

Contents

Mise en situation — Présenter votre pipeline TaskFlow	1
Contexte	1
Ce que vous devez livrer	1
Déroulé de votre passage (8 min)	2
Trame suggérée (5 min)	3
Plan B si la démo live échoue	4
Questions types du jury	4
Lien avec votre dossier professionnel DWWM	5

Mise en situation — Présenter votre pipeline TaskFlow

Évaluation certificative · J4, 16h15 – 16h55 · 5 min de présentation + 3 min de questions
Grille de notation : [grille-evaluation.md](#) (100 points)

Contexte

Vous venez de rejoindre l'équipe qui développe **TaskFlow**, un gestionnaire de tâches (API Node 20 + Express + PostgreSQL 16, front Vite servi par nginx). Pendant quatre jours, vous avez transformé une application qui ne tournait que « sur votre machine » en une application :

- **conteneurisée** (Docker, Docker Compose) ;
- **vérifiée automatiquement** à chaque Pull Request (GitHub Actions) ;
- **publiée** sous forme d'images sur [ghcr.io](#) ;
- **déployée automatiquement** dans une « production » (stack Compose taskflow-prod, dossier ~/taskflow-prod/ de votre poste, qui joue le rôle du serveur), derrière Nginx en HTTPS ;
- **supervisée** (healthcheck /health, logs, Uptime Kuma).

Votre responsable technique vous demande de **présenter ce pipeline** à l'équipe et de **justifier vos choix**, comme vous le feriez lors d'une revue de fin de sprint ou d'un entretien d'embauche.

Ce que vous devez livrer

Avant **16h00**, envoyez au formateur l'**URL de votre dépôt** <https://github.com/<votre-compte>/taskflow> (public, ou formateur invité comme collaborateur). Le dépôt doit contenir :

```
taskflow/
├── api/
│   ├── Dockerfile           # node:20-alpine, npm ci --omit=dev, USER node, HEALTHCHECK
│   ├── .dockerignore
│   └── src/ tests/ ...      # code du starter
└── front/
```

```

├── Dockerfile          # multi-stage : node:20-alpine → nginx:1.27-alpine
│   ├── nginx.conf      # sert dist/, proxy /api/ et /health → api:3000
│   └── .dockerignore
├── compose.yaml        # dev : db + api + front (port 8080), volume, healthchecks
├── compose.prod.yaml   # « prod » (name: taskflow-prod) : db, api, front (images ghcr.io)
├── api/                # front:${IMAGE_TAG}) + proxy nginx (seul à publier 80/443)
│   ├── .env.example    # variables attendues, SANS vraies valeurs
│   ├── .github/workflows/
│   │   ├── ci.yml      # PR + push main : lint, tests, build
│   │   └── release.yml  # push main : images ghcr.io → deploy (self-hosted, lab) → smoke test
│   └── deploy/
│       ├── nginx/taskflow.conf # conteneur proxy : 80 → 301 https, 443 → proxy_pass http://f
│       └── gen-cert.sh        # certificat auto-signé taskflow.localhost (dans ~/taskflow-
├── prod/certs/
│   ├── deploy.sh
│   └── backup.sh
├── docs/
│   ├── ARCHITECTURE.md    # schéma, ports, flux
│   └── DEPLOIEMENT.md     # procédure pas à pas + rollback
└── README.md              # lancement local, badge CI, liens vers docs/

```

Et, **en état de marche sur votre poste** au moment de passer :

- la stack taskflow-prod démarrée : `docker compose -p taskflow-prod ps` montre db, api, front et proxy « Up » ;
- `~/taskflow-prod/` contient `.env` (`chmod 600`), `certs/` et `backups/` — et **aucun** de ces éléments n'est dans votre dépôt ;
- `https://taskflow.localhost` qui affiche TaskFlow (certificat auto-signé accepté) ;
- `curl -kfsS https://taskflow.localhost/health` qui répond `{"status":"ok","db":"up",...}` ;
- le runner lab-runner en état **Idle** dans *Settings* → *Actions* → *Runners*.

Rappel de la règle de validation : au moins **50/100** et les **5 premiers critères** (Dockerfile, Compose, dépôt GitHub, workflow, déploiement automatique) **au moins partiellement atteints**. Une brique manquante ? Présentez quand même ce qui fonctionne et expliquez honnêtement ce qui manque et comment vous le termineriez : c'est valorisé au critère 7.

Déroulé de votre passage (8 min)

Temps	Qui	Quoi
0:00 – 5:00	Vous	Présentation + démonstration (écran partagé ou vidéoprojecteur)
5:00 – 8:00	Jury	2 à 3 questions sur vos choix techniques

Le formateur vous prévient à 4 min. À 5 min, la présentation s'arrête, même si vous n'avez pas fini : **répétez une fois** avant de passer.

Trame suggérée (5 min)

Minute	Contenu	Conseil
0:00 – 0:45	Le problème : pourquoi un pipeline ? (« ça marche sur ma machine », déploiement manuel risqué) et vue d'ensemble : montrez le schéma de docs/ARCHITECTURE.md	Une phrase par brique, pas de détail technique encore
0:45 – 1:45	Conteneurs : les deux Dockerfile (un point fort chacun : multi-stage, USER node), compose.yaml (volume, healthcheck)	Montrez le fichier, pointez 2-3 lignes, ne lisez pas tout
1:45 – 4:15	Démo live : 1. modifiez le titre de la page dans front/index.html sur une branche ; 2. poussez, ouvrez la PR ; 3. montrez le CI qui passe au vert (checks requis) ; 4. mergez ; 5. montrez release.yml : build & push → deploy → smoke test ; 6. rechargez https://taskflow.localhost : le nouveau titre est en ligne, /health affiche la nouvelle version	Lancez la PR au tout début de la démo : le CI prend 1 à 2 min, commentez les fichiers pendant qu'il tourne
4:15 – 4:45	Exploitation : où regarder en cas de problème (docker compose logs, /health, Uptime Kuma), sauvegarde backup.sh, rollback	Citez l'incident de l'étape 5 si vous l'avez résolu
4:45 – 5:00	Bilan : ce que vous amélioreriez (déploiement sur un vrai serveur distant, Let's Encrypt avec un vrai nom de domaine, environnement de staging, monitoring Prometheus...)	Montrer qu'on connaît les limites est un point fort

Plan B si la démo live échoue

Une démo peut échouer (réseau, runner hors ligne, quota GitHub). **Préparez avant 16h00** dans un dossier captures/ de votre dépôt (ou sur votre bureau) :

1. une PR avec les checks CI **verts** ;
2. l'onglet *Actions* montrant un run de `release.yml` vert (jobs build → deploy) ;
3. le log du smoke test ("status": "ok") ;
4. le navigateur sur <https://taskflow.localhost> avec la version déployée ;
5. *Settings* → *Branches* (protection de main) et *Packages* (images ghcr.io).

Si la démo échoue : dites-le calmement, expliquez **ce que vous auriez vu**, basculez sur les captures et, si vous avez le temps, une piste de diagnostic (« le runner est hors ligne, je le vérifierais avec `docker compose logs` dans `ressources/lab/runner` »). Savoir réagir à un incident fait partie du métier.

Questions types du jury

Le jury pose 2 à 3 questions parmi les suivantes (ou proches). Entraînez-vous à répondre en **3 phrases** : quoi, pourquoi, et une limite ou alternative.

Docker 1. Pourquoi un Dockerfile **multi-stage** pour le front ? Que contient l'image finale ? 2. Pourquoi copier `package*.json` et lancer `npm ci` **avant** de copier le reste du code ? 3. Pourquoi `USER node` dans l'image de l'API ? Que risque-t-on en root ? 4. Différence entre une image et un conteneur ?

Compose et données 5. Pourquoi un **volume** pour PostgreSQL ? Que se passe-t-il sans ? 6. À quoi sert le `healthcheck` de db et le `depends_on: condition: service_healthy` ? 7. Pourquoi l'API se connecte-t-elle à `db:5432` et pas à `localhost:5432` ? 8. Où sont vos mots de passe ? Pourquoi `.env` n'est-il pas dans Git ?

Git et collaboration 9. Pourquoi protéger main ? Qu'est-ce que ça empêche concrètement ? 10. Merge ou rebase : lequel utilisez-vous, quand, et quelle est la règle à ne pas enfreindre ? 11. Que signifie votre tag `v1.0.0` ? Que deviendrait-il après une correction de bug ?

CI/CD 12. Quelle différence entre **livraison continue** et **déploiement continu** ? Lequel avez-vous mis en place ? 13. Pourquoi le job `deploy` tourne-t-il sur un runner **self-hosted** et pas sur `ubuntu-latest` ? Quel risque y a-t-il à lui monter le socket Docker ? 14. Comment le workflow s'authentifie-t-il sur `ghcr.io` sans mot de passe en clair ? Où mettriez-vous une clé SSH si vous déployiez sur un vrai serveur distant ? 15. Pourquoi taguer les images avec le **SHA du commit** en plus de `latest` ? 16. Que se passe-t-il si un test échoue dans une PR ?

Déploiement et exploitation 17. À quoi sert le conteneur proxy (Nginx) ? Pourquoi est-il le seul service de `compose.prod.yml` à publier des ports ? 18. Comment feriez-vous un **rollback** si la dernière version casse la prod ? 19. Que vérifie votre smoke test ? Que renvoie `/health` si la base est arrêtée ? 20. Le site affiche 502 : quelles sont vos trois premières commandes ? 21. Sur un vrai serveur avec un nom de domaine, que changeriez-vous (connexion, certificat HTTPS, job de déploiement) ? 22. Votre sauvegarde fonctionne-t-elle ? Comment le savez-vous ?

Lien avec votre dossier professionnel DWWM

Ce pipeline est une **preuve concrète** pour la compétence « Préparer et exécuter le déploiement d'une application web » (activité « Développer la partie back-end d'une application web ou web mobile »). Pour votre dossier professionnel, conservez :

- l'URL du dépôt et une capture de l'onglet *Actions* (pipeline vert) ;
- le schéma de docs/ARCHITECTURE.md ;
- un extrait commenté de release.yml et de compose.prod.yaml ;
- docs/DEPLOIEMENT.md comme exemple de documentation technique ;
- le post-mortem de docs/INCIDENTS.md (démarche de résolution de problème).

Décrivez-les avec vos mots : contexte, ce que **vous** avez fait, les difficultés rencontrées et comment vous les avez surmontées.