You have 4 environments for application ,how do you manage configurations for each environment.

The standard way to manage configurations across multiple environments (Dev, QA, UAT, Prod) in a Spring Boot microservices architecture is using **Spring Cloud Config Server** backed by a Git repository, or using environment-specific `application-{profile}.yml` files.

Here is the best-practice answer for an experienced developer:

1. The Strategy: "Externalized Configuration"

"I follow the **12-Factor App methodology**, which states that config should be strictly separated from code.
We use **Spring Cloud Config Server** to manage configurations centrally for all 4 environments. This allows us to change configuration without redeploying the application."

2. How it works (The Setup)

- **Central Git Repo:** We have a private Git repository (e.g., `config-repo`) storing files like:
  - `payment-service.yml` (Global defaults)
  - `payment-service-dev.yml` (Dev specific DB/Logs)
  - `payment-service-qa.yml` (QA specific)
  - `payment-service-prod.yml` (Prod specific - Secrets encrypted)
- **Config Server:** A dedicated Spring Boot microservice acts as the "Config Server." It connects to the Git repo.
- **Microservices:** All other services (like `PaymentService`) connect to the Config Server at startup to fetch their config.

3. Managing Secrets (Sensitive Data)

"For production secrets (DB passwords, API keys), we never store plain text in Git.

1. **Encryption:** We use Jasypt or Spring Cloud Config's built-in encryption (`{cipher}encrypted-value`) to store secrets encrypted in Git.
2. **Vault:** In more secure setups, we integrate **HashiCorp Vault** with Spring Boot to inject secrets at runtime."

4. Simple Alternative (If no Config Server)

"For smaller applications without a Config Server, I use **Profile-specific properties**:

- I define `application-dev.yml`, `application-prod.yml` inside `src/main/resources`.
- At deployment time, I inject the active profile via an environment variable: `JAVA_OPTS=-Dspring.profiles.active=prod`."

Summary Answer

"We use a centralized **Spring Cloud Config Server** backed by a Git repo. This ensures consistency, version control for configs, and allows us to update properties without rebuilding the jar. Secrets are handled via **Vault** or **Cipher encryption**."

⁂

1. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/429036f4-3bc3-4a7a-8c6f-90246e73b9b9/image.jpg
2. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/24f18c0a-b56e-4862-a619-e2959036a5c2/image.jpg
3. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/97e9ed86-ed11-4035-a18b-1eb05c1bab4e/image.jpg
4. https://ppl-ai-file-upload.s3.amazonaws.com/web/direct-files/attachments/images/81815274/87f62423-96f2-4071-9802-8f6699e0ecd8/image.jpg