1. **Heroku CLI** [**Download and install**](https://devcenter.heroku.com/articles/heroku-cli#download-and-install)

**-** [**MacOS**](https://devcenter.heroku.com/articles/heroku-cli#macos)

[Homebrew](https://devcenter.heroku.com/articles/heroku-cli#macos-homebrew)

To install the Heroku CLI with [homebrew](http://brew.sh/):

$ brew install heroku

Compatible with 10.7+

[MacOS Installer](https://devcenter.heroku.com/articles/heroku-cli#macos-installer)

Download and run the [MacOS installer](https://cli-assets.heroku.com/branches/stable/heroku-osx.pkg).

Compatible with 10.7+

**-** [**Windows**](https://devcenter.heroku.com/articles/heroku-cli#windows)

Download and run the Windows installer [32-bit](https://cli-assets.heroku.com/branches/v6/heroku-windows-386.exe) [64-bit](https://cli-assets.heroku.com/branches/v6/heroku-windows-amd64.exe).

[Verify your installation](https://devcenter.heroku.com/articles/heroku-cli#verify-your-installation)

To verify your CLI installation use the heroku --version command.

$ heroku --version

heroku-cli/6.0.0-010a227 (darwin-x64) node-v8.0.0

You should see heroku-cli/x.y.z in the output. If you don’t, but have installed the Heroku CLI, it’s possible you have an old heroku gem on your system - [uninstall the gem](https://devcenter.heroku.com/articles/heroku-cli#uninstalling-the-legacy-heroku-gem).

**-** [**Getting started**](https://devcenter.heroku.com/articles/heroku-cli#getting-started)

You will be asked to enter your Heroku credentials the first time you run a command; after the first time, your email address and an API token will be saved to ~/.netrc for future use. For more information, see [Heroku CLI Authentication](https://devcenter.heroku.com/articles/authentication)

It’s generally a good idea to login and add your public key immediately after installing the Heroku CLI so that you can use git to push or clone Heroku app repositories:

$ heroku login

Enter your Heroku credentials.

Email: adam@example.com

Password (typing will be hidden):

Authentication successful.

You’re now ready to create your first Heroku app.

You can get more details in <https://devcenter.heroku.com/articles/heroku-cli>.

## 2. Create stack and repository

## - [Tracking your app in git](https://devcenter.heroku.com/articles/git#tracking-your-app-in-git)

Heroku apps expect the app directory structure at the root of the repository. If your app is inside a subdirectory in your repository, it won’t run when pushed to Heroku.

Before you can push an app to Heroku, you’ll need to initialize a local Git repository and commit your files to it. Please extract tweetboard.zip and go to tweetboard directory.

$ cd tweetboard/web

$ git init

Initialized empty Git repository in .git/

$ git add .

$ git commit -m "my first commit"

This is a local repository, now residing inside the .git directory. Nothing has been sent anywhere yet; you’ll need to create a remote and do a push to deploy your code to Heroku.

## - [Creating a Heroku remote](https://devcenter.heroku.com/articles/git#creating-a-heroku-remote)

Before upgrading an existing application to Cedar-14, you should test the upgrade on a Cedar-14 staging app that is not getting production traffic. You can create a staging app to use for testing, let’s call the git remote heroku and the app haloinstaboard:

$ heroku create --stack cedar-14 haloinstaboard

Creating your-cedar-14-app... done, stack is cedar-14

http://your-cedar-14-app.herokuapp.com/ | git@heroku.com:my-app.git

Git remote heroku added

Also add any add-ons (with heroku addons:create) and config (with heroku config:set) to the app that’s required for your code to run.

## - [Setting a buildpack](https://devcenter.heroku.com/articles/using-multiple-buildpacks-for-an-app#setting-a-buildpack)

You can use the buildpacks:set command from the Heroku CLI to insert a single buildpack in your buildpack execution. For example, if you have a Python application, you might set this like so:

$ heroku buildpacks:set heroku/python

This command accepts an optional --index argument, which can be used to set the position of the given buildpack in the order of execution. If --index is provided, the command will overwrite the buildpack at the given position.

## - [Adding a buildpack](https://devcenter.heroku.com/articles/using-multiple-buildpacks-for-an-app#adding-a-buildpack)

You can add additional buildpacks to your application with the buildpacks:add command. In our project we have to add 2 buildpacks to there.

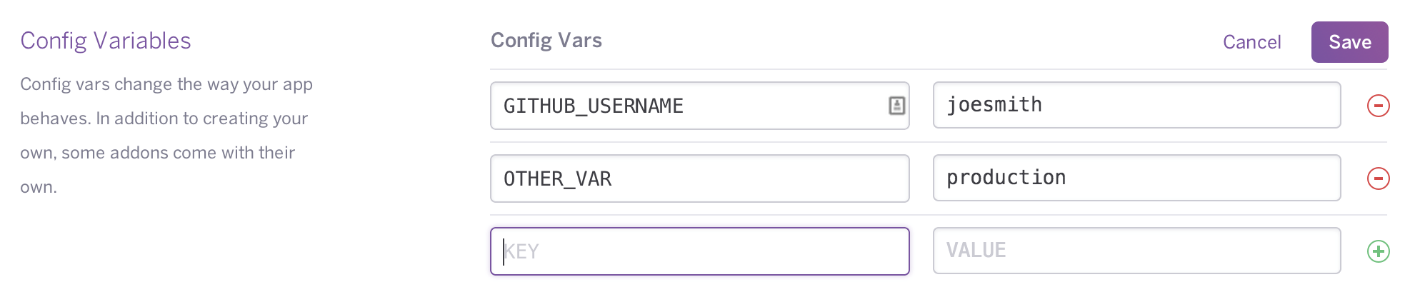
$ heroku buildpacks:add https://github.com/heroku/heroku-buildpack-xvfb-google-chrome

$ heroku buildpacks:add https://github.com/heroku/heroku-buildpack-chromedriver

## - [Setting up config vars for a deployed application](https://devcenter.heroku.com/articles/config-vars#setting-up-config-vars-for-a-deployed-application)

$ heroku config:set CHROME\_PATH=/app/.apt/usr/bin/google-chrome

You can also edit config vars on your app’s settings tab on Dashboard:



## 3. [Create](https://devcenter.heroku.com/articles/cedar-14-stack#create-cedar-14-app) Postgresql database

### - [Create a new Postgres database](https://devcenter.heroku.com/articles/heroku-postgresql#create-a-new-database)

Heroku Postgres offers a variety of plans, spread across different [tiers of service](https://devcenter.heroku.com/articles/heroku-postgres-plans#plan-tiers): hobby, standard, premium, and enterprise. For a list of available plans, see [Choosing the Right Heroku Postgres Plan](https://devcenter.heroku.com/articles/heroku-postgres-plans). If you eventually outgrow your initial plan, you can [upgrade databases](https://devcenter.heroku.com/articles/upgrading-heroku-postgres-databases#upgrade-with-pg-copy).

Heroku Postgres can be attached to a Heroku application via the CLI using this command: heroku addons:create heroku-postgresql:<PLANNAME>.

For example, to provision a [hobby-dev](https://devcenter.heroku.com/articles/heroku-postgres-plans#hobby-tier) plan database:

$ heroku addons:create heroku-postgresql:hobby-dev

Adding heroku-postgresql:hobby-dev to sushi... done, v69 (free)

Attached as HEROKU\_POSTGRESQL\_RED

Database has been created and is available

To provision a [hobby-basic](https://devcenter.heroku.com/articles/heroku-postgres-plans#hobby-tier) plan database:

$ heroku addons:create heroku-postgresql:hobby-basic

Depending on the plan, some databases can take up to 5 minutes to become available. Use pg:wait to track their status:

$ heroku pg:wait

Waiting for database HEROKU\_POSTGRESQL\_RED... done

Once Heroku Postgres has been added a DATABASE\_URL or HEROKU\_POSTGRESQL\_COLOR\_URL setting will be available in the app configuration and will contain the URL used to access the newly provisioned Heroku Postgres service. DATABASE\_URL will be given to the database if it is the first one for the application, otherwise the HEROKU\_POSTGRESQL\_COLOR\_URL will be created. This can be confirmed using the heroku config command.

$ heroku config -s | grep HEROKU\_POSTGRESQL

HEROKU\_POSTGRESQL\_RED\_URL=postgres://user3123:passkja83kd8@ec2-117-21-174-214.compute-1.amazonaws.com:6212/db982398

### [Establish primary DB](https://devcenter.heroku.com/articles/heroku-postgresql#establish-primary-db)

Heroku recommends using the DATABASE\_URL config var to store the location of your primary database. In single-database setups your new database will have already been assigned to DATABASE\_URL.

On apps with multiple databases, you can set the primary database like this:

$ heroku pg:promote HEROKU\_POSTGRESQL\_RED

Promoting HEROKU\_POSTGRESQL\_RED\_URL to DATABASE\_URL... done

At this point an empty PostgreSQL database is provisioned. To populate it with data from an existing data source see the [import instructions](https://devcenter.heroku.com/articles/heroku-postgres-import-export#import) or follow the language-specific instructions in this article to connect from your application.

### [Sharing Heroku Postgres between applications](https://devcenter.heroku.com/articles/heroku-postgresql#sharing-heroku-postgres-between-applications)

You can share one Heroku Postgres between multiple applications.

$ heroku addons:attach my-originating-app::DATABASE --app sushi

Attaching postgresql-addon-name to sushi... done

Setting HEROKU\_POSTGRESQL\_BRONZE vars and restarting sushi... done, v10

The database will be attached with a color (in this example HEROKU\_POSTGRESQL\_BRONZE but it will change each time).

The shared database will not necessarily be the default database on any apps that it is shared with. To promote the shared database to be the primary database, use the pg:promote command with the color, or the addon name, on each of the apps where you want it to be the default database:

$ heroku pg:promote HEROKU\_POSTGRESQL\_BRONZE --app haloinstaboard

Ensuring an alternate alias for existing DATABASE... done, HEROKU\_POSTGRESQL\_SILVER

Promoting postgresql-addon-name to DATABASE\_URL on sushi... done

$ heroku pg:promote postgresql-addon-name --app haloinstaboard

Ensuring an alternate alias for existing DATABASE... done, HEROKU\_POSTGRESQL\_SILVER

Promoting postgresql-addon-name to DATABASE\_URL on sushi... done

## 4. [Deploy app](https://devcenter.heroku.com/articles/cedar-14-stack#create-cedar-14-app) and initialize

## - [Deploy to your Cedar-14 staging app](https://devcenter.heroku.com/articles/cedar-14-stack#deploy-to-your-cedar-14-staging-app)

Deploying to the new Cedar-14 staging app will not affect your currently running app on Cedar. It will merely stage the source code in a separate app and provide the opportunity to verify functionality.

Push the source to the heroku remote repository.

$ git push heroku master

Once the app is deployed, you should verify that it is working correctly on the Cedar-14 stack, and if not, make any required changes.

You can get more details in <https://devcenter.heroku.com/articles/cedar-14-stack>.

## - Initialize the app and connect database

$ heroku run bash

$ python manage.py db init

$ python manage.py db migrate

$ python manage.py db upgrade

$ python manage.py create\_admin

## 5. Restart app

$ heroku restart