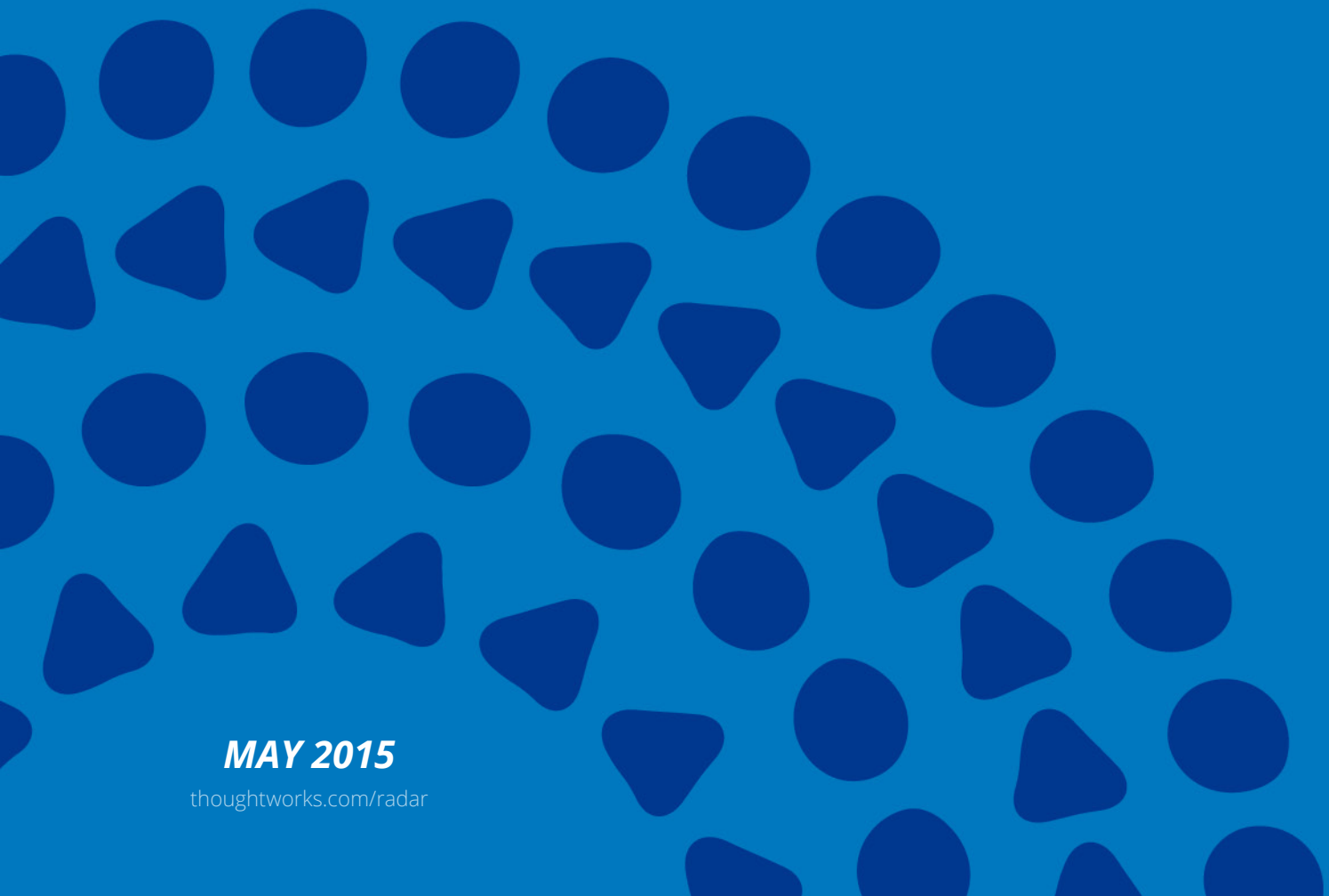
ThoughtWorks®

TECHNOLOGY RADAR

Nuestros pensamientos acerca de la tecnología y las tendencias que están dando forma al futuro



MAY 2015

thoughtworks.com/radar

NOVEDADES

Estas son las tendencias sobresalientes en esta edición

INNOVACIÓN EN ARQUITECTURA

Las organizaciones han aceptado que la nube es la plataforma de-facto del futuro, y los beneficios y la flexibilidad que trae han causado un renacimiento en la arquitectura de software. La infraestructura desechable de la nube, ha habilitado la primera arquitectura nativa a la nube: los microservicios. Entrega continua es una técnica que cambia radicalmente la evolución de negocios tecnológicos y amplía el impacto de la nube como una arquitectura. Esperamos que continúe la innovación arquitectónica con tendencias como la contenedorización y redes definidas por software que proveen aún más opciones técnicas y capacidad.

NUEVA OLA DE APERTURA EN MICROSOFT

Mientras Microsoft se ha aventurado en open-source antes - p.ej. su plataforma de hosting CodePlex - los recursos centrales de la compañía han seguido como secretos privados y cuidadosamente guardados. Ahora, Microsoft parece abrazar una nueva estrategia de apertura, con el lanzamiento de grandes partes de la plataforma .NET y runtime como proyectos open-source en GitHub. Esperamos que eso pueda abrir el camino para que sea Linux una plataforma de hosting para .NET, lo que permitirá la competición del lenguaje C# junto a la bandada actual de lenguajes basados en JVM.

LUCHAS CONTINUAS DE SEGURIDAD EN LAS EMPRESAS

A pesar del incremento en la atención a la seguridad y la privacidad, la industria no ha progresado mucho desde el último Radar y continuamos señalando el problema. Los desarrolladores nos responden con un aumento en infraestructura y automatización de la seguridad, construyendo herramientas como ZAP en canales de despliegue. Éstas forman sólo una parte de una estrategia holística sobre la seguridad y creemos que todas las organizaciones necesiten mejorar en este area.

QUIENES CONTRIBUYEN

El Radar de Tecnología es preparado por el Comité Consultor de Tecnología de ThoughtWorks, compuesto por:

Rebecca Parsons (CTO)

Martin Fowler(Chief Scientist)

Anne J Simmons

Badri Janakiraman

Brain Leke

Claudia Melo

Dave Elliman

Erik Doernenburg

Evan Bottcher

Hao Xu

Ian Cartwright

James Lewis

Jeff Norris

Jonny LeRoy

Mike Mason

Neal Ford

Rachel Laycock

Sam Newman

Scott Shaw

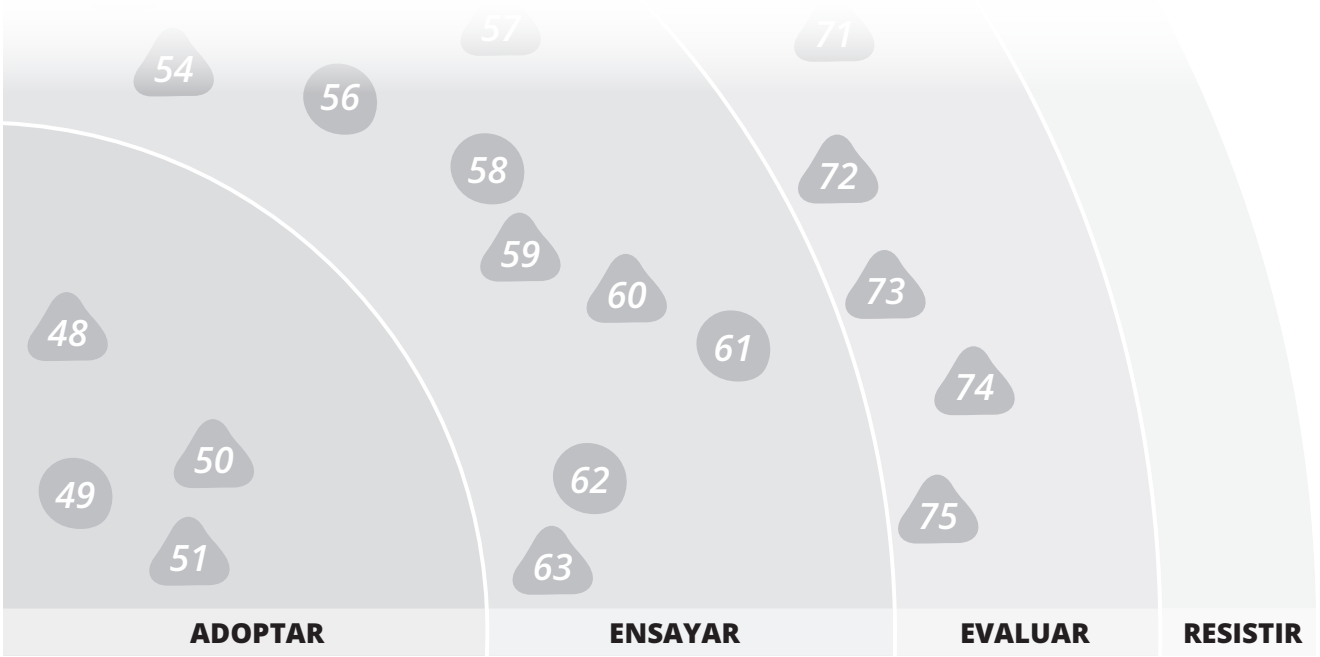
Srihari Srinivasan

Thiyagu Palanisamy

ACERCA DEL RADAR DE TECNOLOGÍA

Los ThoughtWorkers son apasionados por la tecnología. La construimos, la investigamos, la probamos, la publicamos en código libre, escribimos acerca de ella, y constantemente logramos mejorarla para todos. Nuestra misión es liderar la excelencia en el software y revolucionar la industria de la TI. Nosotros creamos y compartimos el Radar de Tecnología de ThoughtWorks como parte de esta misión. El Comité Consultor de Tecnología, es un grupo de experimentados líderes en tecnología de ThoughtWorks que crea el radar. Ellos se reunen regularmente para discutir la estrategia global de tecnología en ThoughtWorks y las tendencias en la tecnología con un impacto significativo en nuestra industria.

El radar captura el resultado de las discusiones del Comité Consultor de Tecnología en un formato que provee valor a un amplio rango de interesados, desde CIOs hasta desarrolladores. El contenido está destinado a ser un resumen conciso. Nosotros alentamos a que explores estas tecnologías en mayour detalle. El radar es gráfico por naturaleza, agrupando a los ítems en técnicas, plataformas, lenguajes y frameworks. Cuando los ítems del radar pueden aparecer en varios cuadrantes, escogemos el que nos parece más apropiado. Además agrupamos estos items también en cuatro anillos para reflejar nuestra posición actual respecto a ellos. Los anillos son:



Nosotros estamos convencidos de que la industria debería adoptar estos ítems. Nosotros los utilizamos cuando es apropiadoen nuestros proyectos.

Valen la pena probar. Es importante entender como construir esta capacidad. Las empresas deberían probar esta tecnología en un proyecto en el que se puede manejar el riesgo.

Vale la pena explorar con la comprensión de el cómo va a afectar su empresa.

Proceder con precaución.

Los ítems que son nuevos o que han tenidos cambios significativos desde el último radar son representados por triángulos, mientras que los ítems que no se han movido son representados como círculos. Son de nuestro interés muchas más ítems de los que pueden caber en un documento de este tamaño, así que removemos gradualmente ítems del último radar para crear espacio para los nuevos ítems. Remover un ítem no significa que ya no sea de nuestro interés.

Para más información acerca del radar, visita thoughtworks.com/radar/faq

EL RADAR

TÉCNICAS

ADOPTAR
1. Consumer-driven contract testing new
2. Focus on mean time to recovery
3. Generated infrastructure diagrams new
4. Structured logging

ENSAYAR
5. Canary builds
6. Datensparsamkeit
7. Local storage sync
8. NoPSD
9. Offline first web applications new
10. Products over projects new
11. Threat Modelling new

EVALUAR
12. Append-only data store
13. Blockchain beyond bitcoin
14. Enterprise Data Lake
15. Flux new
16. Git based CMS/Git for non-code new
17. Phoenix Environments new
18. Reactive Architectures new

RESISTIR
19. Long lived branches with Gitflow
20. Microservice envy
21. Programming in your CI/CD tool
22. SAFe™
23. Security sandwich
24. Separate DevOps team

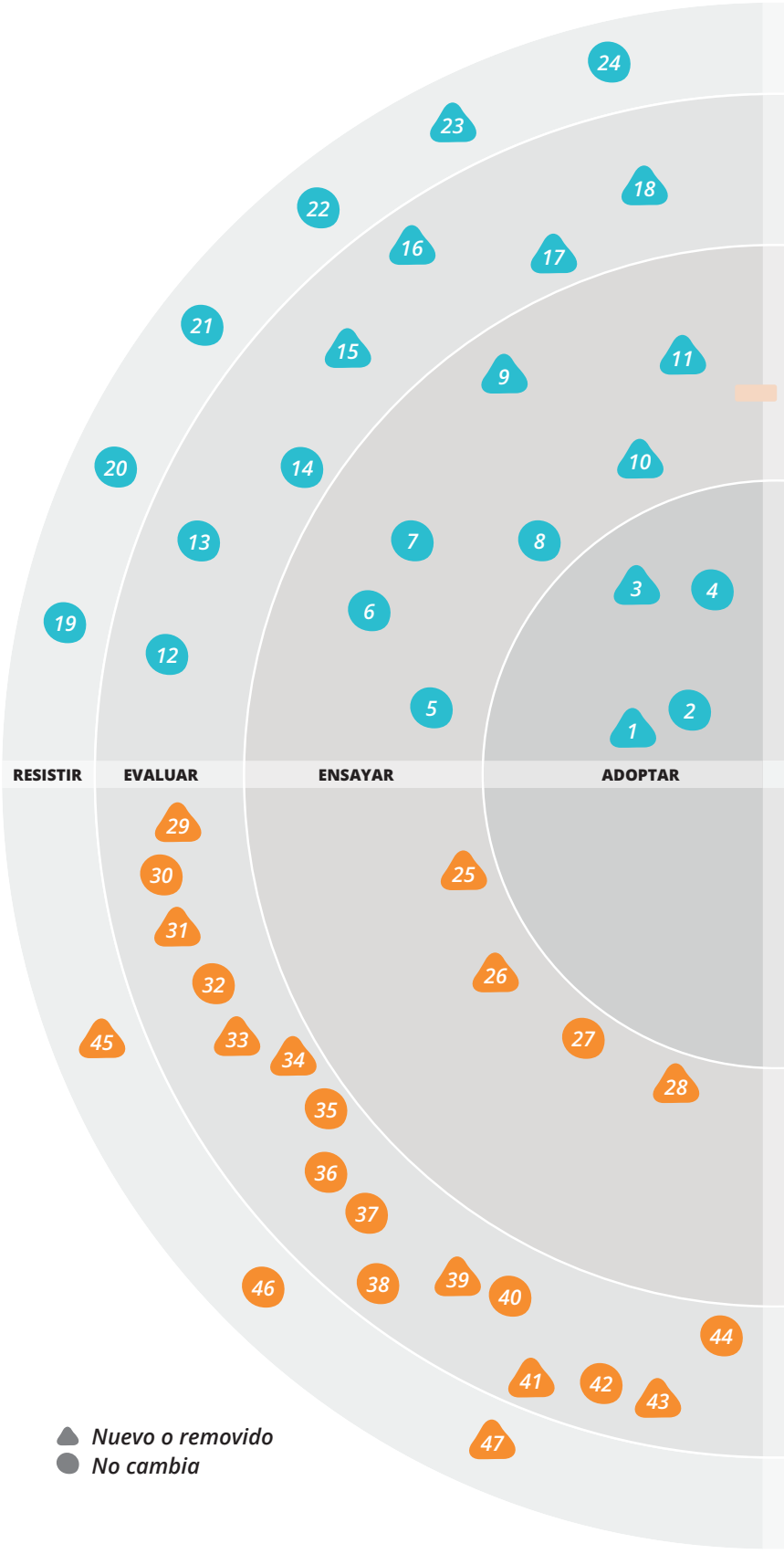
PLATAFORMAS

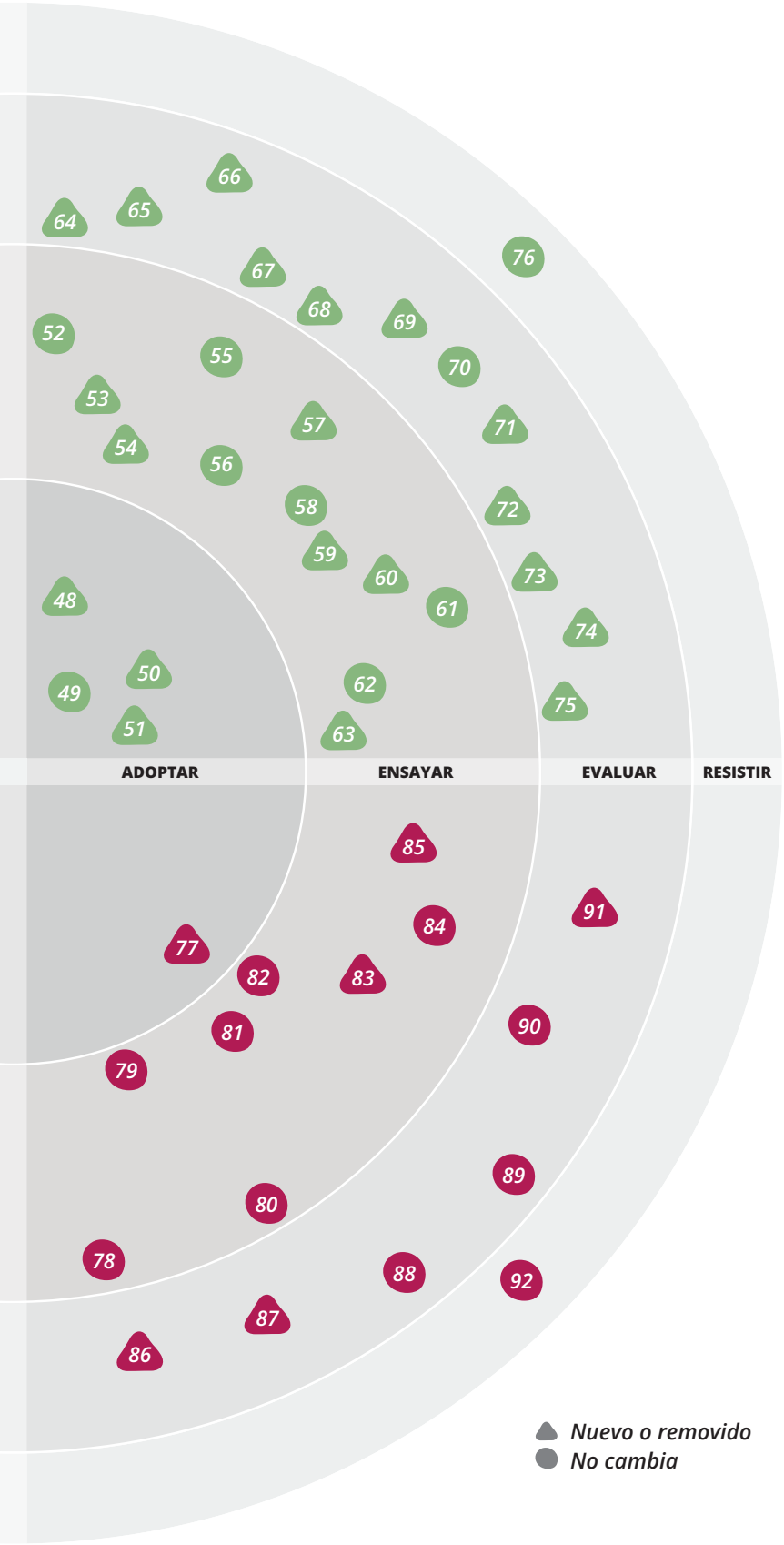
ADOPTAR

ENSAYAR
25. Apache Spark
26. Cloudera Impala new
27. DigitalOcean
28. TOTP Two-Factor Authentication

EVALUAR
29. Apache Kylin new
30. Apache Mesos
31. CoreCLR and CoreFX new
32. CoreOS
33. Deis new
34. H2O new
35. Jackrabbit Oak
36. Linux security modules
37. MariaDB
38. Netflix OSS Full stack
39. OpenAM
40. SDN
41. Spark Photon/Spark Electron new
42. Text it as a service / Rapidpro.io
43. Time series databases new
44. U2F

RESISTIR
45. Application Servers new
46. OSGi
47. SPDY new





El RADAR

HERRAMIENTAS

ADOPTAR
48. Composer
49. Go CD
50. Mountebank
51. Postman

ENSAYAR
52. Boot2docker
53. Brighter new
54. Consul
55. Cursive
56. GitLab
57. Hamms new
58. IndexedDB
59. Polly new
60. REST-assured new
61. Swagger
62. Xamarin
63. ZAP new

EVALUAR
64. Apache Kafka new
65. Blackbox
66. Bokeh/Vega new
67. Gor new
68. NaCl new
69. Origami new
70. Packetbeat
71. pdfmake new
72. PlantUML new
73. Prometheus new
74. Quick new
75. Security Monkey new

RESISTIR
76. Citrix for development

LENGUAJE & FRAMEWORKS

ADOPTAR
77. Nancy

ENSAYAR
78. Dashing
79. Django REST
80. Ionic Framework
81. Nashorn
82. Om
83. React.js
84. Retrofit
85. Spring Boot

EVALUAR
86. Ember.js new
87. Flight.js
88. Haskell Hadoop library
89. Lotus
90. Reagent
91. Swift

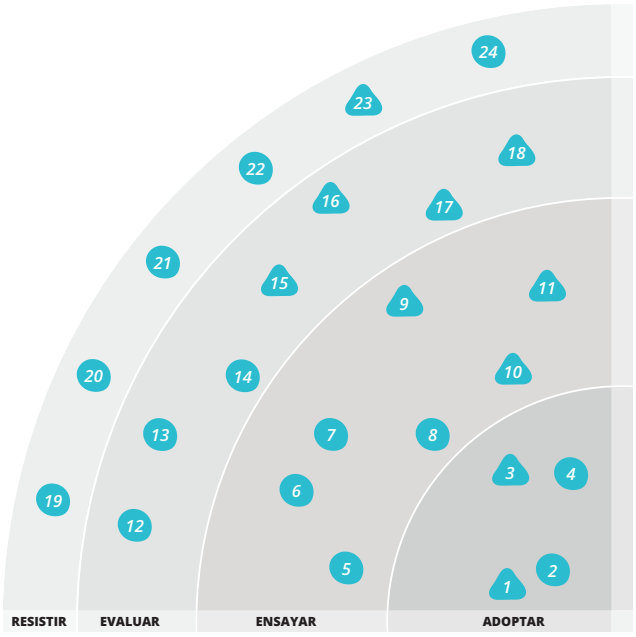
RESISTIR
92. JSF

TÉCNICAS

Cuando dos servicios desarrollados independientemente colaboran, los cambios en el API del proveedor pueden causar errores en todos sus consumidores. Los servicios del lado del consumidor normalmente no pueden probar proveedores en producción debido a la lentitud y la fragilidad de esas pruebas, por lo que se recomienda usar Test Doubles, con lo cual, en cambio, se corre el riesgo de que dichos Test Doubles se desincronicen del servicio proveedor real.

Los equipos consumidores pueden protegerse de estos fallos usando pruebas de integración de contrato, que comparen respuestas reales del servicio con valores de prueba.

Aunque esas pruebas de contrato son de valor, su importancia aumenta cuando los servicios del lado del consumidor proporcionan pruebas al proveedor, quien puede ejecutar pruebas de contrato para determinar si los cambios de los consumidores van a causar problemas. Adoptar pruebas de contrato guiadas por el consumidor es una parte esencial de las pruebas de microservicios.



Tradicionalmente, grupos de operaciones intentan mejorar el tiempo medio entre fallos. Aunque evitar fallas sigue siendo muy importante, lecciones de cloud computing nos han enseñado a esperar las fallas y enfocarnos en el tiempo medio de recuperación. La automatización de la Entrega Continua facilita la implementación rápida de correcciones y también hemos observado un aumento en las técnicas de monitoreo para localizar fallos rápidamente por medio de un "sistema en producción inmune". Los equipos han tenido éxito con el monitoreo semántico y las transacciónes sintéticas para el ejercicio de los sistemas de producción en una manera no destructiva. Ese enfoque combinado permite que los equipos hagan todo más rápidamente y con más confianza en sí mismos, reduce el énfasis en ejecución de pruebas costosas en el ambiente de pre-producción y es especialmente importante en cuanto al manejo de la lista creciente de vulnerabilidades de seguridad que se sigan descubriendo.

Cuando se necesita un diagrama que describe la infraestructura actual o arquitectura física, usamos nuestra herramienta de diagramas técnicos preferida. Si usas la nube o tecnologías virtualizadas, eso no tiene sentido, podemos usar los APIs provistos para investigar la infraestructura y generar un **diagrama de infraestructura real automatizada** usando herramientas simples como GraphViz o generar SVG.

Tratar a los logs como datos nos da un conocimiento más profundo de la actividad operacional de los sistemas que construímos. Los logs estructurados (que significa usar un formato de mensajes predeterminado y consistente que contenga información semántica), contribuyen a esta técnica y permiten que herramientas como Graylog2 y Splunk produzcan conocimientos más profundos. Recomendamos adoptar la práctica de estructurar logs, pues los beneficios valen el esfuerzo mínimo necesario y la práctica se convierte en el estándar por defecto.

Muchos proyectos dependen de código externo, la mayor parte del cual proviene de proyectos de open-source. Para asegurar que nuestras construcciones se pueden

ADOPTAR
1. Consumer-driven contract testing
2. Focus on mean time to recovery
3. Generated infrastructure diagrams
4. Structured logging

ENSAYAR
5. Canary builds
6. Datensparsamkeit
7. Local storage sync
8. NoPSD
9. Offline first web applications
10. Products over projects
11. Threat Modelling

EVALUAR
12. Append-only data store
13. Blockchain beyond bitcoin
14. Enterprise Data Lake
15. Flux
16. Git based CMS/Git for non-code
17. Phoenix Environments
18. Reactive Architectures

RESISTIR
19. Long lived branches with Gitflow
20. Microservice envy
21. Programming in your CI/CD tool
22. SAFe™
23. Security sandwich
24. Separate DevOps team

TÉCNICAS *continua*

reproducir, las integramos con versiones previas, pero eso puede significar que toma más tiempo integrar con versiones nuevas de esas librerias, lo que requiere más esfuerzo de integración en el futuro.

Una estrategia para evitar esto, es tener **Canary Builds** todas las noches, los cuales tratan de tomar las últimas versiones de todas las dependencias. Si la construcción está en verde, sabemos que podemos cambiar las versiones de las que dependemos. El término proviene de legislacíon de privacidad alemana y describe la idea de guardar sólo la cantidad minima de información personal requerida para el negocio o las leyes que aplican. La privacidad de los clientes sigue siendo un tema candente. Parece que companías como Uber acumulan datos altamente personales al mismo tiempo que estan relajados en cuanto a la seguridad. Eso es un desastre inminente. Seguir **datensparsamkeit** o usar técnicas de identificación incluso en jurisdicciónes en las que no se requiere, puede permitir la reducción de la información almacenada. Si no guardas información, no necesitas preocuparte por el robo de la misma.

En la implementación de aplicaciones de una sola página, tarde o temprano asoma la cuestión del uso fuera de línea. Dada la dificultad de acertar en la actualización de un modo fuera de línea en una aplicación que ya existe, hay una tendencia a implementar en este tipo de aplicaciones una configuración de arranque fuera de línea. Una técnica de integración importante con la que hemos tenido éxito, es sincronizar el almacenamiento local. En este caso, el código de cara al usuario nunca hace solicitudes al backend -recupera los datos sólo del almacenamiento local. Un trabajador en segundo plano sincroniza los datos en el almacenaiemnto local con los sistemas backend, normalmente con la auyda de alguna forma de API REST.

NoPSD es un movimiento para la integración de actividades de diseño en los ciclos de feedback iterativos requeridos para construir software maravilloso. El nombre tiene como objetivo desplazar el **PSD** como artefacto de diseño final canónico en lugar de burlarse del software de Adobe. En lugar de empezar con una especificación de diseño supuestamente perfecta, el equipo es instado a adoptar el Diseño contínuo: integrar diseñadores en los equipos de entrega, usar técnicas no tan tecnologicas para el prototipado y colaborar para perfeccionar el diseño en la tecnología de UI adecuada (normalmente HTML y CSS). Esta estrategia acelera la respuesta del feedback real del usuario, permite pruebas de diseños en diferentes plataformas y factores de forma; y abraza la naturaleza dinámica de los productos digitales y su proceso de creación.

Las aplicaciones web de arranque fuera de línea proveen la habilidad de diseñar aplicaciones web para uso fuera de línea utilizando mecanismos de retención y actualización. La implementación requiere un indicador en el DOM para comprobar si el aparato está fuera de línea (en este caso accedería al almacenamiento local) o no (en este caso sincronizaría los datos). Todos los navegadores permiten funciones fuera de línea, con la información local acesible especificando un atributo de manifesto en el html, el cual arranca el proceso de descarga y retención de los recursos (como HTML, CSS, Javascript imágenes, entre otros). Existen herramientas que ayudan a simplificar la configuración de arranque fuera de línea, como **Hoodie** y **CouchBD**, que también ofrecen la posibilidad de trabajar con una aplicación desplegada localmente sobre un almacenamiento local de datos.

En este punto, la mayor parte de los equipos de desarrollo reconocen la importancia de escribir software seguro y manejar los datos de sus usuarios de una manera responsable. Se enfrentan a una curva de aprendizaje escarpada y a una gran cantidad de amenazas potenciales que se extienden desde crimen organizado y espionaje gubernamental hasta adolescentes que atacan sistemas porque sí. El modelado de amenazas es un juego de técnicas, la mayoría de ellas defensivas, que ayudan a entender y clasificar amenazas potenciales. Cuando se convierten en 'historias de usuarios malvados' pueden servir como una estrategia efectiva para hacer un sistema más seguro.

Las estructuras de datos inmutables se vuelven cada vez más populares con lenguajes funcionales como **Clojure** que provee inmutabilidad por defecto. La inmutabilidad permite que el código sea escrito, leido y razonado más facilmente. El uso de almacenes que solo permiten agregar datos, puede otorgar algunos de esos beneficios a la capa de base de datos, así como simplificar la auditoría y las consultas históricas. Hay varias opciones para la implementación, desde almacenes específicos que solo permiten agregar, como Datonic, hasta usar la estrategia 'agregar-no-actualizar' con una base de datos estándar. Mientras el aspecto monetario de Bitcoin y otras cripto-monedas es el más famoso, nosotros estamos igualmente entusiasmados con las posibilidades de usar Blockchain fuera de Bitcoin y transacciones financieras.

TÉCNICAS *continua*

Blockchain es un mécanismo para verificar el contenido de un libro de contabilidad compartido sin depender de un servicio centralizado.

Ya estamos viendo como se usa el Blockchain (la tecnología subyacente igual que el Bitcoin Blockchain público) en el centro de sistemas de gran variedad, como identidad, propiedad, contabilidad, votación, almacenamiento en la nube, e incluso manejo de redes de dispositivos inteligentes. Si estas construyendo sistemas que requieren confianza en lugar de redes decentralizadas, vale la pena evaluar Blockchain.

Una laguna de datos empresariales es un almacén inmutable de una gran cantidad de datos crudos que actúa como un recurso para otros streams de procesos y está directamente disponible para un número significativo de consumidores internos, técnicos a través de un motor eficiente de procesamiento. Como ejemplos se tiene **HDFS** o **HBase** en un framework de procesamiento Hadoop, Spark o Storm. Éste se puede contrastar con un sistema típico que acumula los datos crudos en un espacio altamente restringido, el cual está solo disponible para estos consumidores como el resultado final de un proceso ETL altamente controlado. Abrazar el concepto de la laguna de datos trata de eliminar los cuellos de botella debido a la falta de desarrolladores ETL o diseño excesivo de modelos de datos up-front y fortalecer a los desarrolladores en la creación de sus propios canales de procesamiento de datos de una manera ágil según sus necesidades hasta cierto punto, lo que nos recuerda otro modelo que valoramos mucho, **DevOps**.

Hoy en día, la mayoría de los desarrolladores están acostumbrados a trabajar con **Git** con el fin de controlar el código fuente y colaborar. Hay otras circunstancias en las que se usa como mecanismo fundamental: cuando un grupo necesita colaborar sobre documentos textuales (que se pueden mezclar fácilmente). Cada vez más proyectos usan **Git** como base para un CMS ligero con formatos de edición basados en texto. **Git** tiene características poderosas en cuanto al seguimiento de cambios o la exploración de alternativas, con un modelo de almacenamiento distribuido que es rápido de usar y tolerante a problemas de conectividad. El mayor problema es que no es fácil de aprender para quienes no son programadores, pero esperamos ver más herramientas que construyen sobre la estructura principal de Git.

Estas simplificarían el proceso de trabajo para una audiencia específica, como autores de contenido.

Bienvenidas serían también herramientas que soporten hallar diferencias y mezclar documentos no-textuales. La idea de servidores **Phoenix** está bien establecida y, cuando se aplica a los tipos de problemas adecuados, trae muchos beneficios pero tenemos que tomar en cuenta el ambiente en el que desplegamos los servidores. El concepto de ambientes **Phoenix** nos puede ser útil: podemos usar automatización para crear ambientes completos incluso con configuración de redes, balanceo de carga y puertos de firewall, a través de **CloudFormation** en **AWS**. Podemos probar que el proceso funciona destruyendo completamente los ambientes y reconstruyéndolos regularmente. Los ambientes **Phoenix** soportan el aprovisionamiento de nuevos ambientes para pruebas, desarrollo, UAT, etc. También pueden simplificar el aprovisionamiento de un ambiente de recuperación de desastres. Tal como en el caso de los servidores Phoenix, este patrón tampoco se puede aplicar en todos los casos y necesitamos pensar con mucho cuidado sobre el estado y las dependencias. Una estrategia para reconfigurar el ambiente puede ser tratarlo como un despliegue verde/azul.

Las técnicas de programación funcional reactiva se han vuelto cada vez más populares en los últimos años, y hemos visto un interés creciente en ampliar la aplicación de ese concepto sobre arquitecturas de sistemas distribuidos. Inspiradas en parte de "The Reactive manifesto" esas arquitecturas reactivas están basadas en un flujo asíncrono de eventos inmutables de un solo sentido en una red de procesos independientes (talvez implementados como microservicios). En la situación adecuada, estos sistemas son expandibles y fuertes, y reducen el acoplamiento entre unidades de procesamiento individuales.

A pesar de eso, arquitecturas basadas en el paso de mensajes asíncronos, introducen complejidad y dependen de frameworks privados. Recomendamos la evaluación de las necesidades de desempeño y expansión de su sistema antes de comprometerse a este estilo de arquitectura por defecto.

Gitflow es un patrón estricto de ramificación para liberación de versiones que usan Git. Aunque no es un patrón intrínsecamente malo, está frecuentemente usado mal. Si las características y ramas de desarrollo tienen una vida corta y se fusionan frecuentemente, en realidad se usa el poder de git, lo que facilita estas actividades.

Aún así, el problema que vemos a menudo es que esas ramas se convierten en ramas de larga vida, lo que resulta

TÉCNICAS *continua*

en conflictos de fusión que mucha gente quería evitar con el uso de Git. Una fusión es una fusión, sin importar la herramienta de control de código fuente o el patrón que se use.

Si se toma más tiempo para fusionar, esto puede resultar en un gran conflicto de fusión, lo que puede volverse problemático si tienes un equipo más grande. Si tienes mucha gente esperando una fusión, eso puede causar un cuello de botella. La introducción de patrones como **Gitflow**, requiere la disciplina para que la fusión tenga éxito. De todos modos, usa el patrón, pero sólo si tienes la disciplina de prevenir ramas de larga vida.

Aún estamos convencidos de que los microservicios pueden ofrecer ventajas significativas a las organizaciones en términos de mejorar la autonomía de un equipo y acelerar la frecuencia de cambios. La complejidad adicional que viene de sistemas distribuidos requiere un nivel adicional de madurez e inversión. Nos preocupa de que algunos equipos adoptan rápidamente los microservicios sin entender los cambios de desarrollo, pruebas y operaciones necesarios para hacerlos bien. Nuestro consejo es simple: evita la envidia entre microservicios y empieza con uno o dos antes de apurarte en el desarrollo de más microservicios para permitir que sus equipos se adapten y entiendan el nivel adecuado de granularidad.

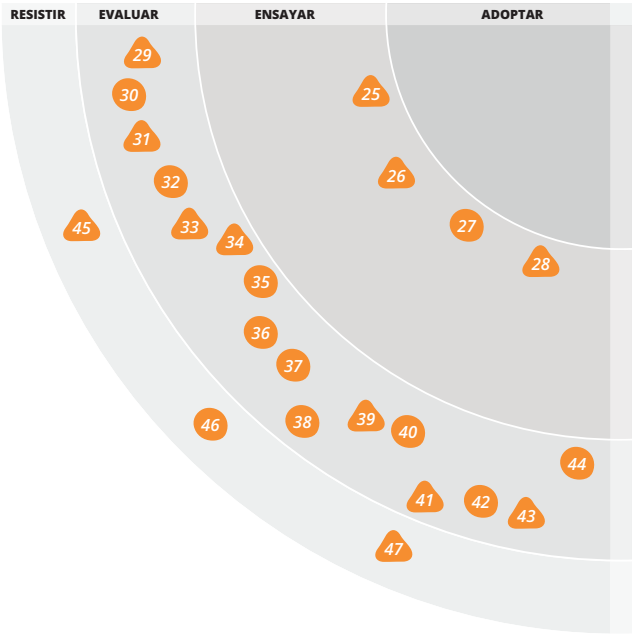
Seguimos viendo equipos que configuran sus herramientas de **CI** y **CD** por medio de la incorporación directa de comandos complejos de múltiples líneas en la configuración de la herrramienta. A menudo esos comandos contienen pasos que serían ejecutados únicamente en el ambiente de construcción, como variables de ambiente específicas para CI, dichos pasos sólo editarían/modificarían los archivos y plantillas en este ambiente, etc. Eso hace el ambiente de construcción muy particular, porque los resultados no se pueden duplicar localmente en las máquinas de los desarrolladores. Esto es extremadamente problemático, porque la herramienta de CI/CD, que supuestamente tiene que exhibir los problemas en el código, se convierte en algo cuyo comportamiento y resultados son difíciles de predecir. La manera de evitar programar en su herramienta de CI/CD es sacar las complejidades del proceso de construcción de la herramienta y ponerles en un script simple que se ejecute por medio de un solo comando. Ese script se puede ejecutar en cualquier máquina de desarrolladores, lo que elimina el estado privilegiado/singular del ambiente de construcción. Escalar "agile" en la empresa es constantemente un reto. Se han propuesto muchas estrategias, con **SAFe** ganando un reconocimiento significativo. Mientras **SAFe** provee una lista útil de areas de interés, esas son fácilmente mal utilizadas al introducir el mismo tipo de tendencias de grandes lanzamientos (como el tren de lanzamientos o procesos cerrados de control) que "agile" elimina. Las empresas en particular buscan un grado de homogeneidad en sus asuntos que SAFe provee popularizando la estandarización agresiva en casos en los que cierto grado de personalización tiene valor significativo. Otras estrategias "lean" (incluso la experiemntación e incorporación de prácticas de mejoramiento continuo como las Improvement Katas) ofrecen un mejor modelo para escalar "agile." Scaled Agile Framework y SAFe son marcas registradas de Scaled Agile, Inc.

Las estrategias tradicionales de seguridad dependen de la especificación inicial y validación final. Ese Security Sandwich es difícil de integrar en equipos Ágiles, porque la gran parte del diseño es parte del proceso y estas estrategias no hacen uso de las oportunidades de automatización que provee la entrega continua. Las organizaciones deben buscar cómo integrar practicas de seguridad en el ciclo de desarrollo ágil. Esto incluye: evaluación del nivel adecuado de Modelamiento de amenazas por hacer inicialmente, cuándo clasificar preocupaciones de seguridad como historias, criterios de aceptación o requerimentos transversales no funcionales, inclusión de pruebas de seguridad automáticas (estáticas y dinámicas) en el canal de construcción, e incluir pruebas más profundas (como pruebas de penetración) en iteraciones de un modelo de entrega contínua. Tal como DevOps ha revolucionado el trabajo conjunto de equipos adversarios, lo mismo está pasando con los profesionales de seguridad y desarrollo. (Pero, a pesar de nuestro fastidio del modelo Security Sandwich, éste es mucho mejor que la falta de consideración de seguridad, lo que tristemente, sigue siendo una circunstancia normal.

En la última edición aconsejamos que no cree un equipo DevOps separado, porque DevOps trata de crear una cultura de responsabilidad compartida en los equipos de entrega. Recomendamos la incorporación de habilidades operacionales en los equipos para reducir la fricción y obtener mejores resultados. Dicho esto, donde hay necesidad de gran inversión en mecanización y automatización, también hay un papel para un equipo separado. En lugar de dar soporte técnico, esos equipos construyen mecanizaciones y permiten que el resto de los equipos desplieguen, monitoreen y mantengan sus propios entornos de producción.

PLATAFORMAS

Apache Spark se ha vuelto cada vez más popular como motor de procesamiento de cargas masivas. Está escrito en Scala y es muy adecuado para applicaciones que reusan el mismo juego de datos para operaciones simultáneas. Diseñado para funcionar como clúster indepeniente o como parte del clúster Hadoop YARN. Puede acceder a fuentes de datos como HDFS, Cassandra, S3 etc. También ofrece muchos operadores de alto nivel con el fin de facilitar el desarrollo de applicaciones simultáneas de datos. Como plataforma genérica de procesamiento de datos, ha habilitado el desarrollo de muchas herramientas de alto nivel, como SQL interactivo (Spark SQL), streaming en tiempo real (Spark Streaming), librería de aprendizaje de máquinas (Mlib) R-on-Spark, etc.



La communidad Hadoop ha estado intentando dar capacidades interactivas SQL de baja latencia a la plataforma Hadoop. Esto ha resultado en un par de sistemas open-source desarrolados activamente en 2014 (Cloudera Impala, Apache Drill, Facebook Presto etc) Creemos que esta tendencia de SQL-en-Hadoop es una señal de cambio importante, pues cambia la proposición de Hadoop de una tecnologia adicional a los bases de datos y orientada a los lotes, a algo que puede ser su competición. **Cloudera Impala** fue una de las primeras plataformas SQL-on-Hadoop. Es un motor de consultas basado C++ distribuido y masivamente simultaneo. El componente clave de esta plataforma es el demonio de Impala que coordina la ejecución de la consulta SQL en uno o más nodos del cluster Impala. Impala es diseñado para leer datos de archivos almacenados en HDFS en todos los formatos populares. Hace uso del catálogo de metadatos de Hive para compartir bases de datos y tablas entre las dos plataformas de bases de datos. Impala incluye una shell y drivers JDBC y ODBC para ser usados por las aplicaciones.

Seguimos usando DigitalOcean para la infraestrucutura computacional básica y el servicio nos sigue impresionando. Si se necesita infraestrucutra de la nube que es fácil de usar, vale la pena investigar ésta.

Las contraseñas siguen siendo un mecanismo pobre para autenticar usuarios y últimamente hemos visto compañías como Yahoo! elegir una solución “sin contraseñas” - se manda un texto con código de un solo uso cada vez que necesitas iniciar una sesión desde otro navegador. En caso de que se sigan usando contraseñas, recomendamos el uso de autentificación de dos pasos, que puede mejorar significativamente la seguridad. **TOTP** es el algoritmo estandar en este caso, con aplicaciones de autenticación gratis de Google y Microsoft.

ADOPTAR

ENSAYAR
25. Apache Spark
26. Cloudera Impala
27. DigitalOcean
28. TOTP Two-Factor Authentication

EVALUAR
29. Apache Kylin
30. Apache Mesos
31. CoreCLR and CoreFX
32. CoreOS
33. Deis
34. H2O
35. Jackrabbit Oak
36. Linux security modules
37. MariaDB
38. Netflix OSS Full stack
39. OpenAM
40. SDN
41. Spark Photon/Spark Electron
42. Text it as a service / Rapidpro.io
43. Time series databases
44. U2F

RESISTIR
45. Application Servers
46. OSGi
47. SPDY

PLATAFORMAS *continua*

Apache Kylin es una solución analítica open-source de eBay Inc. que permite un análisis multidimensional basado en SQL (OLAP) de grandes conjuntos de datos. Kylin es planeada como solución OLAP híbrida (HOLAP) basada en Hadoop que eventualmente soportará análisis multidimensional en el estilo de MOLAP y ROLAP. Con Kylin se puede definr cubos por medio de Cube Designer e iniciar un proceso fuera de línea para la construcción de dichos cubos. Este proceso desempeña un paso antes del join para juntar los hechos y las tablas de dimensión en una estructura plana. Después viene una fase pre-agregación donde se construyen cuboides individuales usando Map Reduce jobs. Los resultados se guardan en archivos de secuancia HDFS y subidos posteriormente a HBase. Las solicitues de datos pueden venir de SQL por medio de una herramienta basada en SQL. El motor de consultas (basado en Apache Calcite) determina si existe el conjunto de datos requerido en HBase. En caso de que sí exista, el motor accede directamente a los datos necesarios y devuelve el resultado con una latencia mínima. Si no, el motor enruta las consultas a Hive, o cualquier otra solución SQL-on-Hadoop activada sobre el cluster.

Mesos es una plataforma que resume los recursos de computación subyacentes para facilitar la construcción de sistemas distribuidos masivamente expandibles. Se puede usar para proveer un nivel de planificación para Docker o actuar como nivel de abstracción para AWS y cosas similares. Twitter la ha usado con mucho éxito para ayudarlos a escalar su infraestructura. Ya hay herramientas construídas sobre Mesos, como Chronos que es un remplazo a cron distribuido y tolerante a fallas.

CoreCLR y CoreFX son la plataforma y framework central para .NET. Aunque no son nuevos, han sido convertidos recientemente en open-source por parte de Microsoft. Un cambio clave es que esas dependencias son “bin-deployable.” No necesitan instalación adelantada. Eso facilita despliegues simultáneos y permite que las aplicaciones usen diferentes versiones del framework sin conflictos. Algo escrito en .NET es entonces un detalle de implementación; se puede instalar una dependencia .NET en cualquier ambiente. Desde una perspectiva de dependencia externa, una herramienta .NET no es diferente de algo escrito en C, lo que hace de la herramienta una opción mucho más atractiva para aplicaciones y utilidades de propósito general. CoreFX está siendo factorado en dependencias individuales NuGet para que las aplicaciones puedan descargar sólo lo que necesitan, mantener el impacto mínimo de las aplicaciones y librerias .NET y facilitar el reemplazo de partes del framework.

CoreOS es una distribución Linux diseñada para ejecutar grandes sistemas escalables. Todas las aplicaciones desplegadas en una instancia de CoreOS son ejecutadas en contenedores independientes de Docker y CoreOS provee un juego de herramientas para su manejo, incluso su propio almacén de configuración distribuida, etc. Servicios más nuevos como fleet ayudan a manejar los clusters asegurando que un número determinado de instancias de servicios está siempre funcionando. FastPatch permite actualizaciones atómicas de CoreOs usando un esquema de partición de raíz activo-pasivo y ayudando con una restauración rápida en caso de que hayan problemas. Esos nuevos desarrollos hacen que valga la pena investigar CoreOS si ya están acostumbrados a Docker.

Heroku, con su modelo de aplicación de 12 factores, ha cambiado nuestra forma de pensar en la construcción, el despliegue y el hosting de applicaciones web. Deis encapsula el modelo PaaS de Heroku en un framework de open-source que se despliega sobre contenedores Docker sin importar su ubicación. Deis sigue evolucionando, pero en cuanto a aplicaciones que usan el modelo de 12 factores, él tiene el potencial de simplificar el despliegue y hosting en cualquier entorno. Deis es otro ejemplo de la riqueza del ecosistema de plataformas y herramientas que aparece alrededor de Docker.

La analítica predictiva se usa en cada vez más productos, frecuentemente con orientación al usuario final. H2O es un nuevo paquete open-source interesante (con una startup detrás de él) que hace la analítica predictiva accesible a los equipos de proyectos debido a su interfaz fácil de usar. Al mismo tiempo, se incorpora con las herramientas preferidas de los data scientists, R y Python, así como Hadoop y Spark. Ofrece un desempeño excelente y, en nuestra experiencia, integración fácil al tiempo de ejecución, sobre todo en plataformas basadas en JVM.

Jackrabbit Oak, anteriormente llamado Jackrabbit3, es una implementación expansible de almacén de contenido jerárquico que se usa como base de sistemas de manejo de contenido. Soporta soluciones de almacenaiemnto basados en archivos, así como almacenamiento en MongoDB y RDMS, que son preferibles en escenarios de gran volumen de uso.

Aunque se implementa en Java, se puede acceder desde varias plataformas a través de estandares como JCR

Mientras robustecer el servidor es una estrategia vieja y considerada como común por los administradores de sistemas que han manejado sistemas de producción, aún no es común en la comunidad desarrolladora. Sin embargo, el aumento en la cultura DevOps ha traído un nuevo enfoque en herramientas como SELinux AppArmor y Grsecurity que intentan simplificar el proceso, al menos en el ecosistema Linux. Cada una de esas herramientas viene con sus propios puntos fuertes y débiles, lo que hace muy difícil escoger una como la mejor y única. Dicho esto, recomendamos a todos los equipos que escogan el correcto Modulo de Seguridad Linux para su caso y que hagan del robustecimiento del servidor y la seguridad una parte de su flujo de trabajo de desarrollo.

Después de la adquisición de MySQL por parte de Oracle, su version empresarial incluye cada vez más módulos de código cerrado. Hay mucha preocupación sobre el futuro de MySQL. MariaDB es una derivación GPL desarrollada por la comunidad, planeada como exclusivamente open-source mientras mantiene su compatibilidad y competitividad con MySQL. Cuenta con adeptos de alto perfil como Google y Wikipedia, así como distribudores Linux claves RedHat y SUSE.

Mientras estamos reluctantes de recommendar la adopción completa de NEtflix OSS Full stack, a menos que no estés entrando al negocio de video distribuido globalmente, el stack es lleno de ideas interesantes con sus implementaciones open-source. Algunas de las herramientas, como Asgard, son fuertemente acopladas en una arquitectura virtualmente preconfigurada, lo que los hace difíciles de usar individualmente. Otras herramientas, como Ice y Hystrix, que ya hemos mencionado en el Radar, se pueden usar individualmente. Creemos que los equipos deben entender las ideas y estrategias encapsuladas en las herramientas, incluso cuando eligen no hacer uso del pack entero.

Cuando Oracle suspendió el desarrollo del OpenSSO de Sun, una plataforma open-source para administración de aceso ForgeRock lo integró en su Open Identity Suite. Actualmente llamado OpenAM, llena el nicho de mercado por una plataforma open-source extendible que soporta OpenID Connect y SAML 2.0. Pero, a causa de su larga historia, OpenAM posee una base de código desgarbada, cuya documentación puede ser inescrutable. Esperamos que salga pronto una alternativa optimizada con mejor soporte para despliegue y aprovisionamiento automatizados.

SDN es un tema amplio, pero con cada vez más importancia. La habilidad de configurar nuestros aparatos de red por medio de software desdibuja el punto final del despliegue de aplicaciones. Incluye todo desde aparatos de red virtuales, como Load Balancers de AWS y Flannel de CoreOS, hasta equipos de red que soportan estandares como OpenFlow. Mientras los proveedores de la nube se han enfocado en computación y almacenamiento antes, esperamos que la tropa de herramientas SDN sea más eficiente en cuanto al manejo de sistemas locales y remotos.

Spark es una solución full stack para servicios conectados a la nube. Spark Photon es un microcontrolador con un módulo wifi. Spark Electron es una variante que se conecta a una red celular. Spark OS añade un API REST a los aparatos. Eso simplifica la entrada a IoT y la construcción de sus propios aparatos conectados.

Rapidpro ofrece la capacidad de instalar o modificar facilmente aplicaciones de mensajes cortos y complejos para negocios sin la necesidad de un desarrollador. Debido al costo menor de los mensajes de texto en comparación a sesiones USSD, ésta es una manera acsequible de construir aplicaciones escalables con enfoque en teléfonos feature* y hemos tenido éxito en nuestros proyectos. Los flujos son bastante simples de construir y se pueden desencadenar las acciones (como enviar un mensaje de texto, correo electronico o incluso llamar a un api externo) en cualquier momento.

Una Base de Datos de Series de Tiempo (TSDB) es un sistema optimizado para manejar datos de series de tiempo. Permite que los usuarios ejecuten operaciones CRUD sobre varias series de tiempo organizadas como proyectos de base de datos. También provee la habilidad de hacer cálculos estadísticos sobre la serie entera. Aunque no son una tecnología completamente nueva, estamos viendo un nuevo interés en estas bases de datos, sobre todo en el área de aplicaciones de IoT. Esto se está facilitando por la gran cantidad de plataformas open-source y comerciales, como OpenTSDB, InfluxDB, Druid, BlueFloodDB, que han aparecido últimamente. También vale la pena mencionar que algunos de estos sistemas usan como motor subyacente de almacenamiento otras bases de datos distribuidas como Cassandra y HBase.

Asegurar cuentas en línea es a la vez extremadamente importante y notoriamente difícil. La autenticación de dos pasos aumenta grandemente la seguridad, y hemos recomendado TOTP como una buena solución. Una nueva solución en este campo es U2F, basada en criptografía de clave pública y USBs baratos.

Mientras fue desarrollada en Google, ahora es un estandar manejado por la Allianza FIDO. Nos gusta la promesa de mejor protección contra el phishing y ataques man-in-the-middle, pero nos preocupa el hecho de que el punto referencial actual del estandar es un algoritmo de firma digital de curva elíptica considerado defectuoso.

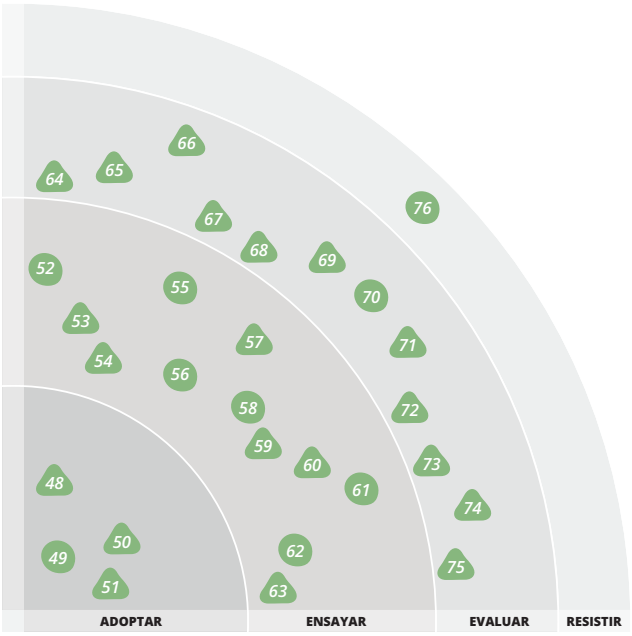
El crecimiento de contenedores, servidores phoenix y entrega continua ha visto una derivación de la estrategia usual para el despliegue de aplicaciones web. Tradicionalmente hemos construido un artefacto y lo hemos instalado en un servidor de aplicaciones. El resultado fue un largo ciclo de retroalimentación para los cambios, tiempos más largos de construcción, y gastos significativos de manejo de estos servidores de aplicaciones en producción. Muchos de ellos son también difíciles de automatizar. La mayoría de los equipos con los que trabajamos prefieren poner un servidor http incorporado en la aplicación web. Hay muchas opciones: Jetty, SimpleWeb, Webbit y Owin Self-Host, entre otras.

La facilidad de automatización y despliegue y la reducción de la cantidad de infraestructura a manejar nos hace recomendar servidores embebidos en lugar de servidores de aplicaciones para futuros proyectos. OSGi es una especificación que trata de corregir la falta de un sistema modular para Java que permita la recarga dinámica de componentes. Mientras algunos proyectos tienen éxito (p.ej. Eclipse), otros usos han expuesto los riesgos de añadir abstracciones a plataformas que nunca fueron diseñadas para ello. Proyectos que dependen de OSGi para la definición de sistemas de componentes, se dan cuenta rápidamente de que la especificación resuelva una parte muy pequeña del problema y añade complejidad accidental, como construcciones más complejas. La mayoría de los proyectos actuales usan archivos JAR o arquitecturas de microservicios para manejar componentes y esperan la solución nativa en Java en la especificación modular Jigsaw.

El protocol SPDY fue desarrollado por Google desde el 2009 como un experiemnto para proveer un protocolo alternativo para manejar faltas de desempeño de HTTP/1.1. El nuevo protocolo estándar HTTP/2incluye muchas de las caracteristicas de desempeño de SPDY y Google anunció que va a abandonar el soporte de SPDY en navegadores para el principio de 2016. Si su application requiere caracteristicas de SPDY, le recomendamos usar HTTP/2.

HERRAMIENTAS

Aunque la idea del manejo de dependencias no es nuevo y se considera como una práctica fundamental de desarrollo, no es ampliamente adoptada por la comunidad PHP. **Composer** es una herramienta para manejar dependencias en PHP. Es considerablemente influenciada por herramientas de otros stacks tecnológicos como npm de Node y Bundler de Ruby. Estamos viendo una adopción amplia en proyectos PHP y es una herramienta bastante madura. Talvez se necesitarían modificaciones para librerias internas, pero se puede usar para la mayoría de las librerias externas.



La entrega continua de software de alta calidad de una manera rápida y confiable, requiere la coordinación de una gran cantidad de pasos automatizados. Go CD es un herramienta open-source construida por ThoughtWorks para manejar este escenario, con el concepto de canales de despliegue en el centro. Maneja flujos de trabajo complejos en nodos múltiples y habilita la promoción transparente y trazable en varios ambientes. Aunque es posible construir canales de despliegue sobre herramientas de entrega continua, nuestros equipos pueden ver los beneficios derivados de una herramienta construida especificamente para este trabajo.

La buena prueba de componentes en un sistema empresarial es crucial y se necesita mejor automatización en este campo debido al énfasis creciente en la separación basada en servicios y automatización del despliegue, factores claves para el éxito de microservicios. El término industrial "visualización de servicios" se refiere a herramientas que pueden emular componentes específicos en un ambiente tal. Hemos visto mucho éxito con Mountenbank, una herramienta ligera para stubbing y mocking HTTP, HTTPS, SMTP y TCP.

Postman es una extensión de Chrome que actua como cliente REST en el browser y permite la creación de solicitudes e inspección de respuestas. Es una herramienta útil en el desarrollo de API o la implementación de un cliente para API. Postman permite la adición de identificadores Oauth1 y Oauth2 a solicitudes cuando sea necesario. La respuesta está disponible en formato JSON adornado o XML. También se puede recuperar la historia de solicitudes para editar rápidamente y probar la respuesta del API con datos diferentes. Así mismo, ofrece un juego de extensiones que permiten su uso como herramienta de evaluación completa, aunque desalentamos el estilo de pruebas "record y replay" que ella promueve.

Boot2Docker es una distribución ligera de Linux que usa Docker, presentada como VM para OSX y Windows. Es una manera maravillosa de empezar a experimentar con Docker. Para los equipos que usan microservicios, también puede ser una manera efectiva de ejecutar servicios múltiples en una máquina local para fines de desarrollo y pruebas, donde la sobrecarga de múltiples VM de vagrant puede ser demasiado.

ADOPTAR
48. Composer
49. Go CD
50. Mountebank
51. Postman

ENSAYAR
52. Boot2docker
53. Brighter
54. Consul
55. Cursive
56. GitLab
57. Hamms
58. IndexedDB
59. Polly
60. REST-assured
61. Swagger
62. Xamarin
63. ZAP

EVALUAR
64. Apache Kafka
65. Blackbox
66. Bokeh/Vega
67. Gor
68. NaCl
69. Origami
70. Packetbeat
71. pdfmake
72. PlantUML
73. Prometheus
74. Quick
75. Security Monkey

RESISTIR
76. Citrix for development

HERRAMIENTAS *continua*

Brighter es una librería open-source para .Net que provee un scaffold para la implementación de Command Invocation. Hemos tenido buenos comentarios de los equipos que lo usan, sobre todo en conjunto con el patrón de puertos y adaptadores; y CQRS. Les gusta particularmente que se incorpora con Polly para proveer funcionalidad interruptoria.

Nos sigue impresionando Consul, una herramienta de descubrimiento de servicios que soporta mecanismos de descubrimiento basados en DNS y HTTP. Hace más que otras herramientas de este tipo porque provee pruebas para servicios registrados que se pueden personalizar, y asegura que las instancias poco saludables estén marcadas correctamente. Han emergido más herramientas que trabajan en conjunto con Consul para hacerlo aún mas poderoso.

Consul Template permite que los archivos de configuración sean llenos de información de Consul, lo que facilita cosas como el balanceo de carga del lado del cliente usando mod_proxy. En el mundo de Docker, registrator puede registrar automaticamente y con poco esfuerzo los contenedores de Docker en cuanto aparezcan con Consul, lo que facilita el manejo de instalaciones basadas en contenedores.

Cursive es un IDE Clojure que funciona como plugin para IntelliJ. Mientras está en etapa de acceso temprano, lo consideramos muy útil para trabajar con bases de código más grandes de Clojure.

Cursive provee un fuerte soporte para renombrar y navegación, es estable y confiable, y es excelente en ambientes con lenguajes de JVM diferentes. Para las organizaciones que adoptan Clojure, Cursive ha sido muy útil en facilitar la transición para los desarrolladores.

Gitlab es una plataforma de almacenamiento local que da a los equipos de desarrollo de software privado el flujo de trabajo conocido y ubicuo que servicios hospedados de control de versiones como Github y BitBucket proveen para los desarrolladores de OSS. Mientras hay una versión gratis para el público, la version comercial provee soporte e integración profunda con servidores LDAP.

Muchas historias de fracasos en nuestra industria son causadas por la suposición de que las redes son siempre fiables y los servidores reaccionan rápidamente y correctamente todo el tiempo.

Hamms es una herramienta open-source interesante que actua como un servidor HTTP malcriado: ella provoca una cantidad de fracasos, incluso fallas de conexión o respuestas lentas o mal-formadas. Puede servir para probar si el software maneja fracasos con elegancia.

Con la creciente disponibilidad y viabilidad de aplicaciones de una sóla página y configuración de arranque fuera de línea, viene una necesidad de persistir datos en el navegador web. El almacenamiento local es muy fácil de usar y bien soportado por parte de los navegadores. Para casos de uso más complejos, existe IndexedDB. Aún si puediera ser una solución buena, recomendamos que se use sólo cuando sea necesario, debido al aumento en complejidad y el API tosco. También hemos tenido experiencia positiva de LocalForage - un framework que provee un nivel de abstracción sobre las soluciones de persistencia.

Varios de nuestros equipos que trabajan en proyectos .Net han recomendado Polly como útil en la constucción de sistemas basados en microservicios. Éste fomenta la expresión fluida de políticas temporales para el manejo de excepciones y patrones de desgravación, incluso políticas como Retry, Retry Forever y Wait and Retry. Ya existen librerías en otros lenguajes (p.ej. Hystrix para Java) y Polly es una bienvenida adición de la comunidad .Net

REST-assured es un lenguaje particular del dominio de Java para probar y validar los servicios RESTful. Sirve para simplificar las pruebas de servicios basados en REST y construidos por encima de HTTP Builder. REST-assured soporta las varias peticiones REST y se puede usar para validar y verificar las respuestas de los APIs. Así mismo provee validación de esquema JSON, y por supuesto, se puede usar para validar que los endpoints devuelvan los tipos correctos de datos esperados.

Swagger es una manera estandar de describir un API RESTful para que se puedan generar automaticamente documentos legibles para humanos y ejemplos de clientes. La actualización a versión 2.0 provee algunas mejoras importantes de flexibilidad y la lista de herramientas para generar documentación continúa creciendo. Hay algunas alternativas al Swagger que asoman de la comunidad de los vendedores, como RAML y API Blueprint.

Nos emociona el progreso de Xamarin en ofrecer una opción sólida para construir aplicaciones móviles para todas plataformas.

Soporta C# y F# como lenguajes primarios, con conexiones a SDKs específicos a la plataforma, y el ambiente runtime Mono que funciona en iOS, Android y Windows phone. Las aplicaciones han sido compiladas en código nativo, lo que les da un look and feel más natural. El uso de este juego de herramientas requiere la separación de la capa específica de UI de las demás, para asegurar la reutilización del código en diferentes plataformas. La reciente apertura del código de la plataforma .NETdebe traer beneficios a Xamarin en términos del acceso más amplio a las herramientas .NET y la facilitación del desarrollo sobre otros sistemas operativos.

El ZED Attack ProxyZAP es un proyecto de OWASP que permite rastrear un sitio para encontrar vulnerabilidades de seguridad de una manera automática. Puede ser parte de pruebas de seguridad periódicas o integradas en un canal CD para prover pruebas contínuas para vulnerabilidades comunes. El uso de una herramienta como ZAP no reemplaza la necesidad de pensar en la seguridad y hacer otras pruebas más profundas, pero como herramienta adicional para asegurar que nuestros sistemas son seguros, es una buena adición a la caja de herramientas.

Gran parte de los desarrollos recientes en software empresarial, giran por secuencias asíncronas de secuencias de eventos inmutables en vez de solicitudes síncronas punto-a-punto que modifican estados.

Apache Kafka es un framework de mensajes open-source que soporta este estilo de arquitectura por medio de la publicación de feeds de mensajes ordenados hacia muchos consumidores independientes ligeros. El diseño único de Kafka permite que el número de consumidores se extienda al mismo tiempo que mantiene un orden estricto de los mensajes.

Blackbox es una herramienta simple para la codificación de archivos específicos mientras éstos se encuentran en el repositorio fuente. Esto puede ser muy útil si se guardan contraseñas o claves privadas. Blackbox funciona con Git, Mercurial y Subversion, y usa GPG para la codificación. Cada usuario posee su propia clave, lo que facilita la revocación del aceso en un nivel granular. Hay mucha actividad en esta area y un par de jugadores a considerar, incluso git-crypt y Trousseau.

En el mundo de data science y análisis, la mayor parte del trabajo se hace usando Python y R, lenguajes que, tristemente, ofrecen muy pocas opciones para el trazo de las visualizaciones accesible a la web. Una estrategia es convertir el resultado del análisis en algo fácil en términos de visualización e interacción en el browser. Conocemos de dos herramientas que intentan hacer eso: Bokeh es una librería Python y JavaScript que permite la creación de visualizaciones interactivas “en el estilo de D3.js” pero de gran rendimiento de conjuntos de datos grandes o corrientes. Vega es una gramática de visualización declarativa para D3 que consume conjuntos de datos JSON generados por el servidor y traduce descripciones de visualización en código D3.js

Gor es una herramienta open-source para capturar y reproducir tráfico real HTTP en un entorno de pruebas con el fin de probar de manera contínua tu sistema con data real. Se puede usar para aumentar la confianza en el despliegeue del código, los cambios en configuración o infraestructura.

La libreria NaCl (que se pronuncia Salt)provee un conjunto de características para encripción, desencripción y firmas, diseñadas para facilitar la implementación de comunicación segura dentro de una red u otros requerimientos criptográficos. Aunque existen esas funciones en otras librerías, NaCl promete velocidad más alta y APIs más fáciles de usar. Actualmente hay soporte para C y C++, con envolventes Python en el camino.

Origami es una herramienta gratuita para diseñar prototipos de usuarios con una gran variedad de atajos para las funciones más comunes. Provee la posibilidad de exportar los prototipos como fragmentos de código a Objective-C para iOS, Java para Android y Javascript para Web. Se puede usar para construir rápidamente prototipos interactivos de cara al usuario y probar flujos de usuario. Recomendamos investigar esta herramienta si el caso de uso se ajusta a la experiencia que hemos recogido de algunos de nuestros equipos.

La creciente complejidad de los sistemas distribuidos requiere herramientas que ayuden a entender su comportamiento en producción.

Packetbeates una herramienta open-source que usa agentes para interceptar tráfico entre nodos, lo que permite ver patrones de tráfico, la tasa de errores y otras informaciones útiles. Requiere Elasticsearch y Kibana para funcionar, pero si ya usas estas herramientas como parte de la agregación de logs, puede ser una adición sencilla para dar más percepción de tu sistema de producción.

Pdfmake es una libreria JavaScript que permite la creación e impresión de documentos PDF directamente en el navegador. Para usar Pdfmake, se necesita la construcción de un objeto de tipo documento que soporte elementos estructurales como tablas, columnas

y estilos enriquecidos, luego los métodos helper pueden crear, imprimir o descargar un PDF sin dejar JavaScript del lado del cliente.

El desarrollo de un sistema de software por medio de trazar un gran número de diagramas detallados es una estrategia que, en nuestra experiencia, no se compara favorablemente con las alternativas. Sin embargo, describir una parte del sistema que es especialmente compleja e intrincada por medio de un diagrama es una buena idea, y el UML ofrece una cantidad de diagramas útiles y comúnmente entendidos. Nos gusta PlantUML para la creación de estos diagramas, porque permite la expresión de la intención de dichos en una forma textual clara, sin tener que lidiar con herramientas gráficas sobrecargadas. El hecho de tener una forma textual permite también el versionamiento y almacenamiento de la mano del código de fuente.

SoundCloud acaba de liberar el código de una alternativa a Graphite: Prometheus. Desarrollado como una reacción a las dificultades con Graphite, **Prometeus** funciona diferente, en primer lugar soporta el modelo HTTP basado en pull (aunque un modelo push más similar a Graphite se soporta también). Al mismo tiempo, va más allá de Graphite, porque es construido para soportar alertas basadas en métricas capturadas, lo que lo hace una parte mucho más activa de un juego de herramientas operacionales. Vale la pena tener cuidado con la adopción de nuevas tecnologías en el espacio de monitoreo de producción, pero por el momento, se reporta que SoundCloud lo usa felizmente en su producción, y Docker está contribuyendo al desarrollo continuo.

Quick es un framework de pruebas para Swift y **Objective-C** que viene en conjunto con Nimble, un framework de comparaciones para pruebas.

Quick sirve para verificar el comportamiento de los programas en Swift y Objective-C. Tiene el mismo sabor sintáctico que rspec y jasmine, y es fácil de implementar. Es muy eficiente, permite la afirmación de tipos de datos y facilita la prueba de código asíncrono.

Security Monkey es otra herramienta en el Netflix Simian Army – un juego de herramientas diseñado para asegurar que los sistemas se construyen en una manera elástica.

Además de proveer una evaluación (configurable) de las vulnerabilidades potenciales de seguridad en la configuración AWS, se puede usar también para monitorear a los cambios contínuos y alertar a los diferentes grupos necesarios. Coincide en algunos términos con el Trusted Advisor Report y Cloudtrail de AWS, porque fue dessarollada antes de que ambos servicios estuvieran disponibles al público, pero sus capacidades superan a estos servicios. Si ningúno de los mencionados sirve, Security Monkey vale la pena.

Por razones de seguridad y conformidad, los equipos distribuidos tienen que usar Citrix para conectarse a una máquina virtual en sitio, donde se desarrolla. A pesar de ser una buena herramienta en algunos casos de uso, Citrix provee una pésima experiencia de desarollo remoto y frecuentemente paraliza el equipo distribuido. Hay muchas mejores soluciones técnicas, como la máquina remota NoMachine o Cloud9 IDE, que proveen una experiencia mucho más práctica.

Mejor aún sería abordar los problemas subyacentes de la seguridad y conformidad. Su equipo de desarrollo remoto sería mucho más productivo si le tuviera confianza en cuanto a tener el código fuente con el que estan trabajando en sus propias máquinas.

LENGUAJES & FRAMEWORKS

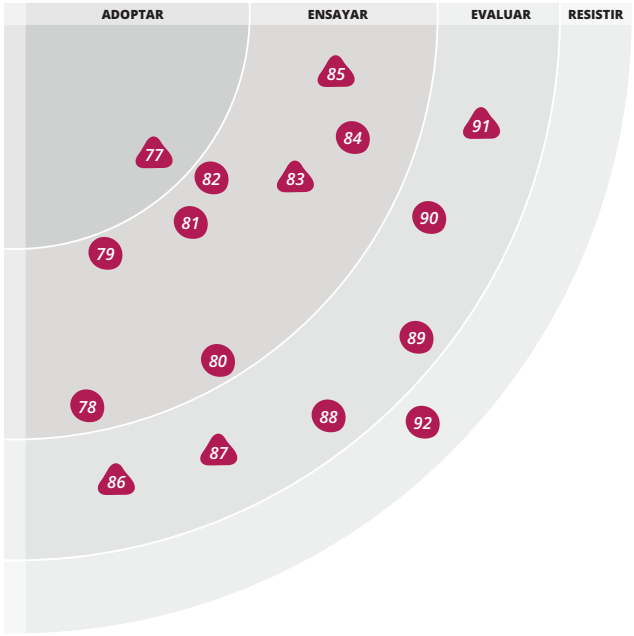
Desde la última vez que discutimos **Nancy** en el Technology Radar, éste se ha convertido en la opción por defecto en nuestros proyectos. Las arquitecturas centradas en rodajas verticales y microservicios simplemente requieren opciones de despliegue ligeras y automatización poco ceremoniosa.

La importancia de visualizaciones grandes y claras ha sido mencionada muy frecuentemente y nosotros valoramos la estrategia de que todos tengan entendida la información clave sobre el estado del software o del equipo. **Dashing** es un sistema de tableros basado en ruby que hemos usado por un largo tiempo para crear visualizaciones optimizadas para pantallas grandes. Es bastante maleable y permite la acumulación de información de varias fuentes como sistemas de build, herramientas de rastreo de tickets o historias, o sistemas de monitoreo de producción.

Hemos usado el framework django rest, que es flexible y se puede personalizar, y que facilita la construcción de APIs web, en algunos de nuestros proyectos. Permite la construcción de APIs restful con django, expone los endpoints del API que se pueden acceder desde el front-end de los consumidores. Django rest ofrece un API web navegable que permite que los desarrolladores visualicen la transferencia de datos a través del API y devuelve ejemplos de las respuestas que la aplicación consumidor recibirá. Provee un número de esquemas de autenticación de paquete y permite la implementación de esquemas a medida.

Ionic framework es un framework de código abierto para front-end que ofrece una librería de componentes y herramientas javascript, html optimizado para móviles y css para construir applicaciones altamente interactivas. Esta construido con SASS y optimizado para AngularJS. Hemos tenido éxito en varios de nuestros proyectos empleando este framework, con su facilidad de instalación y pruebas. Vale la pena investigar este framework cuando es importante el desempeño y la integración sin problemas de un framework para front-end.

Nashorn es un nuevo motor JavaScript para Java, lanzado con Java 8. Es la herramienta elegida en el mundo de Java



a pesar de su tosquedad cuando el mismo código se debe ejecutar en el browser y en el servidor, lo que pasa muy frecuentemente bajo la lógica de validación y migración de datos. No estamos convencidos de que el uso de Nashorn para alojar aplicaciones enteras, a través del soporte Node o del proyecto Avatar, es una buena idea.

Hemos visto mucho interés en Om, un envolvente en ClojureScript del framweork de programación front-end ReactJS de Facebook. Om hace uso de la inmutabilidad inherente de ClojureScript y permite características automáticas como ver el panorama del estado del UI y deshacer. Debido a la eficacia de las estructuras de datos de ClojureScript, algunas aplicaciones Om funcionan más rápidamente que aplicaciones identicas basadas en el framework crudo subyacente React. El ecosistema de componentes y aplicaciones alrededor de Om está creciendo y nuestros equipos empiezan a usarlo.

Un beneficio de la avalancha continua de frameworks JavaScript para front-end, es que de vez en cuándo

ADOPTAR
77. Nancy

ENSAYAR
78. Dashing
79. Django REST
80. Ionic Framework
81. Nashorn
82. Om
83. React.js
84. Retrofit
85. Spring Boot

EVALUAR
86. Ember.js
87. Flight.js
88. Haskell Hadoop library
89. Lotus
90. Reagent
91. Swift

RESISTIR
92. JSF

aparece una nueva idea que nos hace pensar. **React.js** es un framework UI/View en el que las funciones JavaScript generan HTML en un flujo de datos reactivo. Hemos visto proyectos más pequeños lograr éxito con React.js y los desarrolladores se ven atraídos por su estrategia limpia y composeable para componentización.

Retrofit ofrece una manera confiable de construir clientes http en proyectos Android convirtiendo un API REST en un interfaz Java. Retrofit se integra con okhttp y permite que los desarrolladores provean manejo personalizado de errores para las consultas. Hace análisis JSON automaticamente usando GSON y tiene una comunidad muy bien soportada.

Spring Boot permite la instalación fácil de aplicaciones independientes basadas en Spring. Es ideal para arrancar nuevos microservicios y fácil de desplegar. También facilita el acceso a los datos debido a los mapeos de Hibernate con mucho menos código repetido. Nos gusta el hecho de que SPring Boot simplifica los servicios de Java construidos con Spring, pero hemos aprendido de tener cuidado de la multitud de dependencias. Spring sigue en las sombras.

El uso difundido de AngularJS continúa en los proyectos de ThoughtWorks, aunque no es positiva cada experiencia. Continuamos recomendando que los equipos evaluen si la complejidad adicional de una aplicación Javascript de una sóla página es necesaria para cumplir con los requerimientos. También recomendamos la evaluación de frameworks alternativos y en esta edición del Radar señalamos Ember, que se vuelve cada vez más popular en ThoughtWorks. Se elogia a **Ember** por su estrategia de convención de opinión en lugar de la configuración, un equipo central de commiters receptivo, su desempeño, y soporte a herramientas de build a través de ember-cli.

Queremos señalar a **Flight.js** de entre la multitud de frameworks de JavaScript como un framework ligero para construir componentes. Flight añade comportamiento a nodos DOM sin mucha magia. Su naturaleza guiada por eventos y basada en componentes, promueve la escritura de código desacoplado. Eso hace la prueba de componentes individuales comparativamente fácil. Sin embargo, hay que tener cuidado cuando los componentes necesitan interactuar entre ellos. No hay mucho soporte para las pruebas y existe peligro real de terminar en un infierno de eventos. Dicho esto, nos gusta que FLight use mezclas funcionales para comportamiento, como composición en lugar de herencia.

Aunque hay muchos fanáticos de Haskell entre los aficionados de lenguajes en TW, no lo hemos visto mucho en nuestros proyectos hasta ahora.

Algunos proyectos de código abierto usan los modelos de mapeo/reducción de Hadoop junto con la sintaxis de Haskell, lo que algunos desarrolladores y data scientists encuentran atractivo.

No sabemos quién puso el nombre a Lotus, pero podemos presumir que es una persona demasiado joven como para haber trabajado con tal producto de ofimática. **Lotus** es un nuevo framework MVC basado en Rack, escrito en Ruby que se puede implementar modularmente, lo que nos deja usar sólo las porciones que necesitamos. Es una alternativa moderna al framework monolítico Ruby-on-Rails (que cumplió 10 años este año.) Lotus tiene el potencial de facilitar el desarrollo de pila completa con Ruby MVC.

Reagent ha aparecido como una alternativa minimalista y ligera de Om para envolver React.js en ClojureScript. Mientras Om provee un framework de front-end comprehensivo e idiomático para Clojure, Reagent hace uso de la expresividad de Clojure para enfocarse en componentes simples y DSL legible para escribir HTML. Reagent mantiene el desempeño y comprehensibilidad de React.js por medio de representar HTML en términos de datos Clojure, sin incrustar markup foráneo en el código.

Juzgando de nuestra experiencia laboral, Swift promete mucho. Algunos de sus problemas, como los largos tiempos de compilación, se estan resolviendo. Sin embargo, los cambios continuos de lenguaje requieren esfuerzo adicional en términos de desarrollo y hacen la construcción de versiones anteriores más difícil. Probar y refactorizar aún es problemático. En conclusión, vale la pena considerar Swift en cuanto a proyectos de desarrollo para el ecosistema Apple.

Seguimos viendo equipos que tienen problemas con JSF. **JavaServer Faces** y recomendamos que se evite ésta tecnología. Parece que los equipos escogen JSF porque es un estandar Java EE sin evaluar realmente si les conviene el modelo de programación. Como fallo de JSF podemos señalar su modelo de programación porque éste fomenta el uso de sus propios abstracciones en lugar de abrazar completamente el modelo web subyacente. JSF, como los webforms ASP.NET, trata de crear árboles de componentes con estado encima del markup HTML y el protocol HTTP sin estado. Las mejoras en JSF 2.0 y 2.2, tal como la introducción de vistas sin estado y la promoción de GET son pasos en el sentido correcto, talvez un reconocimiento de que el modelo original era con defectos, pero nos viene demasiado tarde. En vez de manejar la complejidad de JSF, recomendamos que los equipos usen frameworks simples y trabajan en conjunto con tecnologías web como HTTP, HTML y CSS

ThoughtWorks es una compañía de software y una comunidad de individuos apasionados cuyo propósito es revolucionar el diseño, creación y entrega de productos de software. Pensamos de forma disruptiva para ofrecer tecnología que haga frente a los retos más difíciles de nuestros clientes, a la vez que buscamos revolucionar la industria de TI y crear un cambio social positivo. Creamos herramientas innovadoras para equipos de desarrollo de software que aspiran a ser grandes. Nuestros productos ayudan a las organizaciones a mejorar continuamente y entregar software de calidad para sus necesidades más críticas. Fundada hace 20 años, ThoughtWorks ha crecido a partir de un pequeño grupo de personas en Chicago hasta convertirse en una compañía de más de 3000 empleados repartidos a lo largo de sus 30 oficinas en 12 países: Australia, Brasil, Canadá, China, Ecuador, Alemania, India, Singapur, Sudáfrica, Uganda, Reino Unido y los Estados Unidos

ThoughtWorks®