Deployment of Nowcasting Intelligence System to Media Outlets/channels

[Synrad and Nowcast Model](#_fkj7g84vfmcj)

[Generating data for Synrad and running model](#_7sbc5zsbsecm)

[Generating data for Nowcast and running model](#_klg15hbdk3ib)

[Designing a Nowcasting Intelligence System](#_xil7oilx3h5u)

[Generating the Architecture pipeline using Mingrammer python package - diagram](#_t9npn1u9pbhu)

[Data Source](#_kvpoy3q3xh)

[Data storage for inputs/outputs](#_m4z7uo7yusvv)

[Hosting options](#_3oshrhcvct0a)

[Compute(On premises or rent on a cloud)](#_x371vqp0r7nh)

[Visualization Tools(Like Power BI/Tableau)](#_ix107a4g6xk3)

[OS(Command line Shell) to interface with the models](#_9ek47adqpak9)

[Distribution of Nowcasting Intelligence system](#_v24utk5nse17)

[Distribution using FAST API](#_t30y2khzpe1e)

[Distribution using WebApp (Streamlit)](#_yhkdxxgx1c4z)

[Skeletal User Interface using Figma](#_vyl3hxd2gz7k)

[Distribution using Docker](#_8nv4rj8lt1d8)

[Distribution using Conda](#_4xsmj58v5yaf)

[Distributing Using Pypi](#_yerrewgge48x)

[Conclusion](#_a9utd5el7mpq)

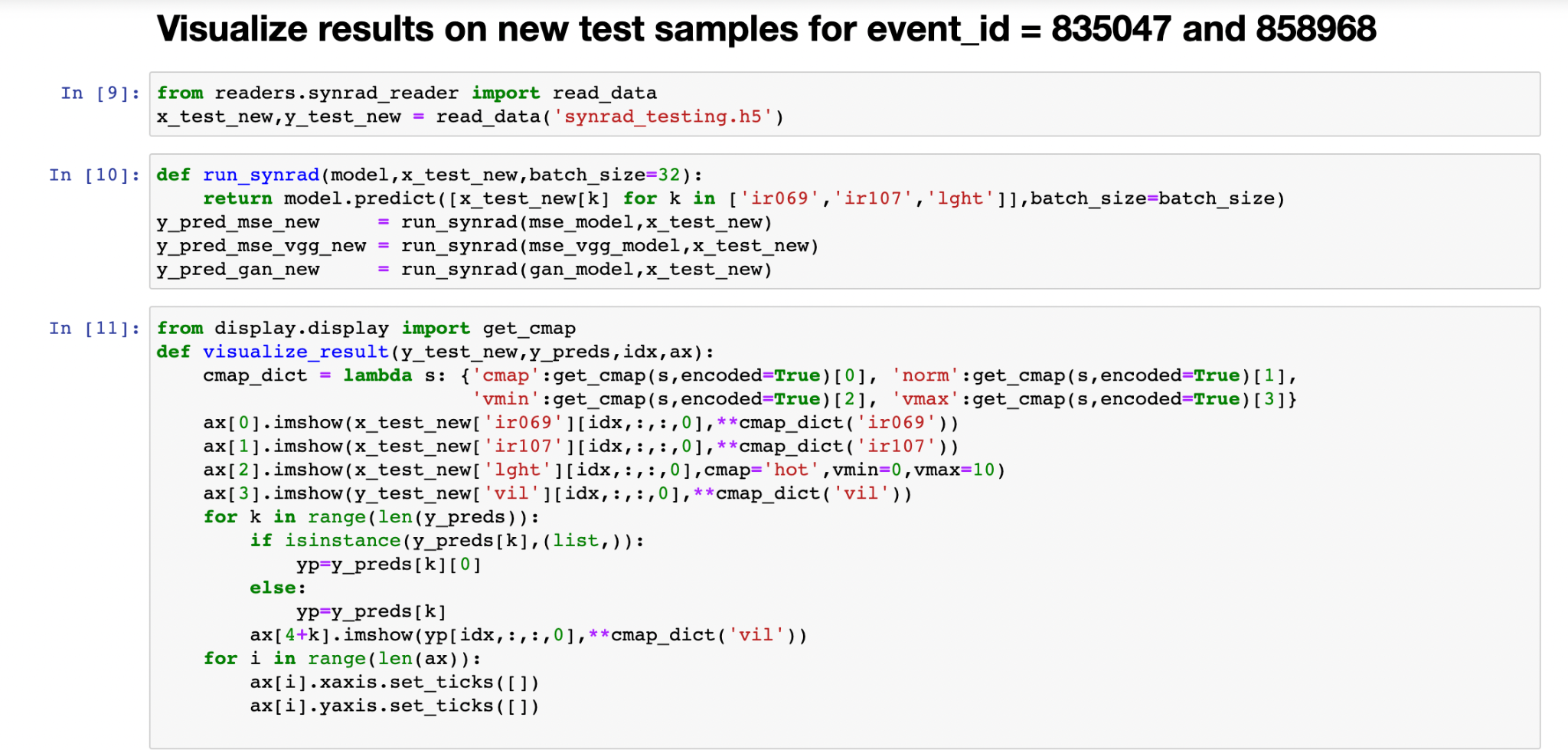
[References](#_bea4876c9xgv)

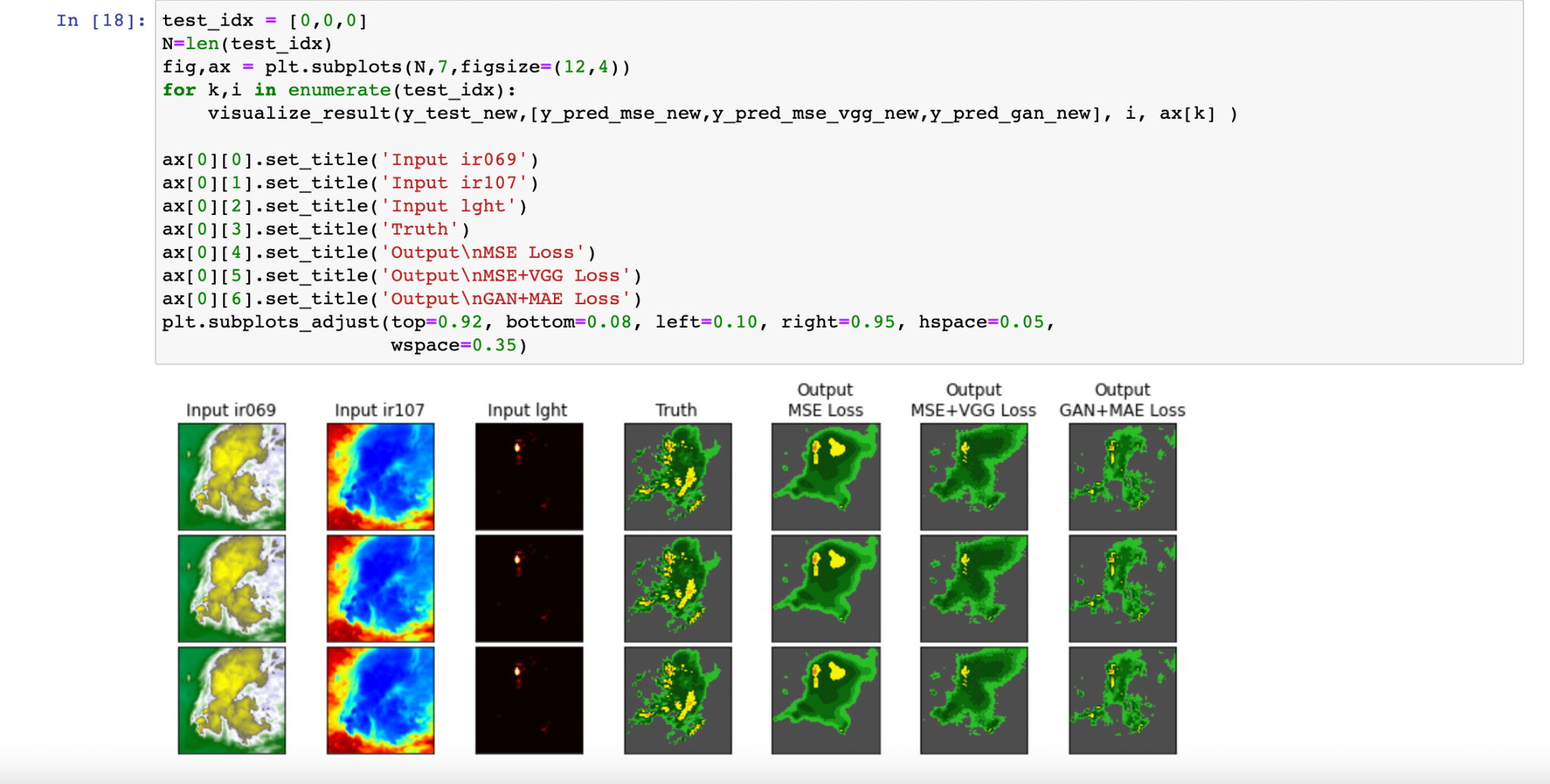
# 

# Synrad and Nowcast Model

## Generating data for Synrad and running model

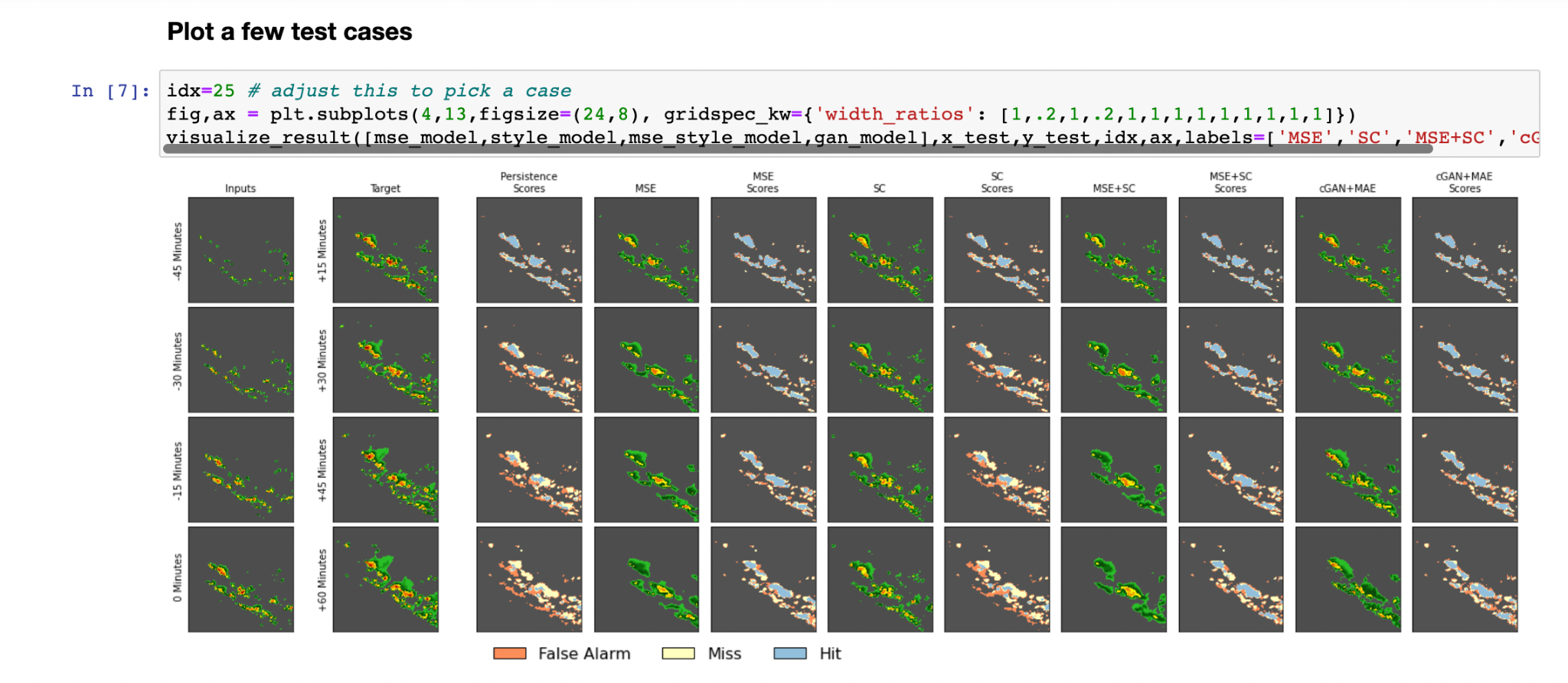
Generated new test dataset for Synrad model for event ids: **858968** and **835047** and generated output for the same.

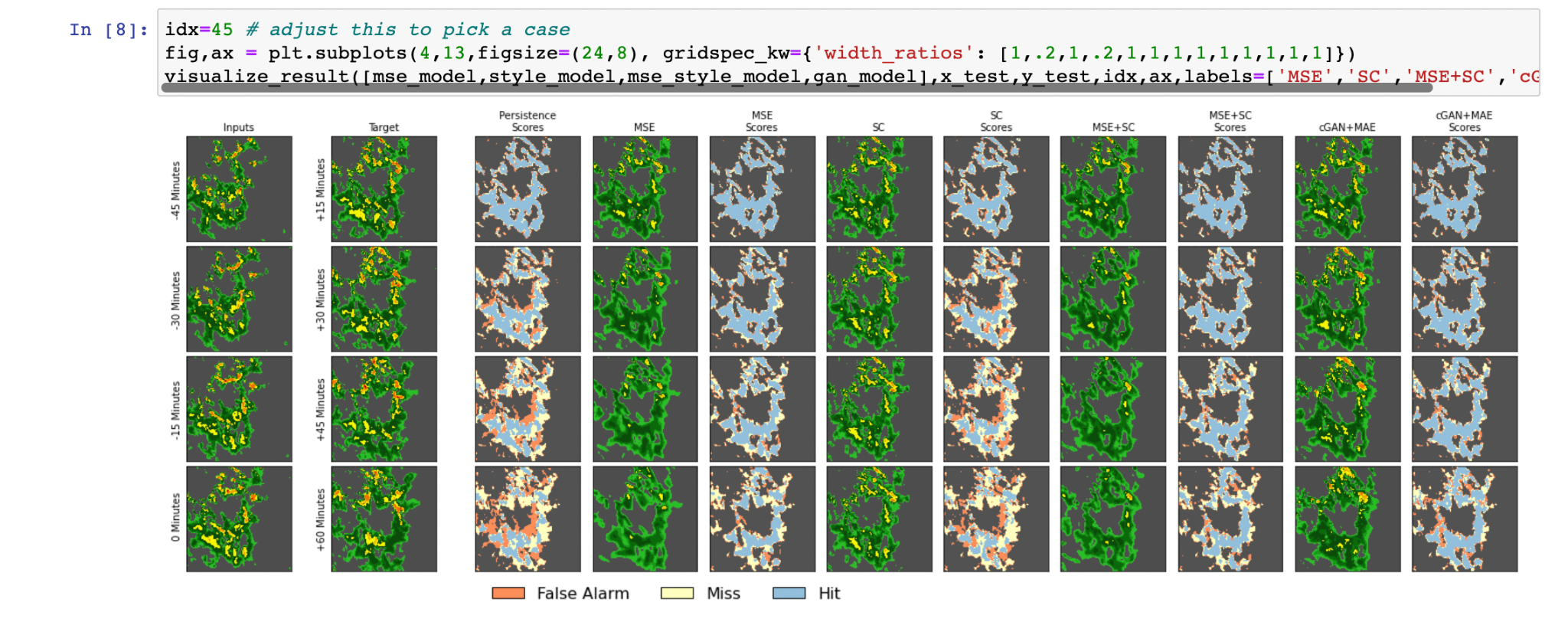




## Generating data for Nowcast and running model

Generated a new test dataset for the Nowcast model and generated output for the same.





## 

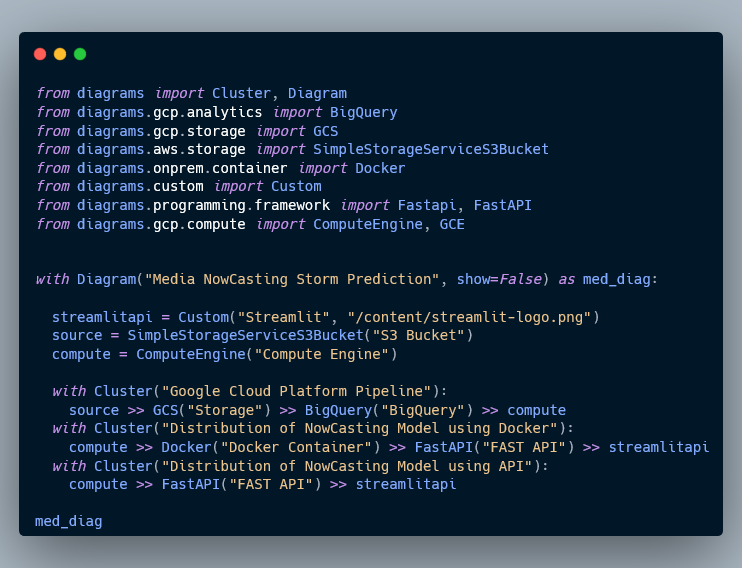
# Designing a Nowcasting Intelligence System

Designing a system that supports millions of users is challenging, and it is a journey that requires continuous refinement and endless improvement. Here we are trying to **build a Nowcasting Intelligence system that supports usage in Media channels and outlets**.

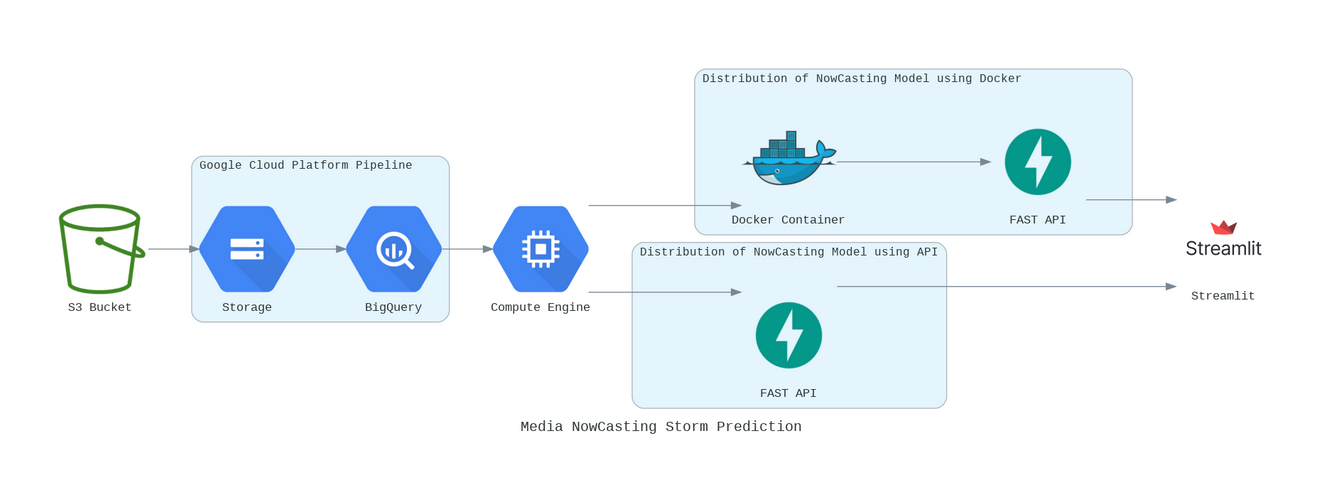
Below image shows the illustration of a data pipeline setup where everything is running on Google cloud platform: storage, database, compute, etc. and distribution of this end product model is done using 2 ways - Using FAST API and Web App(Streamlit), Using FAST API, Docker and Streamlit(Containerized environment)

## Generating the Architecture pipeline using Mingrammer python package - diagram

We generated the architecture pipeline using the python package diagram. Following is the code snippet to generate the same -



Resulting pipeline diagram looks like this -



Big data pipeline architecture using GCP

1. The SEVIR data is read from S3
2. GCP platform used to preprocess/clean and merge data from storm and satellite
3. Compute engine GCP used to generate next hour’s forecast using Nowcasting Model
4. Distribution of the resulting Nowcasting Intelligence Model is done using methods like FAST API/Docker/Streamlit

Let's look at the individual pieces in the above pipeline and why we used them.

### Data Source

* We used data migration tools to migrate data from on-premises or from one cloud to another. Google Cloud offers a storage transfer service like **gsutil** for this purpose
* To ingest data from 3rd party saas services, we used APIs to send the data to the data warehouse. In Google Cloud BigQuery, the serverless data warehouse provides a data transfer service that allows us to bring in data from saas apps such as YouTube, Google Ads, Amazon S3, Teradata, ResShift and more. In our case our source was Amazon S3 Bucket

### Data storage for inputs/outputs

Once processed, we have to store the data into a data lake or data warehouse for either long term archival or for reporting and analysis.

Google Cloud Storage is an object store for images, videos, files and so on which comes in 4 types:

1. Standard Storage: Good for “hot” data that’s accessed frequently, including websites, streaming videos, and mobile apps.
2. Nearline Storage: Low cost. Good for data that can be stored for at least 30 days, including data backup and long-tail multimedia content.
3. Coldline Storage: Very low cost. Good for data that can be stored for at least 90 days, including disaster recovery.
4. Archive Storage: Lowest cost. Good for data that can be stored for at least 365 days, including regulatory archives.

In our case, we go ahead with the Standard Storage option.

BigQuery is a serverless data warehouse that scales seamlessly to petabytes of data without having to manage or maintain any server. Thus, we can store and query data in BigQuery using SQL.

### Hosting options

In the use case of Media Outlets, the trained Nowcasting Model can distributed for use by means of -

1. Exposing the model as an API and the media company(In our case World Weather Forecast Channel) can use Streamlit to create the user interface for the public to see.
2. Standardizing the local environment such that the model runs on all platforms by using Docker containers and thus using the same to expose the model as an API to Streamlit interface.

### Compute(On premises or rent on a cloud)

In our case we choose to rent the compute on the cloud rather than on premises because -

* It is a secure and customizable compute service that lets us create and run virtual machines on Google’s infrastructure.
* It has predefined machine types that can start running quickly with pre-built and ready-to-go configurations, we went with N1 type compute engine for the generation of Nowcasting Prediction images of the next hour.

### Visualization Tools(Like Power BI/Tableau)

There are lots of different tools for data visualization, and most of them have a connector to BigQuery to easily create charts in the tool of your choice.

Google Cloud provides a few tools that you might find helpful to look at.

* Data Studio is free and connects not just to BigQuery but also to many other services for easy data visualization. If you have used Google Drive, sharing charts and dashboards are exactly like that – extremely easy.

Since, the Nowcasting Intelligence Model will be distributed to be used in Media outlets, the visualization of the resulting data will be done using Streamlit library components like - plotly, gee maps etc.,

### OS(Command line Shell) to interface with the models

In case of using Cloud compute on Google Cloud Platform, the compute engine has its own command like Shell that runs on debian linux distribution.

# Distribution of Nowcasting Intelligence system

## Distribution using FAST API

FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.6+

Why use FAST API?   
  
There are other popular API access frameworks for python such as - Django and Flask. We go with FAST API is for the following reasons -

* **Fast**: Very high performance, on par with **NodeJS** and **Go** (thanks to Starlette and Pydantic).
* **Standards-based**: Based on (and fully compatible with) the open standards for APIs: [OpenAPI](https://github.com/OAI/OpenAPI-Specification) (previously known as Swagger) and [JSON Schema](https://json-schema.org/).

The main USP of FAST API is the edge it has over other frameworks due to its adaptation of ***ASGI(Asynchronous Server Gateway Interface)***. ASGI is the successor to the conventional ***WSGI(Web Server Gateway Interface)*** which happens to be slower in execution and access times. Django and Flask use WSGI and hence dont have the high performance that FAST API can achieve.

## 

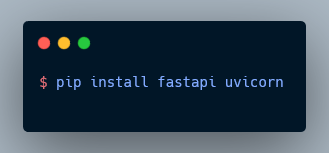
ASGI relies on its Asynchronous nature and handles requests in Concurrency rather than sequentially with help of workers in WSGI. Hence In case of **FAST API** it **can handle** **more number of requests, I/O operations and High Load**.



### Getting Started with FAST API - Installation Steps

In order to use FAST API in our problem, we need to install two package dependencies - **FAST API** and **UVICORN**.

We need UVICORN as that facilitates the ASGI implementation on the server.



### Creating a FAST API server

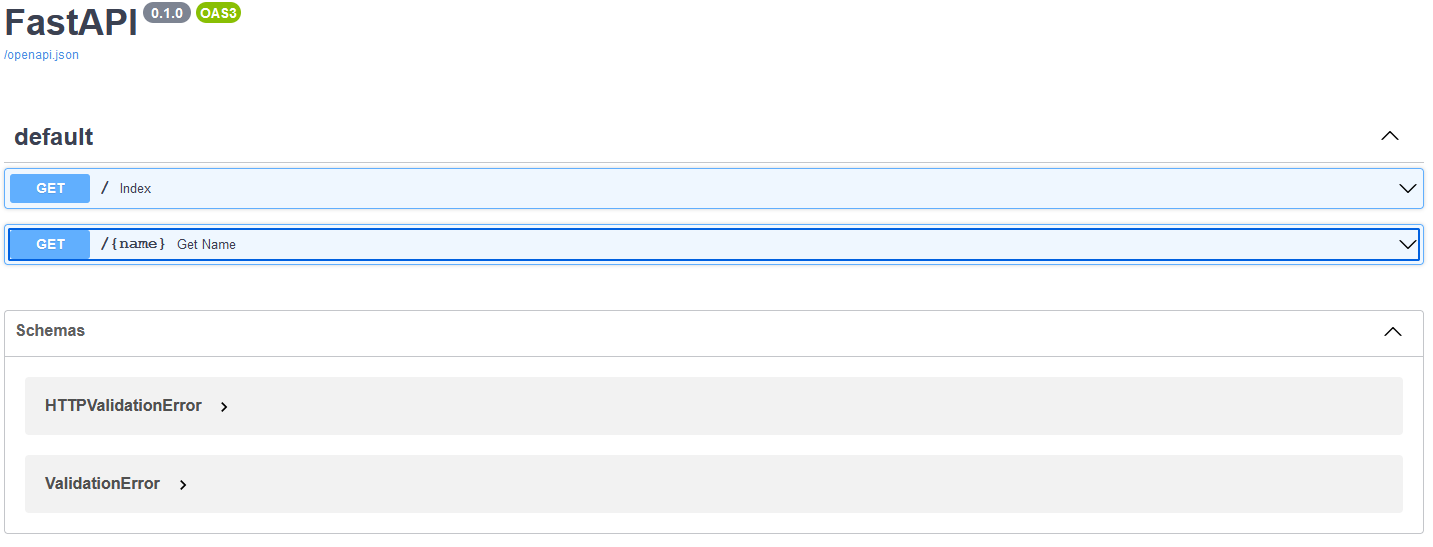
We can create a FAST API server and run it on a local host to test the get and post methods we provide in code.



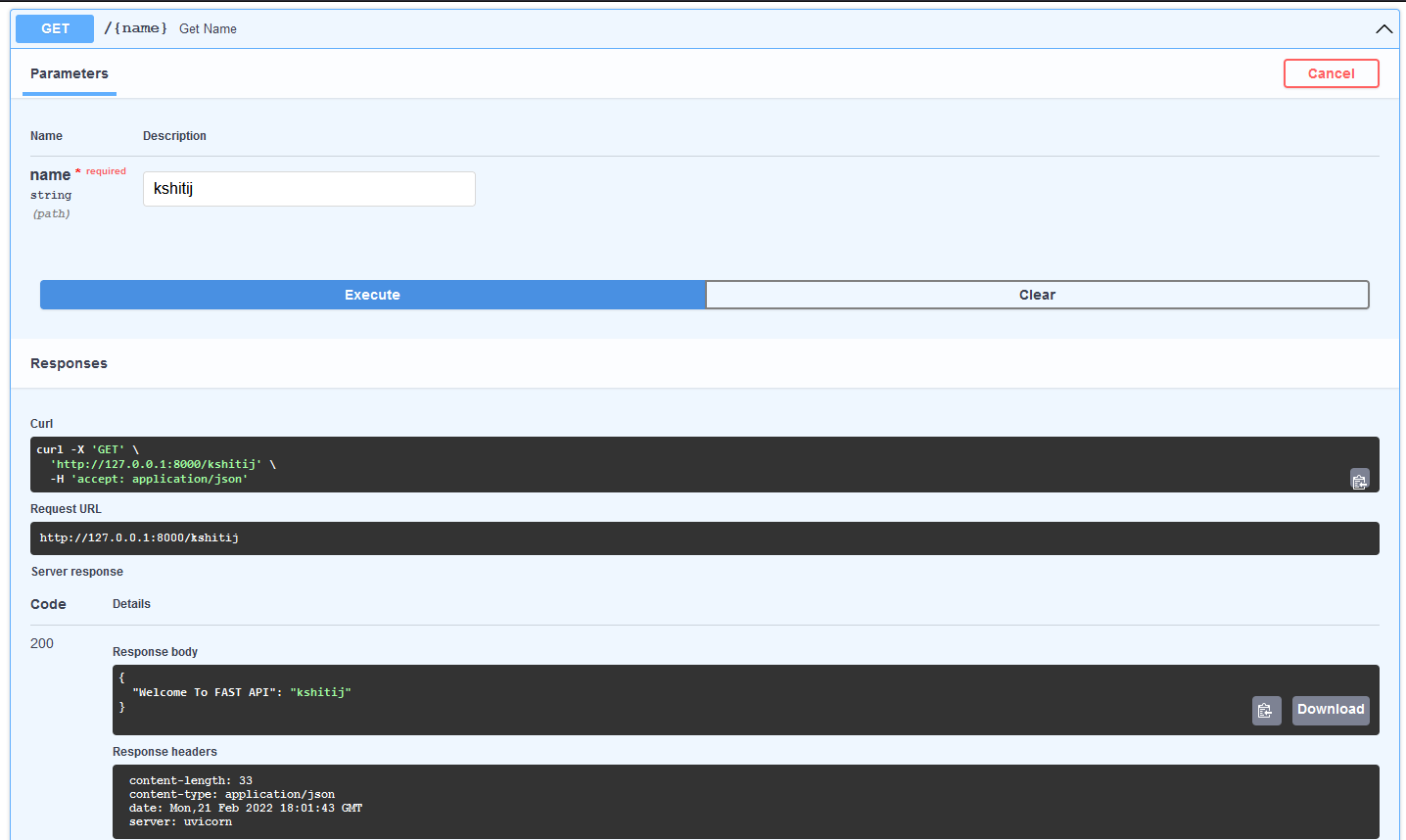
To run the application main.py use the following command -

**uvicorn main:app --reload**

Since FAST API comes with inbuilt OPEN API (previously, Swagger UI) which can be accessed using ‘docs’, by going to http://127.0.0.1:8000/docs we can see the automatic interactive API documentation (provided by [Swagger UI](https://github.com/swagger-api/swagger-ui))



We can submit a get request to check the response -



# 

# Distribution using WebApp (Streamlit)

## What is Streamlit?



Streamlit is an open-source python framework for building web apps for Machine Learning and Data Science. We can instantly develop web apps and deploy them easily using Streamlit. Streamlit allows you to write an app the same way you write a python code.

## Getting started with Streamlit

### Prerequisites

1. Your favorite IDE or text editor
2. Python 3.7 - Python 3.9
3. PIP

### 

### Installing Streamlit

1. Make sure you have python installed in your system
2. Use the following command to install streamlit



### Test whether the Streamlit install is successful:

Type ‘Streamlit hello’ in the command line.



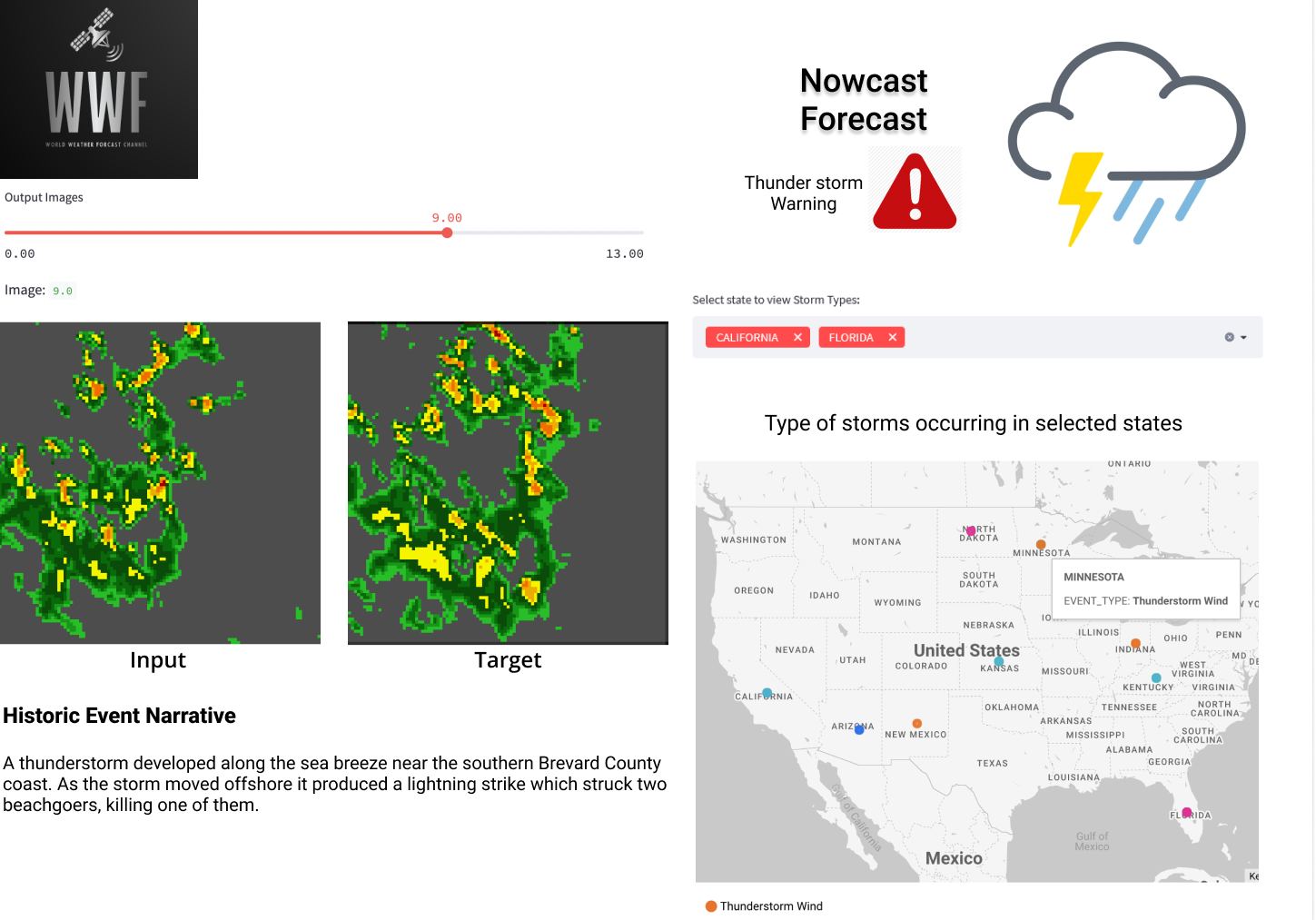
### Running a streamlit app

First, you create a python script with streamlit commands and execute the script using the following command, after running this a sample app will open in a new tab in your default browser.



# 

# Skeletal User Interface using Figma



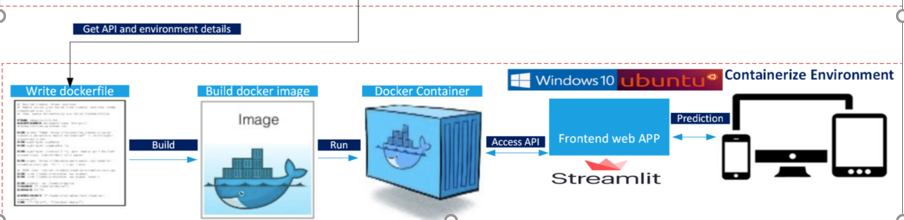
# Distribution using Docker



Docker is an open source project that allows you to package up the application and deliver them to cloud or on-premises without any dependencies.All major providers, AWS,GCP have embraced docker's availability.

## Advantages

1. One of the key benefits of Docker is the way it simplifies matters, users can take their own configuration, put it into code, and deploy it without any problems.
2. As Docker can be used in a wide variety of environments, the requirements of the infrastructure are no longer linked with the environment of the application.One of the greatest benefits is portability.



**Installation**

**Installing Docker based on Ubuntu:**

If docker is not installed in the machine run the below command to set up docker within the ubuntu machine.

* sudo apt-get install docker.io

Once the docker is installed, read, write and execute (daemon) access can be provided using the following command:

* sudo chmod 777 /var/run/docker.sock

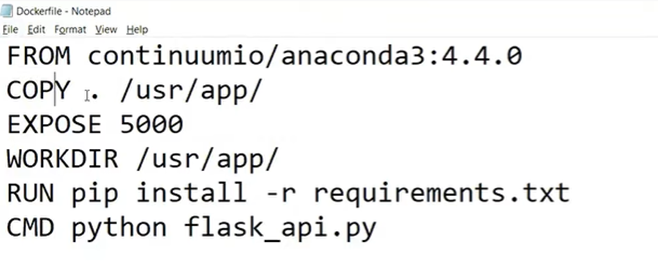
Now we are ready to run the application  
After installing the software one needs to create a Docker Hub account in [**https://hub.docker.com**](https://hub.docker.com)**.**

Using command prompt we can now login by providing login credentials using command line

* docker login

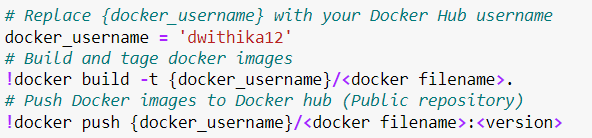
Building a Docker container.

1. Write a Docker file

  
All the required files need to be saved in the same location as the docker file.

2. Building the docker image.

Either Open docker quickstart terminal, and be in the working directory or run the below command in jupyter notebook.



After running the command, the docker image is pushed to docker hub.



Here in the command it will run the docker container.

# Distribution using Conda

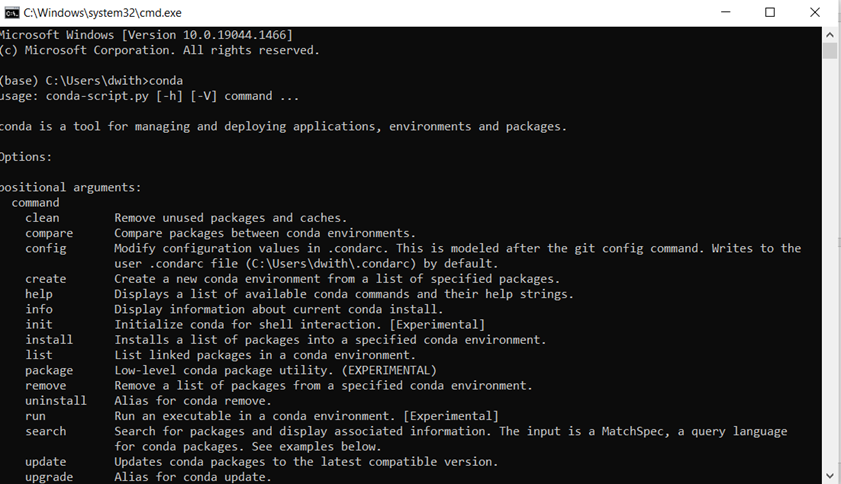
Conda is an open source cross-platform package manager which means it combines the two into a single package system. It was created for python packages but can package and manage any software written in any language. While python installations come with many popular packages installed, one may come across situations where an additional package is required which is not installed. If the required package is available in anaconda.org , it can be easily installed using conda package manager.

**Installation:**

Anaconda.org is a package management service that makes it easy to find, access, and store notebooks and environments, as well as conda and PyPI packages.

Install Anaconda navigator from **https://docs.anaconda.com/anaconda/install/**

After downloading the software, CMD can be used to check if the conda has been installed.



To enable building conda packages ,install and update conda Building a conda package requires a receipe.It is flat directory that contains 3 files namely meta.yaml,build.sh,bld.bd.

**meta.yaml**: It contains all the metadata  
**build.sh**: The script that installs the files for the package on macOS and Linux  
**bld.bd**: The script that installs the files for the package on Windows.

We save all the 3 files in the same directory.  
In the text editor create a new file named bld.bat.



Similarly for bld.bat.



After creating all build files ready, we can start creating a new package with conda-build and install the package on the local machine.

1. Run conda-build

After it is finished it will display the package name and file location.



1. Install your newly built program on your local machine by using the use-local flag:



**Converting packages to use on all platforms.**

Now that the package is build, you can convert it for use on other platforms by using **build.sh** and **bld.bd**

Use command conda.convert with the platform specifier from the following list:

* osx-64
* linux-32
* linux-64
* win-32
* win-64
* all



After converting the file, you may choose to upload the package to anaconda platform.

# Distributing Using Pypi

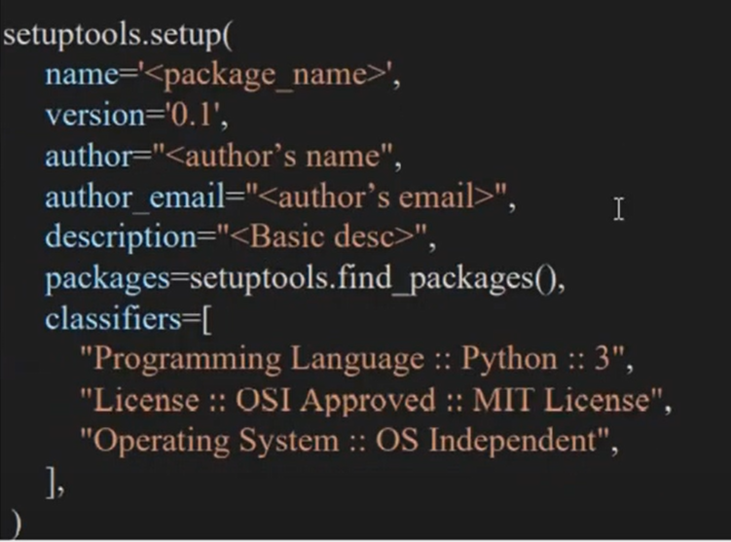
PyPI stands for Python Package Index and is a central repository for maintaining Python packages that have been developed and shared by the Python community. It is a default software repository for Python developers to store created Python programming language software developers and programmers alike use to publicize and share their software, it simplifies the Python packaging process for Python programs.

Make sure you have Python and pip installed on your system. To check the installations:

1. python -V # for python version
2. python -m pip –version

Firstly you will need to create a file name **init***.py* and write a code *and check if it works*Create a file, save it as ***License.txt***, and add the open-source license to our package.

Creating a **setup.py** file is an important part of building our package, this file will hold all the important information the compiler will need to set up our package.



Before uploading the package in the python community, we’ll set up all the needed files, and structure the codebase by installing the below required packages:

**Setuptools** : Setuptools is a package development process library designed to create and facilitate packaging Python projects.

**Wheel** : It is a simple package which provides a bdist\_wheel command for setuptools. It creates a .whl file which is directly installable through the pip install command.

**Twine** : Twine provides us a secured and authenticated way to upload our package PyPi community.

**Tqdm**: It is a package that works internally with twine.

Commands to install the above packages







After installing packages, The next thing you need to do is navigate to the directory of your project where the setup.py file is and then run the below code.



Once the code runs it will create a folder **dist**(it holds the installation files that we’ll deploy to PyPI) and within the folder it will create a .whl file

Another important step is to sign up for an account on [PyPi](https://pypi.org/account/register/) because that is where we will be hosting our package.



Those are the steps required in building and deploying our package, you can run pip install to install it on your computer and try out the package

# Conclusion

1. Generated test dataset for Synrad and Nowcast using generators, observed the result for test dataset for Synrad and Nowcast model.
2. For distributing Nowcast model for Media use case we used following ways:

* API: Fast API
* Web App: Streamlit
* Package: Conda, PyPI
* Docker Containers

1. Designed Skeletal User Interface using Figma.

# References

1. <https://proceedings.neurips.cc/paper/2020/file/fa78a16157fed00d7a80515818432169-Paper.pdf>
2. <https://nbviewer.org/github/MIT-AI-Accelerator/eie-sevir/blob/master/examples/SEVIR_Tutorial.ipynb>
3. <https://cloud.google.com/bigquery/docs/visualize-data-studio>
4. <https://registry.opendata.aws/sevir/>
5. <https://www1.ncdc.noaa.gov/pub/data/swdi/stormevents/csvfiles/>
6. Architecture diagrams - [https://colab.research.google.com/drive/10Nq0KmjbgvAzcaLiqQPwMeDVdkHBCy6y (Links to an external site.)](https://colab.research.google.com/drive/10Nq0KmjbgvAzcaLiqQPwMeDVdkHBCy6y)
7. <https://diagrams.mingrammer.com>
8. [https://systeminterview.com (Links to an external site.)](https://systeminterview.com/)
9. [https://app.diagrams.net (Links to an external site.)](https://app.diagrams.net/)
10. Wireframes - [www.figma.com](http://www.figma.com)