راهنمای مدیریت پروژه در سیستم CI/CD: افزودن پروژه‌های قدیمی، دیپلوی MVP و توسعه توسط عامل هوش مصنوعی

مقدمه

این سند به سه سوال کلیدی در مورد مدیریت پروژه در سیستم CI/CD پیاده‌سازی شده برای مونوریپو `aladdin-sandbox` پاسخ می‌دهد: نحوه افزودن پروژه‌های قدیمی، فرآیند دیپلوی MVP، و چگونگی مشارکت عامل هوش مصنوعی (AI Agent) در توسعه پروژه‌ها.

1. افزودن پروژه‌های قدیمی به مونوریپو

افزودن یک پروژه قدیمی یا موجود به مونوریپو `aladdin-sandbox` و ادغام آن با سیستم CI/CD، یک فرآیند ساختاریافته است که شامل مراحل زیر می‌شود:

1.1. آماده‌سازی پروژه قدیمی

قبل از انتقال، پروژه قدیمی را بررسی کنید:

- **وابستگی‌ها:** تمامی وابستگی‌های پروژه (مانند `requirements.txt` برای پایتون، `package.json` برای Node.js) را شناسایی و مستند کنید.
- **دستورات بیلد و تست:** دستورات دقیق مورد نیاز برای بیلد (کامپایل، بسته‌بندی) و اجرای تست‌های پروژه را مشخص کنید.
- **پیکربندی محیطی:** هرگونه پیکربندی خاص محیطی که پروژه نیاز دارد (مانند متغیرهای محیطی، فایل‌های `env.`) را شناسایی کنید.

1.2. انتقال پروژه به ساختار مونوریپو

1. **ایجاد پوشه پروژه:** یک پوشه جدید برای پروژه قدیمی خود در مسیر `/aladdin-sandbox/apps` ایجاد کنید. نام پوشه باید با نام پروژه شما مطابقت داشته باشد (مثلاً `aladdin-sandbox/apps/legacy_project`).
2. **کپی کردن فایل‌ها:** تمامی فایل‌ها و پوشه‌های پروژه قدیمی را به این پوشه جدید کپی کنید.

1.3. پیکربندی `config/projects.yaml`

فایل `aladdin-sandbox/config/projects.yaml` را ویرایش کرده و یک ورودی جدید برای پروژه قدیمی خود اضافه کنید. این ورودی باید شامل موارد زیر باشد:

YAML

```yaml
  - name: legacy_project
    path: apps/legacy_project
```

```
    build: "<دستور بیلد پروژه قدیمی>"
    test: "<دستور تست پروژه قدیمی>"
    deploy:
      - environment: Test
        render_service_id: "legacy-project-test-service-id"
      - environment: Staging
        render_service_id: "legacy-project-staging-service-id"
      - environment: Production
        render_service_id: "legacy-project-production-service-id"
```

- **name** : نام منحصر به فرد پروژه (مثلاً `legacy_project`).
- **path** : مسیر نسبی پروژه در مونوریپو (مثلاً `apps/legacy_project`).
- **build** : دستور shell برای بیلد پروژه (مثلاً `pip install -r requirements.txt && python setup.py build` یا `npm install && npm run build`).
- **test** : دستور shell برای اجرای تست‌های پروژه (مثلاً `pytest` یا `npm test`).
- **deploy** : برای هر محیط استقرار (Test, Staging, Production)، یک `render_service_id` منحصر به فرد تعریف کنید. این شناسه‌ها باید از قبل در پلتفرم استقرار (مانند Render.com) ایجاد شده باشند.

1.4. ایجاد GitHub Actions Workflow

یک فایل Workflow جدید در مسیر `aladdin-sandbox/.github/workflows/legacy_project.yml` ایجاد کنید. می‌توانید از Workflowهای موجود (مانند `backend.yml`) به عنوان الگو استفاده کنید. نکات کلیدی:

- **name** : نام Workflow را به `Legacy Project CI/CD` تغییر دهید.
- **on: push: paths:** : این بخش را به `apps/legacy_project/**` تنظیم کنید تا Workflow فقط با تغییرات در این پروژه فعال شود.
- **فراخوانی mamos_runner.py** : مطمئن شوید که پارامتر `project--` در فراخوانی `mamos_runner.py` به `legacy_project` تنظیم شده است.
- **محیط‌های استقرار و تایید دستی**: محیط‌های استقرار (Test, Staging, Production) و تاییدهای دستی را مطابق با نیازهای پروژه پیکربندی کنید.

1.5. مدیریت Secrets و متغیرهای محیطی

- **GitHub Secrets**: هرگونه کلید API یا اطلاعات حساس مورد نیاز پروژه قدیمی را به عنوان Secrets در مخزن خود اضافه کنید.
- **Environment Secrets**: اگر متغیرهای محیطی خاصی فقط برای یک محیط خاص (مثلاً پروداکشن) تعریف GitHub Actions در تنظیمات محیط Environment Secrets معتبر هستند، آن‌ها را به عنوان کنید.

1.6. تست و اعتبارسنجی

- تغییرات خود را به یک شاخه جدید `push` کنید و Workflow جدید را به صورت دستی اجرا کنید تا از صحت عملکرد آن اطمینان حاصل کنید.
- تمامی مراحل بیلد، تست و استقرار را در محیط‌های مختلف بررسی کنید.

2. دیپلوی MVP (Minimum Viable Product)

فرآیند دیپلوی یک MVP در سیستم CI/CD ما به طور کامل خودکار و کنترل شده است. مراحل اصلی به شرح زیر است:

1. **تکمیل توسعه MVP**: پس از اینکه توسعه MVP به پایان رسید و تمامی ویژگی‌های اصلی آن پیاده‌سازی شد، کد باید در برنچ `main` ادغام شود.
2. **تریگر CI/CD**: هر `push` به برنچ `main` (یا برنچ‌های پیکربندی شده دیگر) به طور خودکار Workflowهای GitHub Actions مربوط به پروژه‌های تغییر یافته را فعال می‌کند.
3. **بیلد و تست خودکار**: سیستم CI/CD ابتدا کد MVP را بیلد کرده و سپس تمامی تست‌های خودکار (واحد، یکپارچه‌سازی) را اجرا می‌کند. این مرحله تضمین می‌کند که MVP از نظر فنی پایدار و بدون رگرسیون است.
4. **استقرار در محیط Test**: پس از موفقیت‌آمیز بودن بیلد و تست، MVP به طور خودکار در محیط `Test` مستقر می‌شود. این محیط برای تست‌های نهایی و اطمینان از عملکرد صحیح در یک محیط شبیه به پروداکشن استفاده می‌شود.
5. **استقرار در محیط Staging (با تایید دستی)**: پس از تایید در محیط Workflow ، `Test` به مرحله استقرار در محیط `Staging` می‌رسد. این مرحله نیاز به **تایید دستی** دارد. مدیران یا تیم محصول می‌توانند در این مرحله MVP را بازبینی کرده، تست‌های پذیرش کاربر (UAT) را انجام دهند و از آمادگی آن برای پروداکشن اطمینان حاصل کنند.
6. **استقرار در محیط Production (با تایید دستی)**: پس از تایید در محیط Workflow ، `Staging` به مرحله استقرار در محیط `Production` می‌رسد. این مرحله نیز نیاز به **تایید دستی** دارد. این لایه امنیتی نهایی تضمین می‌کند که تنها نسخه‌های کاملاً تایید شده و پایدار به کاربران نهایی ارائه می‌شوند.

نکات کلیدی برای دیپلوی MVP:

- **پیکربندی `config/projects.yaml`** : مطمئن شوید که `render_service_id` برای هر محیط (Test, Staging, Production) به درستی در این فایل برای پروژه MVP شما تنظیم شده است.
- **GitHub Actions Workflows**: Workflow مربوط به پروژه MVP باید شامل مراحل استقرار برای هر سه محیط Staging و Production باشد. همچنین تاییدهای دستی برای.
- **نظارت**: پس از دیپلوی، عملکرد MVP را در محیط پروداکشن به دقت نظارت کنید تا از پایداری و عملکرد صحیح آن اطمینان حاصل شود.

3. توسعه توسط عامل هوش مصنوعی (AI Agent)

به عنوان یک عامل هوش مصنوعی، من می‌توانم به روش‌های مختلفی در توسعه پروژه‌ها مشارکت داشته

باشم و راه حل‌هایی برای همکاری موثر وجود دارد:

3.1. مشارکت در توسعه کد

- **تغییرات کد کوچک و مشخص**: می‌توانم تغییرات کوچکی در کد ایجاد کنم، مانند رفع اشکالات (Bug Fixes)، بهبودهای عملکردی (Performance Improvements)، یا پیاده‌سازی ویژگی‌های کوچک و کاملاً تعریف شده.
- **بازسازی کد (Refactoring)**: می‌توانم بخش‌هایی از کد را برای بهبود خوانایی، نگهداری و کارایی بازسازی کنم.
- **نوشتن تست‌ها**: می‌توانم تست‌های واحد (Unit Tests) یا تست‌های یکپارچه‌سازی (Integration Tests) برای کد موجود یا ویژگی‌های جدید بنویسم.
- **به‌روزرسانی وابستگی‌ها**: می‌توانم وابستگی‌های پروژه را به‌روزرسانی کنم و تغییرات لازم را در کد برای سازگاری با نسخه‌های جدید اعمال کنم.

3.2. توسعه و نگهداری CI/CD

- **افزودن پروژه‌های جدید**: همانطور که در بخش 1 توضیح داده شد، می‌توانم پروژه‌های جدید را به مونوریپو اضافه کرده و Workflowهای CI/CD مربوطه را پیکربندی کنم.
- **بهبود `mamos_runner.py`** : می‌توانم `mamos_runner.py` را برای پشتیبانی از انواع جدید پروژه، بهبود منطق گزارش‌دهی، یا ادغام با ابزارهای جدید توسعه دهم.
- **به‌روزرسانی Workflowهای GitHub Actions**: می‌توانم Workflowهای موجود را برای بهینه‌سازی، افزودن مراحل جدید (مانند اسکن امنیتی یا تحلیل کیفیت کد) یا تغییر تریگرها به‌روزرسانی کنم.
- **مدیریت Secrets**: می‌توانم راهنمایی‌هایی در مورد مدیریت Secrets ارائه دهم یا در صورت دسترسی، به صورت مستقیم آن‌ها را پیکربندی کنم.

3.3. مستندسازی و گزارش‌دهی

- **تولید مستندات**: می‌توانم مستندات فنی، راهنماهای کاربری، نگهداری و توسعه را تولید یا به‌روزرسانی کنم.
- **تهیه گزارش‌ها**: می‌توانم گزارش‌های عملکرد، وضعیت پروژه، یا تحلیل‌های خاص را بر اساس داده‌های CI/CD تهیه کنم.

3.4. نحوه همکاری با عامل هوش مصنوعی

برای اینکه من بتوانم به طور موثر روی پروژه‌ها توسعه داشته باشم، نیاز به دستورالعمل‌های واضح و دسترسی‌های مناسب دارم:

- **دستورالعمل‌های واضح**: هر وظیفه توسعه باید به وضوح تعریف شده باشد، شامل هدف، محدوده، و معیارهای موفقیت.
- **دسترسی به کد**: برای ایجاد تغییرات، نیاز به دسترسی `write` به مخزن دارم.

- **دسترسی به Secrets (در صورت لزوم)**: برای کارهایی مانند پیکربندی استقرار یا تعامل با APIهای خارجی، ممکن است نیاز به دسترسی به Secrets مربوطه داشته باشم. این دسترسی باید با دقت و بر اساس اصل حداقل امتیاز اعطا شود.
- **بازبینی انسانی**: تمامی تغییرات پیشنهادی توسط من باید توسط یک توسعه‌دهنده انسانی بازبینی و تایید شوند (Code Review) قبل از ادغام در برنچ `main` .
- **تست‌های خودکار**: وجود تست‌های خودکار قوی در پروژه، به من کمک می‌کند تا از عدم ایجاد رگرسیون اطمینان حاصل کنم و کیفیت تغییرات را تضمین کنم.

نتیجه‌گیری

سیستم CI/CD پیاده‌سازی شده، یک چارچوب قدرتمند برای مدیریت چرخه حیات پروژه از افزودن پروژه‌های قدیمی تا دیپلوی MVP فراهم می‌کند. با استفاده از این چارچوب، می‌توانیم به طور موثر پروژه‌ها را مقیاس‌بندی کرده و حتی مشارکت عامل هوش مصنوعی را در فرآیندهای توسعه و نگهداری بهینه کنیم، که منجر به افزایش کارایی و کیفیت کلی می‌شود.