

Deploy repetabil cu verificare și revenire

Fișă publică de proiect · exerciții și criterii de acceptare

Pregătești două release-uri ale unei aplicații și documentezi comutarea, health check-ul și rollback-ul. Migrarea datelor este tratată separat de schimbarea codului.

Livrabile

- Plan de deploy
- Config exemplu
- Build
- Migrare compatibilă
- Runbook rollback

Criterii de acceptare

- Secretele nu intră în bundle
- Health check-ul verifică funcția principală
- Workers folosesc release-ul corect

Manifestul unei versiuni publicabile

Un deploy trebuie să identifice exact codul, schema și asseturile care merg împreună. Nu publica un director de lucru cu modificări neinventariate. Construiește un manifest și verifică înainte dependențele, testele și compatibilitatea migrărilor. Schimbarea legăturii către versiunea activă este doar un pas; workerii, cache-urile și fișierele persistente trebuie incluse în plan. Un răspuns HTTP bun nu dovedește că joburile rulează corect.

Exemplu de lucru

Manifest: commit/identificator versiune; PHP; dependențe lock; hash asseturi; migrații; variabile noi; verificări; versiune de revenire.
Ordine: pregătire → verificare → backup → migrare compatibilă → activare → cache/worker → verificare publică → monitorizare.
Persistente: uploaduri, sesiuni și date, separate de codul versiunii.

Interpretarea rezultatului

Versiunea anterioară nu trebuie suprascrisă înainte de confirmarea celei noi. Un director de release trebuie să poată fi asociat manifestului, fără a include secrete în raport. Nu introduce o migrare incompatibilă într-un deploy care pretinde revenire instantanee.

Exercițiu de consolidare

Completează manifestul pentru o schimbare care adaugă un câmp opțional și o foaie CSS nouă.

1. Anatomia unui deploy

Completează manifestul pentru o schimbare care adaugă un câmp opțional și o foaie CSS nouă.

Verificare orientativă

Versiunea anterioară nu trebuie suprascrisă înainte de confirmarea celei noi. Un director de release trebuie să poată fi asociat manifestului, fără a include secrete în raport. Nu introduce o migrare incompatibilă într-un deploy care pretinde revenire instantanee.

De evitat

Succesul copiei fișierelor nu este succesul aplicației.

Secrete diferite între build și rulare

Un secret necesar serverului nu trebuie introdus în bundle-ul browserului. Unele sisteme frontend includ variabilele de build în fișiere publice, de aceea numele și mecanismul trebuie înțelese. Pentru Laravel, configurația cache-uită poate păstra valorile citite la construirea cache-ului. Nu schimbă .env și presupune că toate procesele au preluat imediat noua valoare. Păstrează un inventar al variabilelor fără a copia valorile secrete.

Exemplu de lucru

```
Variabilă;Loc;Publică?  
APP_URL;server;valoare publică  
DB_PASSWORD;server;nu  
Cheie API secretă;server;nu  
URL API public;browser;da  
Verificări: bundle fără chei; .env inaccesibil web; cache configurație actualizat; workerii  
folosesc versiunea nouă.
```

Interpretarea rezultatului

Un prefix de variabilă frontend nu este o protecție; adesea indică exact includerea ei în client. Pentru rotirea unui secret, coordonează emiterea, publicarea și revocarea, astfel încât procesele vechi să nu eșueze necontrolat. Nu afișa secretul pentru a demonstra că există.

Exercițiu de consolidare

Describe o verificare care confirmă configurarea unei chei printr-un rezultat controlat, fără a imprima cheia în log.

2. Variabile de mediu și secrete

Describe o verificare care confirmă configurarea unei chei printr-un rezultat controlat, fără a imprima cheia în log.

Verificare orientativă

Un prefix de variabilă frontend nu este o protecție; adesea indică exact includerea ei în client. Pentru rotirea unui secret, coordonează emiterea, publicarea și revocarea, astfel încât procesele vechi să nu eșueze necontrolat. Nu afișa secretul pentru a demonstra că există.

De evitat

Variabilele cu prefix public pot ajunge în bundle-ul browserului.

Build reproductibil și manifest de asseturi

Instalează dependențele din lockfile în mediul de build și păstrează rezultatul asociat versiunii. Nu edita manual fișierul minificat pentru a corecta sursa. Manifestul leagă intrările de numele cu hash, iar template-urile trebuie să folosească aceeași versiune. Păstrează pentru o perioadă asseturile vechi dacă pagini cache-uite le pot solicita. Un build reușit sintactic trebuie urmat de verificarea încărcării CSS și JavaScript în browser.

Exemplu de lucru

Pași de laborator: npm ci → npm run build → inspectează public/build/manifest.json → verifică fișierele referite → servește pagina → verifică Network și consolă.
Dovezi: versiune lockfile, hash manifest, status 200 pentru asseturi.
Nu combina: manifest nou cu fișiere din build vechi.

Interpretarea rezultatului

Un HTML 200 cu CSS 404 este un deploy defect chiar dacă aplicația răspunde. Cache busting prin hash ajută numai dacă referințele și fișierele sunt publicate coerent. Pentru builduri care citesc variabile, documentează ce valori publice au influențat rezultatul.

Exercițiu de consolidare

Simulează un manifest care indică un fișier absent și descrie verificarea care ar opri publicarea.

3. Build de asset-uri

Simulează un manifest care indică un fișier absent și descrie verificarea care ar opri publicarea.

Verificare orientativă

Un HTML 200 cu CSS 404 este un deploy defect chiar dacă aplicația răspunde. Cache busting prin hash ajută numai dacă referințele și fișierele sunt publicate coerent. Pentru builduri care citesc variabile, documentează ce valori publice au influențat rezultatul.

De evitat

Cache-ul browserului poate ascunde un build lipsă.

Migrare în producție cu criterii de oprire

Înainte unei migrări, verifică volumul, lockurile posibile și compatibilitatea codului vechi. O comandă fără eroare nu dovedește că timpul de blocare a fost acceptabil. Folosește o copie reprezentativă pentru evaluare și planifică pașii mari separat. Backupul și restaurarea trebuie să fie disponibile înainte, nu după apariția problemei. Nu rula migrate:fresh sau alte operații de reset într-un mediu cu date reale.

Exemplu de lucru

Plan: inventar migrare → test pe copie → durată și lockuri → backup verificat → execuție controlată → verificare schemă și date → monitorizare.
Oprire: conexiune DB greșită, backup absent, SQL neașteptat, lock excesiv.
Comandă uzuală de producție: php artisan migrate --force, numai după verificări.

Interpretarea rezultatului

--force confirmă intenția operațională, nu validează designul migrării. Un rollback poate șterge date noi dacă metoda down o face. Definește separat revenirea codului și restaurarea datelor, inclusiv consecințele pierderii operațiilor dintre backup și incident.

Exercițiu de consolidare

Scrie două interogări de integritate pentru o coloană nouă care trebuie să devină unică și obligatorie.

4. Migrări în producție

Scrie două interogări de integritate pentru o coloană nouă care trebuie să devină unică și obligatorie.

Verificare orientativă

--force confirmă intenția operațională, nu validează designul migrării. Un rollback poate șterge date noi dacă metoda down o face. Definește separat revenirea codului și restaurarea datelor, inclusiv consecințele pierderii operațiilor dintre backup și incident.

De evitat

Migrarea reversibilă ca SQL nu garantează recuperarea datelor pierdute.

Continuitatea serviciului ca obiectiv verificabil

„Fără downtime” este un obiectiv care depinde de infrastructură, compatibilitate și trafic, nu o proprietate a unei singure comenzi. Publicarea atomică a codului nu elimină o migrare blocantă sau un restart de worker. Pentru laborator, definește ce cereri trebuie să rămână disponibile și ce degradare este acceptată. Testează cu cereri repetate în timpul schimbării și păstrează rezultatele. Nu promite disponibilitate perfectă fără măsurători și limite.

Exemplu de lucru

Scenariu: versiunea A și B acceptă schema extinsă; asseturile ambelor există; uploadurile sunt comune; sesiunea rămâne compatibilă.

Test: cereri repetate la pagină și endpoint de citire înainte, în timpul și după activare.

Măsuri: statusuri, latență, erori, joburi rămase.

Limite: nu dovedește comportamentul la orice trafic.

Interpretarea rezultatului

O fereastră fără erori în laborator nu garantează aceeași experiență în producție. Notează condițiile și diferențele. Dacă schema este incompatibilă, alege o etapizare sau o fereastră de mentenanță declarată în loc să ascunzi riscul sub un nume de deploy.

Exercițiu de consolidare

Identifică o schimbare care face versiunea veche incompatibilă și rescrie planul în două etape.

5. Deploy fără downtime

Identifică o schimbare care face versiunea veche incompatibilă și rescrie planul în două etape.

Verificare orientativă

O fereastră fără erori în laborator nu garantează aceeași experiență în producție. Notează condițiile și diferențele. Dacă schema este incompatibilă, alege o etapizare sau o fereastră de mentenanță declarată în loc să ascunzi riscul sub un nume de deploy.

De evitat

Queue workers și cache-urile pot continua să folosească versiunea veche.

Rollback cu date și workerii luați în calcul

Revenirea la codul anterior trebuie să fie posibilă și compatibilă cu datele curente. Dacă noua versiune a scris un format diferit, simpla schimbare a directorului activ poate eșua. Păstrează identificatorul release-ului, starea migrărilor și verificările de sănătate. Definește pragurile care declanșează revenirea și cine o execută. După rollback, verifică din nou fluxurile, nu presupune că versiunea veche garantează automat succesul.

Exemplu de lucru

Declanșator: erori peste pragul ales sau flux critic defect.

Pași: oprește publicarea; confirmă schema compatibilă; activează versiunea anterioară; actualizează procesele persistente; verifică pagini, formulare și joburi; păstrează loguri.

Date: nu restaura automat backupul dacă ar pierde operații valide fără evaluare.

Interpretarea rezultatului

Rollbackul codului și restaurarea DB sunt operații diferite. A doua poate pierde date și cere o analiză precisă. Monitorizarea trebuie să urmărească rezultatul utilizatorului, nu doar procesul activ. Păstrează incidentul și corectura pentru următorul deploy.

Exercițiu de consolidare

Construiește o verificare post-rollback pentru o comandă creată între cele două versiuni.

6. Rollback și monitorizare

Construiește o verificare post-rollback pentru o comandă creată între cele două versiuni.

Verificare orientativă

Rollbackul codului și restaurarea DB sunt operații diferite. A doua poate pierde date și cere o analiză precisă. Monitorizarea trebuie să urmărească rezultatul utilizatorului, nu doar procesul activ. Păstrează incidentul și corectura pentru următorul deploy.

De evitat

Rollback-ul codului nu inversează automat toate schimbările bazei.

Atelier final: Două release-uri și un rollback verificat

Timp de lucru estimat: 40 de minute. Folosește datele de laborator de mai jos și salvează rezultatul separat de orice proiect real.

Date de pornire

Release A afișează versiunea A; B afișează B și adaugă un câmp opțional. Datele create de A trebuie să rămână citibile. Health check-ul include o citire din baza de test.

1. Pregătire — 5 minute

Extrage intrările și condițiile din enunț. Notează ce rezultat trebuie produs și ce informații nu sunt confirmate. Pregătește fișierele sau tabelul în care vei păstra dovada lucrului.

2. Construire — 15 minute

Pregătește buildul, migrarea compatibilă și comutarea release-ului. Simulează eșecul health check-ului și revino la A fără să ștergi date.

3. Variație și verificare — 15 minute

Introdu un job în așteptare înainte de comutare. Arată cum verifici ce versiune îl execută și cum previi pierderea lui.

4. Autoevaluare — 5 minute

Compară rezultatul cu reperele de mai jos. Înregistrează cel puțin o eroare identificată sau un caz-limită verificat, plus corectura ori explicația comportamentului observat.

Repere de rezolvare

Rollbackul codului readuce A, iar datele existente rămân valide. Migrarea nu elimină câmpuri necesare vechiului release. Workers și cache-urile sunt tratate explicit în runbook.

Ce predai

- Plan de deploy
- Config exemplu
- Build
- Migrare compatibilă
- Runbook rollback

Criterii de acceptare

- Secretele nu intră în bundle
- Health check-ul verifică funcția principală
- Workers folosesc release-ul corect

Păstrează o versiune inițială și una revizuită, cu explicația diferențelor. Dacă un criteriu nu este îndeplinit, revino la lecția relevantă și repetă verificarea. Durata este orientativă; instalarea mediului și aprofundarea pot necesita timp suplimentar.

Registrul de lucru

Pentru fiecare etapă notează intrarea, rezultatul, dovada verificării, eroarea găsită și corectura. Nu introduce date personale sau secrete în fișa partajată.

Documentație și surse

- Documentația Laravel 11 — versiunea aplicației demonstrative — Laravel
- Ubuntu Server documentation — Canonical

<https://laravel.com/docs/11.x>

<https://documentation.ubuntu.com/server/>