







Dash Python > Deploy Your Dash App

Plotly Studio: Transform any dataset into an interactive data application in minutes with Al. Sign up for early access



Deploying Dash Apps

By default, Dash apps run on localhost —you can only access them on your own machine. To share a Dash app, you need to deploy it to a server.

Our recommend method for securely deploying Dash apps is Dash Enterprise.

Dash Enterprise can be installed on the cloud services of AWS, Azure, or Google.

Find out if your company is using Dash Enterprise

Dash Enterprise Deployment

Dash Enterprise is Plotly's commercial product for developing and deploying Dash apps. In addition to proven, Git-based deployment, the Dash Enterprise platform provides a complete Analytical App Stack. This

- LDAP and SAML Authentication Middleware
- Data App Workspaces
- Job Queue Support
- Enterprise-Wide Dash App Portal
- o Design Kit
- o Reporting, Alerting, Saved Views, and PDF Reports
- Dashboard Toolkit
- **Embedding Dash apps in Existing websites or Salesforce**
- o AI App Catalog
- Big Data Best Practices



2020, 00.40	Beploy Tour Bush to	. , 2 0 0 0	
Dash Open	Documentation Self-serve guide for building a basic Dash app in Python	Dash Open	
Source	Community and Forum-Based Q&A With over 2 million downloads per month, Dash has a passionate community.	Source	
	Plotly App Studio Quickly transform Jupyter Notebooks into professional data apps.		
	Smart Insights Extract key highlights from complex data sets via manual data or Al.	Low-Code	
	Data App Workspaces Write and execute code in an onboard code editor.	Development	
	Snapshot Engine Save and share Dash app views via PDFs.		
	Embedding Middleware Avoid iFrames and embed Dash apps to a web platform.		
Dash Enterprise	App Manager Deploy, share, and manage your Dash apps.	Deployment & Scaling	
	Persistent Filesystem Upload 25GB or 500 million rows of data in the browser.		
	Command Line Interface Automate CI/CD pipelines without needing to handle SSH keys.		
	Viewer Analytics Track end-user behavior and understand organizational usage.		
	Centralized Deployment Manage apps outside the Dash framework like Streamlit and Django.	entralized Deployment e the Dash framework like Streamlit and Django.	
	No-Code Authentication Add a login screen and integrate LDAP, SAML, and OIDC protocols.		
	Row-Level Security Restrict app visibility for users or specific rows.	Enterprise IT	
	Automated Testing Run custom health checks to prevent critical bugs.	Integration	
	Database Compatibility Connect Dash apps to Databricks, Snowflake, and more.		

Heroku for Sharing Public Dash Apps

Heroku is one of the most trusted platforms for deploying and managing public Flask applications. The Git and buildpack-based deployment of Heroku and Dash Enterprise are nearly identical, enabling a smooth transition to Dash Enterprise if you are already using Heroku. **View the official Heroku guide to Python**.

Sign up for Dash Club \rightarrow Two free cheat sheets plus updates from Chris Parmer and Adam Schroeder delivered to your inbox every two months. Includes tips and tricks, community apps, and deep dives into the Dash architecture. <u>Join now</u>.

Here is a simple example for deploying a Dash app to Heroku. This example requires a Heroku account, <code>git</code>, and <code>virtualenv</code>.

Step 1. Create a new folder for your project:

```
$ mkdir dash_app_example
$ cd dash_app_example
```

Step 2. Initialize the folder with git and a virtual env $\,$

```
$ git init  # initializes an empty git repo
$ virtualenv venv # creates a virtualenv called "venv"
$ source venv/bin/activate # uses the virtualenv
```

virtualenv creates a fresh Python instance. You will need to reinstall your app's dependencies with this virtualenv

```
$ pip install dash
$ pip install plotly
```



You will also need a new dependency, gunicorn, for deploying the app:

```
$ pip install gunicorn
```

Step 3. Initialize the folder with a sample app (app.py), a .gitignore file, requirements.txt, and a Procfile for deployment

Create the following files in your project folder:

арр.ру

.gitignore

```
venv
*.pyc
.DS_Store
.env
```

Procfile

```
web: gunicorn app:server
```

(Note that app refers to the filename app.py). server refers to the variable server inside that file)

requirements.txt

requirements.txt describes your Python dependencies. You can fill this file in automatically with:

```
$ pip freeze > requirements.txt
```

Step 4. Initialize Heroku, add files to Git, and deploy

```
$ heroku create my-dash-app # change my-dash-app to a unique name
$ git add . # add all files to git
$ git commit -m 'Initial app boilerplate'
$ git push heroku master # deploy code to heroku
$ heroku ps:scale web=1 # run the app with a 1 heroku "dyno"
```



You should be able to view your app at https://my-dash-app.herokuapp.com (changing my-dash-app to the name of your app).

Step 5. Update the code and redeploy

When you modify app.py with your own code, you will need to add the changes to Git and push those changes to Heroku.

```
$ git status # view the changes
$ git add . # add all the changes
$ git commit -m 'a description of the changes'
$ git push heroku master
```

This workflow for deploying apps on Heroku is very similar to how deployment works with Plotly's Dash Enterprise.

Dash Enterprise 5.2.X further simplifies deployment by providing a CLI that handles these Git operations with a single de deploy command.

Learn more or get in touch.

Dash Python > Deploy Your Dash App

Products	Pricing	About Us	Support	Join our mailing
Dash	Enterprise Pricing	Careers	Community Support	list
Consulting and Training		Resources	Graphing Documentation	
		Blog		Sign up to stay in the loop with all things Plotly — from Dash Club to product updates, webinars, and more!
				SUBSCRIBE
Copyright © 2025 Plotly. All rights reserved.				Terms of Service Privacy Policy