# How late will it land?

*Inspecting the sky, coupling it with your favourite search engine and any other sources available to compute travel delays.*

The current client, [XTravel](https://www.xomnia.com/), is focused on people with severe cases of *allegrophobia.*[[1]](#footnote-0) It has come up with a new concept for european travelers buying flights directly at the counter: *ThereOnTime*TM. This company is offering to let fliers decide on a last-minute, depart-immediately vacation destination that can be safely reached with the least delay.

In order to provide such a service, XTravel requires knowing instantly which flights are delayed; together with categorising and ordering the european airports with the least amount of delayed flights.

As a newly-found startup, XTravel does not [have the resources to buy and] own any dataset; instead, it relies on benevolence and open source data.[[2]](#footnote-1) Luckily, it seems the [OpenSky Network](https://opensky-network.org/), which gathers the output of thousands of sensors owned by aviation enthusiasts, can help us have an idea of the state of the sky and flights therein; together with other websites such as [FlightRadar24](https://www.flightradar24.com/), [FlightAware](https://flightaware.com/) or even [Google](https://www.youtube.com/watch?v=dQw4w9WgXcQ), a rather helpful dataset can be constructed.

In the first proof of concept, XTravel would like to experiment with the [live status of the sky](https://opensky-network.org/apidoc/): which european flights departing from Amsterdam, Frankfurt and Madrid are expected to be delayed. Based on the results of this analysis, ordering the airports and/or countries and/or cardinal destinations depending on a set score would make the product viable.

In parallel, XTravel would like to see if some kind of historical behaviour can be extracted from accumulated data. An engineering team has been brought in to work on the same project, and a solution needs to be devised together.

*The goal of this project is three-folds:*

* *experiment with requesting (in this era of microservice and API), including some online content scraping;*
* *organise [yet] a[nother] dataset, flexing those* Pandas *and/or* Spark *skills;*
* *work together, writing documentation (requirements and/or usage) for the other team.*

*You ate your way through and/or modelled enough [close to] clean datasets during your studies; time to build your own. Let’s see what type of use cases, infrastructure and/or visualization you can come up with!*

For data engineers

|  |  |
| --- | --- |
| ⭐ | **Build our own**   1. Write a tool to download data from the [OpenSky API](https://opensky-network.org/apidoc/) (including failsafes and retries; you might want to look into [asyncio](https://realpython.com/async-io-python/) for instance). 2. Organise an architecture -containerised- to continuously download and enrich the dataset (try [that](https://github.com/Bash-/container-templates), you know who to ask for documentation). 3. Figure out origin and destination from external sources (enrich first dataset; maybe [BeautifulSoup](https://www.crummy.com/software/BeautifulSoup/bs4/doc/) could help). And what about the weather at the landing spot? Any known delay for planes going to this destination? 4. Provide an API to look up and serve the dataset (goal being [this shape](https://www.kaggle.com/usdot/flight-delays)). 5. Deploy on at least two Cloud environments and/or a Raspberry Pi.[[3]](#footnote-2) |
| ⭐⭐ | **Timeseries**   1. Split data in single-trip timeseries. 2. Organise a blob/NoSQL architecture to structure and store this data. 3. Modify the API to look up and serve the dataset. |
| ⭐⭐⭐ | **Event-based**  Organise a pub/sub architecture around the ETL pipeline. |
| ☆ | **Your own use case?** |

## 

## For data scientists

|  |  |
| --- | --- |
| ⭐⭐ | **Build our own**   1. Write a tool to download data from the [OpenSky API](https://opensky-network.org/apidoc/). 2. Figure out origin and destination from external sources (enrich first dataset; maybe [BeautifulSoup](https://www.crummy.com/software/BeautifulSoup/bs4/doc/) could help). 3. Compute arrival time at destination. 4. Filter data to the requested departure locations. 5. Organise and score. 6. Prepare for deployment. |
| ⭐ | **Train a model to predict delays**   1. Err… we will never have enough statistics on time, will we? Let’s use [this](https://www.kaggle.com/usdot/flight-delays) (*i.e.*, the end goal of the engineering project, but for european flights). 2. Model the data according to origin/destination/airline/time of the day/week/year… *sky's the limit*.[[4]](#footnote-3) Let’s make it a team effort, split the work! 3. Train a model to predict delays based on the modelled data (one airline/one airport, one airline/all airports, *etc.*). 4. Prepare for deployment. |
| ⭐⭐⭐ | **Predict destinations**   1. Organise historical data in single-trip timeseries. 2. Use [Dynamic](https://en.wikipedia.org/wiki/Dynamic_time_warping) [Time](https://towardsdatascience.com/dynamic-time-warping-3933f25fcdd) [Warping](https://github.com/slaypni/fastdtw)[[5]](#footnote-4) to find the canonical (averaged) routes. 3. Compare new timeseries to canonical timeseries using DTW. 4. Provide predictions on destinations. 5. Prepare for deployment. |
| ☆ | **Your own use case?** |

1. You Google it. [↑](#footnote-ref-0)
2. And smart consultants. [↑](#footnote-ref-1)
3. For the IoT feeling. [↑](#footnote-ref-2)
4. See what I did there? 😄 [↑](#footnote-ref-3)
5. Three different links, check them out! [↑](#footnote-ref-4)