Credit Management API - Deployment Pipeline

---

1. Development Environment

- Developers work locally using Python virtual environments (`.venv`) and FastAPI.
- Database used: PostgreSQL (local or cloud-based like Aiven).
- Code is maintained in a Git repository on GitHub (`backend-intern-credits`).

---

2. Version Control & Branching

- **Main Branch:** Stable production-ready code.
- **Development Branch:** Active feature development.
- **Feature Branches:** Created for individual features or bug fixes.
- Pull requests (PRs) are reviewed and merged into the development branch.
- After QA approval, development branch merges into the main branch for deployment.

---

3. Continuous Integration (CI)

- CI tools like GitHub Actions or GitLab CI can be used.
- **Steps:**
    1. Trigger CI on every PR or push to `main`/`development`.

2. Install dependencies: `pip install -r requirements.txt`.
3. Run automated tests for API endpoints and database interactions.
4. Check code quality using `flake8` or `black`.

---

4. Dockerization

- Application is packaged in a Docker container for consistency across environments.
- **Dockerfile example:**
    - Base image: `python:3.12-slim`
    - Install dependencies
    - Copy application source code
    - Expose port `8000`
    - Set command: `uvicorn src.main:app --host 0.0.0.0 --port 8000`
- Docker Compose can be used to orchestrate **app + PostgreSQL** for local testing.

---

5. Staging Environment

- Before production, deploy on a staging server (e.g., cloud VM or Kubernetes cluster).
- **Steps:**
    1. Pull latest Docker image from registry.
    2. Run container with environment variables (`.env`) configured.

3. Connect to a staging PostgreSQL database.
4. QA team tests all API endpoints and background jobs.

---

6. Production Deployment

- Deploy the Docker container on a production server or cloud service (AWS, Azure, GCP, or Heroku).
- Use a **managed PostgreSQL database** for production.
- Ensure environment variables are set securely.
- Use **reverse proxy** (e.g., Nginx) to expose the FastAPI app on port 80/443 with HTTPS.
- Enable logging and monitoring for API performance and errors.

---

7. Background Jobs

- The daily credit update job is scheduled using APScheduler or an external task scheduler (e.g., cron job, Kubernetes CronJob).
- Ensure the scheduler runs continuously or restarts with the app.

---

8. Continuous Deployment (CD)

- Optionally, implement CD via GitHub Actions:
    - After merging to main, automatically build Docker image.

    - Push Docker image to container registry (Docker Hub, AWS ECR, etc.).
    - Pull and restart container on production server.

---

9. Monitoring & Maintenance

- Use monitoring tools like Prometheus, Grafana, or cloud provider dashboards.
- Track API uptime, response times, and database performance.
- Backup database regularly.
- Update dependencies and security patches periodically.

Summary

This deployment pipeline ensures **code stability**, **consistent environments**, and **automated testing** before production. By using Docker, CI/CD, and staging, the project can be reliably deployed and scaled for real-world usage.