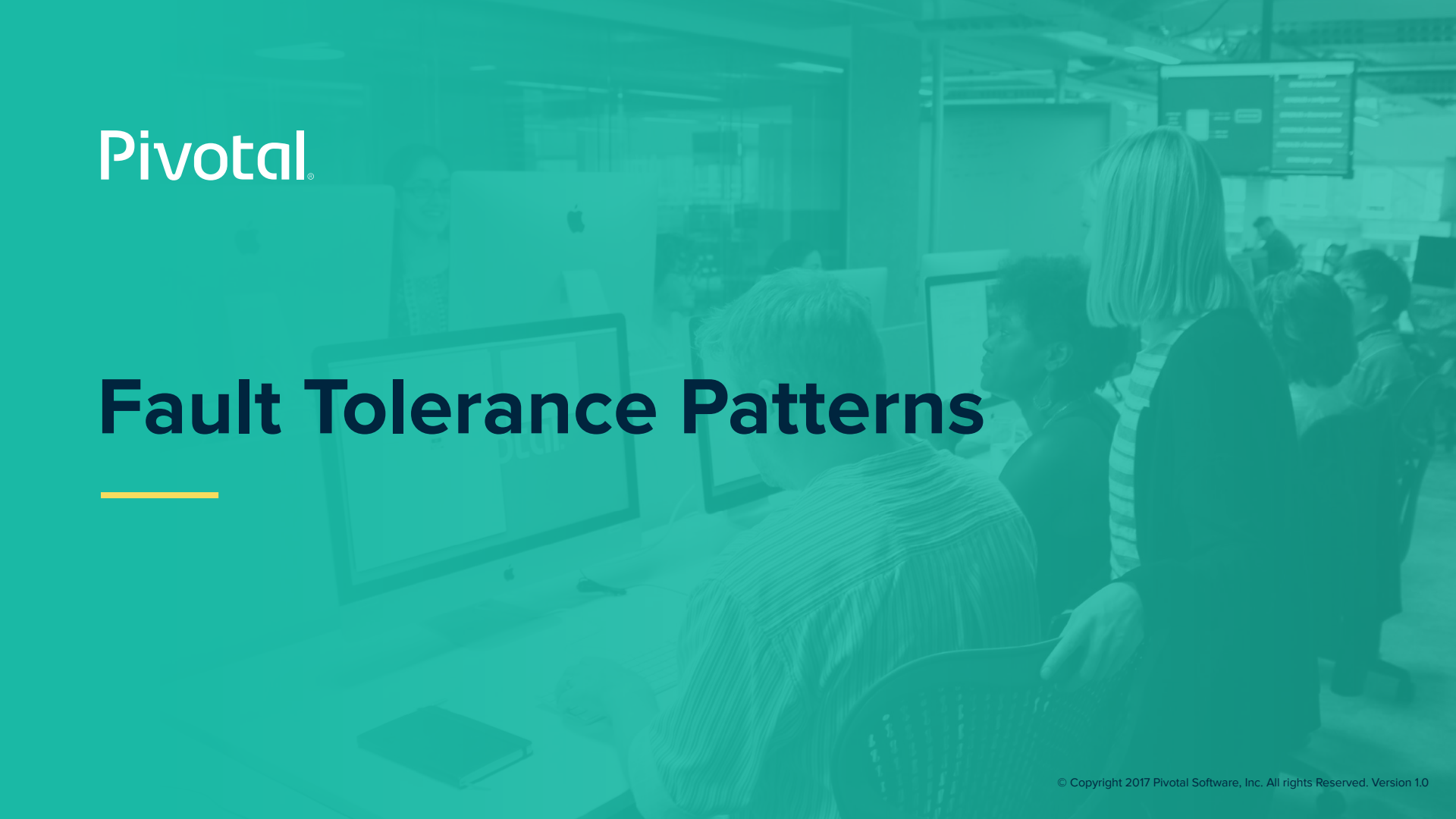
Pivotal

Fault Tolerance Patterns



Motivation

“Applications in complex distributed architectures have dozens of dependencies, each of which will inevitably fail at some point.

If the host application is not isolated from these external failures, it risks being taken down with them.”

Release It!

- V2, 2017
- Michael Nygard
- Covers patterns and case studies backing this unit

Consequences of Failure

- To Users & Customers:
  - Loss of Service to Users
  - Potential Loss of Life
- To Suppliers & Hosting Companies:
  - Lost Revenue
  - Lost Opportunity Cost
  - Punitive Damages
  - Reputation
- To You (Developer, Operations):
  - 3 am pages
  - Lost sleep
  - Reduction of Quality of Life

Key Definitions

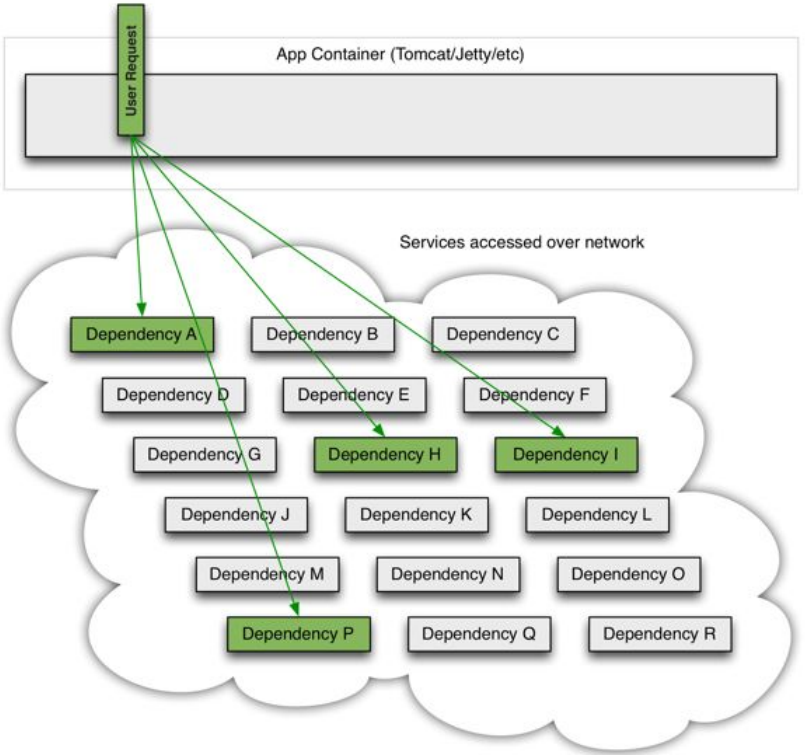
- **Service Level Objective (SLO)**: A goal of a system characteristic that is measurable:
  - Example: User perceived response time for a specific operation is less than 1 second
- **Service Level Indicator (SLI)**: a measurable quantity used to specify an SLO
  - Example: Latency of a specific operation measured in milliseconds
- **Correctness**: A system or component does what it is specified to do.
- **Liveness**: A system or component does what it is specified to do in a specified acceptable timeframe.

Problems Anti-Patterns

- Chain Reactions
- Cascading Failures
- Blocked Threads
- Slow Responses
- Dog Piles

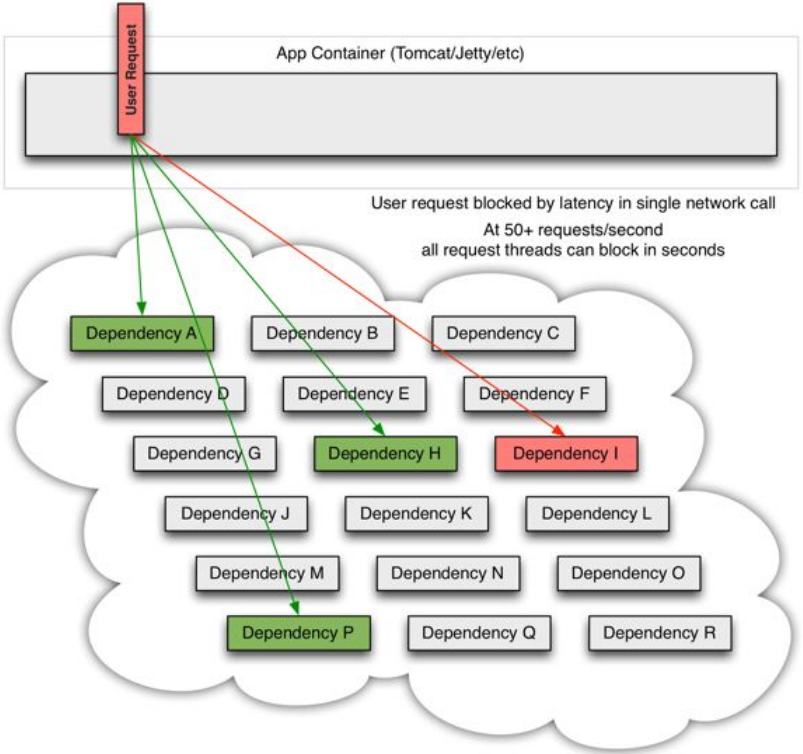
Services Dependency Scenario

- A typical application depending on a number of backing services
- Each backing service is an *Integration Point*
- All services are up and behaving normally



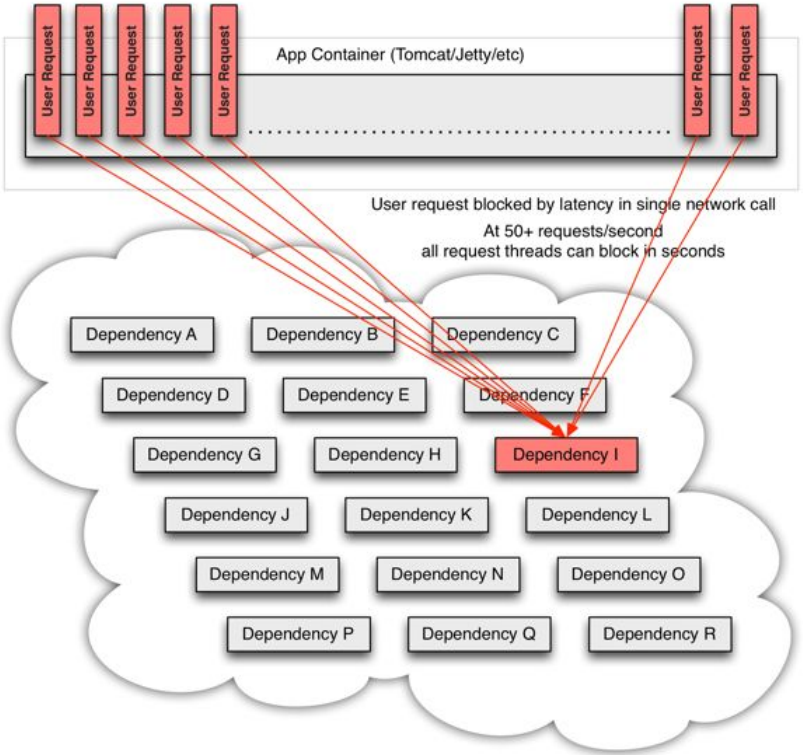
Failing Dependency

- A dependency misbehaves
- Response latency increases, tying up a thread in its calling application
- Large latencies result in *Slow Responses*.



Failure Cascades to Caller

- *Slow Responses* may contribute to *Blocked Threads*.
- Calling application's thread pool is exhausted waiting on misbehaving dependency. This is a *Chain Reaction* induced by *Blocked Threads*
- Failure cascades to caller, this results in *Cascading Failure*.
- Real users will not wait, they will terminate and retry requests, resulting in a *Dog Pile*.



Solutions Patterns covered in this course

- Let it crash
- Handshaking/Health Check
- Fail Fast
- Timeout, Retry and Backoff
- Bulkheads -> Shed Load
- Circuit Breaker

Solutions Patterns scope in this course

- This course covers *Remote Procedure Call (RPC)* style architectures.
- While some of the patterns can be used in Asynchronous or Event-based architectures, this course will not cover other than RPC context