Resolve.ai

Resolve AI Backend: Frequently Asked Questions

System Architecture & Components

Q1: Do we have to make our own emitters or will that be something the reviewer/tester will have ready?

A1: You should make your own emitters, but they can be very lightweight/simple.

Q2: If the latter, can I know about the format of the data they will be sending?

A2: Not applicable - you will be creating your own emitters.

Q3: If I create my own emitters, can it be any sort of data format? Like string, integers, floats, etc. or is there a specific format the data should be in?

A3: No specific format required. You should create your own contract/data model.

Q4: Similarly, do I have to create my own analyzers or will that be something the reviewers will have ready? Basically, do I only have to worry about the Distributor in the middle or each component end-to-end?

A4: Similarly, you should make your own analyzers, but they do not need to actually do anything (e.g., could just print that it received a log or something).

System Assumptions

Core Assumptions

1. **Security validations** (ex: checksum matching, authentication) is out of scope for this problem
2. **The system is expected to support a large and growing number of log emitters** (potentially tens of thousands)
3. **Log emission volume & rates may vary significantly per emitter.** Some emitters may send steady streams while others may produce bursts
4. **This is not a real-time processing system**, some latency is acceptable. Please correct me if real-time guarantees are needed
5. **For the purpose of this problem, the weights for distribution are static**, not dynamically updated during runtime

Additional Clarifications on Assumptions

- **Assumption 1**: In general it would be good to indicate assumptions either in the README or inline in code where the assumption is applicable.
- **Assumption 2**: The design should account for handling a large number of emitters.
- **Assumption 3**: No need to implement that in the emitter. Handling it downstream makes sense though.
- **Assumption 4**: You can call out latency expectations when crafting your stated assumptions/expectations of the system.

System Requirements & Behavior

Q1: Are all logs equally critical, or can I prioritize high-severity logs? (e.g., is it acceptable to drop or delay debug logs under pressure?)

A1: Up to you. Again would be acceptable to handle all logs equally and include some thoughts around extensibility to a solution that supports criticality-based prioritization.

Q2: Is log sampling permissible, or must we ingest every single message?

A2: Sampling shouldn't be necessary, you can think of backpressure mechanisms if the goal is to alleviate load on the system.

Q3: For the purposes of this problem, can I assume that logs from all services have equal priority? (I understand this may not reflect a real-world production setup.) Or would you like me to handle prioritization across services with differing log importance?

A3: Same answer as above

Q4: What are the latency expectations (e.g., sub-millisecond, second-level, minute-level)? I'm assuming throughput is the priority, but I want to understand any upper bounds. Note: I define latency as the time from when a service emits a log message to when an analyzer completes the processing for that message.

A4: No specific expectations. Feel free to call out latency expectations and how they relate to various stated assumptions.

Q5: Should I treat weight distribution as a best-effort goal, or are there strict requirements or penalties for temporary imbalances during rebalancing?

A5: Best effort is good.

Technical Implementation Details

Q: Is there a max number of messages that can be contained in a packet?

A: No

Q: Are all messages completely independent of each other (means no concept of ordering or one message being related to another)?

A: Yes

Q: Are all messages from one packet required to be sent to the same analyzer (so we forward to packet and not the message itself)?

A: Either way is fine, just call out your assumption in your solution

Q: Do messages have a size limit?

A: No size limit. No official SLA, you can optimize either way and call out the tradeoffs in your solution

Q: Is there any processing SLA or ideal focus on high throughput vs ultra low latency or a compromise best of both type deal?

A: No official SLA, you can optimize either way and call out the tradeoffs in your solution

Q: If I find libraries online that basically solve the project requirement or greatly aid in it, can I utilize those? or is the goal completely in house written code.

A: Should be mostly in house code. For small things libraries like lodash or caching libraries are fine but the core logic should be in house code.

External Dependencies and Libraries

Q: Are we allowed to use external libraries (e.g., circuit breakers) to detect and avoid sending traffic to offline analyzers, or is this functionality expected to be implemented from scratch as part of the assignment?

A: Yes, feel free to use external libraries. You don't need to implement everything from scratch.

Infrastructure and Queuing

Q: To prevent service overload, can we use an in-memory queue, or is a production-ready solution like Kafka or SQS required?

A: No requirement to use a solution like Kafka or SQS. An in-memory queue is perfectly acceptable for this assignment.

Packet Processing Requirements

Q: Is maintaining the order of processed packets a requirement for this assignment?

A: No, you can assume that packet ordering is not required.

Failure Handling and Load Balancing

Q: In case an analyzer goes down, I'm considering rebalancing the weights across the remaining active analyzers and routing future traffic accordingly. Is this the expected approach? Alternatively, I was also thinking of maintaining a dedicated queue per analyzer and allowing requests to wait for their assigned analyzer, which may introduce higher latency. Would this be acceptable?

A: It's up to you how you want to handle analyzer failure, but feel free to expand on how you evaluated the tradeoffs and any assumptions you made in your writeup. Both approaches you mentioned are valid - the key is demonstrating your thought process.

Code Quality and Testing

Q: Is the implementation expected to be production-ready, including unit tests and other quality considerations?

A: You should spend roughly 8-10 hours on the problem, so prioritize tasks according to what you think is most important. You should make sure that you implement the key functionalities, and it is OK to mention next steps for a production-ready distributor in the writeup rather than implementing every production concern within the time constraint.

General Guidelines

- Document your assumptions clearly in your README or inline comments
- Focus on creating a working solution that demonstrates your understanding of distributed systems concepts
- Consider scalability and performance trade-offs in your design decisions
- Include thoughts on how your solution could be extended or improved
- The core logic should be implemented in-house, though utility libraries are acceptable.