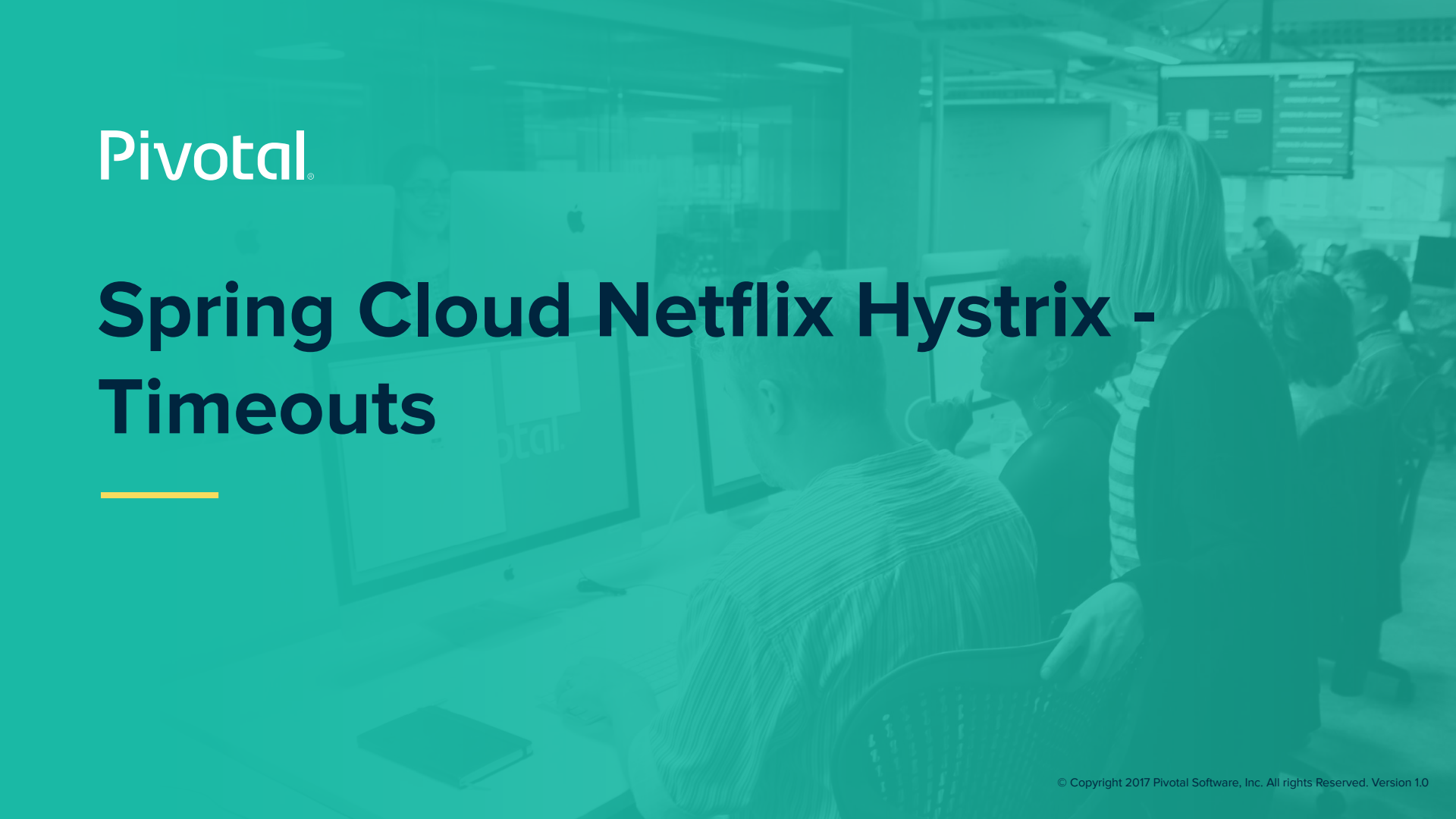
Pivotal

Spring Cloud Netflix Hystrix - Timeouts



Recap: Hystrix Client

- Hystrix client runs in-process to the application being protected
- Hystrix Command uses AOP (and associated Spring Dynamic Proxies) to wrap protected code
- Hystrix Proxies generate keyed thread pool used to run wrapped (protected) method, unless using Semaphores
- Hystrix Proxies will divert call execution path to fallback method upon failures

See following for more info:

- https://github.com/Netflix/Hystrix/wiki/How-it-Works#threads--thread-pools

How does Hystrix Timeout Work?

- Hystrix uses an Observer to monitor duration of the monitored thread execution.
- When the time exceeds the configured timeout, Hystrix will:
  - Throw a *TimeoutException*
  - Route request flow through fallback path
  - It *will not/cannot* force the latent thread to stop work, although it will throw an *InterruptedException* in the case the culprit thread can handle it.

See following for more info:

- https://github.com/Netflix/Hystrix/wiki/How-it-Works#6-hystrixobservablecommandconstruct-or-hystrixcommandrun

Impact of long running or stuck threads

- Given that Hystrix cannot terminate long running threads, it is probable in cascading slow-downs that the hystrix protected command will run out of threads.
- Hystrix will subsequently shed load.
- If you are using Hystrix to protect downstream requests over a network protocol that supports timeouts, use those in favor of Hystrix timeouts.

Monitoring and Contribution to Timeouts

- Hystrix dashboard renders the *TimeoutException* count in yellow.
- The *TimeoutException* count is included in the Circuit Breaker algorithm.

Hystrix Monitoring