Chapter 32: Deploying with Heroku

By Logan Wright

Heroku is a popular hosting solution that simplifies deployment of web and cloud applications. It supports a number of popular languages and database options. In this chapter, you'll learn how to deploy a Vapor web app with a Postgres database on Heroku.

Setting up Heroku

If you don't already have an Heroku account, sign up for one now. Heroku offers free options and setting up an account is painless. Simply visit https://signup.heroku.com/ and follow the instructions to create an account.

Installing CLI

Now that you have your Heroku account, install the Heroku CLI tool. The easiest way to install on macOS is through Homebrew. In Terminal, enter:

```
brew install heroku/brew/heroku
```

If you don't wish to use Homebrew, or are running on Linux, there are other installation options available at https://devcenter.heroku.com/articles/heroku-cli#download-and-install.

Logging in

With the Heroku CLI installed, you need to log in to your account. In Terminal, enter:

```
heroku login
```

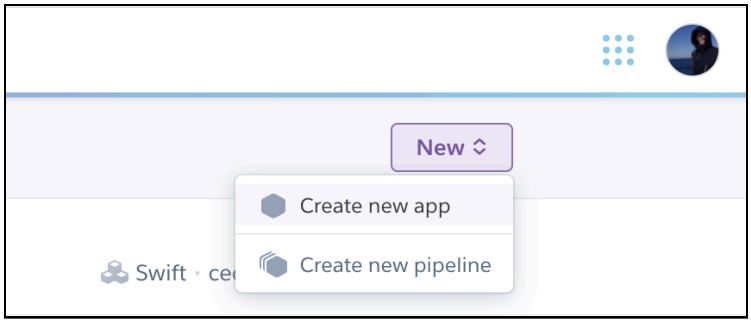
Follow the prompts, entering your email and password. Once you've logged in, you can verify success by checking `whoami` to ensure it outputs the correct email. Use the following command:

```
heroku auth:whoami
```

That's it; Heroku is all setup on your system. Now it's time to create your first project.

Create an application

Visit heroku.com in your browser to create a new application. Heroku.com should redirect you to dashboard.heroku.com. If it doesn't, make sure you're logged in and try again. Once at the dashboard, in the upper right hand corner, there's a button that says **New**. Click it and select **Create new app**.



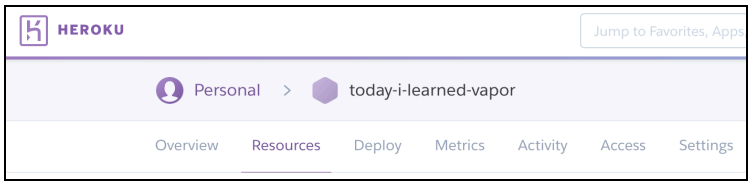
Enter application name

At the next screen, choose the deployment region and a unique app name. If you don't want to choose your app's name, leave the field blank and Heroku automatically generates a unique slug to identify the application for you. Whether you create a name, or Heroku assigns you one, make note of it; you'll use it later when configuring your app.

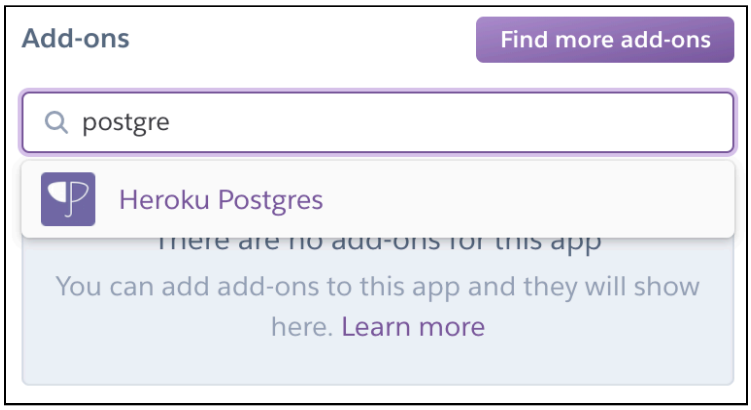
Click **Create app**.

Add PostgreSQL database

After creating your application, Heroku redirects you to your application's page. Near the top, under your application's name, there is a row of tabs. Select **Resources**.

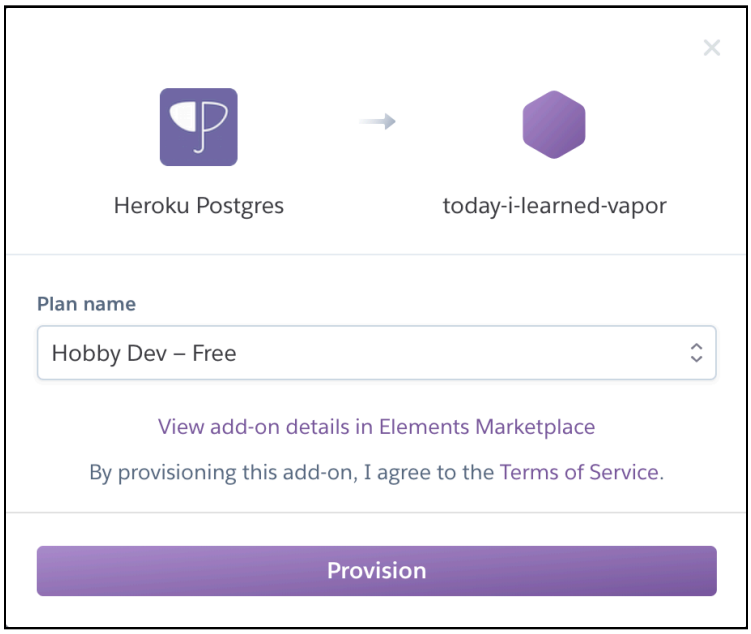


Under the section titled **Add-ons**, enter **postgres** and you'll see an option for **Heroku Postgres**. Select this option.

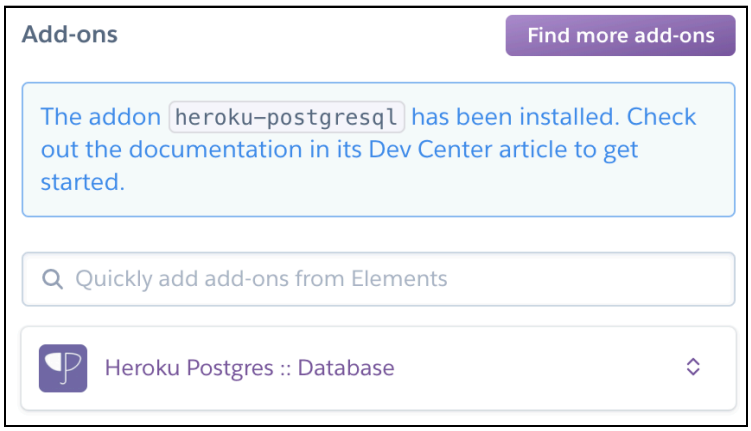


This takes you to one more screen which asks what type of database to provision. For now, provision a **Hobby Dev - Free** version to use.

Click the **Provision** button and Heroku does the rest.



Once you finish, you'll see the database appears under the **Resources** tab.



Setting up your Vapor app locally

Your application is now setup with Heroku; the next step is to configure the Vapor app locally. Download and open the project associated with this chapter. If you've been following along with the book, it should look like the TIL project you've been working on. You're free to use your own project instead.

Git

Heroku uses Git to deploy your app, so you'll need to put your project into a Git repository, if it isn't already.

First, determine whether your application already has a Git repository. To do this, enter the following command in Terminal:

```
git rev-parse --is-inside-work-tree
```

It should output **true**. If it doesn't, then you must initialize a Git repository. Otherwise, skip the next section.

Initialize Git

If you need to add Git to your project, enter the following command in Terminal:

```
git init
```

Branch

Heroku deploys the **master** branch. Make sure you are on this branch and have merged any changes you wish to deploy.

To see your current branch, enter the following in Terminal:

```
git branch
```

The output will look something like the following. The branch with the asterisk next to it is the current branch:

```
* master
  commander
  other-branches
```

> **Note**: If you don't see any output and you've just performed `git init`. You'll need to commit your code first then you'll see output from the `git branch` command.

If you're not currently on **master**, switch there by entering:

```
git checkout master
```

Commit changes

Make sure all changes are in your master branch and committed. You can verify by entering the following command. If you see any output, it means you have uncommitted changes.

```
git status --porcelain
```

If you have uncommitted changes, enter the following commands to commit them:

```
git add .
git commit -m "a description of the changes I made"
```

This ensures your project is in your local repository.

Connect with Heroku

Heroku needs to configure another remote on your Git repository. Enter the following command in Terminal, substituting your app's Heroku name:

```
$ heroku git:remote -a your-apps-name-here
```

You can confirm the format of this command by clicking the **Deploy** tab on the Heroku dashboard in your browser and looking at the command under **Existing Git repository**.

Set Stack

As of 13 September 2018, Heroku's default stack is Heroku 18, which will cause problems in building Swift and Vapor. This means you'll need to ensure your app is built using the Heroku 16 infrastructure. To do this, enter the following command:

```
heroku stack:set heroku-16 -a your-apps-name-here
```

Set Buildpack

Heroku uses something called a Buildpack to provide the recipe for building your app when you deploy it. The Vapor Community currently provides a Buildpack designed for Vapor apps. To set the Buildpack for your application, enter the following in Terminal:

```
heroku buildpacks:set https://github.com/vapor-community/heroku-buildpack
```

Swift version file

Now that your Buildpack is set, Heroku needs a couple of configuration files. The first of these is **.swift-version**. This is used by the Buildpack to determine which version of Swift to install for the project. Enter the following command in Terminal:

```
echo "4.2.2" > .swift-version
```

This creates **.swift-version** with `4.2.2` as its contents.

Procfile

Once the app is built on Heroku, Heroku needs to know what type of process to run and how to run it. To determine this, it utilizes a special file named **Procfile**. Enter the following command to create your Procfile:

```
echo "web: Run serve --env production" \
  "--hostname 0.0.0.0 --port \$PORT" > Procfile
```

This gives Heroku the command it needs to run your app.

Commit changes

As mentioned earlier, Heroku uses Git and the master branch to deploy applications. Since you configured Git earlier, you've added two files: **Procfile** and **.swift-version**.

You need to commit these before deploying or Heroku won't be able to properly build the application. Enter the following commands in Terminal:

```
git add .
git commit -m "adding heroku build files"
```

Configure the database

There's one more thing to do before you deploy your app: You must configure the database within your app. Start by listing the configuration variables for your app.

In Terminal, enter:

```
heroku config
```

You should see output similar to the following. It provides you with information about the database you provisioned for this project.

```
=== today-i-learned-vapor Config Vars
DATABASE_URL: postgres://cybntsgadydqzm:
2d9dc7f6d964f4750da1518ad71hag2ba729cd4527d4a18c70e024b11cfa8f4b@ec2-54-2
21-192-231.compute-1.amazonaws.com:5432/dfr89mvoo550b4
```

There are two parts to this output; the first is **DATABASE_URL**. This represents the name of the environment variable. The second component will be similar to the following:

```
postgres://cybntsgadydqzm:
2d9dc7f6d964f4750da1518ad71hag2ba729cd4527d4a18c70e024b11cfa8f4b@ec2-54-2
21-192-231.compute-1.amazonaws.com:5432/dfr89mvoo550b4
```

This component represents the actual value of the environment variable. In this case, it's the direct link to your Postgres database. You can use this direct url for purposes of manually connecting to the database should you need to for some reason. However, it's important that you **NEVER** hard code this value into your application. Not only is it bad practice and unsafe, Heroku specifies that the value of this environment variable could change at any time, rendering the absolute value useless.

The important part is the environment variable's name: **DATABASE_URL**.

Open your Vapor app in Xcode and navigate to **configure.swift**. Replace:

```
let hostname =
  Environment.get("DATABASE_HOSTNAME") ?? "localhost"
let databaseName: String
let databasePort: Int
if env == .testing {
  databaseName = "vapor-test"
```

```
    if let testPort = Environment.get("DATABASE_PORT") {
      databasePort = Int(testPort) ?? 5433
    } else {
      databasePort = 5433
    }
  } else {
    databaseName = "vapor"
    databasePort = 5432
  }

  let databaseConfig = PostgreSQLDatabaseConfig(
    hostname: hostname,
    port: databasePort,
    username: "vapor",
    database: databaseName,
    password: "password")
```

with the following:

```
  let databaseConfig: PostgreSQLDatabaseConfig
  if let url = Environment.get("DATABASE_URL") {
    databaseConfig = PostgreSQLDatabaseConfig(url: url)!
  } else {
    let hostname =
      Environment.get("DATABASE_HOSTNAME") ?? "localhost"
    let databaseName: String
    let databasePort: Int
    if env == .testing {
      databaseName = "vapor-test"
      if let testPort = Environment.get("DATABASE_PORT") {
        databasePort = Int(testPort) ?? 5433
      } else {
        databasePort = 5433
      }
    } else {
      databaseName = "vapor"
      databasePort = 5432
    }

    databaseConfig = PostgreSQLDatabaseConfig(
      hostname: hostname,
      port: databasePort,
      username: "vapor",
      database: databaseName,
      password: "password")
  }
```

This allows the project to retrieve the database URL from the environment if it's running on Heroku. If **DATABASE_URL** isn't set in the environment, the app continues to use the previous method for determining its database.

Once again, you need to save your changes in Git. Enter the following in Terminal:

```
git add .
git commit -m "configured heroku database"
```

Configure Google environment variables

If you completed Chapter 22, "Google Authentication", and are using that as your project here, you must configure the same Google environment variables you used there.

Enter the following commands in Terminal:

```
heroku config:set \
  GOOGLE_CALLBACK_URL=https://<YOUR_HEROKU_URL>/oauth/google

heroku config:set GOOGLE_CLIENT_ID=<YOUR_CLIENT_ID>

heroku config:set GOOGLE_CLIENT_SECRET=<YOUR_CLIENT_SECRET>
```

You can find your Heroku URL on the **Settings** tab of the Heroku dashboard. This sets the environment variables for **GOOGLE_CALLBACK_URL**, **GOOGLE_CLIENT_ID** and **GOOGLE_CLIENT_SECRET** so they're available at runtime. Remember to visit https://console.developers.google.com to add the Heroku callback URL as an authorized redirect. See Chapter 22, "Google Authentication," if you need a refresher.

Deploy to Heroku

You're now ready to deploy your app to Heroku. Push your master branch to your Heroku remote and wait for everything to build. This can take a while, particularly on a large application.

To kick things off, enter the following in Terminal:

```
git push heroku master
```

Once everything deploys, Heroku notifies you of your app's status. Heroku normally starts your app automatically when it finishes building. In the unlikely event it doesn't, enter the following in Terminal to start your app:

```
heroku ps:scale web=1
```

Going forward, pushing the **master** branch to Heroku will redeploy your app. Open your app by visiting the app URL as seen in the **Settings** tab of the Heroku dashboard in your browser. You can also open the site in a browser by entering the following in Terminal:

```
heroku open
```

Reverting your database

If you followed the chapters in the first three sections, you encountered the need to revert the database on Vapor Cloud. It's a simple matter to run a database revert or migration on Heroku, as well.

To revert the last batch of migrations, enter the following in Terminal:

```
heroku run Run -- revert --yes --env production
```

This tells your Heroku instance to run the program **Run** — your Vapor app's main entry point — and pass it the **revert** command. To revert your entire database, enter the following in Terminal:

```
heroku run Run -- revert --all --yes --env production
```

Finally, to run your migrations again:

```
heroku run Run -- migrate --env production
```

Where to go from here?

In this chapter, you learned how to set up the app in the Heroku dashboard, configure your Git repository, add the necessary configuration files to your project, and deploy your app. Explore your dashboard and the Heroku Help to learn even more options!