GitHub — Flux de treball i pas a producció

> **Objectiu**: treballar en micro-tasques amb **feature branches**, tenir `main` com a **producció**, i desplegar al servidor de forma simple i repetible.

---

0) Config inicial del repositori

1. **Repo privat** (recomanat): crea'l a GitHub.
2. **Branca per defecte**: `main`.
3. **Protecció de** `main` (Settings → Branches → Add rule):
4. ✅Require a pull request before merging (1 review).
5. ✅Dismiss stale approvals on new commits.
6. ✅Require linear history.
7. (Opcional) Require status checks (si tens tests/CI).
8. ✅Restrict who can push to matching branches (només tu o mantenidors).
9. **Secrets**: no commitegis credencials. Mantén `.env` **fora** del repo, i publica `.env.example`.

---

1) Conventions de branques i commits

- `main` = **producció**. Tot el que hi entra s'ha de poder desplegar.
- **Feature branches**: `feat/<tema>`, `fix/<bug>`, `chore/<tasca>`.
- Ex.: `feat/ingesta-endpoint`, `fix/caddy-reload`, `chore/deploy-script`.
- Commits curts i clars:
- `feat(api): add POST /measurements`
- `fix(frontend): handle empty list`

---

2) Cicle de treball (micro-tasca)

1. **Crea issue** (o nota) amb el DoD (Definition of Done).
2. **Branca**:

```
git checkout -b feat/<tema>
```

3. **Desenvolupa i prova** (Codespaces o local):
4. Backend: `cd backend && npm install && npm run dev`
5. Frontend: obre `index.html` o serveix-lo des de l'Express/Caddy.
6. **Commit & push**:

```
git add -A
git commit -m "feat: <canvi>"
git push -u origin feat/<tema>
```

7. **Pull Request** → revisa (tu mateix si cal), comprova DoD i merge a `main`.

8. **Desplega** al servidor (3 opcions):
9. **Manual**: `ssh deploy@<IP> && cd ~/tecnolord-apps/tecnolord && ./scripts/deploy.sh`
10. **Hook local**: fer `git pull` al servidor i s'executa `post-merge` → recrea.
11. **GitHub Actions (push a main)**: workflow que fa SSH i executa el deploy (vegeu §4).

---

3) Secrets i configuració

- `.env.example` al repo amb claus **fictícies** i comentaris.
- `.env` **no** al repo (ja està al `.gitignore`). Al servidor, si cal, guarda'l a `~/tecnolord-apps/tecnolord/.env` i referencia'l des del `docker-compose.yml` amb `env_file:` o variables `environment:`.
- **Claus d'ingesta** i passwords només a `.env` del servidor o com a secrets de GitHub (si fas deploy via Actions).

---

4) Deploy automàtic amb GitHub Actions (opcional)

`.github/workflows/deploy.yml` (push a `main`):

```
name: Deploy
on:
  push:
    branches: [ main ]
jobs:
  ssh-deploy:
    runs-on: ubuntu-latest
    steps:
      - name: Deploy over SSH
        uses: appleboy/ssh-action@v1.0.3
        with:
          host: ${{ secrets.SSH_HOST }}
          username: ${{ secrets.SSH_USER }}
          key: ${{ secrets.SSH_KEY }} # o password: $
{{ secrets.SSH_PASSWORD }}
          script: |
            set -Eeuo pipefail
            cd ~/tecnolord-apps/tecnolord
            git fetch origin main
            git reset --hard origin/main
            docker compose build --pull
            docker compose up -d
            docker image prune -f || true
```

Secrets (Settings → Secrets → Actions): `SSH_HOST`, `SSH_USER`, `SSH_KEY` (o `SSH_PASSWORD`).

---

5) Script de desplegament (repositori del servidor)

Arrel del repo: `scripts/deploy.sh`

```
#!/usr/bin/env bash
set -Eeuo pipefail
cd "$(dirname "$0")/.."

branch="${1:-main}"

echo "[deploy] Pulling latest ($branch) ..."
git fetch origin "$branch"
git reset --hard "origin/$branch"

echo "[deploy] Building images (pull base if newer) ..."
docker compose build --pull

echo "[deploy] Recreating containers ..."
docker compose up -d

docker compose exec -T caddy caddy reload --config /etc/caddy/Caddyfile 2>/
dev/null || true

docker image prune -f || true
```

Hooks (opc.): `.git/hooks/post-merge` i `post-rewrite` que criden l'script.

---

6) Pas a producció (checklist curt)

- [ ] PR **aprovat** i mergejat a `main`.
- [ ] `./scripts/deploy.sh` executat (manual/hook/Actions).
- [ ] `docker compose ps` verd i `logs -f` sense errors.
- [ ] `GET /health` i `GET /api/ping` OK des del món.
- [ ] (Si hi ha canvis de schema) migracions aplicades.
- [ ] Tag opcional: `git tag -a v0.1.0 -m "Fase 1 estable" && git push origin v0.1.0`.

---

7) Rollback ràpid

- **Revertir a l'últim tag estable**:

```
git fetch --tags
git checkout -f v0.1.0
docker compose up -d --build
```

- O **revertir commits** i redeploy: `git revert <hash>`.

---

8) Bones pràctiques

- No treballis directament al clone de prod: considera'l **només per desplegar**.
- Commits i PR **petits** (micro-tasques) i sempre amb DoD.
- Activa **Dependabot** (Security → Code security) per alertes de deps.
- Evita secrets al codi: usa `.env`, Secrets de GitHub i variables d'entorn al compose.

---

9) Cheatsheet Git

```
# nova branca de feina
git checkout -b feat/endpoint-x

# publicar canvis
git add -A && git commit -m "feat(api): endpoint x" && git push -u origin
HEAD

# canviar de branca, actualitzar i desplegar
git checkout main && git pull --rebase
./scripts/deploy.sh

# tags de release
git tag -a v0.1.0 -m "release" && git push origin v0.1.0
```