Retour au dossier de maintenance

Sujets connexes	Cartographie fonctionnelle de LoGeAs Informations sur la version en cours Proc#145 : Procédure de Gestion des Versions Proc#168 - Chaîne de qualification et de livraison Proc#167 - Procédure de mise en ligne d'une version du client « lourd » Proc#169 - Cycle de release LoGeAs Web Proc#170 - Guide pratique — Gestion des versions LoGeAs Web

Cycle de release LoGeAs Web

1. Objet et périmètre

Ce document décrit le dispositif technique de gestion des versions et de publication de LoGeAs Web, depuis l'intégration des développements jusqu'à la production des versions destinées aux utilisateurs.

Il constitue la référence technique du processus de release pour les dépôts `logeas-web` et, lorsque nécessaire, `logeas-lib`.

Il complète les documents suivants :

- Procédure de gestion des versions : règles générales de versionnage et d'identification des versions.
- Chaîne de qualification et de livraison : processus qualité, responsabilités, qualification et validation.
- Guide pratique de gestion des versions : mode opératoire destiné aux développeurs.

Le présent document décrit le fonctionnement technique du dispositif. Il ne remplace ni les décisions de validation ni les contrôles de qualification prévus dans la chaîne de livraison.

2. Objectifs

Le dispositif de release a pour objectifs :

- **Reproductibilité** : exécuter les opérations techniques selon une procédure identifiée et maîtrisée.
- **Traçabilité** : relier chaque version publiée à son code source, à ses commits et aux résultats des contrôles.
- **Sécurisation** : empêcher la production d'une livraison lorsque les contrôles techniques obligatoires échouent.
- **Séparation des environnements** : distinguer les versions de test des versions destinées aux utilisateurs.
- **Maîtrise du versionnage** : éviter les collisions, les incohérences et les versions non identifiables.
- **Conservation des preuves** : permettre de retrouver les informations nécessaires à la justification d'une livraison.

3. Architecture générale du cycle

Le cycle de publication repose sur quatre étapes :

Étape	Opération	Résultat
1	Travail personnel	Développements enregistrés dans une branche individuelle
2	Intégration (`/fusion-dev`)	Code intégré et contrôlé sur la branche commune
3	Publication beta (`/release-beta`)	Version de test, taguée et accompagnée de son rapport
4	Publication client (`/release-client`)	Version de production, taguée et accompagnée de ses éléments de traçabilité

Les étapes 2 à 4 sont assistées par des commandes dédiées de Claude Code.

Le passage d'une étape à l'autre reste une décision humaine. L'automatisation réalise les opérations et les contrôles prévus, mais ne décide pas de la pertinence fonctionnelle d'une livraison.

Une publication client peut exceptionnellement être réalisée sans beta préalable lorsque le contexte le justifie. Les contrôles techniques applicables à la release restent alors obligatoires.

4. Organisation des dépôts et des environnements

4.1 Dépôts

Le cycle concerne principalement les dépôts suivants :

Dépôt	Rôle
`logeas-web`	Interface Angular LoGeAs Web et scripts de génération des versions
`logeas-lib`	Bibliothèques Angular partagées utilisées par les interfaces
`logeas-cartographie`	Conservation et publication des éléments de cartographie et des rapports qualité

Les bibliothèques de `logeas-lib` sont utilisées par `logeas-web` sous forme de paquets construits. Le changement de branche ne suffit donc pas à garantir la cohérence des dépendances : les paquets doivent correspondre aux versions de bibliothèques prévues pour le cycle concerné.

4.2 Environnements de génération

Les builds sont générés localement sur le poste de développement.

Destination	Usage
`D:\Developpements\VersionPublic\LogeasWeb-Test\browser\`	Version beta destinée aux tests

Destination	Usage
`D:\Developpements\VersionPublic\LogeasWeb\browser\`	Version destinée à la production

La séparation des destinations est assurée par les configurations Angular :

- `production,beta` pour la génération beta ;
- `production` pour la génération client.

Cette séparation évite qu'une génération de test écrase directement les fichiers de production.

4.3 Déploiement

La génération des fichiers et leur déploiement sur les serveurs sont deux opérations distinctes.

Le transfert vers les serveurs de test et de production est réalisé manuellement, par une personne disposant des accès nécessaires au bureau à distance.

Les commandes de release ne réalisent pas elles-mêmes ce transfert.

5. Description technique des étapes

5.1 Travail personnel

Chaque développeur travaille sur une branche personnelle issue de la branche de version concernée.

Les commits et les sauvegardes de cette branche restent libres, afin de ne pas imposer les contrôles de publication à chaque sauvegarde intermédiaire.

Cette étape ne constitue pas une intégration validée et ne garantit pas que le code compile ou que les tests passent.

Le premier contrôle technique obligatoire du cycle intervient lors de l'intégration.

5.2 Intégration : /fusion-dev

La commande `/fusion-dev` intègre les développements personnels à la branche de version commune.

Elle réalise les opérations suivantes :

- récupération des informations du dépôt distant ;
- sélection et mise à jour de la branche de version cible ;
- identification des commits à intégrer ;
- fusion de la branche personnelle ;
- constitution d'un message de fusion contenant la liste des commits intégrés ;
- vérification du code après fusion ;

- publication de la branche commune si les contrôles sont satisfaisants.

Les contrôles techniques portent sur :

- la compilation Angular ;
- les tests unitaires ;
- l'analyse statique et le lint.

Les résultats sont examinés avant la publication de la branche commune.

En cas d'échec, la procédure ne doit pas publier une intégration non validée. En cas de conflit métier ambigu, une décision humaine est nécessaire.

La procédure n'utilise pas de `push –force` sur la branche commune.

Résultat attendu : une branche de version intégrée, contrôlée et accompagnée d'un historique de fusion exploitable.

5.3 Publication beta : /release-beta

La commande `/release-beta` produit une version destinée à la validation interne.

Elle est exécutée depuis la branche de version commune, préalablement intégrée et mise à jour.

Les opérations techniques sont les suivantes :

- vérification de l'état du dépôt et de la branche ;
- calcul du numéro de version beta ;
- mise à jour des informations de version ;
- régénération des éléments nécessaires à la compilation ;
- compilation en configuration de production beta ;
- génération du rapport de release ;
- enregistrement des éléments de version dans un commit dédié ;
- création d'un tag Git annoté ;
- publication du commit et du tag sur le dépôt distant.

La génération utilise la commande :

```
npm run build:beta
```

Le script de version est exécuté en mode `beta` et le build Angular utilise la configuration `production,beta`.

Le résultat est déposé dans le répertoire de test :

```
D:\Developpements\VersionPublic\LogeasWeb-Test\browser\
```

Le rapport de release est généré par le script `scripts/release-report.js`.

Il comprend notamment :

- les changements identifiés dans l'historique Git depuis la dernière release ;
- les informations de version ;
- les résultats des tests disponibles et leur couverture ;
- les versions des composants de la Galaxie LoGeAs identifiées à partir des exécutables disponibles.

Le rapport HTML est enregistré dans `src/assets/releases/` et référencé dans `manifest.json`.

Le tag beta suit la convention :

```
X.Y.Z.W_beta
```

La publication beta ne vaut pas validation fonctionnelle. Elle fournit un artefact et des éléments de traçabilité destinés à la phase de test.

5.4 Publication client : /release-client

La commande `/release-client` produit la version destinée aux utilisateurs.

Elle est exécutée sur la branche de version concernée.

Lorsque la release fait suite à une beta, les changements intervenus depuis cette beta sont examinés. Si des évolutions fonctionnelles importantes ont été intégrées, une nouvelle beta peut être nécessaire.

La génération utilise :

```
npm run build
```

Le script de version est exécuté en mode `release`, puis Angular compile l'application avec la configuration `production`.

Les opérations comprennent :

- la vérification de l'état du dépôt ;
- le contrôle de la cohérence avec la beta, lorsqu'elle existe ;
- le calcul du numéro de release ;
- la génération du build de production ;
- la génération du rapport de release ;
- la création du commit de préparation de version ;
- la création du tag Git annoté ;
- la publication du commit et du tag ;
- la mise à jour du rapport qualité NF dans le dépôt `logeas-cartographie`.

Le build est généré dans :

```
D:\Developpements\VersionPublic\LogeasWeb\browser\
```

Le tag de publication suit la convention :

```
X.Y.Z_release
```

Le rapport qualité NF est généré à partir des résultats réels des tests de `logeas-web` et `logeas-lib`. Il doit être relu avant son intégration dans le dépôt de cartographie.

Le déploiement sur le serveur de production reste une opération distincte, réalisée manuellement après validation de la livraison.

6. Gestion des versions

6.1 Format

Les versions de LoGeAs Web utilisent trois ou quatre composantes numériques :

- `X.Y.Z` pour une release ;
- `X.Y.Z.W` pour une beta.

Le suffixe du tag identifie la nature de la publication :

- `_beta` pour une version de test ;
- `_release` pour une version destinée aux utilisateurs.

Les règles générales de numérotation sont définies dans la procédure de gestion des versions.

6.2 Calcul automatique

Le calcul est centralisé dans :

```
scripts/version.js
```

La fonction `resolveNextVersion` détermine le numéro à utiliser en fonction :

- de la version actuelle de `package.json` ;
- du nom de la branche de version ;
- du mode de génération (beta ou release) ;
- des tags de release déjà présents dans Git.

Pour une beta, le quatrième chiffre est incrémenté à chaque nouvelle génération du même cycle.

Lorsqu'une nouvelle branche numérotée correspond à un cycle ultérieur, le script peut utiliser son numéro comme nouvelle base de version.

Pour une release, le quatrième chiffre est supprimé. Le numéro `X.Y.Z` utilisé par les betas est repris s'il n'a pas déjà été publié. Si une release portant ce numéro existe déjà, le script avance vers un nouveau numéro conformément à la règle implémentée.

6.3 Traçabilité de la version

Le fichier `src/assets/version.ts` est régénéré lors de la préparation d'un build.

Il contient notamment les informations permettant d'identifier :

- la branche Git ;
- le commit source ;
- la date de génération.

Ces données permettent de rapprocher une version exécutée de son état source.

6.4 Cas particuliers

Les cas de changement de version mineure, de correction urgente et de travail parallèle sur plusieurs versions sont décrits dans le guide pratique.

Ils ne doivent pas conduire à réécrire l'historique d'une version déjà publiée.

7. Contrôles et garde-fous

Les procédures outillées appliquent des contrôles destinés à éviter les publications incohérentes ou non traçables.

Contrôle	Étape concernée	Objectif
Vérification de l'état du dépôt	Intégration, beta, release	Éviter de publier à partir d'un état local non maîtrisé
Vérification de la branche	Toutes les étapes outillées	Garantir que l'opération concerne la branche attendue
Compilation	Intégration, beta, release	Détecter les erreurs de compilation
Tests unitaires	Intégration et processus de qualification	Vérifier le comportement du code
Contrôle du numéro de version	Beta et release	Éviter les incohérences de versionnage
Vérification des tags	Beta et release	Éviter les collisions et préserver la traçabilité
Séparation des répertoires de sortie	Beta et release	Empêcher la confusion entre test et production
Vérification des commits intégrés	Intégration	Conserver la traçabilité des changements
Rapport de release	Beta et release	Documenter les changements et les résultats disponibles

En cas d'échec d'un contrôle bloquant, la procédure doit s'arrêter et signaler l'anomalie.

Aucune procédure ne doit résoudre silencieusement un conflit métier ou écraser des modifications non validées.

Les opérations de publication ne doivent pas utiliser `push --force` ni supprimer automatiquement des branches ou des tags.

Les éventuels échecs de tests préexistants doivent être identifiés et distingués des régressions introduites par le changement en cours.

8. Rapports et éléments de preuve

Les éléments suivants permettent de reconstituer une publication :

- l'historique Git de la branche concernée ;
- le commit de fusion et la liste des commits intégrés ;
- le commit de préparation de version ;
- le tag annoté de beta ou de release ;
- les informations intégrées dans `version.ts` ;
- le rapport de release HTML ;
- le rapport qualité NF correspondant, lorsqu'il est applicable ;
- les résultats des tests et de la qualification ;
- les éléments attestant de la validation et du déploiement.

Les rapports de version publiés sont consultables dans l'application, notamment depuis l'écran « Rapports de release ».

Le suivi des versions et des tests est également accessible depuis la cartographie fonctionnelle et les outils de suivi associés.

Les modalités de conservation et de validation des preuves relèvent de la chaîne de qualification et de livraison.

9. Opérations restant manuelles

Les opérations suivantes restent sous la responsabilité des personnes habilitées :

- le choix du moment de l'intégration ;
- la décision de créer une beta ;
- la validation fonctionnelle de la beta ;
- la décision de livrer une release ;
- la relecture et la validation des rapports ;
- le transfert des fichiers générés vers les serveurs ;
- la vérification du déploiement sur l'environnement cible.

La séparation entre génération, validation et déploiement permet de conserver un contrôle humain sur les étapes engageant la mise à disposition du logiciel.

10. Maintenance du dispositif

Les scripts et configurations qui mettent en œuvre le cycle de release font partie du dispositif de développement de LoGeAs Web.

Toute modification significative de ces éléments doit être contrôlée et testée avant d'être utilisée pour une publication.

Les évolutions du dispositif doivent préserver :

- la traçabilité des versions ;
- la reproductibilité des générations ;
- la séparation des environnements ;
- la fiabilité des contrôles ;
- la conservation des preuves nécessaires à la certification.

La présente procédure doit être révisée lorsque le fonctionnement effectif du cycle de release évolue.

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

Permanent link:
https://wiki-logeas.fr/certif/doku.php?id=certif:procedure:develop:proceduremiseenligne:logeas-web&rev=1790608491

Last update: **2026/09/28 17:14**