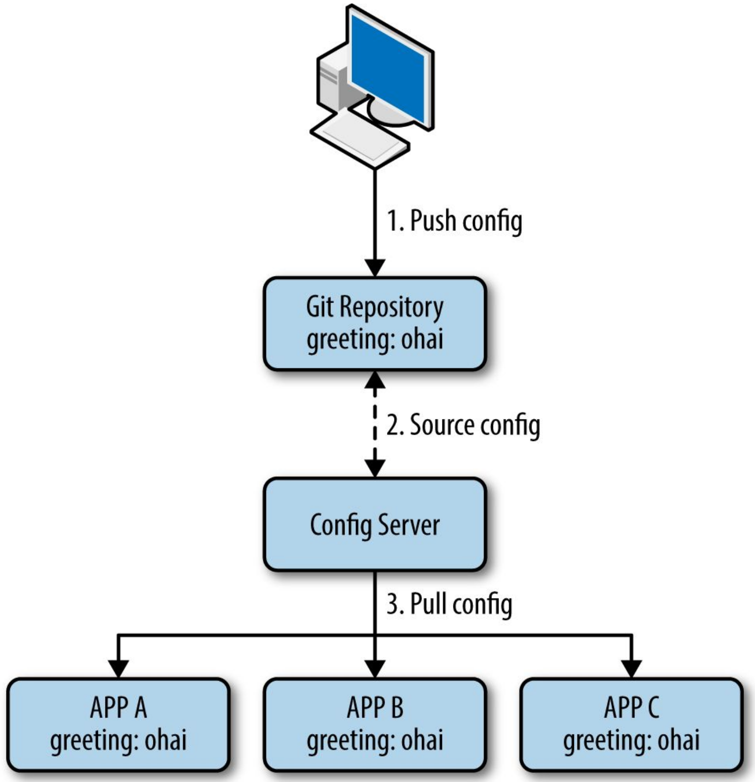
Pivotal

Spring Cloud Config - External Configuration

Spring Cloud Config Server - Design

- A RESTful interface to configuration backends
- Notion of a "backend" where configuration files are stored
- Config server reads configurations from the back-end and exposes HTTP REST endpoints for applications to consume
- Conventions: how files and REST endpoints are named make it easy to expose configuration for different applications in different environments

Spring Cloud Config Server - Design



Spring Cloud Config Server - Backends

- Supports multiple types of backends
  - Git
  - Subversion
  - HashiCorp Vault
  - CredHub
  - File System
  - JDBC
- Supplies extensible interfaces to add backend implementations via Environment adapters
- Also supports a composite of backends

Conventions

- An application’s `spring.application.name` is used to identify an application
- External configured stored in a Spring Cloud external configuration implementation
  - Spring Cloud Config Server
  - Spring Cloud Zookeeper
  - Spring Cloud Consul
  - Spring Cloud Kubernetes Config
- Each implementation will provide:
  - Hierarchical organization of configuration
  - Ability to leverage Spring profiles
  - May (or may not) provide source control versioning

Pivotal

Backend File Naming

Default Pattern:

```
{application}-{profile}.[properties|yml]
```

Example:

```
spring.application.name: greeting

spring.profiles.active: qa
```

Configuration stored in a file: `greeting-qa.yml` (or `greeting-qa.properties`)

NOTE: The pattern can be customized via a configuration property named searchPaths

Spring Cloud Config - HTTP Service Endpoints

```
/{application}/{profile}[/{label}]

/{application}-{profile}.yml

/{label}/{application}-{profile}.yml

/{application}-{profile}.properties

/{label}/{application}-{profile}.properties
```

With the Git backend, `{label}` maps to a branch name.

`{label}` is optional, and if not specified, defaults to master

Configuration Hierarchy

- Application Generic configuration: `application.yml`
- Application Generic configuration for an Environment: `application-{profile}.yml`
- Application Specific configuration: `{spring.app.name}.yml`
- Application Generic configuration for an Environment: `{spring.app.name}-{profile}.yml`

See following for Spring Configuration Orders-of-precedence:
https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html

Pivotal

Recommendations

- You have seen use of Profiles as part of available conventions for handling different environments, but should you use them?
- Ideally you should use different repositories for different environments:
  - Finer grain security
  - Finer grain control of environment specific
  - Different roles may manage configuration in separate environments.

Refresh Configuration

- Refresh via Pull Model
- Spring Boot Actuator /refresh endpoint