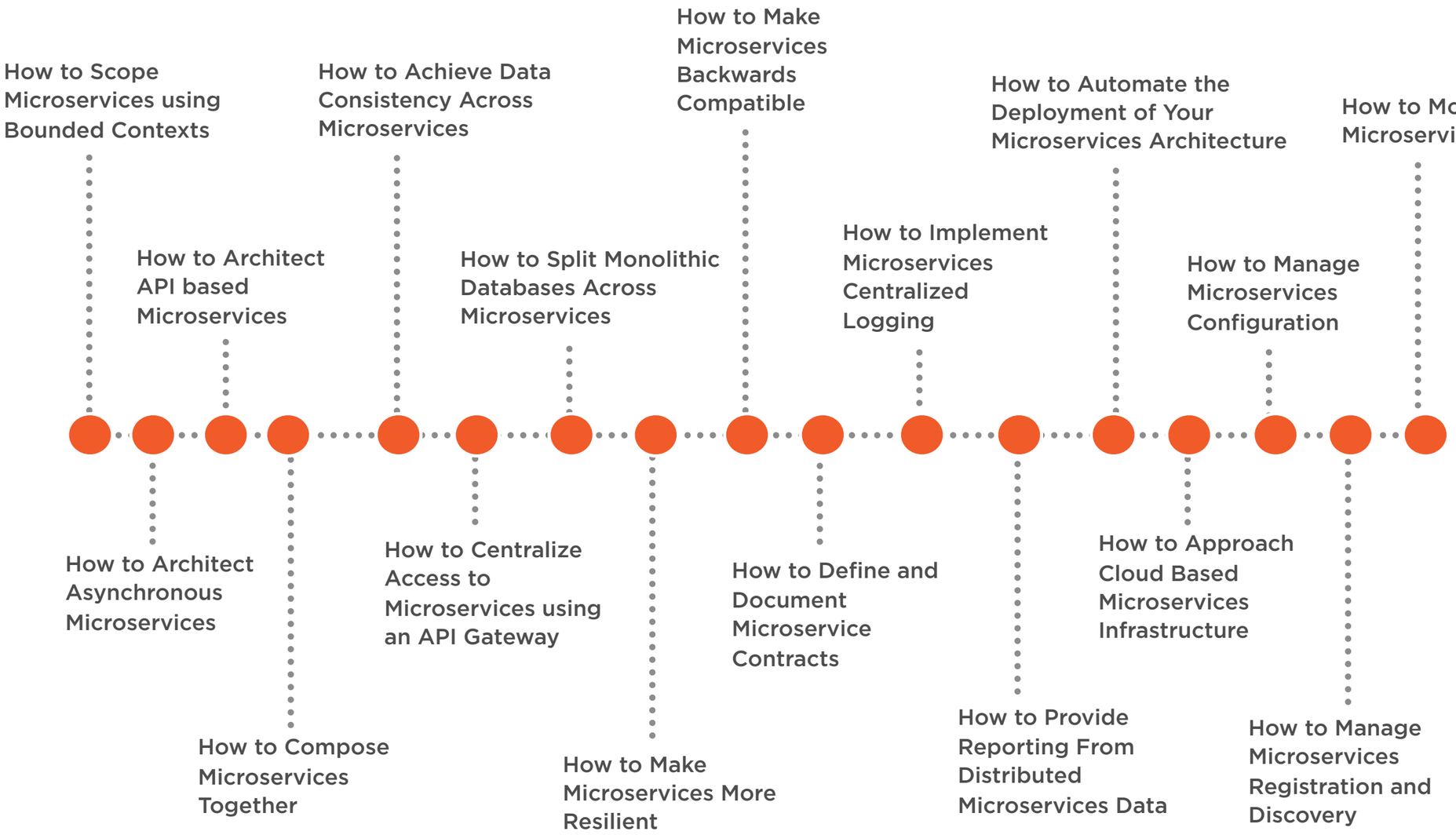
How to Implement Microservices Centralized Logging

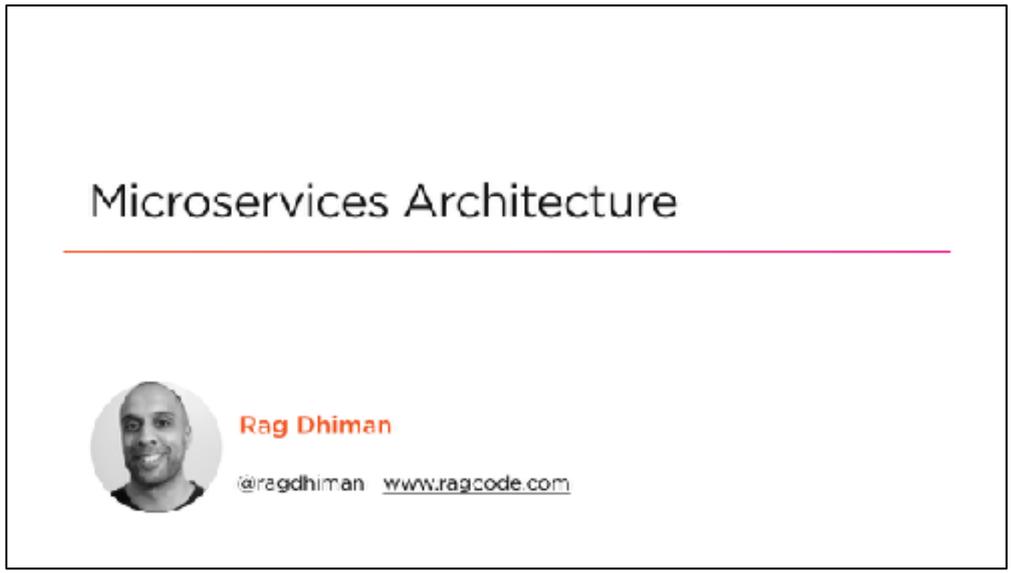
Rag Dhiman

@ragdhiman www.ragcode.com

Microservices Architectural Design Patterns Playbook



Microservices Architectural Design Patterns Playbook



Overview

Centralized Logging

Consistent Logging Format

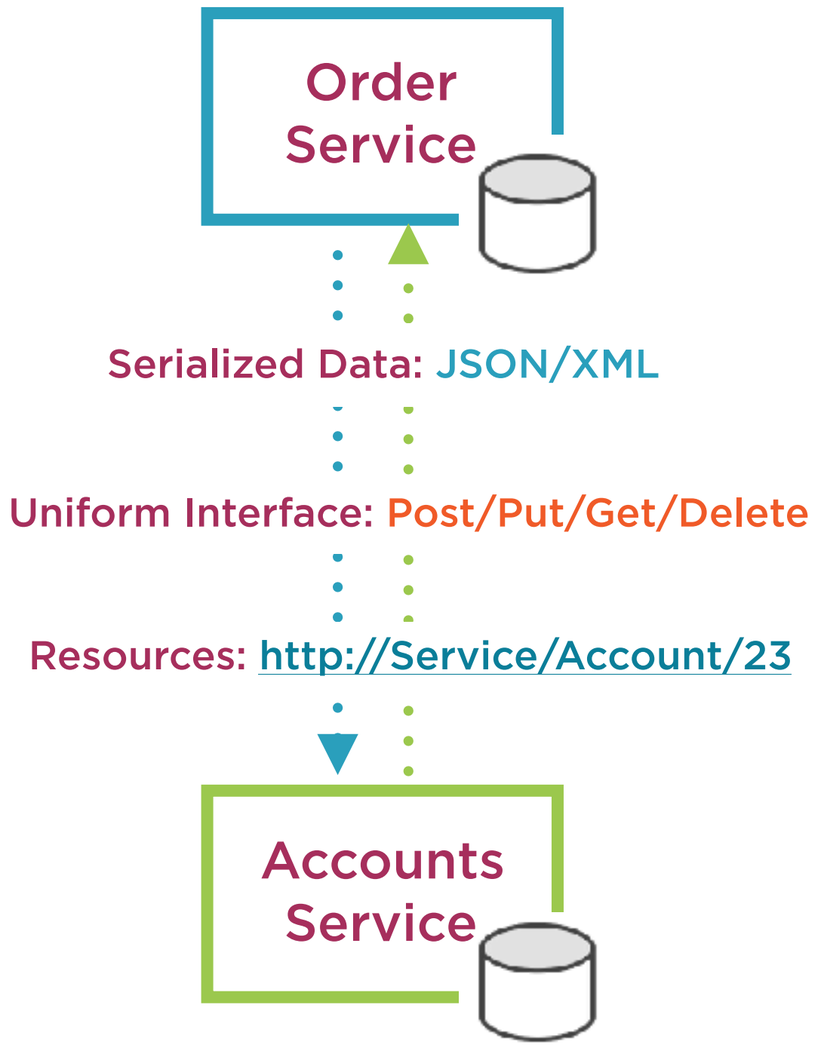
Logging Levels

Transaction Transparency

Centralized Logging Demo

Centralized Logging

Introduction



Distributed architecture

Centralized logging

Consistent format

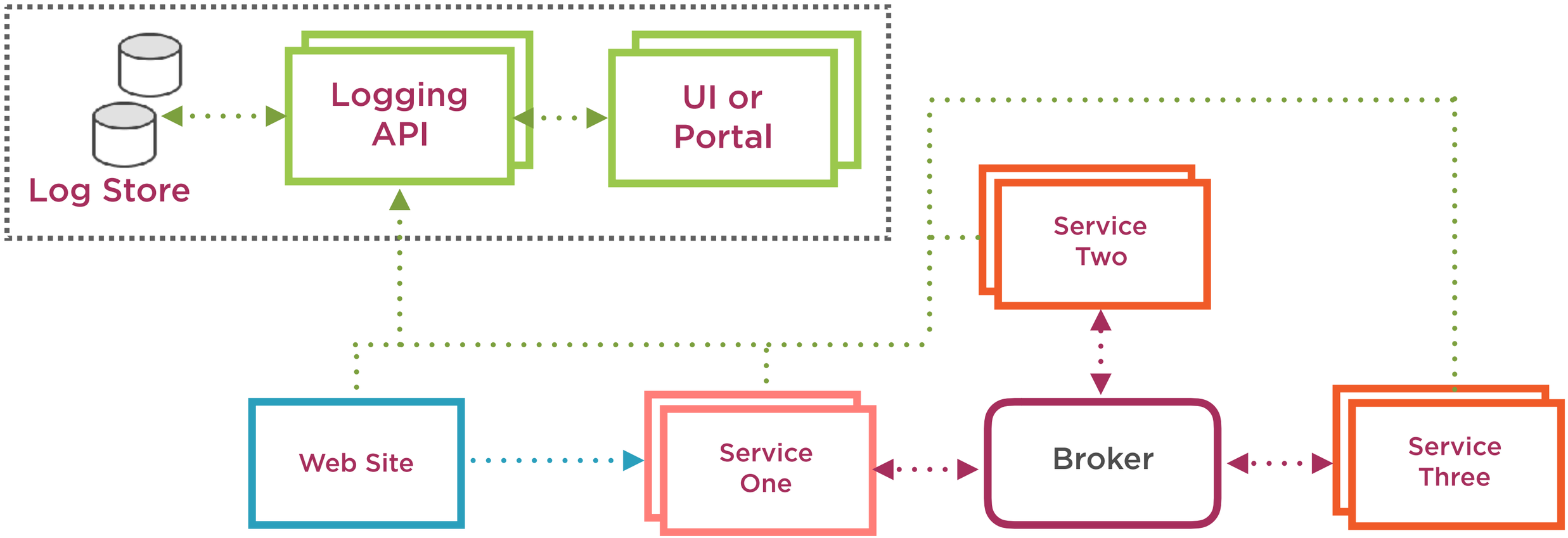
- Structured logging
- Shared libraries

Logging levels

Importance of transaction transparency

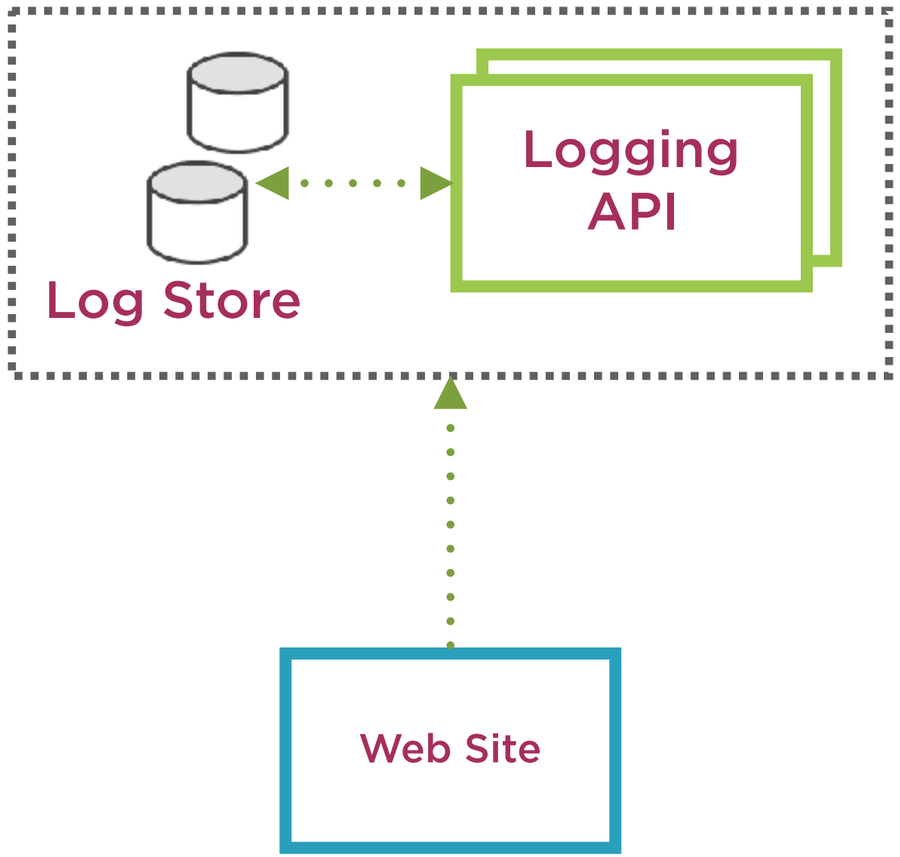
- Use of Correlation ID

Centralized Logging



Consistent Logging Format

Consistent Logging Format



Across the architecture

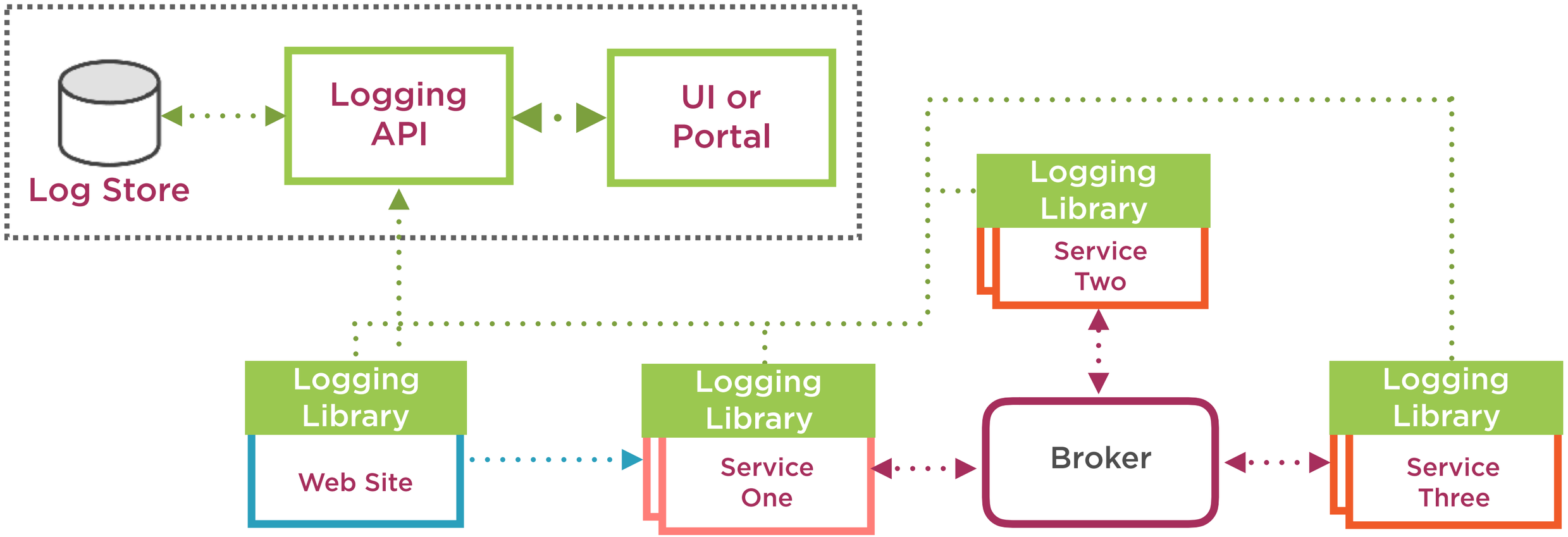
Structured format

- Levels
- Date and time
- Correlation ID
- Host and app information
- Message

Shared library

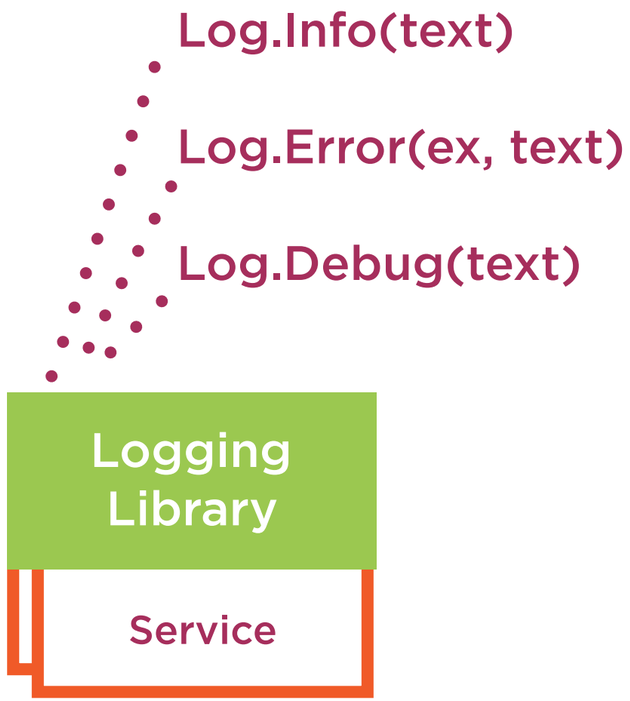
Decoupled from implementation

Consistent Logging Format



Logging Levels

Logging Levels



Information level

- Non-technical descriptive message
- Simple enough for entire business
- Key transaction milestones

Debug level

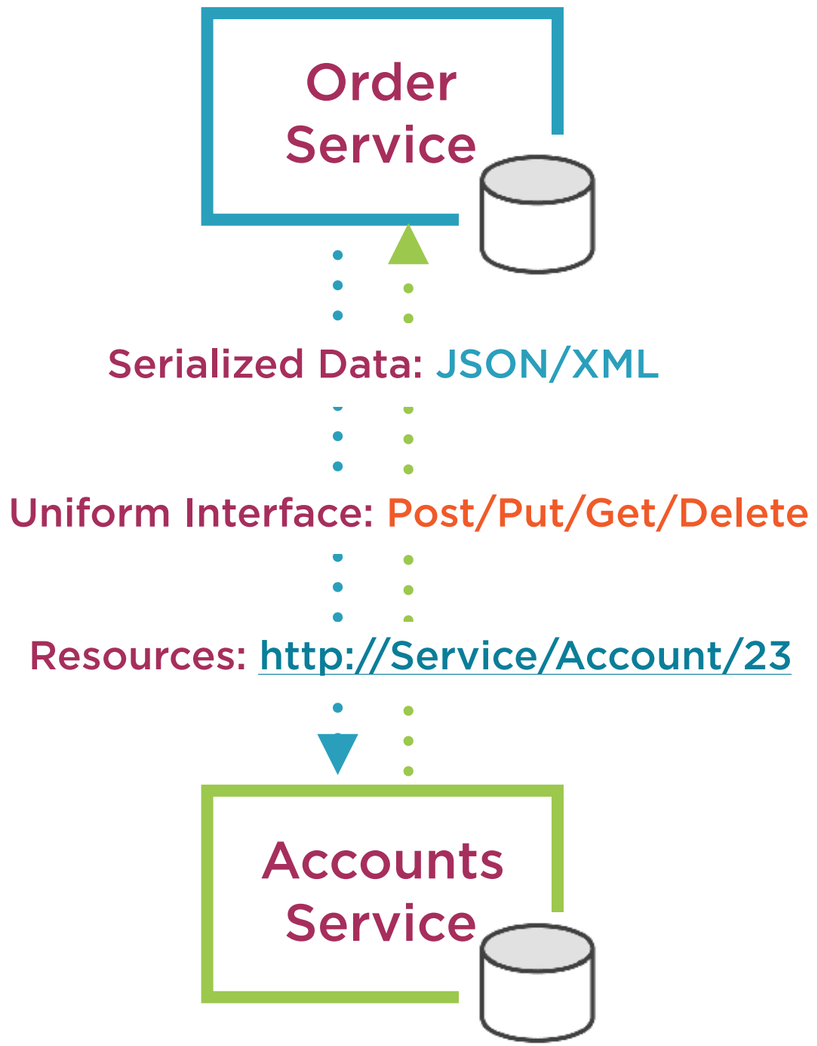
- Technical milestones
- Calls and parameters
- Response times and timeout stats

Error level

- Information on exceptions raised
- Error information and number
- Call stack

Transaction Transparency

Transaction Transparency



Consistent meta data part of the log

- Correlation ID
- Date and time

Live view of transaction

- Avoid log buffering
- Logs in sequence
- Date to support sequence
- Performance test logging frequency

Secure logging

- Controlled access to database and portal
- HTTPS logging connections
- In-line with data retention policies

Summary

- Centralized Logging
- Consistent Logging Format
- Logging Levels
- Transaction Transparency
- Centralized Logging Demo

Microservices Architectural Design Patterns Playbook