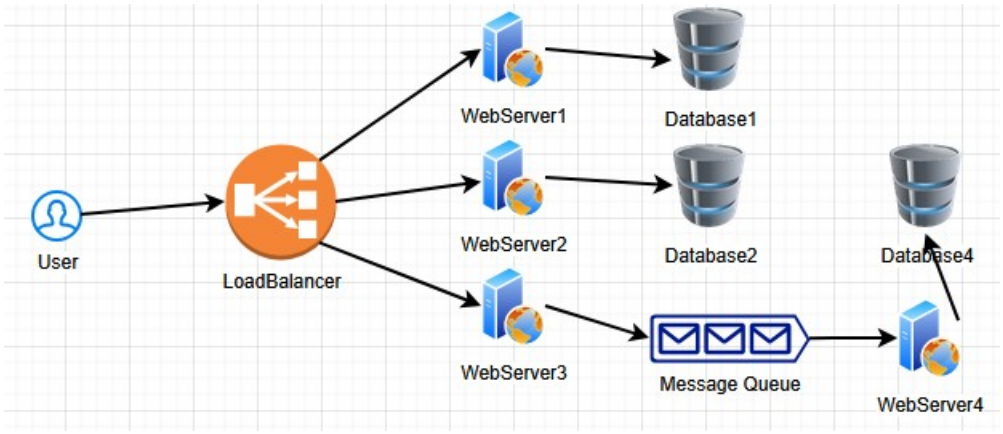
Observability2:

If you want to monitor a Microservice, we got to take care of 1.API and 2.Machine (Infra)

We have many solutions / tools / software for Observability
- Prometheus
- Grafana
- NewRelic infrastructure
- CloudNative => AWS Cloudwatch, GCP monitoring, Azure monitoring
- DataLog

1. Unit testing
2. Load Testing (We want to test multiple APIs using third-party tools like JMeter. In JMeter, we can create different routes of user)
3. Integration testing
4. Manually test our system
5. Chaos testing
    - To test panicky situations
    - works on the principle, we randomly break the system (Example are close connection, close nodes, add latency, increase traffic, increase CPU utilization, Data losses in network, Traffic spikes)



Chaos means under those circumstances, any system can break

Popular tools for Chaos testing
1. Chaos monkey (tool Netflix uses, it used to kill the instances randomly. We look for SLA, SLO, SLI, they must not be impacted when any of the part is down)
2. ChaosMesh (to test Kubernetes where we can inject network failures, to add delay to network, we can add any I/O faults, Freeze memory inside CPU). When we create chaos or fail the system components intentionally, we need to observe how the system behaves. Say if we fail WebServer2, Database2 will be impacted. That means we got to create some communications between databases or connection between WebServer1 and Database2. Core purpose of chaos testing is, to know our system better.

What do we measure?
- We measure the error-rate (How much errors our system have currently?)
- We can track latency via P values P99, P50, P90, P95
- Recovery time (If we break a component, how much time to make that component available?)
- Latency with SLO/SLA
- Alert correctness

Say we have two issues: 1. Price not correct 2. Discount is not calculating correctly
Even though both issues are related, it is recommended to have them as 2 separate bugs