Trailgun

PACCHETTO DI CONFORMITÀ · DOCUMENTO 20

VERSIONE 2.0 · DICEMBRE 2026

Procedura di gestione dei cambiamenti

Come una modifica al codice, alla configurazione o all'infrastruttura arriva in production: dal branch alla pull request, attraverso development e staging, con la verifica automatica obbligatoria, la copia dei dati prima di ogni rilascio e il ritorno alla versione precedente.

AMBIENTI
development · staging · production

VERIFICA
obbligatoria per ogni rilascio

COPIA PRIMA DEL RILASCIO
sempre, conservata 14 giorni

MANUTENZIONE
preavviso di 48 ore

PROCEDURA 20 · PROC-20

Gestione dei cambiamenti

Ogni modifica della piattaforma Trailgun segue la stessa strada, dal branch di lavoro a production, e nessuna arriva su una macchina senza la verifica automatica superata.

CODICE	TITOLARE	APPROVATA DA	VERSIONE
PROC-20	Responsabile della sicurezza (Franco Puppo, CEO)	Organo amministrativo, [data della delibera]	2.0 · stato al 01/12/2026

REVISIONE	DESTINATARI
annuale e dopo ogni incidente causato da un cambiamento	chi sviluppa, rilascia o configura la piattaforma; ICT risk e audit dei Clienti

DOCUMENTI COLLEGATI

politiche P05 e P06; documento 30, Backup e ripristino; documento 24, Vulnerabilità e segnalazione responsabile; documento 29, Valutazione dei fornitori; documento 41, Registro dei rischi ICT; documento 45, Registro dei cambiamenti; documento 55, Accordo sul livello di servizio

1 Scopo e ambito

1.1 La procedura stabilisce come si propone, verifica, approva, rilascia, registra e, se serve, annulla un cambiamento della piattaforma Trailgun, perché nessuna modifica comprometta la disponibilità, l'integrità o la riservatezza del servizio e dei dati dei Clienti.

1.2 Si applica a: codice dell'applicazione, dell'interfaccia e del sito; dipendenze e immagini di base dei container; infrastruttura descritta come codice (AWS CDK); configurazione e segreti degli ambienti; schema del database; impostazioni della piattaforma che il backoffice riserva agli amministratori di Trailgun (modelli degli agenti, testi dei prompt, motore di riconoscimento del testo).

1.3 Non si applica alle impostazioni che ogni studio sceglie per sé dall'Organizzazione o dalle Impostazioni, che non cambiano la piattaforma.

1.4 **Le novità sono opzioni.** Ogni passo nuovo nei flussi di lavoro si attiva e si disattiva dalle Impostazioni e dal modulo della singola pratica; spento, il flusso resta identico a prima, con lo stesso codice e gli stessi prompt. La scelta si congela sulla pratica alla creazione: un rilascio non cambia il modo in cui una pratica già avviata viene lavorata.

2 Ruoli

- **Sviluppatore**: prepara il cambiamento su un branch, apre la pull request, descrive rischio e piano di ritorno, segue la verifica.
- **Responsabile della sicurezza**: approva i cambiamenti significativi e urgenti, autorizza i rilasci manuali, tiene il registro dei cambiamenti, riesamina la procedura.
- **Organo amministrativo**: è informato dei cambiamenti significativi che toccano fornitori, luoghi del trattamento o impegni contrattuali.
- **Integrazione continua** (GitHub Actions): esegue la verifica e il rilascio; è un controllo automatico che nessun ruolo può saltare.

3 Classi di cambiamento

Ogni cambiamento ha una classe, scritta nella pull request. La classe decide chi approva e che cosa si fa in più, non la verifica: la verifica è la stessa per tutti.

CLASSE	ESEMPI	APPROVAZIONE	IN PIÙ
Standard	funzioni, correzioni, testi, aggiornamenti di dipendenze, nuove opzioni spente di default	unione della pull request con la verifica superata	registro dei cambiamenti
Significativo	architettura, flussi dei dati, regioni, nuovo fornitore o subfornitore, controlli di sicurezza, cifratura, autenticazione, migrazioni che riscrivono dati esistenti	Responsabile della sicurezza, prima dell'unione	valutazione nel registro dei rischi (41); per un subfornitore, preavviso ai Clienti (documento 29); aggiornamento della documentazione del pacchetto
Urgente	correzione di una vulnerabilità critica o sfruttata, contenimento di un incidente	Responsabile della sicurezza, anche a voce, annotata subito	stessa verifica; può saltare la sosta in staging; registrazione a cose fatte con il motivo

4 Dal branch a production, passo per passo

1 **Branch.** Ogni modifica nasce su un branch di lavoro dedicato, con nome in inglese, a partire da `development`. Nessuno lavora direttamente sui branch di ambiente `development`, `staging` e `production`, che sono protetti: accettano solo pull request con la verifica superata.

2 **Pull request.** Si apre una pull request verso `development` che descrive che cosa cambia, perché, la classe, l'impatto sui dati, l'eventuale migrazione del database e il piano di ritorno. Il codice si rivede nella pull request, anche con strumenti automatici di revisione.

3 **Verifica.** L'integrazione continua esegue: controllo dei tipi del server, dell'interfaccia e dell'infrastruttura; test unitari; test end-to-end nel browser; scansione dei segreti su tutta la storia del repository (gitleaks); prova automatica di ripristino del database; test del gestionale commerciale interno. Se un controllo fallisce, la pull request non si unisce.

4 **Development.** Con l'unione, l'integrazione continua ripete la verifica e pubblica `app.dev.trailgun.ai`, che ha macchina, disco, pool degli utenti e dati propri e usa servizi esterni finti. Qui si prova la modifica come la userà lo studio.

5 **Staging.** Una pull request da `development` a `staging` porta la modifica su `app.staging.trailgun.ai`, con la stessa verifica. Staging è separato da production come development: nessun dato dei Clienti, servizi esterni finti di default.

6 **Production.** Una pull request da `staging` a `production`, con la verifica superata, pubblica `app.trailgun.ai`. Il rilascio parte solo dal branch `production`.

7 **Rilascio.** Il job di rilascio assume con OIDC il ruolo AWS del proprio ambiente, che vale solo per quel branch e solo per quell'ambiente: in GitHub non ci sono chiavi AWS. Costruisce l'immagine, etichettata con il commit, e la carica nel registro ECR, che la scansiona alla ricezione. Prima del riavvio controlla che non ci siano lavorazioni avviate dal modulo in corso: se ce ne sono, il rilascio si ferma e si rilancia dopo.

8 **Copia dei dati.** Per production, subito prima del riavvio, si fotografa il disco dati (snapshot etichettato con il commit, conservato 14 giorni): un rilascio difettoso si annulla anche sui dati.

9 **Riavvio e controllo.** L'app riparte con l'immagine nuova e i segreti dell'ambiente; il rilascio attende che `/api/health` risponda con la versione appena pubblicata. Le lavorazioni in coda (API, caselle email, lotti) ripartono da sole.

10 **Registrazione.** Il cambiamento si annota nel registro dei cambiamenti (documento 45): commit e pull request, classe, data e ora, esito della verifica, snapshot pre-rilascio, esito del controllo di salute, eventuale ritorno indietro.

5 Migrazioni del database

5.1 Lo schema del database cambia solo con migrazioni numerate, che vanno solo in avanti e restano compatibili con l'immagine precedente: in caso di ritorno indietro, la versione precedente dell'app funziona sui dati già migrati.

5.2 Una migrazione già pubblicata non si modifica mai; una correzione è una migrazione nuova.

5.3 Prima di applicare una migrazione, l'app copia il database con `VACUUM INTO` sullo stesso disco (ultime 5 copie). La prova automatica di ripristino della verifica controlla a ogni esecuzione che applicare di nuovo le migrazioni a una copia ripristinata non cambi nulla.

6 Rilascio manuale

6.1 Lo script di rilascio manuale è lo stesso che usa l'integrazione continua: un secondo percorso sarebbe un secondo modo di rompersi. Si usa per un cambiamento urgente o quando l'integrazione continua non è disponibile, con l'autorizzazione del Responsabile della sicurezza.

6.2 Per production lo script parte solo dal branch `production`, da un albero di lavoro pulito, e solo per un commit che ha superato la verifica; rifiuta il riavvio se ci sono lavorazioni del modulo in corso. Le credenziali sono temporanee, ottenute da IAM Identity Center con l'autenticazione a più fattori: nessuna chiave statica.

6.3 Il riavvio con lavorazioni in corso si forza solo per un'urgenza di sicurezza: gli studi coinvolti lo sanno e le lavorazioni interrotte si riavviano senza costi.

7 Infrastruttura

7.1 L'infrastruttura è descritta come codice (AWS CDK) nello stesso repository e segue la stessa strada fino alla pull request. Si applica solo a mano, ambiente per ambiente e con lo stack nominato, dopo il confronto fra lo stato attuale e quello nuovo: l'integrazione continua non ha i permessi per cambiarla.

7.2 Ogni cambiamento dell'infrastruttura di production è almeno significativo. Il disco dati è conservato anche se lo stack viene rimosso.

8 Finestre e comunicazione agli studi

8.1 Un rilascio ordinario dura meno di 2 minuti e parte solo senza lavorazioni del modulo in corso; una lavorazione interrotta si riavvia senza costi.

8.2 La manutenzione programmata si comunica con almeno 48 ore di preavviso, si svolge tra le 20:00 e le 07:00 o nel fine settimana e non supera le 8 ore al mese. Nei giorni lavorativi fra le 08:00 e le 20:00 non si fa manutenzione programmata.

8.3 Un intervento urgente di sicurezza si fa anche senza preavviso e si comunica appena possibile.

9 Ritorno alla versione precedente

9.1 **Ordinario.** Si annulla la modifica con una pull request verso `production`; l'integrazione continua verifica e pubblica.

9.2 **Urgente.** Si ripubblica con lo script di rilascio l'immagine dell'ultima versione buona (il registro conserva le ultime 15), come cambiamento urgente.

9.3 I dati non si toccano, perché le migrazioni sono compatibili con l'immagine precedente. Se il rilascio ha danneggiato i dati, si ripristina con il documento 30 (casi 1 e 2), dalla copia del database prima della migrazione o dallo snapshot preso prima del rilascio.

10 Dipendenze e immagini di base

Dependabot apre ogni settimana le pull request di aggiornamento delle dipendenze (applicazione, infrastruttura, gestionale interno), delle immagini di base e delle azioni dell'integrazione continua, raggruppate per ecosistema e sempre verso `development`; le correzioni di sicurezza arrivano appena l'avviso è pubblicato. Seguono la strada di ogni altro cambiamento, nei tempi di correzione del documento 24. L'immagine di Node.js e quella del proxy Caddy sono fissate a una versione esatta, e il container dell'app non gira come root.

11 Evidenze e registri

- Storia del repository e delle pull request, con la descrizione, la revisione e l'esito della verifica di ciascuna.
- Esiti dell'integrazione continua, compreso il verbale della prova automatica di ripristino, conservato come artefatto.
- Registro dei cambiamenti (documento 45), compilato per ogni rilascio di production.
- Snapshot pre-rilascio etichettati con il commit; immagini etichettate con il commit nel registro ECR.
- CloudTrail per le azioni sull'infrastruttura (12 mesi, con convalida dell'integrità).

12 Riferimenti

DORA (Reg. (UE) 2022/2554), art. 9, par. 4, lett. e); Reg. del. (UE) 2024/1774, art. 16 (analisi e test del software prima della produzione); ISO/IEC 27001:2022, controlli 8.25 (ciclo di sviluppo sicuro), 8.29 (test di sicurezza), 8.31 (separazione degli ambienti) e 8.32 (gestione dei cambiamenti); GDPR, art. 25 (protezione dei dati fin dalla progettazione) e art. 32.

Trailgun

On target.

In a flash.

Dai documenti al lavoro finito: agenti AI per pratiche legali e finanziarie.

SICUREZZA E INCIDENTI, 24/7
security@trailgun.ai

REPERIBILITÀ PRIORITÀ 1
[+39 ...]

PRIVACY E DPO
privacy@trailgun.ai

Procedura 20 del pacchetto di conformità Trailgun, versione 2.0, stato al 01/12/2026. Attua la politica P05 (sicurezza delle operazioni e gestione dei cambiamenti) e la politica P06 (sviluppo sicuro). Documento riservato, destinato al Cliente e alle sue autorità di vigilanza.