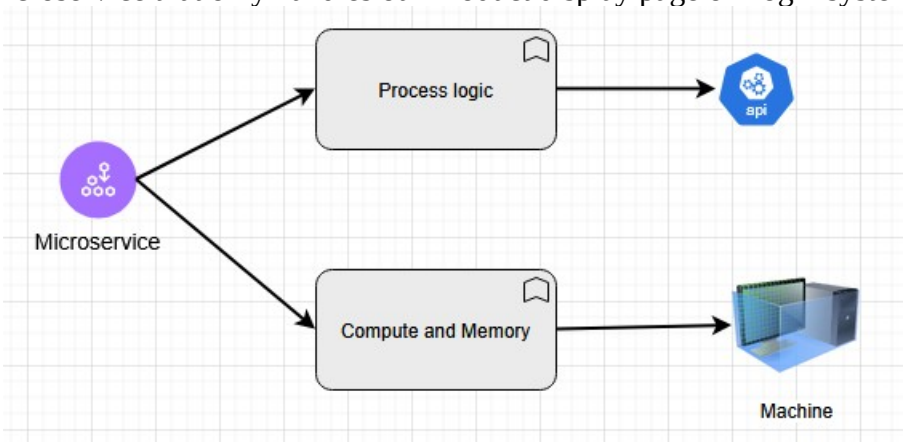
Observability and Monitoring

We got to observe and monitor our system once it is up and running. It is as important as building any system.
How's the system working?

- What are things we monitor?
- Are they any errors?
- Is every component working as planned or not?
- How's the health of each and every component?
- There could be edge case? Is the system keeping track of them in case of edge cases?
- Is any service/component down?
    - if service is down, are we getting alerts?

Observability is required when application moves from development phase to maintenance phase.

In Microservices, what could happen?
Microservice could be providing APIs to other services
We can have a microservice that only handles our Product display page or Login system etc.



Two high-level components of Microservice: 1. Process logic and provide APIs, 2. Compute and utilize memory on the machine. First deal with API, second deal with machine. For providing the APIs, Microservices are using hardware components of the machine.

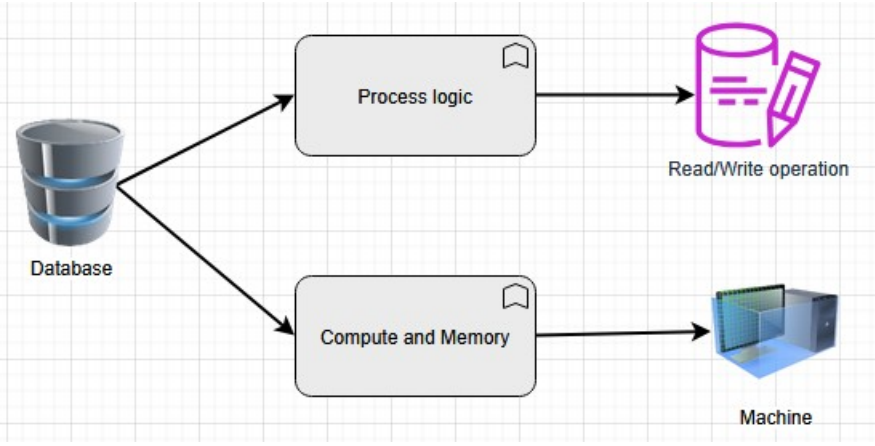
That means, we have two things to monitor: 1. API and 2. Machine (Infra)

<u>1. How do you monitor APIs?</u>
We got to monitor: Throughput, Latency, Error-rate, Health-check
- Throughput:
(our system is handling 10k queries per second, 10k QPS) => Alert if throughput suddenly goes up or down or it presents any anomaly (showing different behavior). We want to see how our system is performing, is it reaching its upper limit? Ok we have reached upper limit of our system, maybe we should think scaling our system? Lets say we are building a system only for Amazon US where we have high traffic from 8Am to 12Am. Sometimes, we get a lot of queries even in the night around 2Am or 3Am, that means we got to monitor what happens even in the night time.
Throughput:
When there is a sudden spike in traffic, the first thought is 'Attacks'. Did we get any attacks? Also some service could be down and the retries could cause sudden spike. With throughput we can monitor latency.

- Latency:
Terminology associated with latency: SLA, SLO, SLI, P99 150ms, P95 130ms, P90 100ms etc. Alert when the latency goes beyond the established thresholds.
- Error-rate:
error codes like 5xx series (something wrong with the server), 4xx series (something wrong with the client), 3xx series (for mostly redirects). We get all these errors in production, we got to see whether we are getting too many errors at any point or not. Put some alerts if error-rate spikes above a set threshold.
- Health-check:
We ping the system to know whether service is up or down. Every component in the service will ping the other component to see whether it is up or down. Say if we have a Loadbalancer, it will keep pinging web-servers to see whether they are up or down or providing the 200 responses or not. 200 for health-check. Alert if any component is failing the health-check.
- Other metrics could be like: there could be some services from which we want more data. We want our system to perform in a certain way, maybe we are logging something specific to that particular request and checking each time whether we are getting correct response. Those additional metrics could also be added into that on top of the 4 metrics mentioned-above.

2. How do we monitor machines?
Monitoring machines
Machine or the hardware part.
- CPU usage => we could create an Alert say when CPU utilization % > 75%. Alerts could be in the form of emails, SMS.
- Memory usage => How much % of memory we use? Memory usage > 90% of disk or ram. At this time also, we should get an alert. If memory usage of a component is too high, it could cause Latency as it cannot work beyond that point.
- Disk I/O => Number of Disk I/O operations per minute. Alert if it exceeds threshold.
- Network I/O => Number of Network I/O operations per minute. Alert if it too high.
Observability: we have to keep observing the patterns our system is trying to tell us.

One more problem is, for each Microservice, we require so many metrics. Imagine there are multiple microservices, components connecting and communicating with each other. That means, we have to measure multiple alerts related to multiple components and multiple alerts related to multiple network requests.
Consider a Database instead of a Service



Database is also processing some logic to perform Read/Write operations
Same thing goes with Compute and memory as well for machines where it is performing operations. Same thing for machine part, we got to monitor, how much CPU is being used? How much memory? Is there any issue with Disk I/O operation? Is there any issue establishing connections between different sections of database?

Imagine, if a Loadbalancer is handling multiple Services S1, S2, S3, S4 and say S1 and S2 have their own databases, S3 has a message queue to send some notifications etc. In this case, imagine how much work is required. Analyze the metrics and observe the working of different units. Accordingly we will conclude our system is working fine or not.