Apertura: jueves, 1 de enero de 2026, 00:00
Cierre: miércoles, 28 de enero de 2026, 23:59

Modalidad: Grupos de 2 a 3 estudiantes
Duración de desarrollo: 1 semana
Contexto del sistema: Aplicación de **Compras/Ventas** (Laravel 12 + Vue 3 + PostgreSQL)
Objetivo central: Implementar **entrega continua reproducible** a staging y producción con **promoción de artefacto, approvals, despliegue atómico por symlink y rollback**, además de observabilidad base.

1) Tareas obligatorias

A. CD a Staging (Día 3)

- Pipeline debe activarse en `merge a main` o `push a main`.
- CI construye **una sola vez** el artefacto de la app.
- Despliegue a staging vía **SSH + rsync + desempaquetado** en nueva carpeta de release.
- En staging:
  - Limpiar cachés (`optimize:clear`)
  - Ejecutar migraciones `--force`
  - Cambiar symlink `current` de forma atómica
- Verificación post-deploy en staging (`/api/health` → 200 OK)

B. CD a Producción (Día 4)

- El pipeline debe quedar **en pausa hasta aprobación**.
- Merge permitido solo si CI staging fue verde.
- Aprobación humana obligatoria (Release Manager o reviewer asignado a producción).
- En producción:
  - Despliegue en **carpeta nueva dentro de /releases**
  - **Switch symlink atómico** (`ln -sfn`)
  - Rollback preparado por versión anterior
  - Logs con `release context`
- Verificación post-deploy producción (`/api/health` → 200 OK)

C. Secrets y segregación

?

- Configurar **GitHub Environments** para `staging` y `production`.

- Secrets nunca en repo, nunca en logs.

D. Observabilidad base

- Logs backend deben incluir:
  - versión (`release`)
  - ambiente (`staging` o `production`)
  - `user_id` en operaciones de compra/venta
- El pipeline debe mostrar trazabilidad del deploy.

2) Reglas de despliegue profesional que deben cumplirse

- Build una sola vez por release.
- Promoción por artefacto, no recompilar por ambiente.
- Despliegue atómico por symlink.
- Rollback inmediato sin reinstalar dependencias.
- Accountability por aprobador.
- Health checks como gates.

3) Estructura esperada en el servidor

Staging y producción deben terminar con:

```
/var/www/comprasventas /releases/<release_id> # release desplegado
/shared/.env # secrets segregados /current -> /releases/<release_id>
```

4) Ejecuciones locales obligatorias antes de PR/Merge

```
composer install --no-dev ./vendor/bin/pint --test ./vendor/bin/phpstan
analyse php artisan test npm ci npm run lint npm run test npm run build
```

5) Crear release notes mínimo (ligado al tag)

Ejemplo:

```
git tag -a v2.0.0 -m "release: CD environments + atomic deploy + health gates"
git push origin v2.0.0
```

> **Regla:** Tags deben ser semánticos (`vX.Y.Z`) y trazables.

6) Pipeline CI/CD mínimo esperado (entregable evaluado)

Crear en el repo:

`.github/workflows/cd-staging-prod.yml`

Ejemplo orientativo:

```
name: CD - Staging + Production on: push: branches: ["main"] tags: ["v*"]
workflow_dispatch: jobs: build: runs-on: ubuntu-latest steps: - uses:
actions/checkout@v4 - run: composer install --no-dev --prefer-dist --no-
interaction - run: npm ci - run: npm run build - run: date +%Y%m%d_%H%M%S >
RELEASE_ID - run: echo "${GITHUB_SHA}" >> RELEASE_ID - uses: actions/upload-
artifact@v4 with: name: release-${{ github.ref_name }} path: | public/build
vendor RELEASE_ID deploy_staging: if: github.ref == 'refs/heads/main' runs-on:
ubuntu-latest environment: staging needs: [build] steps: - uses:
actions/checkout@v4 - run: | RELEASE="${{ github.sha }}" ssh ${{
secrets.STAGING_USER }}@${{ secrets.STAGING_HOST }} "mkdir -p
/var/www/comprasventas/releases/$RELEASE" rsync -avz ./ ${{
secrets.STAGING_USER }}@${{ secrets.STAGING_HOST
}}:/var/www/comprasventas/releases/$RELEASE ssh ${{ secrets.STAGING_USER
}}@${{ secrets.STAGING_HOST }} " cd /var/www/comprasventas/releases/$RELEASE
&& php artisan optimize:clear || true && php artisan migrate --force && ln -
sfn /var/www/comprasventas/releases/$RELEASE /var/www/comprasventas/current "
- run: curl -f "${{ secrets.STAGING_APP_URL }}/api/health" deploy_production:
if: startsWith(github.ref, 'refs/tags/v') || github.event_name ==
'workflow_dispatch' runs-on: ubuntu-latest environment: production needs:
[deploy_staging] steps: - uses: actions/checkout@v4 - run: | RELEASE="${{
github.ref_name:-manual }}" ssh ${{ secrets.PROD_USER }}@${{ secrets.PROD_HOST
}} "mkdir -p /var/www/comprasventas/releases/$RELEASE" rsync -avz ./ ${{
secrets.PROD_USER }}@${{ secrets.PROD_HOST
}}:/var/www/comprasventas/releases/$RELEASE ssh ${{ secrets.PROD_USER }}@${{
secrets.PROD_HOST }} " cd /var/www/comprasventas/releases/$RELEASE && php
artisan optimize:clear || true && php artisan migrate --force && ln -sfn
/var/www/comprasventas/releases/$RELEASE /var/www/comprasventas/current " -
run: curl -f "${{ secrets.PROD_APP_URL }}/api/health" rollback_staging: if:
failure() runs-on: ubuntu-latest needs: [deploy_staging] steps: - run: | ssh
${{ secrets.STAGING_USER }}@${{ secrets.STAGING_HOST }} " PREV=\$(ls -1dt
/var/www/comprasventas/releases/*/ | sed -n '2p' | tr -d '/') && [ -n
\"\$PREV\" ] && ln -sfn \$PREV /var/www/comprasventas/current "
```

```
rollback_production: if: failure() runs-on: ubuntu-latest needs:
[deploy_production] steps: - run: | ssh ${{ secrets.PROD_USER }}@${{
secrets.PROD_HOST }} " PREV=\$(ls -1dt /var/www/comprasventas/releases/*/ |
sed -n '2p' | tr -d '/') && [ -n \"\$PREV\" ] && ln -sfn \$PREV
/var/www/comprasventas/current "
```

7) Evidencias a entregar

Cada grupo debe subir un documento (PDF o reporte) con:

- Pipeline run staging (link o capturas).
- Pipeline run producción (captura de pausa + aprobación).
- Captura de symlink `current` apuntando a release.
- Captura de rollback (si aplica).
- Captura del endpoint `/api/health` en staging y producción.
- Capturas de logs backend con `contexto de release`.

8) Simulación obligatoria de fallo + rollback

8.1 Simular fallo en staging

Introduce error deliberado:

```
git checkout -b feature/fallo-staging echo "ERROR_INTENCIONAL" >>
routes/api.php git add . git commit -m "ci: break staging intentionally" git
push origin feature/fallo-staging
```

Pipeline debe fallar en staging.

8.2 Rollback

Luego corregir:

```
git checkout main git revert HEAD --no-edit git push origin main
```

Verificar en servidor staging:

```
ssh deploy@staging_host "readlink -f /var/www/comprasventas/current"
```

Debe apuntar a versión anterior estable.

9) Criterios de evaluación

Criterio	Validación
Build único	Evidencia RELEASE_ID y artifact único
Secrets segregados	No aparecen en repo/logs
Switch atómico	Symlink correcto
Approvals	Captura del paso pausado y aprobación
Rollback	Rollback ejecutado y verificable
Health checks	200 OK en ambos ambientes
Observabilidad	Logs contienen `release`, `env`, `user_id`

Mensaje clave para maestrantes

> *Un release sin verificación es un riesgo.*
> *Un deploy sin symlink atómico es incertidumbre.*
> *Un pipeline sin approvals en producción es inmadurez.*
> *Un equipo que no puede recuperarse rápido no gobierna el sistema, el sistema lo gobierna a él.*

Agregar entrega

Estado de la entrega

Estado de la entrega	Todavía no se han realizado envíos
Estado de la calificación	Sin calificar
Tiempo restante	21 horas 42 minutos restante
Última modificación	-
Comentarios de la entrega	› Comentarios (0)

◄ Presentación día 4

Ir a...

Presentación día 5 ►