DOI: https://doi.org/10.36719/2663-4619/112/311-315

Yusif Osmanov
Bakı Dövlət Universiteti
magistrant
https://orcid.org/0009-0001-9626-0004
osmanov.yusif@gmail.com

Mikroservislər haqqında ümumi təsnifat

Xülasə

Mikroservislər şəbəkə üzərindən bir-biri ilə əlaqə saxlayan kiçik, müstəqil xidmətlər toplusu kimi proqram təminatının işlənib hazırlanması üçün memarlıq yanaşmasıdır. Bütün funksionallığın vahid kod bazasına sıx inteqrasiya olunduğu monolit proqram qurmaq əvəzinə, mikroservislər tətbiqi daha kiçik, sərbəst birləşdirilən xidmətlərə parçalayır. Bunlar, xüsusi biznes funksiyasını yerinə yetirmək üçün nəzərdə tutulmuş kiçik, sərbəst birləşdirilən xidmətlərdir və hər bir mikroservis müstəqil olaraq inkişaf etdirilə, yerləşdirilə və miqyaslana bilər. Bu arxitektura böyük monolit tətbiqi götürməyə və onu kiçik idarə olunan komponentlərə (xidmətlərə) parçalamağa imkan verir. Həmçinin, müasir tətbiqlərin tikinti bloku hesab olunur. Mikroservislər müxtəlif proqramlaşdırma dillərində və çərçivələrdə yazıla bilər və hər bir xidmət özlüyündə kiçik proqram kimi çıxış edir.

Açar sözlər: *mikroservis, monolit, vahid kod bazası, komponentlər, sərbəst birləşdirilən xidmətlər, xüsusi biznes funksiyası*

Yusif Osmanov
Baku State University
Master student
https://orcid.org/0009-0001-9626-0004
osmanov.yusif@gmail.com

General Classification of Microservices

Abstract

Microservices are an architectural approach to software development that consists of a set of small, independent services that interact with each other over a network. Instead of creating a monolithic application where all functionality is tightly integrated into a single codebase, microservices break the application into smaller, loosely coupled services. These services are designed to perform specific business functions and can be developed, deployed, and scaled independently. This approach allows a large monolithic application to be transformed into small, manageable components (services). Additionally, microservices are considered the building blocks of modern applications. They can be written in different programming languages and frameworks, with each service functioning as an independent program.

Keywords: *microservices, monolith, single code base, components, loosely coupled services, specific business function*

Giriş

Mikroservislər mürəkkəb proqram təminatının yerləşdiyi arxitektura üslubudur, hansı ki, bir və ya bir neçə xidmətdən ibarət olur. Mikroservislər bir-birindən asılı olmadan və ya bir-biri ilə zəif şəkildə əlaqələndirilərək, yerləşdirilə bilər. Bu mikroservislərin hər biri yalnız bir işi yüksək səviyyədə yerinə yetirmək üçün nəzərdə tutulub. Bu tapşırıq, yəni iş adətən kiçik bir biznes imkanını təmsil edir. Həmçinin, mikroservislər istənilən proqramlaşdırma dilində inkişaf etdirilə bilər. Onlar bir-biri ilə dildən asılı olmayan tətbiq proqram interfeysləri (API-lər), məsələn, Representational State Transfer (REST) vasitəsilə əlaqə qururlar.

Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0)

Mikroservislər həmçinin məhdud bir kontekstə malikdirlər. Başqa mikroservislərin arxitekturası və ya daxili iş prinsipləri barədə məlumatları olmamalıdır.

Tədqiqat

1. Mikroservislərin mürəkkəb və fərqli platformalarda tədqiqi

Kiçik və yönümlü - Mikroservislər müəyyən bir işə yönəlik olmalı və buna görə də kiçik olmalıdır. Mikroservisin nə qədər kiçik olması ilə əlaqədar konkret qaydalar yoxdur. Əsasən istinad edilən qaydalardan biri "İki Pizza Komandası" qaydasıdır. Bu qaydaya görə, mikroservisi yaradan komandaya iki pizza kifayət etmirsə, mikroservis çox böyükdür. Mikroservisi yenidən yazmaq və onu asanlıqla idarə etmək üçün onu kifayət qədər kiçik saxlamaq vacibdir.

İnterfeysi kiçik saxlamaq: Burada diqqət, interfeysi kiçik saxlamağa yönəlməlidir. Adətən kiçik bir implementasiya tələb etsə də, bu həmişə belə olmaya bilər.

Mikroservisə həmçinin müstəqil bir tətbiq və ya məhsul kimi yanaşmaq lazımdır. Onun özünə məxsus mənbə kodu idarəetmə anbarı, həmçinin quraşdırma və yerləşdirmə üçün öz xətti olmalıdır. Hərçənd məhsul sahibi mikroservisin təkrar istifadəsini dəstəkləyə bilər, amma təkrar istifadə mikroservislər üçün yeganə iş prinsibi deyil. Burada UI (istifadəçi interfeysi) reaksiya sürətini artırmaq və müştəri ehtiyaclarına daha sürətli cavab vermək kimi digər iş prinsibləri də mövcuddur.

Mikroservisin ölçüsü biznes ehtiyaclarına əsaslanaraq da, müəyyən edilə bilər. Məsələn, paket izləmə, avtomobil sığortası təklifi xidməti və ya hava proqnozu kimi xidmətlər üçüncü tərəf təminatçıları tərəfindən təmin oluna bilər və ya əsas səriştəlilik olaraq, ödənişli xidmət kimi təqdim edilə bilər.

Gecikmə məsələsi: Xidmətləri çox kiçik saxlamaq və ya digər mikroservislərlə çoxlu asılılıq yaratmaq, gecikməyə səbəb ola bilər.

Zəif şəkildə əlaqələndirilmiş - Zəif əlaqə mikroservislərin əsas xarakteristikalarındandır. Bir mikroservisin tək başına yerləşdirilə bilməsi lazımdır. Yerləşdirilmə üçün, digər mikroservislərlə heç bir əlaqələndirməyə ehtiyac olmamalıdır. Bu zəif əlaqə, tez-tez və sürətli işləməyi təmin edir və buna görə də tələb olunan funksionallıqları istifadəçilərə çatdırmağa imkan verir.

Dildən asılı olmayan - Düzgün iş üçün, düzgün alətdən istifadə etmək vacibdir. Mikroservislər, işi yerinə yetirmək üçün ən uyğun olan proqramlaşdırma dili və texnologiya ilə qurulmalıdır. Mikroservislər mürəkkəb tətbiqlərin bir hissəsi kimi birləşdirilir və bu səbəbdən, eyni proqramlaşdırma dilində yazılmalarına ehtiyac yoxdur. Məsələn, bəzi hallarda Java uyğun ola bilər, digərlərində isə Python daha doğru seçim ola bilər. Mikroservislərlə əlaqə dildən asılı olmayan API-lər vasitəsilə, adətən HTTP (Hipermətn Transfer Protokolu) əsaslı mənbə API-ləri, məsələn, REST vasitəsilə həyata keçirilir. Mikroservisin platformasını deyil, inteqrasiyasını standartlaşdırmaq məsləhətdir. Dildən asılı olmamaq, mövcud bacarıqlardan istifadə etməyi və ya optimal dili seçməyi asanlaşdırır.

Məhdud kontekst - Məhdud kontekst dedikdə, konkret bir mikroservisin ətrafında olan digər mikroservislərin daxili iş prinsipləri barədə heç bir məlumatının olmaması nəzərdə tutulur. Əgər hər hansı bir səbəbdən mikroservis başqa bir mikroservisin nə etdiyini və ya necə çağrılacağını bilməlidirsə, o deməkdir ki, burada məhdud kontekst yoxdur (Nasslahsen, 2024, s. 596).

Aşağıda, mikroservislərin istifadəsi ilə yaranan mürəkkəbliklə mübarizə aparmaq üçün bəzi amillər təsvir edilmişdir:

Testlər:

- Mürəkkəb sistemlərin test edilməsi vahid və inteqrasiya test dəstlərini, həmçinin nasaz və ya nasazlaşan komponentləri, xidmətin aşağı düşməsini, stress testlərini və ümumi sistemin gündəlik davranışını simulyasiya edən bir test çərçivəsini tələb edir. Belə test çərçivələri sistemin düzgünlüyünü və ümumi dayanıqlığını təmin edir və eyni zamanda monitorinq vasitəsi kimi istifadə oluna bilər.
- Mikroservis arxitekturasında bir komponentin sıradan çıxması zamanı sistemin davranışını təmin etmək diqqət mərkəzində olduğundan, istehsal mühitində sıradan çıxmanı test etmək qeyri-adi deyil. Avtomatlaşdırılmış testlər bir komponentin sıradan çıxmasını məcbur edir və sistemin bərpa olunma qabiliyyətini və monitorinq imkanlarını test edir.

Monitorinq:

- Ənənəvi monitorinq sistemlərinə əlavə olaraq, yaxşı dizayn edilmiş xidmətlər mesaj növbəsi üzərindən **nəşr/abunə** (publish/subscribe) yolu ilə əlavə sağlamlıq yoxlama məlumatları təqdim edə bilər. Bu məlumatlar monitorinq xidməti tərəfindən toplanaraq zəngin sağlamlıq və konfiqurasiya göstəriciləri təqdim edir.
- Bu göstəricilər xidmətlərin cari sorğu həcmini təmin etmək üçün kifayət qədər potensiala malik olub-olmadığını təmin etmək və cari performansı gözləntilərlə müqayisə etmək üçün istifadə edilə bilər. Əlavə məlumat toplamaq üçün xüsusi sağlamlıq yoxlamaları qeydiyyatdan keçirilə bilər. Bütün toplanmış məlumatlar birləşdirilərək monitorinq panelində göstərilə bilər.

DevOps:

- Mikroservisləri işlək vəziyyətdə saxlamaq üçün əməliyyatların çağırışları yüksək keyfiyyətli **DevOps** bacarıqlarını inkişaf komandasına daxil etməyi tələb edir. Komanda əməliyyat yönümlü və istehsalat bilikli olmalıdır.

Poliqlot proqramlaşdırma:

- Mikroservislərin təbiətinə uyğun istifadəsi sistemin poliqlot olmasını tələb edir ki, bu da hiper-poliqlot sistemin idarə edilməsi ilə bağlı unikal çətinlik yaradır. Bu halda ənənəvi verilənlər bazası administratoru (DBA) artıq noSQL məhsullarını yerləşdirə, işə sala və optimallaşdıra bilən proqramçılarla əvəz olunmalıdır.
- Bacarıqlı inkişaf etdiricilərin, araşdırıcıların və yenilikçilərin boru xəttini saxlamaq, DevOps boru xəttini saxlamaq qədər vacibdir. Bu, platformanı saxlamaq və yeni irəliləyişlər əldə edildikcə sistem dizaynını təkmilləşdirmək üçün kifayət qədər bacarıqlı inkişaf etdiricilərin olmasını təmin edir.
- Qeyd etmək lazımdır ki, əksər təşkilatlar istifadə etdikləri proqramlaşdırma dillərinin qısa siyahısına malikdirlər. Kod və bacarıq təkrar istifadəsi baxımından bunun üstünlükləri var.

Təkamül dizaynı: Mikroservis arxitekturası təkamül dizayn yanaşmasını təmin edir. Əsas fikir odur ki, tətbiqinizə yeni xüsusiyyətlər və imkanlar əlavə etmək üçün yalnız bir mikroservisi dəyişdirməklə bunu edə bilərsiniz. Siz yalnız dəyişdirdiyiniz mikroservisləri yerləşdirmək və istifadəyə vermək məcburiyyətindəsiniz. Bu dəyişiklik yalnız həmin mikroservisin istifadəçilərinə təsir edir və əgər interfeys dəyişməyibsə, bütün istifadəçilər öz funksionallıqlarını davam etdirirlər. Mikroservislərin zəif birləşdirilməsi, kiçik və fokuslanmış olması, eləcə də məhdud kontekstə malik olması səbəbindən onlara dəyişikliklər sürətli və tez-tez edilə bilər, həm də minimal risklə. Mikroservisin asanlıqla yenilənə bilməsi onun təkamülünü təmin edir. Mikroservislər kiçik olduğu və adətən kiçik, "iki pizza" komandası tərəfindən dizayn edildiyi, inkişaf etdirildiyi və saxlanıldığı üçün mikroservislərin tamamilə yenidən yazılması adi bir haldır. Bu, onların təkmilləşdirilməsi və ya saxlanılması əvəzinə baş verir. Ən çox rast gəlinən kompromis isə mikroservisin nə qədər kiçik edilməsidir (Saavedra, Rupp, et al., 2020, s. 244).

Monolit bir tətbiqin mikroservislərə bölünməsində mikroservislərin sayı nə qədər çox olarsa, hər biri o qədər kiçik olar. Hər biri nə qədər kiçikdirsə, ona edilən dəyişikliklərin yerləşdirilməsi və istifadəyə verilməsi riski bir o qədər az olar. Bununla belə, hər bir mikroservis nə qədər kiçikdirsə, bir tətbiqin yenilənməsi zamanı daha çox mikroservisin yerləşdirilməsi tələb olunur və bu, riski artırır. Mikroservislərin müstəqil yerləşdirilə bilməsi və zəif birləşdirilmə kimi prinsipləri hansı imkanın mikroservis olacağına qərar verərkən nəzərə alınmalıdır. Bununla bağlı daha ətraflı təsvir növbəti bölmədə verilmişdir (Raje, 2021, s. 388-390).

Mikroservislərin dizayn edilməsi: Mikroservis arxitekturasının böyük üstünlüklərindən biri texnologiya yığını və mikroservisin ölçüsü ilə bağlı qərarları hər bir xidmət üçün ayrıca vermək azadlığıdır. Həyatda olduğu kimi, böyük azadlıqla böyük məsuliyyət də gəlir. İnkişaf etdiricilərə çoxlu seçimlərin təqdim edilməsi doğru olsa da, bu seçimlər eyni deyil və diqqətlə nəzərdən keçirilməlidir. Bu, mikroservisin istifadəçinin təcrübəsini təmin etməkdəki məqsədini aydın şəkildə anlamaqla başlayır (Gutierrez, 2020, s. 404-405).

Nəticədə, yaxşı dizayn edilmiş bir mikroservis sistemi bu qaydaların birləşməsini istifadə edir. Bu, zamanla və sistemlə təcrübə əldə etməklə əldə edilən yaxşı qiymətləndirmə tələb edir. Bu təcrübə əldə edilmədən əvvəl, təklifimiz odur ki, kiçik başlayın, mikroservislər bir qədər böyük (mini-servislər kimi) olsa da, sonradan daha çox “çatlama” (fault lines) müşahidə edildikcə, subsequent

bölmələrə gedilsin. Yaxşı dizayn edilmiş bir sistemdə, əgər sistemin qarşısında tərs proksi varsa, bu yenidən təşkilatlandırma heç bir kəsilmə olmadan həyata keçirilə bilər. Əgər bir mikroservis iki URL yolunu xidmət edirdisə, iki yeni mikroservis hər biri bir yolu xidmət edə bilər. Əxlaqı belədir ki, mikroservis sisteminin dizaynı davamlı bir prosesdir. Bu, birdən-birə, dərhal ediləcək bir şey deyil (Wells, 2024, s. 448-451).

2. Mikroservislərin əşyaların interneti (IoT) texnologiyalarında tətbiqi

IoT tətbiqlərinin inkişafı, şübhəsiz ki, müəyyən problemlərin yaranmasına gətirib çıxarır, lakin mikroxidmətlərin gətirdiyi faydalar bu mübarizənin antidotu ola bilər.

Son on ildə Əşyaların İnterneti (IoT) sürətlə böyüyüb. Hiperəlaqəlilik paradiqması artıq şəhər planlaşdırılması, nəqliyyat, səhiyyə, avtomobil sənayesi və kənd təsərrüfatı kimi bir çox real həyat ssenarilərində yer tapıb. Bir çox bizneslər üçün bu artıq aspirativ bir texnologiya məqsədi deyil, əməliyyat ehtiyacı halına gəlib. Lakin, IoT ilə əlaqəli sistemlərin böyük miqyası və mürəkkəbliyi, tətbiqlərin idarəolunan və ya təhlükəsiz şəkildə inkişaf etdirilməsi, inteqrasiyası və genişləndirilməsini çətinləşdirir. Bu çətinlikləri aradan qaldırmaq üçün mikroservis memarlığının dizayn prinsipləri və üstünlükləri IoT təşəbbüslərini dəstəkləyə bilər. Konteynerləşdirmə, modulluq, müstəqil yerləşdirmə və zəif əlaqəlilik kimi üstünlüklərdən istifadə etməklə proqram təminatı komandaları IoT tətbiqlərini daha çox əminliklə inkişaf etdirə bilərlər (Khanna, 2024, s. 334).

Baxmayaraq ki, mikroservislər IoT ilə bağlı bütün problemləri həll etməyəcək, gəlin mikroservislərlə IoT-nin bir araya gəldiyi əsas yolları nəzərdən keçirək.

Mikroservislər və IoT-nin birlikdə işləyə biləcəyi 4 əsas yol: IoT-ə investisiya qoymağa qərar verən bizneslər dərhal bir çox problemlərlə üzləşəcəklər. Proqram təminatı komandaları yeni bir sıra son nöqtə cihazlarını, tətbiq serverlərini, verilənlər bazalarını və süni intellekt əsaslı analitik alətləri birləşdirməlidir. Çox vaxt bu, bu texnologiyaların mövcud proqram təminatı və inkişaf prosesləri ilə inteqrasiyasını da tələb edir (Cole, 2018, s. 254).

IoT tətbiq texnologiyasını qəbul etmək qərarına gələn bizneslərin nəzərə almalı olduqları əsas məqamlar və burada mikroservislərlə IoT-nin birlikdə necə fəaliyyət göstərməsi (Han, Wang, Bai, & Liu, 2024, s. 295-296):

- **Bağlantı.** IoT, bulud əsaslı tətbiqlər, verilənlər bazaları, proqram platformaları və ünsiyyət nöqtələri ilə əlaqəli cihazlardan ibarət kompleks bir şəbəkəyə əsaslanır. Konteynerləşdirilmiş mikroservislər yüngül, miqyaslana bilən və istər yerli, istərsə də buludda yerləşdirilə bilən modulluq təmin edir.
- **Təhlükəsizlik.** Təşkilatlar IoT ilə bağlı fiziki cihaz pozulmaları və həssas məlumat mübadilələri ilə əlaqəli risk amillərini idarə etməli və azaltmalıdır. Mikroservislərin zəif əlaqəlilik xüsusiyyəti sayəsində xidmətlər daha geniş sistemlərə giriş qapıları kimi daha az cəlbedici olur və tezliklə təhlükəsizlik yeniləmələri ilə təkmilləşdirilə bilər.
- **Elastiklik.** Tətbiq funksionallığını tez bir zamanda uyğunlaşdırmaq və yeni texnologiyaları dəstəkləmək qabiliyyəti vacibdir, xüsusən də son illərdə IoT-nin inkişaf sürətini nəzərə alaraq. Mikroservislərin adaptiv xüsusiyyəti yeni xidmətləri tez bir zamanda təqdim etməyə və ya köhnə xidmətləri tətbiqə təsir etmədən əvəz etməyə imkan verir.
- **Məlumat bütövlüyü.** IoT, kritik məlumatların toplanması və emalı ilə əlaqədar təşkilatın məntiqli və etibarlı bir plan hazırlamaq qabiliyyətindən tam asılıdır, xüsusən bu məlumatlar mövcud sistemlərdə saxlanıldıqda. Güclü hesablama gücü tələb edən xüsusi mikroservislər ictimai buluda köçürülə bilər, daha sürətli cavab tələb edən və ya daha çox təhlükəsizlik tələb edən xidmətlər isə yerli olaraq yerləşdirilə bilər.

Mikroservis əsaslı IoT komponentləri - Mikroservis əsaslı IoT sistemi ev kompüterləri, ofis şəbəkələri, GPS cihazları, robot istehsal xətləri və daha çox kimi müxtəlif kənar yerlərə qoşulacaq. Mikroservis memarlığının ən böyük üstünlüklərindən biri müxtəlif tətbiq xidmət növləri, cihaz platformaları, proqramlaşdırma dilləri, kodlaşdırma çərçivələri və funksional kitabxanalarla asanlıqla inteqrasiya etmək qabiliyyətidir (Ouaissa, Ouaissa, Boulouard, & Kumar, 2024, s. 134).

Mikroservis əsaslı IoT proqram təminatı sistemində tapa biləcəyiniz bəzi ümumi komponentlər:

- Proqram sistemlərinə qoşulması tələb olunan fiziki son nöqtələr, məsələn sensorlar, avtomatlaşdırılmış maşınlar və mobil cihazlar.
- Bütün xarici müştəri/server tələbləri, xidmət mesajları və API zəngləri üçün tək giriş nöqtəsi və broker kimi fəaliyyət göstərən API şlüzü.
- Hesablama, saxlama və analitikanı buludda yerləşdirən bulud əsaslı infrastruktur – ehtiyaclarınıza uyğun olaraq resurslarınızı artırıb azalda bilərsiniz.
- Avtorizasiya, autentifikasiya və xidmətlərə düzgün girişin təmin edilməsi ilə əlaqəli digər prosesləri idarə edən fərdi təhlükəsizlik tokenləri.
- HTTP, XML, JSON və gRPC kimi müxtəlif proqram sistemləri arasında ünsiyyət və məlumat mübadiləsini asanlaşdıran mesajlaşma protokolları.
- Problemli xidmətləri yeniləyə bilən və nasazlıqları müəyyən edən güclü tətbiq monitorinq alətləri.

Nəticə

Tədqiqatdan alınmış elmi nəticələr aşağıdakılardan ibarətdir:

- Windows, Linux və Unix əməliyyat sistemlərində çalışan istənilən proqram təminatını monolit arxitekturadan, mikroservis tipli arxitekturaya keçirmək mümkündür.
- IoT həllərində dəqiq və sürətli nəticələr əldə etmək olur.
- C#, C++, Python və Java kimi bir çox proqramlaşdırma dili mühitində çalışan, istənilən proqram təminatını monolit arxitekturadan, mikroservis tipli arxitekturaya keçirərək daha sürətli və təhlükəsiz məlumat mübadiləsi aparmaq mümkündür.

Ədəbiyyat

1. Cole, M. R. (2018). *Hands-on microservices with C#: Designing a real-world, enterprise-grade microservice ecosystem with the efficiency of* C# 7, 254.
1. Nasslahsen, B. (2024). *Spring Security: Effectively secure your web apps, RESTful services, cloud apps, and microservice architectures* (4th ed., 596).
2. Saavedra, C., Rupp, H. W., et al. (2020). *Hands-on enterprise Java microservices with Eclipse MicroProfile: Build and optimize your microservice architecture with Java*, 244.
3. Raje, G. (2021). *Security and microservice architecture on AWS: Architecting and implementing a secured, scalable solution,* 388-426.
4. Gutierrez, F. (2020). *Spring Cloud Data Flow: Native cloud orchestration services for microservice applications on modern runtimes*, 404-420.
5. Wells, S. (2024). *Enabling microservice success: Managing technical, organizational, and cultural challenges*, 448-451.
6. Khanna, V. K. (2024). *IoT sensors: An exploration of sensors for Internet of Things (Series in Sensors*), 334.
7. Ouaissa, M., Ouaissa, M., Boulouard, Z., & Kumar, A. (2024). *Artificial intelligence for blockchain and cybersecurity powered IoT applications.*
8. Han, R., Wang, J., Bai, L., & Liu, J. (2024). *IoT signal detection*, 295-308.