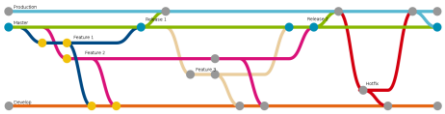


משימת למידה 2: אינטגרציה ועבודה משותפת

כעת אתם כבר מבינים איך לעבוד על פרויקט משלכם ב- git. במשימה זו תתנסו גם בעבודה על פרויקט משותף.

העבודה במשימת הלימוד בזו הינה בזוגות, מצאו בעזרת המדריך חברים נוספים שסיימו והתחילו את המשימה.

אנו נשתמש בצבע שונה לכל משתמש על מנת שההסבר יהיה ברור.

בירוק נציין את הפעולות שעל חבר צוות 1 לבצע.
בכתום נציין את הפעולות שעל חבר צוות 2 לבצע.

חלק א': יצירת פרויקט משותף

- צרו repository חדש ב GitLab וקראו לו Team_project
- גשו בתוך הrepo ל settings->Members, בתיבת People רשמו את חבר הצוות הירוק(אפשר באמצעות כתיבת השם או מייל) ותנו לו הרשאות Master (קריאה למי שרוצה להעמיק: https://goo.gl/ihCVFS)
- בדקו שה repository נוסף.

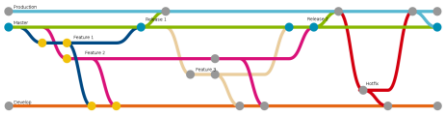
יפה, יש לכם פרויקט משותף. עשו clone לשני המחשבים כך שהפרויקט הריק יהיה קיים גם מקומית.

הוסיפו לתיקייה המקומית את הקובץ School.cpp והעלו אותו ל- repository המשותף.

קראו על הפקודה:

- git pull

- בצעו pull לפרויקט המכיל כרגע את הקובץ School.cpp כך שיהיה לכל מחשב עותק מקומי.



צרו פרויקט מקומי ב VS והריצו אותו.

מוכר לכם מאיפשהו? אז נכון אתם גם כתבתם משהו דומה, אך כנראה שה"כתום" לא עבר קורס עקרונות מתקדמים כי הוא שכח משהו חשוב, והקוד שלו איום ונורא! למרות שהוא עובד...

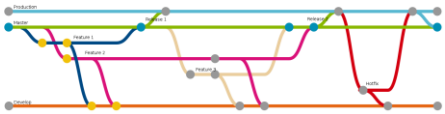
נרצה כעת לשפר את הקוד. עבודה כזאת של "סידור" הקוד או ארגונו מחדש נקראת refactoring.
(הבעיה העיקרית בקוד שלפניכם היא שחסרה חלוקה למחלקות).

- הוסיפו חלוקת מחלקות בתקייה המקומית
  - אמורים להיות לכם כעת 5 קבצים בתיקייה המקומית.(בדקו שהפרויקט עובד כמובן)
  - Course.cpp, Course.h
  - Student.cpp, Student.h
  - School.cpp

- השתמשו ב git diff כדי לראות את השינויים
  - חקרו איך ניתן לראות את כל השינויים מבלי להקיש אנטר כל הזמן (שימוש בדגלים)
  - השינויים אמורים להיות – 4 קבצים שנוספו וכל השינויים הטקסטואליים בתוך School.cpp

- דחפו את השינויים לשרת המרוחק כמו שלמדנו בעבר.
- בצעו pull.

מעולה, כעת יש לכל אחד מהצדדים repo מקומי שמכיל 5 קבצים.

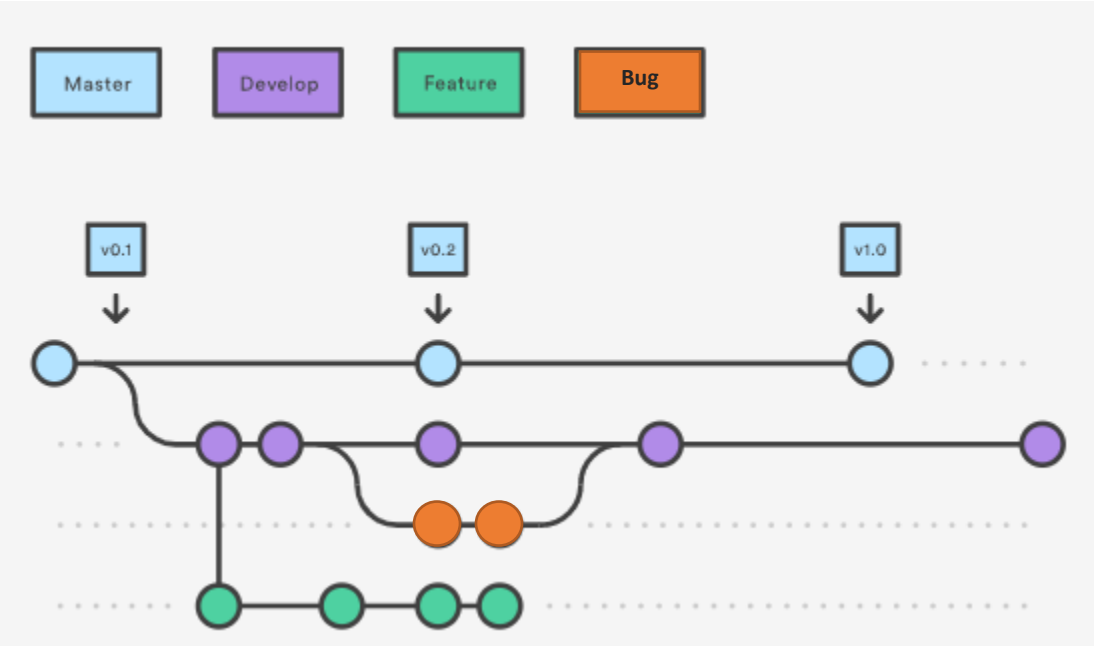


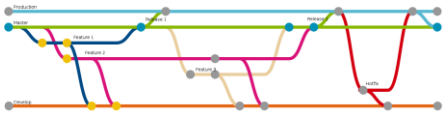
חלק ב': Branch

כעת נרצה להוסיף עוד פיצ'רים לתוכנית בית הספר.
כפי שראינו בשיעור, היכולות של עבודה במקביל קריטית לניהול הגרסאות בצורה מיטבית ונוחה, לשם כך נשתמש בענפים (branch).

חשוב!
יצירת ענף היא כלי חיוני בפרויקטים גדולים או פרויקטים בהם יש מורכבות גדולה ומפתחים רבים (הרבה פרויקטי קוד פתוח הם כאלה, לדוגמא: אנדרואיד, גוגל כרום, MySQL ועוד..).

Git היא מערכת גמישה למדי וכך גם השימוש בענפים. כלומר, כל אחד יכול להשתמש בהם בצורה שונה, דבר שעשוי להוביל לבעיות רבות כאשר עובדים בצוות.
לכן, נהוג להחליט על דרך ולדבוק בה (קונבנציה). קיימות קונבנציות branch רבות, אנו נלמד אחת מרכזית.
נסביר אותה באמצעות התמונה הבאה(הצבעים נועדו לצורך ההסבר, ואין להם משמעות נוספת):





Master – ענף המייצג את הגרסה הסופית (הסגורה) לאחר בדיקות (בדומה לגרסה שמשחררים ללקוח, במקרה שלנו הקוד שיוגש למדריך/ה).

Develop – ענף ראשי לפיתוח אליו ממזגים את כל הענפים שעובדים עליהם, ענף זה ימוזג ל-mater לאחר שהגרסה תיבדק (לרוב בתהליך הנקרא code review).

Bug – ענף היוצא מ develop לצורך טיפול בבאג קיים.
נהוג לתת לענף שם בעל קידומת bug/XXX (כאשר XXX – שם אינפורמטיבי לבאג שנרצה לתקן).

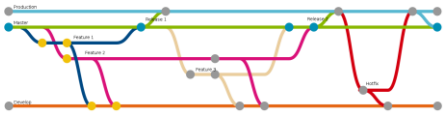
Feature – ענף היוצא מ develop לצורך הוספת פיצ'ר חדש.
נהוג לתת ל branch שם עם הקידומת feature/XXX (כאשר XXX – שם אינפורמטיבי לפיצ'ר שנרצה להוסיף).

ניתן לקרוא בהרחבה (ועוד המון מדריכים טובים) בלינק הבא:
https://goo.gl/ynGsA5

לצורך התרגול אנו ניצור ענפים חדשים (גם עבור פיצ'רים קטנים).
ראשית קראו על הפקודות:

- git branch
- git branch <branch>
- git checkout

מפתח מסוים יכול לעבוד על פיצ'ר, לבצע commits ולאחד את הענף שלו לענף ה develop, (כמובן רק כאשר הפיצ'ר שלו מוכן).



עבודה עם Branch

- צרו הסתעפות ב repository ע"י שימוש ב branch, תנו לו את השם develop.
  - בדקו ע"י git status שהענף אכן נוצר.
  - עברו לענף develop שיצרתם ע"י שימוש ב checkout.

- עברו לענף develop ע"י שימוש ב checkout.
- הוסיפו למחלקה Student את הפונקציות get&set עבור המשתנה name.
  (זכרו שלפני כן עליכם לפתוח ענף ע"פ הקונבנציה של Feature)
- דחפו את השינויים לשרת המרוחק בעזרת push, אך כעת במקום master נדחוף לענף החדש שיצרנו.

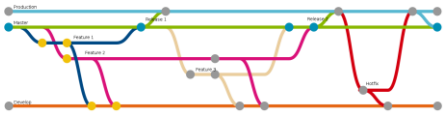
- הוסיפו למחלקה Course את הפונקציה getName
  (זכרו שלפני כן עליכם לפתוח ענף ע"פ הקונבנציה של Feature)
- דחפו את השינויים לשרת המרוחק בעזרת push, אך כעת במקום master נדחוף לענף החדש שיצרנו.

חשוב!

נשאף לפתוח branch של feature עבור תכולה מרכזית בפרויקט (כזאת שתכיל מספר פונקציות/מחלקות),
לדוגמא - הוספת שלב במשחק מחשב, הוספת טסטים לפרויקט וכו'...
(במקרה שלנו לצורך ההדגמה תרגלנו עבור פונקציה קטנה).

אחרי שהצלחתם, שחקו קצת עם הממשק של האתר.

- גלו איך צופים בקוד/commits של כל ענף.
- גלו כיצד ניתן לעבור בין ענפים.
- כנסו ל commit האחרון וגלו כיצד לראות את ה diff.
- מצאו את הגרף שמציג את כל הענפים.



כעת כל מפתח יעבוד בbranch החדש או הישן ע"י שימוש ב checkout.

איך נמזג את ה branch שיצרנו ל develop או לכל ענף אחר?

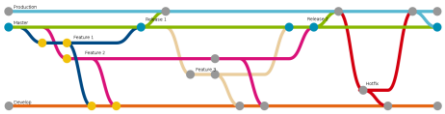
ניתן לבצע דרך הפקודה:

- git merge

- אנו נבצע merge דרך האתר כפי שיוצג בעמוד הבא.
  לאלו מכם שרוצים לתרגל שימוש ב- command line ניתן להיעזר במדריך הבא:
  https://goo.gl/mbSdiH

מיזוג (merge request) דרך האתר GitLab:

1. כנסו באתר אל ה branch שנרצה למזג.
2. מצג ימין למעלה נלחץ על Create merge request
3. בעמוד שנפתח נלחץ מצג ימין על Change branches (ממוסגר באדום)
4. כעת יפתח עמוד חדש נבחר את ה Source branch ואת ה Target branch ונלחץ על Compare branches and continue
5. בעמוד שנפתח נבחר תחת Assignee (ממוסגר בסגול) את בן הזוג על מנת שיבדוק את מה שעשינו ויאשר את המיזוג (הסבר בהמשך)
6. לאחר מכן לחצו על Submit merge request שבתחתית.



adar › Team_project › Merge Requests

New Merge Request

From develop into master

Change branches

Title: Develop

Start the title with WIP: to prevent a **Work In Progress** merge request from being merged before it's ready.

Add description templates to help your contributors communicate effectively!

Description: Write | Preview

Write a comment or drag your files here...

Markdown and quick actions are supported | Attach a file

Assignee: adar

Milestone

Labels

Select assignee

Search assignee

Unassigned

adar @adaroh

Source branch

Target branch: master | Change branches

Remove source branch when merge request is accepted.

Squash commits when merge request is accepted. About this feature

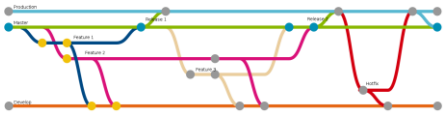
Submit merge request | Cancel

- בצעו מיזוג של הענף שיצרתם אל תוך הענף develop
- בצעו מיזוג של הענף שיצרתם אל תוך הענף develop

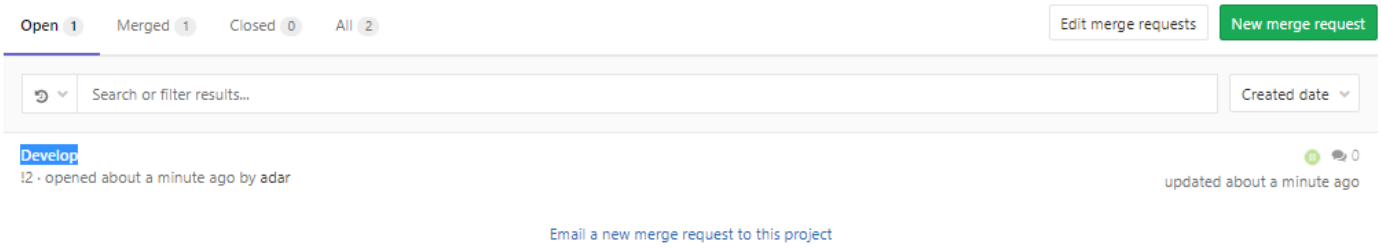
כעת בואו נבדוק ונאשר את השינויים אחד לשני.

לשם כך נבצע את השלבים הבאים:

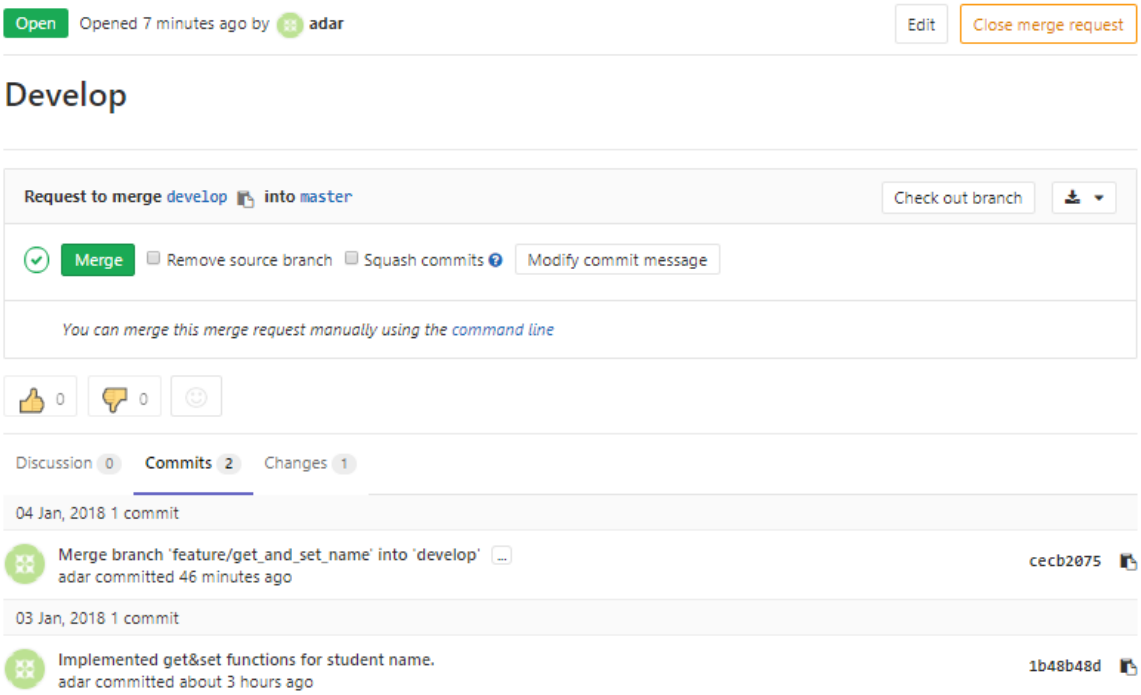
1. לחצו על Merge Requests (נמצא באתר בסרגל שמצד שמאל)
2. בחרו את ה merge request שנרצה לבדוק(במקרה זה מודגש בכחול).



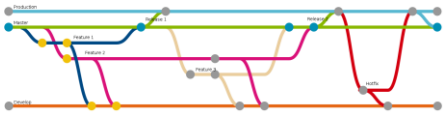
מגשימים



3. בחלון שנפתח הסתכלו על השינויים, במידה ומאושר לחצו על Merge
(זכרו לא לסמן ב V את התיבה על מנת שהענף לא ימחק).



- אשרו את המיזוג לירוק.
- אשרו את המיזוג לכתום.
- הוסיפו את המדריך ל- repo (כפי שראינו בהתחלה).
- בצעו merge request מ develop לתוך master והוסיפו את המדריך ב Assignee על מנת שיבדוק ויאשר.



מגטימינט

סיכום

בסדנה זו רכשתם את הידע הבסיסי וההבנה בצורך בשימוש בתוכנות לניהול קוד, בנוסף למדתם על שיטת עבודה נפוצה בשוק.
נדרשתם ללמוד לא מעט דברים בעצמכם, ויש עוד הרבה מה ללמוד.

אנו משאירים לכם את הבחירה בשימוש בתוכנות שונות עם ממשק משתמש העובדות עם git כמו:

- SourceTree
- Visual studio

כמו כן, לא הספקנו ללמוד נושאים כמו conflicts שעשויים לקרות בעבודה משותפת וגם פרטנית.
כאשר תתקלו בדברים שאתם לא מכירים, זכרו שגם את מה שלמדם כאן, למדתם לבד וכך תוכלו להתגבר גם על קשיים נוספים.

ניתן לגשת בלינק הבא ל- repo שמסביר את התרגיל (ולשחק בו קצת):

https://gitlab.com/adaroh/Team_project/branches

כל הכבוד! סיימתם עוד משימה!
עוד צעד קטן ותהפכו למומחי git (מוסמכים עם תעודות).

כעת תחזרו לתכנית העבודה והמשיכו למשימה הבאה.