ThoughtWorks®

TECHNOLOGY RADAR *NOV '15*

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

thoughtworks.com/radar

NOVEDADES

Estas son las tendencias sobresalientes en esta edición

DOCKER INCITA LA EXPLOSIÓN DE ECOSISTEMAS DE CONTENEDORES

El uso de contenedores ejemplicado por Docker, es muy popular en un número creciente de organizaciones. El interés varía ampliamente entre organizaciones y dentro de ellas; nuestras recomendaciones se encuentran entre Evaluar y Adoptar. El ecosistema (herramientas, plataformas y técnicas) está creciendo y madurando, acelerando aún más el interés. Los lectores astutos se darán cuenta de los temas similares en nuestro radar, yendo desde Docker como una herramienta de desarrollo para manejar dependencias hasta plataformas de gran tamaño en la nube como Mesos y AWS ECS que usan contenedores como unidad de escalamiento.

MICROSERVICIOS Y HERRAMIENTAS RELACIONADAS GANAN POPULARIDAD

El interés no cesa en este estilo de arquitectura, lo que impulsa transitivamente el interés en las herramientas de soporte y las técnicas alrededor de esto: Prácticas de DevOps como el uso de contenedores, lecciones aprendidas como los peligros de programar en tu herramienta de integración y entrega continua, la madurez de las herramientas para descubrir servicios y otras más. Esperamos ver aún más crecimiento y madurez en este espacio en el futuro cercano.

HERRAMIENTAS JAVASCRIPT ESTABLECIDAS COMO "SOLO CAÓTICAS"

Hemos destacado el volcamiento hacia el mundo de las herramientas javascript, pero la comunidad esta gradualmente calmándose y cohesionándose alrededor de algunas prácticas cómunes. Los equipos están descubriendo la mejor combinación (incluyendo ninguna) para herramientas de construcción y gestión de paquetes, y escuchamos menos desacuerdos entre los equipos acerca de prácticas efectivas.

LA SEGURIDAD ES PROBLEMA DE TODOS

La seguridad es un problema que singularmente afecta a todos los roles a través del ciclo de vida del desarrollo de software. Hemos destacado la mejora en el espacio de la seguridad en el último Radar Tecnológico, y estamos complacidos de ver a los equipos utilizando prácticas de seguridad dentro de su SDLC. En esta edición también destacamos enfoques innovadores como recompensa de errores, modelo para enfrentar amenazas, HSTS, TOTP y encripción. Esperamos que la tracción continúe para seguir mejorando en este aspecto.

QUIENES CONTRIBUYEN

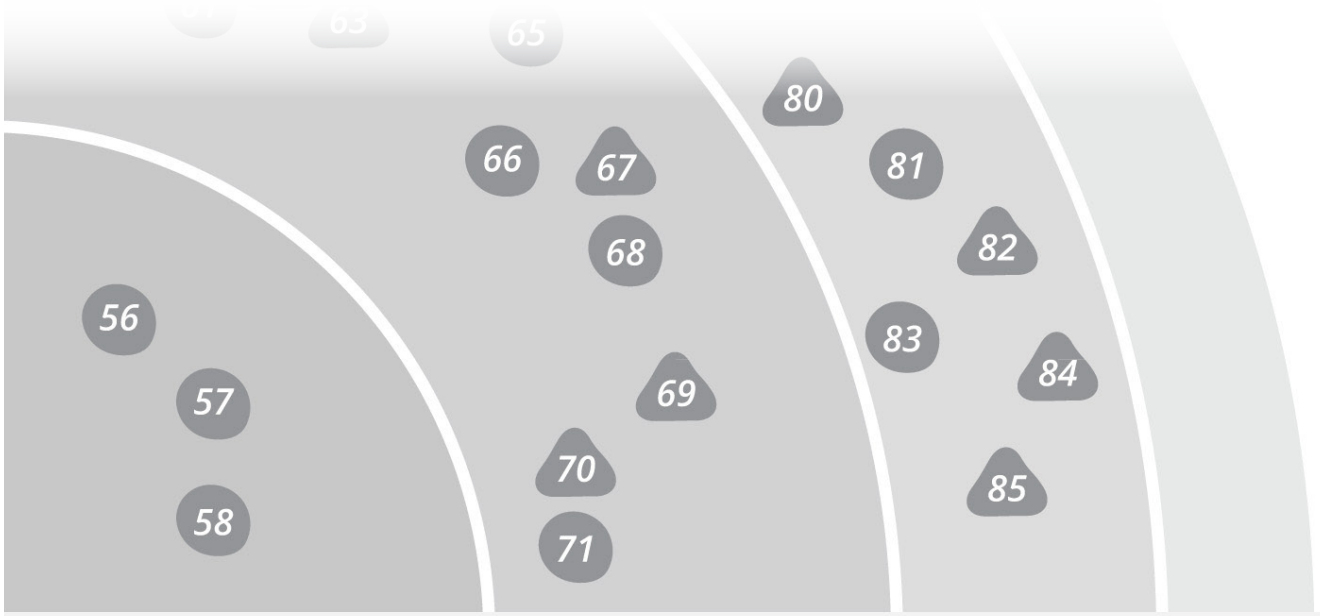
El Radar de Tecnología es preparado por el Comité Consultor de Tecnología de ThoughtWorks, compuesto por:

Rebecca Parsons (CTO)	Claudia Melo	Ian Cartwright	Rachel Laycock
Martin Fowler(Chief Scientist)	Dave Elliman	James Lewis	Sam Newman
Anne J Simmons	Erik Doernenburg	Jonny LeRoy	Scott Shaw
Badri Janakiraman	Evan Bottcher	Mike Mason	Srihari Srinivasan
Brain Leke	Hao Xu	Neal Ford	Thiyagu Palanisamy

ACERCA DEL TECHNOLOGY RADAR

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 apropiado en 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
2. Decoupling deployment from release nuevo
3. Generated infrastructure diagrams
4. NoPSD
5. Products over projects
6. Threat Modelling

ENSAYAR
7. BEM nuevo
8. BFF - Backend for frontends nuevo
9. Docker for builds nuevo
10. Event Storming nuevo
11. Flux
12. Idempotency filter nuevo
13. iFrames for sandboxing nuevo
14. NPM for all the things nuevo
15. Offline first web applications
16. Phoenix Environments
17. QA in production nuevo

EVALUAR
18. Accumulate-only data
19. Bug bounties nuevo
20. Data Lake
21. Hosted IDE's nuevo
22. Monitoring of invariants nuevo
23. Reactive Architectures

RESISTIR
24. Gitflow nuevo
25. High performance envy/web scale envy nuevo
26. Microservice envy
27. Pace-layered Application Strategy
28. Programming in your CI/CD tool
29. SAFe™
30. Separate DevOps team

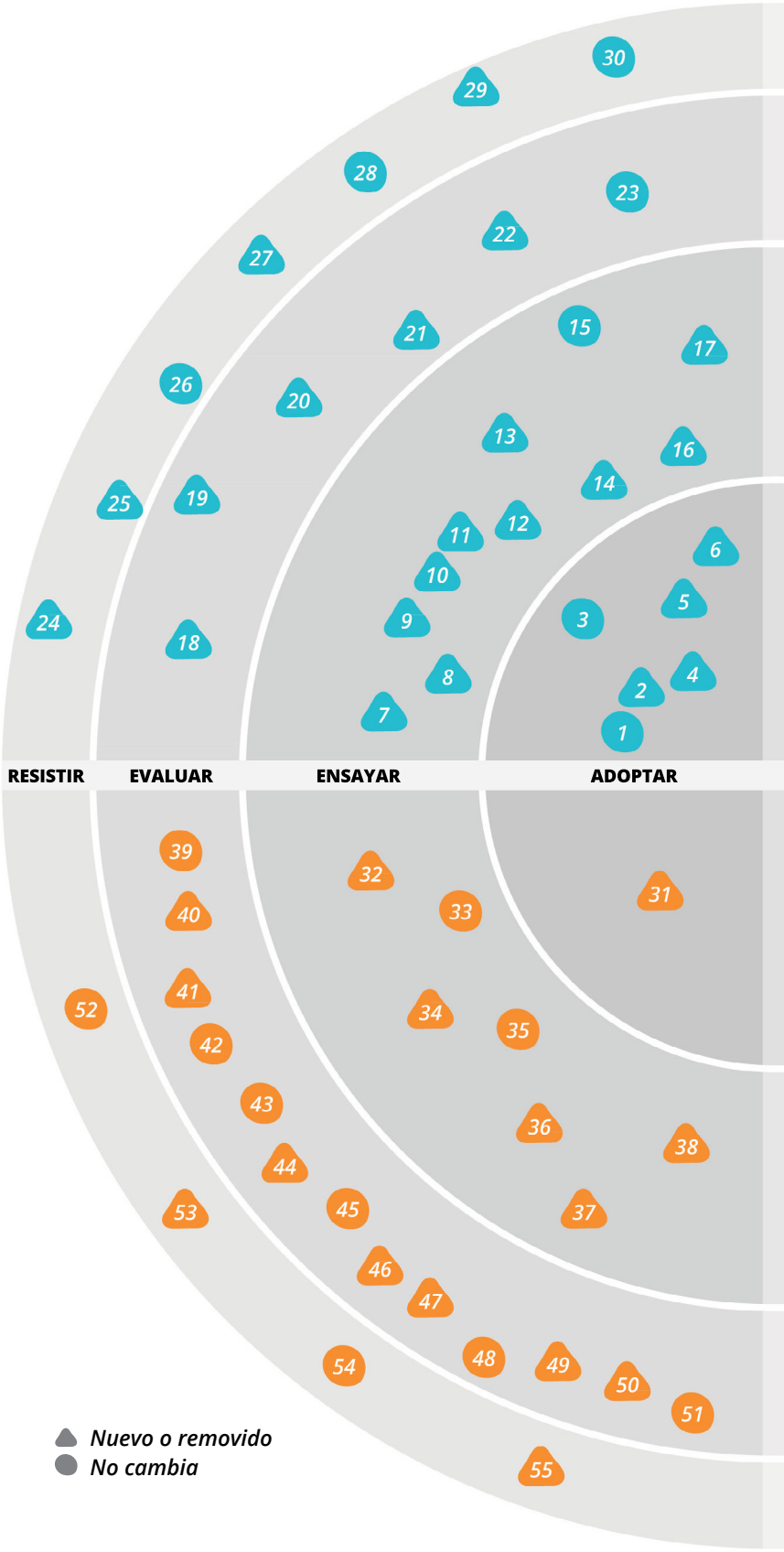
PLATAFORMAS

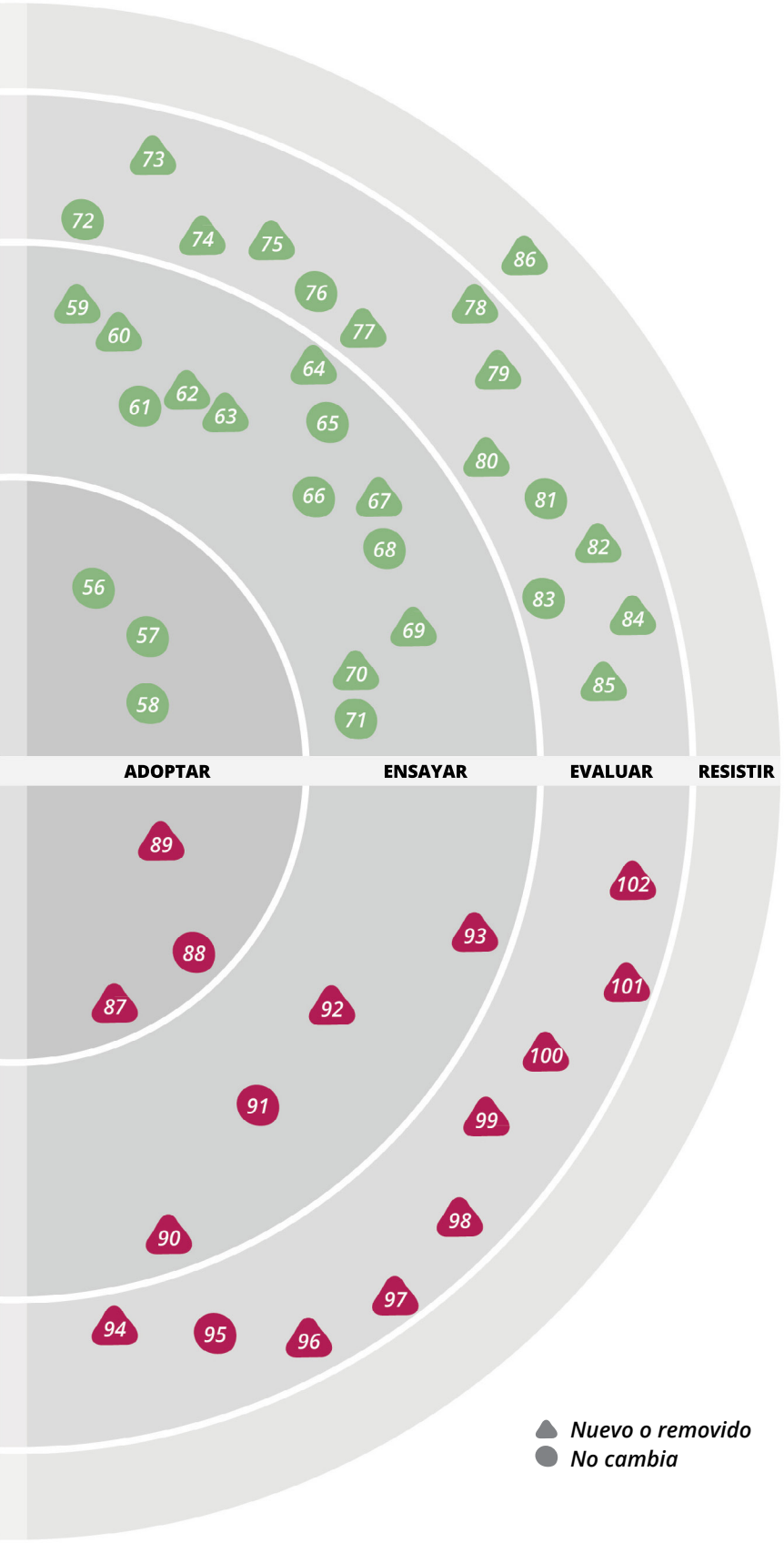
ADOPTAR
31. TOTP Two-Factor Authentication

ENSAYAR
32. Apache Mesos
33. Apache Spark
34. AWS Lambda nuevo
35. Cloudera Impala
36. Fastly nuevo
37. H2O
38. HSTS nuevo

EVALUAR
39. Apache Kylin
40. AWS ECS nuevo
41. Ceph nuevo
42. CoreCLR and CoreFX
43. Deis
44. Kubernetes nuevo
45. Linux security modules
46. Mesosphere DCOS nuevo
47. Microsoft Nano Server nuevo
48. Particle Photon/Particle Electron
49. Presto nuevo
50. Rancher nuevo
51. Time series databases

RESISTIR
52. Application Servers
53. Over-ambitious API Gateways nuevo
54. SPDY
55. Superficial private cloud nuevo





EL RADAR

HERRAMIENTAS

ADOPTAR
56. Composer
57. Mountebank
58. Postman

ENSAYAR
59. Browsersync nuevo
60. Carthage nuevo
61. Consul
62. Docker Toolbox nuevo
63. Gitrob nuevo
64. GitUp nuevo
65. Hamms
66. IndexedDB
67. Polly
68. REST-assured
69. Sensu nuevo
70. SysDig nuevo
71. ZAP

EVALUAR
72. Apache Kafka nuevo
73. Concourse CI nuevo
74. Espresso nuevo
75. Gauge nuevo
76. Gor
77. ievms nuevo
78. Let's Encrypt nuevo
79. Pageify
80. Prometheus
81. Quick
82. RAML nuevo
83. Security Monkey
84. Sleepy Puppy nuevo
85. Visual Studio Code nuevo

RESISTIR
86. Citrix for development

LENGUAJES & FRAMEWORKS

ADOPTAR
87. ECMAScript 6 nuevo
88. Nancy
89. Swift

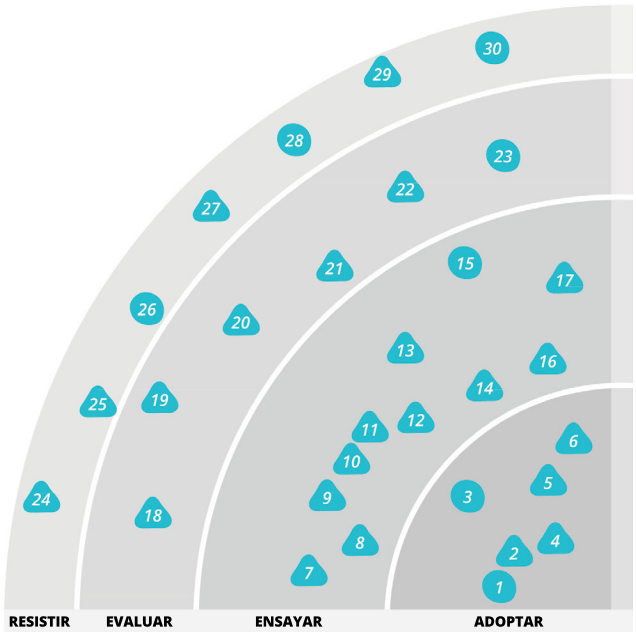
ENSAYAR
90. Enlive nuevo
91. React.js
92. SignalR nuevo
93. Spring Boot

EVALUAR
94. Axon nuevo
95. Ember.js
96. Frege nuevo
97. HyperResource nuevo
98. Material UI nuevo
99. OkHttp nuevo
100. React Native nuevo
101. TLA+ nuevo
102. Traveling Ruby nuevo

RESISTIR

TÉCNICAS

La implementación de la Entrega Continua sigue siendo un reto para muchas organizaciones y continúa siendo importante destacar técnicas útiles tales como **desacoplar el despliegue del lanzamiento**. Recomendamos usar estríctamente el término *Despliegue* al referirse al acto de implantar un cambio en componentes o infraestructura de una aplicación. El término *Lanzamiento* debe utilizarse cuando se divulga un cambio de características a usuarios finales, lo que causa un impacto empresarial. Utilizando técnicas tales como activación de opciones y "dark launches" (bifurcaciones) podemos desplegar cambios en sistemas de producción con más frecuencia sin lanzar características. Despliegues más frecuentes disminuyen el riesgo asociado con el cambio, mientras que las partes interesadas del negocio retienen el control de cuándo se divulgan las opciones a los usuarios finales.



El diseño "Justo a Tiempo" es un concepto importante y útil de diseño visual que el movimiento **NoPSD** intenta captar. No hace falta diseñar toda la aplicación ni cada elemento de la interfaz de usuario, desde el inicio del proyecto. Diseñe las cosas según las necesite con las herramientas más livianas que pueda utilizar. Hemos visto un crecimiento correspondiente en herramientas más sencillas con aprendizajes más rápidos, tales como **Sketch**, además de un retorno a lápiz y papel (especialmente si se combina con una **guía de estilo digital**. Debido a las limitaciones de modelos planos al diseñar para monitores, también se han vuelto cada vez más comunes y valiosas para comunicar las intenciones del diseño las creaciones de prototipos de fidelidad variable con herramientas tales como **Invision**, **FramerJS** ,**Origami** o sencillamente HTML/CSS y un poco de JavaScript.

Por mucho tiempo, hemos promovido la idea de que pensar en el desarrollo de software como un proyecto, es decir con un presupuesto y entrega en un plazo limitado, no encaja con las necesidades de las empresas modernas. Los esfuerzos de desarrollo de software importantes deben ser un producto continuo que apoya y reflexiona sobre el proceso empresarial que están apoyando. Tales esfuerzos estarán completos solo cuando dejen de ser útiles el proceso empresarial y su software. Nuestra observación sobre este enfoque de **productos en vez de proyectos**, tanto para nuestros propios proyectos como aquellos externos, nos ayuda a determinar que este es el enfoque que debe utilizarse en todos los casos, excepto en casos excepcionales.

Debido a la gran cantidad de fallas en la seguridad de alto perfil, en los últimos meses, los equipos de desarrollo de software ya no necesitan convencerse de que deben enfatizar el desarrollo de software seguro y tratar de forma responsable los datos de sus usuarios. Los equipos enfrentan un proceso arduo de aprendizaje, sin embargo, y puede ser abrumadora la gran cantidad

ADOPTAR
1. Consumer-driven contract testing
2. Decoupling deployment from release
3. Generated infrastructure diagrams
4. NoPSD
5. Productos en vez de proyectos
6. Threat Modelling

ENSAYAR
7. BEM
8. BFF - Backend for frontends
9. Docker for builds
10. Event Storming
11. Flux
12. Idempotency filter
13. iFrames for sandboxing
14. NPM for all the things
15. Offline first web applications
16. Phoenix Environments
17. QA in production

EVALUAR
18. Accumulate-only data
19. Bug bounties
20. Data Lake
21. Hosted IDE's
22. Monitoring of invariants
23. Reactive Architectures

RESISTIR
24. Gitflow
25. High performance envy/web scale envy
26. Microservice envy
27. Pace-layered Application Strategy
28. Programming in your CI/CD tool
29. SAFe™
30. Separate DevOps team

TÉCNICAS *continúa*

de amenazas potenciales, desde el crimen organizado y el espionaje gubernamental hasta adolescentes que atacan sistemas solo por diversión ("for the lulz"). **Los Modelos de amenazas** presentan un conjunto de técnicas, en su mayoría desde una perspectiva defensiva, que ayudarán a comprender y clasificar las amenazas potenciales. Transformados en "historias de usuarios malvados", los modelos de amenazas pueden proveer a un equipo un enfoque manejable y eficaz para lograr que sus sistemas sean más seguros.

Depurando problemas de CSS puede ser doloroso. ¿Cuántas veces le ha tocado revisar detenidamente miles de estilos invalidados hasta encontrar la fuente de su problema? Esto ha hecho que muchos de nuestros equipos introduzcan varias guías de cómo evitar estilos de cascada e invalidaciones, haciendo que los estilos sean optativos y enfatizando el nombramiento razonado. **BEM** es una regla sencilla de nombramiento para CSS (BEM es la sigla de Bloque, Elemento, Modificador) que ayuda a presentar una claridad semántica y estructura a su CSS. Al utilizar BEM, se vuelve más fácil comprender las reglas de CSS que están influyendo en la apariencia de un elemento y, lo que es más importante, la intención de tales reglas. Este enfoque puede percibirse como trasladar al mundo de CSS la lección de orientación a objetos que favorece la composición en vez de la herencia.

Los servicios valiosos soportan una gran variedad de clientes, tales como los dispositivos móviles y diversas formas de interfaz web. Resulta tentador diseñar un API único de back-end para dar soporte a todos los clientes con un API reutilizable. Pero las necesidades de clientes varían, al igual que las limitaciones tales como el ancho de banda para dispositivos móviles en relación con el deseo de tener muchos datos en conexiones rápidas de internet. Por tanto, con frecuencia, lo recomendable es **definir diferentes servicios de back end para cada tipo de cliente de front-end**. Estos sistemas de back end deben ser desarrollados por equipos alineados con cada front end, con el fin de garantizar que cada sistema cumpla posteriormente con las necesidades de su cliente.

Uno de los muchos usos innovadores de un **Contenedor Docker** que hemos visto en nuestros proyectos es una técnica para manejar las dependencias en tiempo de construcción (build time). En el pasado, fue común ejecutar agentes de construcción en un sistema operativo, agregando las dependencias requeridas para la versión objetivo. Pero con contenedores Docker es posible ejecutar el paso de compilación en un ambiente aislado junto con las dependencias sin contaminar el agente de construcción. Esta técnica de utilización del **contenedor Docker para construcción** ha resultado particularmente útil al compilar binarios de Golang, y el contenedor de construcción Golang está disponible para este propósito específico.

La lluvia de eventos es una manera útil de crear modelos de dominio rápidamente desde fuera hacia adentro (outside-in). Se comienza con los eventos que ocurren en el dominio en vez de tener un modelo de datos estático. Al ejecutarse como un taller facilitado, se enfoca en descubrir eventos clave de dominio, colocarlos en una línea de tiempo, identificar sus activadores y luego explorar sus relaciones. Este método es bastante útil para personas que toman un enfoque CQRS o de Origen de Eventos. Es importante lograr que las personas adecuadas se reúnan en una sala: una combinación de personas de negocios y técnicas, que aporten preguntas y respuestas. La segunda clave del éxito es garantizar que haya suficiente espacio de pared para los modelos. Observe para descubrir el panorama global, con la meta de comprender colectivamente el dominio en toda su complejidad, antes de comenzar a buscar soluciones.

Flux es una arquitectura de aplicaciones introducida por Facebook. Se menciona, generalmente, junto con **React.js**, y Flux se basa en un flujo de datos de una vía por el proceso de graficación (renderización). Flux aprovecha el entorno moderno de la web con aplicaciones de JavaScript del lado del cliente, que evita los venerables clichés de MV*. Los equipos de ThoughtWorks están ahora intentando obtener algo de experiencia con este estilo de arquitectura y se dieron cuenta que combina bien con la orientación a servicios y resuelven algunos de los problemas inherentes a los enlaces de datos de doble vía.

Muchos servicios, especialmente los de sistemas legados, se escribieron con la suposición de que cualquier pedido ocurrirá una sola vez. Debido a las características de las redes, suele ser difícil lograr esto. Un **filtro idempotente** es un componente sencillo que solamente revisa que no haya pedidos duplicados y garantiza que se envíen al servicio de suministro una sola vez. Tal filtro solo debe realizar esta tarea y puede utilizarse como un decorador sobre llamadas de servicio existentes.

TÉCNICAS *continúa*

Las páginas web modernas tienden a tener una gran gama de componentes y fragmentos de JavaScript que vienen de una variedad de repositorios de código propiedad de terceras personas. Esto podría tener un impacto negativo tanto en la seguridad como en el rendimiento. Mientras seguimos esperando por un aislamiento más completo de JavaScript con componentes web, nuestros equipos se han beneficiado del uso en HTML5 de los **iFrames para zona de pruebas (sandboxing)** de JavaScript no confiable.

El mundo de JavaScript tiene una gran gama de herramientas de dependencia y manejo de paquetes, todos dependen de Node Package Manager (NPM) (Administrador de Paquetes de Nodos). Los equipos comienzan a percibir estas herramientas adicionales como redundantes y ahora recomiendan que se utilice solo NPM para manejo de paquetes y de dependencias, si es que es posible. La simplificación de utilizar **NPM para todo** ayuda a reducir algunas de las acumulaciones del espacio de herramientas de JavaScript.

El tiempo que toma en proveer y actualizar ambientes continúa siendo un cuello de botella significativo en muchos proyectos de software. Los ambientes Phoenix pueden ayudar a resolver esta demora al ampliar la idea de los Servidores Phoenix para cubrir ambientes completos. Consideramos que esta es una técnica valiosa y el ahorro de tiempo que usted debería tomar en cuenta, al probar este enfoque. Mediante la automatización, creamos ambientes completos con configuraciones de redes, balanceo de cargas y puertos firewall, por ejemplo al utilizar **CloudFormation** de AWS. Luego podemos demostrar que funciona el proceso al derribar los ambientes y volver a crearlos desde cero, de forma periódica. **Los ambientes Phoenix** pueden respaldar el desarrollo de ambientes nuevos para pruebas, programación, UAT (User Acceptance Testing - Pruebas de aceptación de usuario) y recuperación de desastres. Al igual que con servidores Phoenix, este patrón no es aplicable siempre y debemos pensar detenidamente acerca de temas como estado y dependencias. Si tratamos todo el ambiente como una implantación azul/verde, este puede ser un enfoque útil cuando se deba realizar una reconfiguración del ambiente.

Tradicionalmente, las funciones de aseguramiento de la calidad (QA) se han enfocado en evaluar la calidad de un producto de software en un ambiente de pre-producción. Con el auge de la "Entrega Continua" (Continuous Delivery), la función de QA se está transformando para incluir el análisis de la calidad del producto de software en un ambiente de producción. Esto incluye el monitoreo de los sistemas de producción ya que despliegan condiciones de alertas al detectar errores urgentes, determinar problemas existentes de calidad y cómo establecer mediciones que se puedan utilizar en el ambiente de producción para que esto funcione. Si bien hay un peligro de que algunas organizaciones vayan demasiado lejos y hagan caso omiso de QA de pre-producción, nuestra experiencia demuestra que **QA en un ambiente de producción** es una herramienta valiosa para empresas que ya han avanzado a un grado razonable de Entrega Continua.

Se están volviendo más populares las estructuras de datos inmutables, con lenguajes funcionales como Clojure y Scala que están proveyendo inmutabilidad de forma predeterminada. La inmutabilidad facilita el desarrollo de código, su lectura y análisis. Al utilizar un **almacén de datos solo acumulables** se pueden lograr algunos de estos beneficios en la capa de base de datos, y se hacen más sencillas las auditorías y consultas históricas. Las opciones de implementación pueden variar desde almacenes de datos específicos y acumulativos, tales como Datomic a un enfoque más simple de "agregar pero no actualizar" de una base de datos tradicional. **Solo acumular** es una estrategia de diseño en la que los datos se retiran mediante retracción en vez de actualizaciones. **Solo agregar al fin** es una técnica de implementación.

Cada vez más organizaciones están comenzando a utilizar **bug bounties** (premio por errores) para alentar los informes de errores, que se relacionan con frecuencia con seguridad, y, en general, que ayudan a mejorar la calidad de software. Para ayudar a estos programas, las empresas tales como HackerOne u BugCrowd pueden ayudar a organizaciones a manejar más fácilmente este proceso. Tenemos experiencia limitada con estas ofertas de nuestra parte, pero nos gusta la idea de alentar a las personas a hacerse escuchar, de forma abierta y transparente, y resaltar lo que, con frecuencia, pueden ser vulnerabilidades perjudiciales. Vale la pena indicar que puede haber algunos asuntos legales en alentar a los usuarios a encontrar partes vulnerables de software, por lo que deben verificar esto antes.

Un **DataLake (Lago de datos)** es un almacén de datos inmutables principalmente con datos no procesados, que actúan como una fuente para analítica de datos. Si bien los Data Warehouse (depósito de datos) filtran

TÉCNICAS *continúa*

y procesa los datos antes de almacenarlos, el lago solo capta datos sin procesar, y deja que los usuarios de estos datos se encarguen de llevar a cabo análisis específicos, según requieran. Entre los ejemplos, podemos mencionar HDFS o HBase en un framework de procesamiento Hadoop, Spark o Storm. Generalmente, solo un grupo pequeño de analistas de datos trabajan con datos no procesados, para desarrollar secuencias de datos procesados de un data marts, que la mayoría de usuarios puede consultar. Los Data Lake deben utilizarse solo para analítica y reportes. Para colaborar entre sistemas operacionales, preferimos aplicar servicios diseñados para tal propósito.

Muchas organizaciones desean aprovechar el desarrollo distribuido o en otros localidades remotas, pero tienen preocupaciones de seguridad por temas de código y propiedad intelectual, que está fuera de su control. El resultado, con frecuencia, es utilizar soluciones de escritorio remoto con mucho tiempo de latencia, para el desarrollo, con el fin de cumplir con controles de seguridad organizacionales, pero que pueden influir negativamente en la productividad de los desarrolladores. Una alternativa es utilizar un **IDE de servidor** al que se accede por un navegador mediante conexión VPN. El IDE, el código y el ambiente para generar versiones se albergan en una nube privada de la organización, para disminuir las preocupaciones de seguridad y así mejorar significativamente la experiencia del desarrollador. Entre las herramientas para este propósito, se incluyen Orion y Che de Eclipse Foundation, Cloud9 y Code Envy.

En el monitoreo, el enfoque común es imaginarse condiciones erróneas y establecer alertas para cuando estas aparezcan. Pero, con frecuencia, es difícil enumerar los infinitos modos de falla de un sistema de software. **Monitoreo de partes invariables** es un enfoque complementario de establecer rangos normales esperados, al revisar el comportamiento histórico frecuente, y alertar cuando un sistema se sale de tales límites.

Creemos firmemente que las bifurcaciones de control de versión, que han existido por mucho tiempo, perjudican las prácticas de ingeniería tales como integración continua, y esta creencia es subyacente a nuestro desagrado por **Gitflow**. Nos encanta la flexibilidad de Git que está subyacente, pero detestamos las herramientas que alientan las malas prácticas de ingeniería. Las bifurcaciones de vida corta son menos perjudiciales, pero casi todos los equipos que vemos que usan Gitflow se sienten con el derecho de usar erróneamente este flujo de trabajo que depende mucho de bifurcaciones. Esto alienta la integración tardía (y desalienta, por tanto, la integración continua real).

Vemos que muchos equipos se enfrentan a problemas ya que han escogido herramientas, frameworks o arquitecturas complejas, porque piensan que el alcance de sus proyectos podría crecer. Compañías tales como Twitter y Netflix tienen que soportar cargas extremas y requieren necesariamente estas arquitecturas, pero también cuentan con equipos de desarrollo increíblemente talentosos que pueden manejar esta complejidad. La mayoría de situaciones no requiere estas hazañas de ingeniería. Los equipos deben tener bajo control su **envidia para escalar la web** a favor de soluciones más sencillas que son igual de exitosas.

La propuesta de la **Estrategia de aplicación "Pace-layered"** (por capas según el ritmo de avance) de Gartner parece estar creando un enfoque poco útil de la idea de capas en una arquitectura. Encontramos más útil pensar en el ritmo de cambio en diferentes **capacidades de negocio** (que pueden estar compuestas de varias capas de arquitectura). El peligro de enfocarse en capas es que muchos tipos de cambios pueden corresponder a varias capas. Por ejemplo, poder agregar una nueva clase de existencias a un sitio web no consiste solamente en tener un CMS fácil de modificar. También se requiere actualizar la base de datos, puntos de integración, sistemas de depósitos de datos, etc. Es útil reconocer que algunas partes de una arquitectura deben ser más manipulables que otras. Sin embargo, el enfoque en capas no ha resultado beneficioso.

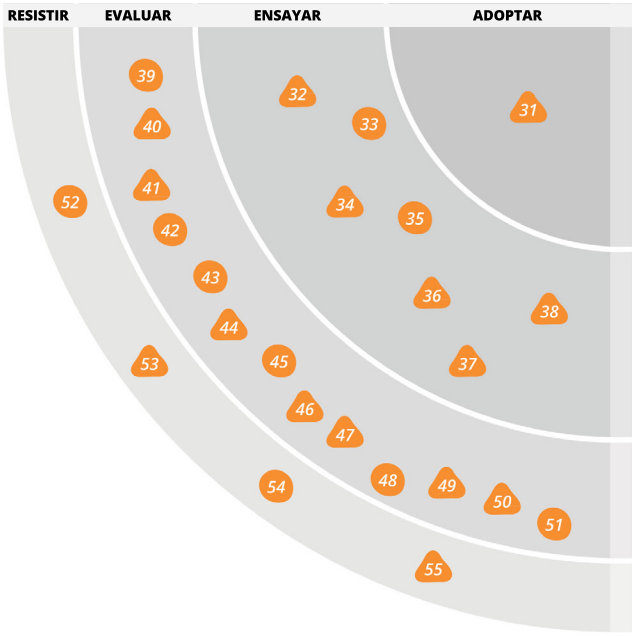
El **Scaled Agile Framework®** (es decir **SAFe™**) (Marco ágil ampliable) es considerado cada vez más por muchas organizaciones en crecimiento. Además, las herramientas y la certificación son aspectos importantes de la adopción de SAFe™. Continuamos preocupados por el hecho de que las verdaderas adopciones pueden ser objeto de una estandarización excesiva y la tendencia es enfocarse en prácticas de difusión a gran escala, lo que puede afectar la adopción ágil. En vez de esto, recomendamos enfoques livianos que incluyen la experimentación y la inclusión de prácticas de mejoras continuas como Improvement Katas que ofrece a las organizaciones un mejor modelo para ampliar proyectos ágiles.

Scaled Agile Framework® y SAFe™ son marcas registradas de Scaled Agile, Inc.º

PLATAFORMAS

La seguridad de contraseñas sigue siendo un tema de debate agitado. Y el gobierno del Reino Unido aconseja controles técnicos que permita a los usuarios recordar contraseñas más sencillas. El consejo de contraseñas de Edward Snowden se describe como “en el límite de lo seguro”. Las contraseñas son generalmente los eslabones más débiles de la cadena de seguridad, por lo que recomendamos utilizar **autenticación de doble factor**, que puede mejorar la seguridad de manera significativa. Contraseña de una vez basada en el tiempo (**TOTP**) es el algoritmo estándar en este espacio, con implementaciones sencillas en el servidor y aplicaciones de autenticación gratuitas para teléfonos inteligentes de Google y Microsoft.

Mesos es una plataforma que no toma en cuenta los recursos de computación subyacentes, para faciliar el desarrollo de sistemas distribuidos ampliables de forma masiva. Se puede utilizar para proveer una capa de programación para un contenedor Docker, o para que actúe como una capa de abstracción en ambientes de AWS. Twitter lo ha utilizado con valiosos resultados con el fin de ayudar a ampliar su infraestructura. Están comenzando a aparecer herramientas desarrolladas sobre la base de Mesos, tales como Chronos, que es un reemplazo distribuido de “cron”, con tolerancia a fallas. Están divulgándose historias notables de éxito, tales como el Siri de Apple, cuya arquitectura fue modificada con el fin de utilizar Mesos.



AWS introduce una gran cantidad de características nuevas casi mensualmente, por lo que puede a veces ser difícil que cualquier oferta nueva de servicios sobresalga, sin embargo **Lambda** ha logrado despertar interés. Lambda inicialmente solo soportaba aplicaciones JavaScript, y ahora funciona con aplicaciones basadas en JVM (seguramente otros tipos serán soportados en el futuro). Lambda ahora permite lanzar procesos de duración muy corta, sea como reacción a un evento o mediante una llamada de un API Gateway (interfaz de comunicaciones) relacionado. Para servicios sin estado esto significa que ya no se necesita preocuparse por ejecutar máquinas de larga duración, lo que posiblemente reducirá costos y mejorará la seguridad a largo plazo. A pesar de que existen otras incursiones en el espacio de PaaS por parte de AWS, Lambda parece ser la mejor opción hasta ahora.

Fastly, uno de los tantos CDN del mercado, tiene una cantidad grande y creciente de fanáticos en los proyectos de ThoughtWorks, y se usa en productos muy comunes de escala de web, como GitHub y Twitter. Sus características, velocidad y precios se combinan para convertirlo en una opción muy atractiva cuando se necesita una solución de caché de vanguardia. También hemos notado ahorros importantes de costos en proyectos que utilizan esta plataforma luego de haber abandonado otro CDN. Si estás buscando un CDN, no es mala idea investigar este.

ADOPTAR
31. TOTP Two-Factor Authentication

ENSAYAR
32. Apache Mesos
33. Apache Spark
34. AWS Lambda
35. Cloudera Impala
36. Fastly
37. H2O
38. HSTS

EVALUAR
39. Apache Kylin
40. AWS ECS
41. Ceph
42. CoreCLR and CoreFX
43. Deis
44. Kubernetes
45. Linux security modules
46. Mesosphere DCOS
47. Microsoft Nano Server
48. Particle Photon/Particle Electron
49. Presto
50. Rancher
51. Time series databases

RESISTIR
52. Application Servers
53. Over-ambitious API Gateways
54. SPDY
55. Superficial private cloud

PLATAFORMAS *continúa*

El análisis predictivo se utiliza cada vez en más productos, con frecuencia, en la funcionalidad orientada al usuario final. **H2O** es un paquete interesante de código abierto (proveniente de una empresa nueva), que permite acceder al análisis predictivo a los equipos de desarrollo. H2O brinda una forma sencilla y clara para usar múltiples elementos analíticos, con gran rendimiento e integración sencilla con plataformas basadas en la JVM. También se integra con las herramientas favoritas de los científicos de datos, R y Python, y con Hadoop y Spark.

HSTS HTTP Strict Transport Security (Seguridad Estricta de Transporte HTTP) es una política ahora ampliamente soportada, que permite que los sitios web se protejan de ataques de degradación. En el contexto de HTTPS, un ataque de degradación es el que puede hacer que los usuarios de un sitio cambien a HTTP, en vez de seguir en HTTPS, lo que los hace vulnerables a ataques como "man-in-the-middle" (suplantación del servidor). Utilizando una cabecera de servidor, se informa a los navegadores que solamente pueden utilizar HTTPS para acceder a su sitio web y que no deben hacer caso a intentos de degradación para comunicarse con el sitio mediante HTTP. El soporte de los navegadores es ahora lo suficientemente amplio para que esta característica de fácil implementación se considere para cualquier sitio que utiliza HTTPS.

El **Elastic Container Service (ECS)** es el componente de AWS con el que Amazon ingresa en el espacio de contenedores Docker soportados por múltiples servidores. A pesar que hay mucha competencia en esta área, no hay gran disponibilidad de soluciones administradas en instalaciones externas. Por el momento, aunque ECS parece ser un buen primer paso, nos preocupa el hecho de que es demasiado complicado y no cuenta con una capa de abstracción robusta. Sin embargo, si se quiere que un contenedor Docker funcione en AWS, esta herramienta definitivamente debería estar al inicio de su lista. Solamente no espere que sea fácil empezar a utilizarla.

Ceph es una plataforma de almacenamiento que puede ser utilizada como almacen de objetos, de bloques, y como sistema de archivos, que normalmente corre en un grupo (cluster) de servidores. Debido a que su primer lanzamiento importante fue en julio de 2012, Ceph ciertamente no es un producto nuevo. Sin embargo, queremos destacarlo en este Technology Radar como un bloque importante para construir nubes privadas. Es particularmente atractivo porque su componente RADOS Gateway puede exponer el almacén de objetos por medio de una interfaz RESTful que es compatible con Amazon S3 y las APIs Swift de OpenStack.

Kubernetes es la respuesta de Google al problema de distribuir contenedores a un grupo de servidores, que se está convirtiendo en un escenario cada vez más común. No es la solución utilizada por Google internamente, pero es un proyecto de código abierto que se inició en Google y ha recibido una cantidad razonable de contribuciones externas. Se admiten como formatos de contenedores Docker y Rocket y los servicios ofrecidos incluyen gestión de salud, replicación y descubrimiento. Una solución similar en este espacio es Rancher.

Mesosphere DCOS es una plataforma construida sobre Mesos. Provee un tipo de abstracción en relación con servidores subyacentes, lo que le brinda un conjunto de almacenamiento y recursos de computación, para que los servicios creados para DCOS operen a una escala masiva. (Ya se pueden utilizar Hadoop, Spark y Cassandra, entre otros.) Esto es posiblemente una exageración para las cargas de trabajo actuales más modestas (en cuyo caso el anterior Mesos sencillo sería lo adecuado). Pero será interesante revisar si Mesosphere comienza a intentar posicionar DCOS como un sistema de propósito general.

En contraste con la nube moderna y las soluciones de contenedor basadas en Linux, incluso Windows Server Core es muy grande y poco manejable. Microsoft está reaccionando y proveyó las primeras vistas previas de **Nano Server**, una versión mucho más simplificada de Windows Server que desecha algunos componentes tales como la interfaz gráfica, soporte para Win32 de 32 bits, inicio de sesión local y soporte de escritorio remoto. El resultado es una ocupación de archivos en disco de cerca de 400MB. Las vistas previas no facilitan el trabajo y la solución final se restringirá al uso de CoreCLR, pero para compañías interesadas en soluciones basadas en .NET, Nano Server definitivamente merece ser revisado, en esta etapa.

Presto es un motor de consultas distribuido de SQL, que fue diseñado y optimizado para ejecutar cargas de trabajo de analítica interactiva. La arquitectura de procesamiento paralelo masivo de Presto, junto con las técnicas avanzadas de generación de código y conductos de procesamiento en memoria, hacen que Presto sea altamente escalable. Admite el uso de un subconjunto

PLATAFORMAS *continúa*

grande de ANSI SQL incluyendo consultas complejas, "join" (uniones), agregaciones y funciones de ventanas (window functions). Presto puede utilizarse junto con una amplia gama de fuentes de datos, incluyendo **Hive**,**Cassandra**,**MySQL** y **PostgreSQL**, para así unificar la interfaz de analítica interactiva en fuentes de datos de una organización. Las aplicaciones pueden conectarse con Presto mediante su interfaz JDBC.

Rancher es una solución de código abierto que permite el despliegue de contenedores en un clúster de servidores (lo que se está convirtiendo en un escenario cada vez más común). Provee servicios tales como gestión de ciclo de vida, monitoreo, health checks y detección. También incluye un sistema operativo en contenedores basado en Docker. El amplio enfoque en contenedores y su pequeña huella son ventajas clave de Rancher. Una solución similar en este contexto es Kubernetes.

Una de nuestras quejas comunes es el traslado de la inteligencia de negocios a middleware (software intermedio), que resulta en servidores de aplicaciones e interconexiones de servicios empresariales con ambiciones de ejecutar lógica crítica de aplicaciones. Estos requieren programación compleja en ambientes no idóneos para tal propósito. Estamos viendo un resurgimiento preocupante de esta enfermedad en los **Gateways de API demasiado ambiciosos**. Los Gateways de API pueden ser útiles para manejar algunos asuntos genéricos, tales como autenticación y límites de tasas, pero los aspectos inteligentes de dominio como la transformación de datos o el procesamiento de reglas de negocio deben residir en aplicaciones o servicios en donde pueden controlarse por equipos de producto que trabajen estrechamente en los dominios a los que brindan servicios.

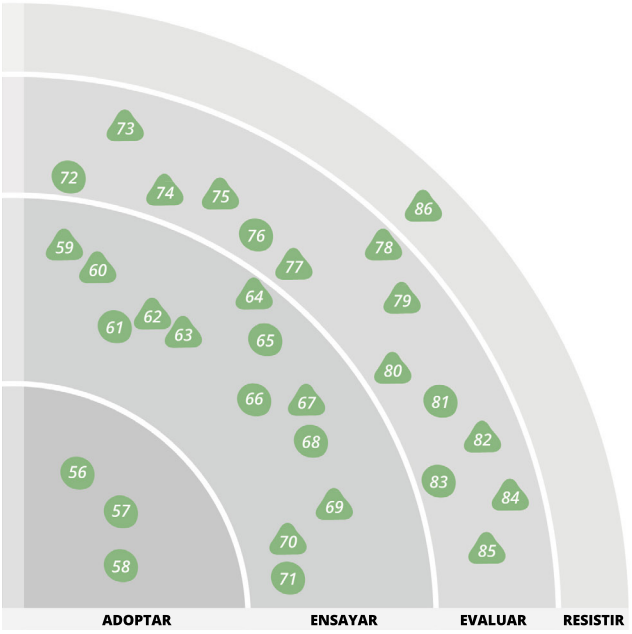
Hemos visto los indiscutibles aumentos de productividad que surgen por la implantación de aplicaciones y servicios en proveedores de nube maduros. Gran parte de esos aumentos provienen de la capacidad de los equipos de implantar y operar sus propios servicios con un alto grado de autonomía y responsabilidad. Ahora vemos con frecuencia en las organizaciones ofertas de **Superficial Private Cloud** (Nube privada superficial) en donde se coloca una etiqueta de "nube" a las plataformas básicas de virtualización. Frecuentemente, los equipos pueden autoproveerse un conjunto restringido de tipos de servicios con acceso limitado y poca capacidad de personalización de "diseños empresariales" gobernados centralmente, que generan soluciones poco elegantes. El avance de la implantación está generalmente limitada por la infraestructura provista manualmente, tal como la red, el cortafuegos (firewall) y el almacenamiento. Alentamos a las organizaciones a considerar mejor los costos de solicitar el uso de una opción inadecuada de nube privada.

HERRAMIENTAS

Hemos tenido comentarios increíblemente positivos de una gran cantidad de equipos de ThoughtWorks que han utilizado **Browsersync**. Conforme crece la cantidad de dispositivos que acceden a nuestras aplicaciones web, también aumenta el esfuerzo requerido para realizar pruebas en todo estos diferentes dispositivos. Browsersync es una herramienta gratuita y de código abierto, que puede disminuir drásticamente este esfuerzo al sincronizar las pruebas manuales de navegadores a lo largo de múltiples navegadores móviles o de escritorio. Al proveer opciones de CLI e interfaz de usuario, la herramienta es compatible con la generación de versiones (build-pipelines) y automatiza tareas repetitivas como por ejemplo, el llenado de formatos.

El manejo de dependencias en proyectos de iOS y OS X solía ser completamente manual o totalmente automático mediante la utilización de CocoaPods. Gracias a **Carthage**, tenemos a disposición un punto intermedio. Carthage gestiona dependencias, es decir, descarga, crea y actualiza frameworks, pero delega al proyecto la integración de estos frameworks en la construcción del mismo, a diferencia CocoaPods, que básicamente asume el control de la estructura del proyecto y la configuración de las versiones. Se debe tomar en cuenta que Carthage solo puede manejar estructuras dinámicas, que no están disponibles en iOS 7 y versiones anteriores.

Previamente, recomendamos boot2docker como una manera de ejecutar Docker en su máquina local Windows u OS X. **Docker Toolbox** ahora reemplaza a boot2docker con herramientas adicionales. Ahora se incluye Kitematic para manejar contenedores y también Docker Compose para la configuración de varios Docker (solo para Mac). Se puede usar de forma segura como un reemplazo de inclusión fácil (drop-in) para boot2docker, y hasta puede gestionar las actualizaciones por usted.



Guardar secretos, como contraseñas y tokens de acceso, de manera segura en repositorios de código, es posible gracias varias herramientas. Por ejemplo: Blackbox, que mencionamos en el anterior Technology Radar. A pesar de la existencia de estas herramientas, es todavía muy común almacenar secretos de forma no protegida. De hecho, es muy común el uso de software para la explotación automatizada de vulnerabilidades que permite encontrar credenciales AWS y generar instancias de EC2 para minar Bitcoins. Así el atacante se apodera de los Bitcoins y el dueño de la cuenta solo paga la factura.

ADOPTAR
56. Composer
57. Mountebank
58. Postman

ENSAYAR
59. Browsersync
60. Carthage
61. Consul
62. Docker Toolbox
63. Gitrob
64. GitUp
65. Hamms
66. IndexedDB
67. Polly
68. REST-assured
69. Sensu
70. SysDig
71. ZAP

EVALUAR
72. Apache Kafka
73. Concourse CI
74. Espresso
75. Gauge
76. Gor
77. ievms
78. Let's Encrypt
79. Pageify
80. Prometheus
81. Quick
82. RAML
83. Security Monkey
84. Sleepy Puppy
85. Visual Studio Code

RESISTIR
86. Citrix for development

HERRAMIENTAS *continúa*

Gitrob tiene un enfoque similar y escanea los repositorios de GitHub de la empresa, marcando todos los archivos que podrían tener información sensible y que no deberían colocarse en el repositorio. Este método es obviamente reactivo. Gitrob solo puede alertar a los equipos cuando es (casi) demasiado tarde. Por este motivo, Gitrob puede ser únicamente una herramienta complementaria, que minimiza el daño.

Git puede parecer confuso. Realmente confuso. E incluso cuando se utiliza en un proceso sencillo de desarrollo basado en trunk, todavía hay bastantes sutilezas de su funcionamiento, que pueden hacer que las personas se confundan de vez en cuando. Cuando esto ocurre, es muy útil entender el funcionamiento de Git en sus aspectos básicos, y **GitUp**, una herramienta de Mac, le brinda justo esto. GitUp provee una representación gráfica de lo que ocurre cuando usted introduce comandos normales de Git en el terminal. Puede aprender varios comandos de Git y a la vez entender lo que cada uno hace mientras los utiliza. GitUp es una herramienta muy útil para novatos en Git y también para aquellos con más experiencia en Git.

Varios de nuestros equipos que trabajan en proyectos .NET han recomendado **Polly** porque indican que es útil para desarrollar sistemas basados en microservicios. Sirve para alentar la expresión fluida de políticas de manejo de excepciones transitorias y el patrón de Interruptores (Circuit Breaker) incluyendo políticas como Intentar, Intentar para Siempre, y Esperar e Intentar. Ya existen en otros lenguajes librerías similares (Hystrix para Java por ejemplo), y Polly es un añadido bienvenido para la comunidad de .NET. **Brighter** se integra bien con Polly. Brighter es otra librería pequeña de código abierto para .Net que provee scaffolding para implementar la invocación de comandos. La combinación de las dos librerías brinda funcionalidad útil de corto-circuito especialmente en el contexto del patrón de Puertos y Adaptadores y CQRS. A pesar de que pueden utilizarse por separado, en algunos ambientes, los equipos piensan que funcionan bien juntos.

Muchas herramientas de monitoreo se crean alrededor del concepto de la máquina o instancia. El creciente uso de patrones como el Phoenix Server y herramientas como Docker significa que este es un método cada vez más inútil de modelar la infraestructura: Las instancias se convierten en transitorias mientras los servicios perduran. **Sensu** permite que una instancia se registre a sí misma como una que desempeña una función específica, y Sensu luego la monitorea basándose en esta premisa. A la larga, distintas instancias que desempeñan tal función pueden aparecer y desaparecer. Con base en estos factores y la madurez creciente de la herramienta, pensamos que ya era hora de que Sensu esté nuevamente en el radar.

A pesar de que **SysDig** no es la herramienta más nueva en el Technology Radar, todavía estamos sorprendidos de la cantidad de gente que no ha escuchado de esta. CLI es una herramienta Open Source, que se puede incorporar, para resolver problemas en el sistema Linux, SysDig tiene características bastante potentes. Una de las cosas clave que nos gusta es la capacidad de generar un rastreo del sistema en una máquina que experimenta problemas, para que luego se plantee preguntas y descubrir lo que está pasando. SysDig también provee soporte para trabajar con contenedores, lo que la convierte en una herramienta todavía más potente.

Muchos equipos de desarrollo están reemplazando simples servidores de integración continua a procesos de Entrega Continua, que a veces abarcan varios ambientes, incluyendo el de producción. Para implementarlos exitosamente y operarlos de forma sostenible, se requiere una herramienta CI/CD que trate a los procesos (pipelines) de generación de versiones y artefactos como ciudadanos de primera clase; desafortunadamente, no hay muchos. **Concourse CI** es nuevo en este campo y luce muy prometedor. Nuestros equipos que lo han probado están emocionados sobre su configuración, porque permite versiones que se ejecutan en contenedores, además tiene una interfaz útil y limpia, lo que desalienta el uso con servidores con versiones snowflake (copos de nieve).

Espresso es una herramienta de pruebas funcionales de Android. Su API de core pequeño esconde los desordenados detalles de implementación y ayuda a escribir pruebas concisas, con una ejecución de pruebas más rápida y confiable.

Gauge es una herramienta liviana de automatización de pruebas que sirve para varias plataformas. Las especificaciones están escritas de forma libre en Markdown, para que se pudieran desarrollar casos de prueba en lenguaje de negocios y se pueda incorporar en cualquier formato existente de documentación.

HERRAMIENTAS *continúa*

Los lenguajes admitidos se implementan como componentes (plug-in) para una implementación de core único, lo que garantiza la consistencia en varias implementaciones de lenguajes. Esta herramienta, es Open Source de ThoughtWorks, también permite la ejecución paralela, sin ninguna personalización adicional, para todas las plataformas soportadas.

Aunque el uso de Internet Explorer está disminuyendo, para muchos productos la base de usuarios de IE no es un porcentaje sin importancia en el mercado y la compatibilidad con el navegador debe probarse. Este es un problema si prefiere la facilidad de utilizar sistemas basados en UNIX para el desarrollo. Para intentar resolver este dilema, **ievms** provee un script (programa) utilitario que combina imágenes de VM distribuidas de Windows con VirtualBox para automatizar la configuración y realizar pruebas de varias versiones de IE, desde 6 hasta Edge.

Aunque más sitios cada día, están implementando HTTPS con el fin de ayudar a proteger sus propios usuarios y mejorar la integridad de la web como un todo, hay muchos más sitios para visitar. Además, vemos que hay un creciente número de personas que utilizan HTTPS en sus empresas, para proveer garantías adicionales de seguridad. Uno de los principales obstáculos a la adopción más amplia ha sido el proceso de obtener un certificado, para iniciar tal adopción. Aparte del costo, el proceso no es nada amigable. **Let's Encrypt**, una nueva Autoridad de Certificado, intenta resolver todo esto. En primer lugar, provee certificados gratuitamente. En segundo lugar, y lo que puede parecer más importante, también provee un API de línea de comando que es muy fácil de usar, lo que facilita la automatización completa del proceso de emisión, actualización e instalación de certificados. Pensamos que Let's Encrypt, en versión beta por el momento, tiene la oportunidad de ser revolucionario para que más sitios web cuenten con HTTPS, y a la vez puede demostrar cómo se pueden crear buenas herramientas automatizables para los preocupados por la seguridad.

Pageify es una librería escrita en Ruby para crear objetos de páginas para pruebas de automatización de interfaz de usuario, que se enfoca en la ejecución más rápida de las pruebas y mayor facilidad en la lectura del código. Ofrece APIs sencillos para definir, operar y asegurar dinámicamente objetos de páginas, lo que permite obtener código legible, al manipular elementos con jerarquías complejas en el DOM. Provee integración para **WebDriver** y **Capybara**.

SoundCloud ha liberado, recientemente, a código libre su conjunto de herramientas de monitoreo y alertas, **Prometheus**. Estas fueron desarrolladas como reacción a las dificultades con Graphite en sus sistemas de producción. Prometheus soporta principalmente un modelo basado en extracción (pull) vía HTTP de los datos (a pesar de que también se puede usar un modelo de push más similar a Graphite). También va un paso más allá al incluir alertas, convirtiéndolas en una parte activa de tu conjunto de herramientas operativa. A la fecha de este escrito, Prometheus está todavía solo en la versión 0.15.1 aunque evoluciona rápidamente. Estamos contentos en ver el enfoque reciente de productos en bases de datos de series de tiempo básicas y capacidades de indexado multidimensionales, que permiten exportar a una variedad más amplia de herramientas gráficas de visualización (front-end).

Debido al reciente crecimiento de una gama de servicios que proveen APIs RESTful, se está volviendo cada vez más importante el proceso asociado a su documentación. Hemos mencionado previamente a Swagger, y en este Technology Radar deseamos resaltar el lenguaje de modelamiento de API RESTful llamado (**RAML**). Nuestros equipos sienten que, en comparación con Swagger, este es más liviano y el enfoque cambia de agregar documentación a APIs ya existentes al proceso de diseño de APIs.

Sleepy Puppy es un framework de manejo de carga útil de cross-site scripting (XSS) demorada, que Netflix divulgó recientemente como código abierto. XSS es una forma de ataque usada en páginas web. Sleepy Puppy permite realizar pruebas de vulnerabilidades XSS una vez vulnerada la aplicación que era el blanco principal y el criminal desee atacar un sistema subyacente secundario. Siendo XSS uno de los ataques incluidos en la lista de los 10 principales de OWASP, vemos que esta plataforma pueda ayudar en la automatización de los chequeos de seguridad para varias aplicaciones. Esta simplifica la captura, gestión y rastreo de la propagación de XSS en períodos largos de tiempo, con cargas útiles personalizables. Sleepy puppy también expone una API que puede integrarse a herramientas de chequeo de vulnerabilidades tales como ZAP, que sirve para verificar automáticamente problemas de seguridad.

Visual Studio Code es un editor de IDE gratuito de Microsoft, que está disponible para múltiples plataformas. Consideramos que la integración con la herramienta de control de versiones Git es muy ventajosa para promover prácticas de integración continua.

HERRAMIENTAS *continúa*

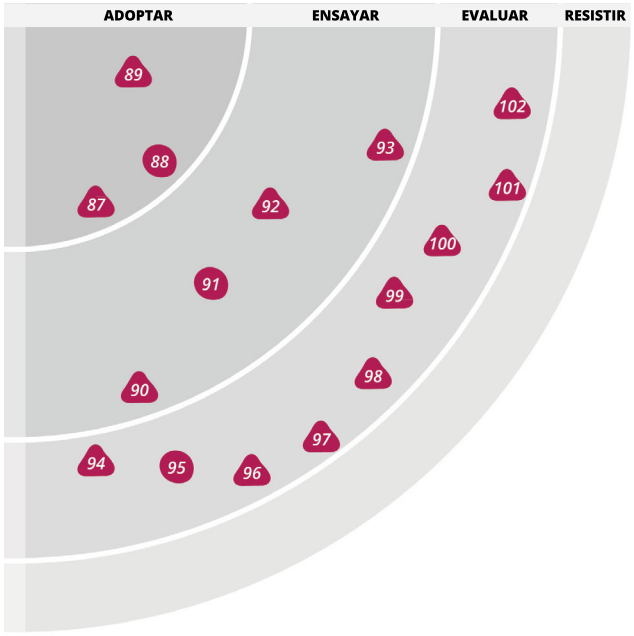
Visual Studio Code también provee una manera de integrar con herramientas externas mediante tareas, con la detección de tareas de grunt/gulp que eliminan la necesidad de ejecutarlas mediante comandos en la terminal al poderse ejecutar directamente desde el propio editor. (Grunt.js y Gulp.js) Debido al crecimiento del ecosistema Docker, este IDE ofrece soporte para el dockerfile con autocompletamiento de bloques de código y definiciones de comandos válidos.

Muchas organizaciones todavía obligan a los equipos de desarrollo, distribuidos o de fuera del país, a utilizar **el escritorio remoto de Citrix para el desarrollo**.

A pesar de que este provee un modelo sencillo de seguridad, ya que los componentes no salen nunca de los servidores de la organización, el uso de escritorios remotos para el desarrollo definitivamente paraliza la productividad del programador. No tiene mucho sentido pagar una tarifa por horas más barata a los desarrolladores, si luego prevalece el lastre generado por el trabajo distribuido y el uso de escritorios remotos. Desearíamos que más empresas que brindan trabajo remoto fuera del país informen a sus clientes sobre estas desventajas. Es mucho mejor utilizar un ambiente seguro de “cuarto limpio” fuera de país que permita el desarrollo local, o tener un IDE en la nube (ejemplo: ievms)

LENGUAJES & FRAMEWORKS

A lo largo de muchos años, JavaScript ha crecido para convertirse en probablemente el lenguaje de programación más utilizado en el mundo. Sin embargo, el lenguaje tiene unos pocos problemas que muchos han intentado resolver al utilizar librerías o incluso implementar sus propios lenguajes por encima de JavaScript (antes, hemos mencionado dos: CoffeeScript y ClojureScript. **ECMAScript 6**, la nueva versión de JavaScript, resuelve muchos de los problemas de las versiones anteriores que se utilizan actualmente. A pesar de que no hay mucho soporte para navegadores, el soporte de algunos trans-compiladores o transpiladores, tales como Babel, le permiten escribir ECMAScript 6 y permiten el uso de navegadores más antiguos. Para proyectos nuevos, sugerimos firmemente que se comience utilizando ECMAScript 6 desde el primer día.



Un año después de su lanzamiento público, **Swift** es ahora nuestra opción predeterminada para desarrollo en el ecosistema de Apple. Gracias al lanzamiento reciente de Swift 2, el lenguaje ha alcanzado un nivel de madurez que provee estabilidad y rendimiento según lo requerido por casi todos los proyectos. Swift todavía tiene algunos problemas, sobre todo de soporte de herramientas, refactorización y pruebas. Sin embargo, creemos que estos no son tan graves como para evitar usar Swift. A la vez, el traslado de grandes bases de código escritas en Objective-C no parece que generaría muchos beneficios. El anuncio de que Swift se convertirá en fuente abierta es un hecho positivo adicional. Tenemos la esperanza de que esto no se convertirá otra vez en colocar código desarrollado internamente en un repositorio público, porque Apple ha declarado claramente que se alientan y se aceptan contribuciones de la comunidad.

La mayoría de las plataformas para el manejo de plantillas como Mustache o FreeMarker combina código con algún lenguaje de marcas en un solo archivo para generar contenido dinámico y complejo. **Enlive** es una estructura de plantilla basada en Clojure que separa por completo el lenguaje de programación del marcado HTML y utiliza selectores de CSS para buscar y reemplazar secciones del documento. Enlive demuestra la potencia de la programación funcional para implementar comportamiento complejo por medio de funciones sencillas estructurables que actúan basadas en una abstracción común. Nuestros equipos que trabajan con Clojure la consideran una herramienta muy útil y sencilla.

Tenemos algunas dudas sobre el uso de WebSockets de HTML5. Al permitir al servidor iniciar acciones en el navegador, WebSockets se sale del modelo sin conexiones y de solicitud/respuesta que es la base de la internet ahora. La seguridad es otro gran riesgo de WebSockets. Por ejemplo, el estándar no impone ninguna política de solicitud con respecto al origen. Sin embargo, reconocemos que para ciertas aplicaciones de monitoreo o alerta, WebSockets puede ser muy útil.

ADOPTAR
87. ECMAScript 6
88. Nancy
89. Swift

ENSAYAR
90. Enlive
91. React.js
92. SignalR
93. Spring Boot

EVALUAR
94. Axon
95. Ember.js
96. Frege
97. HyperResource
98. Material UI
99. OkHttp
100. React Native
101. TLA+
102. Traveling Ruby

RESISTIR

LENGUAJES & FRAMEWORKS *continúa*

Si necesita desarrollar un servidor WebSocket en .NET, **SignalR** implementa convenientemente gran parte del código adicional que sea necesario para una aplicación de producción robusta. Esto incluye algunas prácticas de seguridad recomendadas tales como validar testigos de conexión y activar SSl cuando se requiere cifrado. A pesar de que los equipos de ThoughtWorks han estado muy satisfechos con SignalR, todavía hay asuntos básicos pendientes de WebSockets que se debe considerar antes de entrar a mas detalles.

Spring Boot permite la fácil configuración de aplicaciones independientes basadas en Spring. Es ideal para arrancar con microservicios nuevos y fácil de implantar. También facilita el acceso a datos, gracias a mapeos de Hibernate, con mucho menos código de tipo estándar. Nos gusta la idea de que Spring Boot simplifica los servicios de Java creados con Spring, pero hemos aprendido que debemos tener cuidado de la gran cantidad de dependencias. Spring todavía se esconde debajo de la superficie. Si está escribiendo código de microservicios con Java, puede también considerar usar **DropWizard** o una micro-framework como Spark para lograr los beneficios de Spring Boot sin el enorme peso de Spring.

Si bien todavía tenemos dudas sobre CQRS como un patrón general, el enfoque puede funcionar muy bien en lugares específicos. En esas situaciones específicas, sin embargo, para ejecutar adecuadamente CQRS, el desarrollador tiene que realizar bastante trabajo. **Axon** es un framework que puede ayudar a realizar esto en JVM, y lo hemos usado con bastante éxito. A pesar de que ciertamente no puede considerarse una solución perfecta, ahora, Axon continúa evolucionando y podría tener más sentido utilizarla en vez de comenzar desde cero.

Al igual que muchos otros lenguajes de programación, uno de los favoritos de los aficionados a las computadoras es Haskell, que también está disponible en el JVM y se llama **Frege**. Este aporta un lenguaje de programación completamente funcional a la plataforma, lo que permite una fácil interoperabilidad con otros lenguajes de JVM y librerías.

HyperResource es un framework de Ruby que sirve para desarrollar un cliente de API RESTful. Este framework acepta JSON en formato **HAL** y genera dinámicamente un objeto de modelo completo con vínculos de hipermedios (hypermedia links). A pesar de que el framework está todavía en su fase inicial, nos gusta el hecho de que abarca el nivel 3 de REST de Richardson para detectar mejor los servicios y protocolos de auto-documentación.

Material UI provee componentes reutilizables que se pueden usar en aplicaciones de React que implementan el lenguaje Material Design de Google. Al pertenecer a un espacio similar a Twitter Bootstrap, le permite configurar y ejecutar todo rápidamente, pero no tiene los mismos problemas cuando su aplicación crece. Vale la pena investigar Elemental UI como una alternativa.

OkHttp es una librería HTTP de Java creada por Square que provee una interfaz fluida para crear conexiones, además de soporte para el rápido protocolo HTTP/2. Aún cuando se utiliza HTTP/1.1, OkHttp puede mejorar el rendimiento mediante un pool de conexiones y compresión gzip transparente. Al soportar llamadas síncronas bloqueadoras y asíncronas no bloqueadoras, también se puede utilizar como un reemplazo inmediato para el HttpClient de Apache.

Esta es otra opción en el mundo de desarrollo multi-plataforma para dispositivo móviles, **React Native** de Facebook presenta el modelo de programación React.js para desarrolladores de iOS y Android. Los programas de React Native se escriben en JavaScript, pero a diferencia de frameworks híbridos como Ionic, React Native permite a los desarrolladores acceder a componentes nativos de la interfaz de usuario en la plataforma seleccionada. Este es un enfoque que hemos visto antes (por ejemplo,. en Calatrava), pero React Native ya ha inspirado a una comunidad importante de desarrolladores y está aprovechando el momento generado por React.js. Esta framework podría desempeñar una función importante en el futuro del desarrollo de aplicaciones móviles.

El desarrollo de sistemas utilizando microservicios nos hace pensar más detenidamente sobre el aislamiento de fallas y pruebas. **TLA+** es un lenguaje de especificaciones formal que puede ser útil en ambos escenarios. Para aislamiento de fallas, TLA+ puede utilizarse para identificar partes sin variación de su sistema que se pueden monitorear directamente. Una parte sin variación puede ser el número de pedidos a un servicio dividido por número de pedidos a un segundo servicio, por ejemplo. Cualquier cambio en esta proporción generaría una alerta. TLA+ también se está usando para identificar fallas sutiles de diseño en sistemas distribuidos. Amazon, por ejemplo, utiliza revisión de modelos basados en especificaciones

LENGUAJES & FRAMEWORKS *continúa*

formales desarrolladas en TLA+ para identificar pequeños fallos en Dynamo DB, antes de lanzarla al público. Para la mayoría de sistemas, la inversión requerida para crear la especificación formal y luego realizar la revisión del modelo es posiblemente demasiado grande; sin embargo, para sistemas críticos y complejos, o aquellos con muchos usuarios, pensamos que es muy valioso tener otra herramienta en nuestra caja.

Traveling Ruby facilita la distribución de binarios de Ruby portátiles, listos para la ejecutarse e independientes de la plataformas, sin que sea necesario instalar un intérprete, paquetes ni gemas adicionales. Desacopla la ejecución de aplicaciones de Ruby del ambiente de desarrollo en el que se ejecutan.

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 3500 empleados repartidos a lo largo de sus 35 oficinas en 12 países: Australia, Brasil, Canadá, China, Ecuador, Alemania, India, Singapur, Sudáfrica, Turquía, Reino Unido y los Estados Unidos

ThoughtWorks®