

Avant-propos

1. À qui s'adresse cet ouvrage ?	17
2. Objectifs de l'ouvrage	17
3. Prérequis	18
4. Progression	18
5. Détail des chapitres	19
6. Un point sur les langues	21
7. Remerciements	21
8. Introduction à Git	22

Git et la gestion de version

1. La gestion de version	23
2. Les intérêts de la gestion de version	24
2.1 Une véritable machine à remonter le temps	24

2.2 Une documentation détaillée et datée	24
2.3 Une pierre de Rosette pour collaborer	25
3. Histoire de la gestion de version	25
3.1 Systèmes de gestion de versions locaux	25
3.2 Systèmes de gestion de versions centralisés	26
3.3 Systèmes de gestion de versions décentralisés	27
4. Pourquoi Git ?	30
Installation de Git	
1. Installation sous Linux	33
1.1 Installation à partir de paquets préexistants	33
1.2 Installation à partir des sources	34
2. Installation sous Mac OS X	35
3. Installation sous Windows	36
4. L'aide de Git	40
4.1 Généralités	40
4.2 Types de commandes Git	

4.2.1 Les commandes de porcelaine	42
4.2.2 Les commandes de plomberie	42
5. Configuration requise	43
5.1 Configurer le nom de l'utilisateur	43
5.2 Configurer l'e-mail de l'utilisateur	43
Création d'un dépôt	
1. Créer un dépôt local	45
2. Le contenu du dossier .git	46
3. Le fichier README	47
4. Markdown	49
4.1 Présentation	49
4.2 Éléments de syntaxe	50
4.2.1 Titres	50
4.2.2 Listes non ordonnées	51
4.2.3 Listes ordonnées	51
4.2.4 Mettre en gras	51

4.2.5 Mettre en italique	51
4.2.6 Ligne horizontale	52
4.2.7 Code	52
4.2.8 Tableaux	52
4.2.9 Liens	53
4.2.10 Notes de bas de page	53
5. reStructuredText	54
5.1 Présentation	54
5.2 Éléments de syntaxe	54
5.2.1 Titres	54
5.2.2 Listes autonumérotées	55
5.3 Logiciels	56
6. Outils pour travailler avec Markdown	56
6.1 Sublime Text	56
6.2 Texts	57
6.3 Ulysses	57
7. Configurer le dépôt local	59
7.1 Configuration minimale	59
7.2 Niveaux de configuration	60

7.2.1 Le niveau système	60
7.2.2 Le niveau utilisateur	60
7.2.3 Le niveau dépôt	60
7.3 Les paramètres configurables	61
7.3.1 Définir l'éditeur de texte	61
7.3.2 Modèle de commit	61
7.3.3 Ignorer des fichiers	62
7.3.4 Hashs abrégés	62
7.4 Définition d'alias Git	63
8. Les options de configuration avancées	64
8.1 Pagination	64
8.2 Expressions régulières étendues	64
8.3 Séparateur de mots	64
8.4 Ancêtre commun des conflits	65
Manipulation des fichiers et commit	
1. Gestion des fichiers et commit	67
2. Une histoire de hash	68
2.1 Une identification par contenu	

2.2 Risque de collision	69
	69
3. Les trois zones d'un fichier	70
3.1 Le répertoire de travail	71
3.2 L'index	73
3.3 Le dépôt	74
4. Manipuler les fichiers	77
4.1 Ajouter des fichiers dans l'index	77
4.2 Déplacer ou renommer des fichiers	78
4.3 Supprimer des fichiers	79
4.4 Arrêter de suivre un fichier	80
4.5 Ignorer des fichiers	80
5. Commiter ou enregistrer des modifications	82
5.1 Effectuer un premier commit	82
5.2 Rédiger un bon message de commit	84
5.2.1 Les règles d'un message de commit	84
5.2.2 Méthode pour le titre	85
5.2.3 En quelle langue ?	86

Consultation et manipulation de l'historique

1. Lister les commits avec git log	89
1.1 Limiter le nombre de commits affichés	91
1.2 Afficher les statistiques	92
1.3 Afficher chaque commit sur une seule ligne	93
1.4 Filtrer les commits chronologiquement	93
1.5 Filtrer les commits selon les intervenants	94
1.6 Afficher le graphique des branches	95
1.7 Spécifier un format de sortie	96
1.8 Prendre en compte les merges	98
1.9 Lister les commits impactant un fichier	99
2. Afficher les différences de contenu	100
2.1 Différences en cours dans le répertoire	100
2.2 Différences entre l'index et HEAD	101
2.3 Différences entre le répertoire de travail et HEAD	102
2.4 Différences introduites par un ou plusieurs commits	102
2.5 Différences de mots	103
3. Identifier l'auteur d'une ligne de code	105
4. Rechercher des commits avec le mode pick axe	

	106
5. Supprimer les modifications du répertoire de travail	107
6. Supprimer les modifications de l'index	108
7. Revenir à un état antérieur	108
8. Modifier le dernier commit	109
Les branches et les tags	
1. Les tags	113
1.1 Numérotation des versions	113
1.2 Différents types de tags	114
1.3 Création des tags	115
1.4 Création d'un tag annoté	115
1.5 Liste des tags	116
1.6 Détails d'un tag	116
1.7 Envoi des tags vers le dépôt distant	117
1.8 Suppression d'un tag	118
2. Les branches	119
2.1 Liste des branches existantes	

2.2 Création d'une branche	121
2.3 Positionnement sur une branche	122
2.4 Fusionner deux branches	123
2.4.1 L'avance rapide	125
2.4.2 Nettoyer votre dépôt	126
2.4.3 Les conflits de fusion	129
2.5 Supprimer une branche	129
2.6 Rebaser une branche dans une autre	137
	138

Partager un dépôt

1. Qu'est-ce qu'un dépôt distant ?	143
2. Créer un dépôt distant	145
2.1 Pour un nouveau projet	145
2.2 Pour un projet existant	146
3. Cloner un dépôt distant	147
4. Les protocoles d'échange	148
5. Fonctionnement interne et branches distantes	149

5.1 Les dépôts distants liés	149
5.2 Les branches distantes suivies	150
6. Envoyer ses modifications	151
7. Recevoir les modifications	153
Git-Flow : workflow d'entreprise	
1. Un système de gestion des branches	159
1.1 Les branches éternelles	160
1.1.1 La branche de production (master)	160
1.1.2 La branche de développement (develop)	160
1.2 Les branches éphémères	161
1.2.1 Les branches de versions (release)	161
1.2.2 Les branches de correctifs (hotfix)	162
1.2.3 Les branches de fonctionnalités (feature)	162
1.2.4 Plusieurs commits dans une branche éphémère ?	162
2. Exemple de workflow	163
3. Git-Flow intuitif grâce à Tower	166
3.1 Client Git et Git-Flow	

3.2 Cas pratique d'utilisation	166
	167

Les outils de Git

1. Mettre de côté des modifications avec git stash	175
2. Dépôts intégrés avec submodules	178
2.1 Ajout du dépôt intégré	180
2.2 Cloner un dépôt et ses dépôts intégrés	183
2.3 Modification des dépôts intégrés	183
2.4 Supprimer un dépôt intégré	184
2.5 Inconvénients des dépôts intégrés	185
3. Retrouver un commit erroné	186
3.1 Utilisation pratique de git bisect	187
3.2 Automatiser git bisect	188
4. Journal des références (reflog)	190
5. Les hooks	191
5.1 Les différents types de hooks	192
5.2 Comment utiliser les hooks ?	193

5.3 Exemple de hook : validation de message	194
5.4 Partager les hooks dans le dépôt	195
6. Les notes Git	195
6.1 Créer une note	196
6.2 Afficher les notes	196
6.2.1 Lister les notes	196
6.2.2 Consulter les notes d'un commit	197
6.3 Éditer une note	197
6.4 Supprimer une note	198
6.5 Envoyer les notes vers le serveur	198
Scénario de développeur indépendant	
1. But de ce chapitre	199
2. Contexte du scénario	200
3. Création du dépôt	201
4. Début du développement	201
5. Enregistrer des modifications	204

6. Bitbucket	205
6.1 Création d'un compte	207
6.2 Envoyer un dépôt local vers Bitbucket	210
6.3 Éditer un fichier sur Bitbucket	211
6.4 Récupérer les modifications du dépôt distant	212
7. Intégrer un nouveau développement	212
7.1 Vérifier son code avant l'indexation	213
7.2 Commiter le nouveau développement	214
8. Annuler les modifications d'un fichier	214
9. .gitignore : ignorer une bibliothèque	215
10. Commiter tous les fichiers ajoutés ou modifiés	218
11. Envoyer les commits au dépôt distant	219
12. Afficher les différences entre deux commits	219
13. Cloner le dépôt distant	220
14. Une branche, ça sert à quoi ?	221
15. Changer de branche	225

16. Fusionner deux branches	226
Scénario d'équipe	
1. Contexte du scénario	229
2. Aperçu du projet	230
2.1 Installation de Python	231
2.2 Récupération du dépôt	232
2.3 Installation des dépendances Python	232
2.4 Initialisation des dépôts intégrés	232
2.5 Génération des bibliothèques	233
2.6 Création du fichier de configuration	234
2.7 Création de la base	234
2.8 Création d'un compte root	235
2.9 Lancement du serveur	235
3. Installation de GitLab	236
4. Création des comptes utilisateurs	238
5. Création du projet	239

6. Attribuer des projets aux utilisateurs	242
7. Premier commit du projet	243
7.1 Rédaction du fichier .gitignore	243
7.1.1 Ignorer les bibliothèques	244
7.1.2 Ignorer les fichiers propres à la technologie	244
7.1.3 Ignorer les données sensibles	244
7.1.4 Ajouter les dépôts intégrés	245
7.1.5 Le fichier README	246
7.2 Commit du projet	246
7.3 Création de la branche develop pour Git-Flow	246
8. Phase de développement	247
8.1 Fonctionnalité graphique	247
8.2 Correctif de temps négatif	248
8.3 Intégration du correctif	249
8.4 Fonctionnalité type de tâche	250
8.5 Finalisation des graphiques	250
8.6 Finalisation des types de tâche	252
8.7 Création de la branche de version	252
8.8 Export CSV	253

8.9 Correctif de version	253
8.10 Nouvelle version stable	254
8.11 Finalisation de l'export CSV	254
9. Mise en ligne du dépôt sur GitHub	255
9.1 Création d'un compte GitHub	255
9.2 Création d'un dépôt	256
9.3 Ajout du remote au dépôt local	258
9.4 Envoi des branches	258
9.5 Le fichier LICENSE	258
9.6 Le fichier README	259

Productivité maximale avec Git

1. Alias prêts à l'emploi	261
1.1 Alias simples	261
1.1.1 git last	262
1.1.2 git aa	262
1.1.3 git bv	262
1.1.4 git ba	263
1.1.5 git bd	263
1.1.6 git ca	263

1.1.7 git cb	263
1.1.8 git cmf	264
1.1.9 git co	264
1.1.10 git di	264
1.1.11 git dc	265
1.1.12 git mnff	265
1.1.13 git st	265
1.1.14 git tg	265
1.1.15 git pu	267
1.1.16 git ss	267
1.1.17 git ssu	267
1.1.18 git sr	268
1.1.19 git srp	268
1.1.20 git sl	269
1.1.21 git sp	269
1.1.22 git sa	269
1.1.23 git sd_f	269
1.1.24 git sb	270
1.1.25 git na	270
1.1.26 git nl	270
1.1.27 git napp	270

1.1.28 git ne	271
1.1.29 git ns	271
1.1.30 git nr	271
1.1.31 git ready	272
1.2 Alias complexes	272
1.2.1 git bdate	272
1.2.2 git ll	273
1.2.3 git ld	274
1.2.4 git ls	274
1.2.5 git ln	275
1.2.6 git np	275
1.2.7 git bvn	276
1.2.8 git churn	276
1.2.9 git srr	276
1.2.10 git spr	277
1.2.11 git sdr	277
1.3 Récupérer les alias sur GitHub	277
2. Commandes prêtes à l'emploi	279
2.1 Commandes liées à la configuration	279
2.1.1 Fichier de configuration actif pour une option	

2.1.2	Afficher sa configuration	279
2.1.3	Éditer facilement un niveau de configuration	279
2.2	Commandes d'affichage	280
2.2.1	Afficher les informations techniques d'un commit	280
2.2.2	Afficher les parents des commits	281
2.2.3	Afficher les fichiers en conflit	282
2.2.4	Afficher la liste des fichiers modifiés	282
2.2.5	Afficher l'ancêtre commun	282
2.2.6	Afficher le premier commit d'une branche	283
2.2.7	Utiliser git show en masquant le diff	283
2.2.8	Vérifier une branche sur un dépôt distant	283
2.2.9	Fusionner des branches sans ancêtre commun	283
2.2.10	Afficher les dépôts distants et leur lien externe	284
2.2.11	Afficher les fichiers modifiés par un commit	284
2.2.12	Afficher le chemin du dépôt versionné	284
2.2.13	Consulter l'historique des commandes git	285
2.2.14	Afficher le nombre de commits par auteur	285
2.2.15	Afficher le nombre de commits d'un auteur	285
2.2.16	Afficher la dernière date de modification des branches	286
2.2.17	Lister les branches contenant un commit précis	286
2.2.18	Afficher l'historique avec les diff	286

2.2.19 Chercher un texte/regex dans les commits	287
2.2.20 Chercher un texte/regex dans les stashes	287
2.2.21 Lister les commits des branches	287
2.2.22 Comparer un fichier antérieur	290
2.2.23 Afficher les branches déjà fusionnées dans master	290
2.2.24 Lister les branches non mergées dans master	291
2.2.25 Lister les commits d'une branche non mergée à master	291
2.3 Commandes de manipulation	292
2.3.1 Supprimer ou inverser les modifications d'un commit	292
2.3.2 Supprimer du dépôt les fichiers déjà supprimés du projet	292
2.3.3 Retirer des modifications de l'index	293
2.3.4 Récupérer le fichier d'un autre commit	293
2.3.5 Supprimer les fichiers non suivis du répertoire	293
2.3.6 Supprimer les modifications des fichiers suivis	294
2.3.7 Supprimer une branche distante	294
Git en déploiement continu	
1. Objectifs du chapitre	295
2. Le projet	296

3. Présentation de Django	296
4. Développement de la version initiale	298
4.1 Installation	298
4.2 Création du projet	299
4.2.1 Création du projet Django	299
4.2.2 Création du fichier .gitignore	300
4.2.3 Enregistrement des bibliothèques Python	301
4.2.4 Premier commit	301
4.3 Création des applications users et articles	302
4.4 Création des modèles Django	304
4.4.1 Le modèle BaseModel	305
4.4.2 Le modèle User	306
4.4.3 Le modèle Article	307
4.5 Mise en place du module d'administration	309
4.5.1 Démarrer le serveur de développement	311
4.5.2 Création des pages utilisateur	312
4.5.3 Templates parents	312
4.5.4 Liste des articles	316
4.5.5 Page de consultation d'un article	319
4.5.6 Page "À propos"	321

5. Déploiement initial	322
5.1 Configuration des identifiants SSH	323
5.2 Création du site web Webfaction	323
5.3 Création des applications Webfaction	324
5.4 Création de la base de données	326
5.5 Externalisation du dépôt de la configuration	327
5.6 Préparer le dossier du projet et créer le dépôt	328
5.7 Configuration du dépôt en déploiement automatisé	329
5.8 Configuration du remote et premier push	329
5.9 Création de l'environnement virtuel	330
5.10 Configuration d'Apache	331
5.11 Envoi de la configuration de production	331
5.12 Exécuter les migrations	332
5.13 Synchroniser les fichiers statiques	332
5.14 Redémarrer Apache	332
6. Déploiement automatisé	333
6.1 Développement du hook dans le dépôt	333
6.2 Configuration du dépôt distant	334
7. Fonctionnalité : champ WYSIWYG pour l'article	335

7.1 Développement	335
7.2 Déploiement automatisé	340

Aide-mémoire

1. Les références	341
1.1 HEAD	341
1.2 Les branches	342
1.3 Les tags	342
1.4 Référence des ancêtres	342
2. Les commandes	343
2.1 git add	343
2.2 git archive	344
2.3 git bisect	344
2.4 git blame	345
2.5 git branch	346
2.6 git checkout	347
2.7 git cherry-pick	348
2.8 git clean	348
2.9 git clone	349

2.10 git commit	350
2.11 git config	351
2.12 git diff	352
2.13 git fetch	353
2.14 git gc	354
2.15 git help	354
2.16 git init	355
2.17 git log	355
2.18 git merge	356
2.19 git mv	357
2.20 git pull	357
2.21 git push	358
2.22 git rebase	359
2.23 git reflog	359
2.24 git remote	360
2.25 git reset	360
2.26 git revert	361
2.27 git rm	362
2.28 git show	362
2.29 git stash	363
2.30 git submodule	364

2.31 git tag	365
Index	367