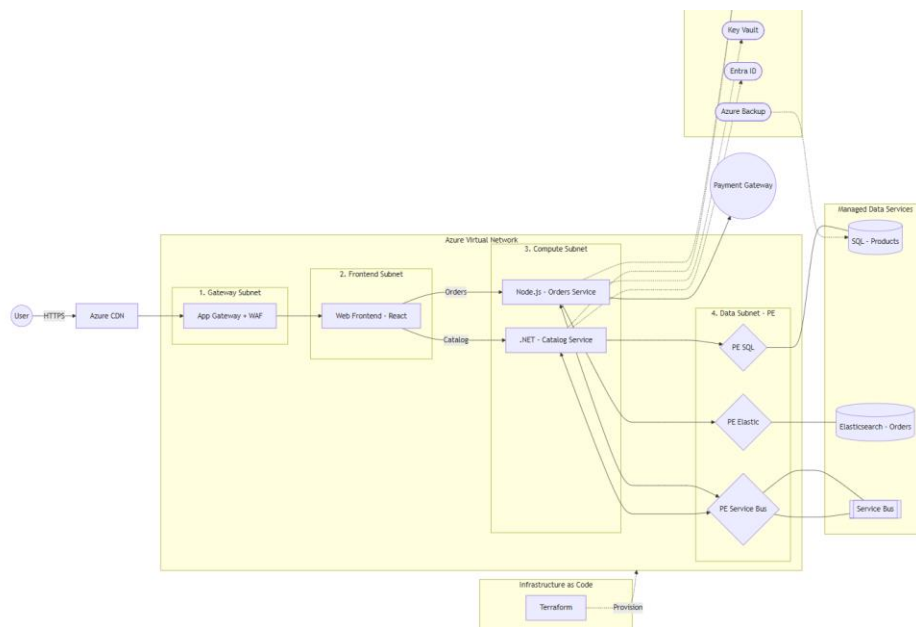


## סקירה כללית:

הקמת תשתית ענן מודרנית מבוססת שירותים מנוהלים (PaaS) ב-Azure, המשלבת הפרדת רשתות מלאה (Segmentation), אבטחה בגישת **Zero Trust** וארכיטקטורת **Microservices** מבוצרת. המערכת מפרידה בין ניהול הקטלוג לניהול ההזמנות כדי להבטיח זמינות גבוהה, ביצועים אופטימליים וחסינות נתונים.

## תרשים ארכיטקטורה (Diagram): במסמך המצורף



## 2. פירוט שכבות הארכיטקטורה

### א. שכבת הכניסה והגנת Edge

- **Azure Application Gateway v2 (WAF Enabled)**: משמש כ-Reverse Proxy וניהול תעבורה בשכבה 7. כולל **WAF Policy** במצב Prevention להגנה מפני תקיפות וסריקת SSL.
- **Azure CDN**: הגשת תכנים סטטיים (תמונות מוצרים) מנקודות קצה קרובות לשיפור מהירות הטעינה ב-50% והעלאת יחס ההמרה.

## ב. שכבת המחשוב (Compute) - Microservices

- **Orders Service (Node.js)**: שירות מנוהל האחראי על תהליך ההזמנות, הסליקה והחיפוש. נבחר בשל יעילות בטיפול ב-I/O אינטנסיבי.
- **Catalog Service (NET.)**: שירות ליבה האחראי על ניהול המוצרים, הקטגוריות והמלאי. נבחר בשל יציבותו ועבודה אופטימלית מול SQL.
- **Scaling & Availability**: שימוש ב-Autoscale Rules מבוססי CPU/RAM ופריסה ב-Deployment Slots (שיטת Blue/Green) למניעת Downtime בעדכוני גרסה.

## ג. שכבת הנתונים והאחסון (Data & Storage)

- **SQL Server Database**: ה-Source of Truth למידע קטלוגי ופיננסי. מוגן באמצעות Always Encrypted ו-TDE.
- **Azure AI Search (Elasticsearch)**: מנוע אינדוקס המאפשר חיפוש "Fuzzy" מהיר ושליפה אגרגטיבית של היסטוריית הזמנות.
- **Azure Cache for Redis**: ניהול Session (עגלת קניות) ושמירת קטלוג בזיכרון למניעת עומס על בסיס הנתונים.

## ד. אופרציה ותשתית כקוד (IaC & DevOps)

- **Terraform**: כלל המשאבים (Data, Compute, Networking) מוגדרים ומנוהלים כקוד. Terraform מבטיח מניעת **Environment Drift**, ניהול גרסאות לתשתית ויכולת **Disaster Recovery** מלאה על ידי הקמת הסביבה מחדש תוך דקות.
- **CI/CD Pipeline**: אוטומציה מלאה של פריסת הקוד והתשתית מ-GitHub ל-Azure ללא מגע יד אדם.
- **Observability**: ניטור מקצה לקצה באמצעות **Application Insights** (לביצועי אפליקציה) ו-**Log Analytics** (ללוגים תשתיתיים).

### 3. אסטרטגיית אבטחה (Zero Trust)

1. **Micro-Segmentation**: חלוקת ה-VNet לסאבנטים מבודדים עם חוקי NSG קשיחים.
2. **Private Link**: חסימת גישה ציבורית לכל שירותי ה-SQL, PaaS, Bus, Redis, Search – גישה פנימית בלבד דרך **Private Endpoints**.
3. **Managed Identities**: חיבורים ללא סיסמה (Secretless) בין השרתים לשירותי הנתונים.
4. **Azure Key Vault**: ניהול מרכזי של סודות ומפתחות הצפנה בגישת RBAC.

### 4. ניתוח סיכונים וחלופות (Design Rationales)

- **למה PaaS ולא VM?** הצורך ב-Scaling מיידי (למשל ב"בלאק פריידי") מחייב שירותים שיכולים לגדול תוך שניות, יכולת שאינה קיימת ב-VMs בצורה יעילה.
- **למה Terraform?** בגלל מורכבות הרשת (Private Links) והצורך בעקביות מוחלטת בין סביבת פיתוח לייצור.
- **למה Service Bus (Premium)?** כדי לאפשר **Decoupling** בין שירותי ההזמנות לקטלוג, והבטחת עמידה בתקני אבטחה של רשת פרטית.

### 3. נימוקים לבחירת הטכנולוגיות (Design Rationales)

שימוש בפרויקט	טכנולוגיה	למה דווקא זו?
ניהול הזמנות	Node.js	מצטיין ב-Asynchronous I/O ובטיפול בהרבה בקשות קטנות ומהירות מול ספקי סליקה ו-Elastic.

ניהול קטלוג	<b>.NET Core</b>	שפה חזקה (Strongly Typed) המתאימה ללוגיקה עסקית מורכבת ומציעה ביצועים מעולים מול SQL Server.
מוצרים וקטגוריות	<b>SQL Server</b>	הבטחת דיוק מוחלט (ACID) במחירים ומלאים, ויכולת ביצוע שאילתות רלציוניות מורכבות.
חיפוש והזמנות	<b>Elasticsearch</b>	מאפשר חיפוש טקסטואלי חכם ושליפה מהירה של היסטוריית הזמנות מתוך נפחי דאטה גדולים.
תקשורת בין שירותים	<b>Service Bus</b>	משמש כ"בולמי זעזועים"; מבטיח שהזמנות לא ילכו לאיבוד גם בעומס חריג ומנתק צימוד בין השירותים.
תשתיות (IaC))	<b>Terraform</b>	מאפשר הקמה ושחזור של כל הסביבה מקוד, מונע טעויות אנוש ומבטיח זהות בין סביבות.