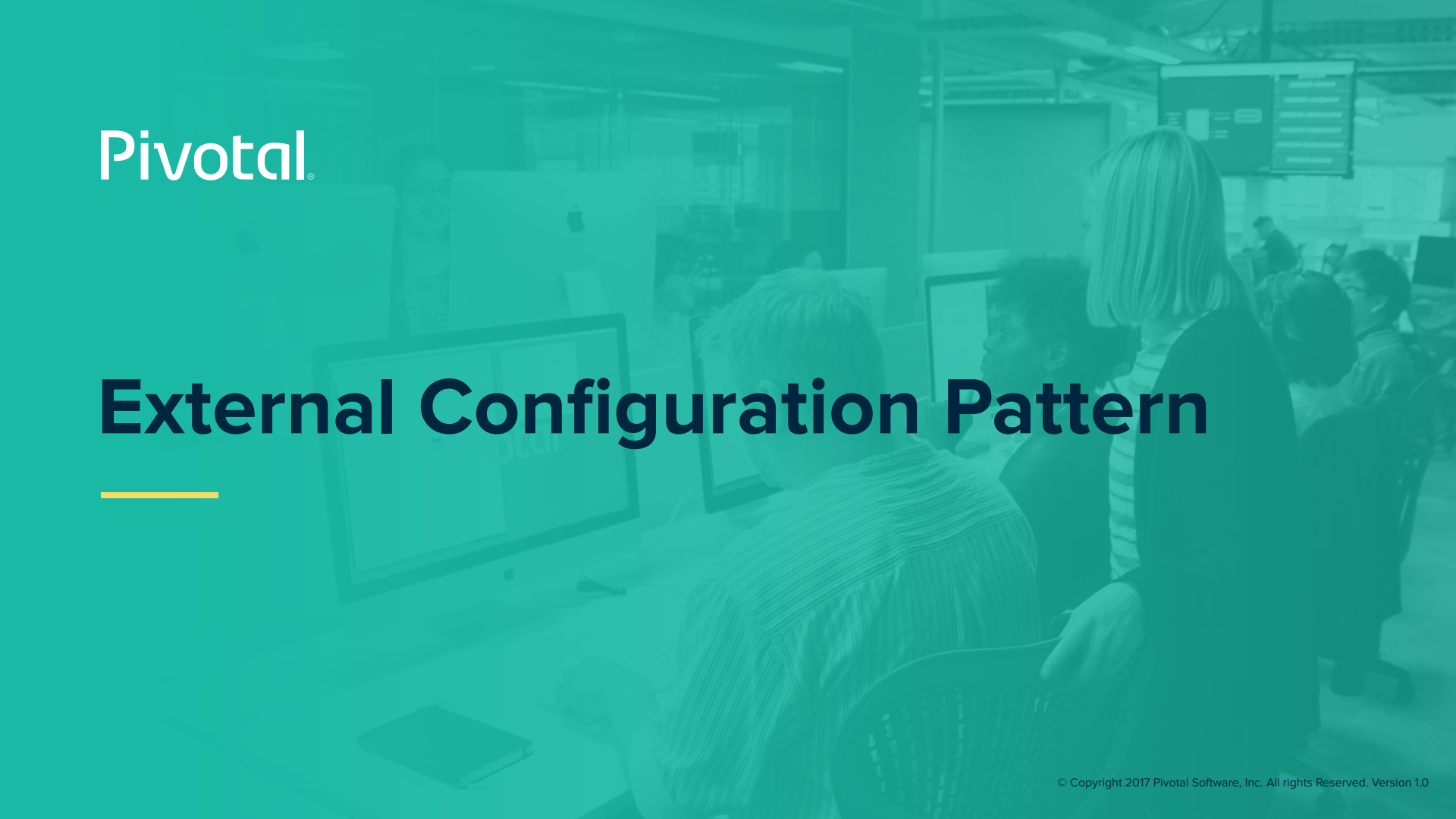
Pivotal

External Configuration Pattern



Background

- Traditionally with Spring applications, configuration is stored with the application, and fetched from the classpath
- Projects place configuration in a .properties or .yml file under src/main/resources
- Spring also supports reading configuration from:
  - Java system properties
  - Environment variables

Configuration Server - Concepts

- Externalization of configuration (outside the application)
- Centralization of configuration information for multiple services and environments
- Configuration as a service

Benefits

- Centralized Configuration
- Allow different rate of change between application (code), And runtime (operations)
- Reduced app restarts
- Configuration of sensitive configuration properties
- Choice of config store backends

Trade-Offs

- Additional distributed system complexity
  - Additional layers in config hierarchy
  - Additional components to scale and keep available

Use Case - Feature Toggles

- Divert Flow Control through External Configuration
  - Experimental Features
  - Release Features - Canaries
  - Non-functional (Operational Toggles)
  - Security (Permission Toggles)

See here for more information: https://martinfowler.com/articles/feature-toggles.html

Feature Toggles - Trade-offs

- Technical Debt
- Risks:
  - Defects
  - Potential vulnerabilities
- Considerations:
  - Release and Experimental Flags should be short-lived
  - Operational and Security Flags should be justified by Engineering review
  - If not using canaries or blue green, do not use release flags.