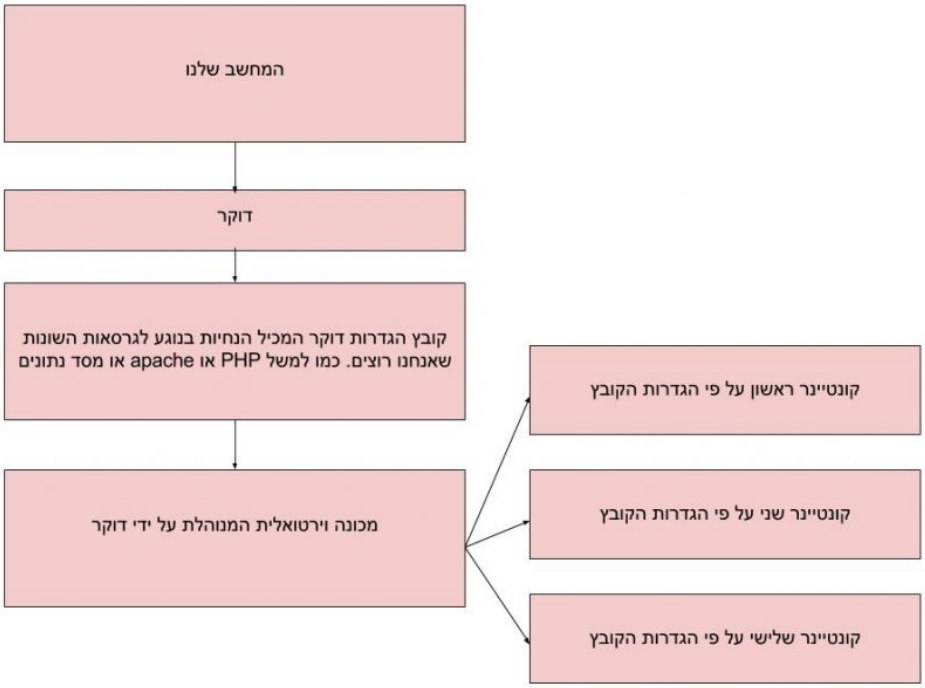
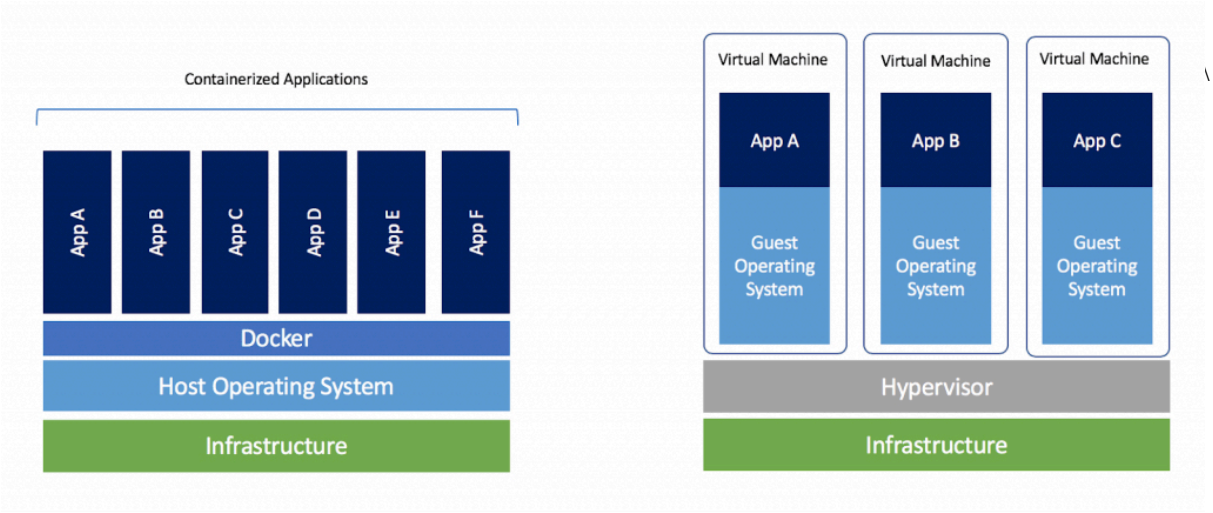
תקציר ל-Docker
כל הזכויות שמורות ל: לחץ כאן (קובץ זה ברובו לא נכתב ע"י!)

דוקר Docker מאפשר לכתוב מה שבא לנו ולהריץ אותו איפה שבא לנו, לדוגמא אם נרצה לכתוב בPHP על חלונות אז זה אפשרי, אם נרצה לכתוב ב-Node אבל לא בא לנו להתקין על מ"ה שלנו Node Version Manages אז זה אפשרי, אם נרצה לפתח וורדפרס אבל לא נרצה להתקין PHP ו-MySQL על חלונות אז זה אפשרי.
דוקר מאפשר לצוות שלם לרוץ על אותה סביבה, במקום לבדוק שלכולם יש אותה גרסה של MySQL או אותה גרסה של Scala או אותה גרסה של לינקס, ניתן לאכוף את אותה סביבת עבודה לכל המפתחים בקלות.

קונטיינר Container – מכונה שמריצה את כל מה שאנחנו צריכים, למשל Php ו-Apace או Node ולינוקס.
כל קונטיינר מוגדר עם קובץ הגדרות, למשל בקובץ ההגדרות כתוב שאנחנו צרכים PHP גרסה 7.
דוקר לוקח את קובץ ההגדרות ויוצר קונטיינר – מכונה וירטואלית שיש בה את מה שאנחנו רוצים ומגדירה כבר עם מערכת הקבצים שמתממשקת אלינו, פורט וכו'.
כך זה נראה:



דוקר מסוגלת לקחת קובץ הגדרות ולשכפל אותו במהירות למכונה או מכונות שנקראות קונטיירים, אבל בפועל לא שונות ממכונה וירטואלית. קונטיינר עולה תוך שניות בעוד שמכונה וירטואלית היא איטית הרבה יותר, קונטיינר מוגדר בקלות באמצעות קובץ ההגדרות בעוד שמכונה וירטואלית לא.



```
docker ps -a
```

מראה את רשימת כל מכונות הדוקר הקיימות.

```
docker run -d -p 81:80 --name <name> -v "$PWD":/var/www/html php:7.0-apache
```

פקודה שלוקחת Docker file והופכת אותו ל- Docker Container כלומר למכונה אמיתית, המחזיקה את האפלקציה שלנו.

הפלאג `-d` מאפשרת למכונה להיות נפרדת מה-CMD

הפלאג `-p` קובע את הפורטים, 81:80 אומר שכל בקשה שמופנית לפורט 81 של הדוקר עוברת לפורט 80 של הקונטיינר, נשתמש בזה אם אנחנו מריצים כמה קונטיינרים במקביל.

הפלאג `--name` בוחר שם למכונה, כל מכונה מקבלת זיהוי מבוסס hash, כיוון שזה לא נכון ניתן לקבוע שם נוסף שיתאים עבור בני אדם.

הפלאג `-v` מקשר בין מערכת הקבצים של המחשב החיצוני למערכת הקבצים של הקונטיינר, כלומר כתוב הנתיב שלי מול כתובת הנתיב של הקונטיינר, במקרה זה יש משתנה סביבה שמפנה למסמכים שלי, אך ניתן לעשות גם פקודה כזאת:

```
docker run -d -p 80:80 --name <name> -v /c/temp:/var/www/html php:7.0-apache
```

הרצה של הפקודה תתחיל להתקין דרך קובץ הדוקר את כל מה שאנחנו צריכים, זה לא מתקין על המחשב המקומי אלא יוצר סוג של מערכת וירטואלית, ההורדה היא חד פעמית של קובץ הקבצים שהדוקר צריך, אם נרצה ליצור כמה מכונות אז הדוקר ישתמש במה שכבר מותקן.

הרצה של MySQL מתבצע ע"י הפקודה:

```
docker run --name mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:5.6
```

`docker run -name mysql` – הפקודה של docker run היא יצירת הקונטיינר, מה שמגיע אחרי הפלאג של name הוא השם שאנו בוחרים לקונטיינר הזה.

הפלאג `-e` מאפשר להכניס משתני סביבה לתוך הקיינטרים עם יצירתו, במזרה זה נכניס משתמש סביבה `MY_ROOT_PASSWORD` שניתן להשתמש בו בגלל הדוקונטציה של קובץ ההגדרות הרשמי.

הפלאג `-d` אומר שהקונטיינר ירוץ ברקע
`mysql:5.6` השם הרשמי של הקונטיינר, מה שמגיע אחרי הנקודתיים הוא הגרסה. במקרה הזה השתמשתי ב-5.6 אבל בדוקומנטציה יש עוד גרסאות ושמות. השם הזה והגרסאות מופיעות בדוקומנטציה.
לאחר הפקודה ניתן להריץ את:

```
docker ps -a
```

ולראות שהקונטיינר פעיל:

תקציר ל-Docker

כל הזכויות שמורות ל: לחץ כאן (קובץ זה ברובו לא נכתב ע"י!)

```
$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS                      PORTS               NAMES
daa8e1cdf73e        mysql:5.6           "docker-entrypoint..."   20 minutes ago      Up 20 minutes               3306/tcp            mysql
c22534dcc868        php:7.0-apache      "docker-php-entryp..."   26 hours ago        Exited (0) 5 hours ago                          my-apache-php-app
08b23bd8c190        wordpress           "docker-entrypoint..."   27 hours ago        Exited (137) 27 hours ago                       some-wordpress
```

המשך הרצה של MySQL: כאן

הערה: ברגע שניצור את הקונטיינר מחדש כל העדכונים שעשינו בנתונים ימחקו, ולכן בד"כ נשתמש בדאטא קונטיינר.

תמונה (Docker Image)

זהו בעצם קובץ הגדרות שקובע על הקונטיינר להתקין מה שנרצה, ניתן להסתמך על קובץ הגדרות רשמי ומוכר או לחלופין ליצור קובץ הגדרות משלי, קובץ ההגדרות זה Image ומה-Image ניצור קונטיינר.

Images are created with the build command, and they'll produce a container when started with run.

הגדרה:

Docker container. An image is essentially built from the instructions for a complete and executable version of an application, which relies on the host OS kernel. When the Docker user runs an image, it becomes one or multiple instances of that container.

למה ליצור תמונה? לדוגמא אפליקציה שיש בה גם PHP וגם Python או נרצה להרים לפעמיים קונטיינר למשהו ספציפי שיעשה עבודה אחת מאוד פשוטה כמו הרצה של `Static code analysis` על קבצי PHP, וכך ניתן לעשות בדיקה ללא התקנה.

מטרה: ליצור תמונה שמייד תיצור קונטיינר על מנת שנוכל לבדוק מה שנרצה או להריץ מה שנרצה.

כיצד עושים זאת?

לרשום קובץ הגדרות (תמונה) שאומר "קח את הקונטיינר הרשמי והתקן עליו PHPCS ועוד משהו"

ניצור קובץ שנקרא `Dockerfile` (עם אות גדולה בהתחלה) ונציב אותו בפרויקט שלנו, הקובץ זה הוא סוג של "תבנית" שבתוכו אפשר להכניס הוראות שונות שיעבדו ממש כאילו אנחנו בתוך המכונה.

המשך קריאה על דוגמא לקובץ: לינק

לאחר כתיבת הקובץ נצטרך להריץ

```
docker build -t ran-lint-machine .
```

כאשר הפקודה `docker build` אומר שאנחנו 'בונים' כאשר הבנייה בעצם עוקבת אחרי כל ההוראות שיש בקובץ ומריצה אותו. אם הכל מצליח ניתן ליצור קונטיינר על הפקודה `run`

ניתן לראות את כל **התמונות** ע"י הפקודה הבאה:

```
docker images --all
```

הבעיה מתחילה כאשר אני צריך כמה קונטיינרים בו זמנית. למה צריך את זה? למשל אם אני רוצה להרים אתר וורדפרס שצריך גם שרת PHP וגם שרת שיש בו MySQL. או למשל אני צריך לבחון קובץ SQL שקיבלתי. אני צריך PHPMyAdmin שהוא ממשק ניהול ל-MySQL וכמובן שרת שיש בו MYSQL. מה אני עושה?

ניתן להרים שני קונטיינרים באופן עצמי, אבל בסופו של דבר **דוקר** ≡ **פשטות** ולכן במקום להציע עם הוראות הכנה מסובכות בכל פעם שנרצה להרים משהו, ניתן לכתוב קובץ `docker compose file` אשר זהו קובץ הגדרות בפורמט `yaml` (פורמט של קובץ הגדרות) שניתן להריץ בשורה אחת

פותר את הבעיה של ליצור כמה קונטיינרים שצריך להריץ בו מקביל

בגדול יוצרים קובץ בשם `docker-compose.yml` ומציבים אותו בתקייה כאשר הקובץ נראה כך:

תקציר ל-Docker
כל הזכויות שמורות ל: לחץ כאן (קובץ זה ברובו לא נכתב ע"י!)

```
version: '2'
# כאן יש לנו את כל הקונטיינרים
services:
# קונטיינר של מסד נתונים
  db:
    image: mysql  (from https://hub.docker.com/)
    environment:
      # Environment variable
      MYSQL_ROOT_PASSWORD: password
    ports:
      - "3306:3306"
# PHPMyAdmin container
  hpMyAdmin:
    image: phpmyadmin/phpmyadmin
    depends_on:
      - db
    ports:
      - "8080:80"
```

סרטון כיצד להריץ Wordpress מההתחלה: https://youtu.be/LoOjhNhRbfI

דוגמת הרצה קלה (להבנה בסיסית)

https://blog.hipolabs.com/understanding-docker-without-losing-your-shit-cf2b30307c63

לסיכום:

דוקר – **הפלטפורמה** שמאפשר לכתוב מה שבא לנו ולהריץ אותו איפה שבא לנו

קונטיינר - **מכונה שמריצה** את כל מה שאנחנו צריכים

Dockerfile – **הקובץ** זה הוא סוג של "תבנית" שבתוכו אפשר להכניס הוראות שונות שיעבדו ממש כאילו אנחנו בתוך, משתמש עבור **תמונה**

כלומר לסיכום:

תמונה – **Dockerfile מקומפל**, חוסך את הזמן של לבנות מחדש Dockerfile כל פעם שנרצה להריץ קונטיינר.

Images are created with the build command, and they'll produce a container when started with run.

Dockerfile > (Build) > **Image** > (Run) > **Container**.

- **Dockerfile**: contains a set of docker instructions that provisions your operating system the way you like, and installs/configure all your software's.
- **Image**: compiled Dockerfile. Saves you time from rebuilding the Dockerfile every time you need to run a container. And it's a way to hide your provision code.
- **Container**: the virtual operating system itself, you can ssh into it and run any commands you wish, as if it's a real environment. You can run 1000+ containers from the same Image.

docker-compose - פותר את הבעיה של ליצור כמה קונטיינרים שצריך להריץ בו מקביל שמתקשר בינהם (קובץ הגדרות בפורמט `.yaml`) לדוגמא כשצריך גם PhpMyAdmin גם MySQL וגם Wordpress