

# לימוד ל לעבוד עם GIT

סיכון זה מבוסס על ווביינר שהשתתפתי בו של קהילת COLA לijk, בהנחיית יונתן איזנשטיין.

להלן קישור להקלטה ביאוטיוב: <https://www.youtube.com/watch?v=pnik7cUrWSQ>

# Topics

- What is - Version control, Git, Github
  - Base commands - git init, git add, git commit
  - Track changes commands - git status, git diff, git log
  - New Github project
  - Branching - git branch, git checkout
  - Understanding CR's and PR's
  - Resolving conflicts
  - Conventions
  - Summery + QA
- מה זה גיט  
version control
- פקודות בסיסיות בגיט - git init, git add, git commit  
- מעצב אחרי השינויים באמצעות git diff, git status, git log
- שמות נכונים של קומיטים  
- ייצירה של פרויקט חדש בgithub לעדכן אותו עם התוכן שיצרנו. הזדהות עם SSH ולמה צריך את זה.
- יציאה ל"ענפים" חדשים באמצעות - git branch, git checkout. ושמות נכונים לענפים.  
- מה זה cr, zk איך עושים, ומה ההבדל בין merge ל squash and merge כוחפים ענף חדש לענף הראשי שלנו.
- איך לפטור קונפליקטים בין ענפים שונים.  
- אם יהיה זמן נעבור על git stash, git reset.
- סיכום

# Version control system

Version control systems are a category of software tools that helps in recording changes made to files by keeping a track of modifications done to the code.

Usually, version controls systems record all file changes into a special "database" called a **repository**. There we can see all the file's history, file changes, who changed them, and even why.



# Benefits

- We control all changes made to the files.  
And if needed, we can revert back the project.
- Every team member has their copy of the project,  
and can add code without overriding other members.
- Informs us about Who, What, When, Why changes  
have been made.
- All team members can access the projects  
from anywhere.



# Git

Is the most popular version control system in the world.

## Why?

- Free
- Open source
- Fastest
- Scalable
- Easy to learn



# Github

It is the most famous platform for hosting projects using git.  
And provides extra features to make our life better.



# First steps

First, we need to configure git with our credentials.  
that way, git will identify us and sign changes we made with  
our credentials

```
-> git config --global [user.name || user.email]
```

And optional but recommended to configure the main editor

```
-> git config --global core.editor "code --wait"
```

for configuring vscode



# First steps

To start working with **git**, we need to create a **repository** for the project. the command to create an empty repository is

-> **git init**

we need to run this command when we are in our projects folder. Git will create an empty **repo** in the folder ( we can see it in the **.git** hidden file ), and git will start tacking for changes.



```
dev/efrohim/git at 🖥 docker-desktop
→ mkdir gitExample
```

```
dev/efrohim/git at 🖥 docker-desktop
→ cd gitExample
```

```
efrohim/git/gitExample at 🖥 docker-desktop
→ code .
```

```
efrohim/git/gitExample at 🖥 docker-desktop
→
```

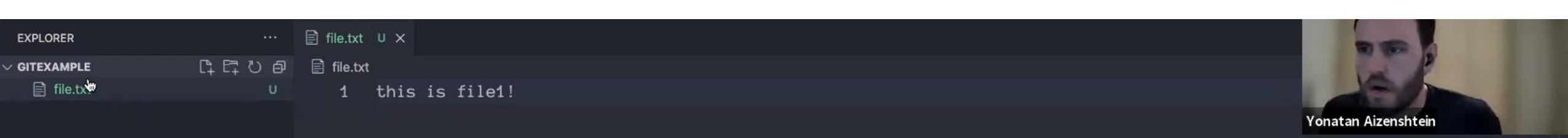
יצירת תיקיה חדשה מהקומנד ליין ופתחתו ב- vscode. יצירה קובץ חדש בתיקיה זו.

```
efrohim/git/gitExample at 🖥 docker-desktop
→ ls
```

```
efrohim/git/gitExample at 🖥 docker-desktop
→ echo "file1" >> file.txt
```

```
efrohim/git/gitExample at 🖥 docker-desktop
→ ls
file.txt
```

```
efrohim/git/gitExample at 🖥 docker-desktop
→ clear
```



**A** - Added (This is a new file that has been added to the repository)  
**M** - Modified (An existing file has been changed)  
**D** - Deleted (a file has been deleted)  
**U** - Untracked (The file is new or has been changed but has not been added to the repository yet)  
**C** - Conflict (There is a conflict in the file)  
**R** - Renamed (The file has been renamed)  
**S** - Submodule (In repository exists another sub repository)

A screenshot of the VS Code terminal window. The terminal shows a sequence of commands being run in a Zsh shell on a Docker Desktop container. The commands include 'git init', 'ls -a', and 'open .git'. The output shows the creation of a new Git repository and its contents. A message from 'Shay Pich' is visible at the bottom of the terminal window.

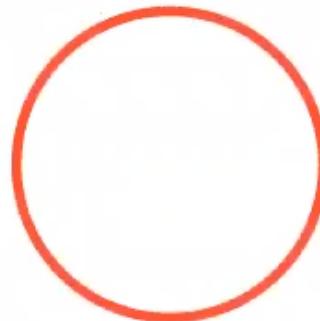
```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE zsh + □ ×  
/Users/yonatanaizenshtein/.oh-my-zsh/oh-my-zsh.sh:source:128: no such file or directory: /Users/yonatanaizenshtein/.oh-my-zsh/themes/spaceship.zsh-theme  
fatal: not a git repository (or any of the parent directories): .git  
  
efrohim/git/gitExample at docker-desktop  
→ git init  
Initialized empty Git repository in /Users/yonatanaizenshtein/dev/efrohim/git/gitExample/.git/  
  
gitExample on master [?] at docker-desktop  
→ ls -a  
.' .. .git file.txt  
  
gitExample on master [?] at docker-desktop  
→ open .git  
  
gitExample on master [?] at docker-desktop  
→ Shay Pich
```

# Workflow

When we work on our project, we all have a similar workflow. We make changes to our files. Then we want to save them to our repository. working with git, we have another special step between the file changes to saving the into the repo called **the staging area**



Project



Staging area



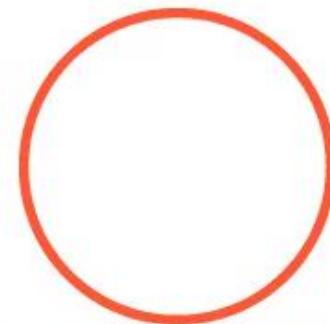
Repository



# Basic commands



Project



Staging area



Repository

```
-> git add <filename>
```



# Basic commands



Project



Staging area



Repository

```
-> git add file 1
```

אם מבצעים שינויים בקובץ המקורי אז נוצרך לעשות שוב add.



# Basic commands



Project



Staging area



Repository

```
-> git add file 1
```

```
-> git commit -m "my message"
```

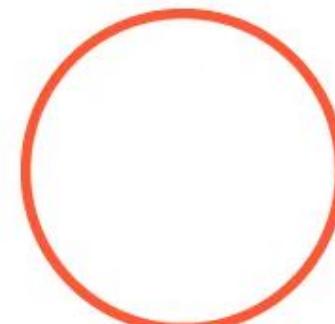
למשל "ערכנו את הקובץ". צריך שזה יהיה אינפורטטיבי.



# Basic commands



Project



Staging area



Repository

```
-> git add file 1
```

```
-> git commit -m "my message"
```



EXPLORER ... file.txt X

GITEMPLETE file.txt

file.txt

1 this is file1!fdsafdas

~/dev/efrohim/git/gitExample/file.txt

PROBLEMS 1 OUTPUT TERMINAL DEBUG CONSOLE

```
gitExample on master [+] at docker-desktop
→ git commit -m "added new file - file.txt"
[master (root-commit) ac1877b] added new file - file.txt
  1 file changed, 1 insertion(+)
  create mode 100644 file.txt
```

```
gitExample on master [?] at docker-desktop
→
```

Shay Pich

GITEXAMPLE

file.txt  
file2.txt

M You, seconds ago | 1 author (You)  
1 fdsafdasfda You, seconds ago • Uncommitted changes

## Track changes

To check our “git” state we can use the command

-> git status

It lets you see which changes have been staged, which haven’t, and which files aren’t being tracked by Git.

```
gitExample on master [!] at docker-desktop
→ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   file2.txt

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

gitExample on master [!] at docker-desktop
→ █
```

.modified - בא לעדכן מה מצב הרפזיטורי שלנו. M -

Shay Pich

GITEXAMPLE

file.txt  
file2.txt M  
file3.txt A

file3.txt  
You, seconds ago | 1 author (You)  
1 new file 3 You, seconds ago • Uncommitted changes

PROBLEMS 3 OUTPUT TERMINAL DEBUG CONSOLE

זהו קיזור - מכניס את כל מה שקיים כרגע ל-.staging area

gitExample on master [!?] at docker-desktop  
→ git add \_

gitExample on master [+] at docker-desktop  
→ clear

EXPLORER

GITEXAMPLE

file.txt  
file2.txt M  
file3.txt A

file3.txt  
You, seconds ago | 1 author (You)  
1 new file 3 You, seconds ago • Uncommitted changes

PROBLEMS 3 OUTPUT TERMINAL DEBUG CONSOLE

gitExample on master [+] at docker-desktop  
→ git status  
On branch master  
Changes to be committed:  
(use "git restore --staged <file>..." to unstage)  
modified: file2.txt  
new file: file3.txt

Shay Pich

gitExample on master [+] at docker-desktop  
→

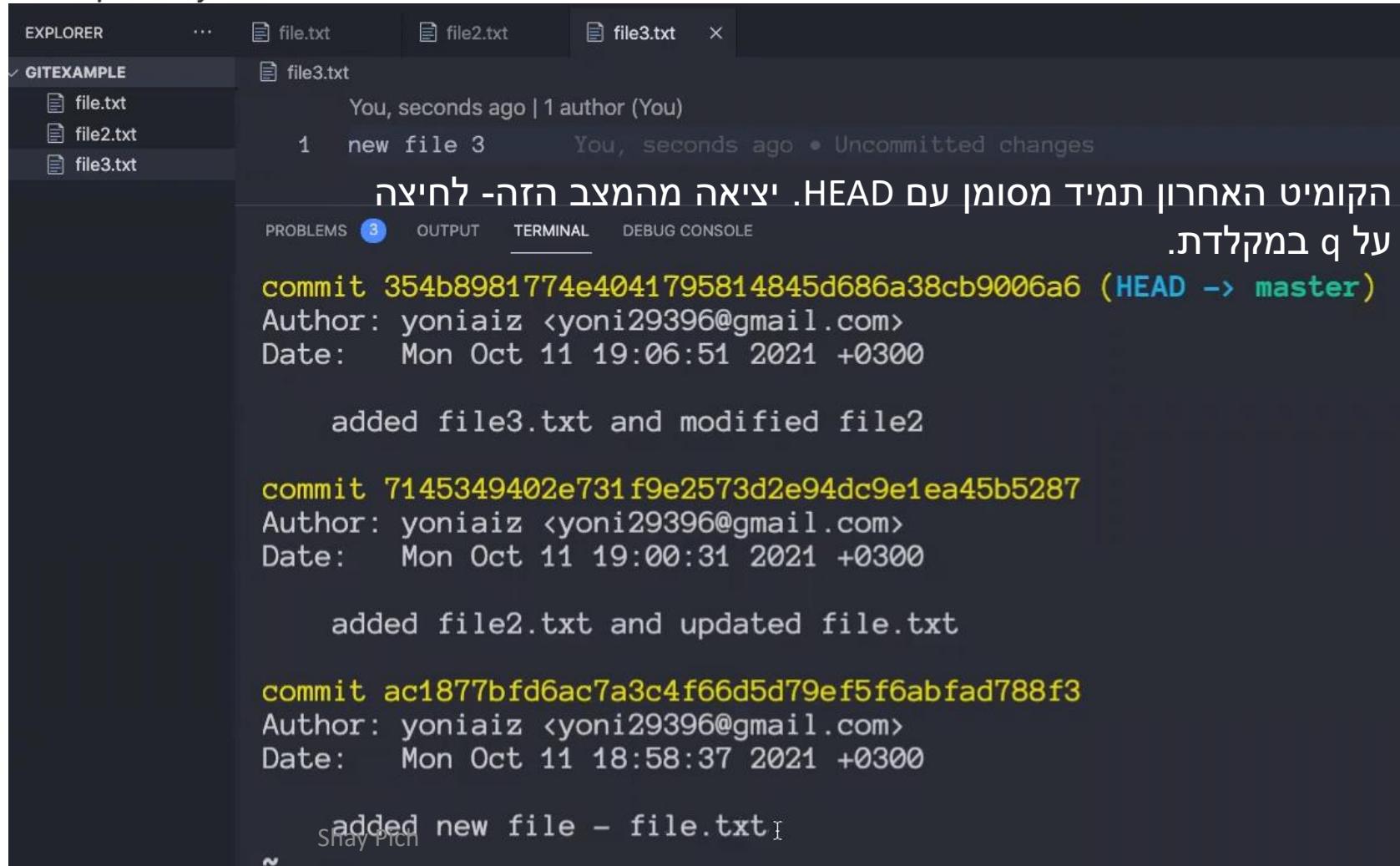
# Track changes

To see all our commit we can use the command

```
-> git log
```

It will output a list with all commit history on the repository

( to exit from the command press Q )



```
EXPLORER ... file.txt file2.txt file3.txt
GITEXAMPLE file3.txt
file.txt
file2.txt
file3.txt

You, seconds ago | 1 author (You)
1 new file 3 You, seconds ago • Uncommitted changes

הקומיט האחרון תמיד מסומן עם HEAD. יציאה מהמצב זהה - לחיצה
על q במקלדת.

commit 354b8981774e4041795814845d686a38cb9006a6 (HEAD -> master)
Author: yoniaiz <yoni29396@gmail.com>
Date: Mon Oct 11 19:06:51 2021 +0300

    added file3.txt and modified file2

commit 7145349402e731f9e2573d2e94dc9e1ea45b5287
Author: yoniaiz <yoni29396@gmail.com>
Date: Mon Oct 11 19:00:31 2021 +0300

    added file2.txt and updated file.txt

commit ac1877bfd6ac7a3c4f66d5d79ef5f6abfad788f3
Author: yoniaiz <yoni29396@gmail.com>
Date: Mon Oct 11 18:58:37 2021 +0300

    added new file - file.txt
```

# Track changes

To see changes between commits we can use the command

```
-> git diff <commit hash1> <commit hash2>
```

It will output all changes between the selected commits

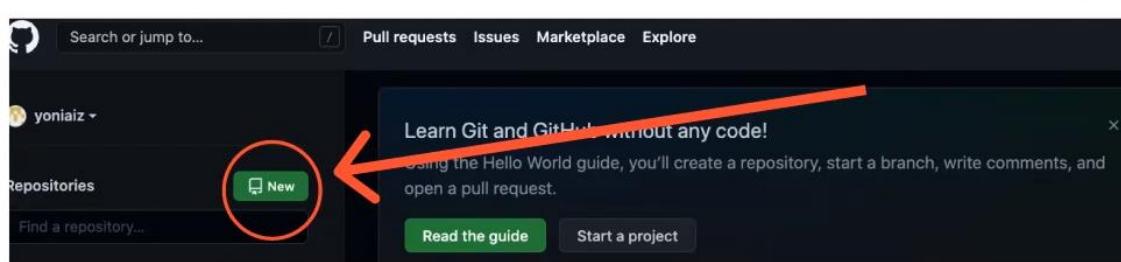
```
gitExample on ① master at ② docker-desktop took 1m 33s
→ git diff file3.txt
```

בערךן לא שימושי מכיוון שיש לנו את גיט, אבל זה משמש לבדיקת  
שוני בין ההיסטוריה קומיטים של קובץ מסוים.

# New Github repo

To create a new Github **repo** we need to login into Github

And press on the **New** button on the left side



## How SSH works?



האפשרות השלישי עברור אחד שכבר קיים אצלנו במחשב - מבדיקים כל פקודה אחת אחד לעורך או בשורת הפקודה.

לאחר מכן הכל כבר נראה בגיטהאב.

ssh - בשайл שגיטהאב יזהה שהמחשב הוא באמת שלי למן תקשורת - נעשה על ידי מפתח אחד במחשב ואחד אצל גיטהאב.

Shay Pich

### Quick setup — if you've done this kind of thing before

Set up in Desktop   or    HTTPS    SSH    git@github.com:yoniaiz/gitRepo.git

Get started by creating a new file or uploading an existing file. We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

### ...or create a new repository on the command line

```
echo "# gitRepo" >> README.md  
git init  
git add README.md  
git commit -m "first commit"  
git branch -M main  
git remote add origin git@github.com:yoniaiz/gitRepo.git  
git push -u origin main
```

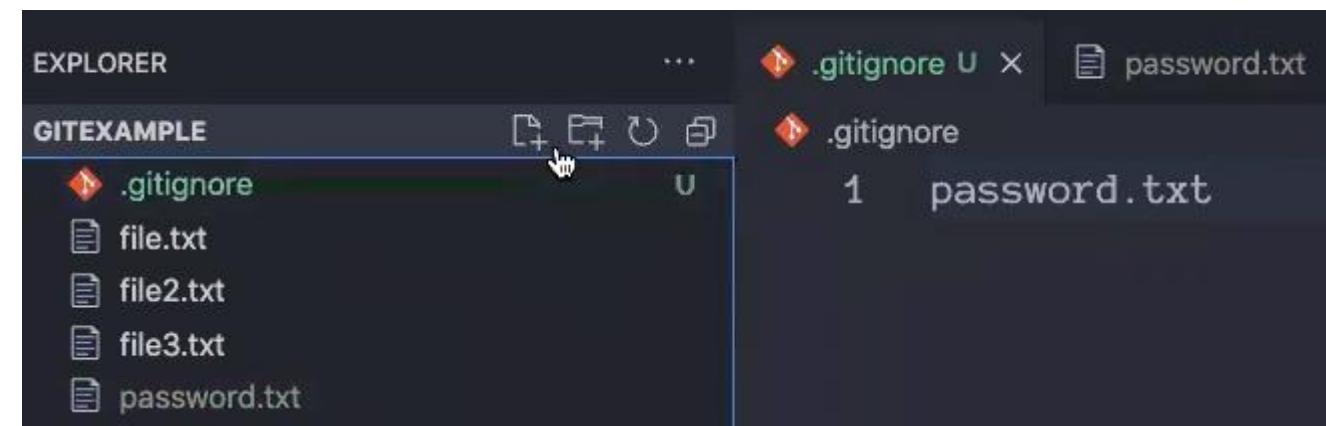
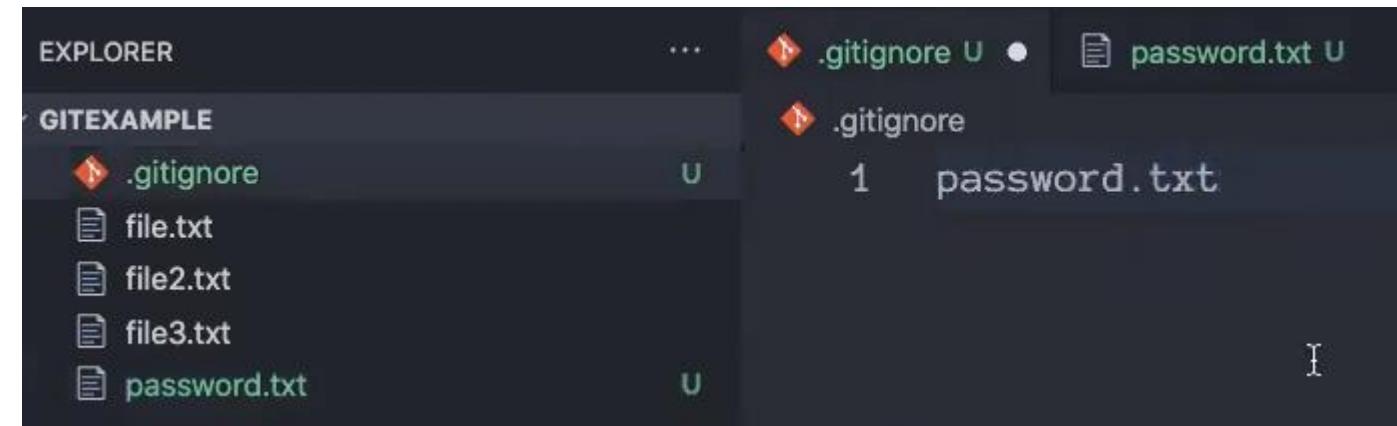
### ...or push an existing repository from the command line

```
git remote add origin git@github.com:yoniaiz/gitRepo.git  
git branch -M main  
git push -u origin main
```

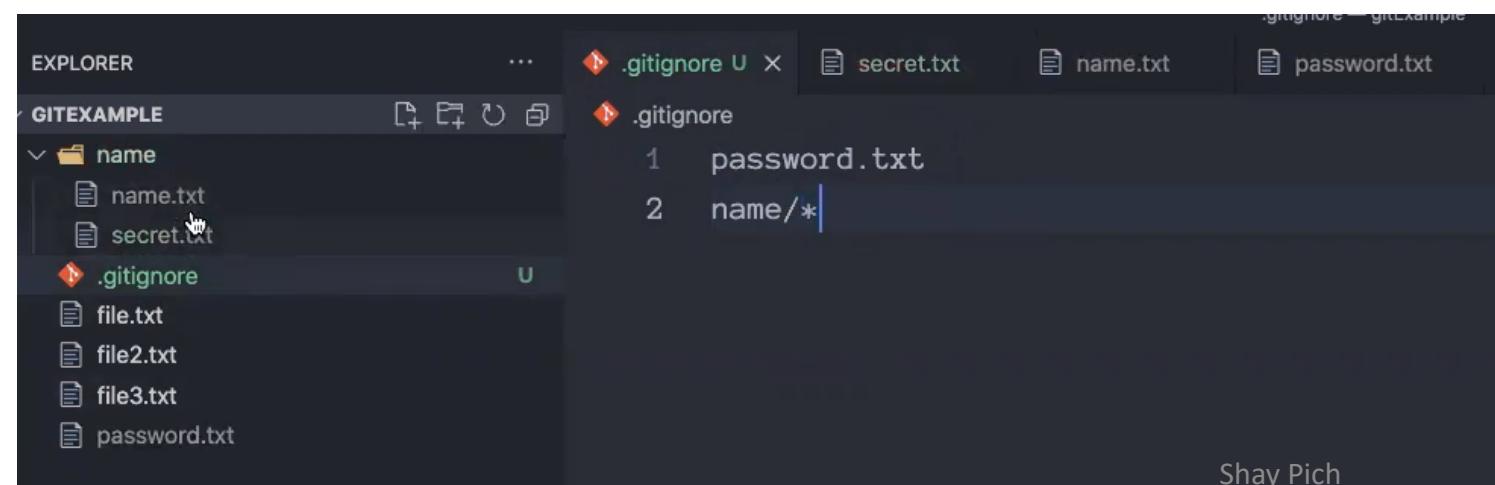
### ...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code



- נשיר אליו קבצים שאנו אף פעם לא נרצה שיישמרו בתוך הרפזיטורי שלנו, שם יהיו רק על המחשב שלנו ולא על השרת (למשל קבצי סימאות או קבצים עצומים גנריים).



✓ GITEXAMPLE

> name  
> node\_modules  
  ↳ .gitignore  
  file.txt  
  file2.txt  
  file3.txt  
  package-lock.json  
  package.json  
  password.txt  
  README.md

README.md > # Title

You, seconds ago | 1 author (You)

1 # Title  
2  
3 description  
4

דחיפת שינויים לרפוזיטורי שלנו שכבר קיים- אז אין צורך כבר ב-ט. כעת הכל עודכן בשרת גיטהאב.

```
gitExample on ⚡ main [!] is 📁 v1.0.0 via 💎v12.22.0 at 🌐 docker-desktop
→ git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 16 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 1.08 KiB | 1.08 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0)
To github.com:yoniaiz/gitRepo.git
  7eef8d1..74e413c  main -> main
```

```
gitExample on main is v1.0.0 via v12.22.0  
→ git branch "feature/login-page"
```

## Branches

Every project is built from a lot of small features combined. In the development process, we develop one feature, then integrate it into the “main” project, and make all pieces work together.

When we develop a feature, we want to make sure we work in an isolated environment from our main project to ensure we won't affect the current working program.

```
gitExample on feature/login-page is v1.0.0 via v12.22.0 at  
→ git commit -am "updated file.txt file2.txt file3.txt"  
[feature/login-page 72805ad] updated file.txt file2.txt file3.txt  
3 files changed, 4 insertions(+), 3 deletions(-)
```

```
gitExample on feature/login-page is v1.0.  
→ git checkout main  
Switched to branch 'main'  
Your branch is up to date with 'origin/main'.
```

Also many times we work in a team, where each member work on their feature. and we need to make sure that we don't override each others code

To help us with this process Git gives us the command

-> git branch

חומר חשוב. כאמור ענף ה-main הוא הענף הראשי והוא זה שמובא ללקוט, לכן הוא רגיש. איזי, נרצה שלא לעבוד עליו שירות אלא לבחון אותו בענף צד טרילי. am- זה קיצור ל- add&commit + הودעה. השינויים לא יוצגו ב-main.

Shay Pich

This command lets us create a new branch that is a snapshot of the current branch we went to a branch from. that way, we can change our code however we want, without affecting the main branch.

# Branches

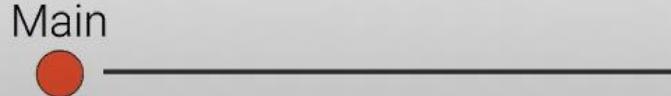
As we discussed earlier, on the GitHub repo section. We changed our master branch name to main ( was optional ). So all this time we already worked in a branch, called master or main. and this is our “main” branch



We want to create a new feature for our project, for example, the login/register page. But we want to make sure our main branch stays in its current state.

We can run the command

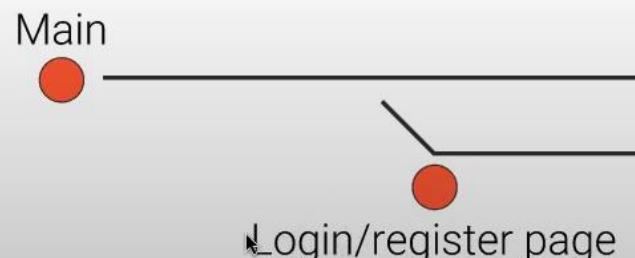
And that way, we will create a branch that is a snapshot of our current branch. That way, we will work in an isolated environment for this feature.



To start working on this new branch we need to run the command

-> **git checkout <the branch name>**

To start working on this new branch, we need to run the command.



We can keep doing this process as many times that we need, and we can checkout between all branches that we created at any moment in time

To see all branches we can run the command

-> **git branch -a**

```
feature/login-page  
* [main]  
  remotes/origin/main  
(END)  
Shay Pich
```

הענף שקיים ברפזיטורי בגיטהאב

- + Create new branch...
- + Create new branch from...
- ↗ Checkout detached...

main cb754542

feature/login-page 72805ad1

feature/register-page 74e413c9

origin/main Remote branch at 74e413c9

צ'קאאוט ישיר שלא מהkomend ליין - בראש העורר,  
צפיה בכל הענפים שקיימים.

```
gitExample on 0 main [↑] is 📦 v1.0.0 via 🏠 v12.22.0
→ git branch -d feature/register-page
Deleted branch feature/register-page (was 74e413c).
```

```
→ git checkout -b "test-branch"
Switched to a new branch 'test-branch'
```

מחיקת ענף

קיצור של יצירת ענף חדש ותצא אליו

## Merge branches

מומלץ למחוק כמה שייתר ענפים ללא משמעות, זאת כדי  
שההיסטוריה הקוד תהיה כמה שייתר נקייה.

We have few options on how to merge branches.  
We will talk about two options.  
Option one - merge the branches locally.  
Option two - open pull request

יצירת תת-ענף לענף קודם:

```
gitExample on feature/login-page is v1.0.0 via v1.0.0
→ git checkout -b "new-login-logic"
Switched to a new branch 'new-login-logic'
```

```
gitExample on new-login-logic is v1.0.0 via v1.0.0
→ git commit -am "created new login logic"
[new-login-logic d2f0f55] created new login logic
  1 file changed, 2 insertions(+), 1 deletion(-)
```

cut נרצה למזג בשלבים לאחר שינוי בקוד: רושמים את תת הענף שרצוים למזג.

```
gitExample on feature/login-page is v1.0.0 via v1.0.0
→ git merge new-login-logic
Updating fbeb1c..d2f0f55
Fast-forward
  file2.txt | 3 ++-
  1 file changed, 2 insertions(+), 1 deletion(-)
```

ונשאר רק למחוק את תת הענף שכבר לא שימושי:

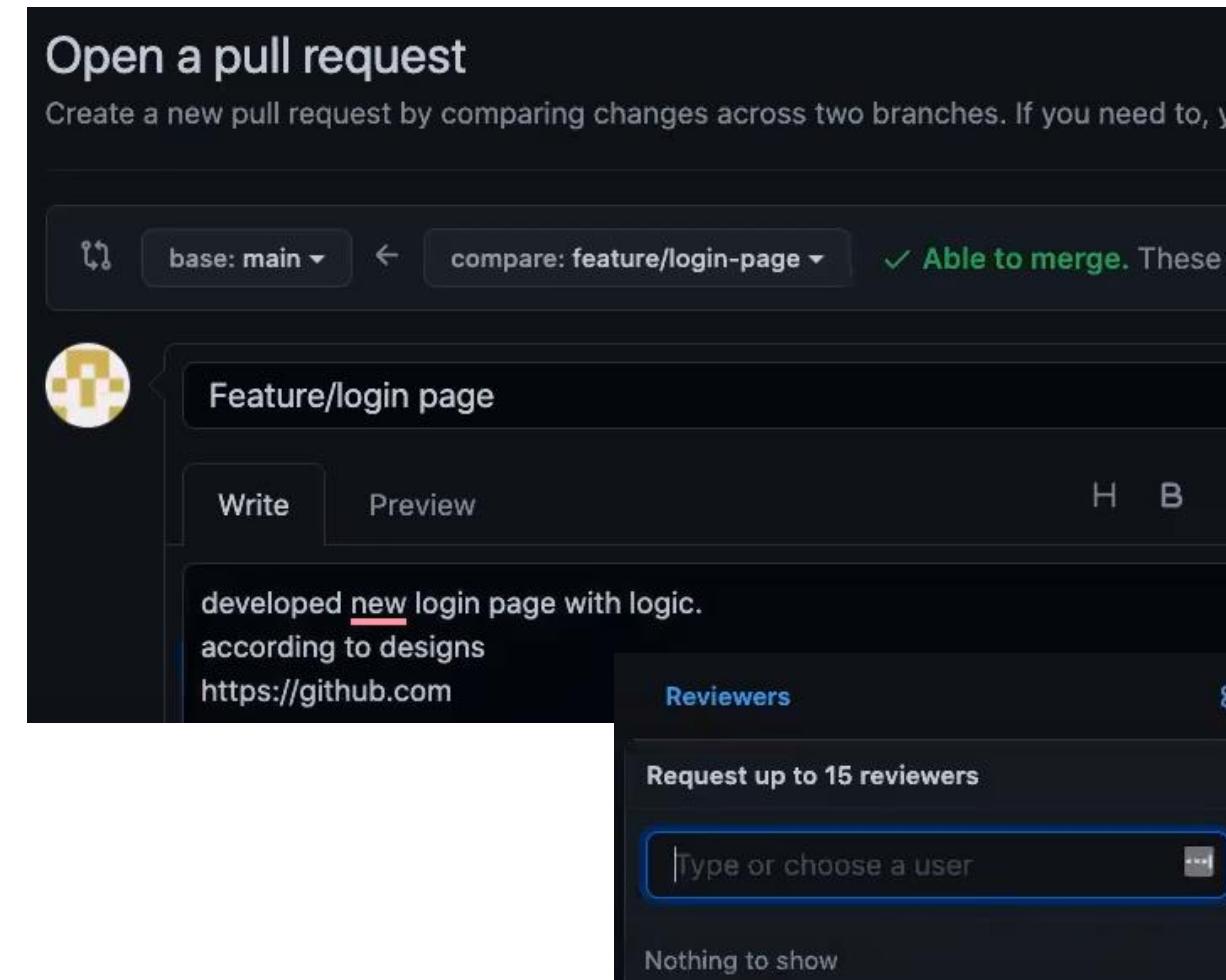
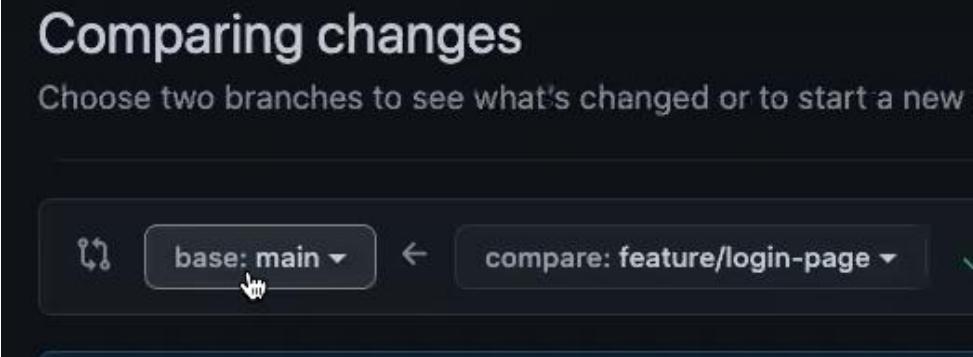
```
gitExample on feature/login-page is v1.0.0 via v1.0.0
→ git branch -d new-login-logic
Deleted branch new-login-logic (was d2f0f55).
```

הערה: צריך להיות על הענף שאנו רוצים למזג אליו כדי למזג ענפים  
לתוכו, ואז רושמים איזה אני רוצה למזג.

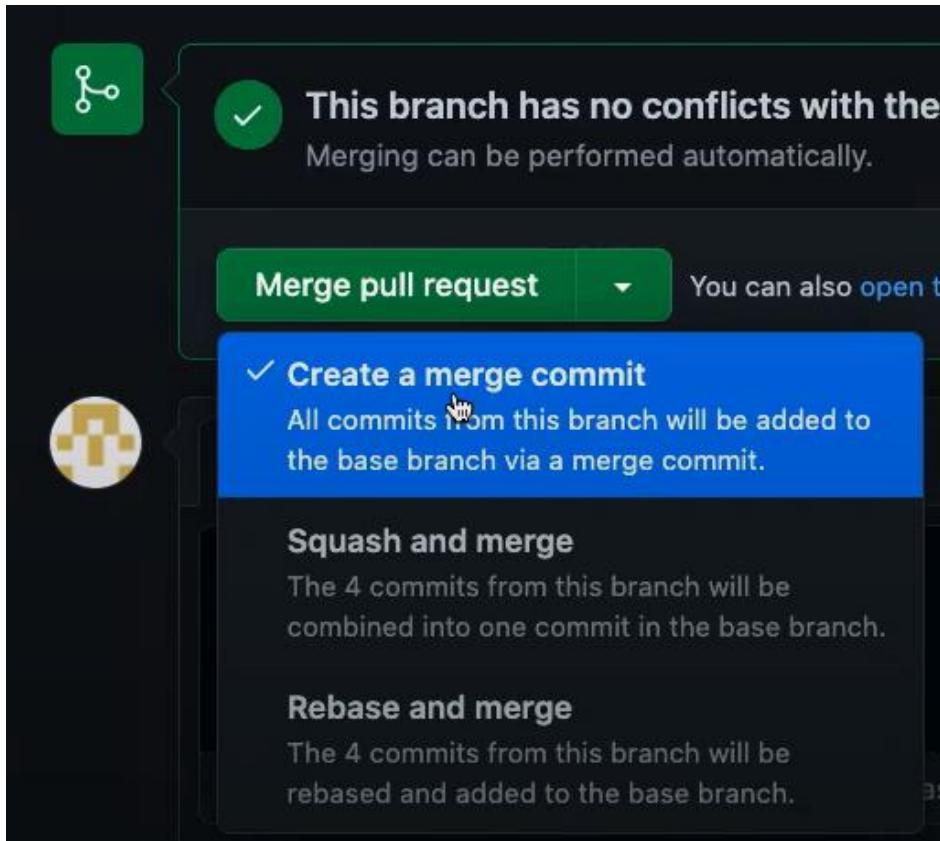
```
gitExample on feature/login-page is  
→ git push origin feature/login-page  
Enumerating objects: 19, done.  
Counting objects: 100% (19/19), done.
```



בפועל ריקווטן: מה שרצים למזג לאן למזג



כאמור בסביבות עבודה כאשר אנו רוצים למזג ענפים ראשיים נוצרה לעשנות pull request. זה מאד נפוץ בפרויקטים של קוד פתוח למשל כדי שלא נהרסו קוד. לאחר קבלת אישור ניתן למזג את הקוד.



המרג' הרגיל - הכנסת כל הלוגים שהיו על הענף לענף הרצוי

הכנסת כל הקומיטים של הענף לקומיט אחד ודחיפה לענף

כעת נשאר לסונכרן בין מה שבשרת לבין מה שבמחשב:

gitExample on main  
→ git pull

GITEXAMPLE

file3.txt

You, seconds ago | 2 authors (You and others)

new file 3

Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes | Start Live Share Session

<<<<< HEAD (Current Change) השינויים שאני עשית!

123

register page

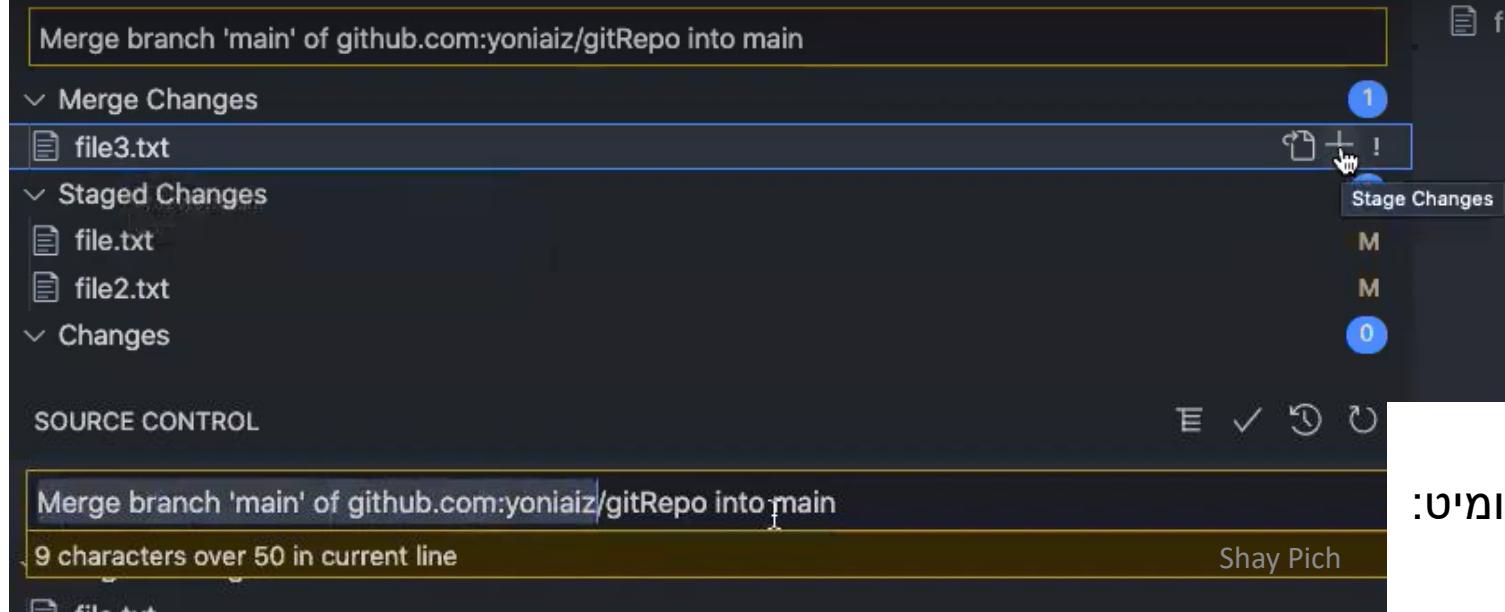
=====

123fdasfdsa

login

update file 3

>>>> 089538df6f408ca0459a99ee0a597f121c947929 (Incoming Change)



אחרי שבוחרים אחת מאופציות הקונפליקט,  
מכנים לסתיג:

קונפליקט - דיסוננס בין שינויים בשרת למחשב.

ולבסוף עושים קומיט:

Add more commits by pushing to the **branch-2** branch on [yoniaiz/gitRepo](#).

This screenshot shows a GitHub pull request interface. At the top, there's a warning message: "This branch has conflicts that must be resolved. Use the [web editor](#) or the [command line](#) to resolve conflicts." To the right is a "Resolve conflicts" button. Below this, under "Conflicting files", it lists "file3.txt". A small icon of a wrench and screwdriver is in the top-left corner.

נגיד וביצענו שינוי בתיקיה כמה פעמים וכל שינוי  
עשינו פול ריקווט, אם אחד מהם אושר אז לפול  
ריקווטים האחרים יוצר קונפליקט. לא כדאי לפתור  
אותו מהגיטהבו? למה?  
כי זה פשוט נכון וגם כי אנו רוצים לבדוק שפתרון  
הكونפליקט עובד! אז נוכל לדוחף אותו.

(בשני ענפים ערכנו את אותה שורה קוד גיט לא יוכל להחליט מי שורת הקוד הנכונה)

נפתרו את הקונפליקט - ניבא את הענף העיקרי ואז נוכל לראות מוחשית את הקונפליקט ולפתור אותו במחשב:

```
→ git pull origin main
```

This screenshot shows a terminal window with a git pull command. The output shows a merge conflict between the local 'main' branch and the remote 'origin/main'. The conflict is in a file named 'file3.txt'. The terminal shows the user navigating through the conflict resolution options: 'Accept Current Change', 'Accept Incoming Change', 'Accept Both Changes', 'Compare Changes', and 'Start Live Share Session'. The 'HEAD (Current Change)' option is highlighted with a yellow dot.

מעבירים לסטיג' לאחר בחירת האפשרות המתאימה (ע"י ה+ בצד שמאל כמו מקודם) ומבצעים קומיט בהתאם:

```
gitExample on [branch-2] (git)-[branch-2]merge]- [!] is 📦  
→ git commit -am "fixed conflict with origin main"  
[branch-2 37d54f7] fixed conflict with origin main
```

```
gitExample on [branch-2] is 📦  
→ git push origin branch-2
```

דוחפים לשרת:  
Shay Pich

אם מוחקים שורה למשל, עושים קומיט ופול אז השרת של גיט יזהה שהקוד אצלנו "חזר בזמן"  
ואז בעצם קיבל את השינוי שעשינו

# Conventions - commits

For commit, we have some guidelines to how we should write a good and informative message, to help us in the future. If we want to add a body to the message - recommended but optional we can run git commit without flags, and it will open our editor in edit commit mode.

עיקרון כל צוות בוחר את הקונבנציות שלו ושיטת העבודה, אבל יש קונבנציות כלליות.

type(scope): subject

BODY  
FOOTER

## Type

**feat** - A new feature  
**fix** - A bug fix  
**docs** - Changes in documentation  
**style** - Style changes, formatting, missing semicolons or whitespaces  
**refactor** - code changes that neither fixes a bug or adds a feature  
**perf** - changes that improve performance  
**test** - Add missing tests  
**chore** - changes the build process

## Body (optional)

Must begin one blank line after the description. Can provide additional contextual information.

Used to indicate breaking changes. For breaking changes the Body must start with "BREAKING CHANGE".

Show Diff

## Scope

A scope is provided in parentheses after a type. A scope is a phrase describing parts of the code affected by the changes. For example "(userservice)".

## Subject

The subject contains a short description of the applied changes

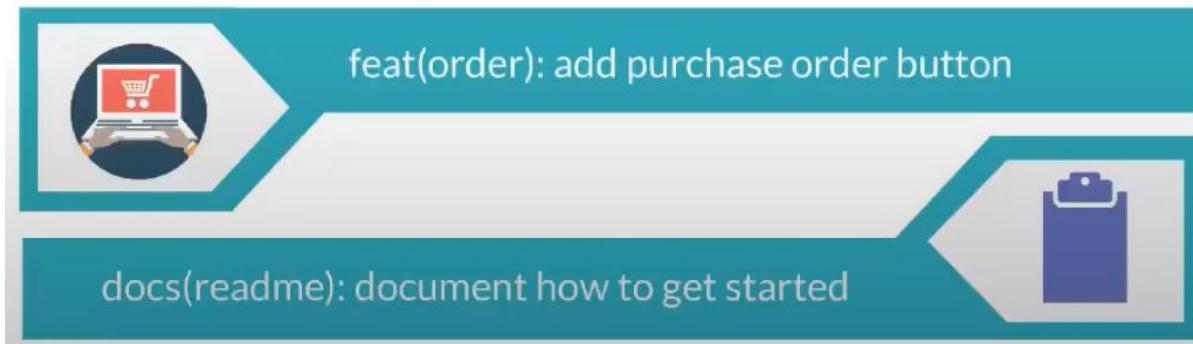
## Footer (optional)

Use to reference issues affected by the code changes. For example "Fixes #13"

Can also be used to indicate breaking changes by starting with "BREAKING CHANGE".

## Example commits

לדוגמאות.



✓ GITEXAMPLE

> name  
> node\_modules  
↳ .gitignore  
file.txt  
file2.txt  
file3.txt  
package-lock.json  
package.json  
password.txt

README.md M

M README.md > # Title

You, seconds ago | 1 author (You)

1 # Title  
2  
3 description  
4  
5 main is the protected branch  
6

PROBLEMS 1 OUTPUT TERMINAL DEBUG CONSOLE

```
gitExample on ⚡ main [$!] is 📦 v1.0.0 via 💎 v12.22.0 at 🌐 docker-desktop
→ git commit -am "docs(readme) - added description about the main branch"
[main d968960] docs(readme) - added description about the main branch
1 file changed, 2 insertions(+)
Shay Dichter
```

# Conventions - commits

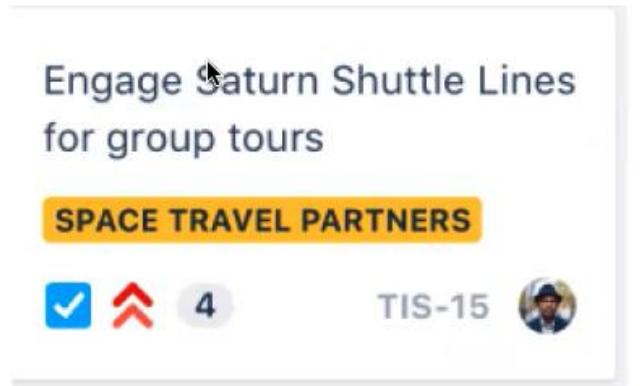
Usually, we work with products like Jira/Trello or even GitHub issues, tickets or tasks created there and assigned to the developers.

Every ticket is created with ID, and it's a good practice to start your commit message with the ID of the ticket.

When we try to debug the commits, we can find the related tickets/tasks in our product and get more details on what has been done.

# Conventions - commits

Jira ticket example



The commit message will look something like this:

ג'ירה - פלטפורמת ניהול משימות.

Also If the project is connected with GitHub it will track changes inside the product.

```
gitExample on ① main [$] is 📦 v1.0.0 via ⬢v12.  
→ git commit -am "TIS-15 feat(bla) - bla"  
[main 64d7739] TIS-15 feat(bla) - bla  
 1 file changed, 3 insertions(+), 1 deletion(-)
```

# Conventions - branches

## Branch types

Regular/ Long-running branches - branches that not deleted from the repository And part of the day to day flows

- master/main - the main branch that contains production ready code
- develop ( not used in all branching stategies ) - contains all development code, get updated by the developers team
- QA ( not used in all branching stategies ) - Test branch, used by the QA team

אודות מהותה ל-

# Branch types

**Temporary / Short-lived branches** - Used for development and deleted when the branch is merged

- Feature branch - used for developing features
- Bug fix - used to fix general bug
- Hot fix - used to fix urgent bug, gets merged to master
- WIP branch or Experimental branches
- Release branches - code ready for release to production

# Conventions - branches

A branch name should be informative and short.

The naming conventions can change but are similar to commits - many workplaces use Jira ( or similar products ) tickets IDs to prefix the branch name - GitHub will also automatically link the branch to the issue ( if configured ).

For example: TIS-15-engage-Saturn-shuttle. Also, sometimes workplace conventions it to add the developer's name:

Yonatan-TIS-15-engage-Saturn-shuttle

Engage Saturn Shuttle Lines  
for group tours

SPACE TRAVEL PARTNERS



4

TIS-15



Shay Pich

# Conventions - branches

## Branching strategies

Examples for different strategies:

- Git flow ( what we will cover )

ותיכון, אך מישן וקצת מעיך  
פופולארי

- GitHub flow

פופולארי

- GitLab flow

פופולארי

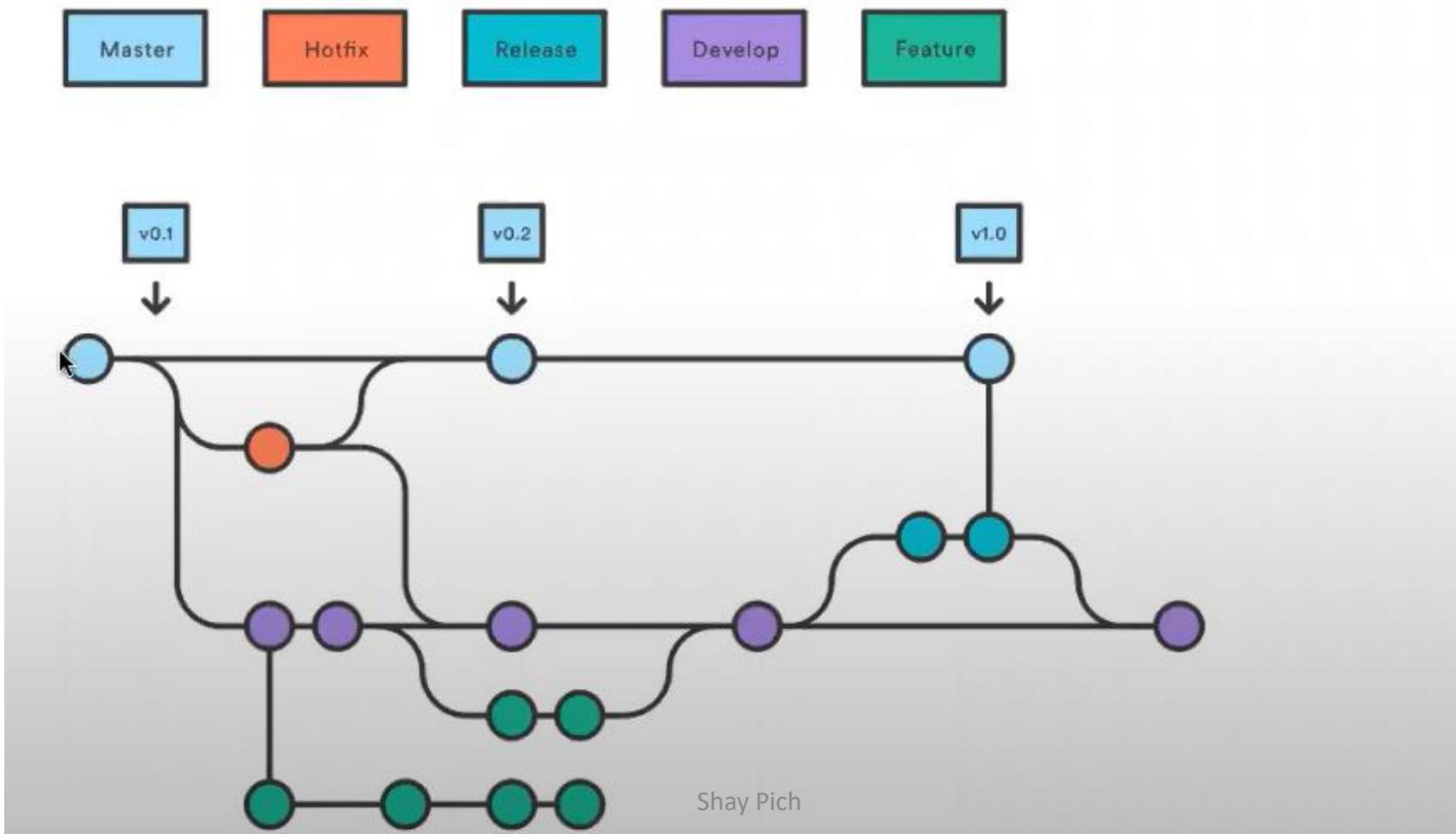
- Trunk-based Development

And more



# Conventions - branches

## GitFlow



# Conventions - branches

## GitFlow

To work with gitflow i recommand using or  
Git GUI helpers like Git Kraken

<https://www.gitkraken.com/>

or use the extantions git-flow-cheatsheet  
<git-flow-cheatsheet/>