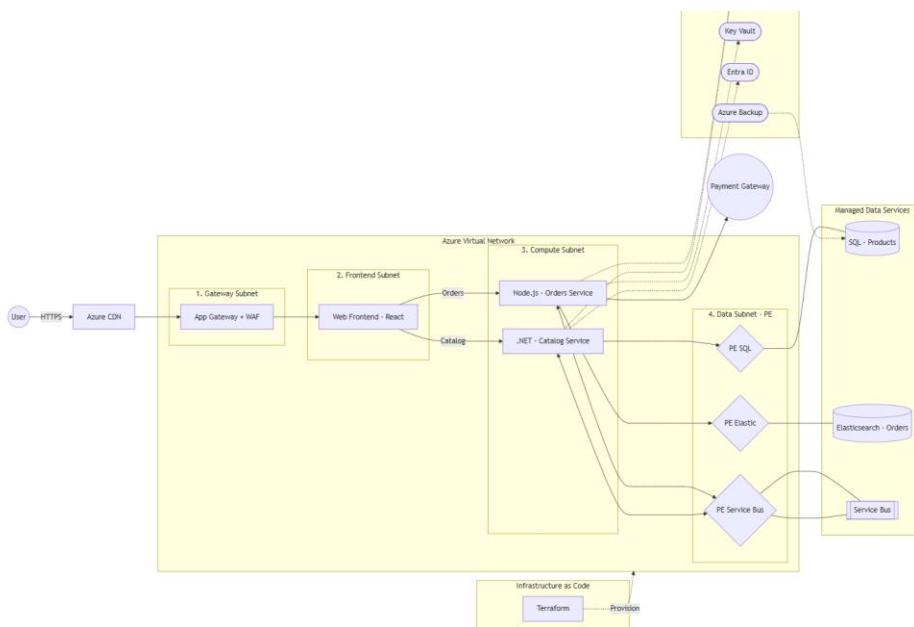


סקירה כללית:

הקמת תשתיית ענן מודרנית מבוססת שירותים מנהליים (PaaS) ב-Azure, המשלבת הפרדת רשות מלאה (Segmentation), אבטחה בגין Segmentation Zero Trust וארQUITטורת Microservices מבוצרת. המערכת מפרידה בין ניהול הקטלוג לניהול הזמן כדי להבטיח זמינות גבוהה, ביצועים אופטימליים וחסינות נתוניים.

תרשים ארכיטקטורה (Diagram): בסיסם המצויר



2. פירוט שכבות הארכיטקטורה

א. שכבת הכניסה והגנת Edge

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

ב. שכבת המחשב (Compute) - Microservices

- **Orders Service (Node.js)**: שירות מנהל האחראי על תהליכי הזמנות, הסליקה והחיפוש. נבחר בשל יעילות בטיפול ב-O/I אינטנסיבי.
- **Catalog Service (NET.)**: שירות ליבה האחראי על ניהול המוצרים, הקטגוריות והמלאי. נבחר בשל יציבותו ועובדת אופטימלית מול SQL.
- **Scaling & Availability CPU/RAM Autoscale Rules**: שימוש ב-Scaling & Availability מבוסס על CPU/RAM ופרישה ב-Slots (שיטה Blue/Green) למינית downtime (זמן חיסכון) בעקבות עדכוני גרסה.

ג. שכבת הנתונים והאחסון (Data & Storage)

- **SQL Server Database**: ה-Source of Truth למידע קטלוגי ופיננס. מגן באמצעות TDE ו-Always Encrypted.
- **Azure AI Search (Elasticsearch)**: מנוע אינדקס המאפשר חיפוש "Fuzzy" מהיר ושליפה אגרגטיבית של היסטוריות הזמן.
- **Azure Cache for Redis**: ניהול קניות (עגלת קניות) ושמירת קטלוג בזיכרון לטווח קצר על בסיס הנתונים.

ד. אופרציה ותשתיית קוד (IaC & DevOps)

- **Terraform**: כל המשאבים (Data, Compute, Networking) מוגדרים ומנווהים כקוד. Terraform מבטיח מינית Environment Drift, ניהול גרסאות לתשתיות ויכולת Disaster Recovery מלאה על ידי הקמת הסביבה מחדש תוך דיקוט.
- **CI/CD Pipeline**: אוטומציה מלאה של פריסת הקוד והתשתיות מ-GitHub ל-Azure ללא מגע ידי אדם.
- **Observability Application Insights**: ניתוח מקצה לקצה באמצעות Application Insights (לביצועי אפליקציה) ו-Log Analytics (לוגים תשתיתיים).

3. אסטרטגיית אבטחה (Zero Trust)

1. **NSG-Micro-Segmentation**: חלוקת ה-NetV לサבנטים מבודדים עם חוקי NSG קשיים.
2. **Private Link**: חסימת גישה ציבורית לכל שירות ה-SQL-PaaS, Redis, Bus, PaaS (Search Private Endpoints) – גישה פנימית בלבד דרך.
3. **Managed Identities**: חיבורם ללא סיסמה (Secretless) בין השירותים לשירותי הנזוניים.
4. **Azure Key Vault**: ניהול מרכזי של סודות ופתחות הצפנה בגישה RBAC.

4. ניתוח סיכונים וחלופות (Design Rationales)

- **למה PaaS ולא VM?** הצורך ב-Scaling מיידי (למשל ב"בלאך פריד") מחייב שירותים שיכולים לגדול תוך שניות, יכולת שאינה קיימת ב-VMs בצורה עיליה.
- **למה Terraform?** בגלל מורכבות הרשות (Private Links) והצריך בעקבות מוחלתת בין סביבת פיתוח לייצור.
- **למה Service Bus (Premium)?** כדי לאפשר Decoupling בין שירות ההזמנות לקטלוג, והבטיח עמידה בתקני אבטחה של רשות פרטית.

3. נימוקים לבחירת הטכנולוגיות (Design Rationales)

למה דואק איזו?	טכנולוגיה	שימוש בפרויקט
מצטיין ב-I/O/Aynchronous ובטיפול בהרבה בקצבות קטנות ומהירות מול ספקי Solikha - Elastic.	Node.js	ניהול הזמינות

שפה חזקה (Strongly Typed) המתאימה ללוגיקה עסקית מורכבת ומציעה ביצועים מעולים מול SQL Server.	.NET Core	ניהול קטלוג
הבטחת דיקט מוחלט (ACID) במקריםים מלאים, ויכולת ביצוע שאלות רציניות מורכבות.	SQL Server	מוצרים וקטגוריות
מאפשר חיפוש טקסטואלי חכם ושליפה מהירה של היסטוריית הזמן מתוך נפח נתונים גדולים.	Elasticsearch	חיפוש והזמנאות
משמש כ"בולם עצזעים"; מבטיח שהזמן לא ייכו לאיוב גם בעומס חריג ומנטק צימוד בין השירותים.	Service Bus	תקשורת בין שירותים
מאפשר הקמה ו恢復 של כל הסביבה מקוד, מונע טיעויות אנוש ומבטיח זהות בין סביבות.	Terraform	תשתיות IaC()