Chaîne de qualification et de livraison

Cette page explique comment une version de LoGeAs Web passe du code fusionné à la livraison client : qualification par les tests, validation, beta, recette, release. Elle s'adresse à toute l'équipe : développeurs, et personne qui valide et recette.

Voir aussi : Lancer les tests E2E (détail du run complet) et la méthodologie des tests (`logeas-cartographie/src/assets/methodologie-tests.html`, section 8).

Pourquoi une chaîne

Avant, les tests et la livraison formaient deux circuits séparés :

- une beta pouvait partir sans rapport qualité validé ;
- entre deux betas, un run de tests était rangé sous la version de la **dernière beta livrée**, alors qu'il testait la **prochaine** ;
- les tests unitaires étaient joués deux fois (run complet, puis rapport de release), sans garantie que ce soit sur le même code.

Désormais, **aucune version n'est livrée sans un rapport qualité validé qui porte sur son code exact**, et deux **portes bloquantes** le vérifient automatiquement.

Vue d'ensemble

Étape	Qui	Comment	Bloquant
1. Fusion vers la branche de version	développeur	`/fusion-dev`	build, tests unitaires, lint
2. Qualification	développeur	`scripts\qualifier-version.bat`	préconditions sur les dépôts
3. Validation du rapport qualité	valideur	carto, écran **Rapports de version**	—
4. Beta	développeur	`scripts\livrer-beta.bat`	**porte beta**
5. Déploiement sur le serveur de test	développeur	copie manuelle (bureau à distance)	—
6. Recette de la beta	valideur	carto, écran **Rapports de version**	—
7. Release client	développeur	`scripts\livrer-release.bat`	**porte release**
8. Déploiement client	développeur	copie manuelle (bureau à distance)	—

La même personne peut valider et recetter (équipe trop petite pour les « quatre yeux »).

Cheminements pas à pas

Chaque cas ci-dessous renvoie, pour le détail, aux sections « Les étapes en détail » et « Les portes ».

Cas 1 : faire une beta (cas normal)

Point de départ : le travail est fusionné sur la branche de version (`/fusion-dev`).

1. **Pousser** la branche de version de *logeas-web* et celle de *logeas-lib* : la qualification refuse un dépôt qui a des commits non poussés ou des modifications locales.
2. **Fermer la fenêtre console d'Angular** si elle est ouverte.
3. Double-clic sur `scripts\qualifier-version.bat`. Noter la **version candidate** affichée (ex. `11.2.0.9`).
4. **Rester devant le poste** pendant la reconstruction de *logeas-lib* (quelques minutes), jusqu'à la **confirmation UAC** du démarrage des serveurs. Ensuite, le run tourne seul (environ 15 minutes).
5. **Ne rien commiter pendant le run**, dans aucun des deux dépôts.
6. Dans la carto, écran **Rapports de version** (connecté) : ouvrir `run-<version>-<date>`, le relire, puis **Valider** (ou **Validé avec réserves**, avec un commentaire ; voir la règle des réserves).
7. Double-clic sur `scripts\livrer-beta.bat`. La porte vérifie tout ; le script fait le bump, le rapport de release, le build, le commit et les tags.
8. Répondre **o** à « Pousser vers origin ? » : c'est aussi ce qui inscrit la beta dans la trace du rapport, sans quoi la release client sera impossible.
9. Copier `D:\Developpements\VersionPublic\LogeasWeb-Test\browser\` sur le serveur de test (bureau à distance).
10. Passer au **cas 2** (recette et release) ou, si un défaut est trouvé, au **cas 3**.

Cas 2 : recetter la beta et livrer la release client

Point de départ : une beta livrée par le cas 1 et déployée sur le serveur de test.

1. Sur le serveur de test, dérouler la liste de contrôle de la recette (section « Recette de la beta »).
2. Dans la carto, écran **Rapports de version**, sur le rapport qualité de la version : section **Recette de la beta `X.Y.Z.W_beta`**. Pour chaque point, choisir **Vérifié**, ou **Sans objet** avec une justification.
3. Cliquer **Recette conforme** (le bouton n'est actif que si tous les points sont traités).
4. Vérifier que **rien n'a été commité** depuis la beta, dans aucun des deux dépôts.
5. Double-clic sur `scripts\livrer-release.bat`, répondre **o** à la question du push.
6. Copier `D:\Developpements\VersionPublic\LogeasWeb\browser\` sur le serveur client.

Cas 3 : un défaut est trouvé après la beta

Que le défaut soit trouvé en recette ou autrement, **aucun correctif ne part sur la beta existante**.

1. Si la recette était en cours : dans la carto, cliquer **Recette non conforme** avec un commentaire qui décrit le défaut (la trace garde ainsi pourquoi la beta n'est pas partie chez le client).
2. Corriger sur sa branche perso, puis `/fusion-dev` vers la branche de version, puis pousser.
3. Reprendre le **cas 1** depuis le début : nouvelle qualification (la version candidate devient la beta suivante, ex. `11.2.0.10`), validation, nouvelle beta, déploiement, recette.

Cas 4 : le rapport qualité n'est pas bon

Un échec nouveau par rapport au dernier rapport validé, ou un résultat inexploitable.

1. Dans la carto, **Refusé**, avec un commentaire (obligatoire).
2. Corriger (branche perso, `/fusion-dev`, push), puis relancer `scripts\qualifier-version.bat`.
3. Le nouveau rapport est à valider ; l'ancien reste consultable, marqué « Remplacé ».
4. Si les échecs sont connus et suivis (chacun rattaché à un ticket ou à un échec déjà connu), on peut au contraire **Valider avec réserves**, en les listant dans le commentaire.

Cas 5 : la qualification a échoué en cours de route

Serveurs qui ne démarrent pas, run interrompu, poste redémarré…

1. Sans résultats des tests d'interface, **aucun rapport n'est produit** : il n'y a rien à refuser.
2. Corriger la cause (fenêtres console des serveurs, par exemple), puis relancer `scripts\qualifier-version.bat` tel quel : le code n'a pas changé, la version candidate non plus.
3. Si un rapport a quand même été produit mais qu'il n'est pas exploitable : le **refuser** et relancer.

Cas 6 : démarrer le cycle d'une nouvelle version

1. Créer la nouvelle branche de version (ex. `11.3.0`) dans *logeas-web* et, si la bibliothèque évolue aussi, dans *logeas-lib*, et les **pousser** (une branche sans branche amont est refusée).
2. Basculer le poste sur ces branches (`scripts\switch-lib-version.ps1 -LibBranch 11.3.0 -WebBranch 11.3.0`, ou un profil de `scripts\switch-lib-profiles.json`).
3. Travailler, fusionner (`/fusion-dev`), puis suivre le **cas 1**. La version candidate est tirée du nom de la branche : la première beta sera `11.3.0.1`.

Cas 7 : corriger une version déjà livrée au client

Même chaîne, sur la branche de la version en ligne.

1. Basculer sur cette version : `scripts\switch-lib-version.ps1 -Profile hotfix` (branches définies dans `scripts\switch-lib-profiles.json`).
2. Corriger, fusionner sur la branche de cette version, pousser.
3. Suivre le **cas 1** puis le **cas 2**. Comme cette version est déjà publiée (tag `_release`), la version candidate passe au cycle suivant : par exemple `11.1.3` publiée → beta `11.1.4.1`, puis release `11.1.4`.
4. Reporter ensuite le correctif sur la version en cours de développement (`/fusion-dev`), puis revenir dessus : `scripts\switch-lib-version.ps1 -Profile next`. Ce script signale d'ailleurs les correctifs d'une version antérieure pas encore reportés.

Cas 8 : urgence, livrer sans la chaîne

À réserver aux vraies urgences (serveur Nono injoignable, blocage en production…). La version sera affichée **« Version non qualifiée »**.

1. Bump : `npm run prebuild:beta` (beta) ou `npm run prebuild` (release).
2. Rapport de release **sans** qualification : `node scripts/release-report.js beta` (ou `release`) ; il rejoue les tests unitaires et affiche le bandeau « Version non qualifiée ».
3. Build : `npx ng build –configuration production,beta` (beta) ou `npx ng build –configuration production` (release). Jamais `npm run build`, qui referait le bump.
4. Commit `Preparation version X`, tag, push, comme d'habitude.
5. Même sans rapport de release, **À propos > Logiciel** affiche « Version non qualifiée » : une version absente des rapports de release est considérée comme construite hors chaîne.
6. **Régulariser** dès que possible : qualification du même code, puis validation.

Cas 9 : une porte refuse

1. Lire le motif affiché : chaque refus dit quoi faire (tableau « Messages fréquents » ci-dessous).
2. Pour revérifier sans rien modifier : `powershell -ExecutionPolicy Bypass -File scripts\livrer-version.ps1 -Nature beta -VerifierSeulement` (ou `-Nature release`), ou demander à Claude `/release-beta` / `/release-client`, qui expliquent le blocage.
3. **Ne jamais contourner une porte** : si le blocage est injustifié, c'est un défaut des scripts, à signaler et corriger.

Les étapes en détail

Qualification

But : produire le rapport qualité de la prochaine version, sur un code identifié sans ambiguïté.

Avant de lancer, pour *logeas-web* ET *logeas-lib* :

- être sur la branche de version (ex. `11.2.0`) ;
- aucune modification locale non commitée ;
- branche à jour avec `origin` et poussée ;
- **Angular arrêté** (fermer sa fenêtre console) : sinon il continuerait de servir l'ancienne bibliothèque après sa réinstallation.

Le script vérifie tout cela lui-même et s'arrête en disant quoi corriger. Pour vérifier sans rien lancer : `powershell -ExecutionPolicy Bypass -File scripts\qualifier-version.ps1 -VerifierSeulement` (ou la commande Claude `/qualifier-version`).

Lancer : double-clic sur `scripts\qualifier-version.bat`. Le script :

1. affiche la **version candidate** : la prochaine beta (ex. `11.2.0.9`), sous laquelle le rapport sera rangé ;
2. reconstruit *logeas-lib* depuis son dernier commit et la réinstalle dans *logeas-web* (quelques minutes) ;

3. demande la **confirmation UAC** du démarrage des serveurs : **rester devant le poste jusque-là** ;
4. enchaîne sans surveillance : tests unitaires des deux dépôts, tests d'interface, rapport qualité, analyse par Claude, dépôt du rapport en brouillon sur le serveur Nono.

Pendant le run : ne rien commiter, ne rien pousser, ni dans *logeas-web* ni dans *logeas-lib*. Le rapport le détecterait (« code modifié pendant le run ») et la porte le refuserait.

Résultat : un rapport `run-<version>-<AAAAMMJJ-HHMM>.html` d'origine « Qualification », visible dans l'écran **Rapports de version** de la carto. Si on relance une qualification pour la même version, un nouveau rapport est créé ; les précédents restent consultables, marqués « Remplacé ».

Validation du rapport qualité

Dans la carto, écran **Rapports de version**, connecté : relire le rapport (en particulier l'analyse rédigée par Claude), puis choisir une décision.

Décision	Quand	Commentaire
Validé	tous les tests passent, analyse relue	facultatif
Validé avec réserves	des échecs, **chacun rattaché à un ticket ou à un échec déjà connu**	obligatoire
Refusé	un échec nouveau par rapport au dernier rapport validé, ou résultat inexploitable	obligatoire

« Validé » et « Validé avec réserves » **publient** le rapport en même temps : il devient visible sans connexion et constitue la pièce qualité (dossier NF Logiciel) de la version.

Beta

Double-clic sur `scripts\livrer-beta.bat`. Le script :

1. passe la **porte beta** (voir plus bas) : s'il manque quelque chose, il s'arrête et l'explique ;
2. fait le bump de version (4e chiffre) et vérifie que c'est bien la version qualifiée ;
3. génère le rapport de release : les tests unitaires et le résumé des tests d'interface (Playwright) y sont **repris** de la qualification (rien n'est rejoué), avec un renvoi vers le rapport qualité pour le détail ;
4. lance le build de production vers `D:\Developpements\VersionPublic\LogeasWeb-Test\` ;
5. crée le commit « Preparation version X.Y.Z.W », le tag `X.Y.Z.W_beta` et, si besoin, le tag `web-X.Y.Z.W_beta` dans *logeas-lib* ;
6. **demande confirmation** avant de pousser. Si on accepte : push, inscription de la beta dans la trace du rapport qualité (indispensable pour la suite), reprise du rapport de release dans la carto.

Si le rapport de release ou le build échoue, le bump est annulé : rien n'est commité ni tagué. Si on refuse le push, le script affiche les commandes pour le faire plus tard, **y compris l'inscription de la beta dans la trace** (`node scripts\porte-livraison.js tracer …`), à ne pas oublier.

Reste à **copier** `LogeasWeb-Test\browser\` sur le serveur de test (bureau à distance).

Recette de la beta

Une fois la beta déployée, dans la carto, écran **Rapports de version**, sur le rapport qualité de la version : la section **Recette de la beta** propose une liste de contrôle minimale.

- la version affichée dans « À propos » est celle de la beta livrée ;
- connexion et ouverture d'une base ;
- annuaire : recherche et ouverture d'une fiche ;
- comptabilité : saisie et enregistrement d'une écriture ;
- états : génération d'un état PDF ;
- sauvegarde / export d'une base ;
- chaque correctif et évolution annoncé dans le rapport de release a été vérifié.

Chaque point est **Vérifié**, ou **Sans objet** avec une justification. Le bouton **Recette conforme** n'est actif que si tous les points sont traités. **Recette non conforme** exige un commentaire.

Seule une recette conforme ouvre la release client. Une recette non conforme impose une correction, donc une nouvelle qualification et une nouvelle beta (cas 3).

Release client

Double-clic sur `scripts\livrer-release.bat`. Même déroulé que la beta, derrière la **porte release** : bump en version à 3 chiffres, build vers `D:\Developpements\VersionPublic\LogeasWeb\`, commit, tag `X.Y.Z_release`, confirmation, push, inscription de la release dans la trace du rapport. Reste à copier `LogeasWeb\browser\` sur le serveur client.

Le rapport qualité, déjà publié lors de sa validation, est la pièce qualité de la release.

Les portes

Les portes lisent le manifeste des rapports et leurs décisions sur le serveur Nono de production, et comparent au code présent sur le poste. Elles sont **bloquantes**. Pour les vérifier sans rien modifier : `powershell -ExecutionPolicy Bypass -File scripts\livrer-version.ps1 -Nature beta -VerifierSeulement` (ou `-Nature release`).

Porte beta :

- les deux dépôts sont sur leur branche de version, propres, à jour et poussés ;
- un rapport d'origine « Qualification » de la version candidate porte **exactement** le dernier commit des deux dépôts (le plus récent, s'il y en a plusieurs) ;
- ce run n'a eu ni modification locale, ni changement de code en cours de route, et les paquets *logeas-lib* installés sont toujours ceux du run ;
- sa décision est « Validé » ou « Validé avec réserves ».

Porte release :

- `package.json` porte une beta, et le tag de cette beta est le dernier commit de *logeas-web* (**aucun commit depuis la beta**) ;
- *logeas-lib* est toujours au commit qualifié, avec les mêmes paquets installés ;
- entre le code qualifié et la beta, seul le commit « Preparation version » (dans `package.json`, seul le numéro de version a pu changer) ;
- le rapport qualité est toujours validé, et la **recette** de cette beta est « Conforme ».

Messages fréquents :

Message	Que faire
aucun rapport de qualification … pour le code actuel	lancer `scripts\qualifier-version.bat` (un commit a été fait depuis la dernière qualification)
le rapport … n'est pas encore validé	le valider dans la carto
le rapport … a été refusé	cas 4
… commit(s) non poussé(s)	pousser la branche
modifications locales non commitées	commiter (`/fusion-dev`) ou annuler
paquets logeas-lib installés différents de ceux du run qualifié	relancer la qualification
logeas-web a reçu des commits depuis la beta	cas 3 : requalifier, puis nouvelle beta
aucun rapport qualifié n'a ouvert la beta …	la beta n'a pas été inscrite dans la trace (push refusé ?) : lancer la commande `tracer` affichée par `livrer-beta`
la beta … n'a pas été recettée	faire la recette dans la carto (cas 2)
la recette de la beta … n'est pas conforme	cas 3
serveur Nono injoignable (manifeste ou ApprobDoc illisible)	attendre son retour : aucune livraison n'est possible sans lui (en vraie urgence : cas 8)

Traçabilité : où trouver quoi

Information	Où
Rapports qualité, décisions, recettes	carto, écran **Rapports de version** (https://cartographie-fonctionnelle.logeas-web.fr/RapportsVersion)
Clé du code d'un rapport (commits des deux dépôts, empreinte des paquets lib)	en tête du rapport ; en détail dans le manifeste (`cleCode`)
Historique d'un rapport (validation, publication, beta, recette, release)	table `ApprobDoc` du serveur Nono, trace JSON du rapport
Livraisons	tags `X.Y.Z.W_beta` / `X.Y.Z_release` (logeas-web), `web-*` (logeas-lib)

Information	Où
Qualification d'une version livrée	son rapport de release et **À propos > Logiciel**

Référence des scripts

Script	Rôle	Options
`scripts\qualifier-version.bat`	préconditions, reconstruction de la lib, run complet de qualification	`-VerifierSeulement`
`scripts\livrer-beta.bat`	porte beta, puis livraison de la beta	`-VerifierSeulement`
`scripts\livrer-release.bat`	porte release, puis livraison de la release client	`-VerifierSeulement`
`scripts\lancer-e2e-complet.bat`	run complet de travail (hors chaîne : origine « Run complet », non accepté par la porte)	`-SansUnitaires`, `-SansServeurs`, `-SansDoc`
`scripts\switch-lib-version.ps1`	bascule le poste sur une paire de branches web / lib (rebuild et réinstallation de la lib)	`-Profile hotfix` / `next`, `-LibBranch`, `-WebBranch`
`node scripts/porte-livraison.js beta` (ou `release`)	porte seule (utilisée par les scripts ci-dessus)	`–resultat <fichier>`, `–serveur <url>`
`node scripts/version.js beta –simuler`	affiche la version candidate sans rien modifier	—

Les commandes Claude `/qualifier-version`, `/release-beta` et `/release-client` ne livrent rien elles-mêmes : elles vérifient les préconditions ou la porte, expliquent ce qui bloque, puis renvoient vers ces scripts.

From:
https://wiki-logeas.fr/certif/ - **dokuwiki-certif**

Permanent link:
https://wiki-logeas.fr/certif/doku.php?id=certif:procedure:develop:chainequalificationlivraison&rev=1790345169

Last update: **2026/09/25 16:06**