בקרת גירסאות

פיתוח תוכנה הוא תהליך מתמשך. אנחנו כותבים קוד, משנים ומשפרים אותו, ואז מגלים שתוך כדי השיפור – קלקלנו משהו שכבר עבד... **בקרת גירסאות** מאפשרת לנו לשמור את כל הגירסאות הישנות של כל קובץ במאגר שלנו, להשוות בין גירסאות, לראות מה שינינו ומתי, וגם לחזור לגירסאות קודמות אם צריך.

יותר מזה, בקרת גירסאות מאפשרת לכמה מפתחים לעבוד ביחד על אותו מאגר. כל אחד מבצע שינויים באופן עצמאי, וכשמחליטים לשלב, ממזגים את כל השינויים של כולם למאגר אחד.

יש דרכים רבות לשמור גירסאות: אפשר פשוט לשמור ידנית עותק מכל קובץ שעובדים עליו, אבל זה יוצר בלגן בתיקיה. אם אתם כותבים מאמר בויקיפדיה, הגירסאות הישנות נשמרות אוטומטית; גם גוגל דרייב שומר גירסאות של קבצים שאתם כותבים שם. אבל הם לא מאפשרים לכם לעבוד על קבצים שלכם.
דרופבוקס שומר גירסאות ישנות למשך 30 יום בלבד, והשמירה מתבצעת אוטומטית – לא לפי בחירתכם.
בבקרת גירסאות של קוד, חשוב שתהיה לנו אפשרות להחליט מתי אנחנו רוצים לשמור גירסה. רצוי שנשמור גירסה של קוד עובד ובדוק, כך שאם מחר נקלקל משהו – נוכל לחזור לגירסה שאנחנו יודעים שעבדה טוב היום.

ישנן מערכות רבות לבקרת גירסאות המאפשרות לכם שמירה של הגירסה לפי בחירתכם. למשל: `CVS, SubVersion, Git, Mercurial`. בקורס זה נתמקד במערכת `Git` (גיט).

בתחילת השיעור נלמד איך לעבוד עם גיט באופן פרטי, כמתכנת יחיד.
בהמשך נלמד איך לשתף פעולה עם מתכנתים נוספים.

א. יצירת מאגר חדש - repository

כדי להתחיל לעבוד עם גיט צריך ליצור מאגר חדש. בשפה של גיט, מאגר נקרא **repository**. זהו המקום שבו נשמרים כל הקבצים שלכם וכל הגירסאות הישנות והחדשות שלהם.

אנחנו נשמור את כל המאגרים שלנו באתר חינמי שנקרא `github` (גיטהאב). פיתחו חשבון בגיטהאב, היכנסו לחשבון שלכם וצרו מאגר חדש (לחצו בצד ימין למעלה על + ועל `New Repository`). תנו שם למאגר שלכם, כיתבו תיאור, והוסיפו לו שני קבצים -

- קובץ `Readme.md` – הקובץ הזה יהיה "חלון הראווה" של המאגר שלכם ויסביר מה יש בו.
- קובץ `.gitignore` - נסביר בהמשך מה התפקיד שלו. בחרו את הקובץ המתאים לשפת `Java`.

ברכות! יש לכם מאגר חדש. אתם יכולים לראות את שני הקבצים במאגר בממשק-ווב של גיטהאב.

ב. יצירת גירסה חדשה - commit

הפעולה הבסיסית שבעזרתה שומרים גירסה בגיט היא – `commit`. בכל פעם שאתם מעדכנים קובץ אחד או כמה קבצים קשורים זה לזה, אתם יכולים "להגיש" את השינויים שלכם על-ידי פקודת `commit`.
השינויים ייכנסו למאגר, אבל הגירסאות הישנות יישמרו.

כדוגמה, אתם יכולים לבצע commit מתוך גיטהאב. לכו לקובץ Readme.md שבמאגר שלכם, לחצו על כפתור העריכה בצד ימין למעלה, והוסיפו כמה מילים. בתחתית, כיתבו משפט שמסביר מה שיניתם, כגון " add a few words to the readme". הגישו את הטופס. אתם אמורים עכשיו לראות את הגירסה החדשה של הקובץ שלכם.

ג. צפיה בגירסאות ישנות – history, blame

כדי לראות את הגירסאות הישנות, אפשר ללחוץ על History. אתם אמורים לראות שם שתי גירסאות: הגירסה הישנה והחדשה. עבור כל גירסה, המערכת שומרת את הנתונים הבאים:

- מזהה ייחודי של הגירסה. המזהה נוצר מתוך הגירסה בצורה מתוחכמת שאמורה למנוע זיוף.
- הסבר במילים – אותו הסבר שכתבתם כשהגשתם את הגירסה.
- מי הגיש את הגירסה; במקרה זה – אתם.
- מתי הוגשה הגירסה.

אם תלחצו על הכותרת של הגירסה החדשה, תראו מה בדיוק שיניתם: בצבע אדום הגירסה הישנה ובצבע ירוק הגירסה החדשה. המערכת מראה לכם כל שורה שהשתנתה, ובתוך כל שורה – כל מילה שהשתנתה. האפשרות הזאת לראות מה בדיוק השתנה היא מאד שימושית כאשר עובדים עם קבצים גדולים.

בתצוגה שמראה לכם את ההבדלים בין הגירסאות, אתם יכולים להוסיף הערות. זה שימושי כשעובדים בצוותים. למשל, אם מישהו אחר בצוות עשה שינוי שאתם לא מבינים, אפשר להוסיף לו הערה "למה שינית?!".

הערה: גיטהאב מראה לכם את השינויים מכל גירסה לגירסה שלפניה. אפשר גם להשוות בין גירסאות מרוחקות יותר (למשל הגירסה של היום לעומת הגירסה של לפני 3 ימים). לפירוט ראו בתיעוד: https://help.github.com/articles/comparing-commits-across-time/

דרך נוספת לראות את ההיסטוריה של הקובץ היא ללחוץ על Blame. אתם אמורים לראות שם את הקובץ, כאשר ליד כל שורה כתוב מי "אשם" בשורה הזאת – מי האחרון ששינה אותה, ומתי. במקרה זה, תראו ששתי השורות השתנו על-ידכם - השורה הראשונה בקובץ (שורת הכותרת) השתנתה קודם, והשורה השניה השתנתה מאוחר יותר. כשעובדים בצוותים, והתוכנית פתאום מפסיקה לעבוד, זה מאד עוזר שיודעים את מי להאשים (כמובן צריך "להאשים" בצורה מכובדת ולשמור על אוירה נעימה בצוות, אבל זה כבר נושא לקורס אחר...).

אפשר גם לראות את כל המאגר כפי שהיה בזמן הגשת גירסה מסויימת; לחצו על Browse files.

ד. העתקת מאגר למחשב מקומי - clone

ראינו איך ליצור גירסאות חדשות של קבצים דרך גיטהאב. זה טוב לתיקונים קטנים בקבצים בודדים, אבל לא לעבודה על מאגרים גדולים. כדי לעבוד על מאגרים גדולים אנחנו צריכים להעתיק אותם למחשב שלנו. הפעולה של העתקת מאגר שלם מהשרת המרכזי למחשב הפרטי שלנו נקראת clone - "שיבוט".

ניתן לבצע אותה דרך אקליפס: קודם-כל פתחו את תצוגת מאגרי גיט – (Window – Show View – Git Repositories). לחצו על המשולש בצד ימין למעלה ובחרו באפשרות "Clone a repository". הכניסו את הכתובת של המאגר בגיטהאב, שהיא מהצורה הבאה:

```
https://github.com/<username>/<repository>.git
```

כאשר username הוא שם המשתמש שלכם בגיטהאב, ו repository הוא השם שנתתם למאגר. אפשר לקבל קישור זה ישירות מגיטהאב ע"י לחיצה על הכפתור הירוק בצד ימין "Clone or download".

כל מאגר של גיט מועתק לתיקיה על המחשב שלכם, ששמה כשם המאגר.

כל מאגר ששיבטתם דרך אקליפס מופיע גם בתוך אקליפס, בתצוגת מאגרי גיט. תוכלו לראות את הקבצים דרך ה Working Tree.

תרגיל כיתה: צרו פרוייקט ג'אבה חדש והעתיקו אותו לתיקיה של המאגר שלכם. העתיקו לשם קבצים שכתבתם בעבר, למשל המחלקות של המונום והפולינום שכתבנו באחד השיעורים הקודמים.

אתם יכולים להגיש גירסאות גם מתוך אקליפס – פשוט צריך לחפש בתפריט את האפשרות "commit". לתהליך יש שני שלבים:

- הוספת קבצים ל"במה" (stage) – פעולה שנקראת גם add – בפעולה זו אתם בוחרים איזה שינויים אתם רוצים להכליל בגירסה.
- הגשת הגירסה – עם כל השינויים שנמצאים על ה"במה". יש לכתוב הערה שמסבירה מה שיניתם, וללחוץ על "commit".

אפשר לראות הסטוריה של כל קובץ דרך תפריט-ימני Team, או תפריט-ימני Show In History.

שימו לב – בגיט אתם יכולים להגיש גירסה הכוללת כמה קבצים בו-זמנית (בניגוד למערכות אחרות כגון מדיה-ויקי, דרופבוקס או גוגל-דרייב, השומרות גירסאות של כל קובץ בנפרד). שמירת גירסאות של כמה קבצים בו-זמנית חשובה במיוחד במערכות תוכנה, שבהם לפעמים שינוי יכול להשפיע על הרבה קבצים.

הערה: אפשר להשתמש בגיט גם משורת-הפקודה. כדי לעשות זאת מתוך אקליפס, אפשר להתקין את פלאג-אין לאקליפס בשם GONSOLE בכתובת http://rherrmann.github.io/gonsole .

ה. חזרה לגירסה קודמת – checkout

אם יש לכם באג ואתם לא יודעים מתי הוא נכנס לתוכנה, פתרון אפשרי הוא לחזור אחורה בזמן לנקודה שבה אתם יודעים שהתוכנה עבדה היטב. הפעולה שמשתמשים בה למטרה זו היא **checkout** (באקליפס, כפתור ימני על שם המאגר, Show In – History, מצאו את הגירסה שאתם רוצים לחזור אליה, כפתור ימני ו Checkout).

ו. סיכום ביניים

ראינו שאפשר ליצור מאגר חדש בגיטהאב, לערוך קבצים ולהגיש גירסאות ולראות הסטוריה.

ראינו שאפשר לשבט מאגר למחשב המקומי, וגם שם אפשר לערוך ולהגיש גירסאות ולראות הסטוריה.

שאלה מעניינת היא: מתי כדאי להגיש גירסה חדשה (commit)?

ההמלצה המקובלת היא להגיש גירסה חדשה בכל פעם שמסיימים שינוי ממוקד בתוכנה, כגון: תיקון של באג מסויים או תוספת של יכולת חדשה. בהערה של הגשת הגירסה, יש לכתוב מה שינינו. כך יהיה קל להבין את ההסטוריה של הפרוייקט, ואם צריך – גם לחזור אחורה בזמן.

ז. יצירת ענפים – branch

לפעמים אנחנו רוצים לעבוד על כמה שינויים נפרדים בו-זמנית. למשל, להוסיף מאפיינים חדשים לתוכנה, ובו-זמנית לתקן באגים בגירסה הישנה. גיט מאפשר לנו לעשות זאת ע"י פתיחת **ענפים** – branch. מתוך אקליפס, אפשר לפתוח ענף חדש ע"י תפריט-ימני – Switch to – New Branch. נותנים שם לענף החדש ומתחילים לעבוד איתו. משנים קבצים ומגישים את השינויים (ע"י commit).

אם רוצים לעבור לענף אחר, עושים שוב Switch to ובוחרים את שם הענף.

מעבר בין ענפים יכול לעשות את יום העבודה שלכם מעניין ומשעשע – בכל פעם שאתם משתעממים מעבודה על נושא מסויים – אתם יכולים לעבור לנושא אחר וחזרה...

שימו לב – ענף הוא בסה"כ תוית לגירסה (commit). הוא מאפשר לנו לקרוא לגירסה בשם משמעותי במקום במזהה הייחודי שלה. לכן, כשמוחקים ענף, הגירסה לא נמחקת – רק התוית נמחקת.

ח. מיזוג ענפים – merge

אחרי שסיימנו לעבוד על ענף, אנחנו יכולים למזג אותו לתוך ענף אחר (בד"כ הענף הראשי) ע"י פעולת merge.

גיט יעשה כמיטב יכולתו למזג את השינויים; אם השינויים הם בקבצים שונים, או אפילו באיזורים שונים באותו קובץ, גיט כנראה יצליח למזג אותם נכון. אבל אם השינויים הם באותה שורה באותו קובץ, גיט לא יידע מה לעשות ויתריע על עימות – conflict, ואז אנחנו נצטרך למזג את השורות ידנית.

ט. תיאום בין עותקים – pull, push

כשאתם מגישים גירסה מהמחשב המקומי שלכם (למשל מאקליפס), אתם רואים חץ קטן ליד שם המאגר, שאומר לכם שהעותק שלכם מתקדם יותר מהעותק המרוחק.

כשאתם מגישים במקביל מהעותק המרוחק (למשל גיטהאב) ואז מסתכלים באקליפס, אתם רואים חץ קטן שאומר לכם שהעותק שלכם מאחר לעומת העותק המרוחק.

איך אפשר לתאם בין העותקים, כך שבכל עותק יהיו כל השינויים?

ישנן שתי פעולות שמאפשרות לנו לבצע תיאום זה:

- pull – "משיכת" שינויים מהעותק המרכזי של המאגר לעותק המקומי שלנו.
- push - "דחיפת" שינויים מהעותק המקומי שלנו לעותק המרכזי.

יש לבצע פעולות אלו לפי הסדר – קודם למשוך ואחר-כך לדחוף. אם נעשה כך, נראה שגם על המחשב שלנו וגם בגיטהאב נמצאים כל השינויים שביצענו.

כשדוחפים ענף מסויים, נדחפות כל הגירסאות החל מהענף הזה ועד לגירסה שכבר נמצאת על העותק המרוחק; גיט מספיק חכם כדי לדעת לא לדחוף גירסאות שכבר נמצאות שם.

י. שיכפול מאגר של מישהו אחר - fork

בגיטהאב ישנם מאגרים רבים שיוצריהם מפרסמים אותם ברישיון של קוד פתוח. אתר גיטהאב מאפשר לכל אחד לשכפל כל מאגר שנמצא בגיטהאב ע"י לחיצה על כפתור `fork`. לאחר השיכפול, יווצר אצלכם עותק של המאגר המקורי, ותוכלו לעבוד בו ולהגיש לו שינויים בלי שזה ישפיע על המאגר המקורי.

אם אתם רוצים למזג את השינויים שלכם למאגר המקורי, תצטרכו לבצע `pull request` – "בקשה למשיכה". בזה אתם מבקשים מבעל המאגר המקורי שימשוך את השינויים שלכם. אם הוא מסכים, אז השינויים שלכם יתמזגו למאגר המקורי, ושמכם יופיע כתורמים למאגר זה.

למה כדאי לתרום לפרוייקטי קוד פתוח? קודם-כל, כי זה כיף לראות את שמכם ברשימת התורמים. אבל יש גם רווח יותר חומרי: אם תלכו להתראיין בחברות גדולות כגון גוגל, ייתכן שיבקשו מכם להראות דוגמת קוד שכתבתם. זה מאד ירשים את המראיינים אם תראו להם שהייתה לכם תרומה משמעותית לפרוייקט קוד פתוח, בפרט אם מדובר בפרוייקט גדול שהרבה אנשים משתמשים בו.

נסכם עכשיו את ההבדלים בין שלושת הדרכים שראינו עד כה לפצל את העבודה על מאגר:

- **`branch`** – יצירת ענף חדש בעץ-הגירסאות של מאגר נתון; משמש בדרך-כלל לפיצול עבודה של אדם יחיד על מחשב יחיד.
- **`clone`** – שיכפול מאגר למחשב מרוחק; משמש בדרך-כלל לפיצול עבודה של אדם יחיד על מחשבים שונים (למשל המחשב האישי והמחשב של גיטהאב).
- **`fork`** – שיכפול מאגר בגיטהאב; משמש בדרך-כלל לעבודה של שני אנשים שונים.

מקורות

- How to use GIT, Udacity course: https://classroom.udacity.com/courses/ud775/lessons/2980038599/concepts/29607789250923

סיכם: אראל סגל-הלוי.