

How To Deploy a Movie and Product Recommendation System?¶

Best Approaches to Deploy a Recommendation System:

The deployment pattern for any recommender is specific to the volumes and volatility of the data on which they are based. The specific business scenarios enabled with the recommenders also affect how you should implement the recommendation platform. There are Five best Practice to Deploy a Right Model:

- **Data Assessment**
- **Evaluation of the right tech stack**
- **Robust Deployment approach**
- **Post deployment support & testing**
- **Change management & communication**

Platforms To Deploy Recommendation Systems

- Amazon SageMaker
- Azure Machine Learning
- Google Cloud AI Platform
- IBM Watson Studio
- Heroku
- Docker
- etc

Why To Choose Heroku for Testing Recommendation Systems

Deploying a machine learning model as a service can solve most of these problems, and predictions will be real-time. But there will be issues like scalability, monitoring, and down-time of service. Though there are many cloud providers to resolve these issues and provide 24*7 support. Still, if you are a small company or just starting in AI/ML and don't want to spend more time to handle cloud deployments or DevOps tasks and want a quick deployment option, then deploying your machine learning model on Heroku using Flask will address all your issues. and it is free Cloud Source Platform and very easy to use.

Heroku using Flask

Heroku is a Platform-as-a-Service tool by Salesforce. Heroku is backed by AWS and all Heroku applications/services are hosted on AWS. AWS provides the infrastructure and handles all the load-balancing, resource utilization, networking, logging, monitoring and Heroku acts as a middle-man to provide a scalable, automated rapid deployment platform with all cloud capabilities. Using Flask will provide UI to test and it can be integrated with enterprise-level applications.

Steps for Recommendation System deployment on Heroku using Flask:

Deployment on Heroku using Flask has 7 steps from creating a machine learning model to deployment. These steps are the same for all machine learning models and you can deploy any ML model on Heroku using these steps.

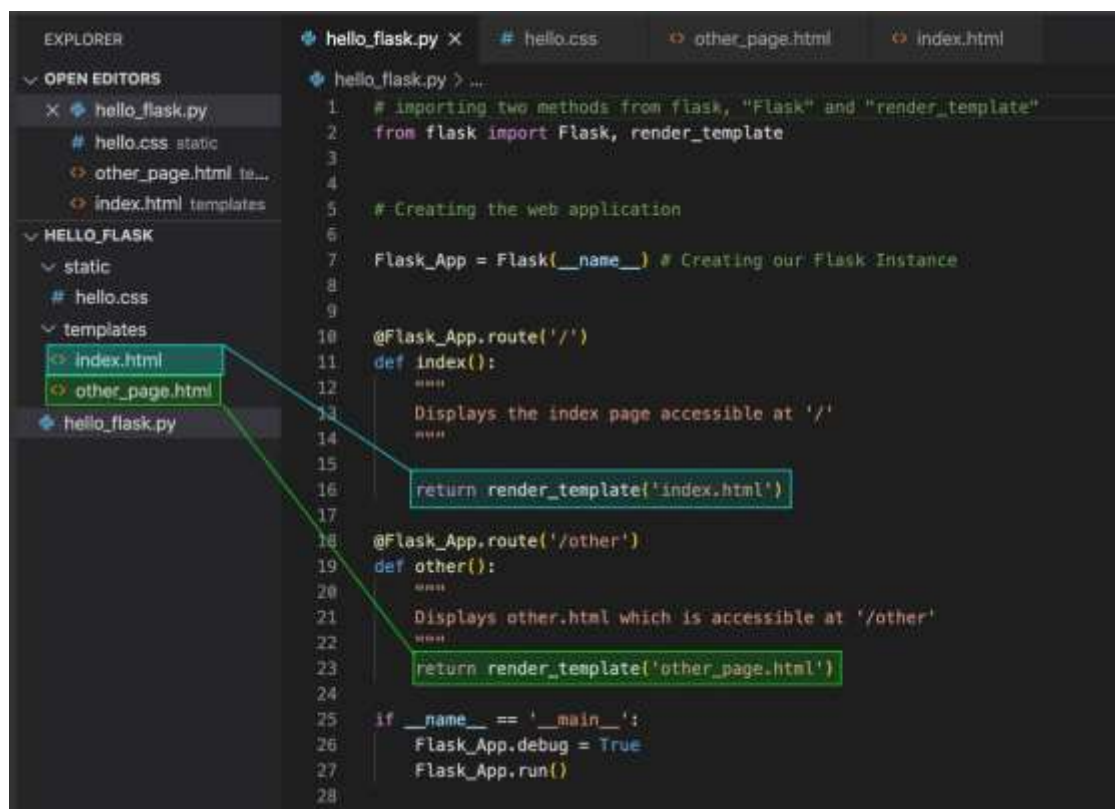
STEP 1:

Create Recommendation ML Model and save (pickle) it

```
1 model.fit(X_train, Y_train)
2 # save the model to disk
3 filename = 'finalized_model.sav'
4 pickle.dump(model, open(filename, 'wb'))
5
6 # load the model from disk
7 loaded_model = pickle.load(open(filename, 'rb'))
8 result = loaded_model.score(X_test, Y_test)
```

STEP 2:

Create Flask files for UI and python main file (app.py) or with the name of any other .py file



that can unpickle the machine learning model from step 1 and do predictions. we will create flask files — index.html and app.py. index.html, is a flask UI file for providing inputs (or features) to the model. app.py, is a python main file that unpickles the gradient boosting model from Step 1, renders the flask UI page index.html and makes predictions based on input from UI.

STEP 3:

Create requirements.txt to setup Flask web app with all python dependencies. In Step 3, we will create requirements.txt to add all of the dependencies for the flask app.

STEP 4:

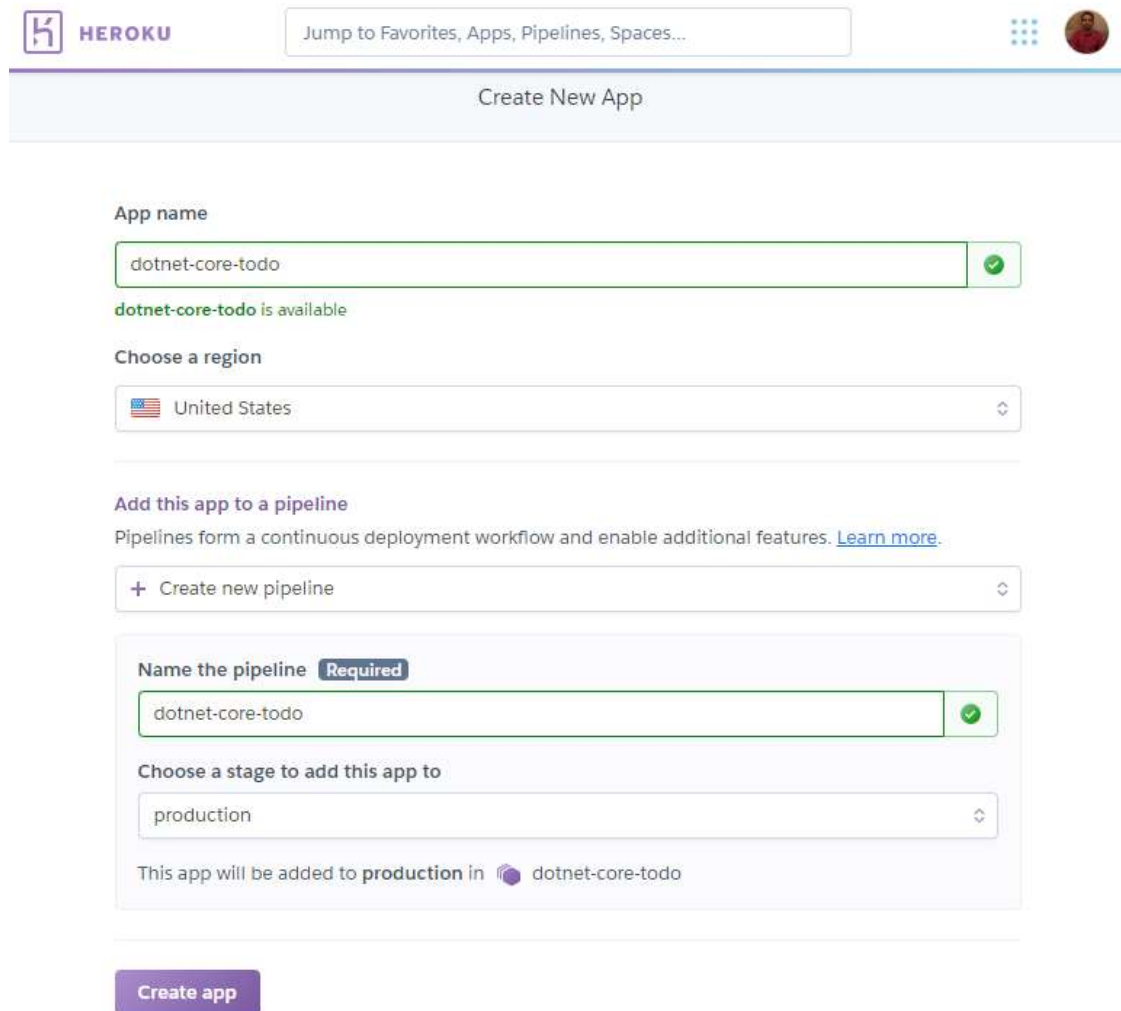
Create Procfile to initiate Flask app command. In Step 4, we will create Procfile to specify the commands that are executed by Heroku app on startup.

STEP 5:

Commit files from Step 1, 2, 3 & 4 in the Github repo

STEP 6:

Create account/Login on Heroku, create an app, connect with Github repo, and select branch.

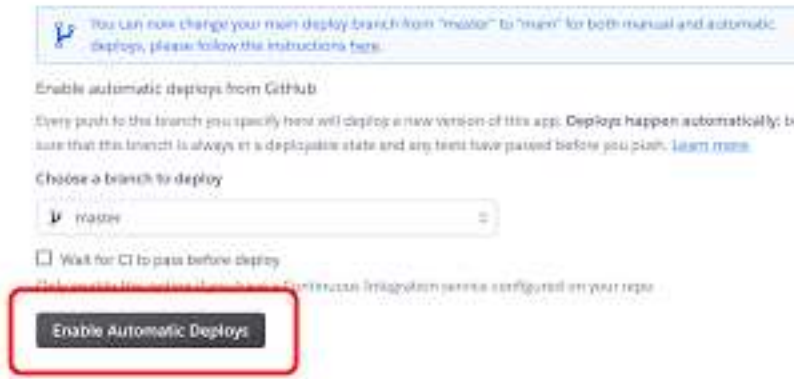


The screenshot shows the Heroku 'Create New App' interface. At the top, there's a navigation bar with the Heroku logo, a search bar containing 'Jump to Favorites, Apps, Pipelines, Spaces...', and a user profile icon. Below this is a light blue header with the text 'Create New App'. The main form area is white and contains several sections: 1. 'App name' section with a text input field containing 'dotnet-core-todo' and a green checkmark icon to its right. Below the input, a green message states 'dotnet-core-todo is available'. 2. 'Choose a region' section with a dropdown menu showing 'United States' and a small expand/collapse icon. 3. 'Add this app to a pipeline' section with a link to 'Learn more' and a dropdown menu showing '+ Create new pipeline'. 4. A light purple box containing: - 'Name the pipeline' section with a 'Required' label, an input field with 'dotnet-core-todo', and a green checkmark. - 'Choose a stage to add this app to' section with a dropdown menu showing 'production'. - A summary line: 'This app will be added to production in dotnet-core-todo'. At the bottom of the form is a purple button labeled 'Create app'.

In Step 6, we will log in on Heroku and create a new app. Next, we will connect the GitHub repo created in step 5 to the Heroku app and select a branch.

STEP 7:

Select manual deploy (or enable Automatic deploys) on Heroku.



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

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically**, be sure that this branch is always in a deployable state and any tests have passed before you push. [Learn more](#)

Choose a branch to deploy

main

☐ Wait for CI to pass before deploy

[Get your app's CI build logs](#) [Get your app's CI build logs](#) [Continuous Integration settings configured on your repo](#)

Enable Automatic Deploys

Finally in Step 7, select manual (or automatic) deploy and you can see the build logs scrolling. Once the application is deployed, you will get the application URL in logs and it will show the success message.