141. What is the difference between logging, monitoring, and observability?

Logging captures application events. Monitoring tracks metrics like CPU and memory. Observability lets you infer internal state using logs, metrics, and traces.

142. Why is observability important in microservices?

It provides visibility into distributed services, helps with root cause analysis, real-time monitoring, and proactive alerting.

143. What are the three pillars of observability?

1. Logs – What happened?
2. Metrics – How much/how fast?
3. Traces – Where did the request go?

144. What is structured logging?

Structured logging outputs logs in key-value format, making them easier to parse and query. Example:
{ "level": "INFO", "user": "Ashif", "action": "LoginSuccess" }

145. What are some common tools used for logging in microservices?

- Logback / Log4j
- ELK Stack (Elasticsearch, Logstash, Kibana)
- Fluentd + Grafana Loki
- Graylog

146. What is centralized logging?

Collecting logs from all services into one place for searching, analyzing, and monitoring across the entire system.

147. What are some common monitoring tools?

- Prometheus
- Grafana
- Datadog
- New Relic
- AppDynamics
- Dynatrace

148. What is distributed tracing?

It traces the journey of a request through multiple services using trace IDs and span IDs.

149. Tools for distributed tracing?

- Spring Cloud Sleuth
- Zipkin
- Jaeger
- OpenTelemetry

150. What is Spring Cloud Sleuth?

A Spring Cloud library that adds traceId and spanId to logs automatically and integrates with Zipkin and Brave for tracing.