גבהים
חינוך • מצוינות • סייבר

Source Control

מה נלמד היום?

- למה צריך Source Control
- איך לעבוד נכון עם Source Control
- שימוש ב- Git ו- Bitbucket

מוטיבציה (1)

- פיתחנו מערכת שעושה מגוון פעולות
- אנחנו רוצים להוסיף "פיצ'ר" (יכולת) למערכת
- שינינו משהו בקוד, ועכשיו המערכת לא עובדת
- מה עושים?

מוטיבציה (1)

- לעיתים, נרצה לחזור למצב התקין הקודם של הקוד
  - או לפחות לראות בדיוק איזה שינויים ביצענו...

- לדוגמה: התחלנו לעבוד על פיצ'ר חדש, ואז גילינו שיש לנו באג במקום אחר ויותר חשוב
  - הקוד שלנו כרגע "לא עובד"
  - נרצה לתקן אותו בגרסה העובדת (כדי שנוכל גם לבדוק)
  - אחרי כן, נאחד אותו עם מה שפיתחנו לפיצ'ר החדש

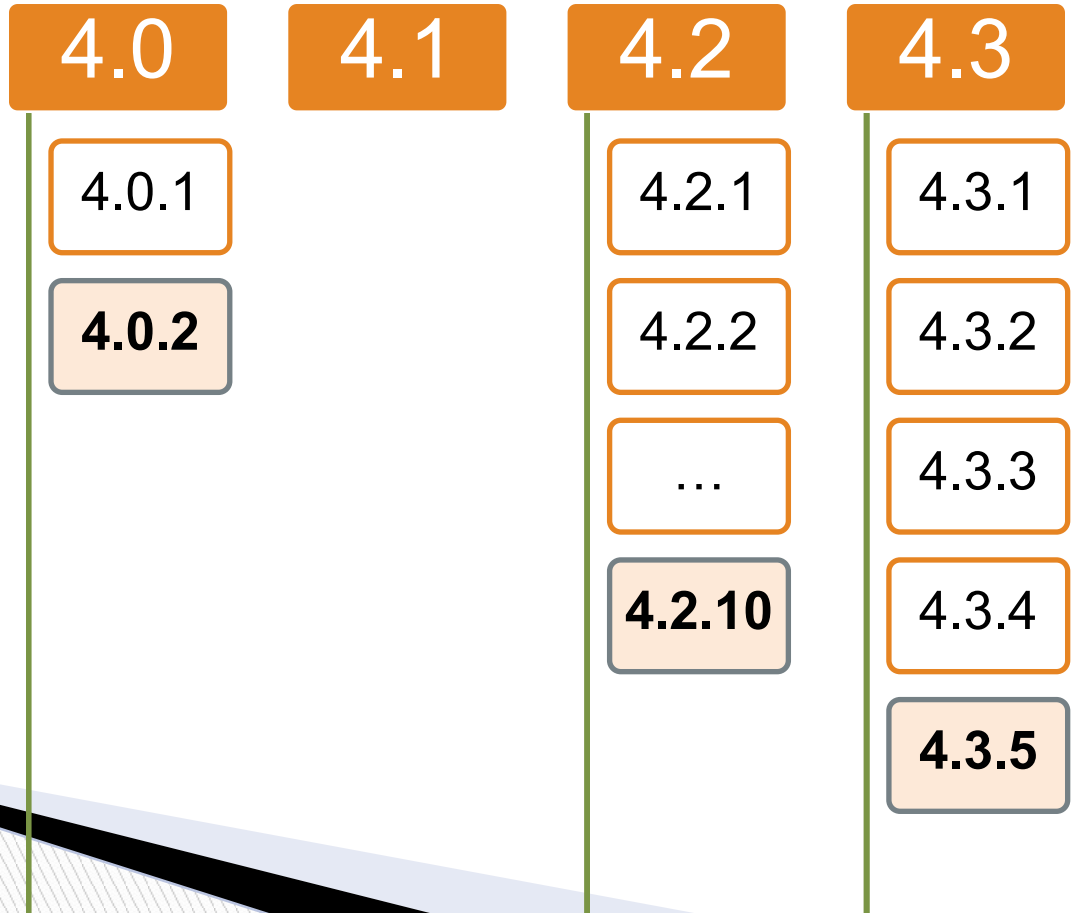
- עוד רעיונות?

מוטיבציה (2)

- בפרויקט פיתוח – עובדים בד"כ יותר ממפתח אחד
- כולם עובדים על אותו הקוד
  - כיצד נסנכרן ביניהם את העבודה?
- איך נדאג שתיקון בקוד שמישהו עושה – יעבור לכולם?
- מה קורה כששני אנשים עורכים את אותו הקובץ?
- איך מתחזקים גרסאות ישנות?

מוטיבציה – ניהול גרסאות

- אם נסתכל למשל על גרסה 4 של iOS



Source Control

- מערכת המאכסנת את כל קוד הפרויקט
- מחזיקה היסטורית שינויים בקוד ותיעוד שלהם
  - מאחסנת את כל הגרסאות שהוצאנו
- מאפשרת למספר מפתחים לעבוד על הקוד
  - ניתן לראות מי אחראי לאיזה חלק קוד
  - מונעת התנגשויות ועוזרת לטפל בהן
- מאפשרת לפתח במקביל יכולות שונות

מושגי יסוד
ב Source Control

ChangeSet

- ניקח לדוגמא - מתכנת שעובד על משחק. מה הוא עושה?
  - "מוסיף סאונד כשלוחצים על הרווח"
  - "מתקן באג בגרפיקה"
- כל משימה כזו ניתן לתאר ב - "מה נשתנה"
- נניח שיש פרויקט רץ ועובד עם בסיס קוד ובו 10 קבצים.
  - כעת רוצים לבצע בו שינוי מסוים
  - למשל - להוסיף Highscore למשחק
  - האם יש צורך לשמור את כל הפרויקט שעשינו עד עכשיו מחדש?
- נמדוד את השינויים לפי הקבצים עצמם (יצירה / מחיקה / עריכה)

ChangeSet

- ניתן לתאר תהליך פיתוח של פרויקט כאוסף גדול של הרבה תתי משימות.

- נגדיר ChangeSet להיות אוסף השינויים בין תת-משימה אחת לזו שבאה אחריה.

- שינוים:
  - הוספת קבצים
  - מחיקת קבצים
  - עריכת קבצים (עריכת הקוד)

ChangeSet

- לדוגמה - אנחנו בעיצומו של פיתוח המשחק סופר-מריו, ורוצים כעת להוסיף מטבעות וטבלת ניקוד.

- הוספת גרפיקה של מטבעות
- הוספת מטבעות למפת המשחק
- הוספת סאונד כשפוגעים במטבע
- הוספת טבלת שיאים
- הוספת כפתור "שיתוף שיא בפייסבוק"

ChangeSet

- אם נסתכל על הקוד – המשימות שלו יראו כך:

1: התחלת עבודה
- A main.cpp
- A graphics.cpp
- A keyboard.cpp

2: הוספת תמיכה בסאונד למשחק
- M main.cpp
- A sound.cpp

3: הוספת מטבעות למשחק
- A coin.jpg
- M main.cpp
- M keyboard.cpp

A – Added M - Modified

ChangeSet

- לכל סט שינוים הצמדנו מספר
- למספר זה קוראים Revision והוא ייחודי
- המספר מייצג את "גרסת" הקוד

- ניתן לתת ל-Revision שם
  - לפעולה זו קוראים Tagging

- השם מייצג בדרך כלל גרסאות או מצבים סגורים (למשל 1.0, 1.1, beta-1.2)

Repository

- מאגר מנוהל של קוד
- עבור כל פרויקט – ניצור Repository
  - בחברה עם 4 פרויקטים – יחזיקו 4 Repository-ים

- המאגר מחזיק את היסטוריית השינויים בקוד
  - כל השינויים (מחיקה / יצירה / שינוי) בקבצים
  - מי ביצע את השינוי
  - גרסאות שהוצאנו
  - צירי פיתוח בעבר וכרגע (branches)

סוגי Source Control

קיימים שני סוגים עיקריים של Source Control:

1. Centralized Source Control

- לדוגמה: cvs, svn
- כל היסטורית השינויים נשמרת בשרת (ורק בו)
- כל פעולה תתבצע ישירות מול השרת
- עבודה כזו דורשת מאיתנו חיבור פעיל כל הזמן
  - כל פעולה מתבצעת מול הרשת
  - הדבר עלול להאט את העבודה

סוגי Source Control

2. Distributed Source Control

- לכל משתמש קיים העתק מלא של המאגר.
- עובדים מול מאגר מקומי על המחשב האישי
- בסיום העבודה מעדכנים בשרת
  - מבצעים דחיפה (push) לשרת מרכזי
- על מנת לקבל שינויים –מתעדכנים מהשרת
  - מבצעים משיכה (pull) מהשרת המרכזי
- שחקנים עיקריים:
  - **git** ושירות אחסון הקוד Github
  - **Mercurial** ושירות אחסון הקוד Bitbucket

git

- Distributed VC
- קיים מימוש מלא ב- python
- השימוש מתבצע באמצעות הכלי git
  - יש מספר ממשקים גרפים שפותחו

- ממשקים גרפיים מומלצים:
  - SourceTree (מומלץ!)
  - Tortoise git
- קיימת אינטגרציה עם git לרוב ה- IDE-ים (eclipse, visual studio, ...)

Bitbucket

- שירות אחסון קוד חינמי ל git ו- mercurial
- מפותח ע"י Atlassian
  - החברה שמתחזקת גם את SourceTree (הממשק הגרפי)
- מספק אחסון קוד פרטי לעד 5 משתמשים בו זמנית

הפקודות שנלמד היום

- Clone
- Status
- Commit
- Push
- Pull
- Update
- Branch
- Merge

git clone

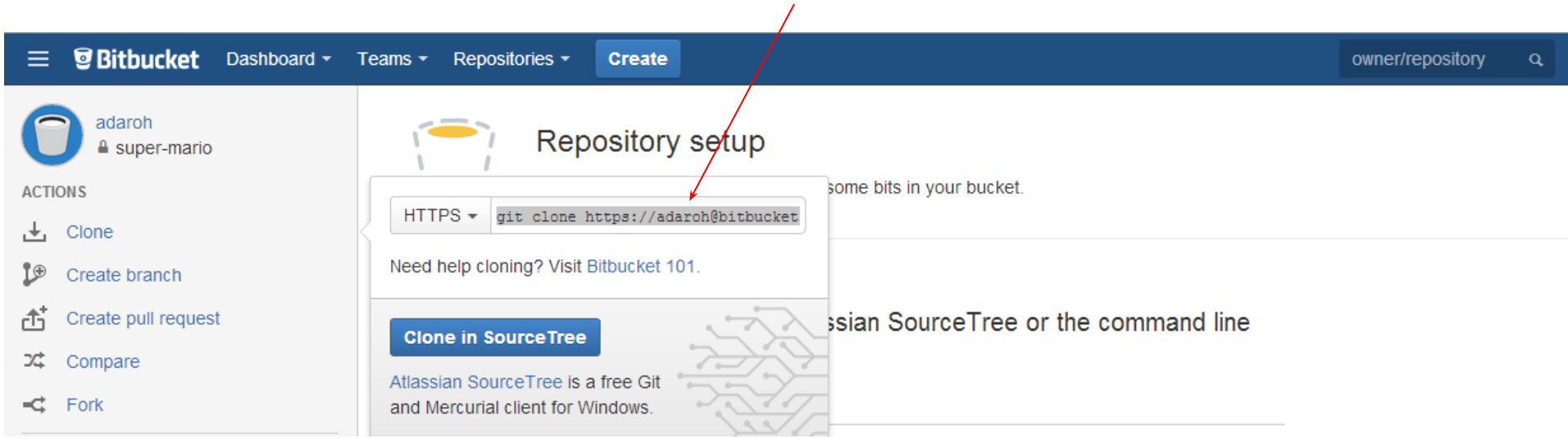
- כל משתמש מחזיק עותק מלא של ה –Repository
- הפעולהclone יוצרת על המחשב את העותק הנ"ל

- הפקודה תיצור על המחשב תיקייה, ובתוכה תשב תמונת מראה מלאה של כל הקוד הכי עדכני.

- מבנה הפקודה:
  - git clone http://user@server.com/repo-name

git clone

- על מנת לעבוד בנוחות, נפתח Repo ב-Bitbucket
- ניתן לראות את הפקודה גם שם:



git clone https://adaroh@bitbucket.org/adaroh/super-mario.git

עריכת קוד ושמירת שינוים

git status

- זוכרים Change-Set?
  - כל שינוי בקוד / יצירת קבצים / מחיקת קבצים / הזזת תיקיות וכו'
- הפקודה status מציגה את הChangeSet הנוכחי
  - למעשה: השינויים שבוצעו מאז הגרסה האחרונה שנשמרה
- השינויים הללו לא נשמרים ב git עד שלא נבצע Commit (פקודה השומרת את סט השינויים עם תיעוד).

git add

- כאשר אנחנו יוצרים קובץ חדש, הוא איננו מנוהל על ידי ה-SourceControl
  - למה זה הגיוני?
  - יש קבצים שלא נרצה שיהיו מנוהלים.
  - למשל: תוצרי קימפול, קבצי גיבוי של עורך הטקסט וכו').

- על מנת שקובץ ינוהל (ותשמר ההיסטוריה שלו), עלינו לבקש זאת במפורש עם הפקודה git add
  - הפקודה מקבלת כפרמטרים את כל הקבצים שנרצה להוסיף.

git add

- <git add <file list
- מציינת לgit להתחיל לעקוב אחר הקבצים

```
C:\Repos\Mario\super-mario>git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        First.txt
        Second.txt
        Thired.txt

nothing added to commit but untracked files present (use "git add" to track)

C:\Repos\Mario\super-mario>git add First.txt

C:\Repos\Mario\super-mario>git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   First.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        Second.txt
        Thired.txt
```

git commit

- כפי שהזכרנו, את כל השינויים ניתן לראות באמצעות הפקודה git status

- לשינויים אלה קראנו ChangeSet.
  - כשמו כן הוא – סט של שינוים המתארים יחד תת-משימה.

- כאשר סיימנו את תת-המשימה עליה עבדנו, נרצה לתת לסט השינויים הזה שם, למשל:
  - תיקון באג ("המשחק נתקע") הדורש שינוי ב 3 קבצים
  - הוספת יכולת
  - וכו'

git commit

- git commit
- רצוי להוסיף הסבר ע"י הדגל m (קיצור ל message)
  - git commit –m “..”
- לוקח את כל השינויים שרואים ב git status
  - מצמיד להם תיאור ו"זוכר" את השינוי הזה.

git commit

```
C:\Repos\Mario\super-mario>git status
On branch master
Your branch is based on 'origin/master', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file:   Second.txt
        new file:   Thired.txt


C:\Repos\Mario\super-mario>git commit -m "Adding two files need to debug"
[master 522d628] Adding two files need to debug
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Second.txt
 create mode 100644 Thired.txt

C:\Repos\Mario\super-mario>git status
On branch master
Your branch is based on 'origin/master', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

nothing to commit, working directory clean
```

שני קבצים שנרצה
להוסיף לפרויקט

git rm

- מוחק קובץ מנוהל (לאחר הפעולה נראה ב- git status שאכן בוצע שינוי)
- נניח שהוספנו שלושה קבצים ל-Repository
- כעת, נמחק אחד מהם:

```
C:\Repos\Mario\super-mario>git rm First.txt
rm 'First.txt'

C:\Repos\Mario\super-mario>git status
On branch master
Your branch is based on 'origin/master', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        deleted:    First.txt


C:\Repos\Mario\super-mario>git commit -m "Deleted First.txt"
[master eb0bf2c] Deleted First.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 delete mode 100644 First.txt

C:\Repos\Mario\super-mario>git status
On branch master
Your branch is based on 'origin/master', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

nothing to commit, working directory clean
```

לא לשכוח

git log

- ראינו מה קורה כשמוסיפים או מוחקים קובץ
- כדי לראות את כל היסטורית השינויים, ניתן להשתמש בפקודה git log

```
C:\Repos\Mario\super-mario>git log
commit eb0bf2ced562166fc465eefac9ab7308a13bc29a
Author: Adar ohana <adaroha@post.bgu.ac.il>
Date:   Thu Sep 4 14:27:54 2014 +0300

    Deleted First.txt

commit 522d628c0d354bbf67d5b35c75f5b2e5fb05d9b9
Author: Adar ohana <adaroha@post.bgu.ac.il>
Date:   Thu Sep 4 14:22:57 2014 +0300

    Adding two files need to debug

commit 526c4dc32d6404294f19c6c1e352658fd8dc7a0f
Author: Adar ohana <adaroha@post.bgu.ac.il>
Date:   Thu Sep 4 14:18:25 2014 +0300

    Add First.txt file
```

עריכה של קובץ

- במקרה שנערוך קובץ, הוא יסומן כקובץ שנערך:

```
C:\Repos\Mario\super-mario>git status
On branch master
Your branch is based on 'origin/master', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Second.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

- נשים לב שכל עריכה נחשבת (גם אם הוספנו רווח או שורה חדשה)

עריכה של קובץ git diff

- מציגה את השינויים בתוך הקבצים שהשתנו
- עוזר לראות אילו דברים ערכנו בקוד עצמו

```
C:\Repos\Mario\super-mario>git diff
diff --git a/Second.txt b/Second.txt
index e69de29..bc7774a 100644
--- a/Second.txt
+++ b/Second.txt
@@ -0,0 +1 @@
+hello world!
\ No newline at end of file
```

כאשר: **(+)** מסמל שורות שנוספו **(-)** מסמל שורות שמחקנו

סיכום ביניים

- עד כה למדנו כיצד לעבוד עם git במהלך הפיתוח:
  - איך להוסיף ולמחוק קבצים
  - איך לבצע שינויים, לשמור אותם ולתת להם תיאור
  - איך לצפות בהיסטורית השינויים
- כזכור, git היא Distributed ולכן כל הפעולות שביצענו נשמרו רק אצלנו על המחשב
- כעת נלמד איך עובדים מול השרת

עבודה במקביל

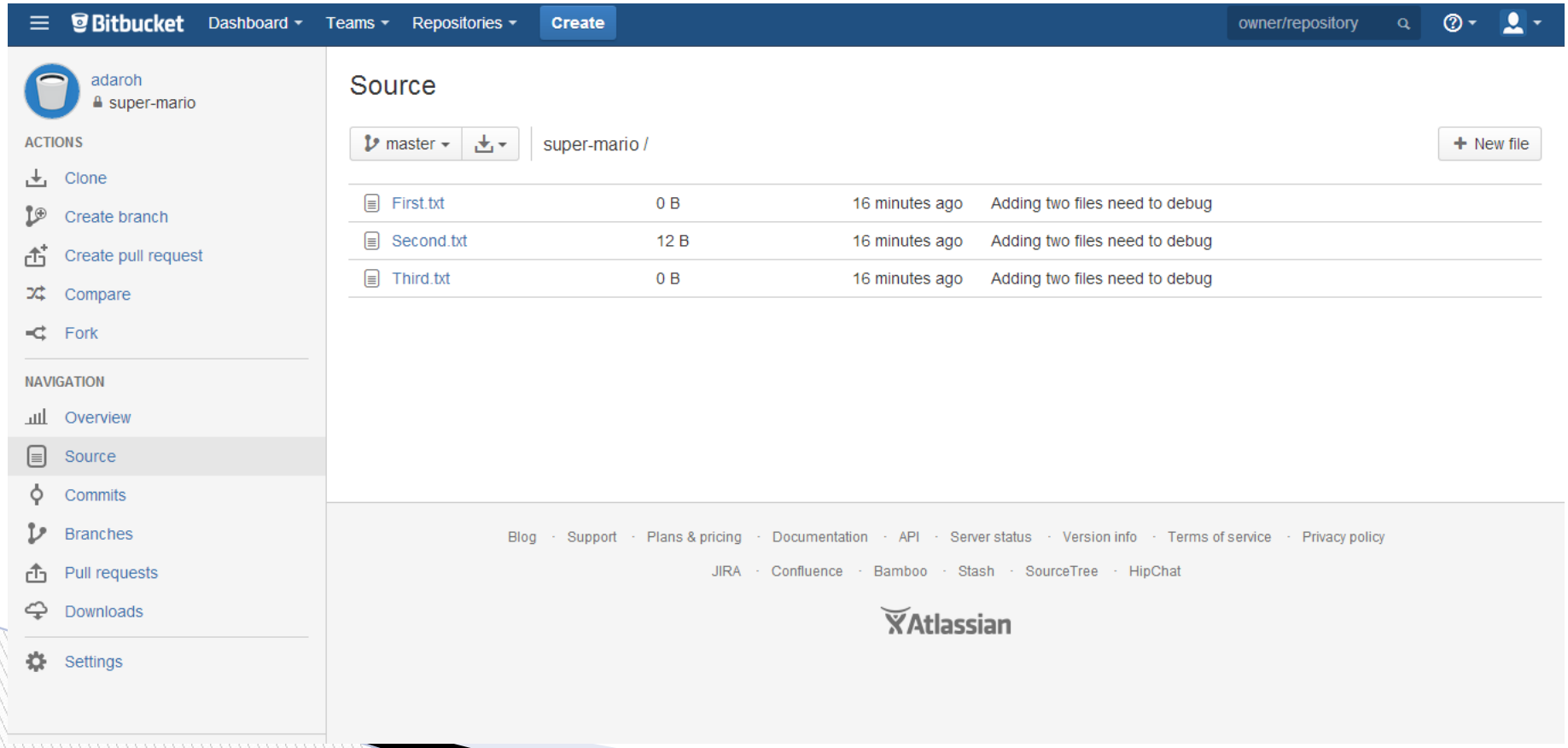
git push/ git pull

- **PUSH**
  - דחיפת כל השינויים שלנו לשרת המרוחק
  - <git push <remote alias> <branch
  - לדוגמא: git push origin master
- **PULL:** קבלת השינויים מהשרת המרוחק

- לא תמיד נוכל לדחוף את השינויים
  - אם הקוד על השרת השתנה מאז שמשכנו
  - במצב כזה:
    - נמשוך מחדש את השינויים אלינו (pull)
    - נאחד את הקוד שלנו עם השינויים שמישהו אחר ביצע
    - נדחוף אותם (push)

git push/ git pull

לדוגמא לאחר הפעולה:git push origin master
נבחין שאכן הקבצים הועלו לBitbucket

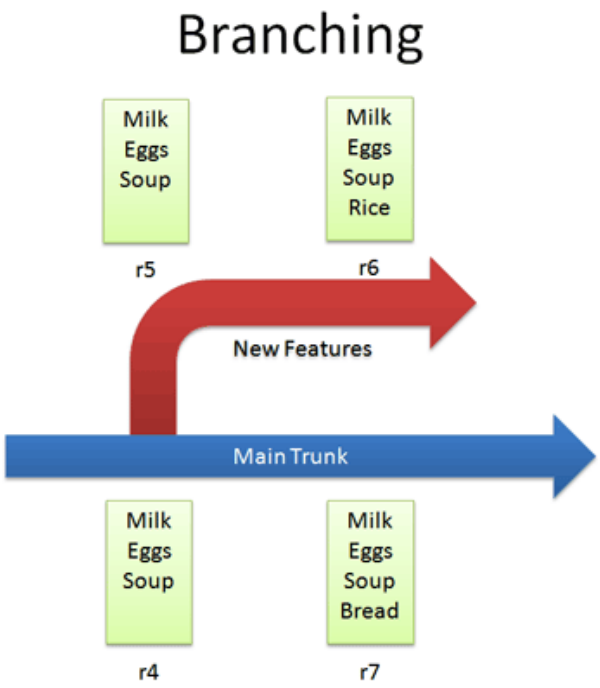


Branches

- לפני שנמשיך ונסביר לעומק את push ו-pull , ננסה להבין את המונח branch
- המונח מייצג ציר פיתוח שמתבצע במקביל.
- אפל למשל מוכרת מכשירים ישנים וחדשים כאחד, מכשירים ישנים כדוגמת iPhone 3gs לא תומכים בגרסה האחרונה של iOS (7), אבל במידה וישנם תיקוני אבטחה, אפל עדיין משחררת את התיקונים לכלל הגרסאות.
- כלומר, היא מחזיקה מספר צירי פיתוח פעילים, בעיקר של גרסאות ישנות על מנת לתחזק אותם לעדכוני אבטחה.

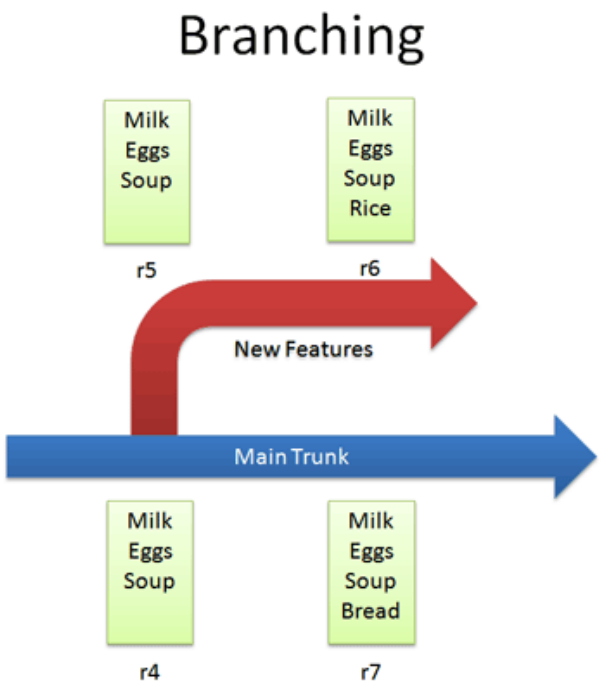
Branches

- Branch הינו ציר פיתוח המתבצע במקביל
- בתחילת העבודה כולם עובדים על master
- נהוג לייצר צירים מקבילים על מנת לא לפגוע אחד לשני בעבודה.
- לדוגמה 2 צוותים מפתחים שתי יכולות חדשות למשחק



Branches

- הפקודה git branch –a תציג את כל הענפים הקיימים
- הפקודה <git branch <name תיצור עץ חדש בשם שנבחר



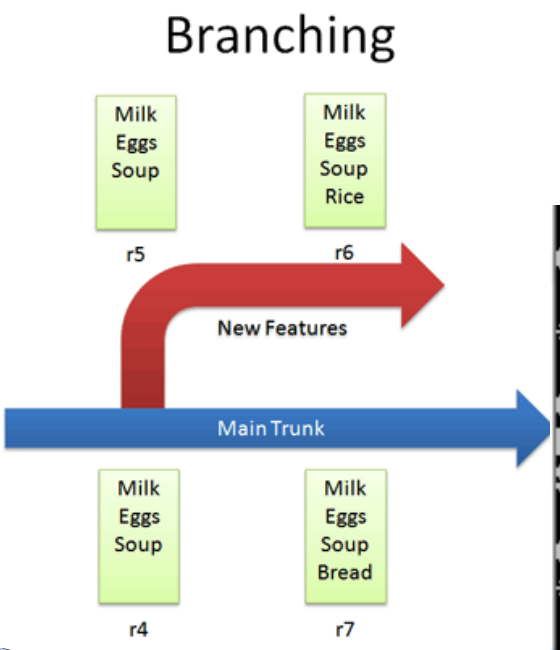

```
C:\Repos\Mario\super-mario>git branch -a
* master

C:\Repos\Mario\super-mario>git branch LoginCode

C:\Repos\Mario\super-mario>git branch -a
  LoginCode
* master
```

Branches

- מעבר בין branch-ים מתבצע באמצעות הפקודה
- **<git checkout <branch-name**



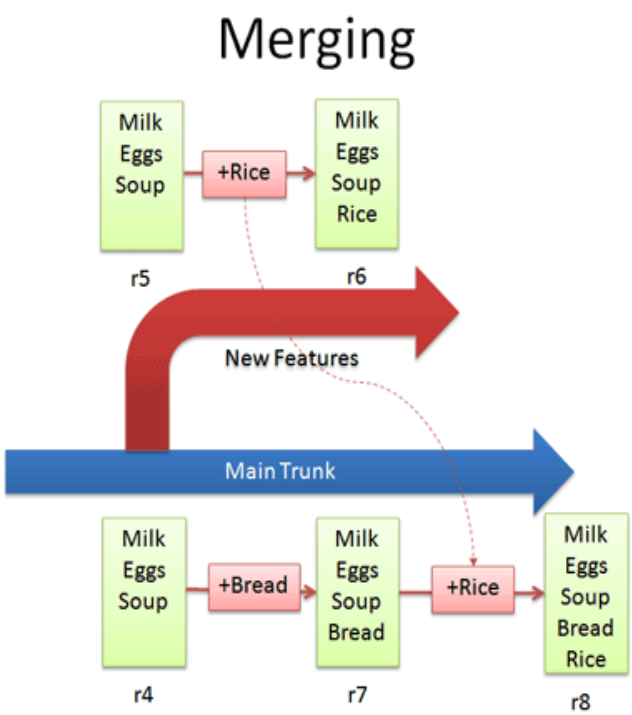

```
C:\Repos\Mario\super-mario>git branch -a
  LoginCode
* master

C:\Repos\Mario\super-mario>git checkout LoginCode
M       Second.txt
Switched to branch 'LoginCode'

C:\Repos\Mario\super-mario>git branch -a
* LoginCode
  master
```

Merge

- מבצע איחוד בין שני Branch-ים
- מתבצע באמצעות הפקודה **<git merge <other-branch**
- יפתור בעצמו בעיות התנגשות.
- במידה ולא יצליח לפתור, יסמן את הקובץ כ-Conflicted ויבקש מהמשתמש לתקן ידנית



- **הערה:**
יש לעשות commit לאחר האיחוד

תרגול

- היכרות עם tortoisegit
  - התקנה: https://code.google.com/p/tortoisegit/
  - הסברים לשימוש: http://tortoisegit.org/docs/tortoisegit/tgit-dug.html

- משימת היכרות עם git
  - https://try.github.io/levels/1/challenges/1

קישורים חשובים:

- האתר של Bitbucket: https://bitbucket.org/
- תיעוד Bitbucket:

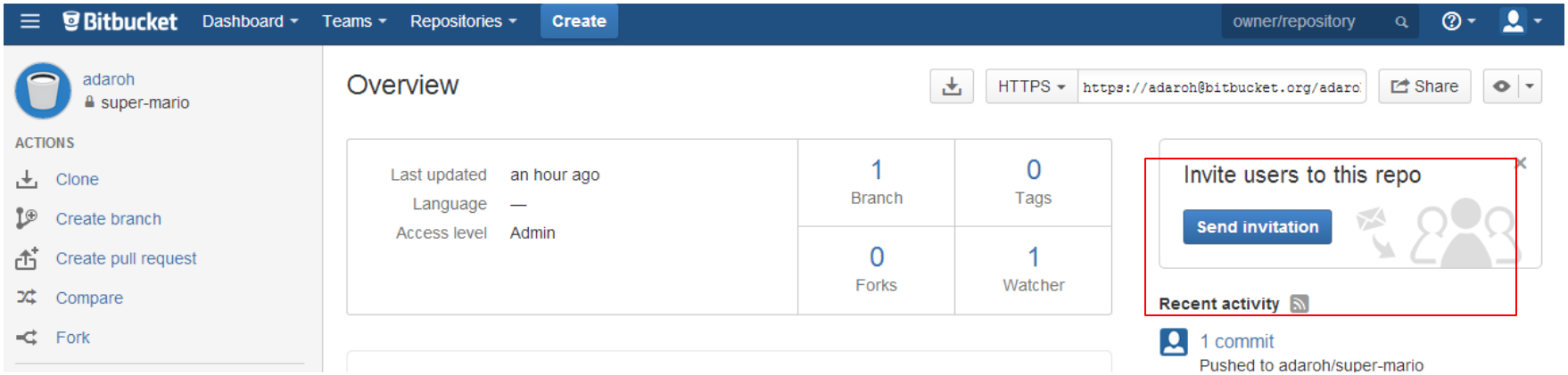
https://confluence.atlassian.com/display/BITBUCKET/Bitbucket+Documentation+Home

- הספר הרשמי של git: http://git-scm.com/book

- מדריך ל git:

https://www.atlassian.com/git/tutorial

צירוף אנשים ל repo



דגשים לעבודה נכונה

- כל הקוד צריך להיות מנוהל דרך git -
  - לא ע"י העברת קבצים ברשת / דיסק-און-קי / מייל וכדומה!
- ביצוע commit-ים במהלך העבודה (גם באמצע פיתוח)
  - בתנאי שהקוד עובד ולא קורס או מבולגן
- תיאורים אינדקטיביים בהודעות ה commit
- ביצוע מידי פעם push לשרת
  - כדי שעבודתכם תמיד תהיה מגובה
- במידה ועובדים על כמה פיצ'רים במקביל – מומלץ לפתוח branch חדש לכל אחד