ThoughtWorks®

TEKNOLOJİ RADARI *NİSAN '16*

Geleceği şekillendiren
eğilimler ve teknoloji üzerine
görüşlerimiz

thoughtworks.com/radar

YENİLİKLER

Bu sayıda öne çıkan başlıklar şunlar:

VERİMLİ BİR YAN ÜRÜN OLARAK AÇIK KAYNAK

Radarımızdaki en etkili yazılımlardan bazıları, esas işi yazılım araçları yaratmak olmayan şirketlerden geliyor. Radar'daki maddelerden bazıları, geleneksel bir yazılım geliştirme aracı üreticisi sayılmayan Facebook'tan geliyor. Geçmişten farklı olarak bugün birçok şirket önemli yazılım varlıklarını açık kaynak haline getirerek yeni çalışanlar için cazip olmaya ve prestij kazanmaya çalışıyorlar. Bu, verimli bir geri besleme döngüsü oluşturuyor: İnovatif açık kaynaklar geliştirme alanındaki başarılı elemanları çekecek, buna karşılık bu yeni elemanların daha inovatif olma ihtimali de artacaktır. Bir yan etki olarak, şirketlerin framework'leri ve kütüphaneleri sektördeki en önemli şeylerden biridir. Bu da yazılım geliştirme ekosistemi içinde büyük bir dönüşüm oluştururken açık kaynak yazılımlarının doğru bağlamda kullanıldıkları takdirde ne kadar verimli olabileceğini kanıtlıyor.

PAAS BULMACASINI ÇÖZÜMLEMEK

Birçok büyük şirket Bulut ve Platform as a Service (PaaS'ı) altyapısını standartlaştırmak, kurulum ve işletimleri kolaylaştırmak ve geliştiricileri daha verimli hale getirmek için bariz bir yöntem olarak görüyorlar. Ancak henüz erken dönemi yaşıyoruz, PaaS'ın tanımı henüz net değil ve birçok PaaS Yaklaşımı tamamlanmamış durumda veya bunları destekleyen framework ve araçların yeterli olgunluğa erişmemiş olmasından olumsuz etkileniyor. Bazı PaaS çözümleri düz Infrastructure as a Service (IaaS) ile daha kolay yapılacak işleri zorlaştırıyor. Örneğin Özel Service Locator veya karmaşık ağ topolojileri kullanmak böyledir ve «Servis olarak konteynır» yaklaşımının aynı değeri daha fazla esneklikle sağlayıp sağlayamayacağı konusunda henüz karar verilmiş değildir. Birçok şirketin hazır satılan bir PaaS kullanmakta veya kendi PaaS'larını kademe kademe kurmakta olduklarını ve farklı derecelerde başarı elde ettiklerini görüyoruz. Bizim düşüncemiz bugün inşa edilen herhangi bir PaaS'ın son durum olmayacağı ancak bir evrim sürecinin parçası olacağı yönünde. Kurumsal olarak Cloud veya PaaS'a göç edilmesi birçok fayda sağlamakla birlikte, özellikle ardışık düzen tasarımı ve araç kullanımı konularında birçok zorluk ve zorlanmaları da beraberinde getiriyor. Bu teknolojilerin tüketicileri kendi bağlamlarında «en yoğun oldukları döneme hazır» olduklarını gösteren dönüm noktasına dikkat etmeli ve PaaS'larının uygulamadaki ayrıntılarına fazla sıkı bir bağlılık göstermekten kaçınmalıdırlar.

DOCKER, DOCKER, DOCKER!

Konteynırlaştırma ve özellikle Docker bir uygulama yönetim tekniği olarak son derece yararlı olduğunu ve ortamlar arasında kurulumu rasyonelleştirirken «burada çalışıyor ama orada çalışmıyor» türünden sorunları basitleştirdi. Geliştirme/test aşamalarının ötesinde ve üretimin sonuna kadar Docker kullanımına - ve özellikle Docker etrafındaki ekosistemin kullanımına - odaklanan önemli miktarda enerji olduğunu görüyoruz. Docker konterynırları birçok PaaS ve «veri merkezi İS» platformları için bir «ölçeklendirme birimi» olarak kullanılıyor ve Docker'ın ivmesinin daha da artmasını sağlıyor. Docker hem geliştirme hem de üretim ortamı olarak olgunlaştıkça, konteynırlaştırma, yan etkileri ve dolaylı sonuçlarına giderek daha fazla dikkat ediliyor.

AŞIRI REAKTİF Mİ?

Zorunlu bağlantılar kullanılması yerine bileşenlerin kendilerine aktarılan verilerdeki değişikliklere reaksiyon gösterdiği reaktif programlama son derece popüler hale geldi. Reaktif uzantılar hemen hemen her programlama dilinde bulunuyor. Özellikle kullanıcı ara yüzleri yaygın bir şekilde reaktif tarzda yazılıyor ve birçok ekosistem bu paradigma üzerine oturuyor. Bu modeli biz de sevdik ancak olay bazlı sistemlerin aşırı kullanılması program mantığını karmaşıklaştırıyor ve anlaşılmasını zorlaştırıyor; geliştiricilerin bu programlama tarzını sağduyulu kullanmaları gerekiyor. Çok popüler olduğu kesin: Bu Radar'da önemli sayıda reaktif framework ve destek araçlarının incelemesini sunuyoruz.

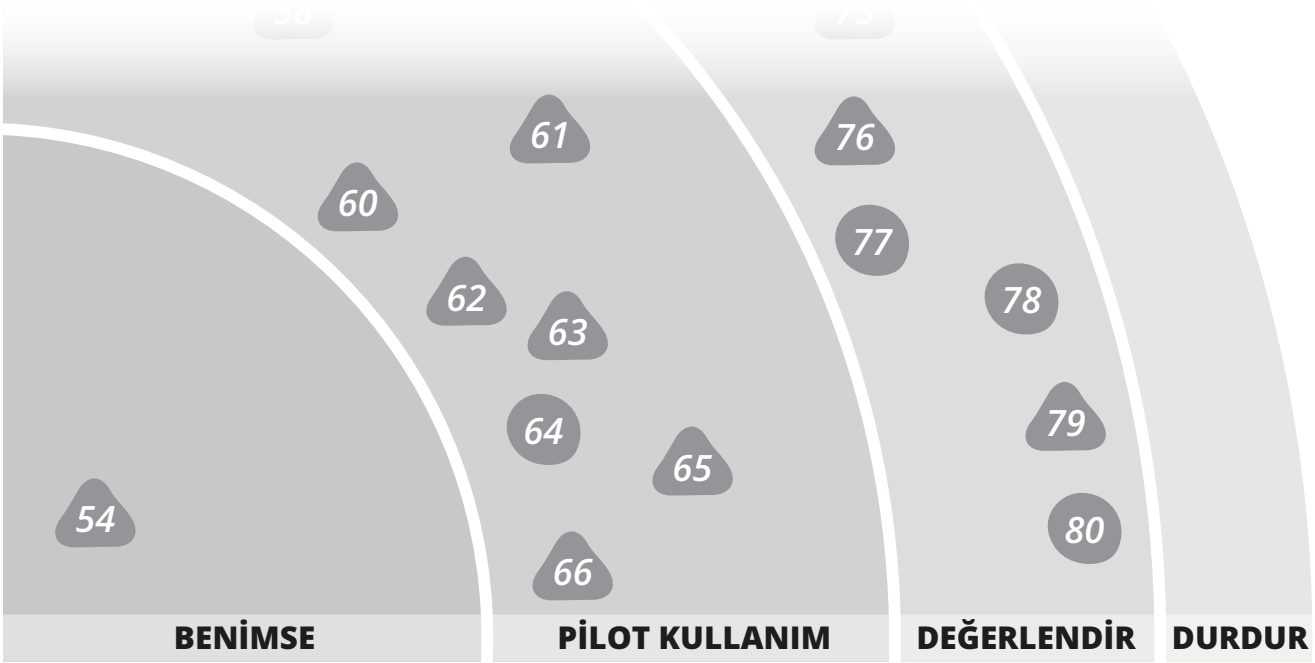
KATKIDA BULUNANLAR

Teknoloji Radarı aşağıdaki ThoughtWorks Teknoloji Danışma Kurulu tarafından hazırlanmıştır:

Rebecca Parsons (CTO)	Dave Elliman	Ian Cartwright	Rachel Laycock
Martin Fowler(Baş Bilim İnsanı)	Erik Doernenburg	James Lewis	Sam Newman
Anne J Simmons	Evan Bottcher	Jonny LeRoy	Scott Shaw
Badri Janakiraman	Fausto de la Torre	Mike Mason	Srihari Srinivasan
Brain Leke	Hao Xu	Neal Ford	Thiyagu Palanisamy

TEKNOLOJİ RADARI 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. Hedefimiz, 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 sık sık toplanarak, ThoughtWorks'ün küresel teknoloji stratejisini ve sektörümüzü ciddi olarak etkileyen teknoloji trendlerini tartışıyor. Radar, Teknoloji Danışma Kurulunun bu tartışmalarını CIO'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 çerçeveler bazında gruplandırıyor. Radarın birden fazla çeyrek dairede çıkabilen maddeleri için en uygun göründükleri çeyrek daireyi seçiyoruz. Bu maddeleri ayrıca şu anda onlar hakkındaki tavrımızı yansıtan dört halka halinde gruplandırıyoruz. Bu halkalar:



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

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

Kurumunuzu nasıl etkileyeceğini anlamak amacıyla takip etmekte fayda var.

Temkinli adım atın.

Yeni veya son radardan beri büyük değişim geçiren maddeler üçgen olarak, aynı kalan maddeler ise yuvarlak olarak verilir. Her bölümdeki detaylı grafikler maddelerin hareketlerini gösterir. Bu büyüklükteki bir belgeye sığdıramayacağımız kadar fazla madde ile ilgilendiğimiz için eski radardan kalan birçok maddeyi yenilerine yer açmak için siliyoruz. Fakat bu sildiğimiz ürünler artık umurumuzda değil anlamına gelmiyor.

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

RADAR

TEKNİKLER

BENİMSE
1. Kurulumun yayından ayrılması
2. Projelerden öncelikli olarak ürünler
3. Tehdit Modelleme

PİLOT KULLANIM
4. Ön yüzler için arka uç
5. Bug keşfetme ödülleri
6. Veri Gölü
7. Event Storming
8. Flux
9. Tekrarlanabilirlik filtresi
10. Sandboxing için iFrame'ler
11. Her şey için NPM
12. Phoenix Environments
13. Üretimde QA
14. Reaktif mimariler

DEĞERLENDİR
15. İçerik Güvenlik Politikaları yeni
16. Barındırılan IDE'ler
17. AB'de PII verilerinin barındırılması yeni
18. Sabitlerin izlenmesi
19. OWASP ASVS yeni
20. Sunucusuz mimari yeni
21. Unikernels yeni
22. Oyunun ötesinde Sanal Gerçeklik yeni

DURDUR
23. Tüm ekipler için tek bir CI instance yeni
24. Büyük Veri özentisi yeni
25. Gitflow
26. Yüksek performans özentisi / web ölçeği özentisi
27. SAFe™

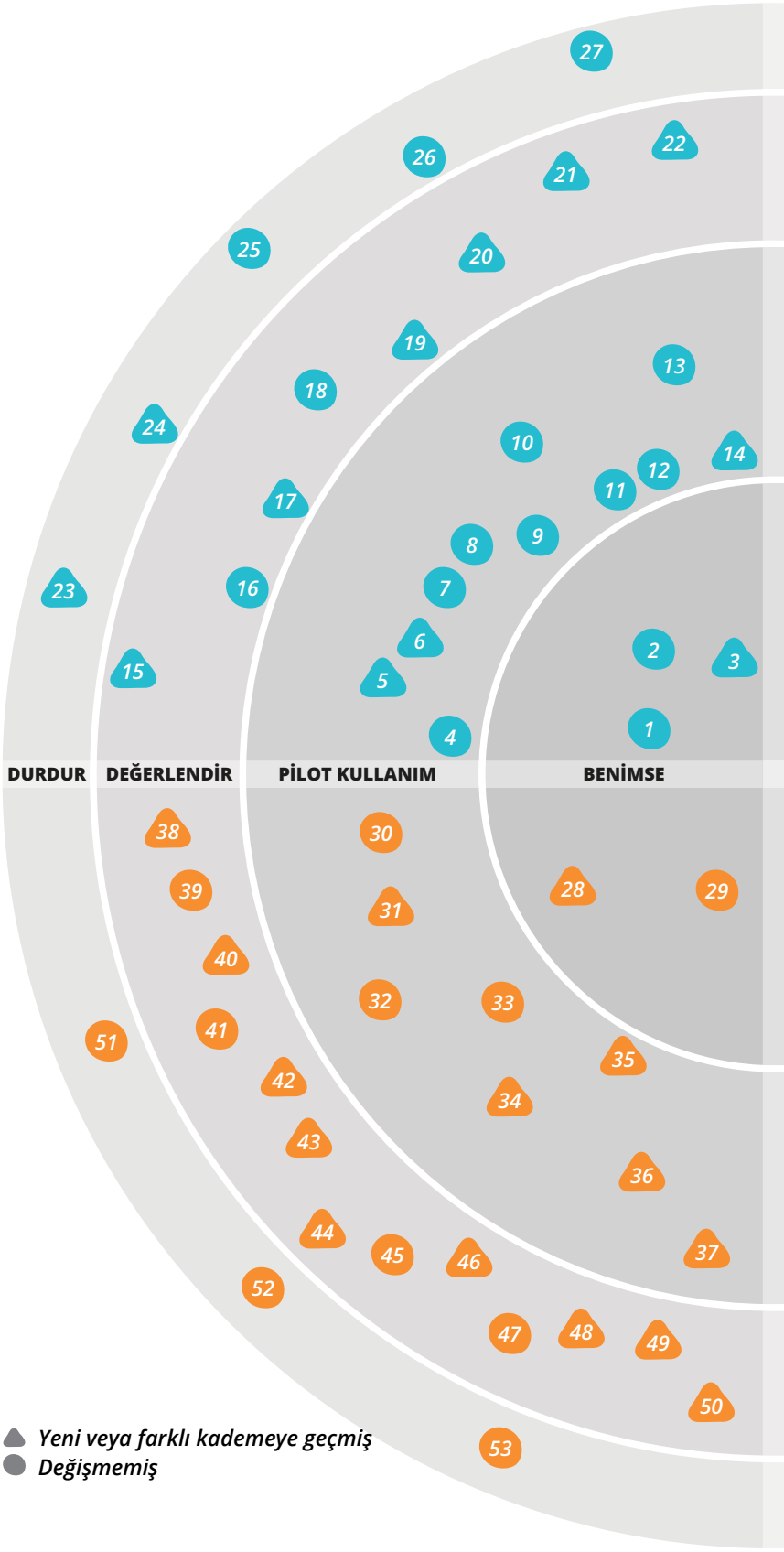
PLATFORMLAR

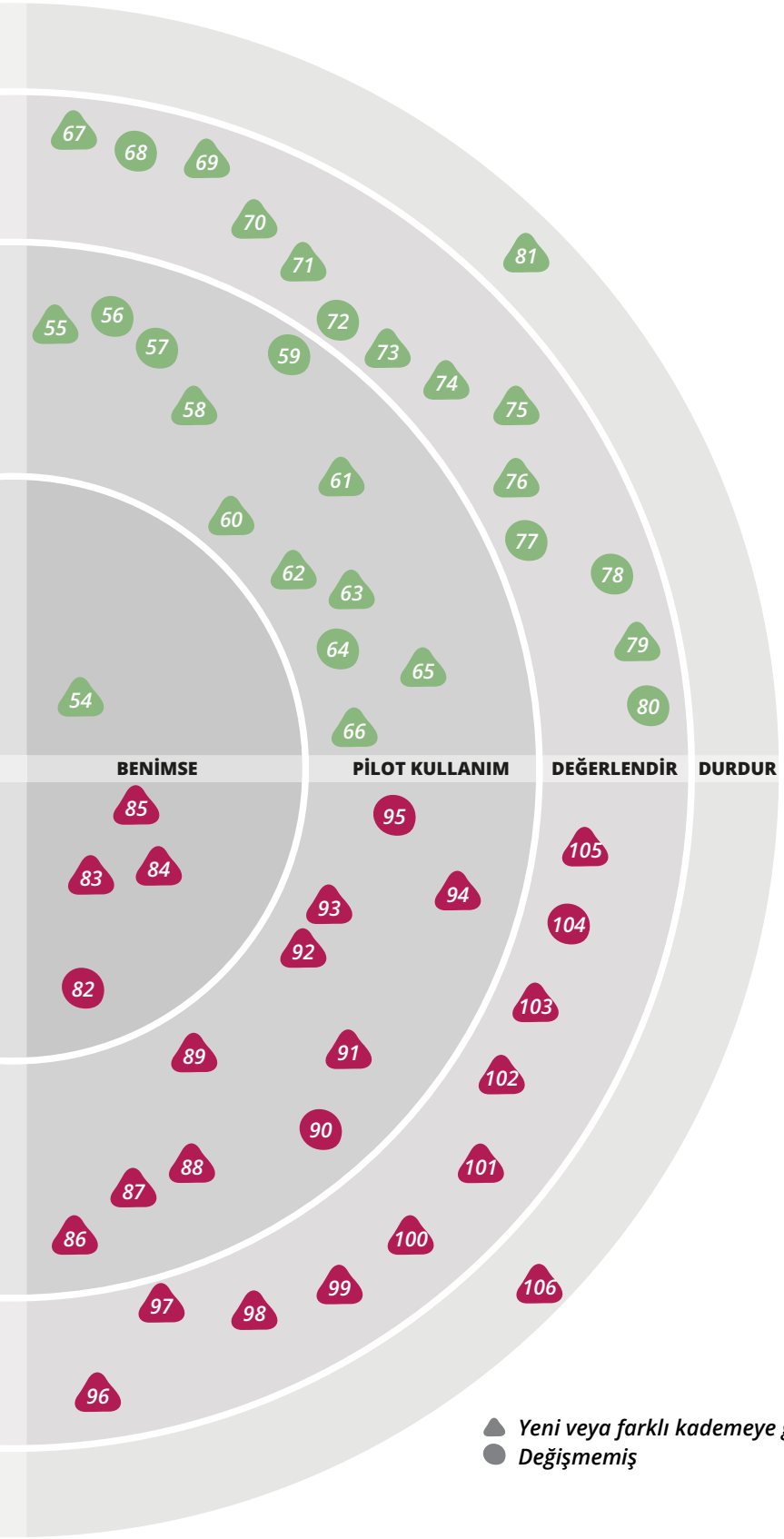
BENİMSE
28. Docker
29. TOTP İki Faktörlü Kimlik Doğrulama

PİLOT KULLANIM
30. Apache Mesos
31. AWS Lambda
32. H2O
33. HSTS
34. Kubernetes
35. Linux güvenlik modülleri
36. Pivotal Cloud Foundry yeni
37. Rancher

DEĞERLENDİR
38. Amazon API Gateway yeni
39. AWS ECS
40. Bluetooth Mesh yeni
41. Ceph
42. Deflect yeni
43. ESP8266 yeni
44. MemSQL yeni
45. Mesosphere DCOS
46. Nomad yeni
47. Presto
48. Realm yeni
49. Sandstorm yeni
50. TensorFlow yeni

DURDUR
51. Uygulama Sunucuları
52. Aşırı İddialı API Gateways
53. Yüzeysel Özel Bulut





▲ *Yeni veya farklı kademeye geçmiş*
● *Değişmemiş*

RADAR

ARAÇLAR

BENİMSE
54. Consul

PİLOT KULLANIM
55. Apache Kafka
56. Browsersync
57. Carthage
58. Gauge
59. GitUp
60. Let's Encrypt
61. Load Impact yeni
62. OWASP Dependency-Check yeni
63. Serverspec yeni
64. SysDig
65. Webpack yeni
66. Zipkin

DEĞERLENDİR
67. Apache Flink yeni
68. Concourse CI
69. Gitrob
70. Grasp yeni
71. HashiCorp Vault yeni
72. ievms
73. Jepsen yeni
74. LambdaCD yeni
75. Pinpoint yeni
76. Pitest yeni
77. Prometheus
78. RAML
79. Repsheet yeni
80. Sleepy Puppy

DURDUR
81. Bir dağıtım hattı olarak Jenkins yeni

DİLLER VE FRAMEWORK'LER

BENİMSE
82. ES6
83. React.js
84. Spring Boot
85. Swift

PİLOT KULLANIM
86. Butterknife yeni
87. Dagger yeni
88. Dapper yeni
89. Ember.js
90. Enlive
91. Fetch yeni
92. React Native
93. Redux yeni
94. Robolectric yeni
95. SignalR

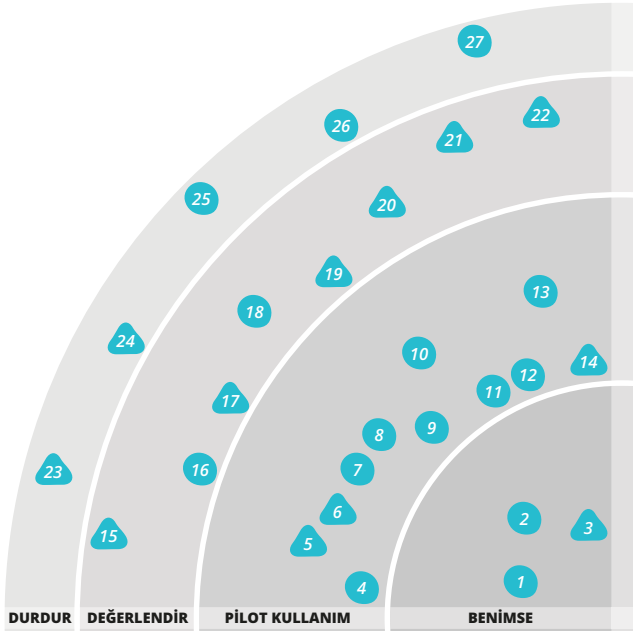
DEĞERLENDİR
96. Alamofire yeni
97. AngularJS
98. Aurelia yeni
99. Cylon.js yeni
100. Elixir
101. Elm
102. GraphQL yeni
103. Immutable.js yeni
104. OkHttp
105. Recharts yeni

DURDUR
106. JSPatch yeni

TEKNİKLER

Son aylardaki yüksek profilli güvenlik ihlallerinin sayısı dikkate alınırsa, yazılım geliştirme ekiplerinin güvenli yazılım üretme ve kullanıcılarının verilerini sorumlu bir şekilde yönetmeye önem vermeleri gerektiği konusunda ikna edilmeleri artık gerekmeyecektir. Ancak ekiplerin önünde dik bir öğrenme eğrisi bulunuyor ve organize suçlardan, devlete yönelik ajanlık faaliyetleri ve sistemlere sırf eğlence için saldıran gençlere kadar uzanan çok sayıda potansiyel tehdit altından kalkılması zor bir hale gelebilir. **Tehdit Modelleme (Threat Modeling)** potansiyel tehditleri erken ve gelişme aşamasında tanımlama ve sınıflandırma imkanı veren çok sayıda teknik sağlıyor. Her zaman tehditlerden önde gitmenin bir stratejinin sadece bir parçası olduğunun anlaşılması önemlidir. Örneğin, bir projenin kullanmakta olduğu teknolojilerin içerdiği yaygın risklere karşı fonksiyonlar ötesi güvenlik gereklilikleri oluşturmak gibi tekniklerle birlikte kullanıldığı ve otomatik güvenlik tarayıcıları da kullanıldığı zaman tehdit modelleme güçlü bir avantaj olabilir.

Şirketler ve önemli devlet kurumları dahil olmak üzere birçok kuruluşta **bug avı ödülleri** kullanılması yaygınlaşmaya devam ediliyor.



Bug avı programları ödüller veya itibar kazandırma karşılığında katılımcıları potansiyel olarak büyük zarar verebilecek zaafları keşfetmeye teşvik ediyor. Hacker One ve Bug crowd gibi şirketler kuruluşların bu süreci daha kolay yönetmelerine yardım etmek üzere hizmetler sunuyorlar ve bu hizmetlerin giderek daha fazla benimsendiğini görüyoruz.

Veri Gölü, büyük ölçüde işlenmemiş «ham» verilerden oluşan ve veri analizleri için kaynak olarak işlev gören sabit bir veri deposudur. Bu tekniğin yanlış kullanılabileceği açık olsa da biz müşterilerimizde bunu başarıyla kullandık ve deneme konumuna geçmesi için bir motivasyon oluşturduk. Operasyonel işbirlikleri için başka yaklaşımları kullanmaya ve veri gölü kullanımını raporlama, analiz ve veri marketlerine veri besleme konularıyla sınırlanmasını tavsiye etmeye devam ediyoruz.

Reaktif mimarilerin benimsenmesi ve başarılarının devam ettiğini ve reaktif dil uzantıları ve reaktif framework'lerin çok popüler olduğunu görüyoruz (Radar'ın bu sayısına bu türden birçok sinyali aldık). Özellikle kullanıcı ara yüzleri reaktif programlama tarzlarından büyük ölçüde yararlanıyor. Geçen sayıdaki uyarılarımız hâlâ geçerliliğini koruyor: Asenkron mesaj iletimine dayanan mimariler karmaşıklığa neden olur ve genel sistemin anlaşılmasını zorlaştırır - sadece program kodunu okuyup sistemin ne yaptığını anlamak mümkün olmaktan çıkar. Bu mimari tarzına girmeden önce sisteminizin performans ve ölçeklendirilebilirlik ihtiyaçlarını değerlendirmenizi tavsiye ediyoruz.

Karma ortamlardan öğeler çeken web siteleriyle işlem yaparken **İçerik Güvenlik Politikalarının (Content Security Policies)** güvenlik araç kitimize yararlı bir ilave olduğunu görüyoruz. Politika, varlıkların nereden gelebileceği (ve satır içi kod etiketlerine izin verilip verilmeyeceği) konusunda bir dizi kural tanımlar. Daha sonra tarayıcı, bu kuralları ihlal eden JavaScript, CSS veya resimleri yüklemeyi ve uygulamayı reddeder. Çıktı kodlama gibi iyi uygulamalarla birlikte kullanıldığında XSS saldırılarına karşı iyi bir hafifletici etki oluşturuyor. İlginç bir şekilde, JSON raporlarını gönderme uç noktası, Twitter'ın ISP'lerin sayfalarına HTML veya JavaScript enjekte ettiğini keşfetmesini sağladı.

BENİMSE
1. Kurulumun yayından ayrılması
2. Projelerden öncelikli olarak ürünler
3. Tehdit Modelleme

PİLOT KULLANIM
4. Ön yüzler için arka uç
5. Bug avı ödülleri
6. Veri Gölü
7. Event Storming
8. Flux
9. Tekrarlanabilirlik filtresi
10. Sandboxing için iFrame'ler
11. Her şey için NPM
12. Phoenix Environments
13. Üretimde QA
14. Reaktif mimariler

DEĞERLENDİR
15. İçerik Güvenlik Politikaları
16. Barındırılan IDE'ler
17. AB'de PII verilerinin barındırılması
18. Sabitlerin izlenmesi
19. OWASP ASVS
20. Sunucusuz mimari
21. Unikernels
22. Oyunun ötesinde Sanal Gerçeklik

DURDUR
23. Tüm ekipler için tek bir CI instance
24. Büyük Veri özentisi
25. Gitflow
26. Yüksek performans özentisi / web ölçeği özentisi
27. SAFe™

TEKNİKLER *devam ediyor*

Dünya genelinde birçok ülkede devlet kurumlarının özel kişisel tanımlama bilgilerine (PII) erişmek istediğini görüyoruz. AB'de en yüksek mahkeme Safe Harbor framework'ünü iptal etti, halefi Privacy Shield'in de iptal girişimiyle karşılaşması bekleniyor. Aynı zamanda, bulut bilişim kullanımı artıyor ve en büyük bulut tedarikçileri - Amazon, Google ve Microsoft - Avrupa Birliği içinde çok sayıda veri merkezi ve bölgesi sunuyor. Bu nedenle, global kullanıcı bazına sahip olanlar başta olmak üzere şirketlerin, en ileri gizlilik yasalarıyla korunan kullanıcılarının verileri için, **AB'de PII Barındırarak** bir güvenli liman oluşturmanın fizibilitesini değerlendirmelerini tavsiye ediyoruz.

Geliştirme döngüsü içerisinde güvenliği erken aşamalara yerleştiren geliştirme ekiplerinin sayısı arttıkça güvenlik risklerini sınırlamaya yönelik gerekliliklerin hesaplanması göz korkutucu bir iş gibi görünebilir. Bir uygulamanın karşılaşabileceği bütün riskleri tanımlamak için gerekli kapsamlı teknik bilgilere çok az sayıda kişi sahiptir ve ekipler nereden başlamaları gerektiğine karar vermeye çalışmaktan öteye gidemeyebilirler. OWASP'ın **ASVS** (Uygulama Güvenliği Doğrulama Standardı) gibi frameworklere güvenmek bunu kolaylaştırabiliyor. Biraz uzun olsa da, doğrulama, erişim kontrolü ve hataların ele alınması ve kaydedilmesi gibi fonksiyonların kategorize ettiği ve gerektiğinde gözden geçirilebilecek tam bir gereklilikler listesini içerir. Yazılımları kontrol etmek amacıyla test yapan kişiler için de yararlı bir kaynaktır.

Sunucusuz mimari uzun zamandır kullanılan uzun soluklu sanal makineler yerine, talep üzerine ortaya çıkan ve kullanıldıktan sonra hemen ortadan kalkan kısa vadeli bilişim gücünü getiriyor. Örnekler arasında Firebase ve AWS Lambda bulunuyor. Bu mimarinin kullanılması, güvenlik yaması ve SSH erişim kontrolü gibi bazı güvenlik kaygılarını hafifletebilir ve bilişim kaynaklarının çok daha etkin kullanılmasını sağlayabilir. Bu sistemlerin işletme maliyeti çok azdır ve dahili ölçeklendirme özellikleri olabilir (özellikle AWS Lambda için geçerlidir). Örnek olarak bir CDN veya S3 tarafından servis edilen statik varlıkları bulunan bir JavaScript uygulaması ile birlikte API Gateway ve Lambda tarafından servis edilen AJAX çağrıları verilebilir. Sunucusuz mimarilerin önemli faydaları olmakla birlikte sakıncaları da vardır: Servisler arasında kodların kurulumu, yönetimi ve paylaşılması daha karmaşıktır ve yerel veya çevrimdışı test etme imkansız değilse bile çok zordur.

Docker'ın benimsenmesi başta olmak üzere konteynır modelinin hakimiyetinin artmaya devam etmesi ile **Unikernel** alanındaki hızlı gelişmenin devam etmesine de dikkat çekmemizin yerinde olacağını düşünüyoruz. Unikernel'ler tek amaca yönelik kütüphane işletim sistemleridir. Üst düzey dillerden derlenebilen bu sistemler, doğrudan ticari bulut platformları tarafından kullanılan hipervizörler üzerinde çalışabilirler. Bu sistemler özellikle süper hızlı açılış süresi ve çok küçük olan saldırı yüzeyi alanı açısından Konteynırlara göre çeşitli avantajlar vaat ediyor. Birçoğu - Microsoft Research, Mirage OS ve HaLVM gibi - henüz araştırma-proje aşamasında bulunuyor ancak fikirlerin çok ilginç olduğunu ve sunucusuz mimari tekniğiyle çok iyi bir bütünlük oluşturduğunu düşünüyoruz.

Sanal gerçeklik fikri 50 yıldan fazla bir süredir gündemde ve bilişim teknolojisinde art arda gelen gelişmelerle birlikte birçok fikir heyecan yarattı ve keşfedildi. Şu anda bir dönüm noktasına yaklaştığımıza inanıyoruz. Modern grafik kartları, ayrıntılı ve gerçekçi sahneleri yüksek çözünürlükle sunmaya yeterli bilişim gücü sağlıyor ve aynı zamanda tüketiciye yönelik en az iki başlık (HTC Vive ve Facebook'un Oculus Rift'i) pazara çıkmak üzere. Makul fiyatlı olan bu başlıklar yüksek çözünürlüklü ekranlara sahip ve hareket izlemede algılanabilir gecikmeleri ortadan kaldırıyor. Bu gecikmeler daha önce baş ağrısı ve bulantı gibi sorunlara neden oluyordu. Başlıklar esas olarak video oyunu meraklılarını hedefliyor ancak, **Oyunların ötesinde sanal gerçeklik** için birçok olasılığa açık olacaklarına inanıyoruz ve özellikle Google Cardboard gibi low-fi yaklaşımlar giderek daha fazla farkındalık yaratıyor.

Tüm ekipler için tek CI (Sürekli Entegrasyon) instance yönetmenin daha kolay olduğu izlenimi olabilir çünkü bu onlara tek bir yapılandırma ve izleme noktası verir. Ancak bir kuruluş içindeki her ekip tarafından paylaşılan şişirilmiş bir instance birçok zarara yol açabilir. Yapım zaman aşımları, yapılandırma çelişkileri gibi sorunlar ve devasa yapım kuyruklarının daha sık ortaya çıktığını tespit ettik. Bu tek başarısızlık noktasının bulunması birçok ekibin çalışmasını kesintiye uğratabilir. Bu tuzaklar ile tek bir yapılandırma noktası bulunması arasındaki dengeyi dikkatle değerlendirin. Birden çok ekibin bulunduğu kuruluşlarda CI instance'larının ekipler tarafından dağıtılmasını, kurumsal kararların tek bir CI kurulumuna değil instance'ların seçimine ve yapılandırmasına dayanmasını tavsiye ediyoruz.

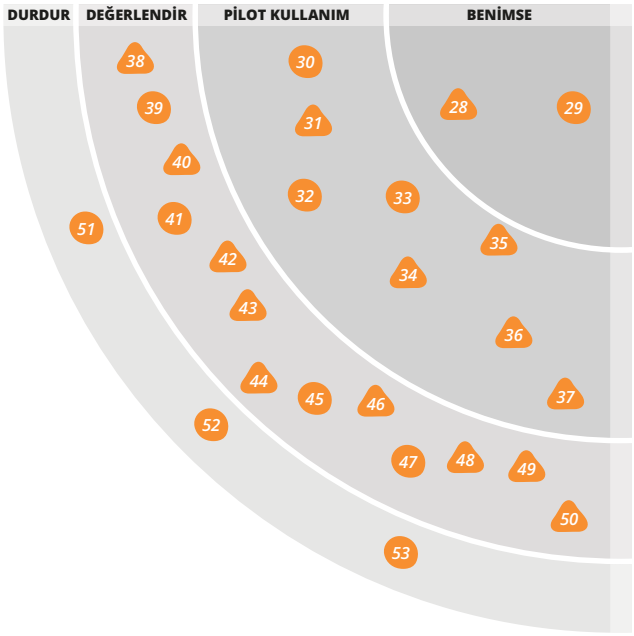
TEKNİKLER *devam ediyor*

İnsanların bizimle nasıl etkileşime girdiklerini anlamak için Büyük Verinin önemini uzun zamandır anlamış olmamıza rağmen, **Büyük Veri özentisiyle** ilgili alarm veren bir trendi fark etmiş bulunmaktayız: «aslında çok da büyük olmayan» Verileri yönetmek için karmaşık araçlar kullanan kuruluşlar. Dağıtık harita indirgeme algoritmaları büyük veri cluster'ları için kullanışlı bir tekniktir ancak gördüğümüz birçok veri cluster'ı kolaylıkla bir tek düğümlü ilişkisel veya grafik veri tabanına uyuyordu.

Bundan daha fazla veriniz varsa bile yapılacak en iyi şey ihtiyacınız olan veriyi almaktır. Bu veriler daha sonra böyle bir tek düğümde işlenebilir. Bu yüzden cluster'larınızı hızlandırmadan önce neyi işlemeniz gerektiğinin gerçekçi bir değerlendirmesini yapmanızı ve uygunsa - belki RAM de olabilir - basit seçeneği kullanmanızı tavsiye ediyoruz.

PLATFORMLAR

Bir araçtan karmaşık bir teknolojiler platformuna doğru evrilmekte olan **Docker** bizi heyecanlandırmaya devam ediyor. Docker görsel formatı geliştirme ile üretim arasında pariteye ulaşmayı kolaylaştırdığı ve kurulumları güvenilir hale getirdiği için geliştirme ekipleri Docker'ı seviyor. Mikro servis tarzı bir uygulamaya, kendi kendine yeten hizmetler için bir paketleme mekanizması olarak doğal bir uyum sağlıyor. İşletim cephesinde Docker'ın izleme araçları (Sensu, Prometheus, cAdvisor, vb.), orkestrasyon araçları (Kubernetes, Marathon, vb.) ve kurulum otomasyon araçlarına verdiği destek, bu platformun giderek olgunlaştığını ve üretimde kullanılmaya hazır olduğunu gösteriyor. Ancak bir konuda ihtiyatlı olmak gerekiyor: Docker ve Linux konteynırlarının «hafif sanallaştırma» olduğu yönünde yaygın bir görüş var ancak biz Docker'ı güvenli bir proses izolasyon mekanizması olarak kullanmayı tavsiye etmiyoruz yine de 1.10 versiyonunda kullanıcı isim alanları ve sec comp profillerinin konulmasına bu açıdan dikkat ediyoruz.



Ekiplerimiz **AWS Lambda**'yı keyifle kullanıyorlar ve sunucusuz mimarilerle deneyimlemek için kullanmaya başlıyorlar. Lambda'yı API Gateway ile birleştirerek görünmez altyapılı, ölçeklendirilebilirliği yüksek sistemler üretiyorlar. Java'yı Lambda fonksiyonları için kullanırken önemli sorunlar ve Lambda konteynırı çalıştırıldığında birkaç saniyeye kadar çıkabilen değişken gecikmeler yaşadık. Şimdilik JavaScript veya Python kullanmaya devam etmenizi tavsiye ediyoruz.

Kubernetes, Google'ın, giderek genel bir senaryo haline gelen bir makine cluster'ına konteynırlar kurma sorununa verdiği karşılığı oluşturuyor. Bu, Google'ın kendi içinde kullandığı bir çözüm değil ancak Google'da başlayan bir açık kaynak projesi olarak önemli sayıda harici katkı aldı. Önceki Radar'da Kubernetes'ten bahsettiğimizden beri, başlangıçta olumlu izlenimimiz teyit edildi ve müşterilerimizin üretimlerinde Kubernetes'in başarıyla kullanılmakta olduğunu görüyoruz.

Radar'ın önceki sayılarında **Linux güvenlik modüllerinin** değerini vurgulamış ve insanların geliştirme iş akışları kapsamında sunucu sertleştirmeyi düşünmelerinin nasıl sağlanabileceğinden söz etmiştik. Daha yakın geçmişte, LXC ve Docker konteynırlarının bazı Linux dağıtımlarında öntanımlı AppArmor profilleri ile gönderilmeye başladı ve birçok ekibi bu eserlerin nasıl çalıştığını anlamaya zorladı. Ekiplerin kendilerinin yaratmadıkları herhangi bir prosesi çalıştırmak için konteynır imajlar kullanmaları halinde, bu araçlar, onlara paylaşılan ana bilgisayar üzerinde kimin hangi kaynaklara erişim hakkının olacağı ve bu konteynırlı hizmetlerin sahip olduğu yetenekler konusundaki soruları değerlendirmekte ve erişim seviyelerinin yönetilmesinde muhafazakar olunmasına yardımcı oluyor.

BENİMSE
28. Docker
29. TOTP İki Faktörlü Kimlik Doğrulama

PİLOT KULLANIM
30. Apache Mesos
31. AWS Lambda
32. H2O
33. HSTS
34. Kubernetes
35. Linux güvenlik modülleri
36. Pivotal Cloud Foundry
37. Rancher

DEĞERLENDİR
38. Amazon API Gateway
39. AWS ECS
40. Bluetooth Mesh
41. Ceph
42. Deflect
43. ESP8266
44. MemSQL
45. Mesosphere DCOS
46. Nomad
47. Presto
48. Realm
49. Sandstorm
50. TensorFlow

DURDUR
51. Uygulama Sunucuları
52. Aşırı İddialı API Gateways
53. Yüzeysel özel bulut

PLATFORMLAR *devam ediyor*

PaaS alanında en son 2012'deki Cloud Foundry'de bahsi geçtiğinden bu yana çok fazla gelişme yaşandı. Açık kaynak çekirdeğinde farklı dağıtımlar mevcutken, **Pivotal Cloud Foundry'**nin sunduklarından ve bir araya getirdiği ekosistemden çok etkilendik. Yapılandırılmamış yaklaşım (Docker, Mesos, Kubernetes, vb.) ve Cloud Foundry ile diğerleri tarafından sunulan daha yapılandırılmış ve daha az esnek derleme paketleri tarzı arasındaki yakınsamanın devam etmesini beklerken, kurumların PaaS benimsemelerini sağlayacak kısıtlamaları ve gelişim hızını kabul etmeye hazır olmalarının faydalı olduğunu düşünüyoruz. Burada özellikle geliştirme ekipleri ile platform operasyonları arasındaki etkileşimin basitleştirilmesi ve standartlaştırılmasından kaynaklanan gelişim hızı dikkat çekiyor

Giderek gelişen Servis olarak Konteynır (CaaS) alanında, çok fazla gelişme görüyoruz ve bu alan, temel IaaS (Servis olarak Altyapı) ve daha az esnekliğe sahip PaaS (Servis olarak Platform) arasında faydalı bir seçenek sunuyor. **Rancher**, diğer bazı oyunculara göre daha az gürültü çıkartmasına karşın, üretim sırasında Docker kapsayıcılarını çalıştırarak sağladığı basitliği beğendik. Hem tek başına tam bir çözüm olarak hem de Kubernetes gibi araçlarla birlikte çalışabiliyor.

Amazon'un geliştiricilerin uygulama geliştirme ara yüzü (API) servislerini internet müşterilerine sunmalarını sağlayan **Amazon API Gateway** de trafik yönetimi, izleme, doğrulama ve yetkilendirme gibi olağan API kapısı özelliklerini içinde barındırıyor. Ekiplerimiz, bu servisi sunucusuz yapıların bir parçası olarak AWS Lambda gibi diğer AWS yeteneklerini yöneltmek için kullanıyor. Aşırı iddialı API Gateway'lerin ortaya çıkardığı zorlukları izlemeye devam ediyoruz, ancak bu aşamada Amazon'un çözümü bu sorunları ortadan kaldırmaya yeterli görünmüyor.

Akıllı cihaz kurulumlarının birçoğunda Wi-Fi bağlantısına ihtiyaç duyulurken, bir merkez ya da kapıya ihtiyaç duymayan **Bluetooth Mesh** şebekelerinin elde ettiği başarıyı da görüyoruz. Wi-Fi'dan daha iyi enerji kullanan ve ZigBee'den daha iyi akıllı telefon adaptasyonu sağlayan Blutooth LE, yerel cihaz alanı şebekelerini bağlamak için ilginç ve yeni yaklaşımlar sunan, kendini iyileştiren bir göz (mesh) kurulumu sağlıyor. Bluetooth SIG'den çıkacak olan resmi yaklaşımı hâlâ bekliyor olsak da şu ana kadar başarılı kurumlar gerçekleştirildi. Dağıtılmış bir şebekeyi ayakta tutacak bir altyapının olmamasını özellikle beğenmemize karşın kapı ve bulut servislerinin eklenmesiyle sistemin "ilerleyici bir şekilde güçlenmesi" seçeneğini de gündemde tutuyoruz.

Deflect; STK, aktivist ve bağımsız medya şirketlerini DDoS saldırılarına karşı koruyan bir açık kaynak servisi. Ticari CDN'e benzer şekilde, dağıtılmış ters sunucu önbelleklemesi kullanılıyor ve aynı zamanda sunucu OP adreslerini saklarken, yönetici URL'lerine dışarıdan erişimi engelliyor. Genellikle bağımsız seslerin haksız yere sansürlenmesi için kullanılan botnet'lerle (zombi şebekeler) mücadeleye özel önem veriliyor.

ESP8266 Wi-Fi mikrokontrolcüsü, sayıları giderek artan donanım hacker'ları arasında heyecan yarattı. Spesifik bir teknoloji inovasyonu olmaktan çok, düşük fiyat düzeyini küçük form faktörüyle birleştirerek, insanların aklında artık geleneksel donanım cihazlarıyla nelerin başarılmasının mümkün olduğu konusunda bir bükülme noktası yarattı. Başlıca özellikleri arasında Wi-Fi yetenekleri (istasyon, erişim noktası ya da her ikisini birlikte yapabiliyor), düşük güç, açık donanım, Arduino SDK programlama, Lua programlama, büyük topluluk desteği ve diğer IoT modüllere kıyasla daha düşük maliyet yer alıyor.

Moore Yasası'nın öngördüğü şekilde, bilgisayar sistemlerinin kapasitesini artırmaya, maliyetlerini ise düşürmeye devam ediyoruz ve böylece daha birkaç yıl önce mümkün görünmeyen yeni işleme teknikleri de olanaklar dahiline giriyor. Bu tekniklerden birisi de bellekte veri tabanı. Veri depolamak için yavaş diskler ya da nispeten yavaş SSD'ler kullanmak yerine, yüksek performans için bu verileri artık bellekte tutabiliyoruz. Bellekte veri tabanı uygulamalarından birisi olan **MemSQL**, yığın üzerinde yatak ölçeklenebilir olması ve SQL tabanlı sorgu diline benzerlik taşıması nedeniyle dikkat çekmeye başladı. MemSQL ayrıca veriyi bir depoda tutmak yerine eş zamanlı veri karşısında analiz için Spark ile bağlantı da kurabiliyor.

HashiCorp, ilginç bir yazılım olmayı sürdürüyor. Son olarak hiç olmadığı kadar kalabalık bir zaman planlayıcı / ortamında yarışan **Nomad** dikkatimizi çekmeyi başardı. En önemli avantajları arasında, konteynırlaştırılmış iş yükleriyle sınırlı kalmamasının yanı sıra çoklu veri merkezi çoklu bölge kurulumlarında da çalışabiliyor olması yer alıyor.

Realm yüksek performansa ulaşmak için kendi kalıcı motoruna sahip, mobil cihazlarda kullanım için tasarlanmış bir veri tabanı. Realm, SQLite ve Core Data'nın alternatifi olarak pazarlanıyor ve ekiplerimiz bu üründen memnun kaldı. Geçişlerin, Realm dokümantasyonunda olduğuna inandığınız kadar düzgün olmayabileceğini not etmek gerekir. Yine de, Realms bizi heyecanlandırdı ve bu ürüne bir bakmanızı öneriyoruz.

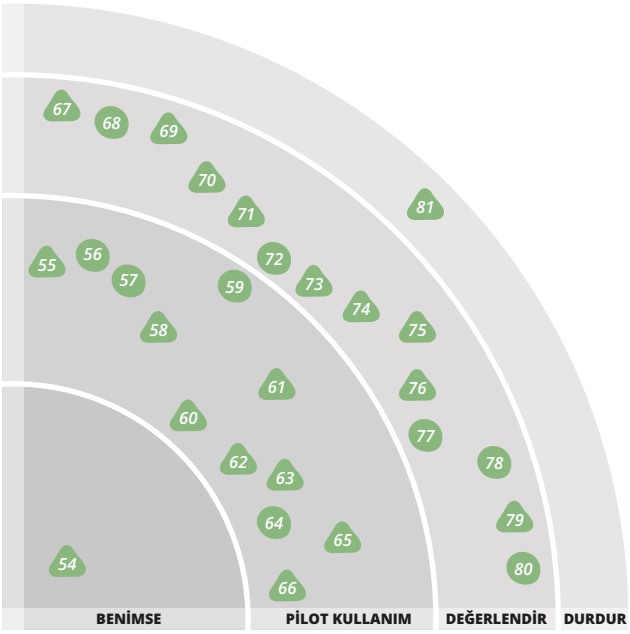
PLATFORMLAR *devam ediyor*

Bulut tabanlı işbirliği araçlarının avantajını kullanmak isteyen ancak istemeyerek büyük bir bulut sağlayıcısının "ürününe dönüşmek" istemeyenler için **Sandstorm** kendi barındırma potansiyeline sahip ilginç bir açık kaynak alternatifi sunuyor. Burada izolasyon yaklaşımı, konteynırlaştırmanın uygulama bazında değil doküman başına yapılması ve kum havuzunun güvenliğini artırmak için eklenen liste bazlı sistem çağrısı doğrulama özellikle dikkat çekiyor.

Google'ın **TensorFlow**'u ise araştırmadan üretime kadar her alanda kullanılabilecek ve geniş bir GPU bilgi işlem cluster'ına kadar mobil CPU'dan gelen bir donanım üzerinde çalışacak bir açık kaynak makine öğrenimi platformu. Derin öğrenme gerektiren algoritmaların uygulanmasını daha erişebilir ve daha kolay kılmasından dolayı önemli bir platform. Yarattığı heyecana rağmen, TensorFlow, algoritma açısından yeni bir şey değil: Bu tekniklerin tamamı bir süredir akademi üzerinden genel kullanıma açık halde bulunuyor. Ayrıca, birçok işletmenin temel öngörücü analizleri bile yapmadığını ve derin öğrenmeye sıçramanın birçok veri setinin anlaşılmasına yardımcı olma ihtimalinin düşük olduğunu kabul etmek önem taşıyor. Doğru sorun ve veri setine sahip olanlar içinse TensorFlow faydalı bir araç kiti.

ARAÇLAR

Hem DNS hem de HTTP tabanlı servis bulma aracı **Consul**'ü Adopt'a taşıdık. Kayıtlı servisler için kişiselleştirilebilir sağlık kontrolleri sunarak, sağlıksız örneklerin uygun şekilde işaretlenmesini sağlaması nedeniyle diğer bulma aletlerinin ötesine geçiyor. Consul ile çalışacak daha fazla aracın ortaya çıkması da bunu daha da güçlendirdi. Consul Template, yapılandırma dosyalarının Consul'den gelen bilgilerle dolmasını sağlayarak, mod_proxy kullanarak müşteri tarafındaki yük dengelemesi gibi işlerin yapılmasını da kolaylaştırıyor. Docker dünyasında registrator, Consul'da belirdikleri şekliyle Docker konteynırlarını otomatik olarak ve büyük bir rahatlıkla kaydedebiliyor ve böylece konteynır tabanlı kurumları yönetmek çok daha kolaylaşıyor. Yine de böyle bir araca ihtiyacınız olup olmadığını ya da daha basit bir şeyin işinizi görüp görmeyeceğine ilişkin uzun ve detaylı bir şekilde düşünmeniz gerekiyor. Ancak yine de kararınız servis bulma kullanma yönünde olursa Consul'ü tercih etmeniz hata olmaz.



Birçok kurum, ölçekli olayların değişmez sıralanmasıyla ilgili bilgiyi yakalayan yeni veri mimarilerini inceliyor. **Apache Kafka**, çok sayıda bağımsız ve hafif tüketiciye yayımlama talimatlı olay beslemeleri için bir çözüm sunan açık kaynaklı bir mesajlaşma framework'ü olarak ivme kazanmaya devam ediyor. Kafka yapılandırmasını yapmak hiç de kolay olmasa da ekiplerimiz bu framework ile olumlu deneyimler yaşadıklarını bildiriyor.

Gauge, hafif bir platformlar arası test otomasyon aracı. Teknik özellikleri serbest form Markdown olarak yazıldı ve böylece test vakaları, daha sıklıkla kullanılan ancak kısıtlayıcı "given-when-then" formatı kullanmak yerine iş dilinde yazılabilir hale geldi. Dil ve IDE desteği, tek çekirdek uygulamasına eklenti olarak uygulanıyor ve bu da testi yapanların otomatik tamamlama ve refactoring gibi güçlü yeteneklerle birlikte ekibin geri kalanıyla aynı IDE'leri kullanmasına olanak tanıyor. ThoughtWorks tarafından açık kaynaklı geliştirilen bu araç ayrıca tüm desteklenen platformlarda sıra dışı paralel uygulamaları da destekliyor.

Let's Encrypt ilk kez Radar'ın geçen sayısında yer aldı ve Aralık 2015'ten bu yana bu proje, beta sürümünü özelden kamuya açarak, kullanıcıların denemek için davetiye alma gerekliliğini ortadan kaldırdı. Let's Encrypt, web sitelerini güvenli hale getirmenin yollarını arayan daha geniş bir kullanıcı kitlesine sertifikaları almak ve yönetmek için daha basit bir mekanizmasına erişim sağlıyor. Ayrıca, güvenlik ve gizlilik açısından ileriye dönük önemli bir adımın atılmasını da destekliyor. Bu eğilim ThoughtWorks'de hâlihazırda başlamış bulunuyor ve projelerimizin birçoğunda artık Let's Encrypt tarafından onaylanmış sertifikalar yer alıyor.

Load Impact, 1,2 milyon eş zamanlı kullanıcıya kadar yüksek gerçekçiliğe sahip yükler yaratabilen bir SaaS yük test aracı. Chrome eklentisi kullanarak web etkileşimleri kaydedebilir ve tekrar oynatabilirsiniz, mobil ya da masaüstü, kullanıcıları için şebeke bağlantılarının benzerini kurabilir ve dünya çapında sayısı10'a kadar çıkabilen farklı noktadan yük oluşturabilirsiniz. Her ne kadar bu, kullandığımız talebe bağlı yük test aracı olmasa da - BlazeMeter'ı da seviyoruz - ekiplerimiz Load Impact için oldukça heyecan duyuyor.

BENİMSE
54. Consul

PİLOT KULLANIM
55. Apache Kafka
56. Browsersync
57. Carthage
58. Gauge
59. GitUp
60. Let's Encrypt
61. Load Impact
62. OWASP Dependency-Check
63. Serverspec
64. SysDig
65. Webpack
66. Zipkin

DEĞERLENDİR
67. Apache Flink
68. Concourse CI
69. Gitrob
70. Grasp
71. HashiCorp Vault
72. ievms
73. Jepsen
74. LambdaCD
75. Pinpoint
76. Pitest
77. Prometheus
78. RAML
79. Repsheet
80. Sleepy Puppy

DURDUR
81. Kurulum ardışık düzeni olarak Jenkins

ARAÇLAR *devam ediyor*

Birçok yazılım geliştiricisinin hayatını kolaylaştıran kütüphane ve araçlarla dolu bir dünyada, bunlardaki güvenlik açıkları artık gözle görülür bir hale geldi ve bunları kullanan uygulamalarda yarattığı kırılganlık yüzeyi de artış gösterdi. **OWASP Dependency-Check** kamuoyuna açıklanmış kırılganlıklar olup olmadığını kontrol ettikten sonra bu tarz kırılganlıklar veri tabanını sürekli güncelleyen yöntemler kullanarak, koddaki potansiyel güvenlik sorunlarını otomatik olarak tespit ediyor. Dependency-Check, (bizim de başarıyla kullandığımız) Java ve .NET'in yanı sıra Ruby, Node.js ve Python'da bu doğrulamayı otomatik olarak yapan bazı arayüz ve eklentilere sahip.

Geçmişte, konfigürasyon testi tavsiye edilen tekniklerin arasında sıralamıştık ve bu sayıda bu testlerin yapılması için popüler bir araç olan **Serverspec'ten** bahsetmek istiyoruz. Her ne kadar bu yeni bir araç olmasa da daha fazla sayıda farklı rolleri içeren teslimat ekiplerinin altyapı tedariki konusunda sorumluluk almaya başlamasıyla birlikte bu aracın kullanımının da arttığını görüyoruz. Serverspec, Ruby kütüphanesi RSpec üzerine kuruldu ve sunucu yapılandırmasının doğru olup olmadığının anlaşılması için kapsamlı bir yardımcı setiyle birlikte sunuluyor.

Webpack JavaScript modülü paketleyicisi tercihimiz olarak konumunu iyice güçlendirdi. Giderek büyüyen yükleyici listesiyle birlikte tüm statik varlıklarınız için tekil ek bağımlılık ağacı sunarak, JavaScript, CSS, vb'lerin esnek bir şekilde değiştirilmesine ve tarayıcıya neyin ne zaman gönderilmesi gerekliliklerini azaltıyor. Özellikle dikkat çeken özelliği AMD, CommonJS ve ES6 modülleri arasında sorunsuz bir entegrasyon sağlaması ve tarayıcı uyumluluğunun önceki versiyonlarına (Babel kullanarak) sorunsuz dönüştürülmesi. Ekiplerimizin büyük bir kısmı, benzer bir alanı kapsayan ancak Node.js modüllerinin müşteri tarafında kullanımının sağlanmasına daha çok odaklanan Browserify'ya da değer veriyor.

Zipkin'deki gelişmeler de hızlı bir şekilde devam ediyor ve 2015'in ortalarından bu yana GitHub'daki openzipkin/zipkin organizasyonuna geçiş yaptı. Artık Python, Go, Java, Ruby, Scala ve C# için bağlamalar ve hızlıca işe koyulmak isteyenler için Docker imgeleri de mevcut. Yine de bu aracı seviyoruz. Bu aracın kullanımı konusunda aktif ve giderek büyüyen bir topluluk mevcut ve uygulama da giderek kolaylaşıyor. Birçok mantıksal isteğin uçtan uca gecikme süresinin ölçülmesini sağlayan bir yönteme ihtiyacınız varsa, Zipkin güçlü bir seçenek olmayı sürdürüyor.

Apache Flink toplu ve akan veriyi islemede kullanılan ölçeklenebilir ve dağıtık, yeni jenerasyon bir platformdur. Temelinde, veri akışı motoru yer alıyor. Ayrıca tablo formatında (SQL benzeri), grafik işleme ve makine öğrenme operasyonlarını da destekliyor. Apache Flink, akan veri işleme için özellik açısından zengin olmasıyla dikkat çekiyor. Olay zamanı, zengin akış üzerinde çerçeveli operasyonlar, hata toleransı ve bir-kerelik semantik. Her ne kadar henüz 1.0 versiyonu çıkmamış olsa da akıtma işleme, bellek yönetimi, durum yönetimi ve yapılandırma kolaylığı sayesinde ciddi bir topluluğun ilgisini çekmeyi başardı.

Saldırganlar, AWS kimliklerini bulmak için kamuya açık GitHub veri havuzlarını taramak ve Bitcoin madenciliği ya da diğer kötü emelleri için EC2 örneklerini kullanmak amacıyla otomatik yazılımlar kullanmayı sürdürüyor. Her ne kadar şifre ve kod depolarına ulaşım sağlayan erişim jetonları gibi sırların güvenli bir şekilde depolamak için git-cyrpt ve Blackbox gibi araçların benimsenmesi giderek artıyor olsa da bu tarz sırların güvensiz bir şekilde depolanması uygulaması hâlâ yaygın olarak yapılıyor. Ayrıca, proje sırlarının yanlışlıkla geliştiricilerin kişisel depolarına girdiğini görmek de nadir bir durum olmaktan çıktı. **Gitrob**, sırların açığa çıkmasının yaratacağı hasarı en az düzeye indirmeye yardımcı olabilir. Kurumun GitHub veri depolarını tarayarak, veri deposuna gönderilmemesi gereken hassas bilgileri içerebilecek tüm dosyaları işaretliyor. Bu aracın en yeni sürümünde bazı sınırlamalar mevcut. Sadece kamuya açık GitHub kuruluşlarını ve üyelerini taramak için kullanılabiliyor ve dosyaların içeriklerine bakmıyor, tüm taahhüt tarihini gözden geçirmiyor ve çalıştığı zamanlarda her defasında tüm veri depolarının bütünü tarıyor. Bu sınırlamalara rağmen, çok geç olmadan ekiplerin uyarılmasına yardımcı olacak faydalı ve reaktif bir araca dönüşebiliyor. Talisman gibi proaktif araçlar için tamamlayıcı bir yaklaşım olarak görülmelidir.

JavaScript'in **Grasp** adı verilen komut satırı refactoring aracını gördüğümüzde, hep birlikte aklımız başımızdan gitti. Zengin bir seçiciler kümesi sunarak ve soyut sintaks ağacına karşı işleyerek, sed ve grep sayesinde önemsiz olmanın ötesine geçiyor. Bu, JavaScript'i birinci sınıf bir dil olarak ele alma yönünde süregelen yolculuğumuzdaki araç kitine faydalı bir ek sunuyor.

Sırları güvenli bir şekilde yönetmek, projelerde giderek önem kazanan bir konu haline dönüşüyor. Sırları ya da ortam değişkenlerini içeren bir dosya tutmayı öngören eski fikirler, özellikle uygulamaların sır katmanlarına erişimin gerektiği durumlara sahip mikroservis ya da mikrokonteynır ortamları gibi çoklu uygulamalara sahip ortamlarda yönetmesi zor bir yönteme dönüşüyor. **HashipCorp Vault**, birleşik bir arayüz üzerinden sırlara güvenli erişim için mekanizmalar sunarak bu sorunu çözmeye çalışan umut verici bir araç. Şifreleme ve bilinen araçlar için otomatik olarak sırlar üretme gibi hayatı kolaylaştıran bazı özelliklere sahip.

ARAÇLAR *devam ediyor*

NoSQL veri depolarının kullanımını artması ve süreklilik konusunda çok dilli yaklaşımların popülerliğinin yükselmesi sonucunda ekiplerimizin de önünde artık verilerini depolama alanında çok sayıda seçenek yer alıyor. Bu durum birçok avantajı beraberinde getirirken, kırılgan ağlarla ürün davranışı bazı durumlarda ürün geliştiricileri tarafından bile çok iyi anlaşılmayan gizli (ve bazen çok da gizli olmayan) sorunlar çıkarabiliyor. **Jepsen** araç takımı ve beraberinde sunulan blogu, farklı veri tabanlarının ve kuyruk teknolojilerinin olumsuz koşullar altında nasıl tepki verdiklerini anlamak isteyenlere fiili bir referans haline dönüştü. Daha da önemlisi, işlemlerde müşteriyi de dahil eden test yaklaşımı, mikroservis geliştiren birçok ekip için olası hata modlarını mercek altına alıyor.

LambdaCD, ekiplere Clojure'da Sürekli Dağıtım ardışık düzenlerinin tanımlanması için bir yöntem sunuyor. 'Kod halinde(veya olarak) Altyapi'nin faydalarini 'Surekli Dagitim' sunucularina getirmis oluyor. Bu faydalari kaynak kontrol yonetimi, birim testi, refactoring ve kodun yeniden kullanimi olarak soyleyebiliriz. "Kod olarak ardışık düzenler" alanında, LambdaCD, hafif, bağımsız ve tamamen programlanabilir olmasıyla dikkat çekiyor ve ekiplerin aynı kodlarıyla yaptıkları gibi kendi ardışık düzenleriyle çalışmalarına olanak tanıyor.

Anka Sunucu veya Anka Ortam teknikleri kullanan ekipler, Uygulama Performans Yönetimi (APM) araçlarından çok fazla destek bulabilmiş değil. Uzun süre çalışan, sınırlı miktarda kutuya sahip lisanslama modelleri ve kısa ömürlü donanımla çalışma zorluğu, genellikle değer kazandırmaktan çok sıkıntı çıkardıkları anlamına geliyor. Ancak, dağıtık sistemlerin izlenmesi gerekiyor ve bir aşamadan sonra ekipler bir APM aracına ihtiyaçları olduğunun farkına varıyorlar. Bu alanda geliştirilen bir açık kaynak aracı olan **Pinpoint**'in AppDynamics ve Dynatrace karşısında bakılmaya değer bir alternatif olduğunu düşünüyoruz. Zipkin gibi diğer hafif açık kaynak araçlarını kullanarak önemli bir mesafe kat edebileceğinizi düşünüyor olsak da pazarda bir APM arıyorsanız Pinpoint bakılmaya değer.

Pitest, mutasyon test tekniğini kullanan bir Java test kapsamı analiz aracı. Geleneksel test kapsamı analizleri, testleriniz tarafından gerçekleştirilen satır sayısını ölçmeye çalışır. Bu nedenle yalnızca kesinlikle testten geçirilmemiş kodu tanımlayabiliyor. Buna karşın mutasyon testi ise test kodunuz tarafından gerçekleştirilen ve yine de içinde genel hatalar barındırabilen satırların kalitesini test etmeye çalışıyor. Bazı sorunlar bu şekilde belirlenebiliyor ve böylece ekibin de sağlıklı bir test takımı ölçmesine ve geliştirmesine yardımcı oluyor.

Bu tarz araçların önemli bir bölümü yavaş ve kullanımı zor olurken, Pitest daha iyi performans gösteriyor, kurulumu kolay ve aktif bir şekilde destekleniyor.

Botlar kullanılarak web uygulamalarına yapılan saldırılar gün geçtikçe daha karmaşık bir hal alıyor. **Repsheet** projesinin amacı bu ajan yazılımları ve hareketlerini tespit etmektir. Apache ve NGINX sunucularıyla birlikte kullanılabilen bu eklenti, kullanıcının aktivitelerini kaydediyor, önceden ya da kullanıcılar tarafından tanımlanmış ajan yazılımların izini sürüyor ve sonra bu saldırgan yazılımları engellemek de dahil, önlem alınabilmesini sağlıyor. Eklentinin ajan yazılımları güncel olarak göstermesi sayesinde, ekip üyeleri botlar nedeniyle oluşan tehditleri yönetebiliyor. Böylece ekiplerin güvenlik açıkları konusundaki farkındalığı artırılmış oluyor. Sıklıkla görülmeyen ancak gerçek bir problem olan bot kaynaklı saldırıları çözen güzel bir basit araç örneği olduğu için bunu beğendik.

Biz de rakip bir araç geliştirdiğimiz için burada riskli bir konumda olduğumuzu biliyoruz ancak uzun yıllardır devam eden bir problemi ele almamız gerektiğini hissediyoruz. Cruise Control ve Jenkins gibi Sürekli Entegrasyon araçları yazılım geliştirme için çok değerlidir ama yazılım geliştirme süreciniz karmaşıklaştıkça Sürekli Entegrasyonun ötesinde bir şey gerekiyor, o da dağıtım hattının oluşturulması. Eklentiler yardımıyla **Jenkins'i bir dağıtım hattı** olarak kullanmak isteyenlerle sıklıkla karşılaşıyoruz ancak deneyimlerimiz bunun hemen karışıklığa yol açtığını gösteriyor. Jenkins 2.0 "Pipeline as Code"u kullanıma sunmuş olsa da ardışık düzenleri eklentileri kullanarak modellemeye devam ediyor ve en önemli Jenkins ürünü ardışık düzenleri doğrudan modelleme konusunda sınıfta kalıyor. Bizim tecrübelerimize göre, kurulum ardışık düzenlerini birinci sınıf şekilde temsil etmek üzere yazılan araçlar diğerlerine göre çok daha uygundur ve bizi CruiseControl yerine GoCD'yi seçmeye iten en önemli sebep de budur. Günümüzde ConcourseCI, LambdaCD, Spinnaker, Drone ve GoCD'nin de aralarında bulunduğu pek çok ürünün kurulum ardışık düzenlerini içerdiğini görüyoruz.

DİLLER VE FRAMEWORK'LER

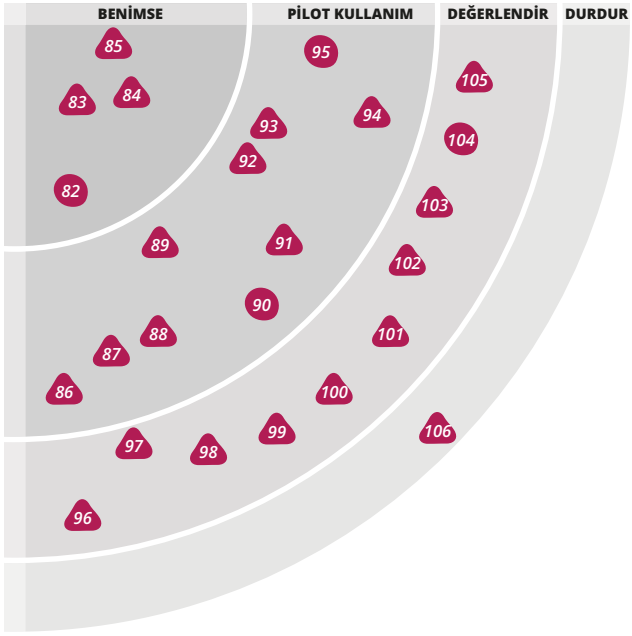
Çığ gibi gelen ön kullanıcı JavaScript framework'leri arasında **React.js** reaktif veri akışı etrafında geliştirilen tasarımıyla ön plana çıkıyor. Tek yönlü veri bağlamasına izin vermesi, oluşturma mantığını büyük oranda hafifletiyor ve diğer framework'ler ile yazılmış olan uygulamalarda sıklıkla ortaya çıkan sorunların önemli bir kısmını bertaraf ediyor. React.js'nin sağladığı faydaları sayısı giderek artan küçük ya da büyük birçok projede görebilirken, aynı zamanda AngularJS gibi diğer sevilen framework'lerin durumu ve geleceği hakkında endişemiz sürüyor. Bu da React.js'nin JavaScript framework'leri arasında doğal tercihimiz haline gelmesine neden oldu.

Spring Boot'ta karmaşıklık ve bağımlılıkları azaltmaya yönelik çok sayıda çalışma yapıldı ve bu da daha önce belirttiğimiz kaygıları büyük oranda ortadan kaldırıyor. Spring ekosisteminde yaşıyorsunuz ve mikroservislere geçiş yapıyorsunuz, Spring Boot artık bariz tercih haline geldi. Springland'de olmayanlar için ise Dropwizard ciddi bir şekilde değerlendirmeye tabi tutulmaya layık görünüyor.

Swift artık Apple ekosistemindeki geliştirmeler için doğal tercihimiz haline geldi. Swift 2 sürümüyle birlikte kullanılan dil de birçok proje için gereken istikrar ve performansı sağlayacak olgunluk düzeyine yaklaştı. iOS gelişimini destekleyen SwiftyJSON, Quick gibi çok sayıda kütüphane artık Swift'e taşındı ve diğer uygulamalarında bunu izlemesi gerekiyor. Swift artık açık kaynaklı ve iOS'te geliştirme süreçlerinin sürekli iyileştirilmesi konusunda kararlı olan bir geliştirici topluluğunun oluşmaya başladığını görüyoruz.

Butterknife bir alan ve yöntem bağlayıcı bir view-injection kütüphanesi. İsteğe bağlı obje, görüntüleme ve dinleyicinin eklenmesine olanak tanıyarak, Android gelişimi için daha az birleştirici kod sayesinde daha anlaşılır bir kod oluşturulmasını sağlıyor. Butterknife ile birlikte çoklu görüntülemeler, XML yapılandırmalara fazla bağımlı olmadan görüntülemelere eş zamanlı olarak uygulanan ortak eylemlerle liste ya da sıralama olarak gruplanabiliyor.

Android tabanlı uygulamalara olan ihtiyacın artmasıyla birlikte, Dagger tamamen statik, derleme zamanı bir bağımlılık enjeksiyon framework'ü sunuyor. **Dagger**'ın tamamen oluşturulmuş uygulaması ve yansıma tabanlı çözümlere



bağımlı olmaması birçok performans ve geliştirme sorunun önüne geçerek, Android gelişimine uygun bir hale geliyor. Dagger ile çağrı yığınının tamamı önlem ve yaratım süreçlerine açık hale geldiğinden dolayı kolay hata ayıklama sayesinde tam bir izlenebilirlik sağlıyor.

Dapper .NET türleri için minimal ve hafif bir ORM. Sizin için SQL sorguları yazmak yerine, Dapper SQL sorgularını dinamik objelere haritalıyor. Her ne kadar yeni bir marka olmasa da Dapper, zaman içerisinde .NET alanında çalışan ThoughtWorks ekiplerinin takdirini kazandı. C# programcısı için, ilgili sorguları objelere haritalandırma angaryalarının bazılarını ortadan kaldırırken, SQL ya da depolanan prosedürler üzerinde tam kontrolün devam etmesini sağlıyor.

Ember.js proje deneyimlerini temel alarak daha da destek verecek şekilde geliştirildi ve JavaScript uygulama framework'leri alanında güçlü bir rakip haline dönüştü. Ember, AngularJS gibi diğer framework'lere kıyasla daha az sürpriz içermesi nedeniyle geliştirici deneyimi açısından övgüler topladı. Ember CLI yapım işleme, kuralların yapılandırmadan önce gelmesi yaklaşımı ve ES6 desteği de ayrıca olumlu geri dönüşler almasını sağladı.

BENİMSE
82. ES6
83. React.js
84. Spring Boot
85. Swift

PİLOT KULLANIM
86. Butterknife
87. Dagger
88. Dapper
89. Ember.js
90. Enlive
91. Fetch
92. React Native
93. Redux
94. Robolectric
95. SignalR

DEĞERLENDİR
96. Alamofire
97. AngularJS
98. Aurelia
99. Cylon.js
100. Elixir
101. Elm
102. GraphQL
103. Immutable.js
104. OkHttp
105. Recharts

DURDUR
106. JSPatch

DİLLER VE FRAMEWORK'LER *devam ediyor*

Ekiplerimiz, JQuery ya da ham XHR'den uzaklaşarak, uzaktan JavaScript çağrılarına ve özellikle de **Fetch** polifili tercih etmeye başladı. Anlamsallıklar benzer olsa da taahhütlere daha açık destek veriyor ve CORS desteğini de sağlıyor. Bunu yeni fiili yaklaşım olarak görüyoruz.

React Native ile hızlı çoklu platform mobil geliştirme alanındaki başarıların devam ettiğini gözlemliyoruz. Sürekli geliştirmelerden geçtiği için ortaya çıkan bazı sorunlara rağmen, yerli ve yerli olmayan kod ve görüntüleme, hızlı geliştirme döngüsü (anlık yeniden yükleme, Chrome'da sorunları giderme, Flexbox tasarımı) arasındaki önemsiz entegrasyonun avantajları ve React tarzındaki genel büyüme görüşümüzü olumlu etkiliyor. Diğer birçok framework'te olduğu gibi, kodu iyi yapılandırmak için özen gösterilmesi gerekiyor ancak Redux gibi özenli bir aracın kullanılması bu noktada oldukça yardımcı oluyor.

Redux ekiplerimizin önemli bir kısmının müşteri tarafı uygulamalarındaki durumu yönetmek için geliştirdikleri düşünce tarzlarını yeni bir çerçeveye oturtmasına yardımcı olan mükemmel ve olgun bir araç. Flux tarzında bir yaklaşım benimseyerek, mantığa oturtması kolay olan gevşek bağlantılı durum makinesi yapısına izin veriyor. Ember ve React gibi tercih ettiğimiz JavaScript framework'lerinden bazılarına iyi bir eşlikçi olduğunu keşfettik.

Android uygulaması geliştirme dünyasında **Robolectric** teknik topluluğumuz içerisinde farklı ekipler tarafından kullanılan bir birim test etme framework'ü. Android bileşenlerine uzayan ya da bunlarla doğrudan etkileşime geçen ve JUnit testlerini destekleyen gerçek birim testleri yazılmasına olanak tanıyanlar arasında en iyi seçeneği sunuyor. Ancak, Android SDK'nin bir uygulaması olmasından dolayı Robolectric'teki bazı testlerden geçilmesiyle ilgili cihazlara özel sorunların ortaya çıkma ihtimalinden dolayı ihtiyatlı davranıyoruz. Android bağımlılıklarını manuel olarak taklit etmek amacıyla yalnızca test içinde sistem testinin yapılmasının sağlanması çok sayıda karmaşık kod ihtiyacını ortaya çıkarıyor ve bu framework de bunu etkin bir şekilde yerine getiriyor.

iOS uygulamaları içinde şebekelendirme ve kodlama, uzun yıllardır zorlu bir olmuştur. Bu süregelen sorunu çözmek için çok sayıda kütüphane ve girişim yapıldı. Görünüşe göre, **Alamofire** JSON'u kodlamak için en güçlü ve en geliştirici dostu kütüphaneyi sunuyor. Objective-C günlerinde uzun süre kullanılan Objective-C counterpart (AFNetworking) ile aynı yaratıcı tarafından yazıldı.

AngularJS kullanarak çok sayıda başarılı proje tamamlamış olmamıza ve kurumsal ortamlarda kurulumunda hızlanma görmemize karşın, Radar'ın bu sayısında Angular'ı yeniden Değerlendir'e kaydırmaya karar verdik. Bu kaydırma işlemi bir uyarıyla birlikte geliyor: React.js ve Ember güçlü alternatifler sunuyor, Angular versiyon 1'den versiyon 2'ye geçiş belirsizlik yaratıyor ve bazı kurumların tek sayfa uygulamaların ihtiyaçlarına uygun olup olmadığını düşünmeden framework seçtiklerini görüyoruz. Bu konuda kendi içimizde hararetli tartışmalar yaşanıyor ancak kod tabanlarının iki yönlü bağlayıcı ve tutarsız durum yönetimi modellerinin birleşiminden aşırı derecede karmaşık bir hale geldiğinin de farkındayız. Katı bir çerçevenin atılma ihtiyacının doğmasından ziyade bu sorunların dikkatli tasarım ve başlangıçtan itibaren Redux ya da Flux kullanılarak çözülebileceğine inanıyoruz.

Aurelia yeni nesil JavaScript müşteri framework'ü olarak görülüyor ve JavaScript'in modern versiyonu olan ECMAScript 2016 kullanılarak yazıldı. Aurelia, Durandal'ın yaratıcısı Rob Eisenberg tarafından yaratıldı. Zamanını bu projeye adayabilmek için Angular 2.0 çekirdek ekibinden ayrılmıştı. Aurelia'nın en iyi yanı çok modüler olması, basit ve küçük kütüphaneler içermesi ve kolayca kişiselleştirilebilecek şekilde tasarlanması. Aurelia, kuralların yapılandırmadan önce gelmesi modelini takip ediyor ve bu da modüllerin üretim ve tüketimi kolaylaştırmasına karşın, kendinizi bağlamak isteyeceğiniz herhangi bir güçlü kural da bırakmıyor. Aurelia'nın geniş bir kitlesi var ve proje web sitesinde eğitim bölümlerini kullanarak hakkında daha fazla bilgi alabilirsiniz.

IoT cihazlar ile JavaScript ekosisteminin kesişim noktası, ilginç olanakları da beraberinde getiriyor. **Cylon.js** robotikler ve teknik topluluğumuzda heyecan yaratan Şeylerin İnterneti için arayüz geliştirilesini sağlayan bir JavaScript kütüphanesi. 50'den fazla platform cihazı için destek sunmasının yanı sıra cylon-gpio modülü tarafından sağlanan paylaşılan sürücü setleriyle genel amaçlara uygun girdi/çıktı desteği de sağlıyor. Bundan sonra cihazların kontrolü bir web tarayıcısı arayüzüyle yapılabiliyor.

Birçok kişide **Elixir** program dilini kullanmanın yarattığı büyük heyecanı gözlemlemeye devam ediyoruz. Erlang sanal makinenin üzerine yapılandırılan Elixir, eş zamanlı ve hata toleranslı sistemler yaratmak konusunda umut vaat ediyor. Elixir, geliştiricilerin UNIX komut satırında yapabildikleri gibi fonksiyon boru hatları geliştirmelerine olanak tanıyan Pipe operatörü gibi farklı özelliklere sahip. Paylaşılan bayt kodu, Elixir'in Erlang ile birlikte çalışmasına ve mevcut kütüphaneleri kullanırken, Mix yapı aracı, Iex interaktif kabuk ve ExUnit birim test framework'u gibi araçların desteklenmesine olanak tanıyor.

Redux framework'ünün hızlı bir şekilde benimsenmesi nedeniyle **Elm** hakkındaki düşüncelerimizi yeniden gözden geçirmek zorunda kaldık. Redux için ilk ilhamı veren Elm, Redux'ın öngörülebilir durumuyla React.js'nin görüntüleme bileşenleştirme ve reaktiflik özeliklerini, derlenmiş ve güçlü bir şekilde yazılmış fonksiyonel dille bir araya getiriyor. Elm, Haskell'de yazıldı ve Haskell benzeri sentaksa sahip ancak tarayıcı için HTML, CSS ve JavaScript derleyebiliyor. React.js ve Redux kullanmak için acele eden JavaScript programcıları, bazı uygulamalar için tür güvenliği olan bir alternatif olarak Elm'e de bakabilirler.

Ortalıkta kullanılan REST uygulamalarına baktığımız zaman sık sık REST'in sadece müşteri ile sunucu arasındaki konuşkan (chatty) etkileşimler üzerinden obje grafiklerinin alınması için hatalı kullanıldığını görüyoruz. Facebook'un kullandığı **GraphQL** bu genel kullanım durumu için REST'e göre daha iyi bir yaklaşım alternatifi sunuyor. Obje grafiklerinin uzaktan çekilmesi için bir protokol olan GraphQL, son dönemde dikkatleri üzerine çekti. GraphQL'i en ilginç özelliklerinden birisi müşteri odaklı yapısı: Yanıtın yapısı sunucu değil, tamamen müşteri tarafından yönetiliyor. Bu müşterinin bağlantısının kopmasına ve sunucunun Postel'in yasasına uymasına neden oluyor. Müşteri uygulamaları şu an birçok programlama dilinde mevcut olsa da React.js'nin durum bulundurmayan bileşen modelini desteklemek için tasarlanan bir JavaScript framework'ü olan Facebook'un Relay'ine büyük bir ilgi olduğunu görüyoruz.

Değişmezlik, fonksiyonel program mantığında sıklıkla vurgulanıyor ve birçok dilde bir kere yaratıldıktan sonra değiştirilemeyen değişmez objeler yaratılması özelliği bulunuyor. **Immutable.js** modern JavaScript sanal makineler üzerinde çok verimli olan kalıcı olarak değiştirilemez veri yapıları sağlayan JavaScript için geliştirilmiş bir kütüphane. Ancak, Immutable.js objeleri normal objeler değil. Dolayısıyla değiştirilemez objelerden JavaScript objelerine referanslar yapılmamalı. Ekiplerimiz mutasyonun takibi ve durumun korunması için bu kütüphanenin kullanılmasına önem veriyor ve geliştiricilere özellikle Facebook yığınının geri kalanıyla birleştirildiğinde incelenmesi gereken bir kütüphane olduğunu düşünüyoruz.

Recharts'ın D3 çizimleri React.js'ye anlaşılır ve bildirimsel bir şekilde eklemesini kullanmaktan büyük keyif alıyoruz.

Birçok iOS geliştiricisi, uygulamalarına dinamik bir şekilde yama yapmak için **JSPatch** kullanıyor. JSPatch'in etkinleştirildiği bir uygulama çalışırken, çok sayıda JavaScript yüklüyor (muhtemelen güvenli olmayan bir HTTP bağlantısı üzerinden) ve daha sonra da davranış değiştirmek, hataları düzeltmek ve diğer başka işlemler için ana Objective-C uygulama koduna bağlantı kuruyor. Her ne kadar kolay olsa da canlı uygulamalara monkey-patching yapılmasının kötü bir fikir olduğunu ve bundan uzak durulması gerektiğini düşünüyoruz. Herhangi bir miktarda artımsal yamalama yapılırken, işlevselliği düzgün bir şekilde doğrulamak için test sürecinizin son kullanıcının deneyimine uygun olması büyük önem taşıyor. Alternatif yaklaşım olarak uygulama için React Native ve küçük güncelleme ile yeni özellikler göndermek için AppHub ve CodePush kullanılabilir.
