- ⏱ 6 minutes

Heroku And You

In the old days, developers would host websites with Apache running on a dedicated computer in a closet plugged into their home internet. These days, you can use services like Heroku to host your server code. Heroku is a distributed cloud platform with massive data centers containing racks of enormously powerful CPUs, tremendous bandwidth and a 24/7 team of dedicated maintenance engineers.

You can get access to all that power for free, or a tiny fraction of the cost. There are different tiers that provide more computing power, storage space, and analytics, but for your purposes the free dev tier will be sufficient.

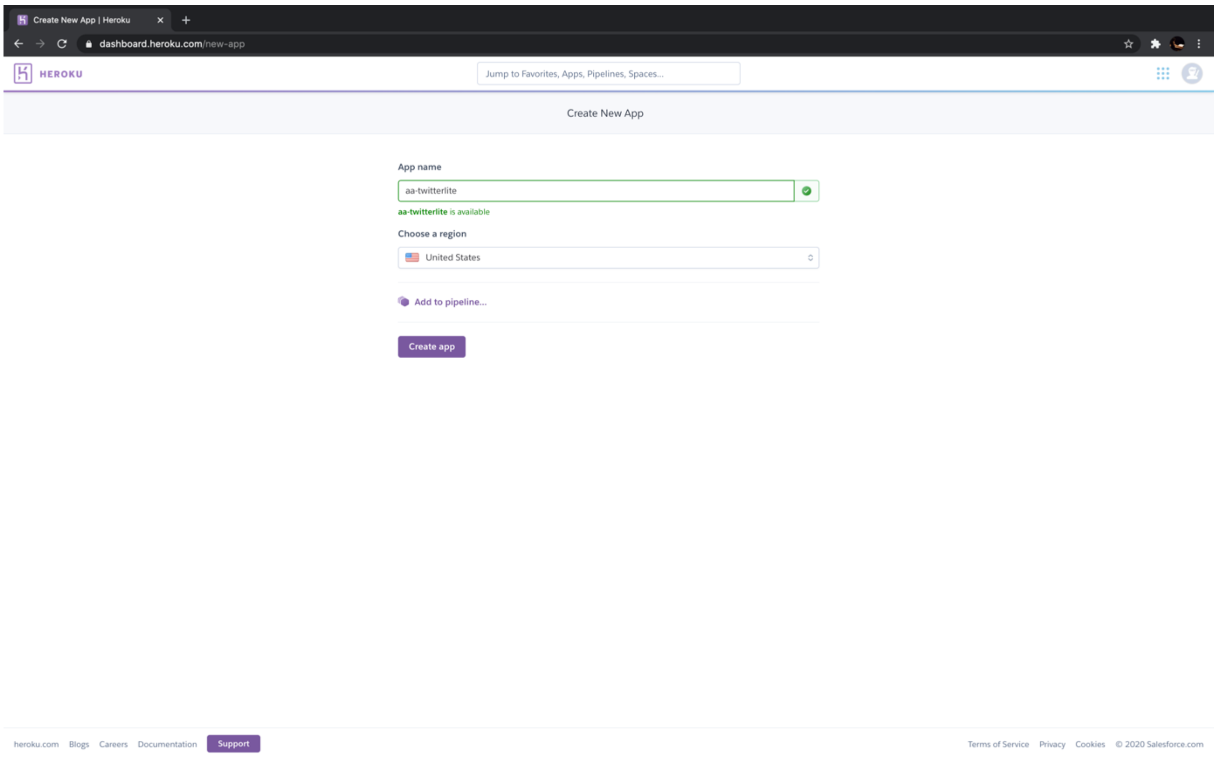
Heroku

Heroku has great step-by-step instructions for you to follow to get set up and deployable. In the following sections, we provide specific instructions for how to get your app (and Heroku) set up for a Node.js, Express.js, and Sequelize powered app.

If you'd like to practice deploying an app to Heroku, clone the Twitter Lite Walk-through repository and follow the deployment steps below.

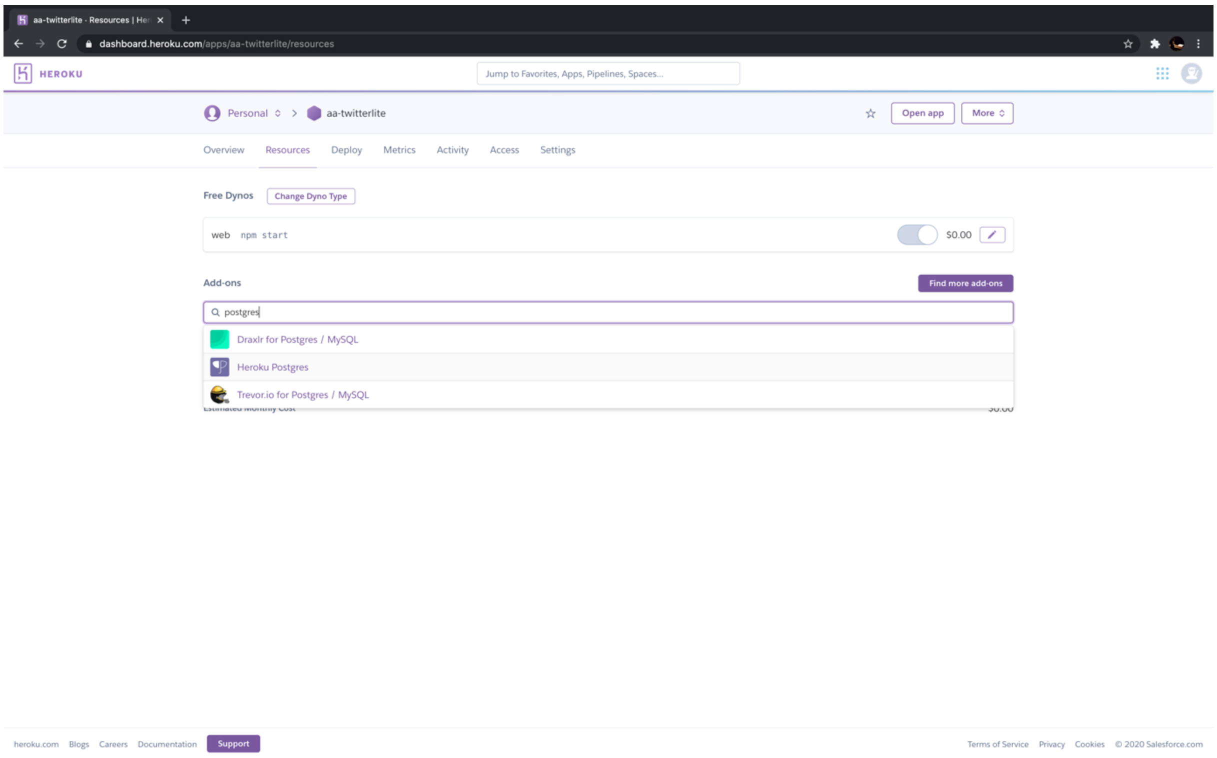
Step 1: Getting started on Heroku

1. Install the Heroku CLI. (For WSL users, see "Standalone Installation" instructions)
2. Create your free Heroku account.
3. Log in and create a new Heroku app.
   Note: For your group projects, the owner of the GitHub repository should create the Heroku app.


Create New App | Heroku
dashboard.heroku.com/new-app
HEROKU
Jump to Favorites, Apps, Pipelines, Spaces...
Create New App
App name
aa-twitterlite
aa-twitterlite is available
Choose a region
United States
Add to pipeline...
Create app
heroku.com Blogs Careers Documentation Support
Terms of Service Privacy Cookies © 2020 Salesforce.com

Step 2: Database

After creating your Heroku app, navigate to the **Resources** tab and set up a `Heroku Postgres` database for your application.


dashboard.heroku.com/apps/aa-twitterlite/resources

Personal > aa-twitterlite

Free Dynos

web `npm start` $0.00

Add-ons

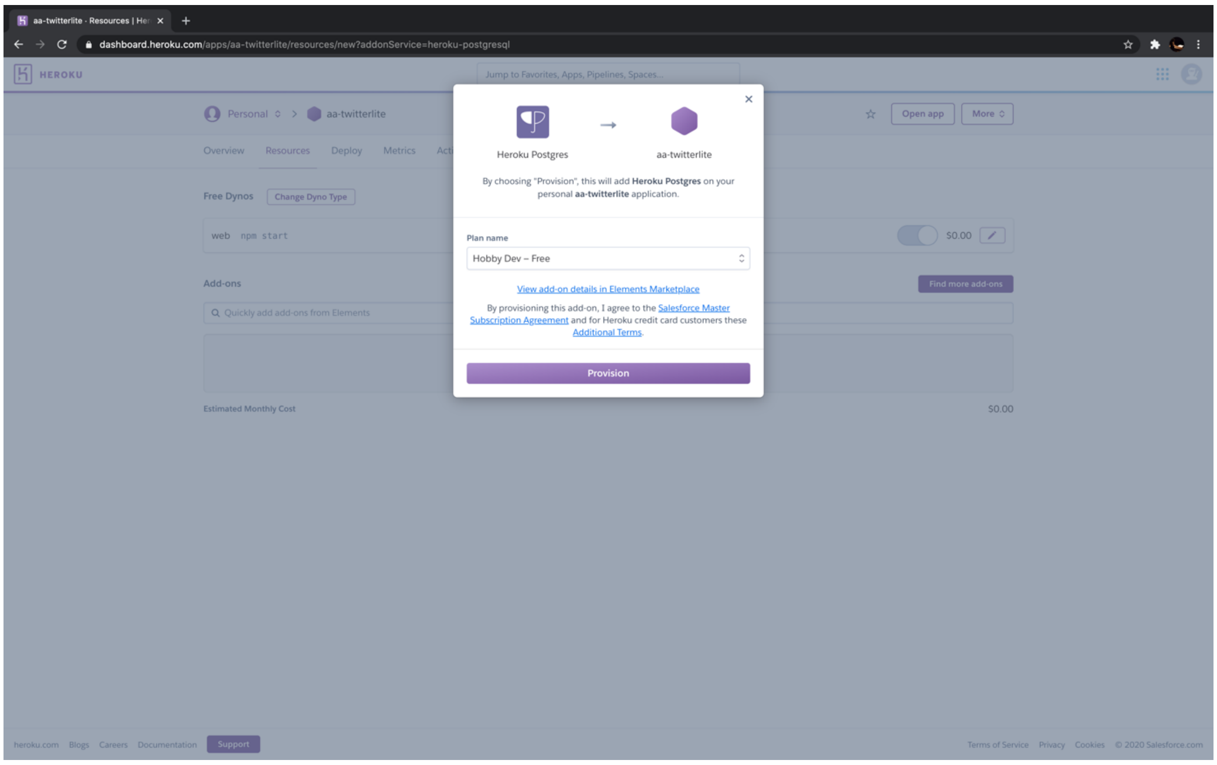
postgres

Draxlr for Postgres / MySQL

Heroku Postgres

Trevor.io for Postgres / MySQL

Select the `Hobby Dev - Free` plan.


dashboard.heroku.com/apps/aa-twitterlite/resources/new?addonService=heroku-postgresql

Heroku Postgres → aa-twitterlite

By choosing "Provision", this will add **Heroku Postgres** on your personal **aa-twitterlite** application.

Plan name

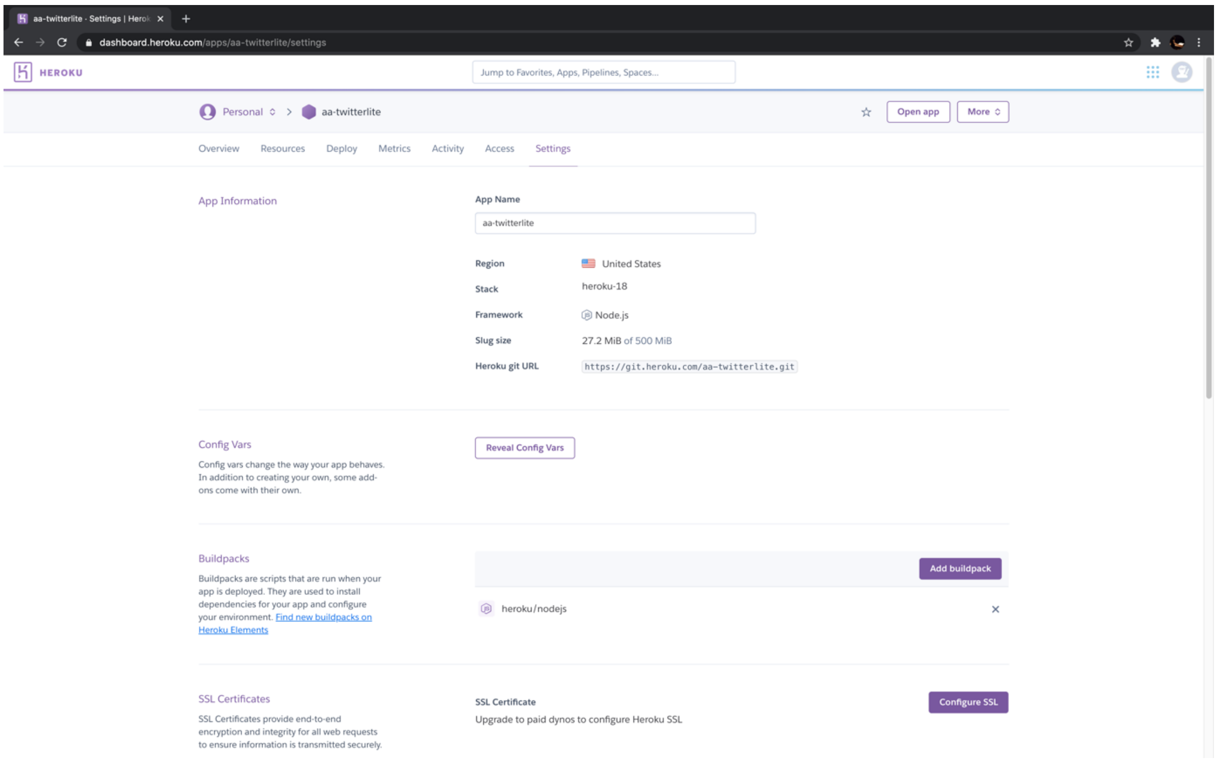
Hobby Dev – Free

View add-on details in Elements Marketplace

By provisioning this add-on, I agree to the Salesforce Master Subscription Agreement and for Heroku credit card customers these Additional Terms.

Step 3: Environment variables

In Heroku, you can set `Config Vars` instead to set your production environment's environment variables. Environment variables set in a `.env` file won't work on Heroku. Remember that you should **NEVER** check in `.env` files or any private keys - always keep your credentials safe from malicious users!
Navigate to the **Settings** tab to `Reveal Config Vars` and set the environment variables needed to run your application.



App Information

App Name: aa-twitterlite

Region: United States
Stack: heroku-18
Framework: Node.js
Slug size: 27.2 MiB of 500 MiB
Heroku git URL: https://git.heroku.com/aa-twitterlite.git

Config Vars
Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.

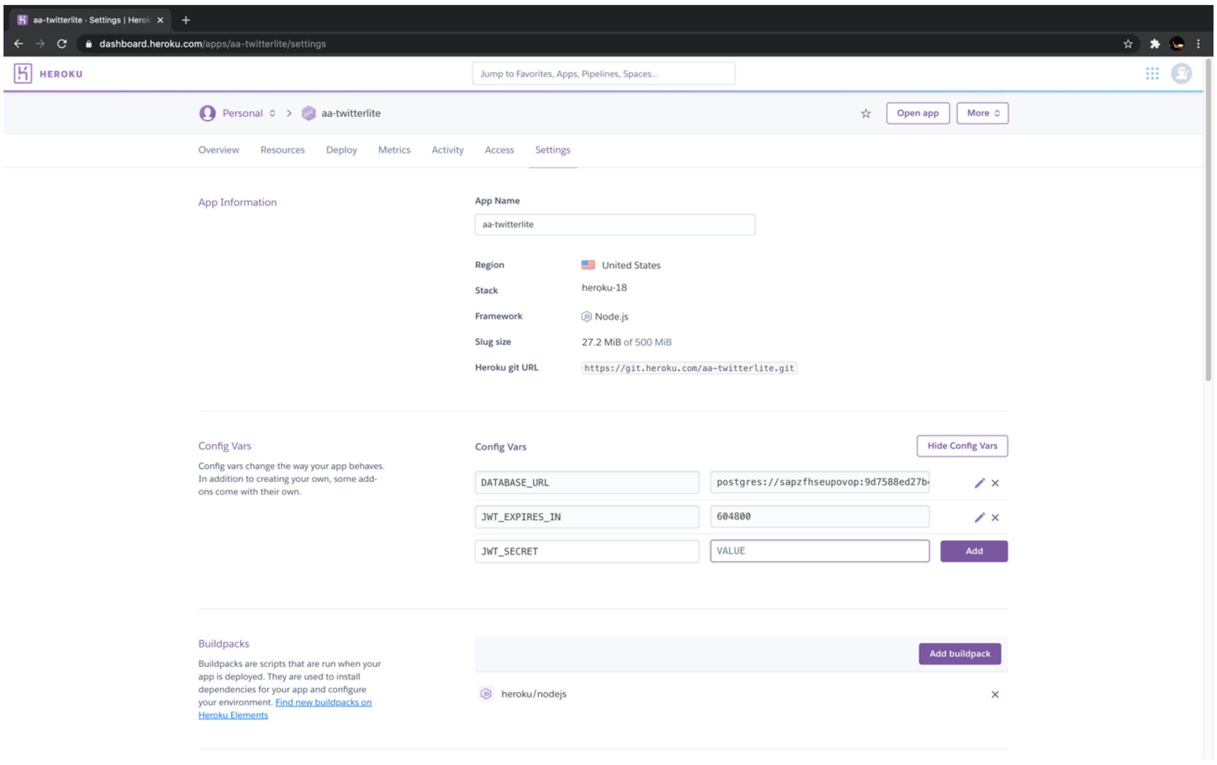
Buildpacks
Buildpacks are scripts that are run when your app is deployed. They are used to install dependencies for your app and configure your environment. Find new buildpacks on Heroku Elements

heroku/nodejs

SSL Certificates
SSL Certificates provide end-to-end encryption and integrity for all web requests to ensure information is transmitted securely.

SSL Certificate
Upgrade to paid dynos to configure Heroku SSL

You'll see that your `DATABASE_URL` environment variable is already set. This was done when you set up the `Heroku Postgres` database in the previous step.
The `DATABASE_URL` takes care of your database credentials - this means you don't need to set a `DB_USERNAME`, `DB_PASSWORD`, or `DB_DATABASE`. Remember to set the other environment variables needed for your application (i.e. API keys or `JWT_SECRET` if you used JSON Web Tokens to allow for user login).


aa-twitterlite · Settings | Herok
dashboard.heroku.com/apps/aa-twitterlite/settings
HEROKU
Jump to Favorites, Apps, Pipelines, Spaces...
Personal
aa-twitterlite
Open app
More
Overview
Resources
Deploy
Metrics
Activity
Access
Settings

App Information

App Name	aa-twitterlite
Region	United States
Stack	heroku-18
Framework	Node.js
Slug size	27.2 MiB of 500 MiB
Heroku git URL	`https://git.heroku.com/aa-twitterlite.git`

Config Vars

Config vars change the way your app behaves. In addition to creating your own, some add-ons come with their own.

Config Vars

DATABASE_URL	postgres://sapzfhseupovop:9d7588ed27b
JWT_EXPIRES_IN	604800
JWT_SECRET	VALUE

Buildpacks

Buildpacks are scripts that are run when your app is deployed. They are used to install dependencies for your app and configure your environment. Find new buildpacks on Heroku Elements

heroku/nodejs

Step 4: Configure your app to use the Heroku Postgres database

Now that you have a Heroku database configured, you'll need to tell your application how to use it. There are two ways to configure your `production` environment: with **dotenv** and a `.sequelizerc` file that points to a `config/database.js` file **or** the Sequelize CLI's auto-generated `config.json` file.

With .sequelizerc and dotenv

If you're using the **dotenv** package with a `.sequelizerc` file, you can update your `config/database.js` file with a `production` key, like in this commit.
Your application would reference `use_env_variable` to reference the `DATABASE_URL` set by Heroku in your `Config Vars`. The `module.exports` in your `config/database.js` file should look something like this:

```
module.exports = {
  development: {
    username,
    password,
    database,
    host,
    dialect: 'postgres',
  },
  production: {
    use_env_variable: 'DATABASE_URL',
    dialect: 'postgres',
    seederStorage: 'sequelize',
  }
};
```

With Sequelize CLI's config.json

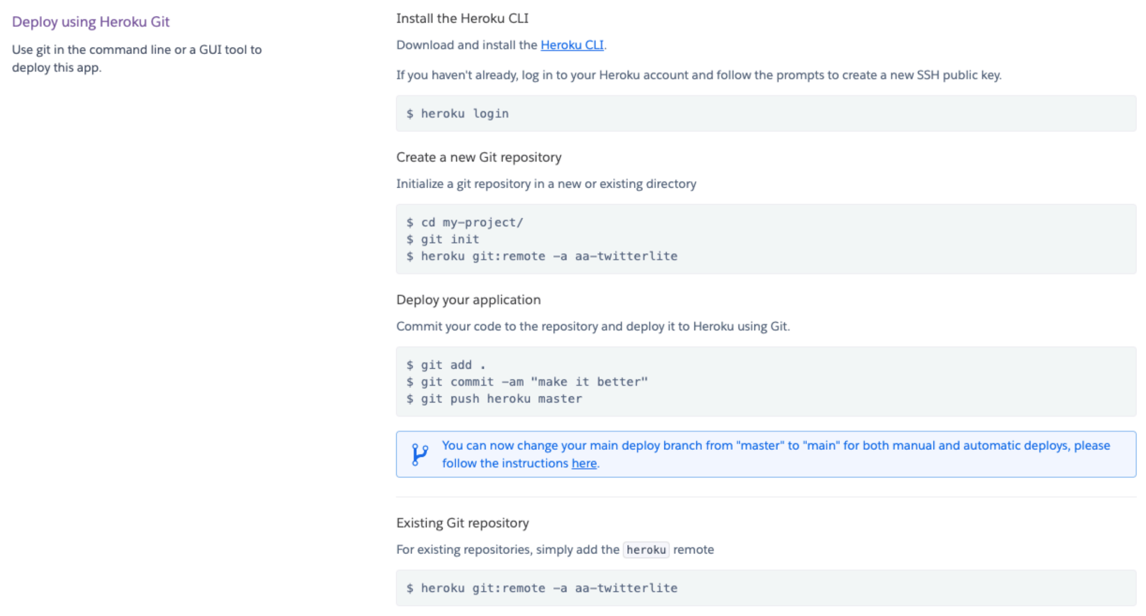
If you're using Sequelize and its **config.json** file, then you should change the "production" entry to look like this.

```
"production": {
  "dialect": "postgres",
  "seederStorage": "sequelize",
  "use_env_variable": "DATABASE_URL"
}
```

Step 5: Push to Heroku

Pushing your code to Heroku is similar to pushing your code to Github. Follow the steps outlined below to set up your Heroku app as a **git remote** and push using the Heroku CLI. (**Note**: you can also find these steps in the `Deploy using Heroku Git` section under the **Deploy** tab.)

1. Make sure you are in the root of your repository directory and log into Heroku with `heroku login`.
2. Add a new remote to your GitHub configuration with `heroku git:remote -a «your-app-name»`.
3. Add all your changes with `git add .`.
4. Commit your changes with a message with `git commit -m`. (Alternatively, you can use `git commit -am` to `add` and `commit` in the same command.)
5. Push your changes to Heroku with `git push heroku`!


Deploy using Heroku Git

Use git in the command line or a GUI tool to deploy this app.

Install the Heroku CLI

Download and install the Heroku CLI.

If you haven't already, log in to your Heroku account and follow the prompts to create a new SSH public key.

```
$ heroku login
```

Create a new Git repository

Initialize a git repository in a new or existing directory

```
$ cd my-project/
$ git init
$ heroku git:remote -a aa-twitterlite
```

Deploy your application

Commit your code to the repository and deploy it to Heroku using Git.

```
$ git add .
$ git commit -am "make it better"
$ git push heroku master
```

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.

Existing Git repository

For existing repositories, simply add the `heroku` remote

```
$ heroku git:remote -a aa-twitterlite
```

If everything works, you should see a successful build message.

```
Enumerating objects: 56, done.
Counting objects: 100% (56/56), done.
Delta compression using up to 12 threads
Compressing objects: 100% (50/50), done.
Writing objects: 100% (56/56), 28.60 KiB | 4.08 MiB/s, done.
Total 56 (delta 11), reused 0 (delta 0)
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Node.js app detected
remote:
remote: -----> Creating runtime environment
remote:
remote:        NPM_CONFIG_LOGLEVEL=error
remote:        NODE_ENV=production
remote:        NODE_MODULES_CACHE=true
remote:        NODE_VERBOSE=false
remote:
remote: -----> Installing binaries
remote:        engines.node (package.json):  unspecified
remote:        engines.npm (package.json):   unspecified (use
default)
remote:
remote:        Resolving node version 12.x...
remote:        Downloading and installing node 12.16.2...
remote:        Using default npm version: 6.14.4
remote:
remote: -----> Installing dependencies
```

```
remote:        Installing node modules (package.json + package-lock)
remote:
remote:        > core-js@2.6.11 postinstall
/tmp/build_b1c9c6698f55a0ef025fdf935a601ccd/node_modules/core-js
remote:        > node -e "try{require('./postinstall')}catch(e){}"
remote:
remote:        added 232 packages from 299 contributors and audited
583 packages in 7.55s
remote:
remote:        4 packages are looking for funding
remote:          run `npm fund` for details
remote:
remote:        found 1 low severity vulnerability
remote:          run `npm audit fix` to fix them, or `npm audit` for
details
remote:
remote: -----> Build
remote:
remote: -----> Caching build
remote:        - node_modules
remote:
remote: -----> Pruning devDependencies
remote:        audited 583 packages in 2.016s
remote:
remote:        4 packages are looking for funding
remote:          run `npm fund` for details
remote:
remote:        found 1 low severity vulnerability
remote:          run `npm audit fix` to fix them, or `npm audit` for
details
remote:
remote: -----> Build succeeded!
remote: -----> Discovering process types
remote:        Procfile declares types     -> (none)
remote:        Default types for buildpack -> web
remote:
remote: -----> Compressing...
remote:        Done: 27.4M
remote: -----> Launching...
remote:        Released v5
remote:        https://«your-app-name».herokuapp.com/ deployed to
Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/«your-app-name».git
```

Step 6: Run migrations on Heroku

When you want to migrate your `Heroku Postgres` database, you'll need to run the migration command prefaced with `heroku run` from inside your repository.

```
heroku run npx sequelize-cli db:migrate
```

You should see the normal output from the Sequelize CLI.

If you need to seed, run the seed command prefaced with `heroku run` as well.

```
heroku run npx sequelize-cli db:seed:all
```

You should see the normal output from the Sequelize CLI.

If you ever need to roll back, **DO NOT DROP YOUR DATABASE**! Instead, migrate down and up.

```
heroku run npx sequelize-cli db:seed:undo:all
heroku run npx sequelize-cli db:migrate:undo:all
heroku run npx sequelize-cli db:migrate
heroku run npx sequelize-cli db:seed:all
```

If undoing the migrations and seeds don't work, you can reset the entire database by removing and adding the `Heroku Postgres` add-on that you added in the beginning of Step 2. Then migrate and seed again.

Step 7: Additional pushes to Heroku

Before pushing to Heroku again in the future, you **must** add and commit your changes. Your general workflow should be something like this:

1. Team member commits and pushes an update
2. Somebody reviews and merges the Pull Request to the repository's base branch.
3. The "Team Lead" pulls the latest code
4. The "Team Lead" pushes the code to Heroku using the command `git push heroku`
5. The person that deploys the app should run any new database migrations. See the next section for that.

You can push non-master branches to Heroku by aliasing your branch to be the master branch when pushing like so:

```
git push heroku <branch-name>:master
```

Did you find this lesson helpful?

No

Yes

✔ Mark As Complete