# Setup Instructions for Planetary IR DeepDive application

1. Install homebrew using the following command on terminal if you don’t have it already

/usr/bin/ruby -e "$(curl -fsSL <https://raw.githubusercontent.com/Homebrew/install/master/install)>"

1. Install DeepDive using the following command:

bash <(curl -fsSL git.io/getdeepdive) deepdive

1. Install postgres as the database system to work with DeepDive. It works best as compared to any other database. Go to the below mentioned website and follow the installation guide. If you already have a postgres instance installed, you have to make changes as indicated in Step 7 and skip this step for postgres installation.

postgresapp.com

Once installed, launch the psql terminal and create a database for deepdive with your preferred name. You will be using this instance going forward.

1. Add the deepdive command to $PATH variable for your environment. It should be saved at ~/local/bin on OS X for Mac

Also add postgres to your $PATH variable(optional) which will be stored at Applications/Postgres.app/Contents/Versions/latest/bin

1. Check that you have your JAVA\_HOME variable also defined in your bash\_profile. This is important for a few java based components of DeepDive.

1. Checkout/Clone code from <https://github.com/sne3091/PlanetaryIR.git>
2. Change to the directory where your code is downloaded, you should be able to see the different sub-folders in there. Switch to app/planetaryir and open the db.url file

Change the database name to the one you created in step 3 in the portion highlighted below. In my case the database name was deepdive\_planetaryIR\_SnehaS.

postgresql://localhost/deepdive\_planetaryIR\_$USER

1. On the terminal prompt, ensure that you are on the app/planetaryir directory of the project. This contains the deepdive application structure and skeleton. You will find an explanation of the structure here. Read on if you want to understand it: <http://deepdive.stanford.edu/deepdiveapp>
2. Run the script located at input/init.sh to install the bazaar dependency for CoreNLP’s parser
3. I have pre-loaded some articles into a json list which is DeepDive’s input format and this is available at input/articles.jsonl

Instructions about how to create this using more PDF/Text files of your own will be provided later. We will work with this set of articles for now, run the below command which will load the articles into your DeepDive database. This will ask for a comment but you can always hit :wq! in the vi editor to save and run the script.

deepdive compile

deepdive do articles

You can check if the table was created in your psql prompt which can be opened by clicking on the elephant icon at the top right of the menu bar on your system. **Use redo instead of do** if it says that there is nothing to do.

Note: deepdive compile should be run every time there is a change in the scripts or input data

1. Next, run the below command to break these articles into sentences for text processing. This will create a table called sentences. As always just hit :wq! in the vi editor for running without comments and this will continue the run.

deepdive do sentences

1. Next, we will run the command to extract authors which are essentially person mentions in this context.

deepdive do person\_mention

You should now be able to see a table created in the database called person mentions which are associated with doc\_id indicating the document from which it was extracted. The script to extract these person mentions are located under udf/map\_person\_mention.py

1. The next step is to be able to perform target extractions on this document. The script udf/map\_target\_mention.py is going to help you do that and can be invoked by running the below command.

deepdive do target\_mention

You should now be able to see target\_mention table created in the database. If you perform a select operation you should be able to see the various targets and their associated documents along with their start and end indexes in the sentence of that document.

1. We will perform some feature extractions to try and understand these targets better which will enable DeepDive to use it for machine learning models. This step is called feature extraction and can be done by executing the below command.

deepdive do target\_feature

This can also be thought of trying to extract nearby tokens, lemmas etc. that occur around target extractions.