**מדריך ל-Git גיט – ניהול גרסאות קוד:**

<https://www.codebrain.co.il/%D7%9E%D7%93%D7%A8%D7%99%D7%9A-%D7%9C-git-%D7%A0%D7%99%D7%94%D7%95%D7%9C-%D7%92%D7%A8%D7%A1%D7%90%D7%95%D7%AA-%D7%A7%D7%95%D7%93/>

פקודות git באמצעות command line:

git --version

הגדרת משתמש:

git config --global user.name "David"

הגדרת מייל:

git config --global user.email "david@example.com"

בדיקת הגדרות:

git config --list

ניקוי מסך:

clear

שינוי סיפריה:

cd /c/xampp/htdocs/myRepo

ליצור את ה-Repository שלנו על ידי הפקודה git init

git init

רשימת כל הקבצים כולל קבצים נסתרים

Ls -a

רשימת כל הקבצים לא כולל קבצים נסתרים

Ls

רשימת כל המאגרים כולל אלו המרוחקים

Git remote -v

**Working Copy**

ה-Working Copy זוהי התחנה המסמלת את הקבצים אשר נמצאים בתיקיית הפרויקט שלנו, על המחשב שלנו.  
כאשר אנו עובדים על קובץ מסוים ושמרנו את השינויים, ה-Working Copy שלנו השתנה.

**Staging Copy**

כאשר ביצענו שינוי, אנו צריכים "להוסיף" את השינויים שביצענו למעקב של Git.  
ה-Staging Copy הוא האיזור שבו נמצאים הקבצים שהועלו. יש לשים לב – עדיין לא עשינו Commit לקבצים אלו!

**Committed Copy**

ה-Committed Copy זוהי התחנה בה נמצאים הקבצים "שעברו"Commit.  
מרגע שביצענו Commit לקבצים, השינוי "נדחף" אל ה-Repository ונוסף Commit חדש למערכת.  
במילים אחרות, נוצר "העתק" של כל הקובץ עם השינוי החדש בתוספת הודעה קצרה המפרטת על השינוי שביצענו.

באפשרותנו להבין מה קורה ב-Repository שלנו על ידי הפקודה git status:

Git status

הוספת קובץ ל Staging Copy

Git add filename

הוספת כל הקבצים ל Staging Copy

git add .

סביר להניח כי ברוב הפרויקטים שלנו, ישנם קבצים/תיקיות שנרצה ש-Git לא תעקוב אחריהם והם יועלו ל-Repository שלנו.  
קובץ gitignore. הוא הפיתרון.  
יצירת הקובץ מתבצעת על ידי הפקודה touch:

touch .gitignore

כעת נוכל להבחין כי Git יצרה עבורנו קובץ חדש בתיקיית הפרויקט שלנו (סביר להניח שהקובץ ללא שם).  
נערוך את הקובץ ונוסיף את הקבצים/התיקיות שנרצה ש-Git תתעלם מהן באופן הבא:

.cms

/db.php

Copy

השורה הראשונה תגרום להתעלמות מתיקייה שלמה בשם cms והשורה השנייה תגרום להתעלמות מקובץ בשם db.php.  
ניתן ולפעמים אף רצוי להוסיף את קובץ gitignore ל-Repository שלנו, כל מקרה לגופו.

החזרת קובץ ל- Working Copy

git rm --cached index.html

ביצוע Commit

בין המרכאות נכתוב את מהות ה-Commit הנוכחי, מה בדיוק בוצע.

git commit -m "init commit"

לבצע Commit "מהיר", ולדלג על הוספת הקובץ על ידי פקודת git add, נוכל לעשות זאת על ידי הפקודה הבאה:

\* הערה:  לא מומלץ לעשות זאת מאחר ואין לנו את האופציה לעשות unsgate לקובץ

git commit -am "index.html"

כדי לצפות ברשימת הקומיטים שבוצעו במאגר הנוכחי:

git log

על מנת לצפות בקומיטים שנוצרו על ידי משתמש מסוים:

git log --author "David"

לצפות ב10 קומיטים אחרונים:

git log -n 10

במידה וביצענו שינוי ב-Working Copy הוספנו אותו ונרצה להשוות בין קובץ הנוכחי לקובץ שנמצא במאגר שלנו, נוכל לעשות זאת עם הפקודה הבאה:

git diff --staged

על מנת להשוות בין הגרסא שיש לנו ב-Working Copy לבין ה-Repostiory נשתמש בפקודה הבאה:

git diff

על מנת להחזיר קובץ מסוים לקומיט כלשהו, נשתמש בפקודת checkout באופן הבא:

git checkout 45bee -- index.html

על מנת למחוק קובץ מה-Working Copy שלנו יש להשתמש בפקודה git rm באופן הבא:

git rm index.html

על מנת להעביר קובץ, בדוגמא זו לתיקייה כלשהי, יש להשתמש בפקודה git mv באופן הבא:

git mv index.html folder/index.html

פקודת Clone מאפשרת לנו ליצור עותק של Repostiory מרוחק על המחשב שלנו.  
בדוגמא זו המאגר הוא של Bootstrap והוא נמצא ב-GitHub.  
על מנת לקבל העתק של המאגר נפתח את חלון ה-Bash ונשתמש בפקודה הבאה:

git clone https://github.com/twbs/bootstrap.git

יש לנו מאגר מקומי ומרוחק, על מנת לדחוף את המאגר המקומי למאגר המרוחק נצטרך

**להגדיר** את ה-Remote שלנו על ידי הפקודה הבאה:

**git** **remote add origin** https://github.com/YourGitHubUserName/YourGitHubRepositoryName.git

על מנת לדחוף את המאגר המקומי שלנו למאגר המרוחק נשתמש בפקודה הבאה:

git push -u origin master

אז מהם בידיוק ענפים (branches)?  
ענף הוא "גרסא" של ה-Repostiory שלנו.  
לצורך דוגמא, נאמר ופיתחנו מערכת כלשהי, הענף הראשי נקרא master והוא "מכיל" המערכת בגרסתא "הקלאסית".  
אם נרצה להעתיק את המערכת ולהוסיף לה יכולות מיוחדות/מותאמות אישית ללקוח כלשהו או בכלל לעבוד על גרסא חדשה למערכת, כדאי לנו ליצור ענף חדש.

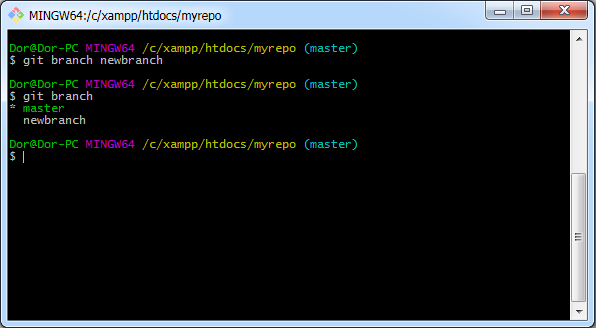
לאחר סיום העבודה ניתן גם למזג (merge) בין הענפים.

על מנת ליצור ענף חדש, נשתמש בפקודה הבאה:

git branch newbranch

על מנת לקבל את רשימת הענפים שיש ברשותינו:

git branch



כפי שניתן לראות בתמונה, ענף אחד צבוע בירוק ובעל כוכבית. המשמעות היא שהענף הזו הוא הענף שאנו עובדים

עליו כרגע.  
איך נעבור לענף אחר?

git checkout newbranch

ניתן ליצור ענף חדש ולעבור עליו באמצעות פקודה אחת בלבד:

git checkout -b version2

כיצד נמחק ענף?

git branch -D newbranch

לאחר שלמדנו על עבודה עם שרת מרוחק וענפים, נשאלת השאלה מה יקרה אם ביצענו clone למאגר מרוחק, ביצענו שינויים כלשהם אבל הענף המקורי (זה שבשרת המרוחק) עודכן?  
אנו בוודאי נרצה "למשוך" אלינו את השינויים.  
פקודת git pull תעשה זאת:

git pull origin master

origin הוא הכינוי לשרת המרוחק שלנו

master זהו השם של הענף אותו נרצה

pwd – הצג מיקום נוכחי.

$ cd /c/mk – שנה מיקום לספריה mk

$ cd /c/"test eli" - שנה סיפריה ל- eli test (בגלל הרווח יש לשים "")

**מדריך ל-Git גיט – ניהול גרסאות קוד:**

<https://www.codebrain.co.il/%D7%9E%D7%93%D7%A8%D7%99%D7%9A-%D7%9C-git-%D7%A0%D7%99%D7%94%D7%95%D7%9C-%D7%92%D7%A8%D7%A1%D7%90%D7%95%D7%AA-%D7%A7%D7%95%D7%93/>

פקודות git באמצעות command line:

git --version

הגדרת משתמש:

git config --global user.name "David"

הגדרת מייל:

git config --global user.email "david@example.com"

בדיקת הגדרות:

git config --list

ניקוי מסך:

clear

שינוי סיפריה:

cd /c/xampp/htdocs/myRepo

ליצור את ה-Repository שלנו על ידי הפקודה git init

git init

רשימת כל הקבצים כולל קבצים נסתרים

Ls -a

רשימת כל הקבצים לא כולל קבצים נסתרים

Ls

רשימת כל המאגרים כולל אלו המרוחקים

Git remote -v

**Working Copy**

ה-Working Copy זוהי התחנה המסמלת את הקבצים אשר נמצאים בתיקיית הפרויקט שלנו, על המחשב שלנו.  
כאשר אנו עובדים על קובץ מסוים ושמרנו את השינויים, ה-Working Copy שלנו השתנה.

**Staging Copy**

כאשר ביצענו שינוי, אנו צריכים "להוסיף" את השינויים שביצענו למעקב של Git.  
ה-Staging Copy הוא האיזור שבו נמצאים הקבצים שהועלו. יש לשים לב – עדיין לא עשינו Commit לקבצים אלו!

**Committed Copy**

ה-Committed Copy זוהי התחנה בה נמצאים הקבצים "שעברו"Commit.  
מרגע שביצענו Commit לקבצים, השינוי "נדחף" אל ה-Repository ונוסף Commit חדש למערכת.  
במילים אחרות, נוצר "העתק" של כל הקובץ עם השינוי החדש בתוספת הודעה קצרה המפרטת על השינוי שביצענו.

באפשרותנו להבין מה קורה ב-Repository שלנו על ידי הפקודה git status:

Git status

הוספת קובץ ל Staging Copy

Git add filename

הוספת כל הקבצים ל Staging Copy

git add .

סביר להניח כי ברוב הפרויקטים שלנו, ישנם קבצים/תיקיות שנרצה ש-Git לא תעקוב אחריהם והם יועלו ל-Repository שלנו.  
קובץ gitignore. הוא הפיתרון.  
יצירת הקובץ מתבצעת על ידי הפקודה touch:

touch .gitignore

כעת נוכל להבחין כי Git יצרה עבורנו קובץ חדש בתיקיית הפרויקט שלנו (סביר להניח שהקובץ ללא שם).  
נערוך את הקובץ ונוסיף את הקבצים/התיקיות שנרצה ש-Git תתעלם מהן באופן הבא:

.cms

/db.php

Copy

השורה הראשונה תגרום להתעלמות מתיקייה שלמה בשם cms והשורה השנייה תגרום להתעלמות מקובץ בשם db.php.  
ניתן ולפעמים אף רצוי להוסיף את קובץ gitignore ל-Repository שלנו, כל מקרה לגופו.

החזרת קובץ ל- Working Copy

git rm --cached index.html

ביצוע Commit

בין המרכאות נכתוב את מהות ה-Commit הנוכחי, מה בדיוק בוצע.

git commit -m "init commit"

לבצע Commit "מהיר", ולדלג על הוספת הקובץ על ידי פקודת git add, נוכל לעשות זאת על ידי הפקודה הבאה:

\* הערה:  לא מומלץ לעשות זאת מאחר ואין לנו את האופציה לעשות unsgate לקובץ

git commit -am "index.html"

כדי לצפות ברשימת הקומיטים שבוצעו במאגר הנוכחי:

git log

על מנת לצפות בקומיטים שנוצרו על ידי משתמש מסוים:

git log --author "David"

לצפות ב10 קומיטים אחרונים:

git log -n 10

במידה וביצענו שינוי ב-Working Copy הוספנו אותו ונרצה להשוות בין קובץ הנוכחי לקובץ שנמצא במאגר שלנו, נוכל לעשות זאת עם הפקודה הבאה:

git diff --staged

על מנת להשוות בין הגרסא שיש לנו ב-Working Copy לבין ה-Repostiory נשתמש בפקודה הבאה:

git diff

על מנת להחזיר קובץ מסוים לקומיט כלשהו, נשתמש בפקודת checkout באופן הבא:

git checkout 45bee -- index.html

על מנת למחוק קובץ מה-Working Copy שלנו יש להשתמש בפקודה git rm באופן הבא:

git rm index.html

על מנת להעביר קובץ, בדוגמא זו לתיקייה כלשהי, יש להשתמש בפקודה git mv באופן הבא:

git mv index.html folder/index.html

פקודת Clone מאפשרת לנו ליצור עותק של Repostiory מרוחק על המחשב שלנו.  
בדוגמא זו המאגר הוא של Bootstrap והוא נמצא ב-GitHub.  
על מנת לקבל העתק של המאגר נפתח את חלון ה-Bash ונשתמש בפקודה הבאה:

git clone https://github.com/twbs/bootstrap.git

יש לנו מאגר מקומי ומרוחק, על מנת לדחוף את המאגר המקומי למאגר המרוחק נצטרך

להגדיר את ה-Remote שלנו על ידי הפקודה הבאה:

**git** **remote add origin** https://github.com/YourGitHubUserName/YourGitHubRepositoryName.git

על מנת לדחוף את המאגר המקומי שלנו למאגר המרוחק נשתמש בפקודה הבאה:

git push -u origin master

אז מהם בידיוק ענפים (branches)?  
ענף הוא "גרסא" של ה-Repostiory שלנו.  
לצורך דוגמא, נאמר ופיתחנו מערכת כלשהי, הענף הראשי נקרא master והוא "מכיל" המערכת בגרסתא "הקלאסית".  
אם נרצה להעתיק את המערכת ולהוסיף לה יכולות מיוחדות/מותאמות אישית ללקוח כלשהו או בכלל לעבוד על גרסא חדשה למערכת, כדאי לנו ליצור ענף חדש.

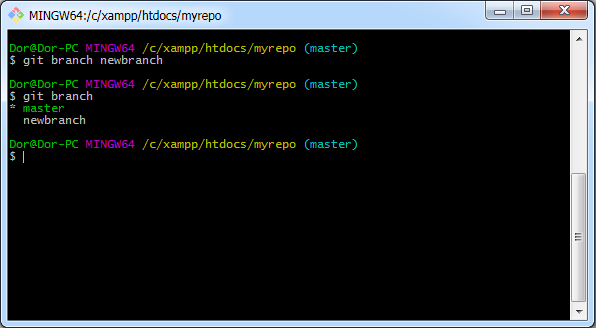
לאחר סיום העבודה ניתן גם למזג (merge) בין הענפים.

על מנת ליצור ענף חדש, נשתמש בפקודה הבאה:

git branch newbranch

על מנת לקבל את רשימת הענפים שיש ברשותינו:

git branch



כפי שניתן לראות בתמונה, ענף אחד צבוע בירוק ובעל כוכבית. המשמעות היא שהענף הזו הוא הענף שאנו עובדים

עליו כרגע.  
איך נעבור לענף אחר?

git checkout newbranch

ניתן ליצור ענף חדש ולעבור עליו באמצעות פקודה אחת בלבד:

git checkout -b version2

כיצד נמחק ענף?

git branch -D newbranch

לאחר שלמדנו על עבודה עם שרת מרוחק וענפים, נשאלת השאלה מה יקרה אם ביצענו clone למאגר מרוחק, ביצענו שינויים כלשהם אבל הענף המקורי (זה שבשרת המרוחק) עודכן?  
אנו בוודאי נרצה "למשוך" אלינו את השינויים.  
פקודת git pull תעשה זאת:

git pull origin master

origin הוא הכינוי לשרת המרוחק שלנו

master זהו השם של הענף אותו נרצה "למשוך"

pwd – הצג מיקום נוכחי.

$ cd /c/mk – שנה מיקום לספריה mk

$ cd /c/"test eli" - שנה סיפריה ל- eli test (בגלל הרווח יש לשים "")

echo "# GitNS2" >> README.md

git init

git add README.md

git commit -m "first commit"

git remote add origin https://github.com/elibgu/GitNS2.git

git push -u origin master