amazon.com* and sign up. When you first register for AWS, you initially sign in as the *root user*. This user account has access permissions to do absolutely anything in the account, so from a security perspective, it's not a good idea to use the root user on a day-to-day basis. In fact, the *only* thing you should use the root user for is to create other user accounts with more-limited permissions, and then switch to one of those accounts immediately.[4]

To create a more-limited user account, you will need to use the *Identity and Access Management* (IAM) service. IAM is where you manage user accounts as well as the permissions for each user. To create a new *IAM user*, go to the IAM Console, click Users, and then click the Add Users button. Enter a name for the user, and make sure "Access key - Programmatic access" is selected, as shown in Figure 2-1 (note that AWS occasionally makes changes to its web console, so what you see may look slightly different than the screenshots in this book).

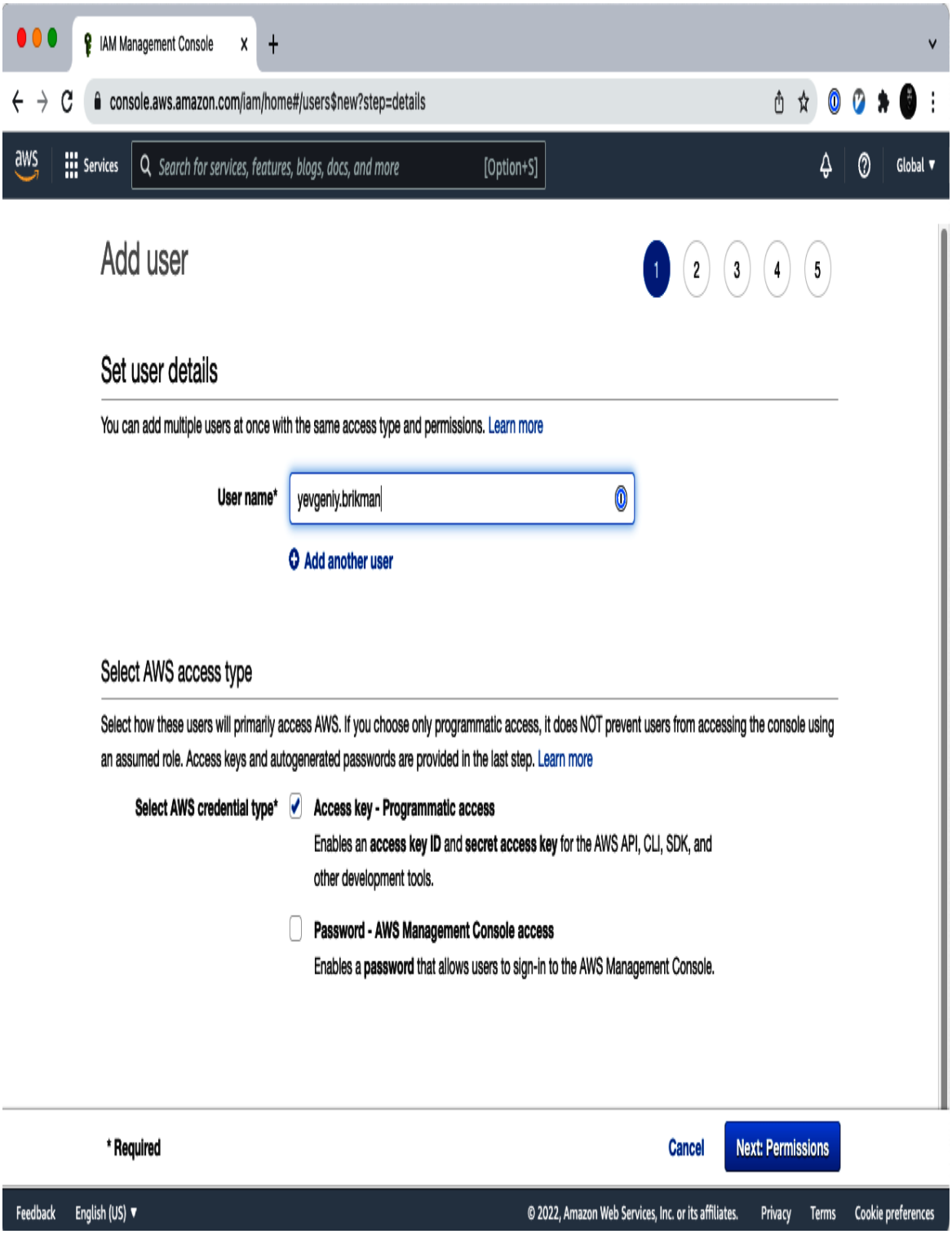

Figure 2-1. Use the AWS Console to create a new IAM user.

Click the Next button. AWS will ask you to add permissions to the user. By default, new IAM users have no permissions whatsoever and cannot do

anything in an AWS account. To give your IAM user the ability to do something, you need to associate one or more IAM Policies with that user's account. An *IAM Policy* is a JSON document that defines what a user is or isn't allowed to do. You can create your own IAM Policies or use some of the predefined IAM Policies built into your AWS account, which are known as *Managed Policies*.[5]

To run the examples in this book, the easiest way to get started is to add the `AdministratorAccess` Managed Policy to your IAM user (search for it, and click the checkbox next to it), as shown in Figure 2-2.[6]

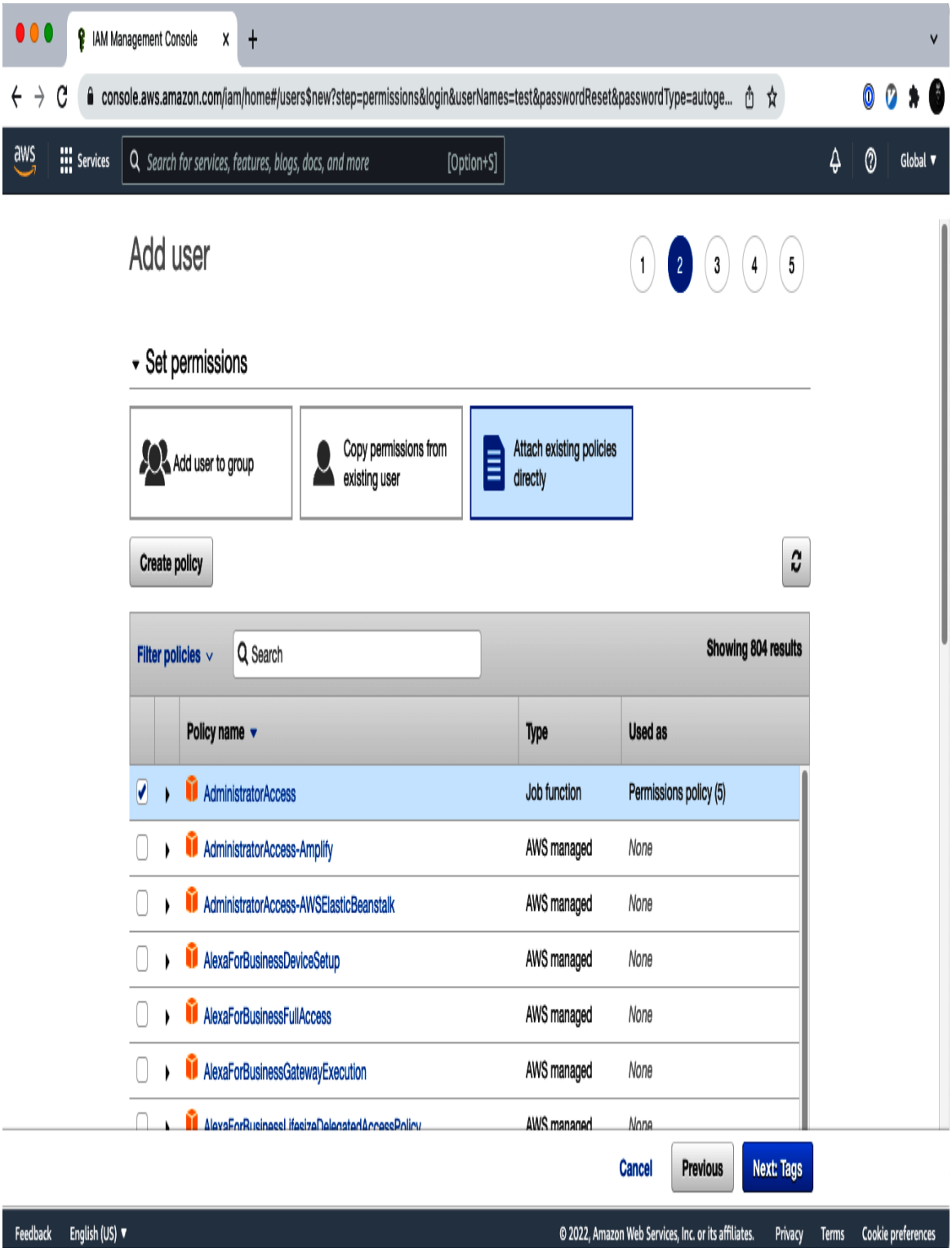

Figure 2-2. Add the `AdministratorAccess` Managed IAM Policy to your new IAM user.

Click Next a couple more times and then the "Create user" button. AWS will show you the security credentials for that user, which consist of an

Access Key ID and a *Secret Access Key*, as shown in Figure 2-3. You must save these immediately because they will never be shown again, and you'll need them later on in this tutorial. Remember that these credentials give access to your AWS account, so store them somewhere secure (e.g., a password manager such as 1Password, LastPass, or macOS Keychain), and never share them with anyone.

After you've saved your credentials, click the Close button. You're now ready to move on to using Terraform.

A NOTE ON DEFAULT VIRTUAL PRIVATE CLOUDS

All of the AWS examples in this book use the *Default VPC* in your AWS account. A *VPC*, or virtual private cloud, is an isolated area of your AWS account that has its own virtual network and IP address space. Just about every AWS resource deploys into a VPC. If you don't explicitly specify a VPC, the resource will be deployed into the Default VPC, which is part of every AWS account created after 2013. If for some reason you deleted the Default VPC in your account, either use a different region (each region has its own Default VPC) or create a new Default VPC using the AWS Web Console. Otherwise, you'll need to update almost every example to include a `vpc_id` or `subnet_id` parameter pointing to a custom VPC.