Національний університет “Києво-Могилянська Академія”

Диплом

Розробка веб-застосування на мікросервісній архітектурі та порівняння Microsoft Azure та AWS для його розгортання

Виконав:
Ахмадов Олексій Сергійович

Науковий керівник:
Гречко Анастасія Валеріївна

Актуальність

Поширення хмарних технологій

Зростання популярності мікросервісної архітектури

Microsoft Azure та AWS є лідерами серед хмарних провайдерів із широким набором сервісів

Провайдер	Частка ринку
AWS	30%
Azure	21%
Google Cloud	12%

Мета та задачі

Мета: Розробити адаптивну менторську платформу з мікросервісною архітектурою та реалізувати її деплой у Microsoft Azure та AWS для порівняння часу, вартості, легкості та зручності використання цих сервісів та опанування навичок роботи з цими платформами

Проєктування

Визначення архітектури, вибір технологій

Розробка

Написання коду, реалізація бізнес-логіки

Деплой Azure

Налаштування Azure AKS, реєстру ACR, бази, secrets

Деплой AWS

Налаштування AWS: EKS, ECR, IAM, EC2

Порівняння

Порівняння Azure і AWS за критеріями: вартість, швидкість деплою, масштабованість, моніторинг

Ринок менторських платформ

ADPList — глобальний волонтерський маркетплейс

MentorCruise — платна підписка від $50/міс

GrowthMentor — бізнес-фокус, $60–120/міс

Ключові порівняння

Платформа	$ / міс	Локалізація	Кастомізація
ADPList	0*	EN	—
MentorCruise	50–150	EN	—
GrowthMentor	60–120	EN	—
MentorMatch	0	UA / EN	+

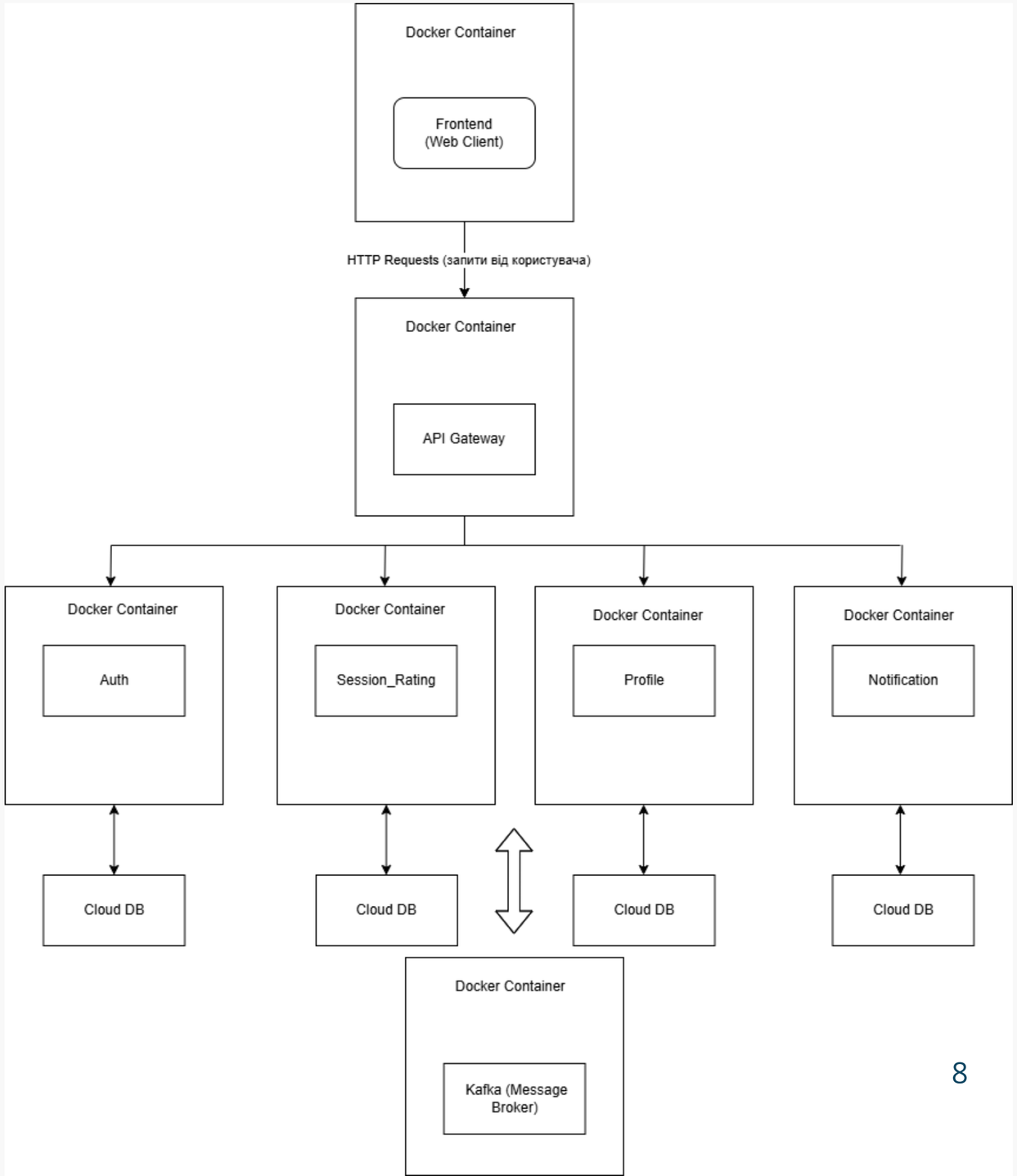
Функціональні вимоги до системи

1. *Реєстрація та авторизація через JWT, підтримка ролей: MENTEE / MENTOR / ADMIN*
2. *Профілі менторів і менті з фільтрацією за навичками, компаніями, рейтингом*
3. *Бронювання сесій: запит, підтвердження, скасування, завершення*
4. *Система відгуків і рейтингів для обох сторін*
5. *Сповіщення подій у системі*

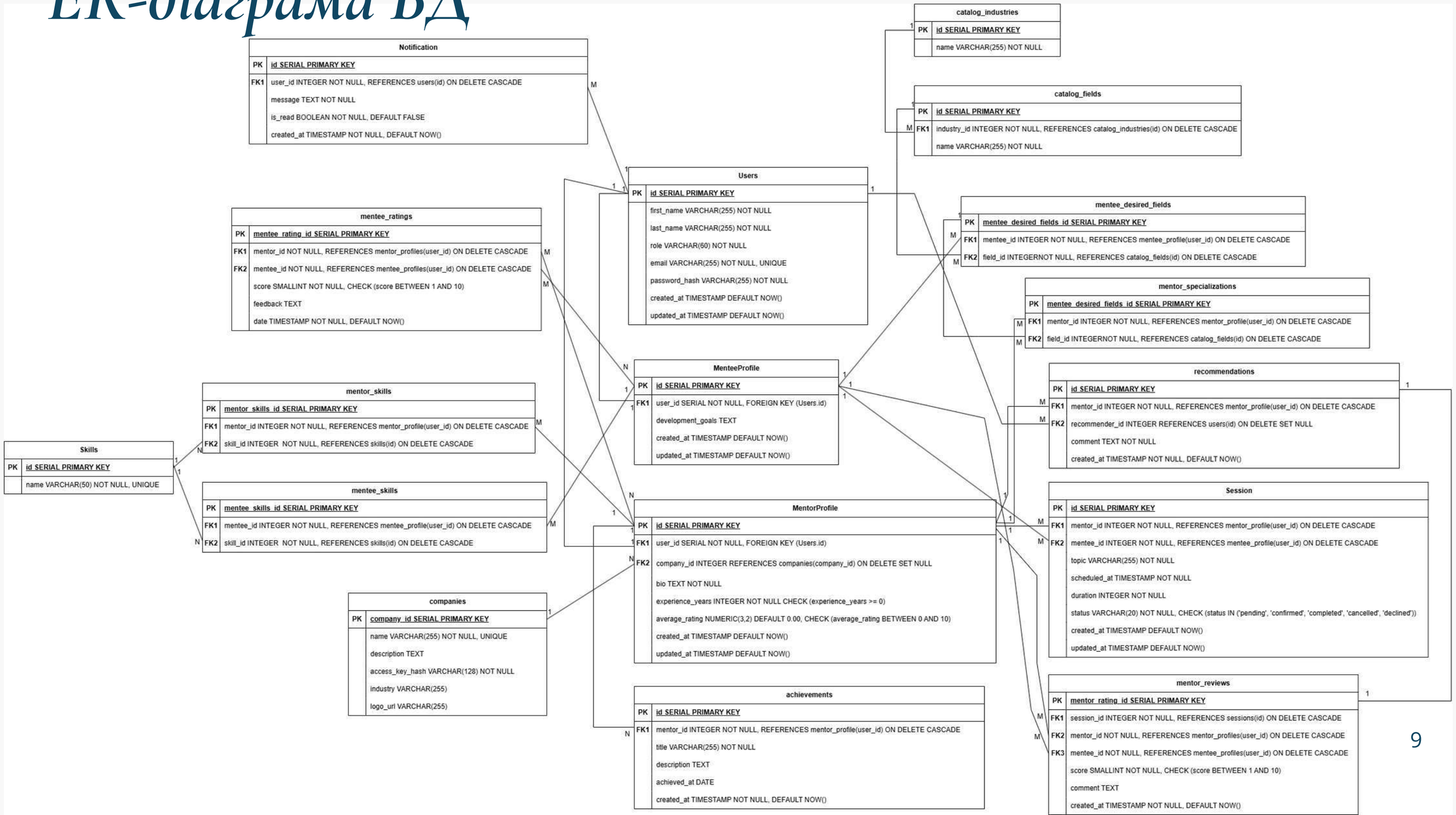
Обрані технології розробки

Категорія	Обрано
БД	Azure Database for PostgreSQL, Amazon RDS for PostgreSQL
Події	Kafka
Оркестрація	Kubernetes (AKS для Azure, EKS для AWS)
Моніторинг	Prometheus + Grafana

Загальна архітектура

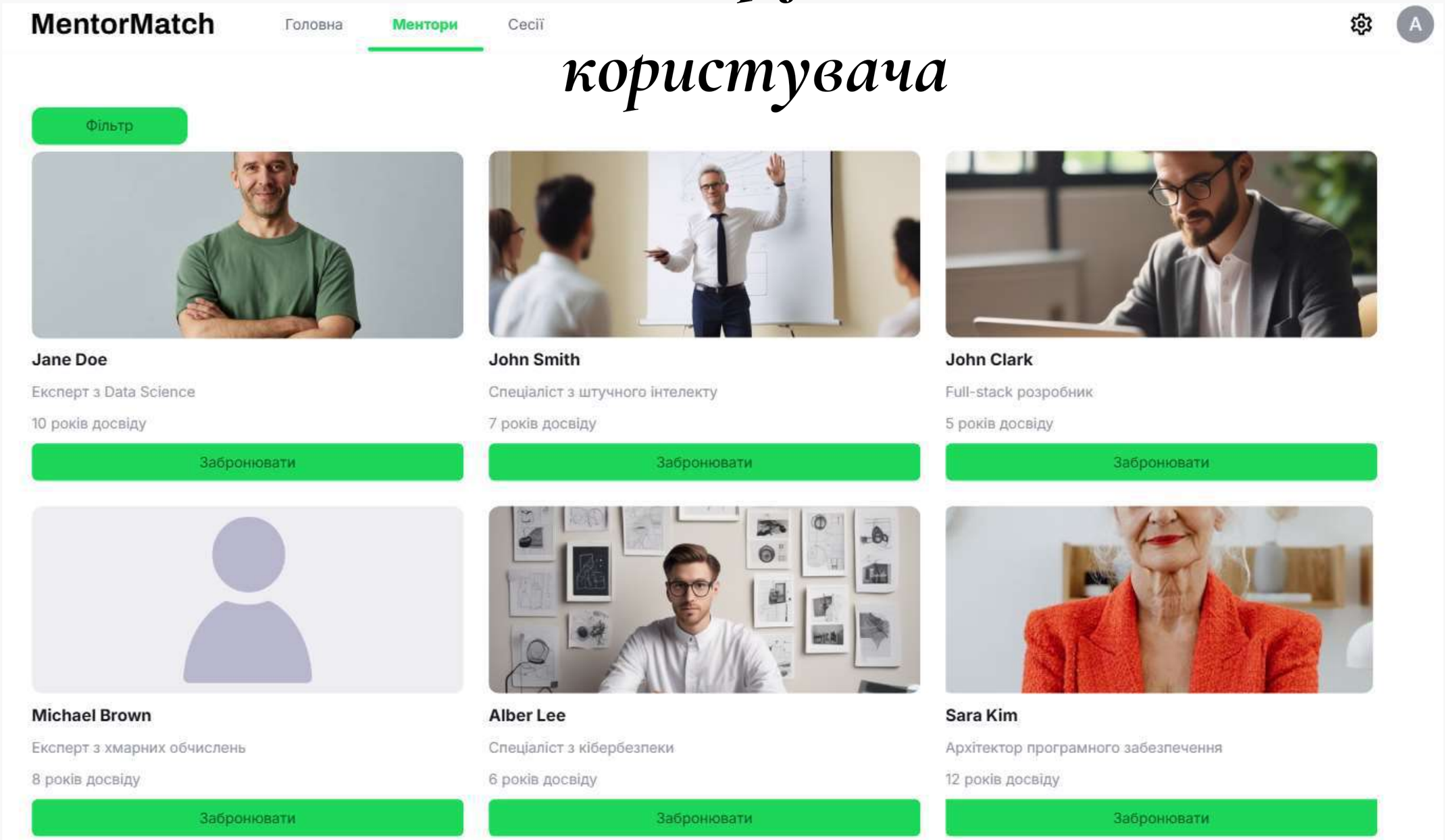


ER-діаграма БД



Інтерфейс користувача

Інтерфейс користувача



Створення AKS та EKS

Microsoft Azure

```
az aks create \
  --resource-group mentormatch-rg \
  --name mm-cluster \
  --node-count 2 \
  --node-vm-size Standard_B2s \
  --generate-ssh-keys
```

Azure створює кластер та нод-групу за замовчуванням автоматично. Менше контролю, але швидкий старт.

Amazon AWS

```
aws eks create-cluster \
  --name "mentormatch-eks" \
  --region "eu-north-1" \
  --kubernetes-version "1.32" \
  --role-arn "arn:aws:iam::476864926250:role/AmazonEKSAutoClusterRole" \
  --resources-vpc-config subnetIds=subnet-0e74b2149aeaec979, \
  subnet-08962afe0b5881d70,subnet-0b7fc881f09e2fb49,endpointPublicAccess=true


aws eks create-nodegroup \
  --cluster-name "mentormatch-eks" \
  --nodegroup-name "worker-v1" \
  --node-role "arn:aws:iam::476864926250:role/AmazonEKSNodeRole_main" \
  --subnets subnet-0e74b2149aeaec979 subnet-08962afe0b5881d70 subnet-0b7fc881f09e2fb49 \
  --scaling-config minSize=1,maxSize=2,desiredSize=1 \
  --instance-types t3.medium \
  --disk-size 20 \
  --region "eu-north-1"
```

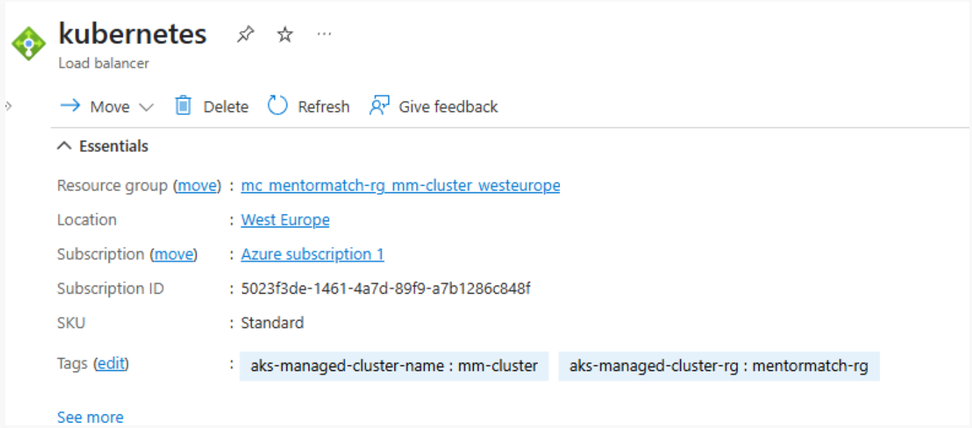
У AWS конфігурація кластера і нод-групи розділена. Більше контролю — складніше налаштування.

Load Balancer ma Ingress

Microsoft Azure

```yaml
apiVersion: v1
kind: Service
metadata:
  name: api-gateway
spec:
  type: LoadBalancer
  selector:
    app: api-gateway
  ports:
    - name: http
      port: 80
      targetPort: 80
```

Kubernetes Service типу LoadBalancer (створює Azure Load Balancer)

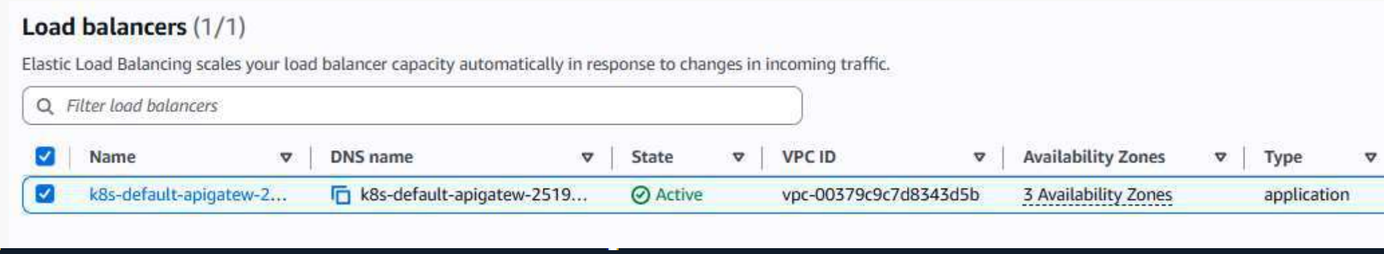


Load Balancer в Azure Portal та автоматичне створення зовнішнього порта

Amazon AWS

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: api-gateway-ingress
  annotations:
    alb.ingress.kubernetes.io/scheme: internet-facing
    alb.ingress.kubernetes.io/target-type: ip
    alb.ingress.kubernetes.io/listen-ports: '[{"HTTP": 80}]'
    alb.ingress.kubernetes.io/backend-protocol: HTTP
    kubernetes.io/ingress.class: alb
```

Ingress з ALB та анотаціями (створює AWS Application Load Balancer)



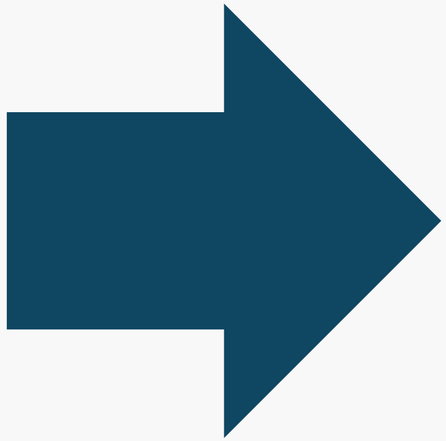

```
~ $ kubectl get ingress
NAME                CLASS    HOSTS   ADDRESS                                                          PORTS
api-gateway-ingress <none>   *       k8s-default-apigatew-25193b2fe1-1428406214.eu-north-1.elb.amazonaws.com   80
~ $
```

ALB у AWS Console, створений через Ingress Controller

Secret

.ENV

```
# Django settings
SECRET_KEY=your_secret_key_here

# PostgreSQL AWS connecting
DB_NAME=your_db_name
DB_USER=your_db_user
DB_PASSWORD=your_db_password
DB_HOST=your_db_host_url
DB_PORT=5432

ADMIN_EMAIL=admin@example.com
ADMIN_FIRST_NAME=Admin
ADMIN_LAST_NAME=User
ADMIN_PASSWORD=your_admin_password
```

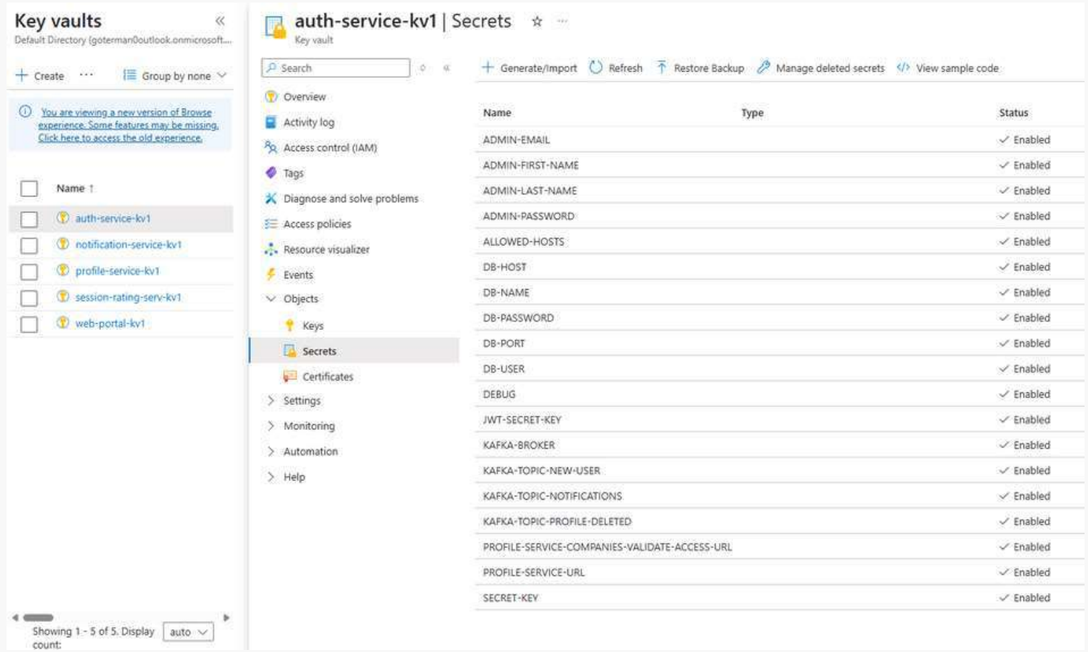


service-env.yaml

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: auth-service-env
type: Opaque
stringData:
  # Django settings
  SECRET_KEY: "your_secret_key_here"
  # PostgreSQL AWS connecting
  DB_NAME: "your_db_name"
  DB_USER: "your_db_user"
  DB_PASSWORD: "your_db_password"
  DB_HOST: "your_db_host_url"
  DB_PORT: "5432"
  # Admin user
  ADMIN_EMAIL: "admin@example.com"
  ADMIN_FIRST_NAME: "Admin"
  ADMIN_LAST_NAME: "User"
  ADMIN_PASSWORD: "your_admin_password"
```

Зберігання секретів у хмарі та інтеграція з Kubernetes через CSI Driver

Microsoft Azure - Key Vault

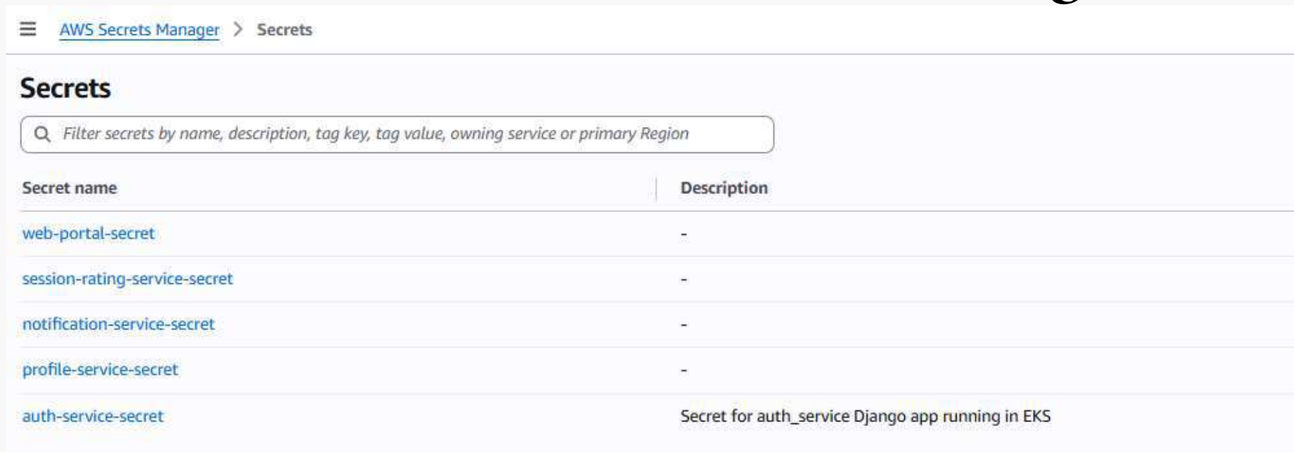


Key Vault зі списком секретів для кожного мікросервісу

```
1. Встановлення CSI драйвера:
   helm upgrade --install csi-secrets-store secrets-store-csi-driver/secrets-store-csi-driver \
     --namespace kube-system

2. Встановлення Azure provider:
   kubectl apply -f https://raw.githubusercontent.com/Azure/secrets-store-csi-driver-provider-azure/master/deployment/secretproviderclasspodstatus-crd.yaml
   kubectl apply -f https://raw.githubusercontent.com/Azure/secrets-store-csi-driver-provider-azure/master/deployment/provider-azure-installer.yaml

3. Створення SP (Service Principal) з доступом до Key Vault:
   az ad sp create-for-rbac \
     --name "k8s-keyvault-access" \
     --role "Key Vault Secrets User" \
     --scopes /subscriptions/.../vaults/auth-service-kv1
```

Налаштування інтеграції з Azure Key Vault

Amazon AWS - Secrets Manager



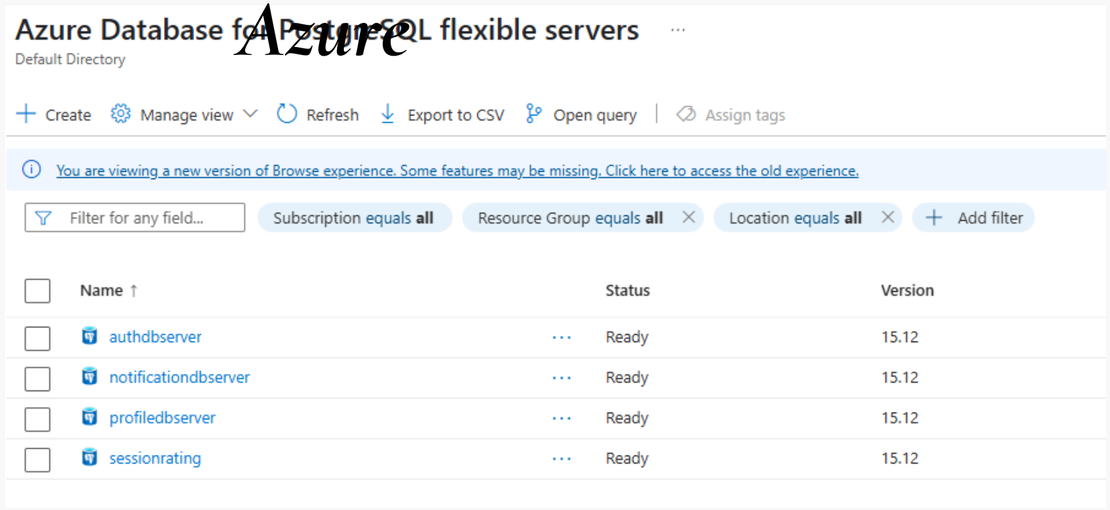
Секрети для кожного мікросервісу зберігаються в AWS Secrets Manager

```
helm repo add secrets-store-csi-driver https://kubernetes-sigs.github.io/secrets-store-csi-driver/charts
helm repo update

aws eks update-kubeconfig --name mentormatch-eks --region eu-north-1

helm upgrade --install csi-secrets-store secrets-store-csi-driver/secrets-store-csi-driver \
  --namespace kube-system
kubectl get pods -n kube-system | grep csi-secrets-store

kubectl apply -f \
https://raw.githubusercontent.com/aws/secrets-store-csi-driver-provider-aws/main/deployment/aws-provider-install


kubectl apply -f csi-driver-secret-rbac.yaml
```

Команди для встановлення AWS Secrets Store CSI Driver

Хмарні бази даних для мікросервісів

Microsoft Azure

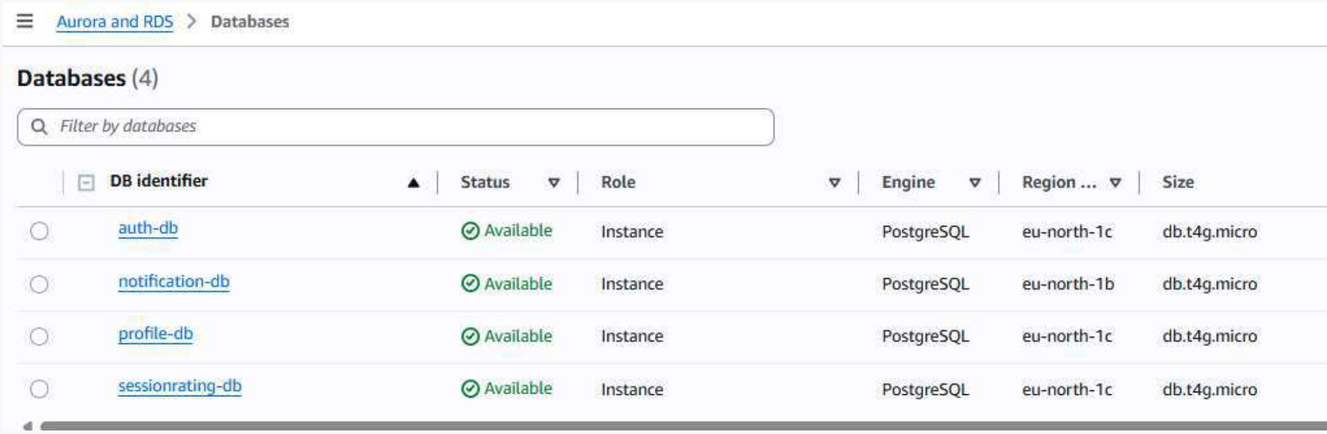


Azure Database for PostgreSQL Flexible Server

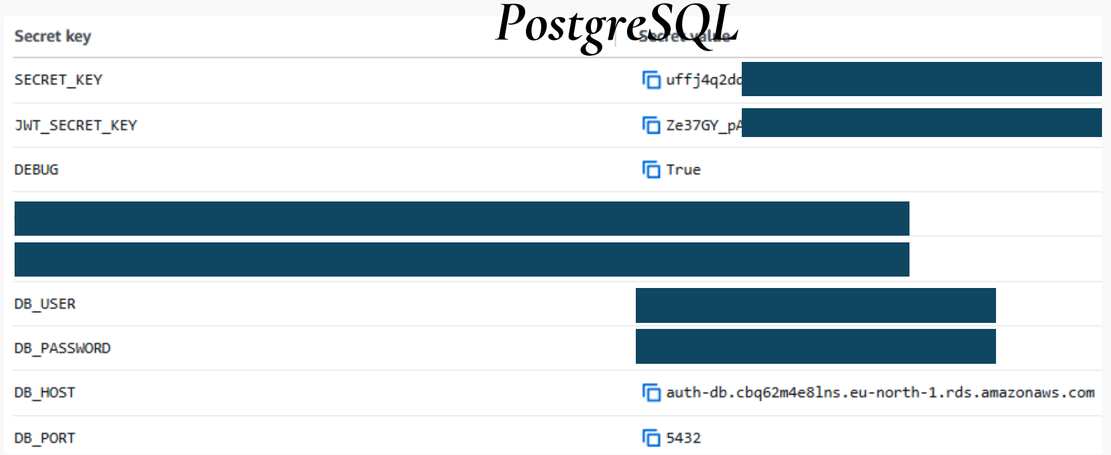
```
# PostgreSQL Azure connecting
DB_NAME=postgres
DB_USER=
DB_PASSWORD=
DB_HOST=authdbserver.postgres.database.azure.com
DB_PORT=5432
```

Підключення до Azure PostgreSQL

Amazon AWS



Amazon RDS for PostgreSQL



Підключення до AWS RDS

Скрипти

Microsoft Azure

```
az login

echo "===== [2/7] Create resource group, ACR, AKS ====="
az group create --name mentormatch-rg --location westeurope

az acr create --resource-group mentormatch-rg --name mmregistry --sku Basic

az aks create \
  --resource-group mentormatch-rg \
  --name mm-cluster \
  --node-count 2 \
  --node-vm-size Standard_B2s \
  --generate-ssh-keys

az aks update --name mm-cluster --resource-group mentormatch-rg --attach-acr mmregistry

echo "===== [3/7] Get AKS credentials ====="
az aks get-credentials --resource-group mentormatch-rg --name mm-cluster --overwrite-existing

# === CSI Driver install (Kube System Namespace, as required) ===
echo "===== [4/7] Install Azure Key Vault CSI Driver ====="
helm repo add secrets-store-csi-driver https://kubernetes-sigs.github.io/secrets-store-csi-driver/charts
helm repo update

helm upgrade --install csi-secrets-store secrets-store-csi-driver/secrets-store-csi-driver \
  --namespace kube-system

kubectl apply -f https://raw.githubusercontent.com/Azure/secrets-store-csi-driver-provider-azure/master/deployment/secr
kubectl apply -f https://raw.githubusercontent.com/Azure/secrets-store-csi-driver-provider-azure/master/deployment/prov

# === OIDC/Federation: This is important for production (no static secrets!) ===
echo "===== [5/7] Setup AKS OIDC and federated identity ====="
OIDC_ENABLED=$(az aks show --resource-group mentormatch-rg --name mm-cluster --query "oidcIssuerProfile.enabled" -o tsv
if [[ "$OIDC_ENABLED" != "true" ]]; then
  az aks update --resource-group mentormatch-rg --name mm-cluster --enable-oidc-issuer
fi

OIDC_URL=$(az aks show --resource-group mentormatch-rg --name mm-cluster --query "oidcIssuerProfile.issuerUrl" -o tsv)
echo "OIDC Issuer URL: $OIDC_URL"
```

Фрагмент Bash-скрипт для деплою в Microsoft Azure

Amazon AWS

```
eksctl create iamserviceaccount \
  --region eu-north-1 \
  --cluster mentormatch-eks \
  --namespace kube-system \
  --name aws-load-balancer-controller \
  --attach-policy-arn arn:aws:iam::476864926250:policy/AWSLoadBalancerControllerIAMPolicy \
  --approve \
  --override-existing-serviceaccounts

# 8. Install AWS Load Balancer Controller with Helm
echo "[Step 8] Install AWS Load Balancer Controller"
VPC_ID=$(aws eks describe-cluster \
  --name mentormatch-eks \
  --region eu-north-1 \
  --query "cluster.resourcesVpcConfig.vpcId" \
  --output text)

helm uninstall aws-load-balancer-controller -n kube-system || true

helm install aws-load-balancer-controller eks/aws-load-balancer-controller \
  -n kube-system \
  --set clusterName=mentormatch-eks \
  --set serviceAccount.create=false \
  --set serviceAccount.name=aws-load-balancer-controller \
  --set region=eu-north-1 \
  --set vpcId=$VPC_ID

kubectl get pods -n kube-system -l app.kubernetes.io/name=aws-load-balancer-controller

# 9. Install Zookeeper and Kafka with persistent storage
echo "[Step 9] Install Zookeeper and Kafka with persistent storage"
helm install zookeeper bitnami/zookeeper \
  --set replicaCount=1 \
  --set persistence.enabled=true \
  --set persistence.storageClass=gp2 \
  --set persistence.size=2Gi

helm install kafka bitnami/kafka \
  --set replicaCount=1 \
  --set zookeeper.enabled=false \
  --set externalZookeeper.servers=zookeeper.default.svc.cluster.local:2181 \
```

Фрагмент Bash-скрипт для деплою в AWS

Azure vs AWS

Характеристики	*Azure*	AWS
Безкоштовний пакет послуг	200$	$100 - навчальний заклад є учасником AWS Educate $30–$75 - не є учасником AWS Educate
Панель управління	Azure Portal — легко зрозумілий для початківця	AWS Console гнучкіша, але складна. Для всього потрібні налаштування політик та ролей
Підтримка Kubernetes	AKS має простішу інтеграцію з ACR та автоматичне оновлення kubeconfig	EKS дає більше контролю, але потребує ручного налаштування (OIDC, IAM, LoadBalancer)
Балансування навантаження	Під капотом	Повне ручне налаштування

Висновки

- Проаналізовано існуючи системи-аналоги та сформульовано вимоги до створюваної системи
- Розроблено повноцінне веб-застосування із мікросервісною архітектурою, з можливостями локалізації та кастомізації. Проект має потенціал до масштабування як внутрішній інструмент для компаній, освітніх закладів та спільнот.
- Використано сучасні інструменти, які застосовуються у продакшн-середовищах: Kafka, Docker, Kubernetes, Prometheus, Grafana, хмарні бази даних та системи зберігання секретів.
- Порівняно розгортання системи на хмарних платформах Azure та AWS

Питання