שם הפרויקט: User profile

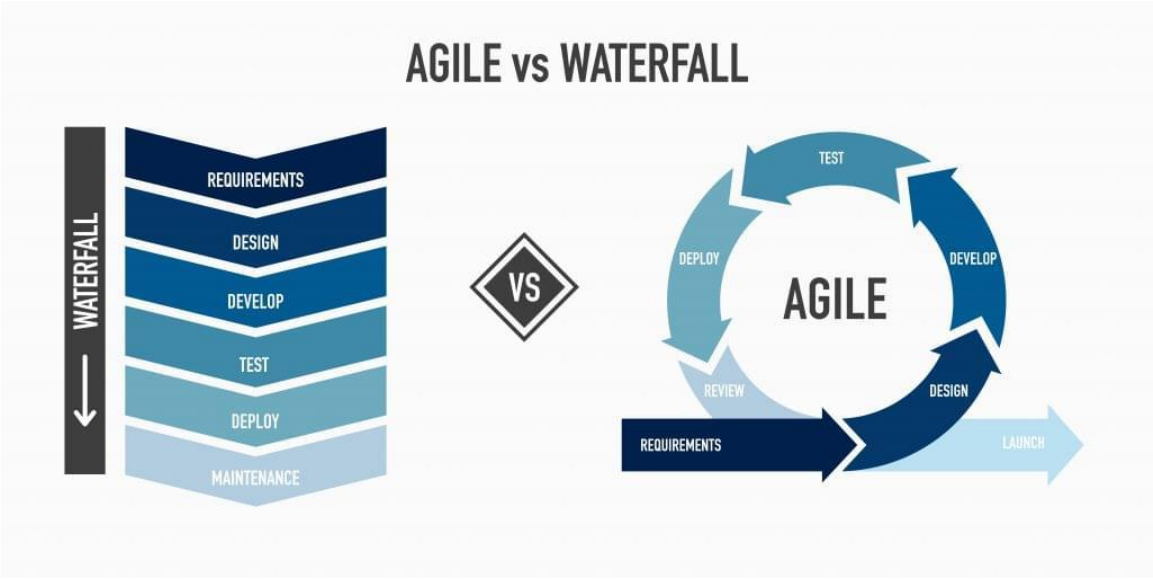
רקע:

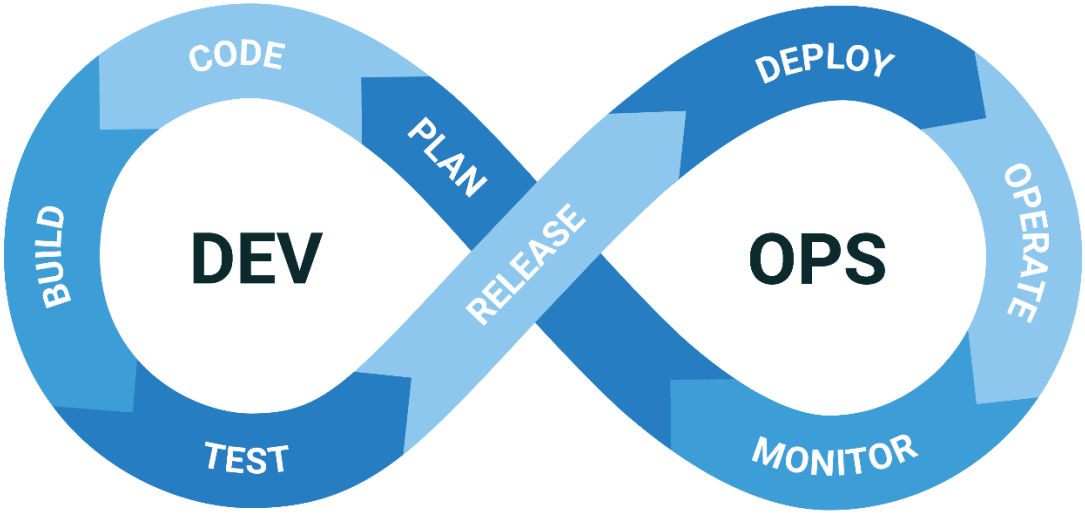
מהו דבאופס:

דְּבָאוֹפְּס (DevOps) הוא שיטת עבודה בעולם ההייטק שמשלבת בין צוותי הפיתוח (Development) וצוותי התפעול (Operations) בארגון בצורה יעילה וחלקה. המטרה העיקרית של DevOps היא לשפר את תהליכי השותפות בין הצוותים ולקצר את זמני השחרור של יישומים ושירותים לשוק. דבאופס דוגל באוטומציה של תהליכי אספקת תוכנה ושל שינויים בתשתיות, ברגילות תהליכים, ובפתרון מהיר ויעיל של בעיות.
DevOps מקדם שיתוף פעולה ותקשורת טובה בין הצוותים, ומעודד פיתוח באיכות גבוהה והפעלה של מוצרים באמצעות תהליכי CI/CD .

היסטוריה:

עד שנות ה90 שלטו בעולמות הניהול והפרויקטים מתודולוגיות פיתוח מאוד מוסדרות, מתוכננות יתר על המידה ומנוהלות ברמת המיקרו. גישה ניהולית זו נקראת גם שיטת מפל המים (Waterfall), מכיוון שכל שלב התחיל רק לאחר שקודמו הסתיים – אפיון דרישות, בניית תוכן, פיתוח קוד, בדיקות. אך במציאות של היום, ישנו המון אי-וודאות בהובלה ופיתוח של פרויקטים: אירועים בלתי מתוכננים, לקוחות תובעניים, התאמות, שינויים וסביבה דינמית ומשתנה. מתוך הכרח זה נולדה הגישה האג'ילית.
אג'ייל (Agile = זריז) או לעיתים גם נקרא בעברית "פיתוח תוכנה זריז", היא גישה מודרנית לניהול והובלת פרויקטים אשר מסייעת לצוותים וארגונים לספק תוצרים ללקוחות שלהם בצורה יעילה, מהירה ומבלי המון כאב ראש. אותו "תוצר" יכול להיות פיצ'רים חדשים במוצר, תוכנה, אפליקציה, שירות או כל דבר אחר שהחברה מספקת.
ניהול אג'ילי טומן בתוכו תפיסה איטרטיבית (Iteration) - במקום לעבוד ב"צ'אנקים" גדולים שלעיתים עשויים לקחת חודשים או שנים של עבודה, הגישה האג'ילית תומכת בעבודה בפרקים קטנים ותכופים (כמה שבועות בודדים) כאשר בסופם יש תוצר כלשהו שאפשר להעביר ללקוח לשימוש וקבלת פידבק כמה שיותר מהר.
בארגונים אשר אימצו את גישת פיתוח תוכנה זריז (Agile) חלה עלייה בכמות גרסאות התוכנה היוצאות לאור. במקור, שיטת ה-DevOps נוצרה מהעלייה בפופולריות של פיתוח זריז.
שיטת הדבאופס משתלבת נהדר עם הגישה האג'ילית ומיישמת בפועל גישה זריזה לפיתוח תוכנה ע"י תהליך פיתוח שחוזר על עצמו ומאפשר שחרור גרסאות חדשות בצורה חלקה ומהירה באמצעות תהליכי אוטומציה.
שיטות ה-Agile ו-DevOps דומות אך שונות במספר היבטים חשובים: Agile מהווה שינוי בצורת החשיבה, בעוד ש-DevOps מיישמת שינוי בתרבות הארגונית הלכה למעשה. אחת המטרות של DevOps היא לייסד סביבה בה יכולים להתקיים הוצאות לאור של גרסאות תוכנה אמינות יותר ובקצב גבוה יותר.


AGILE vs WATERFALL
WATERFALL
REQUIREMENTS
DESIGN
DEVELOP
TEST
DEPLOY
MAINTENANCE
VS
AGILE
TEST
DEVELOP
DESIGN
LAUNCH
REVIEW
DEPLOY
REQUIREMENTS


DEV
OPS
CODE
PLAN
BUILD
TEST
RELEASE
DEPLOY
OPERATE
MONITOR

מטרת הפרויקט:

מטרת הפרויקט היא ישום של תהליך ה Devops שהוא תכנון וכתיבת אפליקציה ,העלאת תשתיות באופן אוטומטי, ולבסוף בניית פייפליין CI/CD על מנת לאפשר שחרור גרסאות מהיר ואוטומטי.

בפרויקט זה בחרנו להדגים את התהליך בעזרת אפליקציית Nodejs ,שימוש בשפת Html לפרונט-אנד, ושימוש ב MongoDB לשכבת הנתונים .

מפני שזהו פרויקט Devops עיקר הפרויקט אינו באפליקציה, אלא בתהליך השחרור של האפליקציה וגרסאות חדשות.

הבעיות שאותן דבאופס מגיע כדי לפתור:

- חוסר שיתוף פעולה ותקשורת בין צוותי פיתוח ותפעול.
- שחרור גרסאות חדשות צריך להתבצע בזריזות וביעילות רבה יותר ע"מ לענות על צרכי השוק.
- יש צורך בתהליך אוטומטי על מנת לייעל את העבודה ולהגביר את הקלות שבפיתוח פיצ'רים חדשים.
- שינוי התשתיות שעליהם יושבת האפליקציה יכול להיות הרסני לכן יש צורך לעקוב אחר השינויים ולנהל את ההרשאות לבצע אותם.
- לחברות כיום אין את המשאבים או הרצון להשתמש בשרתים שלהם בלבד כתשתית לאפליקציה, במיוחד חברות סטארט-אפ צעירות שלא מעוניינות להתעסק בזה.

דרישות מערכת:

- גישה לאינטרנט
- חשבון פעיל באחד משירותי הענן, בפרויקט נשתמש בשירותי הענן של חברת Amazon (AWS)
- מכונה וירטואלית עם מערכת הפעלה לינוקס (נשתמש ב Ubuntu)
- חשבון ב GitHub
- VScode
- Git
- Docker

- Docker Compose
- Terraform

בעיות צפויות במהלך הפיתוח:

- ככל שצומחת הפופולריות של האפליקציה, התשתית התומכת בה צריכה להיות מסוגלת להרחיב את גודלה כדי לעמוד בביקוש.
- כל עדכון או שינוי קבצי מקור (source) של האפליקציה יכול לייצר באגים.
- נפילת קונטיינרים יכולה לגרום לנפילת השירות כולו ואף לאיבוד מידע חיוני.
- ביצוע שילוב מתמשך של הקוד ופריסה מתמשכת של האפליקציה (צינור CI/CD) הוא מורכב ודורש שילוב של מגוון כלים וטכנולוגיות.
- התשתית של האפליקציה היא מורכבת ויש צורך בטכנולוגיה שתאפשר להעלות סביבות מורכבות בקלות.

פתרונות:

- שימוש בתשתיות ענן התומכות בהרחבת הגודל ע"פ צורך.
- הרצת בדיקות תוכנה על כל שינוי בקוד, באופן אוטומטית וכך וידוא שהקוד תקין.
- שימוש בטכנולוגיית קוברנטיס לניהול ופריסה אוטומטית של קונטיינרים.
- שימוש ב GitHub Actions – טכנולוגיית CI/CD מתקדמת שמאפשרת שימוש חוזר בפעולות נפוצות בתהליך בקלות.
- שימוש ב Terraform (כלי תשתית כקוד) שמאפשר הקמת תשתיות מורכבות, ע"י קבצי קונפיגורציה פשוטים יחסית.

טכנולוגיות בשימוש:

דוקר קונטיינר:

קונטיינר Docker הוא פורמט אריזה שאורז את כל הקוד והתלות של אפליקציה בפורמט סטנדרטי המאפשר לה לפעול במהירות ובאמינות על פני סביבות מחשוב. קונטיינר Docker הוא מיכל קל משקל, עצמאי, בר הפעלה, הכולל את כל הדרוש להפעלת אפליקציה, כולל ספריות, כלי מערכת, קוד וזמן ריצה. Docker היא גם פלטפורמת תוכנה המאפשרת למפתחים לבנות, לבדוק ולפרוס יישומים מכוללים במהירות.

Docker היא מסגרת פיתוח אפליקציות פתוחה שנועדה להועיל ל-DevOps ולמפתחים. באמצעות Docker, מפתחים יכולים בקלות לבנות, לארוז, לשלוח ולהפעיל יישומים כמכולות קלות משקל, ניידות ועצמאיות, שיכולות לפעול כמעט בכל מקום. קונטיינרים מאפשרים למפתחים לארוז אפליקציה עם כל התלות שלה ולפרוס אותה כיחידה אחת. על ידי אספקת מיכלי יישומים מובנים מראש ומקיימים את עצמם, מפתחים יכולים להתמקד בקוד האפליקציה ולשימוש מבלי לדאוג לגבי מערכת ההפעלה הבסיסית או מערכת הפריסה.

Docker hub:

מקום אחסון לאימג'ים שהם בעצם תמונות מצב של קונטיינר הכוללים את כל המידע הדרוש להפעלת אפליקציה או שירות ספציפיים. ניתן לאחסן במאגר פרטי או ציבורי.

גיט:

גיט (Git) היא מערכת ניהול גרסאות מבוססת תוכנה. השימוש בה מאפשר למתכנתים שונים ואף רבים לעבוד בצוות על פרויקט משותף, לצפות בשינויים שבו ולנהל את הגרסאות שלו באופן חכם.

יתרונה הגדול הוא באחסון, מעקב, ברירה וגיבוי של גרסאות, בדרך הארוכה לפיתוח גרסת תוכנה שלמה וסגורה. על אף שהיא משמשת כיום בעיקר אנשי תכנות, גיט יודעת לנהל לא רק קבצי תוכנה, אלא גרסאות של כל סוג אפשרי של קבצים ומסמכים.

לינוקס:

Linux הנה מערכת הפעלה בעלת קוד פתוח אשר אין בה הגבלות וכל תכניה פתוחים לכלל משתמשיה. המערכת נוצרה כתחליף למערכת היוניקס היקרה בתחילת שנות ה-90 והיא מופצת בחינם לכל מי שמעוניין בה, בניגוד ל- Windows של Microsoft, אשר על מנת להשתמש בה ניאלץ לוותר על כמה מאות שקלים ורוב תכניה אינם ניתנים לשינוי.

סביבת ההפעלה לינוקס הפכה לאלטרנטיבה מובילה למי שחיפש פלטפורמה אמינה ללא תשלום. לינוקס היא מערכת הפעלה בקוד פתוח. כלומר, המקור זמין לכולם, לא צריך לשלם עליה ויתרה מכך, ניתן לשנות ולהתאים אותה לצרכים.
היא מאוד גמישה ויכולה לשרת מערכות שונות, שרתים, יישומי שולחן עבודה. מערכת הפעלה זו תומכת בכל שפות התכנות הנפוצות כמו C/C++, Java, Python, Ruby ועוד.

מחשוב ענן:

טכנולוגיית ענן (או מחשוב) הוא כוח מחשוב שהלקוח קונה כגון שירותים, מסדי נתונים, יישומים וכו' באינטרנט תמורת תשלום. חברות ועסקים משתמשים במחשוב ענן במקום שיהיה להם תשתית מחשוב או מרכזי נתונים (שרתים) משלהם, חברות יכולות לשכור גישה לכל דבר, החל מאפליקציות ועד אחסון מספק שירותי ענן.אחד היתרונות של שימוש בשירותי מחשוב ענן הוא שחברות יכולות להימנע מעלויות וקשיים הקשורים בבעלות ותחזוקה של תשתית IT משלהם, ובמקום זאת פשוט לשלם עבור השימוש לחברה חיצונית. שירותי מחשוב ענן מכסים כיום מגוון רחב של יכולות, החל מאחסון נתונים, כוח מחשוב ועד עיבוד מידע בשפה טבעית ובינה מלאכותית, כמו גם ליישומי משרד סטנדרטיים. כמעט כל שירות שאינו דורש קרבה פיזית לציוד המחשבים ניתן לספק כעת דרך הענן.

קוברנטיס:

קוברנטיס היא מערכת קוד פתוח לניהול ופריסה אוטומטית של יישומים על גבי קונטיינרים. יחידת ההרצה הבסיסית היא Pod, שמשמשת כשכבת הפשטה של קונטיינר או קבוצת קונטיינרים, שנמצאים על אותו השרת ויכולים לחלוק משאבים. כל פוד מקבל כתובת IP ייחודית, והודות לכך יכולים יישומים שונים להשתמש בצורה חופשית בפורטים בלי חשש להתנגשויות ביניהם. בפרויקט נשתמש ב EKS - שירותי הקוברנטיס של אמזון.

גיטהאב:

הפלטפורמה של GitHub מבוססת על שירות ענן (SaaS) ומאפשרת לנהל תהליכי פיתוח, לעקוב אחר שינויים בפרויקט בקוד, לאחסן סקריפטים, ולהעלות תוכן טקסטואלי. המטרה העיקרית של גיטהאב היא לספק עבודת צוות, שלמות של המידע, מהירות בתהליכי פיתוח ותמיכה בתהליכים מבוזרים.

גיטהאב מבוסס על מערכת גיט. מערכת גיט מספקת בבסיסה יכולות מגוונות כמו ניהל גרסאות, ניהול שינויים, תיוג גרסאות ויכולות הנוספות שבעיקרן מבוססות על ממשקים שונים. גיטהאב היא פלטפורמה שמספקת מאגר העוקבת אחר השינויים בקובצי קוד של אותו פרויקט, וכל שינוי המבוצע בקובץ או בקוד מעודכן במאגר. השינוי שחל על אותו קובץ מכיל בתוכו תיאור עם אותו שינוי, מי ביצע את השינוי ומי אחראי על תהליך השינוי. במצב כזה מבוצעת השליחה של השינויים (נקרא Commit).

Terrafrom:

תוכנת קוד פתוח המאפשרת למשתמשים להגדיר תשתית באמצעות קובץ הגדרות קריא, בשפת תצורה הצהרתית המכונה HCL. זהו כלי Infrastructure as Code שמאפשר להרים סביבות שלמות בפקודה אחת: Terraform apply. התוכנה פועלת ע"י יצירת קובץ state בו היא שומרת את המידע האחרון הידוע על הסביבה ובכל ביצוע של terraform apply, בודקת התוכנה אם יש הבדל בין המצב הרצוי שמתואר בקבצי הקונפיגורציה לבין ה state (המצוי) אם יש הבדל, התוכנה יודעת להרים ולמחוק את התשתיות בהתאם.

:Mongodb

מסד נתונים מונחה מסמכים. כלומר, לא משתמש בטבלאות ובשורות כדי לאחסן נתונים, במקום זאת אוספים של מסמכים דמוי JSON . מסמכים אלה תומכים בשדות משובצים, כך שניתן לאחסן נתונים קשורים בתוכם.

:ArgoCd

כלי פריסה מתמשכת חדש יחסית (Continues Deployment) במיוחד לאפליקציות שבנויות על גבי קוברנטיס. עובד בצורה כזאת כך שהוא יודע לעקוב אחר מאגר קבצי קונפיגורציה של רכיבי קוברנטיס, ולפרוס את האפליקציה שנובעת מהם. לאחר מכן הוא ממשיך לעקוב אחר שינויים בקובצי הקוברנטיס ומיישם כל שינוי בפועל ע"י ביצוע הפקודה kubectl apply לכל שינוי.

Prometheus & Grafana:

Prometheus הוא כלי מוניטורינג שמאפשר איסוף מטריקות שונות על מערכות. לדוגמא אחוז ניצול הדיסק, אחוז ניצול ה cpu, ועוד.

Grafana מאפשרת הצגת המטריקות בדשבורדים נוחים. באמצעות Grafana נוכל לבנות תצוגה נוחה עם יכולת פילטור על מגוון מטריקות שנבחר. היא מספקת יכולות ויזואליזציה שונות בשביל שנוכל לבנות dashboard אינפורמטיבי ומועיל.

Grafana Prometheus

שפות בשימוש בפרוייקט:

Node.js- ספרייה של ג'אווה סקריפט (Java Script) ושפה מונחית עצמים עליה בנויים צד השרת וצד הלקוח.

Yaml- שפת קונפיגרציה שהינה תרגום של ג'ייסון (Json) לשפת אדם, משמשת לעבודה עם קוברנטיס ו GitHub Actions.

HCL- שפת הקונפיגורציה של Terraform.

Bash- שפת פקודות וסקריפטים בלינוקס.

היכן מאוחסנים הקבצים:

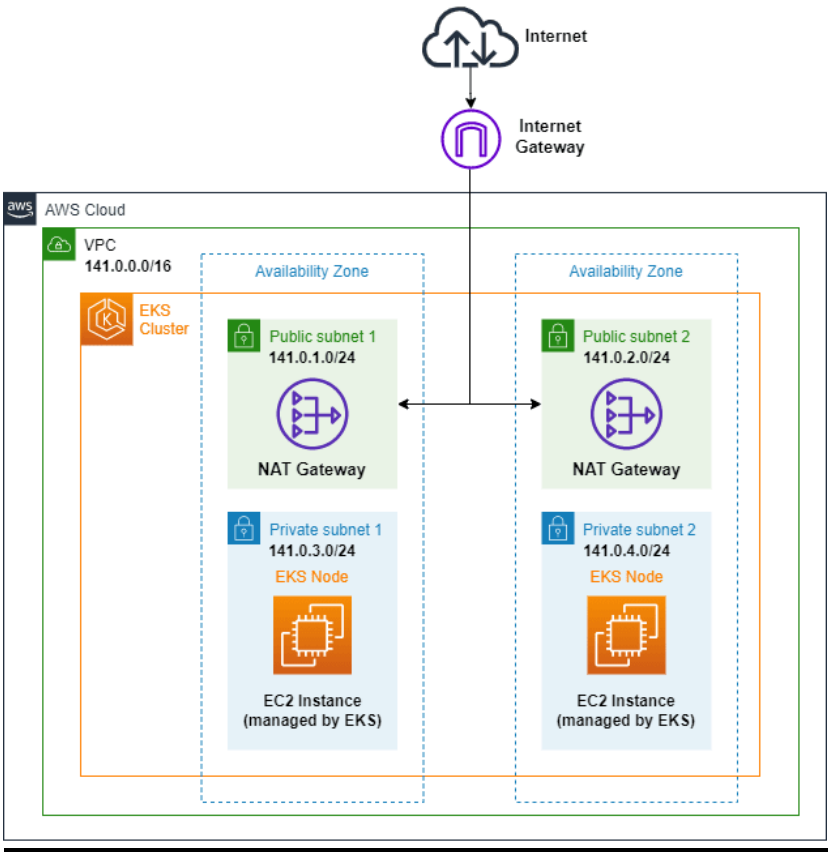
קבצי המקור מאוחסנים במאגר בגיטהאב. במאגר נוסף מאוחסנים קבצי הקונפיגורציה של טראפורם וקוברנטיס.

תיאור הפרויקט:

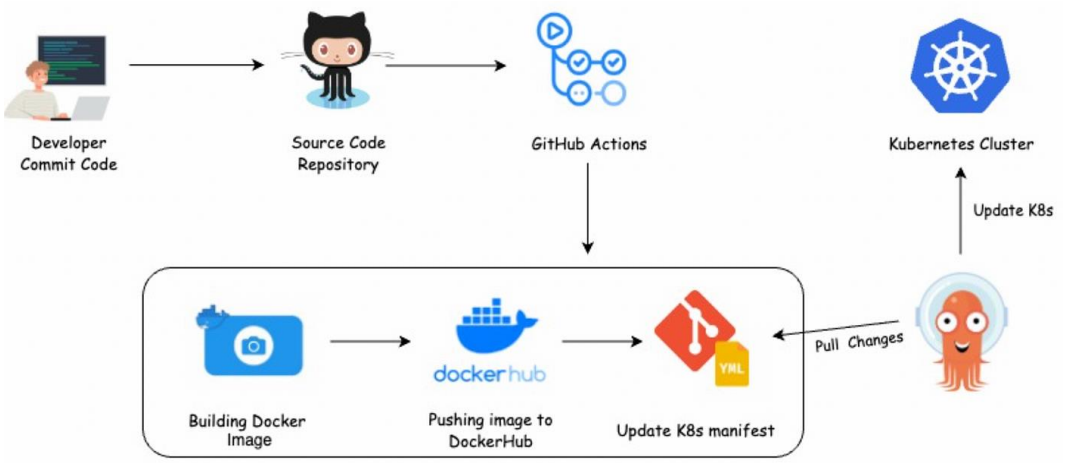
האפליקציה- אפליקציה פשוטה הכתובה ב Node.js, לפרופיל משתמש באתר בו המשתמש יכול לשים את הפרטים עליו: שם, כתובת מייל ותחביב. הפרטים נשמרים ב Database (Mongo DB) ומוחזרים למשתמש.

הפרויקט- פרויקט דבאופס הפועל במקביל בשני מסלולים. המסלול הראשון והבסיסי הוא התשתית לכל הפרויקט. התשתית יושבת על שירותי הענן של אמזון, בתוך סביבה וירטואלית פרטית (vpc) שעל גביה יושב EKS (Elastic Kubernetes Service) שמהווה קלאסטר מנוהל של קוברנטיס.

הקלאסטר בנוי ממאסטר ומכונות פועלות. במבנה הנוכחי שתי מכונות וירטואליות יושבות באזור זמינות נפרד ברשת משנה פרטית (ללא גישה ישירה למשתמשים) כדי לייצר שרידות, זמינות ואבטחה גבוהים יותר. הפודים שבתוכם הקונטיינרים שמהוויים את האפליקציה ימצאו על המכונות הפועלות בשתי איזורי הזמינות. האפליקציה בנויה מקונטיינר אחד של mongo וקונטיינר אחד של Nodejs שמהווה גם את צד השרת וגם את צד הלקוח. הגישה לאינטרנט של האפליקציה היא דרך שתי רשתות משנה ציבוריות שבתוכם נמצא NAT Gateway שמהווה חיבור לאינטרנט עבור האפליקציה וכך יש גישה לרשת הכללית למרות שהאפליקציה עצמה יושבת על רשת פרטית.

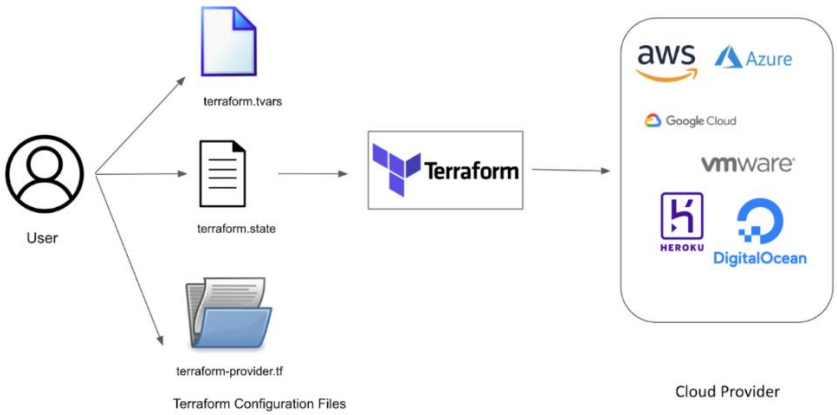


המסלול השני הוא תהליך ה CI/CD שמבוצע ע"י GitHub Actions. קובץ ה Yaml של GitHub Actions נקרא Workflow. הקובץ יושב באותו מאגר של קבצי באפליקציה בגיטהאב, ומפעיל תהליך CI/CD שלם ומקיף על כל שינוי בקוד של האפליקציה. התהליך כולל בנית אימג' חדש של האפליקציה עם השינויים בקוד, הרצת בדיקות, ודחיפה לדוקר האב עם גרסה חדשה. לאחר מכן שינוי האימג' שעליו יושבים הקונטיינרים של האפליקציה בקבצי הקונפיגורציה של קוברנטיס, ובנוסף הדגמה של הפצה ל staging על גבי minikube ובדיקה של האפליקציה בסביבה דומה לסביבת הפרודקשן (AWS).



תשתית ו Terraform:

כל התשתית מורמת ע"י טראפורם. התשתית כוללת VPC, EKS קלאסטר שעליו מותקן האפליקציה ובנוסף גם Argocd וPrometheus stack . לאחר שיצרנו את קבצי הקונפיגורציה בהם הגדרנו לטראפורם מהי התשתית הנדרשת נבצע את הפקודה terraform plan. לאחר פקודה זו יוצג לנו מהי התוכנית ליצירת התשתית ואילו משאבים יווצרו. לאחר שאנחנו מרוצים מהתשתית נבצע את הפקודה terraform apply שתרים את כל התשתית הנדרשת. לאחר הרמת התשתית טראפורם שומר את המצב של התשתית ב state file שנמצא על סל אחסון פרטי ב AWS. וזאת לצרכי אבטחה שהרי הקובץ מכיל מידע רגיש.



:CI/CD

תהליך ה CI מבוצע על ידי שימוש ב גיטהאב אקשיינס. יש לשימוש בטכנולוגיה הזאת מספר יתרונות שגרמו לנו לבחור לעבוד איתה:

- Runners שהן המכונות שמבצעות בפועל את כל תהליך ה CI מנוהלות על ידי GitHub ובכך חוסכות את המאמץ בניהול ופריסה שלהן. ניתן פשוט להגדיר מה סוג מערכת ההפעלה הנדרשת ומאחורי הקלעים עולה קונטיינר עם אותה מערכת הפעלה.
- GitHub Actions מציעה ממשק פשוט ונוח לשימוש בפעולות נפוצות (Actions) מתוך Marketplace מרכזי וכך ניתן לבצע פעולות מורכבות יחסית כמו דחיפה לדוקר האב או פריסה על גבי Minikube בקלות.
- Workflow - הקובץ שמכיל את תהליך ה CI יושב באותו מאגר עם ה Source code וכך התגובה על כל שינוי בקוד היא מיידית ופשוטה ללא צורך בהגדרות מיוחדות.

תהליך ה CD מבוצע על ידי שימוש ב ArgoCd. הטריגר ל Argo הוא שינוי בקבצי ב Yaml של האפליקציה וכך בעצם Argo שעוקב אחר אותו מאגר קבצים מזהה כל שינוי ומבצע את השינוי שהוא בעצם Image חדש של האפליקציה שעל בסיסו בנוי הקונטיינר שמריץ אותה. בנוסף Argo מציע ממשק נוח לצפייה בכל מרכיבי האפליקציה.