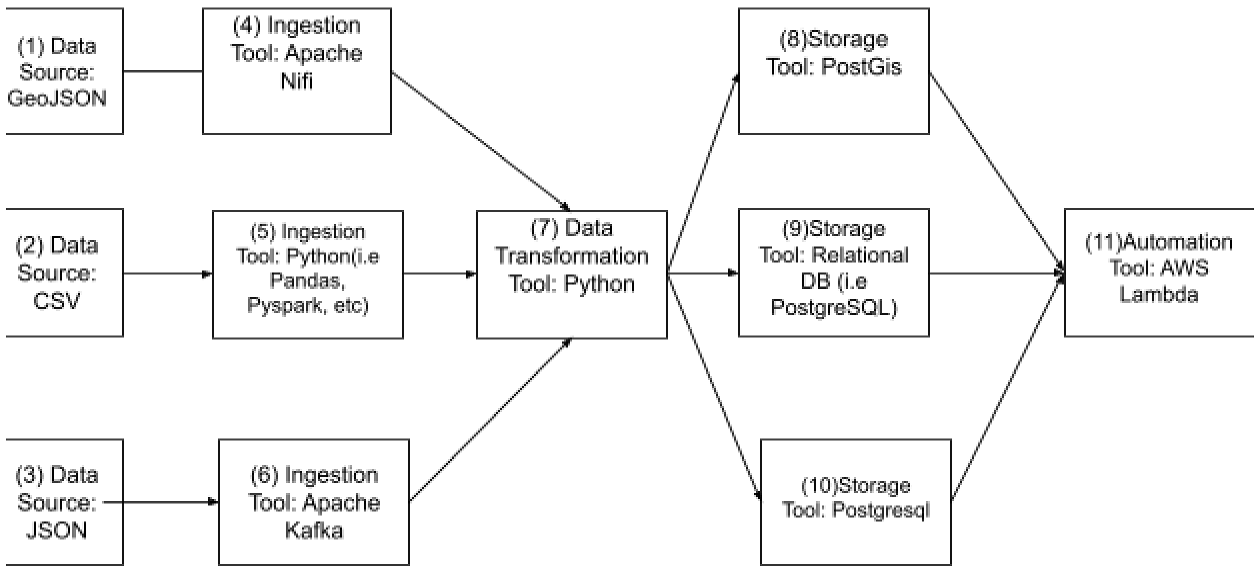
System Design Explained

(1) GeoJSON source data
(2) CSV source data
(3) JSON data
(4) I chose Apache Nifi because it is designed to handle real time data ingestion and updates. You can set it up to trigger the ingestion process by having it monitor new updates and or by running it automatically according to some set schedule (i.e day, hourly, etc). This specific tool has built in features designed to help validate, parse and deal with geospatial data.
(5) and (7) I would use Python for ingestion because it allows access to libraries like Pandas as Pyspark to ingest csv data and easily manipulate and transform data in general.
(6) I would use Apache Kafka because it suits well for high throughput data and real-time ingestion and processing.
(8) PostGIS is specifically designed to store and query geospatial data via different geospatial metrics. The advantage of using this is that it is integrated with Postgresql and can be used with non geospatial data. On a high level, the data is structured and so we need a relational DB. Within a relational DB, I chose PostGIS/Postgresql for the reasons above.
(9) Because the CSV data is structured, I would use a relational database. For consistency, I would also use Postgresql for this as well.
(10) I would use Postgresql here as well because the JSON data will undergo transformation and become structured data. Once it is in the appropriate format, it will then be fed into the Postgresql database, also for consistency.
(11) I would use AWS Lambda because it can trigger automations based on events. It doesn’t require infrastructure maintenance and is sufficient for the job while also minimizing cost.


(1) Data Source: GeoJSON
(4) Ingestion Tool: Apache Nifi
(8)Storage Tool: PostGis
(2) Data Source: CSV
(5) Ingestion Tool: Python(i.e Pandas, Pyspark, etc)
(7) Data Transformation Tool: Python
(9)Storage Tool: Relational DB (i.e PostgreSQL)
(11)Automation Tool: AWS Lambda
(3) Data Source: JSON
(6) Ingestion Tool: Apache Kafka
(10)Storage Tool: Postgresql