

Les éléments à télécharger sont disponibles à l'adresse suivante :

<http://www.editions-eni.fr>

Saisissez la référence de l'ouvrage **EI4GIT** dans la zone de recherche et validez. Cliquez sur le titre du livre puis sur le bouton de téléchargement.

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

Chapitre 1

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

Chapitre 2 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	51
4.1 Généralités	51
4.2 Types de commandes Git.	52
4.2.1 Les commandes de porcelaine	52
4.2.2 Les commandes de plomberie	53
5. Configuration requise.	53
5.1 Configurer le nom de l'utilisateur	53
5.2 Configurer l'e-mail de l'utilisateur	54

Chapitre 3 Création d'un dépôt

1. Créer un dépôt local	55
2. Le contenu du dossier .git	56
3. Le fichier README	58
4. Markdown	59
4.1 Présentation	59
4.2 Éléments de syntaxe.	61
4.2.1 Titres	61
4.2.2 Listes non ordonnées	61
4.2.3 Listes ordonnées	61
4.2.4 Mettre en gras.	62
4.2.5 Mettre en italique.	62
4.2.6 Ligne horizontale	62

4.2.7	Code	62
4.2.8	Tableaux	63
4.2.9	Liens	63
4.2.10	Notes de bas de page	63
5.	reStructuredText	64
5.1	Présentation	64
5.2	Éléments de syntaxe	64
5.2.1	Titres	64
5.2.2	Listes autonumérotées	65
5.3	Logiciels	66
6.	Outils pour travailler avec Markdown	66
6.1	Sublime Text	66
6.2	Texts	67
6.3	Ulysses	68
7.	Configurer le dépôt local	69
7.1	Configuration minimale	69
7.2	Niveaux de configuration	70
7.2.1	Le niveau système	70
7.2.2	Le niveau utilisateur	70
7.2.3	Le niveau dépôt	71
7.3	Les paramètres configurables	71
7.3.1	Définir l'éditeur de texte	71
7.3.2	Modèle de commit	71
7.3.3	Ignorer des fichiers	72
7.3.4	Hashs abrégés	72
7.4	Définition d'alias Git	73
8.	Les options de configuration avancées	74
8.1	Pagination	74
8.2	Expressions régulières étendues	74
8.3	Séparateur de mots	74
8.4	Ancêtre commun des conflits	75
8.5	Configurer le cache de l'authentification	75

Chapitre 4

Manipulation des fichiers et commit

1. Gestion des fichiers et commit	77
2. Une histoire de hash	78
2.1 Une identification par contenu	79
2.2 Risque de collision	79
2.3 Fonctionnalité expérimentale de hash SHA2-256	80
3. Les trois zones d'un fichier	81
3.1 Le répertoire de travail	82
3.2 L'index	84
3.3 Le dépôt	85
4. Manipuler les fichiers	88
4.1 Ajouter des fichiers dans l'index	88
4.2 Déplacer ou renommer des fichiers	89
4.3 Supprimer des fichiers	90
4.4 Arrêter de suivre un fichier	91
4.5 Ignorer des fichiers	91
5. Commiter ou enregistrer des modifications	93
5.1 Effectuer un premier commit	93
5.2 Rédiger un bon message de commit	95
5.2.1 Les règles d'un message de commit	95
5.2.2 Méthode pour le titre	96
5.2.3 En quelle langue ?	98

Chapitre 5

Consultation et manipulation de l'historique

1. Lister les commits avec git log	99
1.1 Limiter le nombre de commits affichés	101
1.2 Afficher les statistiques	102
1.3 Afficher chaque commit sur une seule ligne	103
1.4 Filtrer les commits chronologiquement	103

1.5	Filtrer les commits selon les intervenants	104
1.6	Afficher le graphique des branches	105
1.7	Spécifier un format de sortie	106
1.8	Prendre en compte les merges	108
1.9	Lister les commits impactant un fichier	109
1.10	Afficher des dates plus lisibles	110
2.	Afficher les différences de contenu	110
2.1	Différences en cours dans le répertoire	111
2.2	Différences entre l'index et HEAD	112
2.3	Différences entre le répertoire de travail et HEAD	113
2.4	Différences introduites par un ou plusieurs commits	113
2.5	Différences de mots	114
2.6	Visualiser les blocs de code déplacés	115
3.	Identifier l'auteur d'une ligne de code	116
4.	Rechercher des commits avec le mode pick axe	117
5.	Supprimer les modifications du répertoire de travail	118
6.	Supprimer les modifications de l'index	119
7.	Revenir à un état antérieur	120
8.	Modifier le dernier commit	121
9.	Afficher un résumé des commits	122

Chapitre 6

Les branches et les tags

1.	Les tags	125
1.1	Numérotation des versions	125
1.2	Différents types de tags	126
1.3	Création des tags	127
1.4	Création d'un tag annoté	127
1.5	Liste des tags	128
1.6	Détails d'un tag	128

1.7	Envoi des tags vers le dépôt distant.	129
1.8	Suppression d'un tag	130
2.	Les branches	131
2.1	Liste des branches existantes	133
2.2	Création d'une branche	135
2.3	Positionnement sur une branche.	137
2.4	Fusionner deux branches	139
2.4.1	L'avance rapide	140
2.4.2	Nettoyer votre dépôt	143
2.4.3	Les conflits de fusion	143
2.5	Supprimer une branche	151
2.6	Rebaser une branche dans une autre.	152

Chapitre 7

Partager un dépôt

1.	Qu'est-ce qu'un dépôt distant ?	157
2.	Créer un dépôt distant	159
2.1	Pour un nouveau projet	159
2.2	Pour un projet existant	160
3.	Cloner un dépôt distant.	161
4.	Les protocoles d'échange	162
5.	Fonctionnement interne et branches distantes	163
5.1	Les dépôts distants liés.	164
5.2	Les branches distantes suivies	164
6.	Envoyer ses modifications	165
7.	Recevoir les modifications	167

Chapitre 8

Git-Flow : workflow d'entreprise

1. Un système de gestion des branches	173
1.1 Les branches éternelles	174
1.1.1 La branche de production (master)	174
1.1.2 La branche de développement (develop)	175
1.2 Les branches éphémères	175
1.2.1 Les branches de versions (release)	175
1.2.2 Les branches de correctifs (hotfix)	176
1.2.3 Les branches de fonctionnalités (feature)	176
1.2.4 Plusieurs commits dans une branche éphémère ?	177
2. Exemple de workflow	178
3. Git-Flow intuitif grâce à Tower	180
3.1 Client Git et Git-Flow	180
3.2 Cas pratique d'utilisation	181

Chapitre 9

Les outils de Git

1. Mettre de côté des modifications avec git stash	189
2. Dépôts intégrés avec submodules	192
2.1 Ajout du dépôt intégré	194
2.2 Cloner un dépôt et ses dépôts intégrés	197
2.3 Modification des dépôts intégrés	197
2.4 Supprimer un dépôt intégré	198
2.5 Inconvénients des dépôts intégrés	199
3. Retrouver un commit erroné	200
3.1 Utilisation pratique de git bisect	201
3.2 Automatiser git bisect	202
4. Journal des références (reflog)	204

5. Les hooks	205
5.1 Les différents types de hooks	206
5.2 Comment utiliser les hooks ?	207
5.3 Exemple de hook : validation de message	208
5.4 Partager les hooks dans le dépôt	209
6. Les notes Git	209
6.1 Créer une note	210
6.2 Afficher les notes	210
6.2.1 Lister les notes	210
6.2.2 Consulter les notes d'un commit	211
6.3 Éditer une note	211
6.4 Supprimer une note	212
6.5 Envoyer les notes vers le serveur	212

Chapitre 10

Scénario de développeur indépendant

1. But de ce chapitre	213
2. Contexte du scénario	214
3. Création du dépôt	215
4. Début du développement	215
5. Enregistrer des modifications	218
6. Bitbucket	219
6.1 Création d'un compte	221
6.2 Envoyer un dépôt local vers Bitbucket	225
6.3 Éditer un fichier sur Bitbucket	226
6.4 Récupérer les modifications du dépôt distant	227
7. Intégrer un nouveau développement	228
7.1 Vérifier son code avant l'indexation	229
7.2 Commiter le nouveau développement	229
8. Annuler les modifications d'un fichier	230

9. .gitignore : ignorer une bibliothèque	230
10. Commiter tous les fichiers ajoutés ou modifiés	233
11. Envoyer les commits au dépôt distant	235
12. Afficher les différences entre deux commits	235
13. Cloner le dépôt distant	236
14. Une branche, ça sert à quoi ?	237
15. Changer de branche	241
16. Fusionner deux branches	242

Chapitre 11 Scénario d'équipe

1. Contexte du scénario	245
2. Aperçu du projet	246
2.1 Installation de Python	247
2.2 Récupération du dépôt	248
2.3 Installation des dépendances Python	248
2.4 Initialisation des dépôts intégrés	248
2.5 Génération des bibliothèques	249
2.6 Création du fichier de configuration	250
2.7 Création de la base	250
2.8 Création d'un compte root	251
2.9 Lancement du serveur	251
3. Installation de GitLab	252
4. Création des comptes utilisateurs	254
5. Création du projet	255
6. Attribuer des projets aux utilisateurs	258

7. Premier commit du projet	259
7.1 Rédaction du fichier .gitignore	259
7.1.1 Ignorer les bibliothèques	260
7.1.2 Ignorer les fichiers propres à la technologie	260
7.1.3 Ignorer les données sensibles	260
7.1.4 Ajouter les dépôts intégrés	261
7.1.5 Le fichier README	262
7.2 Commit du projet	262
7.3 Création de la branche develop pour Git-Flow	262
8. Phase de développement	263
8.1 Fonctionnalité graphique	263
8.2 Correctif de temps négatif	264
8.3 Intégration du correctif	265
8.4 Fonctionnalité type de tâche	266
8.5 Finalisation des graphiques	266
8.6 Finalisation des types de tâche	268
8.7 Création de la branche de version	268
8.8 Export CSV	269
8.9 Correctif de version	269
8.10 Nouvelle version stable	270
8.11 Finalisation de l'export CSV	271
9. Mise en ligne du dépôt sur GitHub	271
9.1 Création d'un compte GitHub	272
9.2 Création d'un dépôt	274
9.3 Ajout du remote au dépôt local	274
9.4 Envoi des branches	275
9.5 Le fichier LICENSE	275
9.6 Le fichier README	275

Chapitre 12

Productivité maximale avec Git

1. Alias prêts à l'emploi.	277
1.1 Alias simples.	278
1.1.1 git last.	278
1.1.2 git aa.	278
1.1.3 git bv.	278
1.1.4 git ba.	279
1.1.5 git bd.	279
1.1.6 git bdp.	279
1.1.7 git ca.	280
1.1.8 git cb.	280
1.1.9 git cmf.	280
1.1.10 git co.	281
1.1.11 git di.	281
1.1.12 git dc.	281
1.1.13 git mnff.	282
1.1.14 git st.	282
1.1.15 git tg.	284
1.1.16 git pu.	284
1.1.17 git ss.	284
1.1.18 git ssu.	284
1.1.19 git sr.	285
1.1.20 git srp.	285
1.1.21 git sl.	285
1.1.22 git sp.	285
1.1.23 git sa.	286
1.1.24 git sd_f.	286
1.1.25 git sb.	286
1.1.26 git na.	287
1.1.27 git nl.	287
1.1.28 git napp.	287
1.1.29 git ne.	287

1.1.30	git ns	288
1.1.31	git nr	288
1.1.32	git ready	288
1.2	Alias complexes	289
1.2.1	git bnew	289
1.2.2	git bold	290
1.2.3	git ll	290
1.2.4	git ld	291
1.2.5	git ls	292
1.2.6	git ln	293
1.2.7	git slv	293
1.2.8	git np	294
1.2.9	git bvn	294
1.2.10	git churn	295
1.2.11	git srr	295
1.2.12	git spr	295
1.2.13	git sar	296
1.2.14	git sdr	296
1.3	Récupérer les alias sur GitHub	296
2.	Commandes prêtes à l'emploi	299
2.1	Commandes liées à la configuration	299
2.1.1	Fichier de configuration actif pour une option	299
2.1.2	Afficher sa configuration	299
2.1.3	Éditer facilement un niveau de configuration	300
2.2	Commandes d'affichage	300
2.2.1	Afficher les informations techniques d'un commit	300
2.2.2	Afficher les parents des commits	301
2.2.3	Afficher les fichiers en conflit	302
2.2.4	Afficher la liste des fichiers modifiés	302
2.2.5	Afficher l'ancêtre commun	302
2.2.6	Afficher le premier commit d'une branche	303
2.2.7	Utiliser git show en masquant le diff	303
2.2.8	Vérifier une branche sur un dépôt distant	303

2.2.9 Fusionner des branches sans ancêtre commun.	303
2.2.10Afficher les dépôts distants et leur lien externe	304
2.2.11Afficher les fichiers modifiés par un commit	304
2.2.12Afficher le chemin du dépôt versionné	304
2.2.13Consulter l'historique des commandes git	305
2.2.14Afficher le nombre de commits par auteur.	305
2.2.15Afficher le nombre de commits d'un auteur.	305
2.2.16Afficher la dernière date de modification des branches. .	306
2.2.17Lister les branches contenant un commit précis	306
2.2.18Afficher l'historique avec les diff	307
2.2.19Chercher un texte/regex dans les commits.	307
2.2.20Chercher un texte/regex dans les stashes	307
2.2.21Lister les commits des branches.	307
2.2.22Comparer un fichier antérieur	310
2.2.23Afficher les branches déjà fusionnées dans master	310
2.2.24Lister les branches non mergées dans master	311
2.2.25Lister les commits d'une branche non mergée à master .	311
2.3 Commandes de manipulation	312
2.3.1 Supprimer ou inverser les modifications d'un commit . .	312
2.3.2 Supprimer du dépôt les fichiers déjà supprimés du projet	312
2.3.3 Retirer des modifications de l'index.	313
2.3.4 Récupérer le fichier d'un autre commit.	313
2.3.5 Supprimer les fichiers non suivis du répertoire	313
2.3.6 Supprimer les modifications des fichiers suivis	314
2.3.7 Supprimer une branche distante	314

Chapitre 13

Git en déploiement continu

1. Objectifs du chapitre	315
2. Le projet.	316
3. Présentation de Django	316

4. Développement de la version initiale	318
4.1 Installation	318
4.2 Création du projet	319
4.2.1 Création du projet Django	319
4.2.2 Création du fichier .gitignore	320
4.2.3 Enregistrement des bibliothèques Python	321
4.2.4 Premier commit	321
4.3 Création des applications users et articles	322
4.4 Création des modèles Django	324
4.4.1 Le modèle BaseModel	325
4.4.2 Le modèle User	326
4.4.3 Le modèle Article	327
4.5 Mise en place du module d'administration	329
4.5.1 Démarrer le serveur de développement	331
4.5.2 Création des pages utilisateur	332
4.5.3 Templates parents	332
4.5.4 Liste des articles	336
4.5.5 Page de consultation d'un article	339
4.5.6 Page "À propos"	341
5. Déploiement initial	342
5.1 Configuration des identifiants SSH	343
5.2 Création du site web Webfaction	343
5.3 Création des applications Webfaction	344
5.4 Création de la base de données	346
5.5 Externalisation du dépôt de la configuration	347
5.6 Préparer le dossier du projet et créer le dépôt	348
5.7 Configuration du dépôt en déploiement automatisé	349
5.8 Configuration du remote et premier push	349
5.9 Création de l'environnement virtuel	350
5.10 Configuration d'Apache	351
5.11 Envoi de la configuration de production	351
5.12 Exécuter les migrations	352
5.13 Synchroniser les fichiers statiques	352

5.14 Redémarrer Apache	352
6. Déploiement automatisé	353
6.1 Développement du hook dans le dépôt.	353
6.2 Configuration du dépôt distant.	354
7. Fonctionnalité : champ WYSIWYG pour l'article	355
7.1 Développement.	355
7.2 Déploiement automatisé	360

Chapitre 14

Aide-mémoire

1. Les références	361
1.1 HEAD	361
1.2 Les branches	362
1.3 Les tags	362
1.4 Référence des ancêtres	362
2. Les commandes	363
2.1 git add	363
2.2 git archive	364
2.3 git bisect	364
2.4 git blame	365
2.5 git branch	366
2.6 git checkout.	367
2.7 git cherry-pick.	368
2.8 git clean	368
2.9 git clone	369
2.10 git commit.	370
2.11 git config	371
2.12 git diff	372
2.13 git fetch	373
2.14 git gc	373
2.15 git help.	374

2.16	git init	374
2.17	git log	375
2.18	git merge	376
2.19	git mv	376
2.20	git pull	377
2.21	git push	377
2.22	git rebase	378
2.23	git reflog	379
2.24	git remote	379
2.25	git reset	380
2.26	git revert	381
2.27	git rm	381
2.28	git show	382
2.29	git stash	382
2.30	git submodule	383
2.31	git tag	384
3.	GitHub	385
3.1	Gestion des dépôts	385
3.2	GitHub-Flow (fork et pull request)	388
3.3	Les "issues" GitHub	392
	Index	403