مدیریت مراحل چرخه حیات محصول در سیستم CI/CD جدید

مقدمه

در نسخه‌های قبلی مونوریپو، ممکن بود برای هر `app` (پروژه) فولدرهایی با نام‌های `idea` ، `development` ، `stage` ، `deploy` و `product` وجود داشته باشد. هدف از ایجاد چنین ساختاری، تفکیک فیزیکی مراحل مختلف چرخه حیات توسعه نرم‌افزار برای هر پروژه بود. این رویکرد در برخی سناریوها می‌تواند مفید باشد، اما در یک مونوریپو با سیستم CI/CD پیشرفته، معمولاً به سمت یک مدل مدیریت متمرکزتر و خودکارتر حرکت می‌کنیم.

هدف از فولدرهای قبلی (Idea, Development, Stage, Deploy, Product)

1. **`idea` (ایده/مفهوم‌پردازی)**:
   - **هدف احتمالی**: نگهداری مستندات اولیه، طرح‌های مفهومی، تحقیقات بازار، یا حتی نمونه‌های اولیه (proof-of-concept) بسیار ابتدایی برای یک ویژگی یا محصول جدید. این فولدر می‌توانست محلی برای جمع‌آوری اطلاعات قبل از شروع کدنویسی واقعی باشد.
2. **`development` (توسعه)**:
   - **هدف احتمالی**: نگهداری کد در حال توسعه، فایل‌های پیکربندی محلی برای توسعه‌دهندگان، یا اسکریپت‌های خاص توسعه. این فولدر ممکن بود برای جداسازی کد ناپایدار یا ویژگی‌های در حال ساخت از نسخه‌های پایدارتر استفاده شود.
3. **`stage` (استیجینگ)**:
   - **هدف احتمالی**: نگهداری نسخه‌های کد آماده برای استقرار در محیط استیجینگ. این فولدر می‌توانست شامل فایل‌های پیکربندی خاص محیط استیجینگ، اسکریپت‌های استقرار مربوط به آن محیط، یا حتی بیلدهای کامپایل شده باشد.
4. **`deploy` (دیپلوی)**:
   - **هدف احتمالی**: نگهداری اسکریپت‌ها و ابزارهای مورد نیاز برای فرآیند استقرار (Deployment) در محیط‌های مختلف. این فولدر می‌توانست شامل دستورالعمل‌های گام به گام برای انتشار، فایل‌های تنظیمات سرور، یا ابزارهای اتوماسیون استقرار باشد.
5. **`product` (پروداکشن)**:
   - **هدف احتمالی**: نگهداری نسخه‌های کد آماده برای استقرار در محیط پروداکشن. مشابه `stage` ، این فولدر می‌توانست شامل فایل‌های پیکربندی خاص محیط پروداکشن و اسکریپت‌های انتشار نهایی باشد.

چالش‌های این رویکرد: در حالی که این ساختار می‌تواند مراحل را به وضوح تفکیک کند، اما در یک مونوریپو می‌تواند منجر به تکرار کد، پیچیدگی در مدیریت وابستگی‌ها، و دشواری در هماهنگ‌سازی فرآیندهای CI/CD شود. هر پروژه نیاز به مدیریت جداگانه این فولدرها و محتویات آن‌ها داشت که مقیاس‌پذیری را کاهش می‌داد.

کنترل مراحل کار در سیستم CI/CD جدید (بدون فولدرهای مرحله‌ای)

در سیستم CI/CD جدید که پیاده‌سازی شده است، این مراحل به جای تفکیک فیزیکی در ساختار فولدرها، به صورت **منطقی و فرآیندی** در خط لوله CI/CD مدیریت می‌شوند. این رویکرد مزایای قابل توجهی در اتوماسیون، مقیاس‌پذیری و کاهش تکرار دارد:

1. **مرحله ایده/مفهوم‌پردازی (Idea/Conception):**

- **مدیریت در سیستم جدید:** این مرحله عمدتاً خارج از کدبیس و در ابزارهای مدیریت پروژه، جلسات تیم، و مستندات اولیه (که می‌توانند در فولدر `/docs` مونوریپو نگهداری شوند) مدیریت می‌شود. زمانی که ایده به اندازه کافی پخته شد، کدنویسی آغاز شده و مستقیماً به شاخه‌های توسعه (مانند `feature branches`) اضافه می‌شود.

2. **مرحله توسعه (Development):**

- **مدیریت در سیستم جدید:** تمامی کد توسعه در فولدر اصلی `<apps/<project_name` نگهداری می‌شود. تفکیک بین کد در حال توسعه و کد پایدار از طریق **سیستم کنترل نسخه (Git)** و **شاخه‌های مختلف (Branches)** انجام می‌شود. برای مثال، `feature branches` برای توسعه ویژگی‌های جدید و `main` برای کد پایدار استفاده می‌شود. سیستم CI/CD با هر `push` به این شاخه‌ها فعال شده و تست‌های مربوطه را اجرا می‌کند.

3. **مرحله استیجینگ (Staging):**

- **مدیریت در سیستم جدید:** دیگر نیازی به فولدر فیزیکی `stage` نیست. در عوض، **محیط‌های استقرار (Deployment Environments)** در GitHub Actions (مانند `Test` و `Staging`) به صورت منطقی تعریف می‌شوند. فایل `config/projects.yaml` مشخص می‌کند که هر پروژه چگونه در این محیط‌ها مستقر شود. `mamos_runner.py` با استفاده از `render_service_id` مربوط به محیط استیجینگ، استقرار را در Render.com آغاز می‌کند. **تایید دستی (Manual Approval)** در GitHub Actions تضمین می‌کند که استقرار در استیجینگ تنها پس از بازبینی و تایید انسانی انجام شود.

4. **مرحله دیپلوی (Deploy):**

- **مدیریت در سیستم جدید:** فرآیند دیپلوی به صورت یک **گام در خط لوله CI/CD** تعریف شده است. `mamos_runner.py` مسئول اجرای دستورات استقرار است که شامل فراخوانی API پلتفرم‌های ابری (مانند Render.com) برای آغاز استقرار می‌شود. این فرآیند به صورت خودکار پس از موفقیت‌آمیز بودن مراحل ساخت و تست آغاز می‌شود و بسته به محیط (استیجینگ یا پروداکشن) ممکن است نیاز به تایید دستی داشته باشد.

5. **مرحله پروداکشن (Production):**

- **مدیریت در سیستم جدید:** مشابه استیجینگ، محیط پروداکشن نیز به صورت یک **محیط استقرار منطقی** در GitHub Actions تعریف می‌شود. استقرار در پروداکشن نیز از طریق `mamos_runner.py` و با استفاده از `render_service_id` مربوط به محیط پروداکشن انجام می‌شود.

این مرحله نیز دارای **تایید دستی اجباری** است تا از انتشار ناخواسته یا دارای مشکل به محیط زنده جلوگیری شود.

مزایای رویکرد جدید

- **کاهش تکرار (Reduced Duplication)**: با حذف فولدرهای تکراری، از نگهداری چندین کپی از کد یا پیکربندی‌های مشابه جلوگیری می‌شود.
- **اتوماسیون پیشرفته (Enhanced Automation)**: تمامی مراحل از توسعه تا استقرار به صورت خودکار توسط خط لوله CI/CD مدیریت می‌شوند، که سرعت و کارایی را افزایش می‌دهد.
- **مدیریت متمرکز (Centralized Management)**: پیکربندی CI/CD برای تمامی پروژه‌ها در فایل‌های `config/projects.yaml` و `GitHub Actions Workflows` به صورت متمرکز مدیریت می‌شود.
- **کنترل و امنیت (Control and Security)**: استفاده از محیط‌های GitHub Actions و تاییدهای دستی، کنترل دقیقی بر روی زمان و نحوه انتشار کد به محیط‌های حساس فراهم می‌کند.
- **مقیاس‌پذیری (Scalability)**: افزودن پروژه‌های جدید به مونوریپو و تعریف CI/CD برای آن‌ها بسیار ساده‌تر است، زیرا نیازی به ایجاد ساختارهای فولدر پیچیده و تکراری نیست.

نتیجه‌گیری

در حالی که فولدرهای مرحله‌ای در گذشته ممکن بود برای تفکیک بصری مراحل مفید باشند، سیستم CI/CD جدید با استفاده از ابزارهای اتوماسیون و پیکربندی متمرکز، این مراحل را به صورت کارآمدتر و مقیاس‌پذیرتری مدیریت می‌کند. این رویکرد به تیم توسعه اجازه می‌دهد تا با تمرکز بیشتر بر روی کدنویسی و نوآوری، محصولات را با کیفیت بالاتر و سرعت بیشتری به دست کاربران برسانند.