Spring Boot Microservices

Beginner to Guru

Distributed Tracing

What is Distributed Tracing?

- Monoliths have the luxury of being self contained, thus tracing typically is not needed
- Transactions in microservices can span many services / instances, and even data centers
- Distributed tracing provides the tools to trace a transaction across services and nodes
- Distributed tracing is used for two aspects:
  - Performance monitoring across steps
  - Logging / troubleshooting

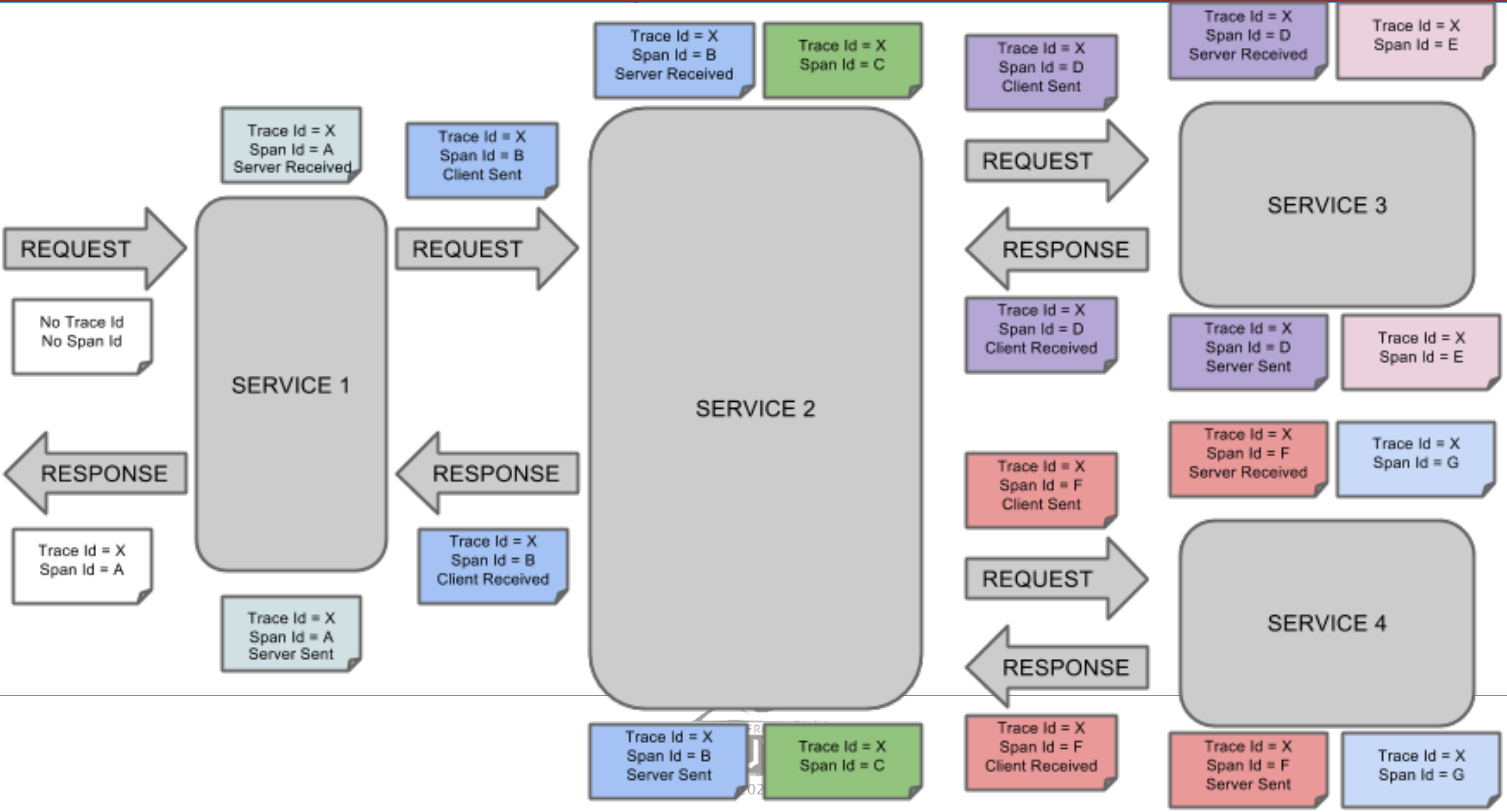
Spring Cloud Sleuth

- Spring Cloud Sleuth is the distributed tracing tool for Spring Cloud
- Spring Cloud Sleuth uses an open source distributed tracing library called Brave
- Conceptually what happens:
  - Headers on HTTP requests or messages are enhanced with trace data
  - Logging is enhanced with trace data
  - Optionally trace data can be reported to Zipkin

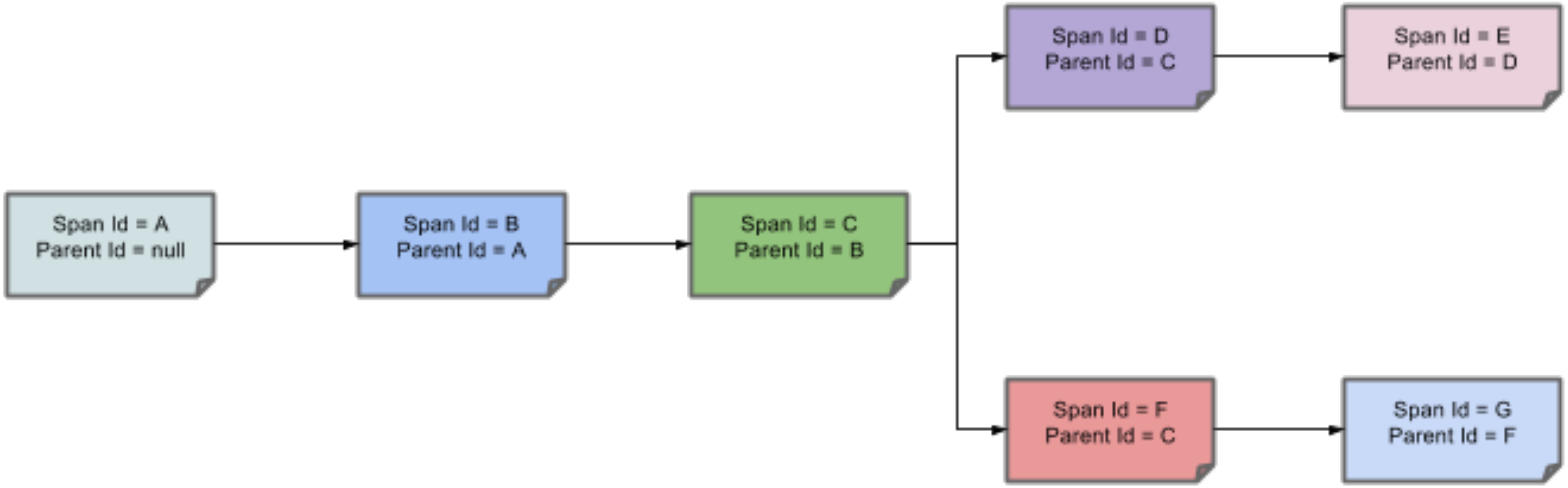
Tracing Terminology

- Spring Cloud Sleuth uses terminology established by Dapper
  - Dapper is a distributed tracing system created by Google for their production systems
- **Span** - is a basic unit of work. Typically a send and receive of a message.
- **Trace** - A set of spans for a transaction
- **cs / sr** - Client Sent / Server Received - aka the request
- **ss / cr** - Server Sent / Client Received - aka the response

Source Pivotal



Source Pivotal



Zipkin Server

- Zipkin is an open source project used to report distributed tracing metrics
- Information can be reported to Zipkin via webservices via HTTP
  - Optionally metrics can be provided via Kafka or Rabbit
- Zipkin is a Spring MVC project
  - Recommended to use binary distribution or Docker image
  - Building your own is not supported
- Uses in memory database for development
  - Cassandra or Elasticsearch should be used for production

Zipkin Quick Start

- Via Curl:
  - `curl -sSL https://zipkin.io/quickstart.sh | bash -s`
  - `java -jar zipkin.jar`
- Via Docker (Recommend for course):
  - `docker run -d -p 9411:9411 openzipkin/zipkin`
- View traces in UI at:
  - `http://your_host:9411/zipkin/`

Installing Spring Cloud Sleuth

- org.springframework.cloud:spring-cloud-starter-sleuth
  - Starter for logging only
- org.springframework.cloud:spring-cloud-starter-zipkin
  - Starter for Sleuth with Zipkin - includes Sleuth dependencies
- Property spring.zipkin.baseUrl is used to configure Zipkin server

Logging Output

- **Example**: - `DEBUG [beer-service,39853b63c1c3f919,419b9ac9a073bbba,true]`
  - [Appname, TraceId, SpanId, exportable]
- **Appname** - Spring Boot Application Name
- **TraceId** - Id value of the trace
- **SpanId** - Id of the Span
- **Exportable** - Should span be exported to Zipkin? (Programmatic configuration option)

Logging Configuration

- Microservices typically will use consolidated logging
- Number of different approaches for this - highly dependent on deployment environment
- Consolidated logging will be covered in a future section of the course
- To support consolidated logging, log data should be available in JSON
- Spring Boot by default uses logback, which is easy to configure for JSON output
  - To be covered in separate lesson

SPRING FRAMEWORK GURU