Docker

4:44 PM Monday, July 22, 2024

- Docker File

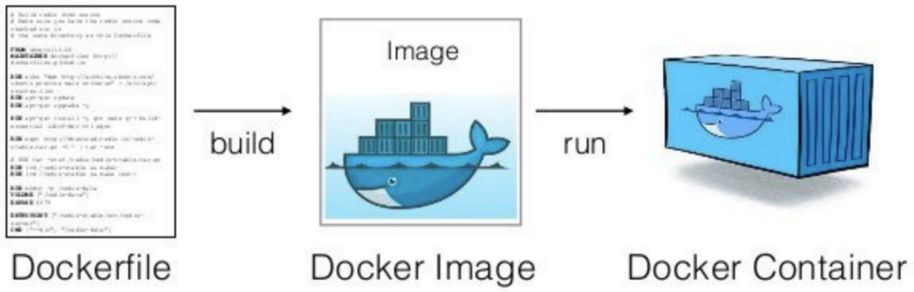
Docker File, הוא קובץ טקסט המכיל הוראות לבניית תמונה (Docker Image). הוא מגדיר את הסביבה, התלויות וכל מה שנדרש ליצירת קונטיינר מבוסס התמונה. זה כמו מתכון המפרט את השלבים והמרכיבים הדרושים להכנת מנה (במקרה שלנו המתכון הוא ה-Docker File והמנה היא ה-Docker Image).

- Docker Image

Docker Image היא תבנית קריאה בלבד המכילה את כל מה שנדרש להרצת האפליקציה - קוד, ספריות, כלים, תלויות ועוד... זה כמו תבנית להקמה של קונטיינרים. ניתן ליצור תמונות באמצעות Docker File או להוריד תמונות מוכנות מראש ממאגר כמו Docker Hub.

- Docker Container

Docker Container הוא מופע פעיל (running instance) של Docker Image. כאשר אתה מפעיל תמונה, היא הופכת לקונטיינר.
קונטיינרים הם אפליקציות מבודדות הכוללות את כל מה שנדרש להרצת האפליקציה. ("dedicated virtual machine")



- Docker File פקודות

שם הפקודה	תיאור
ADD	הוסף קבצים ל-Container דרך URL או path הוא בשונה מCOPY הוא יכול לעשות Extract לקבצי ZIP ולהוסיף קבצים מurl חיצוני
ARG	יצירת משתנים שה-User יכול להשתמש בהם בזמן בניית התמונה ב-Docker Build
CMD	ביצוע פקודות ב-cli, בשונה מ-RUN הפקודות מתבצעות בזמן ההרצה של ה-Docker Image
COPY	הוסף קבצים ל-Container דרך Path
ENTRYPOINT	דומה ל-CMD רק שיותר קשה לדרוס את פקודה זו, כיוון שצריך לכתוב את הפרמטר --entrypoint
ENV	יוצר משתנים קבועים גלובליים שכל Layer יכול להשתמש בו, במידה ומתבצע לאחר הגדרתו
EXPOSE	תיאור על איזה Ports האפליקציה פועלת
FROM	שימוש בתמונה שתהווה בסיס לקונטיינר, מעין "ספרית בסיס"
HEALTHCHECK	בודק את תפקוד ה-Container על ידי ביצוע פעולות שהוגדרו על ידי ה-User או שנורשו מתמונת הבסיס, או מעקף של פעולות שנורשו מתמונת הבסיס
LABEL	הוספת דוקומנטציה לקובץ
MAINTAINER	הוספת דוקומנטציה אודות כותב הקובץ
ONBUILD	פקודה שמתבצעת שמשתמשים בתמונה זו כ-Base Image
RUN	הרצת פקודות ב-CLI בזמן בניית התמונה, יוצרת Layer נוסף לאחר הבנייה
SHELL	פקודה שמגדירה את ה-Default Shell
STOPSIGNAL	מציינת את ה-Stop Signal שישלח ל-Process של ה-Container עבור עצירת ה-Container
USER	מציינת את ה-User שה-Container משתמש בו בעת ריצה
VOLUME	מציין את ה-Volume בו האפליקציה תשתמש על מנת שבעת מחיקת ה-Container לא ימחק ה-Data הקשור אליו/שיתוף מידע בין Containers
WORKDIR	הגרת ה-Default Path ב-Container

- Volume

הוא מנגנון לשמירת ולשיתוף מידע בין Containers כך שנתונים ישמרו גם לאחר הפסקתו או מחיקתו.

כאשר אנו שומרים מידע ב-Container לדוגמא, Container שמריץ Data Base.
לאחר שנמחק את ה-Container, אילולא ה-Volume בל ה-Data הקשורה ל-Data Base הייתה נמחקת.

מספר Containers שונים יכולים לגשת לאותו Volume.

- Docker Compose

Docker Compose הוא כלי לניהול אפליקציות מרובות Containers (Multi-Container Applications) על ידי שימוש בקובץ YAML לקביעת התצורה של ה-Services, הרשתות וה-Volumes הנדרשים לאפליקציה.

- Docker Staging

Multi-Stage Build הוא שימוש ביותר מפקודת FROM אחת כאשר אנו כותבים את ה-Docker-File שלנו, כל FROM חדש מסמל התחלה של Stage חדש.
כל Stage מייצר Docker-Image משלו, אבל רק ה-Stage האחרון אפשרי לשימוש לוקאלי לאחר שה-Build מסתיים.
היתרון של Multi-Stage Build הוא שניתן לשתף מידע בין כל Stage ובכך התמונות שנוצרו הם קטנות יותר, המידע שמועבר קטן יותר דבר שמגדיל את הביצועים, וה-Container יותר מאובטח. הוא עוזר לנו ליצור הפרדה ולהפוך את הקוד שלנו לקריא וברור יותר.

- Docker Networking

המושג Docker Networking מתייחס אל היכולות של Containers לתקשר אחד עם השני או לחלופין, תקשורת של Containers עם אובייקטים שהם אינם של Docker.
ל-Containers יש Networking שפועל כברירת מחדל ובעזרתם הם יכולים ליצור קשרים ולהעביר מידע בין אובייקטים, ה-Containers עצמם אינם יודעים באיזה רשת הם נמצאים או האם האפליקציות שמחוברות אליהם של Docker או לא. Container רק רואה Network Interface עם כתובת IP, Gateway, Routing Table, DNS Service...

User-Defined Networks: רשתות שמוגדרות על ידי המשתמש, על ידי שיטה זו אתה יכול לחבר כמה Containers לאותה הרשת, וכאשר ה-Containers מחוברים לאותה הרשת, הם מתקשרים ביניהם באמצעות כתובת ה-IP שלהם או שם הContainer.

Network Drivers Overview: הרשת של Docker ניתנת לחיבור באמצעות מנהלי התקנים, ישנם כמה מנהלי התקנים.

1. רשתות Bridge הם ברירת המחדל של הרשת, רשתות Bridge הם בשימוש נפוץ כאשר נרצה להריץ כמה Containers שיתקשרו אחד עם השני על אותו ה-host.
2. רשתות Host מבטלים את הבידוד בין ה-Host ל-Container ובכך מאפשרים להשתמש ברשת של ה-Host ישירות.
3. רשתות Overlay מחברים כמה Docker Daemon (תהליך רקע של Docker שמנהל את ה-Images ה-Containers, הרשתות, וה-Volumes) יחדיו ומאפשרים ל-Swarm Services/Containers (כמה Docker Daemon) לתקשר באמצעות Nodes (העברת נתונים בין פודים שמתארחים בצמתים שונים בתוך אשכול Kubernetes).
4. רשתות Ipvlan נותנות למשתמש שליטה מלאה בכתובות IPv4 ו-IPv6 לניתוב.
5. רשתות Macvlan מספקות ל-Mac Address Container וגורמות לו להופיע בתור מכשיר פיזי על הרשת, ה-Docker Daemon מנווט את תעבורת הרשת על פי כתובות ה-Mac של ה-Containers ושימוש ב-Macvlan זוהי הבחירה הטובה ביותר כאשר נרצה שאפליקציות יהיו מחוברות ישירות לרשת פיזית.
6. None מבודד את ה-Container מה-Host ומה-Containers האחרים.

Container Networks: ניתן להתחבר ישירות בין Containers על ידי הפקודה <--network container:<name|id

Published ports: כאשר אנחנו מריצים Containers ה-Docker Bridge איננו מייצא את הפורט ל-Host ולכן נשתמש בפקודה p-- על מנת להפוך את ה Port -Services גלויים אל מחוץ לרשת ה-Bridge.

IP address and hostname: כברירת מחדל Containers מקבלים את הכתובת ה-IP של רשת ה-Docker שאליה הם מקושרים, ה-Docker daemon יוצרת רשת משנה ומקצה כתובות IP עבור קונטיינרים. לכל רשת יש גם מסכת רשת משנה ושער ברירת מחדל.
ניתן לחבר Container לכמה רשתות על ידי העברת ה-network Flag-- כמה פעמים או docker network connect במידה וה-Container רץ בשתי המקרים יש לציין את ה-ip Flag-- או ip6-- על מנת לציין את כתובת ה-IP של ה-Container ברשת.

הצעות לייעול ה-Docker File -

ישנם כמה כללים שצריך לעבור עליהם על מנת לייעל את ה-Docker-File והם:

1. להימנע מהתקנת חבילות שאינם נחוצות, כאשר אנחנו מתקינים חבילות שאינם נחוצות זה מעלה את ה-Build Time ואת הגודל של התמונה, בנוסף כל פעם שתבצע שינויים ב-Docker File תצטרך לבנות את התמונה הגדולה שוב.
אפשר להשתמש בקובץ Requirements על מנת להתקין את כל החבילות הנחוצות באמצעות הרצת הפקודה, RUN pip3 install -r requirements.txt
2. שרשור פקודות RUN, כל פקודות ה-RUN יוצרות Cacheable Unit שבונות שכבה חדשה מהתמונה כל פעם את זאת אפשר למנוע על ידי שרשור פקודות ה-RUN בפקודה אחת כך על ידי הסימן && \ בין כל פקודה (יש לשרשר בגבול הטעם הטוב ולא להגזים עם השרשור)
3. שימוש בקובץ dockerignore., שימוש בקובץ לא יוסיף קבצים לא נחוצים ל-Container ובכך יקטינו את גודל התמונה ויעלו את הביצועים של ה-Build.
4. סידור נכון של ה-Statements, יש לשים את ה-Statements בשימוש הכי נפוץ בסוף ה-Docker File זאת משום שברגע ש-Cache "נשבר" אז כל הקבצים שאחריו צריכים הפעלה מחדש, ולכן יש לשים פקודות RUN מעל פקודות COPY, ופקודות כגון CMD/ENTRYPOINT בסופו של ה-Docker File.
5. להימנע מהתקנת Dependencies לא נחוצים, ניתן לעשות זאת באמצעות ה-Flag no-install-recommends--.
6. שימוש ב-Base Images מינימליים, זאת באמצעות התקנת חבילות כגון alpine.