Git Commands Overview

Git הוא כלי מאוד רב-שימושי שמאפשר ניהול גרסאות ושיתוף פעולה בפרויקטים של פיתוח. יש בו הרבה פקודות חשובות שצריך להכיר כדי לעבוד ביעילות. אני אפרט כאן את הפקודות העיקריות של Git, כדי שתוכל לעבוד בצורה מסודרת ומקצועית עם כל פרויקט.  
  
1. git init  
תפקיד: יצירת מאגר Git חדש בספריה המקומית שלך.  
- פקודה זו מתחילה מעקב אחרי שינויים בקבצים ומכינה אותם לניהול גרסאות.  
```  
git init  
```  
  
2. git clone  
תפקיד: העתקת מאגר מרוחק למחשב המקומי.  
- כשיש לך מאגר מרוחק ב-GitHub, GitLab או כל שרת אחר, אתה יכול להוריד עותק שלו למחשב שלך.  
```  
git clone <URL>  
```  
  
3. git status  
תפקיד: בדיקת מצב המאגר.  
- הפקודה הזו תראה לך איזה קבצים שונו, נוספו, או נמחקו מאז הפעם האחרונה שביצעת commit.  
```  
git status  
```  
  
4. git add  
תפקיד: הוספת קבצים ל"אזור הבמה" (staging area).  
- לפני ביצוע commit, צריך להוסיף את הקבצים ששונו ל-staging area כדי ש-Git יעקוב אחריהם ויבצע commit.  
```  
git add <file>  
# להוסיף את כל הקבצים בבת אחת:  
git add .  
```  
  
5. git commit  
תפקיד: שמירת השינויים שבוצעו בקבצים בגרסה מסוימת.  
- אחרי שהוספת קבצים ל-staging, אתה יכול לשמור את השינויים הללו בהיסטוריה של ה-Git. חשוב לתת תיאור ברור (message) לכל commit.  
```  
git commit -m "הודעה שמתארת את השינויים"  
```  
  
6. git pull  
תפקיד: משיכת שינויים מהמאגר המרוחק לתוך המאגר המקומי.  
- פקודה זו מבטיחה שהקוד שלך מעודכן ומסונכרן עם המאגר המרכזי.  
```  
git pull origin main  
```  
  
7. git push  
תפקיד: דחיפת השינויים מהמחשב המקומי אל המאגר המרוחק.  
- אחרי שביצעת commit, אתה יכול לשלוח את השינויים שלך למאגר המרכזי, כך שאחרים יוכלו לראות ולעבוד איתם.  
```  
git push origin main  
```  
  
8. git branch  
תפקיד: יצירת ענף חדש או צפייה בענפים קיימים.  
- Git מאפשר עבודה עם ענפים (branches) נפרדים כדי לבודד שינויים, כך שכל פיתוח חדש לא ישפיע על הקוד המרכזי עד שתבצע מיזוג (merge).  
```  
git branch # מציג את כל הענפים הקיימים  
git branch <branch\_name> # יוצר ענף חדש  
```  
  
9. git checkout  
תפקיד: מעבר בין ענפים או חזרה לגרסה קודמת.  
- כדי לעבור בין ענפים או לחזור לגרסה ישנה בקוד.  
```  
git checkout <branch\_name> # מעבר לענף אחר  
git checkout <commit\_id> # מעבר לגרסה קודמת על פי מזהה commit  
```  
  
10. git merge  
תפקיד: מיזוג שינויים מענף אחד לענף אחר.  
- כשסיימת לעבוד על פיצ'ר חדש בענף נפרד, אתה יכול למזג את השינויים חזרה לענף המרכזי.  
```  
git merge <branch\_name>  
```  
  
11. git log  
תפקיד: הצגת היסטוריית השינויים (commits) שבוצעו במאגר.  
- Git שומר תיעוד של כל השינויים, ופקודה זו מציגה את היסטוריית ה-commit, כולל שמות המבצעים ותיאורים.  
```  
git log  
```  
  
12. git remote  
תפקיד: עבודה עם מאגרים מרוחקים.  
- פקודה זו מנהלת את הקשרים בין המאגר המקומי שלך למאגר המרוחק.  
```  
git remote add origin <URL> # הוספת קישור למאגר מרוחק  
git remote -v # הצגת רשימת מאגרים מרוחקים  
```  
  
13. git stash  
תפקיד: שמירת שינויים זמניים מבלי לבצע commit.  
- אם אתה באמצע עבודה וצריך לעבור למשימה אחרת, אתה יכול לשמור את השינויים מבלי לבצע commit ולחזור אליהם מאוחר יותר.  
```  
git stash # שמירת השינויים  
git stash pop # שליפת השינויים מה-Stash  
```  
  
14. git rebase  
תפקיד: החלת שינויים מענף אחד על ראש ענף אחר.  
- פקודה זו עוזרת לסדר מחדש את ההיסטוריה של ה-commits כדי לשמור על היסטוריה נקייה וישרה.  
```  
git rebase <branch\_name>  
```  
  
15. git reset  
תפקיד: ביטול commit או הסרת קבצים מה-Staging.  
- אם טעית ורוצה לחזור למצב קודם, אפשר להשתמש בפקודה זו כדי לאפס את המאגר למצב קודם.  
```  
git reset --soft <commit\_id> # מחזיר רק את ה-commit מבלי לשנות את הקבצים  
git reset --hard <commit\_id> # מחזיר את ה-commit ואת הקבצים למצב שהיה ב-commit הספציפי  
```  
  
סיכום:  
- `git init`: יצירת מאגר חדש.  
- `git clone`: העתקת מאגר מרוחק למקומי.  
- `git status`: הצגת מצב הקבצים במאגר.  
- `git add`: הוספת קבצים ל-staging לקראת commit.  
- `git commit`: שמירת השינויים במאגר.  
- `git pull`: משיכת שינויים מהמאגר המרוחק.  
- `git push`: דחיפת שינויים למאגר המרוחק.  
- `git branch`: עבודה עם ענפים.  
- `git checkout`: מעבר בין ענפים או חזרה לגרסה קודמת.  
- `git merge`: מיזוג ענפים.  
- `git log`: הצגת היסטוריית commits.  
- `git remote`: ניהול מאגרים מרוחקים.  
- `git stash`: שמירת שינויים זמניים.  
- `git rebase`: סידור מחדש של commits.  
- `git reset`: ביטול commit או הסרת קבצים מה-staging.  
  
Git הוא כלי חזק מאוד לניהול גרסאות ומאפשר שיתוף פעולה פורה ויעיל בפרויקטים של פיתוח.