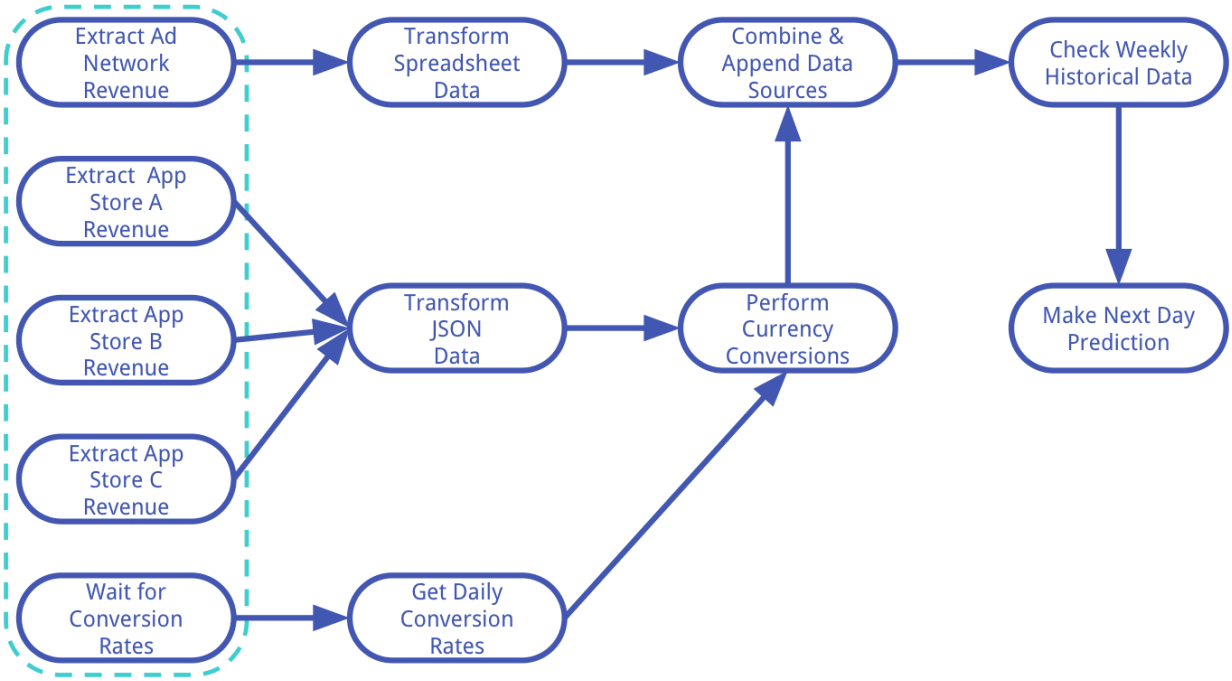
Workflow Management Service - Candidate Brief

Introduction

In this assignment you will be prototyping an architecture for the workflow management service. Service should be able to execute an unlimited amount of pipelines with an unlimited amount of tasks in a sequential or parallel order. For the simplification of the task let's assume that pipelines are absolutely independent from each other.



How to approach this

In general, approach this the way you would if this was a project you undertook as part of your regular job as a software architect.

It is up to you whether you implement a prototype and actual coding, or just describe the architecture in diagrams, images or just plain text. Imaginge, that the team of developers will

need to produce a production-ready product based on the artifacts and documentation you provide as a software architect.

Where the specification is unclear or falls short, you should make reasonable assumptions and design choices. When doing this, is it important to thoroughly document your assumptions and design in the README or other relevant documentation artifacts you choose to produce (e.g. code comments or user manual).

To help you make these decisions, keep in mind that we ask candidates to complete these product design exercises in order to:

1. See how they would implement a software design and prototype the architecture based on a problem statement which reflects some aspects of the problems we solve on a daily basis.
2. Gauge their technical strengths, which we can use for follow-up conversations.
3. See how they can describe the solution and explain to other developers that will actually develop the software.

Where a candidate already has a relevant public code portfolio or architectural publications, we often skip this step. We are, therefore, less interested in seeing how well you can stick to every single detail of a detailed specification (although that definitely helps, especially if the specification is clear and unambiguous) as much as we are interested in seeing what you are capable of.

Evaluation

We will evaluate proposed solution based on the following criteria:

- How easy it is to understand (good structure, presentation, English).
- Elegance of the solution (patterns, techniques)
- Complexity of implementation (it should not take several months/years to implement)
- Efficiency in terms of the resource allocation
- Efficiency in terms of execution time

Requirements

Name	**Description**
Smart Scheduling	It should be possible to schedule the run time of each pipeline separately. Scheduling should be possible once a day, once a month, or even several times a day.
Dependency Management	A simple, concise interface for defining

	dependencies amongst tasks. Dependency management should handle not only the dependency of task execution, but also failures and retries. A system should be able to take advantage of any lack of dependencies amongst tasks to increase workflow efficiency. For example, in the workflow example introduced in the **Introduction**, the five leftmost tasks (outlined by dotted line) are all independent of one another and can be executed in parallel. Additionally, there can be a single task on which many other child tasks depend on its completion. In this case, it's preferred that the parent task be executed as early as possible.
Resilience	As mentioned above, workflows will always act unexpectedly and tasks will fail. The system should be able to retry failed tasks and offer a concise interface for configuring retry behavior. In addition, system should be able to gracefully handle timeouts and alert the team when failures occur or when tasks are taking longer-than-normal to execute (i.e. when service level agreement — or, SLA — conditions have been violated).
Scalability	A system should not only address current data processing needs, but also scale with the future growth without requiring substantial engineering time and infrastructure changes.
Flexibility	Pipeline should be able to run different types of tasks. Some of them might be CPU-heavy, some of them might just send a request and wait for a response for a certain amount of time.
Programmatic Pipeline Definition	A large number of workflow managers use static configuration files (e.g. XML, YAML) to define workflow orchestration (e.g. Jenkins, Dart, and Fireworks). In general, this approach is not a problem, as the structure of most workflows are fairly static. However, the sources and targets (.e.g. file names, database records) of many workflows are often dynamic. Thus it should be possible to

	programmatically generate workflows on the fly given the current state of the data warehouse.