Tasman Senior Data Engineer - Assessment Task

Many thanks for taking the time to complete this assessment for the position of Senior Data Engineer at Tasman!

The assessment sets out a series of tasks that progressively increase in complexity. We don't necessarily expect you to finish them all. We encourage you to send the code our way after you have spent the time you are comfortable with.
We don't expect everyone to finish all tasks - if pushed for time, please describe how you would have completed the remaining tasks. However, we do expect that the sections you have completed are production-ready and are representative of the standard of work you are capable of.

The assessment is an end-to-end task which includes:
- Connecting to a novel data source,
- Making sense of the data,
- Containerising it for ease of use
- Infrastructure as Code (IaC)

This is highly representative of the work you will be asked to do at Tasman. We care a lot about good understanding of the requirements, collaboration and early feedback - so if you get stuck or have any questions, please do not hesitate to reach out.

> [!NOTE]
> We outline what we expect you to do, but feel free to use your best judgement and do what you think makes most sense - for example, swapping Postgres for a different database engine. Remember to document these decisions, as we will ask you about them.

Data dependent

Dataset

Size ?
Structure ?
Schema ?
Validation

- The USA Jobs reporting database (https://www.usajobs.gov/) is a very extensive repository of all job openings in the US government.
- It has a big, well-documented REST API documented [here](https://developer.usajobs.gov/api-reference/) → format ? Constraints ? limits ?
- The API can be accessed through the [root URL](https://data.usajobs.gov/api/). We will share an API key via email.

security for credentials → .env ? secret manager / parameter store ?

Assessment

1. Write a simple ETL script that performs a search of the USAJOBS API, with the criteria detailed in the `Search Criteria` subsection.
2. The script should load the data _durably_ into a Postgres database container.
   break down for task requirements
3. Containerise the extraction service, so it can be run daily to populate the Postgres database.
   orchestration
   once daily or more ?
   • API constraints
   • Data update frequency
   • Data demand at DB level (hypothetical)
4. Write the IaC configuration that would pull this Docker image from a public (or private) container repository and deploy the extract service in a cloud provider of your choice. Trigger this service on a schedule, preferably also defined in IaC.
   - Build and deploy image to a Docker repository, if you choose to

Docker ?
Terraform ?

Pull image for a single run extraction run ?
Pull image to deploy live service that orchestrator runs ?

simple scheduler ?
orchestration framework ?

Prototype + test DB locally, deploy in cloud

use a private cloud hosted container registry, make this a part of your IaC config. Otherwise, a public Docker Hub repository is fine.
  - When deployed in the cloud, you don't want to use a local Postgres container. Either work under the assumption that the extract service is accessing a cloud hosted Postgres instance, or also deploy the cloud hosted Postgres instance.

Cloud DB instance stays live

Normalisation ? Fuzzy matching ?

DB and host are data dependent

Search Criteria
- Perform a search of jobs that have `data engineering` as a *keyword*. Does API have search functionality ? OR load data in intermediate location(s) for search ?
- Parse a subset of fields from the response that a job seeker based in Chicago, would find Consider full schema relation to Chicago useful in their own database of job postings. At the very least, retrieve the following fields: `PositionTitle`, `PositionURI`, `PositionLocation`, `PositionRemuneration`. Focus on retrieving key fields, rather than optimising the API request search parameters.

2 different tables ? OR Views from the same table ?

compare methods later based on task and consider scaled implementation (s)

data lake ?

medallion architecture from bronze to gold standard for DB loading ?

S3

What we are looking for
- Quality, readability, and clarity of intent. All best code and doc practices
- Informative README document that provides context and instructions on usage and deployment.
- A sensibly structured codebase, using modern tooling.
- A clear database design of table(s) and (if necessary) data models.
- In the ETL script: separation of concerns, extensibility, robustness. Good handling of the API responses. Appropriate handling of errors, and respecting the API rate limits. SOLID OOP + DRY
- Prioritization. Depending on the time you are able to dedicate to the task, we'd prefer seeing the task finished end-to-end with callouts to potential improvements to each component, rather than a perfectly written Python script and nothing else.
- Finally, we have to be able to run and reproduce what you build. Make sure we can run the service locally, with minimal additional work, and starting from scratch (assuming we have Docker and Python installed). Seamless integration and deployment is KEY

sufficient exception handling

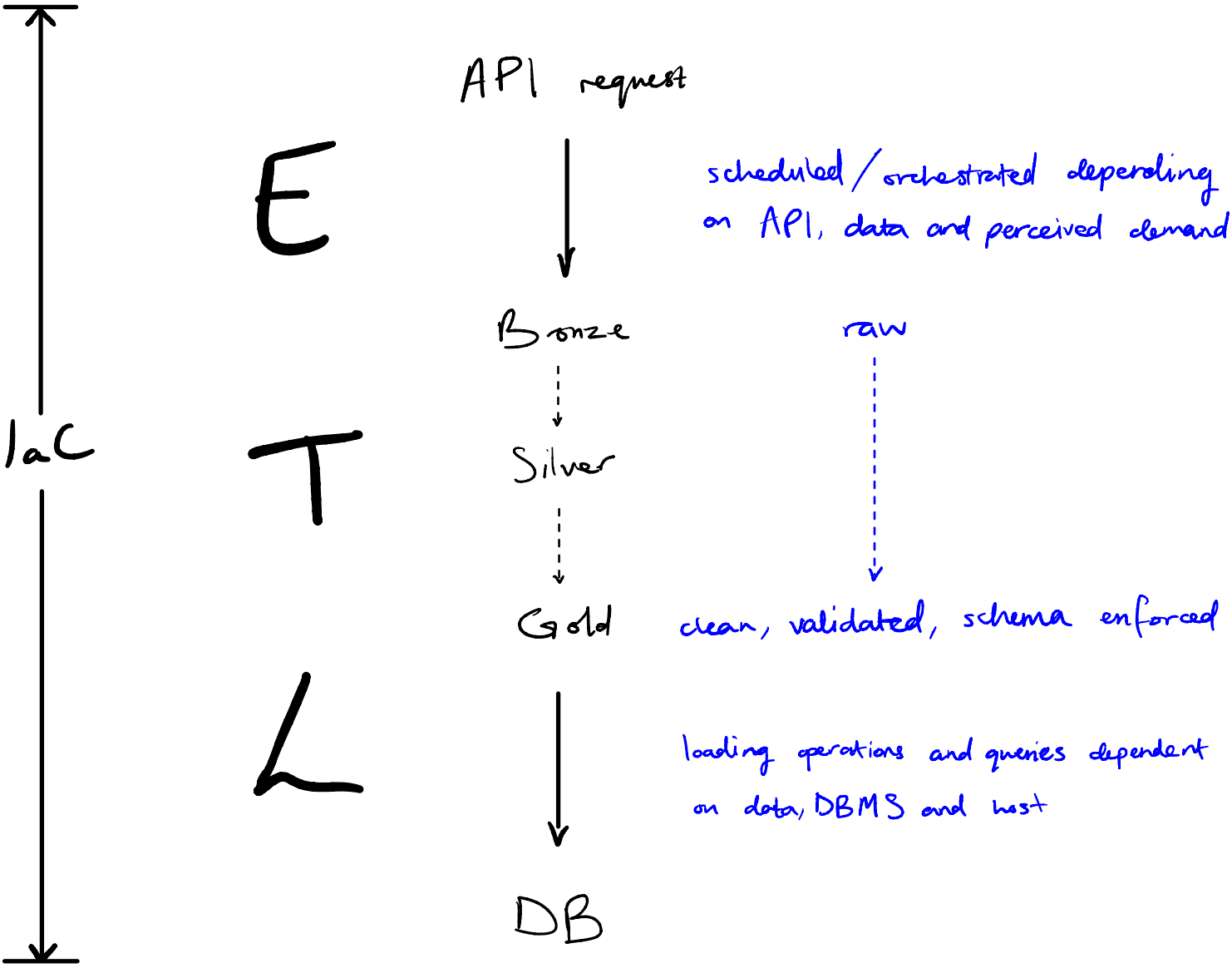
Docker + Python min. requirement + AWS creds additional = Automated setup, provision, deploy (.env file entry) ------> (with minimal additional work ?)

Further comments
- Feel free to use any tools, packages, utilities or services you want to accomplish this.
  - This does include use of LLM's. We expect our colleagues to use it in their work, so we believe it's fair to allow it in the interview phase too.
  - However, *do not* simply give this assessment to Claude Code (or similar) and let it do the task instead of you. We expect you to be driving the whole time - _all_ the decisions need to be yours, and we will ask you to explain them during the followup interview. We will also ask about how you used it, what the prompts were, as well as how you use it in your day-to-day work. Detailed design docs ----> keep dev diary
- Considering you might be working with a Free Tier of a Cloud provider, please, make sure to monitor and restrict your usage. We

Maximum cost optimisation is a KEY requirement
↳ keep it FREE and give alternatives for if there was a budget

don't want you to incur any unexpected costs.
- Deliver your codebase and scripts in a Git repository, with a view to presenting a working version of it
  in the Technical Interview.
  - If working with GitHub or GitLab, you can invite the following usernames to your repository.
    - `sprckt`
    - `sakce`
    - `jurrigerretsen`
    - `marcellovictorino`

Solution Flow



NOTES:

- Extract service is now extract + transform service
  - Potentially loading too depending on data, DBMS and host