ThoughtWorks®

TECHNOLOGY RADAR *VOL.17*

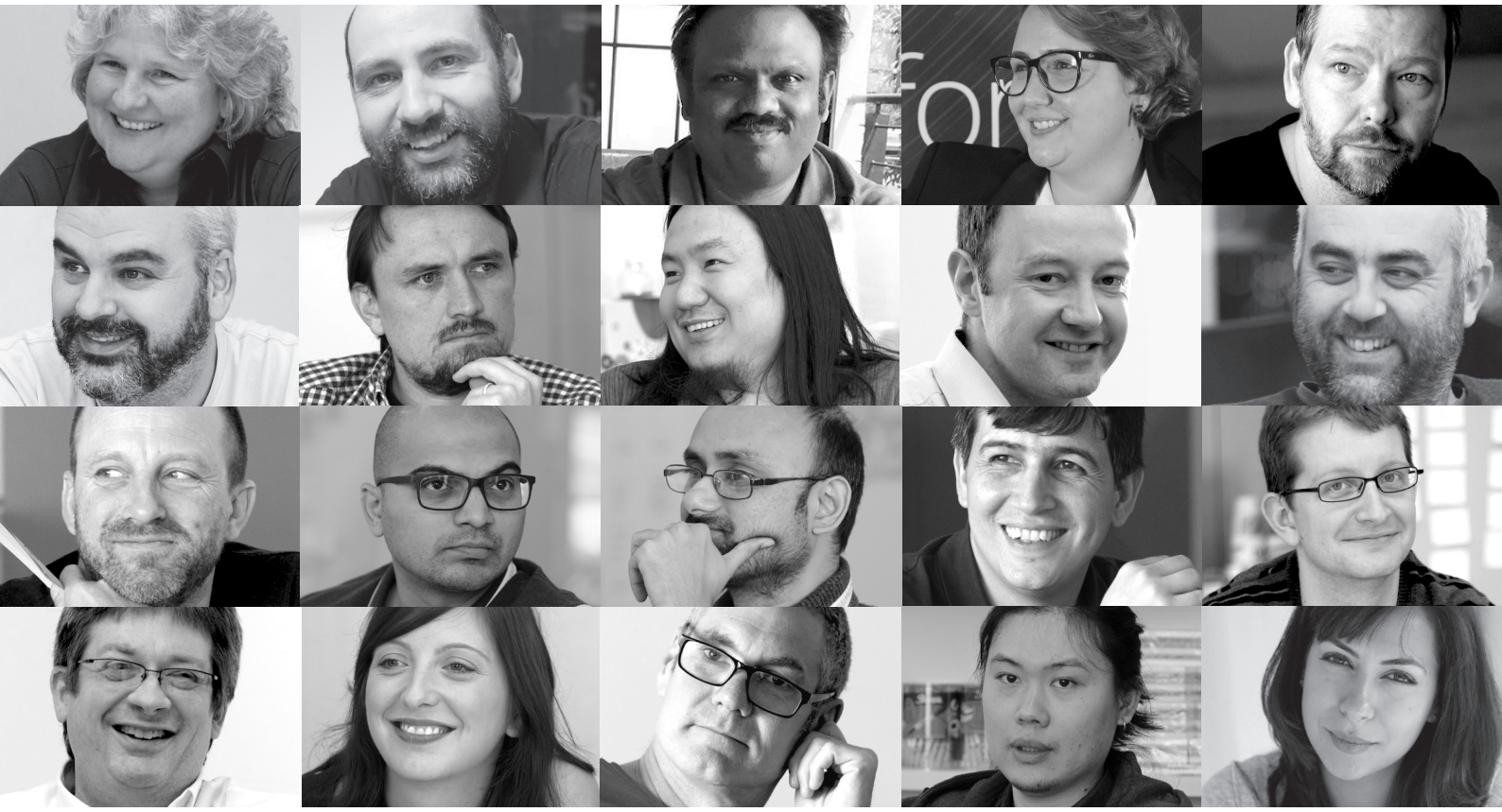
Geleceği şekillendiren
teknoloji ve trendler üzerine
görüşlerimiz

thoughtworks.com/radar

#TWTechRadar

KATKIDA BULUNANLAR

Teknoloji Radarı aşağıdaki ThoughtWorks Teknoloji Danışma Kurulu tarafından hazırlanmıştır:



Rebecca Parsons (CTO) | Martin Fowler (Chief Scientist) | Bharani Subramaniam | Camilla Crispim | Erik Doernenburg
Evan Bottcher | Fausto de la Torre | Hao Xu | Ian Cartwright | James Lewis
Jonny LeRoy | Ketan Padegaonkar | Lakshminarasimhan Sudarshan | Marco Valtas | Mike Mason
Neal Ford | Rachel Laycock | Scott Shaw | Shangqi Liu | Zhamak Dehghani

YENİLİKLER

Bu sayıda öne çıkan konular şunlar:

ÇİN'İN YÜKSELİŞİNDE AÇIK KAYNAK

Sular yükseliyor. Hem tutum hem de politika değişikliklerinden dolayı Alibaba ve Baidu gibi büyük Çinli şirketler açık kaynak frameworklerini, araçlarını ve platformlarını hızla serbest bırakıyorlar. Ekonomik olarak genişledikçe, yazılım ekosistemlerinin büyümesi hızlanmaktadır.

Devasa Çin pazarı ve hızla artan yazılım projesi sayısı göz önüne alındığında, GitHub ve diğer açık kaynaklı sitelerde görünen açık kaynak projelerinin sayısı ve kalitesi artması bekleniyor. Neden Çinli şirketler bu kadar çok varlığı açık kaynak yapsınlar ki? Silikon Vadisi gibi sıcak yazılım pazarlarında olduğu gibi, geliştiriciler için rekabet çok sıkı ve ödenen ücretler artıyor.

Diğer zeki geliştiricilerle birlikte en son açık kaynak projeler üzerinde çalışma ihtimali evrensel bir teşvik unsuru. Büyük açık kaynaklı buluşların, önce çince sonra ingilizce yazılan README dosyalarının trendinin devam etmesini bekliyoruz.

KUBERNETES, KONTEYNER ORKESTRASYONUNDA TEK SEÇİM

Kubernetes ve onun birçok diğer proje üzerindeki baskın varlığı etrafında birçok Radar girdisi oluşturdu. Yazılım geliştirme ekosisteminin, ölçekleme ve konteynerları çalıştırma ile ilgili yaygın problemleri çözmek için Kubernetes ve ilgili araçları kullandığı görülüyor.

GKE, Kops ve Sonobuoy gibi Radar maddeleri, Kubernetes'in benimsenmesi ve çalıştırılmasında ki kullanıcı deneyimini iyileştiren, kullanıma hazır platform servislerini ve araçlarını tanıtmaktadır. Gerçekten de, tek bir planlama birimi olarak birden çok konteyneri çalıştırma kabiliyeti, uç nokta güvenliği için servis kafesi ve servis ağını kolayca yapabilmeyi sağlıyor.

Kubernetes, konteynerler için varsayılan işletim sistemi haline geldi: birçok bulut sağlayıcısı, açık ve modüler mimarisinden yararlanarak Kubernetes'i benimsiyor ve çalıştırıyor; diğer araçlar ise iş yükleri, kümeler, yapılandırma ve depolama gibi soyutlamalara erişmek için açık API'lerinden yararlanıyor.

Kubernetes'i bir ekosistem olarak kullanan daha fazla ürün görüyoruz, bu da onu mikroservisler ve konteynerlerden sonraki yeni soyutlama katmanı haline getiriyor. Bu, geliştiricilerin, dağıtık sistemlerin doğal karmaşıklığına rağmen modern mimari stillerini başarıyla kullanabileceklerinin bir başka göstergesi.

YENİ STANDART OLARAK BULUT

Bu baskıda, yaygın olarak konuştuğumuz diğer bir konu ise belirgin bir şekilde "bulutlu" olduğu. Bulut sağlayıcıları daha yetenekli ve geniş özelliklere ulaştığında, halka açık bulut modeli pek çok kuruluş için yeni varsayılan model haline geliyor.

Birçok şirketin yeni projelere başlarken sorduğu "Neden bulutta?" sorusunun yerini artık "Neden bulutta değil?" sorusu alıyor. Elbette, bazı yazılım türleri hala kurum içi sistemleri zorunlu tutsa da, fiyatlar düştükçe ve kapasiteler arttıkça, bulut-bazlı gelişim gittikçe daha uygun hale geliyor.

Temel özellikler, büyük bulut çözüm sağlayıcıları arasında benzer olsa da, her biri aynı zamanda spesifik çözüm türlerinde kendilerini ayrıştırmak için benzersiz teklifler de sunuyor. Bu nedenle, şirketlerin, Bulut'ta çeşitlilik aracılığıyla çeşitli sağlayıcılar arasından müşterilerinin ihtiyaçlarına en uygun platformu seçtiklerini görmekteyiz.

BLOK ZİNCİR'E OLAN GÜVEN ARTIK DAHA DENGELİ DAĞILIYOR

Piyasadaki kripto para birimlerini çevreleyen kaosa rağmen, müşterilerimizin çoğu, dağıtık defterler ve akıllı kontratlar için blok zincir çözümlerinden faydalanmanın yollarını bulmaktadır. Techradar girişleri gösteriyor ki, akıllı kontratları uygulamak için farklı tekniklerini ve programlama dillerini kullanan blok zincir çözümlerinin kullanımı olgunlaşıyor.

Blok zincir eski bir problem olan dağıtık güven, paylaşılan ve silinmez defterlerin uygulama sorununu çözüyor. Günümüzde şirketler, kullanıcılarının blok zincir içerisinde kullanılan temel mekanizmaya olan güvenini arttırmaktadır. Birçok sanayi farklı dağıtık güven problemlerine sahip; blok zincir çözümlerinin bunları çözmenin yollarını bulmaya devam etmesini bekliyoruz.

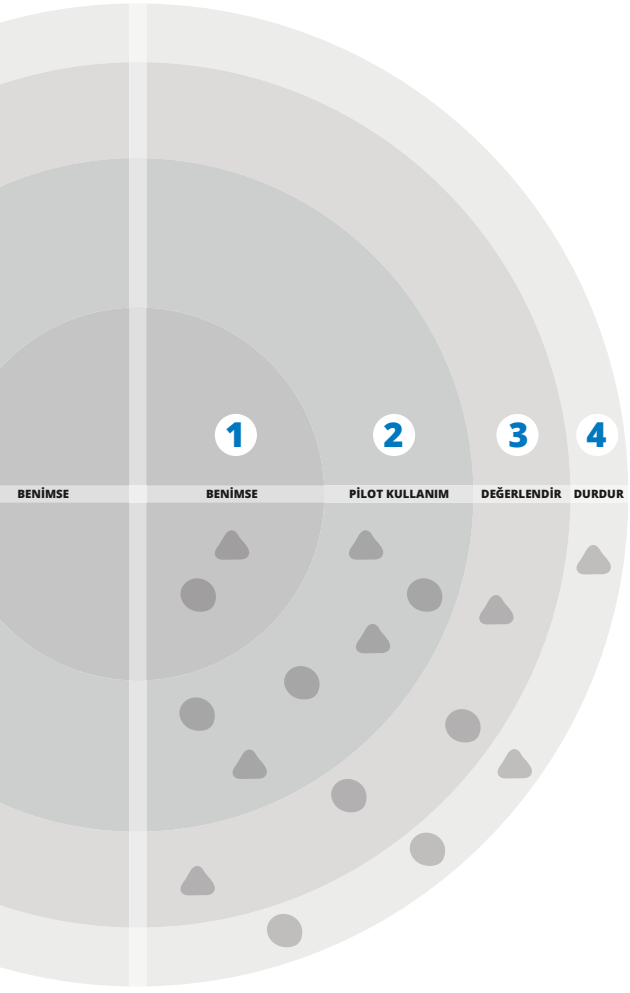
RADAR HAKKINDA

ThoughtWorks çalışanları için teknoloji bir tutkudur. Teknoloji üretiyoruz, araştırmalar ve deneyler yapıyoruz, teknolojiyi açık kaynak olarak sunuyoruz, teknoloji hakkında yazılar yazıyoruz ve herkes için teknolojiyi geliştirmek amacıyla sürekli çalışıyoruz. Misyonumuz, yazılımda mükemmeliyeti savunmak ve BT alanında devrim yapmaktır. Bu misyon doğrultusunda ThoughtWorks Teknoloji Radarı'nı çıkarıp paylaşıyoruz. Radarı, ThoughtWorks'teki üst düzey teknoloji liderlerinden oluşan ThoughtWorks Teknoloji Danışma Kurulu hazırlıyor. Kurul düzenli toplantılar yaparak, ThoughtWorks'ün küresel teknoloji stratejisini ve sektörümüzü ciddi olarak etkileyen teknoloji trendlerini tartışıyor.

Radar, Teknoloji Danışma Kurulunun bu tartışmalarını CTO'lardan geliştiricilere kadar uzanan geniş bir paydaş yelpazesine hitap eden bir formatta sunuyor. İçeriğin az ve öz olması amaçlanıyor.

Sizi bu teknolojileri daha detaylı bir şekilde incelemeye davet ediyoruz. Esas olarak grafiksel bir yayın olan Radar, konuları maddeleri teknikler, araçlar, platformlar ve diller ve framework'ler bazında gruplandırıyor. Radarın birden fazla çeyrek dairede yer alabilecek maddeleri için en uygun göründükleri çeyrek daireyi seçiyoruz. Bu maddeleri ayrıca şu anda onlar hakkındaki tavrımızı yansıtan dört halka halinde gruplandırıyoruz.

Radar ile ilgili daha fazla bilgi için:
thoughtworks.com/radar/faq



BİR BAKIŞTA RADAR

1 BENİMSE

Sektörün bu teknolojileri kullanması gerektiğine kesinlikle inanıyoruz, biz de yeri geldikçe kendi projelerimizde de kullanıyoruz.

2 PİLOT KULLANIM

Takip etmeye değer. Bu maddeleri kullanma kapasitesine nasıl sahip olabileceğinizi anlamak önemlidir. Kurumlar bu teknolojiyi, yalnızca riski kaldırabilecek projelerde bir deneme süreci olarak uygulamalıdır.

3 DEĞERLENDİR

Kurumunuzu nasıl etkileyeceğini anlamak amacıyla takip etmekte fayda var.

4 DURDUR

Temkinli adım atın.

YENİ VEYA DEĞİŞTİRİLMİŞ

DEĞİŞİKLİK YOK

Yeni veya son radardan beri büyük değişim geçiren maddeler üçgen olarak, aynı kalan maddeler ise yuvarlak olarak verilir.

Radarımız ileriye dönüktür. Yeni maddelere yer açmak için son zamanlarda hareket olmayan maddeleri kaldırıyoruz. Bu onların değersiz olduğu anlamına gelmiyor.

THE RADAR

TEKNİKLER

BENİMSE
1. Mimari Tasarım Kayıtları

PİLOT KULLANIM
2. Organizasyon içi platformlara ürün bakış açısı *YENİ*
3. Mimari uygunluk fonksiyonu *YENİ*
4. Otonom baloncuk patterni *YENİ*
5. Kaos Mühendisliği *YENİ*
6. Gizli bilgi yönetimini kaynak koddan ayırmak
7. DesignOps *YENİ*
8. Eski sistemi bir kutuya kapatmak
9. Mikro önyüzler
10. Kod olarak altyapı için hatlar *YENİ*
11. Sunucusuz Mimari
12. TDD ile konteyerlar *YENİ*

DEĞERLENDİR
13. Algoritmik IT Operasyonları *YENİ*
14. Dağıtılmış uygulamalar için Ethereum *YENİ*
15. Bilgi kaynağı olarak olay akışı *YENİ*
16. Platform Mühendisliği Ürün Ekipleri
17. Polycloud - Bulut'ta çeşitlilik *YENİ*
18. Servis ağı *YENİ*
19. Uçtan uca güvenlik için servis kafesleri *YENİ*
20. Güvenliğin üç temel ilkesi *YENİ*

DURDUR
21. Tüm takımlar için tek bir CI kurulumu
22. CI tiyatrosu
23. Kurum genelini kapsayan entegrasyon test ortamları
24. Kafka ile ESB kalıplarının tekrar yaratılması *YENİ*
25. Spec tabanlı kod üretmek

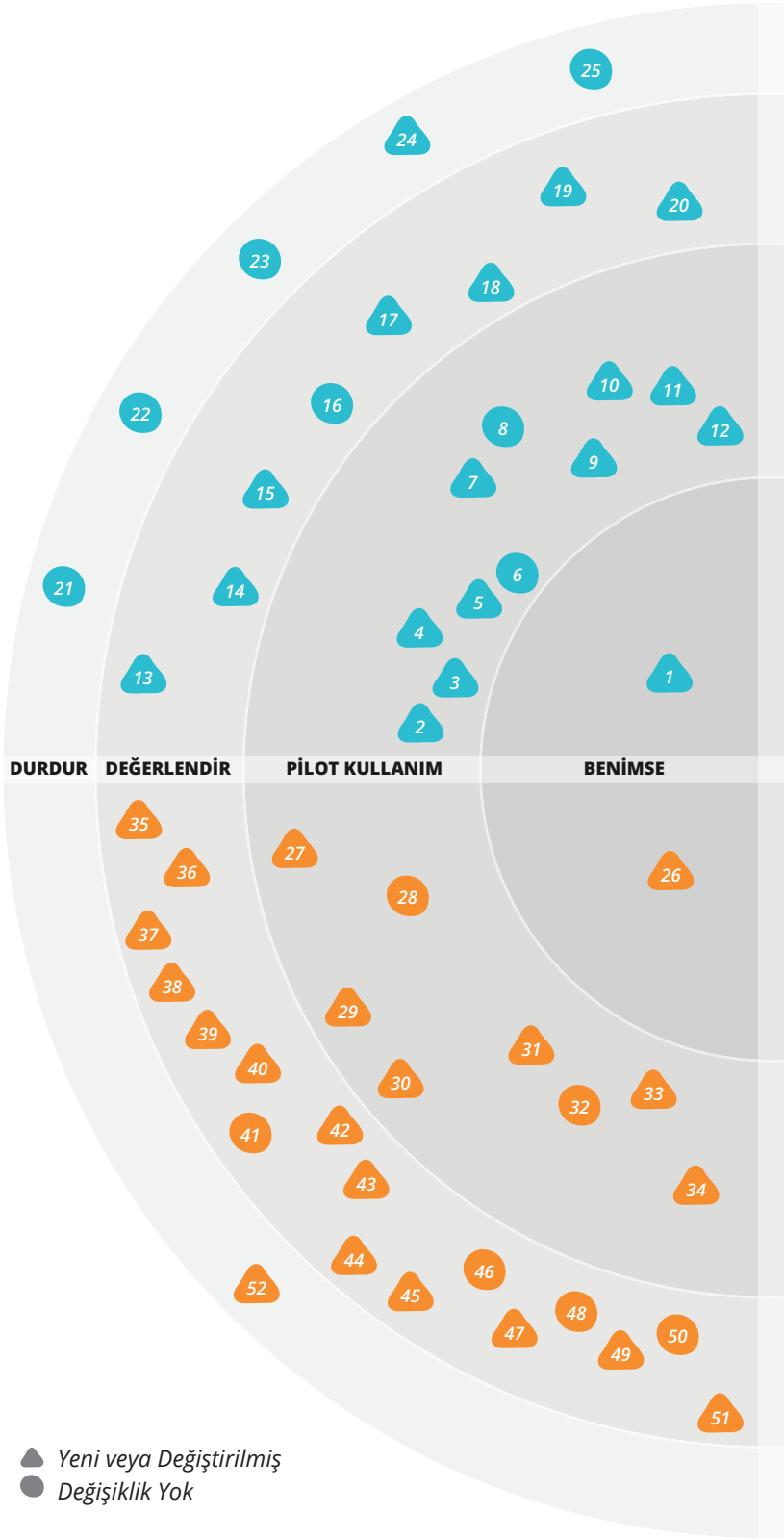
PLATFORMLAR

BENİMSE
26. Kubernetes

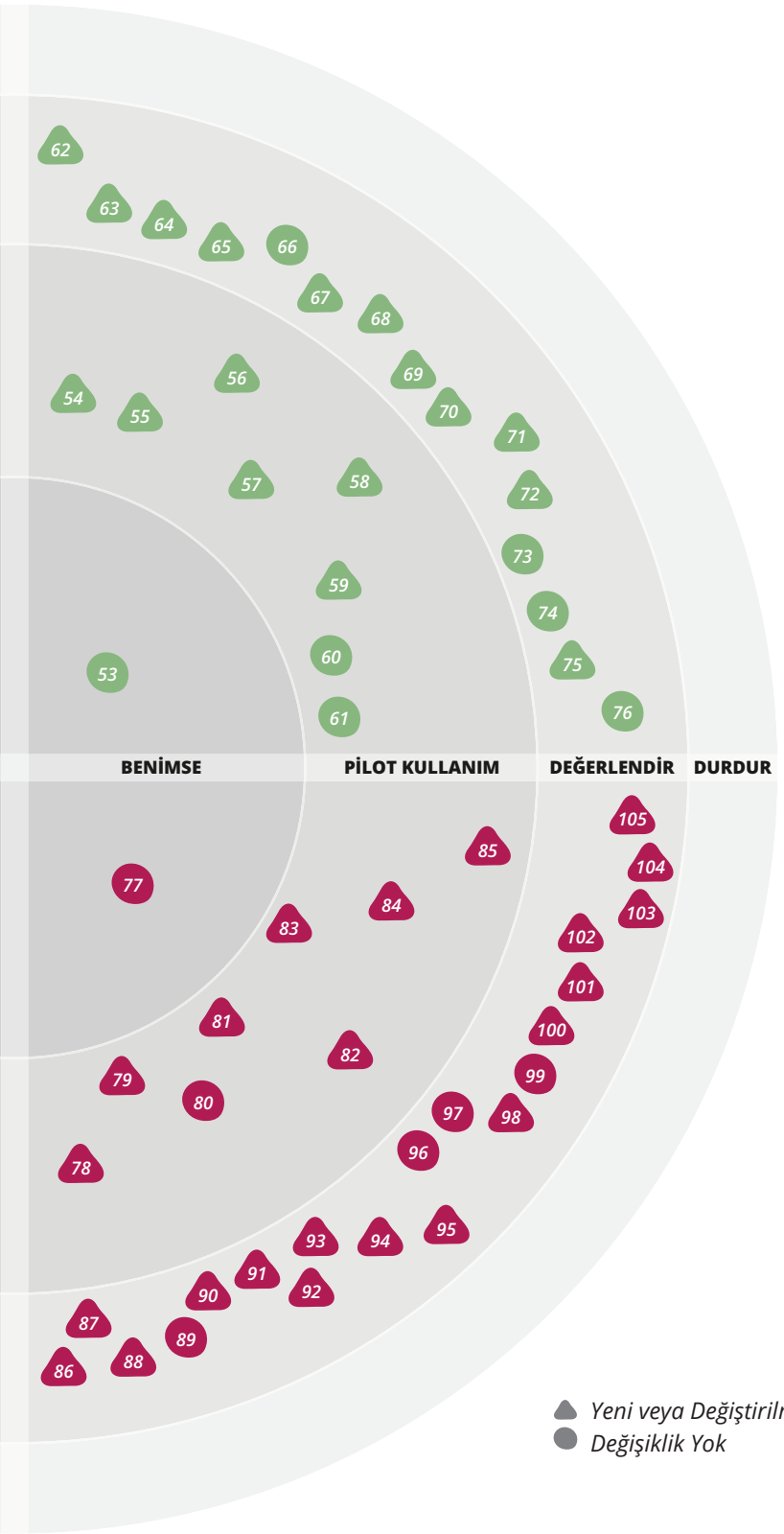
PİLOT KULLANIM
27. .NET Core
28. AWS Device Farm
29. Flood IO *YENİ*
30. Google Cloud Platform *YENİ*
31. Keycloak
32. OpenTracing
33. Oyun ötesinde Unity
34. WeChat *YENİ*

DEĞERLENDİR
35. Azure Service Fabric *YENİ*
36. Cloud Spanner *YENİ*
37. Corda *YENİ*
38. Cosmos DB *YENİ*
39. DialogFlow
40. GKE *YENİ*
41. Hyperledger
42. Kafka Streams
43. Language Server Protocol *YENİ*
44. LoRaWAN *YENİ*
45. MapD *YENİ*
46. Mosquitto
47. Netlify *YENİ*
48. PlatformIO
49. TensorFlow Serving *YENİ*
50. Ses Platformları
51. Windows Konteynerları *YENİ*

DURDUR
52. Aşırıya Kaçmış API Ağ geçidi



THE RADAR



ARAÇLAR

BENİMSE
53. fastlane

PİLOT KULLANIM
54. Buildkite *YENİ*
55. CircleCI *YENİ*
56. gopass *YENİ*
57. Önyüz testleri için Headless Chrome *YENİ*
58. jsoniter *YENİ*
59. Prometheus
60. Scikit-learn
61. Serverless Framework

DEĞERLENDİR
62. Apex *YENİ*
63. assertj-swagger *YENİ*
64. Cypress *YENİ*
65. Flow *YENİ*
66. InSpec
67. Jupyter *YENİ*
68. Kong API Gateway *YENİ*
69. kops *YENİ*
70. Lighthouse *YENİ*
71. Rendertron *YENİ*
72. Sonobuoy *YENİ*
73. spaCy
74. Spinnaker
75. Spring Cloud Contract *YENİ*
76. Yarn

DURDUR

DİLLER VE FRAMEWORK'LER

BENİMSE
77. Python 3

PİLOT KULLANIM
78. Angular
79. AssertJ *YENİ*
80. Avro
81. CSS Grid Yerleşimi *YENİ*
82. CSS Modülleri *YENİ*
83. Jest *YENİ*
84. Kotlin
85. Spring Cloud

DEĞERLENDİR
86. Android Mimari Bileşenleri *YENİ*
87. ARKit/ARCore *YENİ*
88. Atlas ve BeeHive *YENİ*
89. Caffe
90. Clara rules *YENİ*
91. JS'de CSS *YENİ*
92. Digdag *YENİ*
93. Druid *YENİ*
94. ECharts *YENİ*
95. Gobot *YENİ*
96. Instana
97. Keras
98. LeakCanary *YENİ*
99. PostCSS
100. PyTorch *YENİ*
101. single-spa *YENİ*
102. Solidity *YENİ*
103. TensorFlow Mobil *YENİ*
104. Truffle *YENİ*
105. Weex *YENİ*

DURDUR

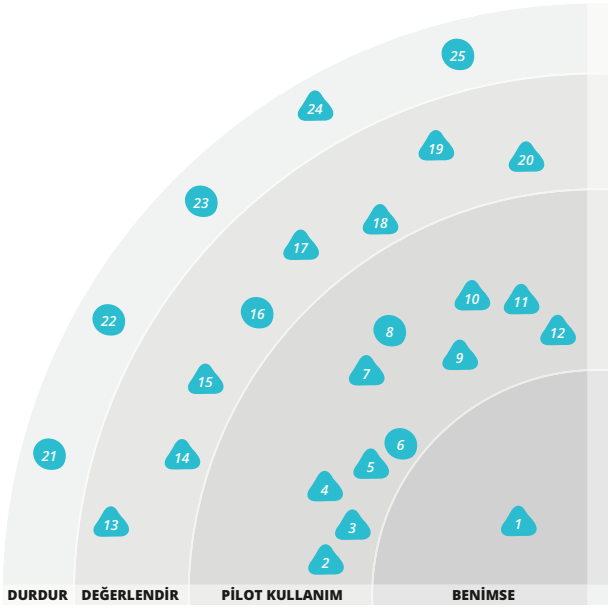
TEKNİKLER

BENİMSE
1. Mimari Tasarım Kayıtları

PİLOT KULLANIM
2. Organizasyon içi platformlara ürün bakış açısı *YENİ*
3. Mimari uygunluk fonksiyonu *YENİ*
4. Otonom baloncuk patterni *YENİ*
5. Kaos Mühendisliği *YENİ*
6. Gizli bilgi yönetimini kaynak koddan ayırmak
7. DesignOps *YENİ*
8. Eski sistemi bir kutuya kapatmak
9. Mikro önyüzler
10. Kod olarak altyapı için hatlar *YENİ*
11. Sunucusuz Mimari
12. TDD ile konteynerlar *YENİ*

DEĞERLENDİR
13. Algoritmik IT Operasyonları *YENİ*
14. Dağıtılmış uygulamalar için Ethereum *YENİ*
15. Bilgi kaynağı olarak olay akışı *YENİ*
16. Platform Mühendisliği Ürün Ekipleri
17. Polycloud - Bulut'ta çeşitlilik *YENİ*
18. Servis ağı *YENİ*
19. Uçtan uca güvenlik için servis kafesleri *YENİ*
20. Güvenliğin üç temel ilkesi *YENİ*

DURDUR
21. Tüm takımlar için tek bir CI kurulumu
22. CI tiyatrosu
23. Kurum genelini kapsayan entegrasyon test ortamları
24. Kafka ile ESB kalıplarının tekrar yaratılması *YENİ*
25. Spec tabanlı kod üretmek



Belgeleme işlemlerinin (dökümantasyonun) bir çoğu okunabilir kod ve testler ile değiştirilebilir. Evrimsel mimari dünyasında ise, belli başlı tasarım kararlarının kaydedilmesi gelecekteki takım üyelerini ve harici kontroller için önem taşımaktadır. **MİMARİ TASARIM KAYITLARI**; bağlamları ve sonuçları ile birlikte önemli mimari kararları kayıt altına almak için kullanılan bir tekniktir. Bu detayların, wiki ya da bir web sitesi yerine kaynak kontrol sistemlerinde saklanmasını öneriyoruz. Böylece bu kayıtlar kodun değişimi ile eş zamanlı olarak tutulabilir. Birçok projede, bu tekniği kullanmamak için bir sebep görmüyoruz.

Son 12 aydır dijital platformlara karşı ilginin katlanarak arttığını gözlemliyoruz. Efektif ve hızlı bir şekilde dijital çözümler devreye almak isteyen şirketler, kendi çözümlerini inşa edip operasyonunu yapabilecek takımlar oluşturuyorlar. Bu takımların ihtiyacı olan API'lere erişimi, araçları, bilgiyi ve desteği, self-servis olarak alabilmeleri için de organizasyon içi platformlar yaratma yoluna gidiyorlar. Bu tip platformlardan ancak onlara dışarıya sunulan bir ürün ile aynı ciddiyet ile yaklaşılırsa en etkin biçimde faydalanılabileceğine inanıyoruz. **ORGANİZASYON İÇİ PLATFORMLARA ÜRÜN BAKIŞ AÇISI** getirebilmek için kullanıcılarla (bkz: geliştiriciler) empati kurmak ve tasarımda onlarla işbirliği yapmak gerekir. Platform ürün yöneticileri yol haritaları belirler ve platformun kurum için değer ürettiğinden ve geliştiricilerin günden güne deneyimini iyileştirdiğinden emin olur. Bazı ürün yöneticilerinin şirket içi platform için marka yönetimi yaptıklarını ve bununla beraber yazılımcı arkadaşlarına pazarladıklarını da gözlemliyoruz. Platform ürün yöneticilerinin bunun dışında kullanım metrikleri toplamaları ve sürekli bir şekilde platformu iyileştirmeleri gerekir. Organizasyon içi platforma ürün gözüyle bakmak şirket içinde canlı bir ekosistem oluşturur ve kurumun sıradan ve az kullanılan bir servis odaklı mimari üretmesinin önüne geçer.

Uygunluk fonksiyonu, evrimsel algoritmalarda bahsedildiği gibi, eldeki bir tasarım çözümünün belirlenen hedeflere ne kadar yakın olduğunu özetler.

(Mimari uygunluk fonksiyonu)

Uygunluk fonksiyonu, evrimsel algoritmalarda bahsedildiği gibi, eldeki bir tasarım çözümünün belirlenen hedeflere ne kadar yakın olduğunu özetler. Evrimsel bir algoritma tasarlarken 'daha iyi' bir çözüm aranır ve uygunluk fonksiyonu ise bu bağlamda 'daha iyi'nin tanımını yapar. **MİMARİ UYGUNLUK FONKSİYONU** Evrilen Mimariler Tasarlamak (Building Evoluntionary Architectures) kitabında tanımlandığı şekliyle varolan mimarinin özelliklerini tarafsız bir şekilde değerlendirecek bir yapı sunar. Bu yapı, halihazırda varolan, birim testleri, metrikler ve izleme sistemlerini kapsayabilir. Biz evrilen mimariler yaratabilmek için mimarların otomatik ve devamlı bir şekilde mimari kararları doğrulayabileceğine, koruyabileceğine ve iletebileceklerine inanıyoruz.

CI ve CD araçları, sunucu yapılandırmasını test etmek, sunucu imaj oluşturma, ortam provizyonu ve ortamların entegrasyonu için kullanılır.

(Kod olarak altyapı için hatlar)

Birlikte çalıştığımız birçok kurumun, modern mühendislik yaklaşımlarıyla yeni ürünler geliştirirken, bir yandan da varolan eski sistemleri işletip, korumaya çalışırken zorlandıklarını gözlemliyoruz. Böyle durumlarda başarılı olduğunu gözlemlediğimiz nispeten eski bir strateji de Eric Evans'ın **OTONOM BALONCUK PATTERNI** (Autonomous Bubble Pattern). Bu yaklaşım, yeni uygulamayı eski uygulamaların karışıklıklarından ayırmak ve korumak için tamamen ayrı ve yeni bir durum yaratmayı öngörür. Ancak bu iyi bildiğimiz anticorruption layer şablonundan bir adım daha ötededir. Otonom baloncuk şablonunda yeni uygulamanın içinde bulunduğu baloncuk kendi verisinin tamamen sahibidir ve bu veri eski sistemin verisiyle asenkron bir şekilde eşlenir. Bu baloncuğun sınırlarını korumak ve eski ve yeni olmak üzere iki dünyanın verilerini tutarlı halde tutmak için biraz emek gerekir. Ama sonucunda getirdiği özerk mimari parçalar ve geliştirme sırasında gittikçe azalan sürtüşmeler daha modern bir mimari kurabilmek için önemli bir adımdır.

Radar'ın önceki sürümlerinde, çalışan bir sistemin sunucularını rasgele devre dışı bırakıp sonuçlarını ölçerek üretimdeki kesintilerle nasıl başa çıktığını test etmek için Netflix'in Chaos Monkey aracını kullanmayı konuştuk. **KAOS MÜHENDİSLİĞİ**, bu tekniğin daha geniş uygulama alanı bulduğu yeni bir terimdir. Çalışan dağıtık sistemler üzerinde deneyler yaparak, bu sistemlerin çalkantılı koşullar altında beklendiği gibi çalıştığına dair güven inşa edebiliyoruz. Kaos Mühendisliği Prensipleri web sitesi, bu tekniği daha iyi anlamak için doğru bir başlangıç olur.

DESIGNOPS, DevOps hareketinden de ilham alarak ortaya çıkmış, kültürel bir dönüşüm hareketidir.Ve bir organizasyonda kaliteden ve hizmet bütünlüğünden ödün vermeksizin ürünlerin sürekli bir şekilde yeniden tasarlanabilmesine ve takımların özerk kalabilmesine izin veren bir dizi pratikten oluşur. DesignOps, yeni UI konseptleri ve çeşitlemelerini yaratmaya imkan verecek bir tasarım altyapısı oluşturmayı savunur ve bunla hızlı ve güvenilir bir feedback döngüsü yaratmayı hedefler. Sıkı iş birliğini öne çıkaran Storybook gibi araçlar ile önden yapılması gereken analiz ve spesifikasyon el değişimleriyle gelen yükü minimuma indirir. DesignOps ile tasarım, artık sadece uzmanlarına ait bir iş olmaktan çıkıp herkesin işi olmaya başlıyor.

Takımların servislerini birbirinden bağımsız bir şekilde kurmalarına ve bakım yapmalarına izin veren microservice mimarisinin önemli faydalarını gördük.Ancak ne yazık ki birçok takımın, arkada hizmet veren servislerinin önünde, tek parça ve giderek büyüyen önyüz monolit uygulamaları inşa ettiğini de gördük.Bizim tercih ettiğimiz (ve çalıştığını gördüğümüz) yaklaşım, tarayıcı tabanlı kodu **MİKRO ÖNYÜZLERE** ayırmaktır.Bu yaklaşımda, web uygulaması özelliklerine ayrılır ve her özellik arayüz'den backend servislerine kadar farklı bir ekip tarafından geliştirilir. Bu, her özelliğin diğer özelliklerden bağımsız olarak geliştirilmesini, test edilmesini ve konuşlandırılmasını sağlar. Web uygulaması özelliklerini (bazen sayfalar olarak, bazen bileşenler olarak) birleştirerek birbirine bağlı kullanıcı deneyimine dönüştürmek için birden fazla teknik mevcuttur.

Yazılımda sürüm sürecini düzenlemek için sürekli teslimat hatlarının (Continuous Delivery Pipeline-CDP) kullanılması temel konsept haline geldi. Bununla birlikte, altyapı kodundaki değişikliklerin otomatik olarak test edilmesi yaygın şekilde anlaşılmış değil. Sürekli entegrasyon (Continuous Integration-CI) ve sürekli teslimat (Continuous Delivery-CD) araçları, sunucu yapılandırmasını test etmek (örneğin Chef tarifleri, Puppet modülleri, Ansible scriptleri), sunucu imaj oluşturma (ör. Packer), ortam provizyonu (ör. Terraform, CloudFormation) ve ortamların entegrasyonu. **KOD OLARAK ALTYAPI IÇIN HATLAR** kullanılması, geliştirme ve test için kullanılan ortamlar da dahil olmak üzere, yapılan değişiklikler operasyonel ortamlara uygulanmadan önce hatalarının bulunabilmesini sağlar. Ayrıca, altyapı araçlarının, bireysel makinalardan çalıştırılmasının aksine CI / CD ajanları üzerinden tutarlı olarak çalıştırılmasını sağlayan bir yöntem de sunar. Bununla birlikte, bazı zorluklar devam etmektedir, konteynerların ve sanal makinelerin ayakta tutulmasıyla

ilişkili daha uzun geri bildirim döngüleri gibi. Yine de, bunu değerli bir teknik olarak görüyoruz.

SUNUCUSUZ MİMARİ kullanımı, devreye almak için sunduğu geniş seçenekler ile uygulamalarını bulut üzerinde devreye alan kurumlar için hızla kabul gören bir yaklaşım haline gelmiştir. Geleneksel kurumlar dahi sunucusuz teknolojileri parçalı da olsa kullanıyorlar. Bu alanda uygun kullanım desenleri hala ortaya çıkarken, tartışmaların çoğu, bir servis olarak fonksiyon üzerinde toplanmaktadır. Örneğin AWS Lambda, Google Cloud Functions, Azure Functions, sunucusuz fonksiyon kullanımı geleneksel yönlemlerde harcadığımız sunucu hazırlığı, işletim sistemi yapılandırması ve orkestrasyonu gibi değer yaratmayan eforu net bir şekilde ortadan kaldırmaktadır. Buna rağmen, sunucusuz fonksiyon kullanımı her gereksinim için uygun değildir. Bu gibi özel gereksinimler için konteyner veya sunucu kurulumlarına hazır olmalısınız. Bu arada, servis olarak Backend gibi sunucusuz mimarinin diğer bileşenleri neredeyse varsayılan seçim haline gelmiştir.

Pek çok geliştirme ekibi, faydaları nedeniyle yazılım geliştirirken test odaklı geliştirme (TDD) yöntemini uyguluyorlar. Bunun yanında bazı ekipler de yazılımlarını paketlemek ve dağıtmak için konteynerları kullanmaktalar. Konteyner paketlerini oluşturmak için otomatik çalıştırılan script dosyalarını kullanmak kabul edilmiş yaygın bir uygulamadır. Şimdiye kadar gördüğümüz kadarı ile çok az takım bu iki pratiği birleştirerek konteyner scriptlerini test odaklı yöntem ile yazıyorlar. Serverspec ve Goss gibi frameworkler sayesinde, hem izole edilmiş hem de orkestra edilen konteynerler için burada hedeflenen işlevselliği, kısa geribildirim döngüleri ile elde edebilirsiniz. Bu, yazılım geliştirmede savunduğumuz ilkelerin aynısının **TDD İLE KONTEYNERLERDE** kullanılabileceği anlamına gelmektedir ve bu konuda elde ettiğimiz ilk tecrübeler oldukça olumlu.

BT operasyonları ile toplanan verilerin miktarı yıllardır artmaktadır. Örneğin, mikroservis eğilimi, daha fazla uygulamanın kendi operasyonel verilerini üretmekte olduğu anlamına gelir ve; Splunk, Prometheus veya ELK seti gibi araçlar, operasyonel analizler elde etmek için verilerin depolanması ve ardından işlenmesini kolaylaştırırlar. Bununla birlikte; makina öğrenme araçlarının artan bir şekilde kullanımının yayılması ile, operatörlerin istatistiksel modelleri ve eğitimli sınıflandırma algoritmalarını araç takımlarına dahil etmeleri kaçınılmazdır. Bu algoritmaların yıllardır var olmasına ve servis yönetimlerini otomatikleştirmek için çeşitli girişimlerde de bulunulmasına rağmen, makinelerin ve insanların, servis kesintilerini önceden tanımlamak veya arızaların kaynağını belirlemek için nasıl işbirliği yapabileceğini ancak anlamaya başlıyoruz.Tabi ki **ALGORITMIK IT OPERASYONLARININ** aşırı derecede şişirilme riski olmasına rağmen, makine öğrenme algoritmalarındaki istikrarlı gelişim, yarının veri merkezlerini işletmede insanların rolünü kaçınılmaz olarak değiştirecektir.

Blok zincirler uzun süredir fintek ile ilgili ne varsa, bankacılıktan dijital para birimlerine kadar zincir transparanlığını sağlamak sözkonusu olduğunda her derde deva olarak görülmekte. Daha önceki radarlarımızda "smart contract"lar gibi özelliklerinden dolayı Ethereum'dan bahsetmiştik. Şimdi ise başka alanlarda **DAĞITILMIŞ UYGULAMALAR IÇIN ETHEREUM** ile geliştirmeler yapıldığını gözlemliyoruz. Her ne kadar oldukça yeni bir teknoloji de olsa, gün geçtikçe kriptopara ve bankacılık uygulamalarının ötesinde dağıtık uygulamalar için daha çok kullanıldığını görüyoruz.

Blok zincirler uzun süredir fintek ile ilgili ne varsa, bankacılıktan dijital para birimlerine kadar zincir transparanlığını sağlamak sözkonusu olduğunda her derde deva olarak görülmekte.

(Dağıtılmış uygulamalar için Ethereum)

Apache Kafka olay akışı (event streaming) platformları popülerlik kazanırken, birçokları bu platformları sadece olayları iletmekte kullanılan mesaj kuyruklarının biraz daha gelişmiş versiyonları olarak görmeye devam ediyor. Bu şekilde kullanıldığında bile, olay akışları, geleneksel mesaj kuyruklamasına göre daha fazla fayda sağlamakta. Yine de, insanların özellikle Kafka gibi platformlar aracılığıyla değiştirilemez olay kayıtlarını kalıcı bir şekilde saklayarak bir **BİLGİ KAYNAĞI OLARAK OLAY AKIŞINI** nasıl gerçekleştirdikleri daha çok ilgimizi çekiyor. Bu çerçevede Event Sourcing tasarımına sahip bir servis olayları tutmak için Kafka'yı kullanabilir ve daha sonrasında bu olay kayıtları diğer servislerin tüketebilmesi için ortada olur. Böylelikle, bu teknik sayesinde entegrasyon ve yerel saklama yaparken aynı eforun tekrar tekrar sarfedilmesinin önlenebileceğini düşünüyoruz.

Büyük bulut sağlayıcıları (Amazon, Microsoft ve Google), ürünlerini sadece bazı yan yetkinliklerde farklılaştırırken temel bulut yetkinlikleri üzerinde eşgüdüm sağlamak için agresif bir yarışa girdiler. Bu yaklaşım ile bazı organizasyonlar tüm bulut yetkinliklerini tek bir

sağlayıcıda aramaktansa, çoklu bulut (**POLYCLOUD**) stratejisini benimsemekteler ve bu sayede farklı sağlayıcılara farklı iş yükleri göndererek en optimum çözümleri sağlayabiliyorlar. Örneğin AWS'ye standart servisleri koyarken, Google'ı makina öğrenmesi için, SQLServer kullanan .Net uygulamaları için Azure veya potansiyel olarak Blok zincir çözümleri için Ethereum Consortium kullanıyorlar. Bu yaklaşım, sağlayıcılar arasında taşınabilirliği amaçlayan, maliyeti yüksek ve ortak paydada düşünmeye zorlayan bulut-agnostik stratejiden oldukça farklı. Polylcloud, bunun yerine her bulut sağlayıcısının en iyi sunduğu özellikleri kullanmaya odaklanıyor.

Bir servis ağı, API ağ geçidi veya ESB gibi paylaşılan bir şeye ihtiyaç duymadan, tutarlı servis keşfi, güvenlik, izleme, takip ve hata yönetimi sunabilir.

(Servis Ağı)

Büyük organizasyonlarda, kendi mikro servislerinin sahibi ve işleticisi olacak şekilde bağımsız takımlara doğru bir geçiş görmekteyiz. Fakat, bu kurumlar merkezi bir barındırma altyapısına ihtiyaç duymadan bu hizmetler arasında gerekli tutarlılık ve uyumluluğu nasıl sağlayabilirler? Birlikte verimli bir şekilde çalışma için, otonom mikro servislerin bile bazı kurumsal standartlarla uyumlu olması gerekir. Bir **SERVİS AĞI**, API ağ geçidi veya ESB gibi paylaşılan bir şeye ihtiyaç duymadan, tutarlı servis keşfi, güvenlik, izleme, takip ve hata yönetimi sunabilir. Tipik bir uygulama olarak; her servis ile birlikte, belki de ayrı bir konteynerde konuşlandırılabilecek reverse-proxy uygulamaları örnek gösterilebilir. Bu proxyler, servis kayıtları, kimlik sağlayıcıları, log toplayıcıları vb. gibi yerler ile iletişim kurar. Servislerin birlikte çalışabilirliği ve gözlemlenebilirliği, bu proxy'nin canlı ortamda paylaşımlı kullanılan tek bir süreci ile değil, uygulanma yöntemi olarak paylaşılması ile elde edilir. Bir süredir mikro servis yönetimi için merkeziyetçi olmayan bir yaklaşım önerdik ve bu yaklaşımın kalıcı bir örnek olarak belirmesinden mutluluk duyuyoruz. linkerd ve Istio gibi açık kaynak projeleri, olgunlaşmaya ve servis ağlarının kullanımını daha da kolaylaştırmaya devam edecektir.

Varlıklarını ve yeteneklerini API'ler aracılığıyla ve artan bir saldırı yüzeyi ile hizmete açan Mikroservice mimarisi, "sıfır güven" güvenlik mimarisini uygulamayı zorunlu kılar; "asla güvenme, her zaman doğrula". Bununla birlikte, geliştirilen kodların karmaşıklığı ve birden fazla programlama dilinin kullanıldığı bir ortamda eksik kütüphane ve dil desteği nedeniyle, servisler arasında iletişim için güvenlik denetimlerinin uygulanması genellikle ihmal edilmektedir. Bu karmaşıklığı gidermek için bazı takımlar, güvenlik işlerini, servisin kendisiyle aynı anda çalıştırılan ve servis ile aynı uygulama bilgisine sahip başka bir servise, kafes gibi taşıyıcı bir servise delege ederler. Servis kafesleri, iletişimin ve TLS sonlandırmasının şeffaf şifrelenmesini sağladığı gibi çağıran servisin veya son kullanıcının kimlik doğrulama ve yetkilendirmesi gibi güvenlik özelliklerini uygularlar. Kendi **UÇTAN UCA GÜVENLİK İÇİN SERVİS KAFESLERİ** oluşturmadan önce Istio, Linkerd veya Envoy'a bakmanızı öneririz.

Kurumsal güvenliğe geleneksel yaklaşımlar genellikle birşeyleri kilitlemeyi ve değişim hızını yavaşlatmayı vurgular. Diğer taraftan bir saldırganın bir sistemi tehlikeye atmak için harcadığı zaman ne kadar çok olursa, vereceği potansiyel zararın da o kadar arttığını biliyoruz. Kurumsal güvenliğin üç temel ilkesi -değiştirme, onarma ve yeniden yapılandırma - ile saldırı imkanlarını ortadan kaldırmak için, altyapı otomasyonundan ve sürekli teslimattan yararlanır. Kimlik bilgilerini sürekli değiştirmek, yamaları hazır oldukları gibi en kısa zamanda uygulamak, sistemleri bilinen en son güvenli durumuna hızlıca yeniden yüklemek - bunları birkaç dakika veya saat içinde yapmak- saldırganların başarılı olmasını zorlaştırır. **GÜVENLİĞİN ÜÇ TEMEL İLKESİ**, modern bulut tabanlı mimarilerin ortaya çıkışı ile artık daha da uygulanabilir. Uygulamaların konteynerler halinde derlenip, yüklenip, test edildiği tamamen otomatikleşmiş bir geliştirme hattında, bir güvenlik yamasının gönderilmesi, sadece bu hat üzerinden tek bir tıklama ile giden küçük bir sürümden ibarettir. Tabii ki, en iyi dağıtık sistem pratiklerinde, geliştiricilerin uygulamalarını beklenmedik sunucu kesintilerine karşı dirençli olarak tasarlamaları gerekir. Bu Chaos Monkey'ın sizin geliştirme ortamınıza uygulandığında oluşturacağı etkiye benzer.

Kafka mesajlaşma çözümü olarak çok popüler hale geliyor ve onunla birlikte Kafka Streams, akış mimarilerine olan ilgi patlamasında en ön planda yer almaktadır. Ne yazık ki, Kafka'yı veri ve uygulama platformlarının kalbine yerleştirmeye başladıkça, ürün veya hizmet ekipleriyle birlikte yaşatılmasına izin vermek yerine, Kafka ekosistem bileşenlerini (bağlayıcılar ve akış işlemcileri) merkezileştirerek **KAFKA İLE ESB KALIPLARINI TEKRAR YARATAN** bazı organizasyonlar görüyoruz. Bu bize, merkezi olarak yönetilen bir ESB'ye daha fazla akıl, orkestrasyon ve dönüştürme (transformation) mantığı getiren ve merkezi bir ESB takımına ciddi bir bağımlılık oluşturan, problemli ESB kalıplarını hatırlatıyor. Bu kusurlu modelin daha çok uygulanmasını engellemek için dikkatinizi çekmek istiyoruz.

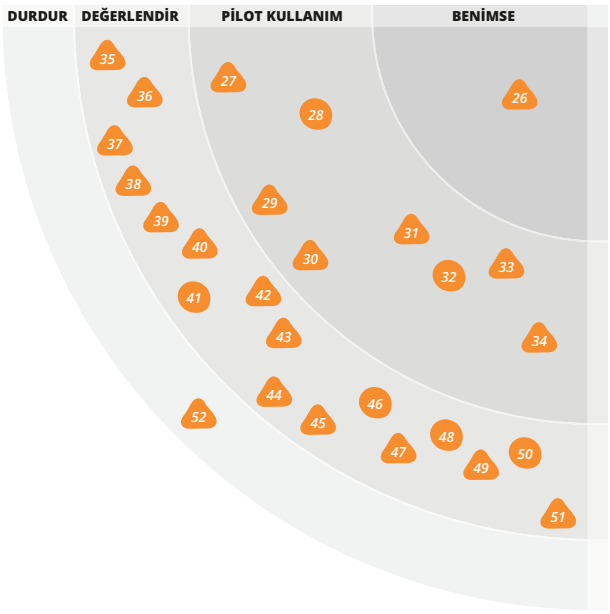
PLATFORMLAR

BENİMSE
26. Kubernetes

PİLOT KULLANIM
27. .NET Core
28. AWS Device Farm
29. Flood IO *YENİ*
30. Google Cloud Platform *YENİ*
31. Keycloak
32. OpenTracing
33. Oyun dışında Unity
34. WeChat *YENİ*

DEĞERLENDİR
35. Azure Service Fabric *YENİ*
36. Cloud Spanner *YENİ*
37. Corda *YENİ*
38. Cosmos DB *YENİ*
39. DialogFlow
40. GKE *YENİ*
41. Hyperledger
42. Kafka Streams
43. Language Server Protocol *YENİ*
44. LoRaWAN *YENİ*
45. MapD *YENİ*
46. Mosquitto
47. Netlify *YENİ*
48. PlatformIO
49. TensorFlow Serving *YENİ*
50. Ses Platformları
51. Windows Konteynerleri *YENİ*

DURDUR
52. Aşırıya Kaçmış API Ağ geçidi



Radar'da son olarak **KUBERNETES** 'den bahsettikten sonra, bir dizi makineye konteyner yükleme durumunda müşterilerimizin çoğunun varsayılan çözümü haline geldi. Alternatifler çok fazla bilgi paylaşımı sağlamadı ve bazı durumlarda müşterilerimiz 'ana motorlarını' Kubernetes'e değiştiriyorlar. Kubernetes, Microsoft'un Azure Konteyner Servisi ve Google Cloud gibi büyük genel bulut platformları için tercih edilen konteyner orkestrasyon platformu haline geldi (GKE şablonuna bakın). Ve hızla büyüyen Kubernetes ekosistemini zenginleştiren pek çok faydalı ürün var. Bununla birlikte, Kubernetes'i soyutlayarak karmaşıklığını gizlemeye çalışan platformlar henüz kendini ispatlamış değiller.

Açık kaynak kodlu çoklu-platform yazılım frameworku olan **.NET CORE**'un daha fazla benimsendiğini görüyoruz. .Net Core ile .NET uygulamalarının geliştirme ve yüklemeleri Windows, macOS ve Linux üzerinde de yapılabiliyor. .NET Standard 2.0 sürümünün .NET platformları arasındaki standart API'lerin sayısının artırması ile birlikte, .NET Core'a geçiş yolu daha da net hale geldi. Bu sürüm ile .NET Core'daki kütüphane desteğiyle ilgili konular daha az sorunlu hale geliyor ve artık Windows dışı platformlarda da üretken geliştirmeye olanak tanıyan, çok iyi çoklu-platform araçları da mevcut. .NET Core servislerini konteyner ortamlarına kolay entegre etmek için de Docker imajları sağlanmış durumda. Projelerimizden aldığımız geribildirimler ve topluluktaki olumlu yönelim, .NET Core'un yaygın kullanıma hazır olduğunu gösteriyor.

Yük testi, Gatling ve Locust gibi araçların olgunlaşmasıyla oldukça kolaylaşmış durumda. Aynı zamanda esnek bulut altyapıları da çok sayıda istemciyi simüle etmeyi mümkün kılıyor. Bu teknolojilerden yararlanarak Flood ve diğer bulut platformlarının ilerlemesini görmekten memnuniyet duyuyoruz. **FLOOD IO**, test komut dosyalarını buluttaki yüzlerce sunucu arasında dağıtmaya ve çalıştırmaya yardımcı olan bir SaaS yük testi hizmetidir. Ekibimiz, mevcut Gatling komut dosyalarını kullanarak performans testini Flood'a taşımayı oldukça kolay buluyor.

GOOGLE CLOUD PLATFORM (GCP), bulunduğu coğrafi bölgeler ve servislerinin olgunluğu açısından geliştikçe, müşteriler dünya genelinde bulut stratejileri için GCP'yi daha ciddi olarak değerlendirmekteler. GCP bazı bölgelerde, başlıca rakibi Amazon Web Services ile fonksiyon açısından başabaş hale geldi. Diğer alanlarda ise, özellikle erişilebilir makine öğrenme platformları, veri mühendisliği araçları ve servis çözümü (GKE) olarak kolay çalışılabilir bir Kubernetes servisi ile kendisini farklılaştırdı. Ekibimiz pratik kullanımda, yazılımcı deneyimi açısından, GCP araçları ve API'lerini için övgüden başka söyleyecek söz de bulamıyor.

Google Cloud Platform, bulunduğu coğrafi bölgeler ve servislerinin olgunluğu açısından gelişmekte ve dünya genelinde bulut stratejileri daha ciddi olarak değerlendirmektedir.

(Google Cloud Platform)

Mikroservislerde veya diğer dağıtık mimarilerde, en yaygın ihtiyaçlardan biri, kimlik doğrulama ve yetkilendirme özellikleri aracılığıyla, servisleri veya API'leri güvenli hale getirmektir. Bu noktada **KEYCLOAK** devreye giriyor. Keycloak, uygulamalara veya mikro servislere az veya hiç kod yazmadan güvenlik eklemeyi kolaylaştıran açık kaynak kodlu bir kimlik ve erişim yönetimi çözümüdür. Tek oturum açmayı (SSO), sosyal medya ile oturum açmayı ve OpenID Connect, OAuth 2.0 ve SAML gibi standart protokolleri destekler. Ekiplerimiz bu aracı şuan kullanıyor ve öngörülebilir bir gelecek için de kullanmaya devam edecek. Fakat kurulum biraz zahmetli. Çünkü, ürünü yapılandırma, hem başlatma sırasında ve hem de API'lar aracılığıyla çalışma zamanında gerçekleştiğinden, kurulumun tekrarlanabilir olmasını sağlamak için script yazmak gereklidir.

Önceki Techradar'larda Unity'nin sunduğu soyutlama, yetkin bir platform olarak sağladığı araçlar ve alternatifi olan Unreal Engine'den daha erişilebilir olması ile birlikte, sanal gerçeklik (Virtual Reality-VR) ve artırılmış gerçeklik (Augmented Reality-AR) uygulama geliştirmesinde, tercih edilen platform olduğunu belirmiştik. IOS için ARKit ve Android için ARCore'un yakın zamanda piyasaya sürülmesiyle, iki ana mobil platformu da artırılmış gerçeklik uygulamaları geliştirmek için güçlü yerel SDK'lara kavuşmuştur. Bir çok takımı, özellikle de oyun geliştirme konusundaki geniş deneyime sahip olmayanları, soyutlama kullanarak **OYUNUN ÖTESİNDE UNITY** kullanmaya davet ediyoruz. Bu, teknolojiye aşina olmayan geliştiricilerin tek bir SDK'ya odaklanabilmesini sağlıyor. Ayrıca yerel SDK'lar tarafından desteklenmeyen çok sayıda Android cihazına çözüm de sunuyor.

Sıklıkla WhatsApp uygulamasının esleniği olarak adından söz ettiren **WECHAT**, Çin'de fiilen iş platformu haline gelmeye basladı. Birçok insanın haberi olmayabilir, ancak WeChat aynı zamanda en popüler online ödeme sistemlerinden birisi. Uygulamanın parçası olan içerik yönetim sistemi ve üyelik yönetim sistemi ile küçük işletmeler artık ticaretlerini tamamıyla WeChat üzerinden yürütüyorlar. Hizmet Hesabı özelliği ile, büyük kurumlar kendi iç sistemlerini çalışanlarına açabiliyorlar. Çinlilerin yüzde 70'inden fazlasının WeChat uygulamasını kullandığı gerçeği, Çin marketine açılmak isteyen işletmeler için WeChat uygulamasını önemli bir konuma getiriyor.

AZURE SERVICE FABRIC, mikro servisler ve konteynerler için geliştirilmiş bir dağıtık sistem platformudur. Kubernetes gibi konteyner orkestrasyonu araçları ile karşılaştırılabilir, ancak aynı zamanda eski tip servisler ile de çalışır. Seçtiğiniz dildeki basit servislerden başlayıp, bir SDK kullanılarak oluşturulan Docker konteynerleri veya servislere kadar, şaşırtıcı şekilde farklı yollarla kullanılabilir. Birkaç yıl önce piyasaya çıktığından beri, Linux konteyner desteği de dahil olmak üzere birçok özellik eklendi. Kubernetes, konteyner düzenleme araçlarının meşhur çocuğu olmakla birlikte, .NET Uygulamaları için varsayılan seçenek Service Fabric'tir. Bunu ThoughtWorks'teki birkaç projemizde kullanıyoruz ve şimdiye kadar gördüğümüz kadarı ile oldukça beğendik.

CLOUD SPANNER yüksek erişilebilirliği ve güçlü tutarlılığı, düşük oranda gecikmeyi ödün vermeden sağlayan ilişkisel bir veri tabanı servisidir. Google, uzunca bir süredir Spanner olarak isimlendirilen, küresel ölçekte dağıtık bir veri tabanı üzerinde çalışıyordu. Yakın zamanda da bu servisi dış dünyaya Cloud Spanner olarak açtı. Cloud Spanner'ı kullanarak, veri tabanınızı, veri tutarlılığı konusunda endişelenmeden, birden bine kadar dünya genelindeki düğümler üzerinde ölçekleyebilirsiniz. Yüksek erişilebilirliğe sahip ve dağıtık saat TrueTime 'ın yardımıyla, Cloud Spanner verinin okunması ve anlık görüntülemeler için güçlü tutarlılık sağlar. Cloud Spanner'dan veri okumak için standart SQL'i kullanabilirsiniz. Ancak yazma işlemleri için, sağlanan RPC ara yüzlerini kullanmak zorundasınız. Her servis küresel ölçekte dağıtık bir veri tabanına ihtiyaç duymasa

da, Cloud Spanner'ın genel erişilebilirliği, veri tabanlarına mevcut bakış açımızı büyük ölçüde değiştiriyor. Aynı zamanda dizaynı da CockroachDB gibi bazi açık kaynaklı ürünleri etkiliyor.

Blok zincir alanının önemli bir oyuncusu R3, kapsamlı bir araştırmadan sonra blok zincirin hedeflenen amaca uymadığını fark ettiler ve bu nedenle **CORDA**'yı yarattılar. Corda, finansal alanda odaklanmış bir defteri kebir teknolojisi (DLT-Distributed Ledger Technology) platformudur. R3, çok net bir değer önermesine sahip ve problemin pragmatik bir teknoloji yaklaşımı gerektirdiğini biliyor. Bu, bizim tecrübelerimiz ile de örtüşmekte ki; madencilik maliyetleri ve operasyonel verimsizlik nedeniyle mevcut blok zincir çözümleri, bazı iş vakaları için uygun bir seçenek olmayabilir. Bugüne kadar Corda ile geliştirme tecrübesi mükemmel olmamasına ve API'lerinin v1.0 sürümünden sonra dahi hala istikrarlı olmamasına rağmen, gelecekte DLT alanının daha da olgunlaşmasını bekliyoruz.

COSMOS DB, Microsoft'un global olarak dağıtık ve çok modlu veritabanı hizmeti olup, bu yılın başlarında kullanıma hazır duruma gelmiştir. En modern NoSQL veritabanları ayarlanabilir tutarlılık sunarken, Cosmos DB bunu temel dayanak olarak yapıyor ve beş farklı tutarlılık modeli sunuyor. Atom kayıt dizisi (ARS) olarak adlandırılan iç veri modeline eşleşen birden çok model sunması ile; anahtar-değer, belge, sütun ailesi ve grafik de desteklediğini vurgulamamız gereklidir. Cosmos DB'nun diğer ilginç bir özelliği ise; gecikme, verimlilik, tutarlılık ve mevcudiyet konusunda hizmet düzeyi anlaşmaları (SLA'lar) sunmasıdır. Geniş bir alanda uygulanabilirliği ile diğer bulut satıcılarının karşılaması gereken yüksek bir standart oluşturmuştur.

Son zamanlarda yayımlanan chatbots ve ses platformları ile paralel olarak, metinden niyeti anlamak ve karşılıklı konuşma akışlarının yönetmek için servisler sunan araçların ve platformların yaygınlaştığını görüyoruz. **DIALOGFLOW** (Google tarafından satın alınan, eski adıyla API.ai), bu türden doğal-dil-anlayış hizmeti sunan bir çözümdür ve bu alandaki wit.ai ve Amazon Lex ile rekabet halindedir.

Yazılım geliştirme ekosistemi konteyner orkestrasyonu platformu olarak Kubernetes'e yakınlaşırken, Kubernetes kümelerinin işletilmesi hala karmaşık bir halde devam ediyor. GKE (Google Container Engine), Kubernetes kümelerinin işletilmesi ve sürdürülmesinin operasyonel yükünü hafifleten, konteynerlenmiş uygulamaları dağıtmak için yönetilen bir Kubernetes servis çözümüdür. Ekiplerimiz, güvenlik yamalarının uygulanması, düğümlerin izlenmesi, otomatik olarak onarılması, çok kümeli ve çok bölgeli ağın yönetilmesi gibi zorlu işlerin platformu olarak GKE'yi kullanırken iyi bir deneyime sahip olmuşlardır. Deneyimlerimize göre, Google'ın platform yeteneklerini ortaya çıkarmaya yönelik API öncü yaklaşımı yanı sıra servis yetkilendirmesi için OAuth gibi endüstri standartlarını kullanması yazılımcı deneyimini geliştirmektedir. GKE'nin bizi geçmişte geçici olarak etkilediği hızlı bir gelişme altında olduğunu dikkate almak gerekir, geliştiricilerinin altta yapılan değişiklikleri kullanıcılardan soyutlama çabalarına rağmen. Terraform ile GKE ve benzeri araçlarda Kodlama ile Altyapı olgunluğu odağında sürekli gelişim bekliyoruz.

Dil sunucuları otomatik tamamlama, çağıranlar bulma ve yeniden biçimlendirme gibi özellikleri ile her hangi bir metin editörünün soyut sözdizimi ağacıyla çalışabilmesini sağlar.
(Language Server Protocol)

KAFKA STREAMS, akış uygulamaları oluşturmak için hafif bir kütüphanedir. Akış işlemeyi basitleştirmek ve asenkron servisler için ana akım bir uygulama programlama modeline kolayca erişilebilir kılmak için tasarlanmıştır. Probleminize akış işleme modeli uygulamak istediğiniz senaryolarda, bir küme (genellikle tam teşekküllü akış işleme frameworkleri tarafından sunulan) çalıştırmanın karmaşıklığına girmeyeceğiniz iyi bir alternatif olabilir. Yeni gelişmeler arasında, Kafka kümesinde 'tam olarak bir kez' akış işleme özelliği de yer alıyor. Bu özellik, Kafka üreticilerinde eşkuvvetlilik (idempotency) getiren ve birden fazla bölüm arasında atomik yazmalara olanak sağlayan yeni İşlem API'ları ile sağlanıyor.

Gelişmiş IDE'lerin gücünün çoğu, bir programı soyut bir sözdizimi ağacına (AST-Abstract Syntax Tree) ayırıp, daha sonra bu AST'yi program analizi ve manipülasyonu için kullanma becerisinden gelir. Bu, otomatik tamamlama, çağıranları bulma ve yeniden biçimlendirme gibi özellikleri destekler. Dil sunucuları bu yeteneği, herhangi bir metin düzenleyicisinin AST ile çalışmak için bir API'ya erişmesini sağlayarak elde eder. Microsoft, OmniSharp ve TypeScript Server projeleri sayesinde **LANGUAGE SERVER PROTOCOL**'ün (LSP) oluşturulmasına öncülük etmiştir. Bu protokolü kullanan herhangi bir editör, LSP-uyumlu sunucu desteği olan herhangi bir dil ile çalışabilir. Birçok dilde zengin metin düzenleme modları sunan programlama dillerinden vazgeçmeden

favori editörlerimizi kullanmaya devam edebileceğimiz anlamına geliyor - Emacs bağımlılarının hoşuna gidecek şekilde.

LORAWAN düşük güç tüketimi ve düşük oranlı veri transferi ile uzun mesafelerde iletişim için tasarlanmış bir geniş alan ağıdır. Cihazlar ve ağ geçitler arasında iletişimi sağlar ve daha sonra verileri örneğin uygulamalar veya sunuculara iletebilir. Tipik bir kullanım, uzun pil ömrü ve uzun menzilli iletişimin şart olduğu dağıtık sensör veya Nesnelerin İnterneti (IoT-Internet of Things) cihazları içindir. LoRaWAN, bu tür uygulamalar için normal Wi-Fi kullanıldığında ortaya çıkan temel sorunlardan ikisine hitap eder: mesafe ve güç tüketimi. Birkaç uygulaması var, dikkat çekici olan The Things Network, özgür, açık kaynaklı bir uygulama.

Deneysel kullanımdan üretim ortamında kullanıma geçerken; makine öğrenme modellerine uzaktan erişebilmek ve tüketici sayısıyla ölçeklendirebilmek için modelleri barındıracak ve derleyecek güvenilir bir yönteme ihtiyaç duyuyoruz.
(TensorFlow Serving)

MAPD, GPU'da çalışacak şekilde oluşturulmuş SQL destekli bir bellek içi sütun analiz algoritması veritabanıdır. Veritabanı iş yükünün aslında G/Ç veya CPU kullanım konusunda sınırlandırılıp sınırlandırılmadığını düşünürken , GPU'nun getirdiği paralelleşme ve VRAM'ın büyük bant genişliği ile kombine edilerek kullanıldığı durumlarda oldukça yararlı olabileceğini görmekteyiz. MapD, VRAM'daki en sık kullanılan verileri (grup, filtreler, hesaplamalar ve katılma koşullarıyla ilgili sütunlar gibi) şeffaf bir şekilde yönetir ve geri kalan verileri ana bellekte saklar. Bu bellek yönetimi kurulumu ile MapD, indekse ihtiyaç duymadan önemli sorgu performansı sağlar. GPU üzerine geliştirilmiş başka veritabanları olmasına rağmen, MapD, ana veritabanının açık kaynaklı yeni sürümüyle ve GPU Açık Analitik Girişimi ile bu segmenti yönetiyor. Eğer, analitik iş yükünüz hesaplama açısından yoğunsa, GPU paralelliklerinden yararlanabilir durumdaysa ve ana belleğe sığabilirse, MapD'yi değerlendirmenizi öneririz.

Tek bir sorunu çok iyi çözen basit araçları çok beğeniyoruz ve **NETLIFY** bu tanıma en iyi uyanlardan biri. Statik web sitesi içeriği oluşturabilir, bunu GitHub'a yükleyebilir ve daha sonra hızlı ve kolay bir şekilde sitenizi canlı ortama alıp, erişebilir hale getirebilirsiniz. Bu işlemi kontrol edebilmek için bir CLI bulunmaktadır ve içerik dağıtım ağları (Content Delivery Networks - CDN) desteklenmektedir. Grunt gibi araçlarla birlikte çalışabilir ve en önemlisi, Netlify HTTPS'i destekler.

Makine öğrenme modelleri günlük iş uygulamalarına da yayılmaya başlamış durumda. Yeterli öğrenme veri seti mevcut olduğunda, bu algoritmalar daha önce karmaşık istatistiksel modeller veya buluşsal yöntemler gerektiren problemleri çözebiliyorlar. Deneysel kullanımdan üretim ortamında kullanıma geçerken; bu modelleri barındıracak, dağıtacak ve uzaktan erişilerek tüketici sayısıyla ölçeklendirebilecek güvenilir bir yönteme ihtiyacımız ortaya çıkıyor. Bu sorunu, **TENSORFLOW SERVING**, remote gRPC arabiriminden eğitilmiş modele erişim sunarak adreslemekte. Bu sistem sayesinde eğitimli bir modelin çeşitli şekillerde yüklenebilmesi sağlanıyor. TensorFlow Serving, sürekli öğrenme güncellemelerini içeren bir model akışını da girdi olarak alabilmektedir. Ayrıca geliştiriciler, yükleme sürecini Docker ile destekleyerek kolaylaştırmışlar. gRPC seçimi TensorFlow çalıştırma (execution) modeliyle uyumlu olmasına rağmen, kod üretme ve yerel bağlamalar (native bindings) gerektiren protokollere karşı dikkatli olmakta fayda var.

Microsoft, konteyner alanındaki gelişmelere **WINDOWS KONTEYNERLERİ** ile yetişiyor. Bu yazı hazırlanırken Microsoft, Docker konteyneri olarak iki adet Windows işletim sistemi imajı sağlıyor, Windows Server 2016 Server Core ve Windows Server 2016 Nano Server Windows Konteyner'lerin gelişmesi gereken alanlar olmasına rağmen (örneğin; büyük imaj boyutlarını düşürmek, ekosistem desteğini ve dokümantasyonunu zenginleştirmek), ekiplerimiz bunları derleme ajanları gibi diğer konteynerların başarılı bir şekilde çalıştığı senaryolarda kullanmaya başladı bile.

Yazılımlarda, iş mantığı ve işlem düzeni ara katman sistemlerine (middleware) kaymasından endişe ediyoruz; özellikle bunun tek nokta üzerinden ölçeklendirme ve yönetim için konusunda uzman yetkinlik ve takım gerektirdiğini düşündüğümüzde. Son derece rekabetçi olan API ağ geçidi pazarındaki sağlayıcılar da, ürünlerini öne çıkarmaya çalıştıkları özellikleri genişleterek bu eğilimi sürdürüyor. Bu, temelde bir ters proxy olan ürünlerin abartılarak **AŞIRIYA KAÇMIŞ API AĞ GEÇİDİ** haline gelmesine neden olup, test edilmesi ve dağıtımı oldukça zor olan tasarımları teşvik etmektedir. API ağ geçitleri, kimlik doğrulama ve hız sınırlaması gibi bazı temel gereksinimleri çözmeye yardımcı olurlar, ancak iş akıllılarının uygulama veya servis katmanında tutulması gereklidir.

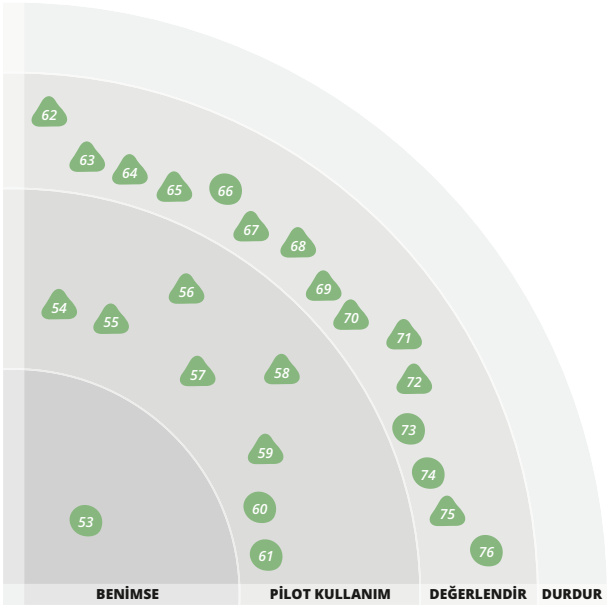
ARAÇLAR

BENİMSE
53. fastlane

PİLOT KULLANIM
54. Buildkite *YENİ*
55. CircleCI *YENİ*
56. gopass *YENİ*
57. Önyüz testleri için Headless Chrome *YENİ*
58. jsoniter *YENİ*
59. Prometheus
60. Scikit-learn
61. Serverless Framework

DEĞERLENDİR
62. Apex *YENİ*
63. assertj-swagger *YENİ*
64. Cypress *YENİ*
65. Flow *YENİ*
66. InSpec
67. Jupyter *YENİ*
68. Kong API Gateway *YENİ*
69. kops *YENİ*
70. Lighthouse *YENİ*
71. Rendertron *YENİ*
72. Sonobuoy *YENİ*
73. spaCy
74. Spinnaker
75. Spring Cloud Contract *YENİ*
76. Yarn

DURDUR



Ekiplerimiz sunduğu basitlik ve hızlı kurulumu sayesinde servis olarak sunulan bir CI/CD aracı olan **BUILDKITE**'ı çok beğendiler. Buildkite ile, derlemeleri çalıştırmak için - kendi veri merkezinizde veya bulutta - makinalarınıza basit bir derleme ajanı yükleyerek, sunulan servise bağlayabiliyorsunuz. Bir çok durumda, derleme yapılan makinaların konfigurasyonunda bu seviyede bir yetkiye sahip olmak, servis olarak sunulan ajan makinalarına kıyasla bir avantaj.

CIRCLECI, SaaS olarak ve yerinde (on-promise) kullanılabilen bir sürekli entegrasyon (Continuous Integration - CI) motorudur. CircleCI uzun süredir, düşük uyuşmazlık ve kolay kurulumlu bir derleme ve dağıtıma ihtiyacı olan bir çok geliştirme takımımızın genel geçer CI aracı oldu. CircleCI'ın 2.0 versiyonu giriş ve çıkış yelpazeleri döngüsü ve manuel geçitler ile derleme işlerinin iş döngülerini desteklemekle beraber mobil uygulama geliştirmeyi de desteklemektedir. Uygulama geliştiricilerin derleme hatlarının yerel ortamlarında kullanmasına, Slack ve diğer bildirim ve alarm sistemlerine kolay entegrasyonuna olanak sağlamaktadır. CircleCI'ın güvenlik pratiklerine dikkatlice göz atmanızı tavsiye ediyoruz, tıpkı şirketinizin varlıklarını barındıran diğer SaaS ürünlerine yaptığınız gibi.

Pass'a bağlı olan gopass şunlara benzer özellikler getirmiştir: tek bir ağaçta alıcı yönetimi ve birden çok şifre deposu desteği; interaktif bir arama fonksiyonu; zaman esaslı bir kerelik şifre (TOTP) desteği; ve binary verilerin depolanması.

(gopass)

GOPASS, GPG ve Git üzerine kurulmuş bir şifre yönetim çözümüdür. Pass'ın torunudur ve şunlara benzer özellikler getirmiştir: tek bir ağaçta alıcı yönetimi ve birden çok şifre deposu desteği; interaktif bir arama fonksiyonu; zaman esaslı bir kerelik şifre (TOTP) desteği; ve binary verilerin depolanması. Pass tan gopass'a geçerken veri dönüştürülmesi oldukça

basittir, çünkü pass'ın kullandığı format ile büyük ölçüde uyumludur. Bu aynı zamanda, depolanmış bir parola yapılan tek bir çağrı ile provizyon akışına entegrasyonun sağlanabileceği anlamına gelir.

2017 yılının ortalarından bu yana, Chrome kullanıcıları tarayıcıyı arayüzsüz (headless) modda çalıştırma seçeneğine sahip olmuşlardır. Bu özellik, bir ekranda işlemlerin görüntülenmesinin yükü olmadan önyüz tarayıcı testlerini çalıştırmak için ideal bir çözümdür. Eskiden, bu büyük oranda PhantomJS'in uzmanlık alanıydı, ancak Headless Chrome, hızla JavaScript tarafından yönlendirilen WebKit yaklaşımının yerini alıyor. Headless Chrome'daki testler daha hızlı ve gerçek bir tarayıcı gibi davranmakta. Ancak ekibimiz PhantomJS'den çok daha fazla bellek kullandığını fark ettiler. Tüm bu avantajlar sayesinde, **ÖNYÜZ TESTLERİ İÇİN HEADLESS CHROME**, standart olabilir.

Go ve Java'da yüksek performanslı bir JSON kodlayıcı ve kod çözücü arıyorsanız, açık kaynaklı **JSONITER** kütüphanesine bir göz atın. Kütüphane Go'daki standart JSON kodlama paketi ile de uyumludur.

Prometheus, Soundcloud tarafından geliştirilmiş, izleme ve zaman serileri veritabanı aracıdır ve sürekli ilerleme kaydetmektedir.

(Prometheus)

PROMETHEUS, Soundcloud tarafından geliştirilmiş, izleme ve zaman serileri veritabanı aracıdır ve sürekli ilerleme kaydetmektedir. Prometheus temelde çekme tabanlı (pull-based) HTTP modelini desteklerken, sunduğu uyarılar ile operasyon araç setinizin aktif bir parçası haline gelmektedir. Bu yazı kaleme alınırken, Prometheus 2.0 canlıya alınma aşamasında ve gelişmeye devam ediyor. Prometheus geliştiricileri çabalarını temel zaman serisi veri tabanları ve erişilebilir ölçütlerin çeşitlenmesi üstüne yoğunlaştırdı. Grafana, Prometheus kullanıcıları için tercih edilen görselleştirme aracı olmuştur ve Grafana ile birlikte dağıtımı sunulmaktadır. Ekiplerimiz, Prometheus izleme özelliğinin Elastic Stack'in indeksleme ve arama yetkinliklerini tamamladığını gözlemlemiştir.

APEX, AWS Lambda fonksiyonlarını kolaylıkla oluşturmak, devreye almak ve yönetmek için kullanılan bir araçtır. Apex ile, AWS'de doğal olarak desteklenmeyen dillerde, Golang, Rust vb. fonksiyonlar yazabilirsiniz. Bir alt süreç oluşturan ve olayları stdin ve stdout aracılığıyla işleyen bir Node.js shim tarafından sağlanır. Apex, geliştirici deneyimini iyileştiren bir çok hoş özelliğe sahip. Biz fonksiyonları yerel ortamlarda test edebilmek ve AWS kaynaklarında uygulamadan önce değişiklikleri prova edebilmeyi çok sevdik.

Bir AssertJ kütüphanesi, **ASSERTJ-SWAGGER**, API'nin kontrat şartnamesine uygunluğunu doğrulamanızı sağlar. Ekiplerimiz, API uç noktası geliştirmesindeki değişikliği, Swagger şartnamesini güncellemeden veya güncellenen dokümanın yayınlanmasında oluşabilecek sorunları yakalamak için assertj-swagger kullanıyor.

Sürekli Entegrasyonda (Continuous Integration-CI) uçtan uca testlerdeki başarısızlıklarının çözülmesi, özellikle arayüzsüz modda acı veren bir deneyim olabilir. **CYPRESS**, geliştiricilerin uçtan uca testler derlemesini kolaylaştıran ve tüm test adımlarını bir video olarak MP4 dosyasına kaydeden kullanışlı bir araçtır. Geliştiriciler, arayüzsüz modda sorunu yeniden üretmektense, düzeltmek için test videosunu izleyebilir. Cypress, yalnızca bir test framework'u değil, aynı zamanda güçlü bir platformdur. Şu aşamada, projelerimizde Cypress CLI'larını arayüzsüz modda CI süreçlerine entegre ediyoruz.

FLOW, kodlarınıza aşamalı olarak tür denetimi eklemenize izin veren JavaScript için statik bir tür denetleyicisidir. Farklı bir dil olan Typescript'in aksine, Flow, ECMAScript'in 5., 6. ve 7. sürümlerini destekleyen mevcut bir JavaScript koduna aşamalı olarak eklenebilir. En çok endişelendiren koddan başlayarak, sürekli entegrasyon hattınıza eklemenizi öneriyoruz. Flow, kodun netliğine katkıda bulunur, yeniden yapılandırma güvenilirliğini artırır ve türle ilgili hataları derlerken erkenden yakalar.

Son yıllarda, analitik notebooklarının istikrarlı yükselişinin farkındayız. Bunlar; metinlerin, görsellerin ve kodların içinde yaşadığı, Mathematica'dan ilham almış çalıştırılabilen uygulamalardır. Bir önceki sayıda bunların Clojure versiyonu olan GorillaREPL'den bahsetmiştik. Ancak makine öğrenmesine olan ilginin artması ve özellikle Python gibi programlama dillerine bu alandaki uzmanların yönelimi, ilgiyi Python notebooklar üzerinde odakladı. Bu yönelim ile **JUPYTER** ThoughtWorks takımları arasında ilgi görmeye başlamış durumda.

Kong, Mashape tarafından geliştirilmiş ve desteklenen bir açık kaynak API ağ geçitidir. Ayrıca Mashape, Kong'u kendi API analitik ve geliştirici portal araçlarıyla

entegre ettiği kurumsal bir paket de sunmaktadır. Kong, uç API ağ geçidi veya dahili API proxy'si gibi çeşitli yapılandırmalarda kullanılabilir. OpenResty, Nginx modülleri aracılığıyla, Lua eklentileri ile yazılan uzantılar sayesinde güçlü ve performanslı bir temel sağlar. Kong, PostgreSQL'i tek bölgeli dağıtımlar için veya Cassandra'yı çok bölgeli yapılandırmalar için kullanabilir. Geliştiricilerimiz, Kong'un yüksek performansını, önce API yaklaşımını (yapılandırmasının otomasyonunu mümkün kılıyor) ve konteyner olarak çalıştırılmasının kolaylığını çok sevdi. Aşırıya kaçan API ağ geçitlerinin aksine, **KONG API GATEWAY**, daha küçük bir özellik kümesine sahiptir, ancak trafik kontrolü, güvenlik, günlüğe kaydetme, izleme ve kimlik doğrulama gibi API ağ geçidi özelliklerinin temel setini uygular. Kong'u yakın gelecekte bir servis kafesi konfigürasyonunda değerlendirmekten heyecan duyuyoruz.

KOPS, canlı ortamdaki yüksek kullanılabilirlik ihtiyacı olan Kubernetes kümeleri yaratmak ve yönetmek için bir komut satırı aracıdır. Başlangıçta AWS'yi hedeflemişken, diğer sağlayıcılara da deneysel destekler sunduğunu görüyoruz. Projeyi hızlı bir şekilde ayağa kaldırıp çalıştırmayı sağlar, hatta bazı özellikler (dönen güncellemeler gibi) henüz tam olarak geliştirilmemiş de olsa, topluluğun ilgisine hayran kaldık.

LIGHTHOUSE Google tarafından web uygulamalarının Progressive Web App standartlarına uyumluluğunu analiz eden bir araçtır. Bu sene yayınlanan Lighthouse 2.0 versiyonu ile, performans metrikleri ve erişilebilirlik kontrolleri de temel araç setine eklendi. Yeni eklenen bu özellikler, standart Chrome geliştirici araçlarında, denetim sekmesinin altına eklendi. Lighthouse 2.0'in bir diğer faydası ise Chrome'un arayüzsüz çalışan modu. Lighthouse 2.0 komut satırından veya bağımsız Node.js uygulaması gibi çalıştırılabilmektedir. Bu yüzden Pa11y ve benzeri araçlar, sürekli dağıtım sistemlerinde erişilebilirlik kontrolü yapan araçlara alternatif oluşturmaktadır.

JavaScript ağırlıklı web uygulamaları için sürekli bir sorun, bu sayfaların dinamik kısmını arama motorları tarafından nasıl kullanılabilir hale getirileceğidir. Geçmişte geliştiriciler, React ile sunucu tarafında render yaparak, harici hizmetler veya prerendering de dahil olmak üzere çeşitli yöntemlere başvurmuşlardı. Artık Google Chrome'un yeni arayüzsüz modunu kullanan **RENDERTRON**, bu probleme yeni bir çözüm getiriyor. Rendertron, tek başına bir HTTP sunucusu olarak dağıtmaya hazır bir Docker konteynerinde arayüzsüz bir Chrome örneğini sarar. JavaScript render edemeyen botlar bu sunucuya yönlendirilerek onlar adına render yapması sağlanabilir. Geliştiriciler her zaman kendi arayüzsüz Chrome proxy'lerini ve ilişkili yönlendirme makinesini kurabilirler ancak Rendertron, yapılandırma ve dağıtım işlemini basitleştirir ve botların saptanması ve yönlendirilmesi için örnek orta katman kodu sağlar.

JavaScript yoğun web uygulmalarındaki süre gelen sorun arama motorları için sayfaların dinamik kısımlarını nasıl kullanılabilir hale getirileceğidir. JavaScript'ı render edemeyen botlar Renderton sunucularına yönlendirilebilir.

(Rendertron)

SONOBUOY, herhangi bir Kubernetes setinde uçtan-uca uygunluk testlerini tahribatsız bir şekilde çalıştırmak için kullanılan bir teşhis aracıdır. Kubernetes projelerinin iki yaratıcısı tarafından kurulan Heptio ekibi, Kubernetes setlerin birlikte çalışabilirliği konusunda açık kaynak standardizasyonunu izlerken, Kubernetes dağıtımlarının ve yapılandırmalarının da en doğru şekilde yapılmasının kontrolünü sağlamak amacıyla bu aracı geliştirdi. Sonobuoy ile altyapı olarak kod oluşturma hattının bir parçası olarak çalıştırmayı ve Kubernet kurulumlarımızı sürekli olarak izleyerek, tüm setin davranışını ve sağlığını doğrulamak üzere deneme kullanımları yapıyoruz.

Eğer Spring framework kullanarak Java servisleri geliştiriyorsanız, istemci odaklı kontrat (CDC) testleri için **SPRING CLOUD CONTRACT** 'ı düşünebilirsiniz. Bu aracın mevcut ekosistemi, müşteri isteklerini ve kontrata karşı sunucu uygulamasını doğrulamayı destekliyor. Açık kaynaklı, tüketici odaklı bir sözleşme test aracı seti olan Pact'e kıyasla, kontratların aracılık edilmesi ve diğer programlama dillerine destekten yoksundur. Bununla birlikte Spring ekosistemi ile çok iyi entegre olmaktadır, örneğin Spring Integration ile mesaj yönlendirmek gibi.

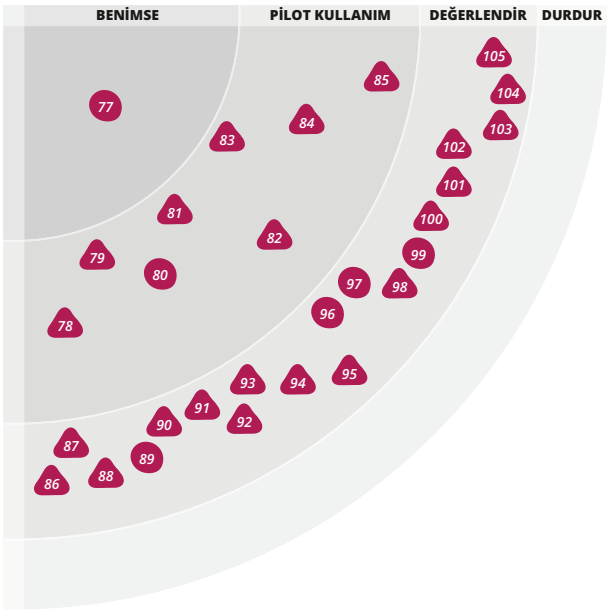
DİLLER VE FRAMEWORK'LER

BENİMSE
77. Python 3

PİLOT KULLANIM
78. Angular
79. AssertJ *YENİ*
80. Avro
81. CSS Grid Yerleşimi *YENİ*
82. CSS Modülleri *YENİ*
83. Jest *YENİ*
84. Kotlin
85. Spring Cloud

DEĞERLENDİR
86. Android Mimari Bileşenleri *YENİ*
87. ARKit/ARCore *YENİ*
88. Atlas ve BeeHive *YENİ*
89. Caffe
90. Clara rules *YENİ*
91. JS'de CSS *YENİ*
92. Digdag *YENİ*
93. Druid *YENİ*
94. ECharts *YENİ*
95. Gobot *YENİ*
96. Instana
97. Keras
98. LeakCanary *YENİ*
99. PostCSS
100. PyTorch *YENİ*
101. single-spa *YENİ*
102. Solidity *YENİ*
103. TensorFlow Mobil *YENİ*
104. Truffle *YENİ*
105. Weex *YENİ*

DURDUR



React, Vue veya Ember için önceki Radar sürümlerinde, **ANGULAR**'a güçlü bir öneri sunmaktan çekiniyorduk, çünkü esas itibariyle eskiden sevdiğimiz AngularJS ile sadece ismini paylaşan yeni ve çok da şaşırtıcı olmayan bir framework idi. Bu sırada, şu an sürüm 5 olan Angular, git gide geriye dönük uyumluluk da sağlarken istikrarlı bir şekilde gelişti. Birkaç farklı ekibimiz yeni Angular ile yaptıkları uygulamaları yayına aldılar ve gördüklerinden memnun kaldılar. Birkaç ekibimiz Angular'ı sağlam bir seçim olarak görmeye başladığından Radar'da Trial'a çekmeyi uygun gördük. Bununla birlikte, ekibimizin çoğunun, yine de React, Vue veya Ember'ı Angular'a tercih ettikleri görüyoruz.

ASSERTJ, test kodunda amacı iletmeyi kolaylaştıran, iddalar (assertion) için akıcı arayüz sağlayan bir Java kütüphanesidir. AssertJ, okunabilir hata mesajları, hafif iddialar, iyileştirilmiş koleksiyonlar ve istisna desteği sağlar. Bazı takımların Hamcrest ile birleşik JUnit kullanmak yerine varsayılan olarak AssertJ kullandığını gözlemliyoruz.

CSS Grid Yerleşimi, iki boyutlu, grid tabanlı bir yerleşim sistemidir; tahmini boyutlandırma davranışlarını kullanarak yerleşim için mevcut alanı, sütunlara ve satırlara bölen bir mekanizmadır.

(CSS Grid Yerleşimi)

CSS, sayfalarının yerleşiminin oluşturulmasında çok açık destek sağlamasa da, web sayfalarını düzenlemek için tercih edilen bir seçenektir. Flexbox daha basit, tek boyutlu düzenlerde yardımcı oldu, ancak geliştiriciler daha karmaşık yerleşimlerde kullanmak üzere kütüphaneler ve araçlara

eriştiler. **CSS GRİD YERLEŞİMİ** , iki boyutlu, grid tabanlı bir yerleşim sistemidir; tahmini boyutlandırma davranışlarını kullanarak yerleşim için mevcut alanı, sütunlara ve satırlara bölen bir mekanizmadır. Grid herhangi bir kütüphane gereksinimi duymaz, Flexbox ve diğer CSS öğeleri ile uyumludur. Ancak IE11 sadece parçalı desteklenmektedir, dolayısıyla Windows 7'de bir Microsoft tarayıcısına bağımlı olan kullanıcıları yoksayar.

Birçok büyük CSS kod tabanları, adlandırma çakışmalarını önlemeye yardımcı olmak için karmaşık adlandırma şemalarının kullanılmasını gerektirir. **CSS MODÜLLERİ** bu tür sorunları, tüm sınıf adlarını bir CSS dosyasında yerel kapsamı oluşturarak çözmektedir. Bu dosya, CSS sınıflarının dizeler olarak referanslandığı bir JavaScript modülüne aktarılır. Ardından, derleme hattı (Webpack, Browserify vb.) içinde, sınıf adları üretilen tekil metinler ile değiştirilir. Bu sorumluluklarda önemli bir değişikliktir. CSS Modülleri öncesinde, bir kişinin sınıf adlarındaki çakışmayı önlemek için elle global namespace'i yönetmesi gerekiyordu. Şimdi bu sorumluluk geliştirme araçlarındadır. Karşılaştığımız küçük bir dezavantaj, fonksiyonel testler genellikle yerel kapsamında dışındadır ve bu nedenle CSS dosyasında tanımlanan isim ile referanslanamaz. Bunun yerine, kimlik bilgilerini (IDs) veya veri özelliklerini kullanmanızı öneririz.

Jest, 'sıfır konfigürasyon' deneyimi sunan bir ön-yüz test aracıdır, sahip olduğu yepyeni özellikleri, mocking ve kod kapsama, React ve diğer JavaScript framework'lerini hedefler.

(Jest)

Ekiplerimiz, ön-yüz testleri için **JEST** kullanmanın sonuçlarından oldukça memnun. Jest,'sıfır-yapılandırma' deneyimi sağlar ve mocking ve kod kapsama alanı gibi özelliklere sahiptir. Bu test frameworkunu yalnızca React uygulamaları için değil, aynı zamanda diğer JavaScript framworklerde de kullanabilirsiniz. Jest'in sıklıkla bahsedilen özelliklerinden biri UI anlık görüntü testidir. Anlık görüntü testi, test piramidinin üst katmanına iyi bir katkı olabilir, ancak birim testlerin hala temel olduğunu unutmayın.

First-class Android desteğinin geldiğinin duyurusu hızla ilerleyen **KOTLIN** diline ekstra bir destek sağladı ve ayrıca lokal çalıştırılabilir dosyaları derlemek için Kotlin/Native'in LLVM destekli özelliğini de yakından takip ediyoruz. Android geliştirmeye yönelik Anko kütüphanesinin yanı sıra, Null kontrolü, veri sınıfları ve DSL oluşturma kolaylığı oldukça keyif aldığımız faydalardan diyebiliriz. İlk derleme sırasındaki yavaşlığı ve birinci sınıf IDE desteği için IntelliJ'ye olan bağımlılık eksikliklerine rağmen, bu taze ve özlü modern dili denemenizi öneririz.

SPRING CLOUD gelişmeye ve ilginç yeni özellikler eklemeye devam ediyor. Örneğin Spring-cloud-streams projesinde Kafka Streams'e bağlama konusundaki desteği, Kafka ve RabbitMQ için geliştirilen konektörler ile iletiye dayalı (message driven) uygulamaların oluşturulmasını göreceli olarak kolaylaştırmıştır. Karmaşık mimarilere getirdiği basitliği, örneğin ZooKeeper ve dağıtık sistemleri oluştururken karşılaşılan ortak sorunlara verilen desteği, örneğin spring-cloud-sleuth ile takibi, kullanan ekipler takdir etmektedir. Her zamanki uyarılar halen geçerli ancak başarılı bir şekilde bir çok projede kullanıyoruz.

Geçmişte, Google'ın Android örneklerinin dokümantasyonu, mimari ve yapısal açıdan eksikti. Bu durum, geliştiricilerin daha iyi mimariler ile Android uygulamaları geliştirmelerine yardımcı olan ve sabit fikirli kütüphanelerden oluşan **ANDROID MİMARİ BİLEŞENLERİ** sürümüyle değişiyor. Bu sürümde Android gelişiminde uzun süredir devam eden sıkıntılı noktalar adresleniyor; yaşam döngüleriyle uğraşmak, sayfalama (pagination), SQLite veritabanları ve konfigurasyon değişikliklerinde veri saklama. En çok ihtiyaç duyduğunuz kütüphaneleri seçip mevcut projenize entegre edebilirsiniz, hepsini birlikte kullanmanız gerekmez.

Mobil artırılmış gerçeklik alanında bir sürü yeni hareketlilik gözlemlemekteyiz. Bunların çoğunu, artırılmış gerçeklik (Augmented Reality-AR) kütüphaneleri olan **ARKIT VE ARCORE** oluşturmakta, sırasıyla Apple ve Google tarafından kullanılan. Bu kütüphaneler mobil AR teknolojilerini ana akım haline getiriyor. Şirketler için asıl zorluk ise aldatmacanın ötesine geçip temelde kullanıcı deneyimini iyileştiren orijinal çözümlere dair kullanım örnekleri bulmak olacaktır.

Çoklu uygulama stratejisi gerçekten tartışmalı, özellikle de her geçen gün daha az sayıda kullanıcının yeni uygulama indirmekte olduğu şu dönemde. Yeni bir uygulama sunmak ve indirme sayılarıyla uğraşmak yerine, takımlar istenen özellikleri, yoğun olarak indirilmiş mevcut uygulamalar (single app) üstünden geliştirmesi, teslim etmesi gerekir ki bu da mimari zorlukların oluşmasına sebep olur. **ATLAS VE BEEHIVE** sırasıyla Android ve iOS uygulamaları için modülerleştirmesini sağlayan çözümlerdir. Atlas ve BeeHive, geliştirme takımlarının fiziksel olarak yalıtılmış modüller üzerinde çalışmasını, yeniden derlemesini veya dinamik olarak ilgili modülleri bir uygulama üstünden yüklemelerini sağlar. Alibaba azalan yeni uygulama indirme

ve tek bir uygulma mimarisi zorlukları ile karşılaşmıştı ve her ikisini de açık kaynak kodlu proje olarak geliştirmiştir.

Normal şartlar altında kural motorlaru seçmede ilk kuralımız bir kural motoruna ihtiyacınız olmadığıdır. Bir çok insanın, duruma özel kod yazmak daha iyi bir çözümken, sahte sebepler ile kendilerini test etmesi zor, kara kutu kural motorlarına bağladığını gördük. Bununla birlikte, **CLARA RULES**'u kural motoru kullanmanın mantıklı olduğu yerlerde başarılı olarak kullandık. Kuralları ifade ederken ve değerlendirirken basit Clojure kodu kullanmasını sevdik, bu yenilemeye, test etmeye ve kaynak kontrolüne olanak sağlıyor. Clara rules, iş uzmanları direkt olarak kuralları manipüle etmesi ilizyonunu takip etmek yerine, iş uzmanları ve yazılım geliştiriciler arasındaki iş birliğini arttırmaktadır.

JS'DE CSS, JavaScript programlama dilinde CSS biçimlendirmesi yazmak için kullanılan bir tekniktir. Bu görsel ve mantıksal endişeleri saptamak için geçerli olan JavaScript bileşeni ile stil yazmanın yaygın bir patterni teşvik eder. Yeni oyuncular - JSS, emotion ve styled-components'u içeren, JS'de CSS koduna çevirmek için kullanılan araçlara güvenebilirsiniz. Böylelikle CSS stil sayfalarını tarayıcı uyumlu hale getirebilirsiniz. Bu JS'de CSS yazmaya yönelik ikinci nesil yaklaşımdır. Önceki yaklaşımlardan farklı olarak satır içi stillere güvenmemektedir. Bu, tüm CSS özelliklerini desteklenmesini, npm ekosistemini kullanmayı ve bileşenlerin birden fazla platformda paylaşılabilmesini sağlar. Ekiplerimiz çalışmalarında gözlemlediği styled-components bileşen bazlı frameworkler ile iyi çalışıyor, örneğin React, ve CSS'lerin birim testi için jest-styled-components. Bu alan yeni ve sürekli değişiyor; yaklaşım tarayıcıda oluşturulan sınıf adlarının el ile ayıklanması için biraz çaba gerektiyor. Önyüz mimarisinin tekrar kullanılan bileşenleri desteklemediği veya genel stil gerektiren bazı projelere uygulanmayabilir.

DIGDAG cloud üzerinden karmaşık veri hatlarını oluşturan, çalıştıran, zamanlayan ve izleyen bir araçtır. Kullanıma hazır operatörler kullanarak veya API üzerinden kendiniz oluşturarak, YAML ile veri hattı tanımlayabilirsiniz. Digdag bağımlılık yönetimi, tekrar kullanımı sağlamak için modüler iş akışı, güvenli gizlilik yönetimi ve çoklu dil desteği gibi bir çok veri hattı çözümünün sağladığı özelliklere sahiptir. Bizi heyecanlandıran özelliği ise AWS RedShift, S3 ve Google BigQuery'de bulunan verileri birleştirme ve taşımaya olanak sağlayan çoklu cloud desteğidir. Daha fazla cloud sağlayıcısının önerdiği veri işleme çözümlerinin yanında Digdag'ın (ve benzeri araçların) görev için en iyi seçeneğin kullanılmasında yararlı olacağını düşünüyoruz.

DRUID, zengin izleme özellikleri olan bir JDBC bağlantı havuzudur. Veritabanında yürütülen SQL deyimlerini semantik izlemeyi sağlayan gömülü SQL ayrıştırıcı içerir. Enjeksiyonlar veya şüpheli SQL işlemler engellenecek ve JDBC katmanından log'lara direk kayıt edilecektir. Dahası sorgular semantiklerine göre birleştirilebilir. Druid, Alibaba tarafından geliştirilmiş açık kaynaklı bir projedir ve Alibaba'nın kendi veritabanı sistemlerini işletirken edindiği öğrenimleri yansıtır.

Android Mimari Bileşenleri, sabit kütüphanelerden oluşur, geliştiricilere daha iyi mimarilere sahip uygulamaları yaratmalarında yardımcı olur.

(Android Mimari Bileşenleri)

ECHARTS, farklı grafikler ve etkileşim türleri için zengin desteği olan hafif bir grafik kütüphanesidir. ECharts tamamen Canvas API'ını temel aldığından 100K üzerinde veri ile dahi çalışırken inanılmaz bir performans gösterir. Aynı zamanda mobil kullanım için de optimize edilmiştir. Kardeş projesi ECharts-X ile birlikte 3D çizimi destekliyor. ECharts bir Baidu açık kaynak projesidir.

Go programlama dilinin fiziksel donanımlar üstünde derleme yapabiliyor olması, gömülü sistemlerde çalışan geliştiricilerin ilgisini çekti. **GOBOT**, robotik, fiziksel bilgi işleme ve Nesnelerin İnternet'i (IoT) gibi Go Programlama dili ile yazılmış ve çeşitli platformları destekleyen bir frameworktür. Gobot'u gerçek zamanlı cevabın şart olmadığı deneysel robotik projeler için kullandık ve Gobot'la açık kaynak yazılım sürücüleri oluşturduk. Gobot HTTP API'leri, daha zengin uygulamalar oluşturmak için mobil cihazlarla kolay donanım entegrasyonunu yapabilmeyi sağlar.

Mobil ekiplerimiz, Android ve Java'da can sıkıcı hafıza sızıntılarını tespit etmek için kullanılan **LEAKCANARY** hakkında heyecanlanmış durumda. Kanca yapmak basittir ve sızıntının sebebini net bir şekilde takip etmeyi sağlayan bildirimler sunar. Bunu araç kitinize eklediğinizde, birden fazla cihazda yaşanan bellek aşımı hatalarının giderilmesinde sıkıcı saatler kaybedilmesinin önüne geçebilirsiniz.

PYTORCH , Torch makine öğrenme frameworkünün Lua'dan Python'a tam bir yeniden yazımıdır. Programcılar, Tensorflow ile karşılaştırıldığında oldukça yeni ve olgun olmamasına rağmen PyTorch ile çalışmanın daha kolay

olduğunu keşfediyorlar. Object-orientation ve native Python olması nedeniyle, modeller daha açık ve özlü olarak ifade ediliyor ve yürütme sırasında debug yapılabiliyor. Bu frameworklerin bir çoğunun kısa bir süre önce ortaya çıkmasına rağmen, PyTorch, CUDA mimarileri için sürekli destek sağlamayı garanti eden NVIDIA da dahil olmak üzere Facebook ve geniş ortak organizasyonlar tarafından destekleniyor. ThoughtWorks ekipleri, PyTorch'un modellerini denemek ve geliştirmek için kullanışlı buluyor; ancak yine de TensorFlow'un üretim ölçekli eğitim ve sınıflandırma performansına güveniyor.

SINGLE-SPA, tek uygulama içerisinde farklı frameworklerle birden fazla mikro önyüz kullanılmasını sağlayan bir JavaScript frameworküdür. Genel olarak, bir uygulama için birden fazla framework kullanılmasını önermiyoruz, ancak bunu yapmaktan kaçınamayacağımız bazı durumlar olabilir. Örneğin, eski bir uygulama ile çalışırken veya varolan frameworkün yeni bir sürümü ile bir özellik geliştirmek isterseniz, single-spa oldukça faydalı olabilir. Pek çok JavaScript frameworkünün kısa ömrünü göz önüne aldığımızda, gelecekteki framework değişiklikleri ve yerelleştirilmiş denemeleri tüm uygulamayı etkilemeden yapabilecek bir çözüme ihtiyaç olduğunu görüyoruz. Single-spa, bu yönde iyi bir başlangıç gibi gözüküyor.

Akıllı kontratlar için programlama, işlemler için komut dizisi dilinden (Scripting) daha fazla ifade kabiliyeti olan bir dil gerektirir. **SOLIDITY**, akıllı kontratlar için tasarlanan yeni programlama dilleri arasında en popüler kaynaktır. Solidity, sözleşmeye dayalı, statik olarak yazılmış bir dildir ve söz dizimi JavaScript'e benzer. Akıllı sözleşmelerde kendi kendini zorlayan iş mantığını yazmaya yönelik soyutlamalar yapabilmeyi sağlar. Birbiriyle uyumlu çalışan yazılım uygulamaları açısından hızla büyümektedir. Solidity, Ethereum platformunda geliştirme yapanlar için birincil tercihtir. Akıllı kontratların değişmez doğası göz önünde bulundurulduğunda, bağımlılıkların titiz testi ve sıkı denetimin hayati önem taşıdığını söylememeliyiz.

TENSORFLOW MOBİL, geliştiricilerin iOS veya Android uygulamalarına, geniş kapsamlı bir anlama ve sınıflandırma tekniğini dahil etmelerini sağlar. Cep telefonlarında bulunan sensör verisi genişliği göz önüne alındığında bu daha da önem arz etmektedir. Önceden eğitilmiş TensorFlow modelleri bir mobil uygulamaya yüklenebilir ve canlı video görüntülerine, metin veya konuşma gibi girdilere uygulanabilir. Cep telefonları, bu hesaplama modellerini uygulamak için şaşırtıcı derecede uygun bir platform sunmaktadır. TensorFlow modelleri, geliştirmeciler için bazı problemler çıkartabilecek protobuf dosyaları olarak dışa aktarılır ve yüklenir. Protobuf'un binary formatı, modelleri incelemeyi zorlaştırabilir ve doğru protobuf kütüphane sürümünü mobil uygulamanıza bağlamanızı gerektirir. Fakat, lokal model çalıştırma özelliği, uzakta çalıştırmanın getirdiği iletişim yükünü ortadan kaldırarak TensorFlow Serving'e karşı cazip bir alternatif sunmaktadır.

Clara Rules'u kural motoru kullanmanın mantıklı olduğu yerlerde başarılı olarak kullandık. Kuralları ifade ederken ve değerlendirirken basit Clojure kodu kullanmayı tercih ettik, böylelikle kurallar yenileme, test etme ve kaynak kontrolüne tabi oldular.

(Clara rules)

TRUFFLE, modern web geliştirme deneyimini Ethereum platformuna getiren bir geliştirme çerçevesidir. Akıllı kontrat derleme (smart contract compiling), kütüphane bağlama ve devreye alma işini ve farklı blok zinciri ağlarındaki eserlerle uğraşmayı ele alıyor. Truffle'ı tercih etmemizin sebeplerinden biri insanları akıllı kontratlar için test yazmaya teşvik etmesidir. Testleri gerçekten ciddiye almalısınız çünkü akıllı kontrat programlaması genellikle para ile ilgilidir. İçinde var olan test framework'u ve TestRPC ile entegrasyonuyla Truffle sözleşmeyi TDD (Test Driven Development) ile yazmayı mümkün kılar. Blok zinciri için sürekli entegrasyonu desteklemek üzere Truffle'a benzer daha fazla teknoloji görmeyi bekliyoruz.

WEEX, Vue.js bileşen sentaksı kullanarak çapraz platform mobil uygulamaları oluşturmak için bir frameworktür. Ayrıca Weex, native mobil uygulamalarda Vue.js'nin sadeliğini tercih edenler için uygun bir seçenektir, ancak daha karmaşık uygulamalar için de çok iyi çalışır. Bu framework ile çok başarılı bir şekilde geliştirilen oldukça karmaşık mobil uygulamalar olduğunu görüyoruz. Örneğin Çin'de en popüler mobil uygulamalardan ikisi olan TMall ve Taobao. Weex Alibaba tarafından geliştirilmiştir ve şuan bir Apache kuluçka projesidir.

Teknoloji Radarı çıktığında ilk öğrenen siz olun ve özel web seminerleri ve içerik hakkında güncel bilgilere ulaşmak için

HEMEN ABONE OLUN

thght.works/Sub-TR

ThoughtWorks®

ThoughtWorks, yazılım danışmanlığı, üretimi ve ürünleri konusunda uzmanlaşmış tutkulu ve hedef sahibi bireylerin oluşturduğu bir topluluk ve danışmanlık şirketidir. Müşterilerimizin teknolojiyi işlerinin merkezine yerleştirmelerine yardım ediyoruz. Olumlu toplumsal değişime adanmış bir topluluk olarak misyonumuz insanlığı yazılım üzerinden daha iyi anlamak ve aynı doğrultuda mücadele eden birçok kuruluşla ortak hareket etmektir.

20 yıl önce kurulan ThoughtWorks, bugün yazılım ekipleri için öncü araçlar üreten bir ürün birimini de kapsayan ve 4500'den fazla çalışanı olan bir şirkete dönüşmüştür. ThoughtWorks 15 ülkede (Avustralya, Brezilya, Şili, Çin, Ekvador, Almanya, Hindistan, Singapur, Güney Afrika, Türkiye, Birleşik Krallık ve ABD) 42 ofisi bulunmaktadır.

thoughtworks.com