مدیریت چرخه حیات محصول (MVP تا پروداکشن) توسط سیستم CI/CD

مقدمه

سیستم یکپارچه‌سازی مداوم/استقرار مداوم (CI/CD) که برای مونوریپو `aladdin-sandbox` پیاده‌سازی شده است، یک چارچوب جامع برای مدیریت تمامی مراحل چرخه حیات توسعه محصول، از ایده اولیه (MVP) تا استقرار نهایی در محیط پروداکشن، فراهم می‌کند. این سیستم با خودکارسازی فرآیندها، تضمین کیفیت و ایجاد کنترل‌های لازم، به تیم‌ها کمک می‌کند تا محصولات خود را با سرعت، اطمینان و کارایی بالا به بازار عرضه کنند.

1. توسعه محصول حداقل پذیرفتنی (MVP Development)

در مرحله توسعه MVP (Minimum Viable Product)، تمرکز بر روی ساخت سریع و ارائه قابلیت‌های اصلی محصول است. سیستم CI/CD در این مرحله نقش حیاتی ایفا می‌کند:

- **تکرار سریع و بازخورد فوری:** توسعه‌دهندگان می‌توانند تغییرات کوچک و مکرر را به کد اعمال کرده و بلافاصله از طریق خط لوله CI/CD بازخورد دریافت کنند. این امر به شناسایی و رفع سریع مشکلات در مراحل اولیه کمک کرده و امکان تکرار سریع طراحی و پیاده‌سازی را فراهم می‌آورد.
- **تست خودکار:** حتی در مرحله MVP، تست‌های خودکار (واحد و یکپارچه‌سازی) اجرا می‌شوند تا اطمینان حاصل شود که قابلیت‌های اصلی به درستی کار می‌کنند و تغییرات جدید باعث شکست قابلیت‌های موجود نمی‌شوند. این امر از انباشته شدن بدهی فنی جلوگیری می‌کند.
- **محیط‌های ایزوله برای تست:** `mamos_runner.py` و GitHub Actions امکان اجرای تست‌ها را در محیط‌های ایزوله فراهم می‌کنند، که تضمین می‌کند نتایج تست‌ها قابل اعتماد هستند و تحت تأثیر عوامل خارجی قرار نمی‌گیرند.

2. توسعه (Development)

این مرحله شامل پیاده‌سازی کامل ویژگی‌ها، بهبود کد و رفع اشکالات است. CI/CD در اینجا به عنوان ستون فقرات فرآیند توسعه عمل می‌کند:

- **یکپارچه‌سازی مداوم (Continuous Integration):** هر بار که کدی به شاخه `main` (یا شاخه‌های مشخص شده دیگر) `push` می‌شود، خط لوله CI/CD به طور خودکار فعال می‌شود. این شامل:
- **ساخت (Build):** دستورات `build` تعریف شده در `config/projects.yaml` (مانند `pip install -r requirements.txt` برای پایتون یا `npm install && npm run build` برای Node.js) اجرا می‌شوند تا اطمینان حاصل شود که کد قابل ساخت و بسته‌بندی است.
- **تست (Test):** دستورات `test` (مانند `pytest` یا `npm test`) اجرا می‌شوند. این تست‌ها شامل تست‌های واحد، یکپارچه‌سازی و گاهی اوقات تست‌های عملکردی هستند تا کیفیت و صحت عملکرد کد تضمین شود.

- **گزارش‌دهی**: نتایج ساخت و تست در گزارش‌های Markdown (فایل‌های `_report.md` در `/reports/details`) ثبت می‌شوند و یک گزارش خلاصه (`summary.md`) وضعیت کلی را ارائه می‌دهد. این گزارش‌ها به توسعه‌دهندگان کمک می‌کنند تا به سرعت مشکلات را شناسایی و رفع کنند.
- **بررسی کیفیت کد و امنیت**: می‌توان ابزارهای تحلیل استاتیک کد و اسکنرهای امنیتی را به خط لوله CI/CD اضافه کرد تا کیفیت کد و آسیب‌پذیری‌های امنیتی به طور خودکار بررسی شوند.

3. استقرار در محیط استیجینگ (Staging Deployment)

محیط استیجینگ یک کپی نزدیک به محیط پروداکشن است که برای تست‌های نهایی، تست‌های پذیرش کاربر (UAT) و نمایش به ذینفعان استفاده می‌شود. سیستم CI/CD این مرحله را به شرح زیر مدیریت می‌کند:

- **استقرار خودکار/نیمه‌خودکار**: پس از موفقیت‌آمیز بودن مراحل ساخت و تست در CI، کد می‌تواند به طور خودکار به محیط استیجینگ مستقر شود. در سیستم پیاده‌سازی شده، این مرحله می‌تواند از طریق `workflow_dispatch` (اجرای دستی) با انتخاب محیط `Staging` فعال شود.
- **تایید دستی (Manual Approval)**: برای اطمینان از اینکه فقط کدهای تایید شده به محیط استیجینگ می‌رسند، GitHub Actions Workflows شامل یک مرحله تایید دستی است. این مرحله نیاز به بازبینی انسانی و تایید صریح قبل از ادامه استقرار دارد. این امر به تیم QA و ذینفعان اجازه می‌دهد تا آخرین بررسی‌ها را انجام دهند.
- **محیط‌های تعریف شده**: هر محیط استیجینگ دارای `render_service_id` خاص خود در `config/projects.yaml` است که امکان استقرار هدفمند و ایزوله را فراهم می‌کند.

4. استقرار در محیط پروداکشن (Production Deployment)

محیط پروداکشن جایی است که محصول نهایی در دسترس کاربران قرار می‌گیرد. این مرحله حساس‌ترین بخش چرخه حیات است و CI/CD با دقت آن را مدیریت می‌کند:

- **استقرار کنترل شده**: استقرار در پروداکشن نیز معمولاً از طریق `workflow_dispatch` و با انتخاب محیط `Production` آغاز می‌شود.
- **تایید دستی چند مرحله‌ای**: همانند استیجینگ، استقرار در پروداکشن نیز شامل یک مرحله تایید دستی اجباری است. این مرحله حتی سخت‌گیرانه‌تر است و ممکن است نیاز به تایید چندین نفر یا تیم داشته باشد. این کنترل اضافی برای جلوگیری از انتشار ناخواسته یا دارای مشکل به محیط زنده ضروری است.
- **استراتژی‌های استقرار**: سیستم CI/CD می‌تواند از استراتژی‌های پیشرفته استقرار مانند استقرار قناری (Canary Deployment) یا آبی-سبز (Blue-Green Deployment) پشتیبانی کند، اگرچه در پیاده‌سازی فعلی به صورت مستقیم پیکربندی نشده‌اند، اما زیرساخت لازم برای افزودن آن‌ها وجود دارد.
- **بازگشت به عقب (Rollback)**: در صورت بروز هرگونه مشکل پس از استقرار در پروداکشن، امکان بازگشت سریع به نسخه پایدار قبلی از طریق ابزارهای Render.com یا مکانیزم‌های مشابه فراهم است.

5. دیپلوی و استقرار (Deployment & Orchestration)

فرآیند کلی دیپلوی و استقرار توسط `mamos_runner.py` و GitHub Actions Workflows هماهنگ می‌شود:

- **`mamos_runner.py`**: این اسکریپت به عنوان یک ارکستراتور محلی عمل می‌کند که دستورات `build`, `test` و `deploy` را برای هر پروژه بر اساس پیکربندی `projects.yaml` اجرا می‌کند. این اسکریپت مسئول تعامل با Render.com API برای آغاز فرآیندهای استقرار است.
- **GitHub Actions Workflows**: این workflowها محیط اجرایی برای `mamos_runner.py` را فراهم می‌کنند. آن‌ها مسئول:
  - **راه‌اندازی محیط**: نصب وابستگی‌های لازم (مانند Python، PyYAML، requests).
  - **مدیریت مجوزها**: اطمینان از دسترسی‌های لازم برای `checks`, `pull-requests`, و `contents` و استفاده از `secrets` (مانند `RENDER_API_KEY`).
  - **اجرای مراحل**: فراخوانی `mamos_runner.py` با پارامترهای مناسب برای هر پروژه و محیط.
  - **گزارش‌دهی و آرتیفکت‌ها**: آپلود گزارش‌های CI/CD به عنوان آرتیفکت‌های GitHub Actions و همچنین `commit` کردن آن‌ها به ریپازیتوری.
  - **مدیریت محیط‌ها و تاییدها**: استفاده از قابلیت‌های `environments` گیت‌هاب برای تعریف URLهای محیط و اعمال قوانین محافظت (مانند تایید دستی).

نتیجه‌گیری

سیستم CI/CD پیاده‌سازی شده یک رویکرد ساختاریافته و خودکار برای مدیریت چرخه حیات محصول ارائه می‌دهد. از توسعه سریع MVP و تست‌های مداوم در مرحله توسعه گرفته تا استقرارهای کنترل شده و ایمن در محیط‌های استیجینگ و پروداکشن، تمامی جنبه‌ها به دقت مدیریت می‌شوند. این سیستم با کاهش خطاهای انسانی، افزایش سرعت تحویل و تضمین کیفیت، به تیم توسعه کمک می‌کند تا با اطمینان خاطر بیشتری محصولات نوآورانه را به دست کاربران برسانند.