

# Weather Predictor Architecture

## System Components

The diagram on the following page describes the modules of which Weather Predictor consists and their dependencies on each other. The following table briefly describes each component.

Note that part of the solution is deployed in the Cloud while another part of it runs offline in scripted mode and is triggered by an Engineer.

Component	Function	Implementation	Dependencies
<b>Cloud - Deployed</b>			
<b>Webapp</b>	Exposes public URLs through which the User can access forecast information and see it in HTML	Flask Python Application that renders data found in Data Store	<u>Config</u> (to display deployed Model Info)
<b>Data Service</b>	Populates the Data Store for Webapp to use to generate User HTML.	Regularly scheduled Python Service.  Runs on two schedules:  (1) Hourly to get recent readings and to generate forecast for the next 24h.  (2) As needed to backfill less recent readings and forecast based on them (in order to enable historic forecast quality audit)	<u>Weather Forecasting Service</u> on the Web (wunderground) to populate Recent Readings (Reading Retriever)  <u>Trained Models</u> to feed those readings to generate Forecast (Predictor)  <u>Config</u> to know how to process Recent Readings and feed them to Models; also to map locations to wunderground station codes.
<b>Data Store</b>	Source of all data displayed by <u>Webapp</u> . Has two data collections: recent weather readings from required locations and predictions based on those readings and <u>Trained Models</u> .	CSV files with designated names and in designated locations inside Container's file system	<u>Data Service</u> populates the Data Store
<b>Trained Models</b>	16 models (4 predicted variables X 4 lookahead intervals) used to generate forecasts	<u>.h5</u> TensorFlow files residing inside Container's File System	<u>Trainer</u> must generate the models offline to be included in deployment.
<b>Config</b>	(1) information on features required for each model (so that <u>Data Service</u> can generate the right Tensors to feed to Models)	Human+Machine Readable file inside Container's File System	<u>Trainer</u> must generate model info for each model trained

	(2) Geographic location mapping for Weather Underground		
	(3) Model Info generated during Training to be displayed by <u>Webapp</u>		
<b>Offline / Engineer-triggered</b>			
<b>Trainer</b>	Trains Models to be used by <u>Data Service</u> for Forecasting	Python/TensorFlow script triggered by Engineer as part of Deployment Pipeline	<u>NOAA Files</u> that contain Training Data for associated geographic locations (must be processed by <u>Preprocessor</u> )  <u>Config</u> containing all models and the definitions of what features and in what order are needed for each model to be trained
<b>Preprocessor</b>	Pre-processes esoterically formatted NOAA files into a form suitable for ML	Python Script triggered by Engineer as part of Deployment Pipeline	<u>NOAA Files</u>
<b>Raw NOAA Weather Files</b>	Manually ordered from NOAA CSV files containing weather records from a number of geographic locations that are proximate to the Target Location of Chicago	CSV files stored in project directory structure	A person needs to order from the NOAA website manually

## Weather Predictor Architecture







