Radar Tecnológico

Una guía con opiniones sobre las tecnologías de vanguardia

/thoughtworks

Volume 26

Sobre el Radar

Thoughtworkers son personas a las que les apasiona la tecnología. La construimos, la investigamos, la probamos, abogamos por el código abierto, escribimos sobre ella y constantemente tratamos de mejorarla - para todas las personas. Nuestra misión es defender la excelencia del software y revolucionar la TI. Creamos y compartimos el Radar Tecnológico de Thoughtworks en apoyo de esa misión. El Technology Advisory Board de Thoughtworks, un grupo de líderes tecnológicos de alto nivel de Thoughtworks, crea el Radar. Se reúnen periódicamente para debatir la estrategia tecnológica global de Thoughtworks y las tendencias tecnológicas que tienen un impacto significativo en nuestra industria.

El Radar recoge el resultado de los debates del Technology Advisory Board en un formato que proporciona valor a una amplia gama de partes interesadas, desde las personas desarrolladoras hasta CTOs. El contenido pretende ser un resumen conciso.

Te animamos a explorar estas tecnologías. El Radar es de naturaleza gráfica y agrupa los elementos en técnicas, herramientas, plataformas y lenguajes y frameworks. Cuando los elementos del Radar podían aparecer en varios cuadrantes, elegimos el que nos pareció más apropiado. Además, agrupamos estos elementos en cuatro anillos para reflejar nuestra posición actual al respecto.

Para más información sobre el Radar, consulta thoughtworks.com/es/radar/faq.

Un vistazo al Radar

El Radar se dedica a rastrear cosas interesantes, a las que nos referimos como blips. Organizamos los blips en el Radar utilizando dos elementos de categorización: cuadrantes y anillos. Los cuadrantes representan los diferentes tipos de blips. Los anillos indican en qué fase del ciclo de vida de la adopción creemos que deberían estar.

Un blip es una tecnología o técnica que desempeña un papel en el desarrollo de software. Los blips son cosas que están "en movimiento", es decir, que su posición en el Radar está cambiando, lo que suele indicar que cada vez tenemos más confianza en ellos a medida que avanzan por los anillos.

Adoptar: Estamos convencidas de que la industria debería adoptar estos ítems. Nosotras los utilizamos cuando es apropiado en nuestros proyectos.

Probar: Vale la pena probarlos. Es importante entender cómo desarrollar estas capacidades. Las empresas deberían probar esta tecnología en proyectos en que se puede manejar el riesgo.

Evaluar: Vale la pena explorar con el objetivo de comprender cómo afectará a su empresa.

Resistir: Proceder con precaución.

Nuevo **Desplazado adentro/afuera** **Ningún cambio**

Nuestro Radar está orientado al futuro. Para dar paso a nuevos artículos, desvanecemos los que no se han movido recientemente, lo cual no es un reflejo de su valor, sino de nuestro limitado espacio en el Radar.

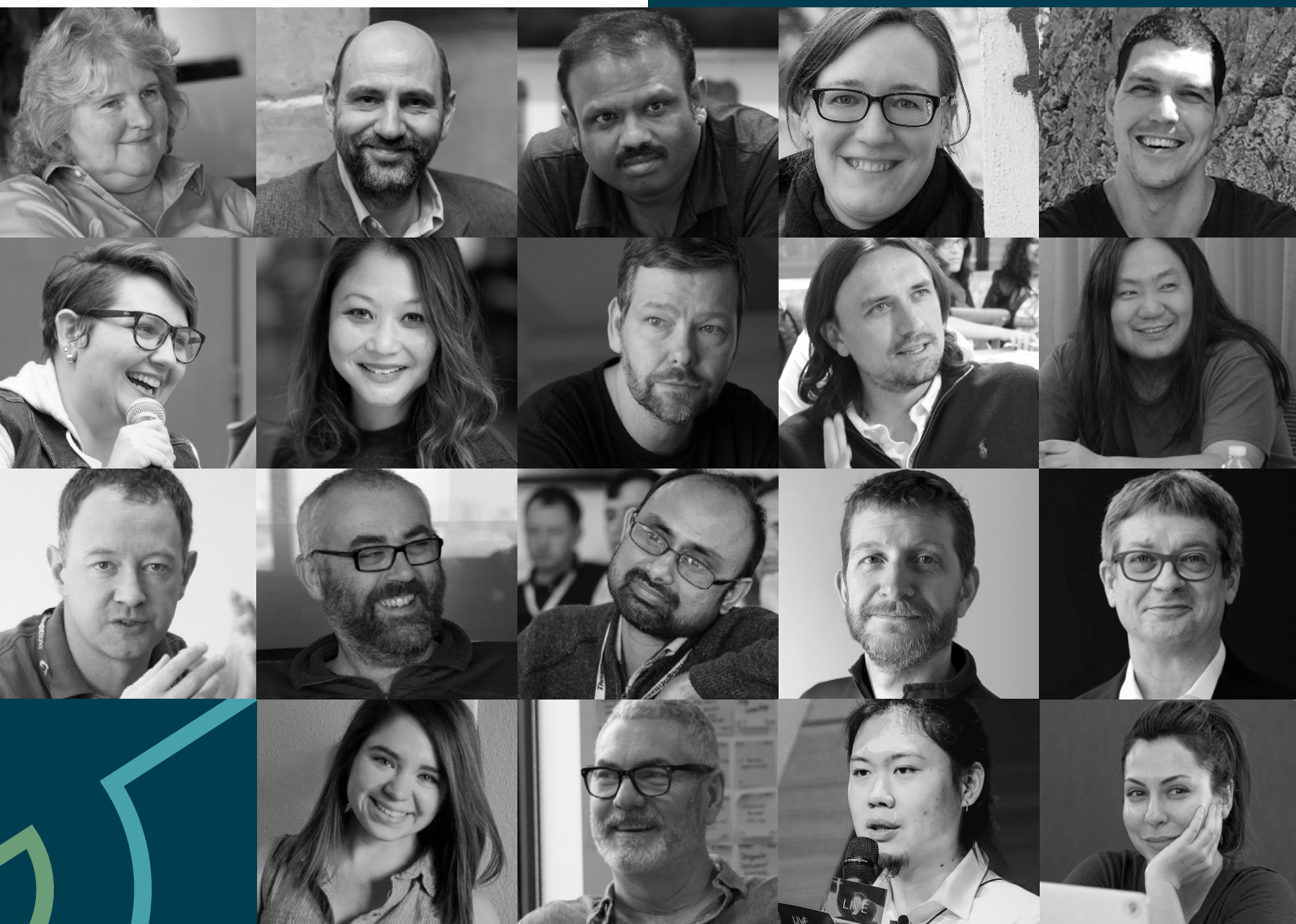
Equipo de traducción al Español: Alejandro Batanero, Alex Ulloa, Alexandra Ortiz, Ana Jiménez Valbuena, Ana María Zúñiga, Andrea Peralta Bravo , Andrés Cáceres, Andrés de los Reyes, Anna Mascaró, Araceli Correa, Ari Handler, Bryan Zuñiga, Camila Paredes, Camila Vigneaux, Carlos Barroso, Carlos Guerrero, Catalina Solís, Christian Álvarez, Cristian Montero, Daniel Negrete, David Benitez, Diana Barreno, Diana Luna, Diana Pila, Elizabeth Parra, Erika Vacacela, Esteban Grijalva, Esteban Villacis, Eugenia Samaniego, Eva Villarroya, Fausto de la Torre, Fernando Tamayo, Francisco Pérez , Gabriel Frías Rivas, Gabriel Loja, Gabriela Guamán, Gaby Gurfinkel, Geovanny Campoverde, Giovanni Serrano, Glenn Wolfschoon, Gonzalo Vázquez Cao, Gorka López de Torre, Gustavo Chiriboga, Gustavo Marin, Helena Gomez , Inigo Crespo Soria, Irene Amo, Javier Alcivar, Javier Escobar, Jéssica Garrigós, Jesús Cardenal, Jhosep Marin, Joan Sanchez, João Lucas Santana, Jorge Agudo Praena, Jorge Palacios, Jorgina Arrés Cardona, Jose Herdoiza, Juan Mateos, Juan Mite, Julieta Corvi, Katherine Ayala , Lara Berra, Laura Mirás, Lucía Parga Basanta, Luis Bustamante, Mafer Escudero, Magdalena Grondona, María José Lalama, Mayfe Yépez, Miguel Hernandez , Milber Champutiz, Oscar Garcia, Pablo Porto, Paola Cajilema, Paula Forero, Paula Marin, Pedro Grijalva, Ricardo Briceño, Rubén Trujillo , Sebastian Muñoz, Silvina Calderon, Valentina Morales , Viviana Proaño, Viviana Quilape y Yanet García

Contribuyentes

El Technology Advisory Board (TAB) es un grupo de 18 personas tecnólogas senior de Thoughtworks. El TAB se reúne dos veces al año en persona y quincenalmente por teléfono. Su función principal es ser un grupo de asesoramiento para el CTO de Thoughtworks, Rebecca Parsons.

El TAB actúa como un organismo amplio que puede examinar los temas que afectan a la tecnología y a las personas tecnólogas de Thoughtworks. Con la actual pandemia mundial, hemos vuelto a crear este volumen del Radar Tecnológico a través de un evento virtual.

Rebecca Parsons (CTO)
Martin Fowler (Chief Scientist)
Bharani Subramaniam
Birgitta Böckeler
Brandon Byars
Camilla Falconi Crispim
Cassie Shum
Erik Dörnenburg
Fausto de la Torre
Hao Xu
Ian Cartwright
James Lewis
Lakshminarasimhan Sudarshan
Mike Mason
Neal Ford
Perla Villarreal
Scott Shaw
Shangqi Liu
Zhamak Dehghani



Temas

El Bazar Extraño: El cambio económico en Open-Source Software

En Thoughtworks, somos fieles seguidores del software de fuente abierta hace tiempo, popularizado en parte por el famoso ensayo de Eric Raymond “The Cathedral and the Bazaar”. El software de fuente abierta abre una ventana al desarrollo de software y promueve la colaboración abierta hacia la corrección de errores y la innovación. Sin embargo, los intentos de comercialización del software de fuente abierta han demostrado la enorme complejidad económica del actual ecosistema. Esto se ha visto, por ejemplo, en el forking de AWS “y” Elasticsearch hacia OpenSearch en Septiembre de 2021 como respuesta al cambio en la licencia por parte de Elastic al requerir a los proveedores de cloud-service, que se beneficiaban de su trabajo, a contribuir de vuelta. Esto muestra cuán difícil puede ser para un comercial de software de fuente abierta el mantener un “foso económico”. (La misma preocupación se aplica con los software de fuente cerrada gratuitos, pues hemos sido testigos de que algunas compañías exploran alternativas a Docker Desktop debido al esfuerzo continuo de Docker en encontrar un modelo comercial óptimo). A veces, la dinámica de poder puede funcionar al contrario: debido a que Facebook fundara Presto como un producto open-source, los “maintainers” (“los que se quedaron”) fueron capaces de mantener la IP y cambiarle el nombre de la marca por Trino después de que estos se hubieran marchado de la compañía, beneficiándose así de la inversión de Facebook.

La historia se vuelve incluso más engorrosa cuando hablamos de la la cantidad de infraestructura crítica que no es patrocinada por corporaciones, donde las compañías se dan cuenta de cuánto de verdad necesitan apoyarse en “trabajo no pagado” cuando aparecen bugs críticos de seguridad (como ha pasado recientemente con Lo4J).
En algunos casos, financiar a “hobbyist maintainers” a través de GitHub o Patreon proporciona el suficiente impulso para hacer la diferencia; en otros casos, simplemente se crea un sentimiento de responsabilidad, adicional a su trabajo diario y contribuye al agotamiento. Seguimos siendo ávidos seguidores del software de fuente abierta pero somos conscientes que la economía alrededor se está convirtiendo cada vez más en algo más bizarro, y no hay soluciones fáciles para encontrar el equilibrio adecuado.

Innovaciones de la Cadena de Suministro de Software

Instancias públicas de problemas severos — La filtración de datos de Equifax, el ataque a SolarWinds, la vulnerabilidad de día cero de Log4J entre otros — que fueron causadas por una escasa gobernanza de la cadena de suministro de software. Los equipos ahora entienden que las prácticas de ingeniería responsables incluyen validar y gestionar las dependencias de los proyectos, esto es un motivador de varios blips de esta edición del Radar. Las entradas incluyen listas de verificación y estándares tales como **Niveles de la Cadena de Suministro para Artefactos de Software (SLSA, por sus siglas en inglés)**, un consorcio respaldado por Google para proveer guías de las amenazas

estándar a la cadena de suministro, y **CycloneDX**, , otro grupo de estándares manejados por la comunidad OWASP. También presentamos herramientas concretas como **Syft**, la cual genera una **Lista de Materiales del Software (SBOM)** desde imágenes de contenedores. Los hackers están cada vez tomando más ventaja de la naturaleza asimétrica de la ofensa y defensa en el campo de la seguridad — solo necesitan encontrar vulnerabilidades, mientras que los defensores deben asegurar toda la superficie de ataque — mientras usan técnicas de hackeo cada vez más sofisticadas. Una seguridad mejorada de la cadena de suministro es una pieza crítica de nuestra respuesta al trabajar para mantener seguros los sistemas.

Por qué los programadores siguen implementando State Management en React?

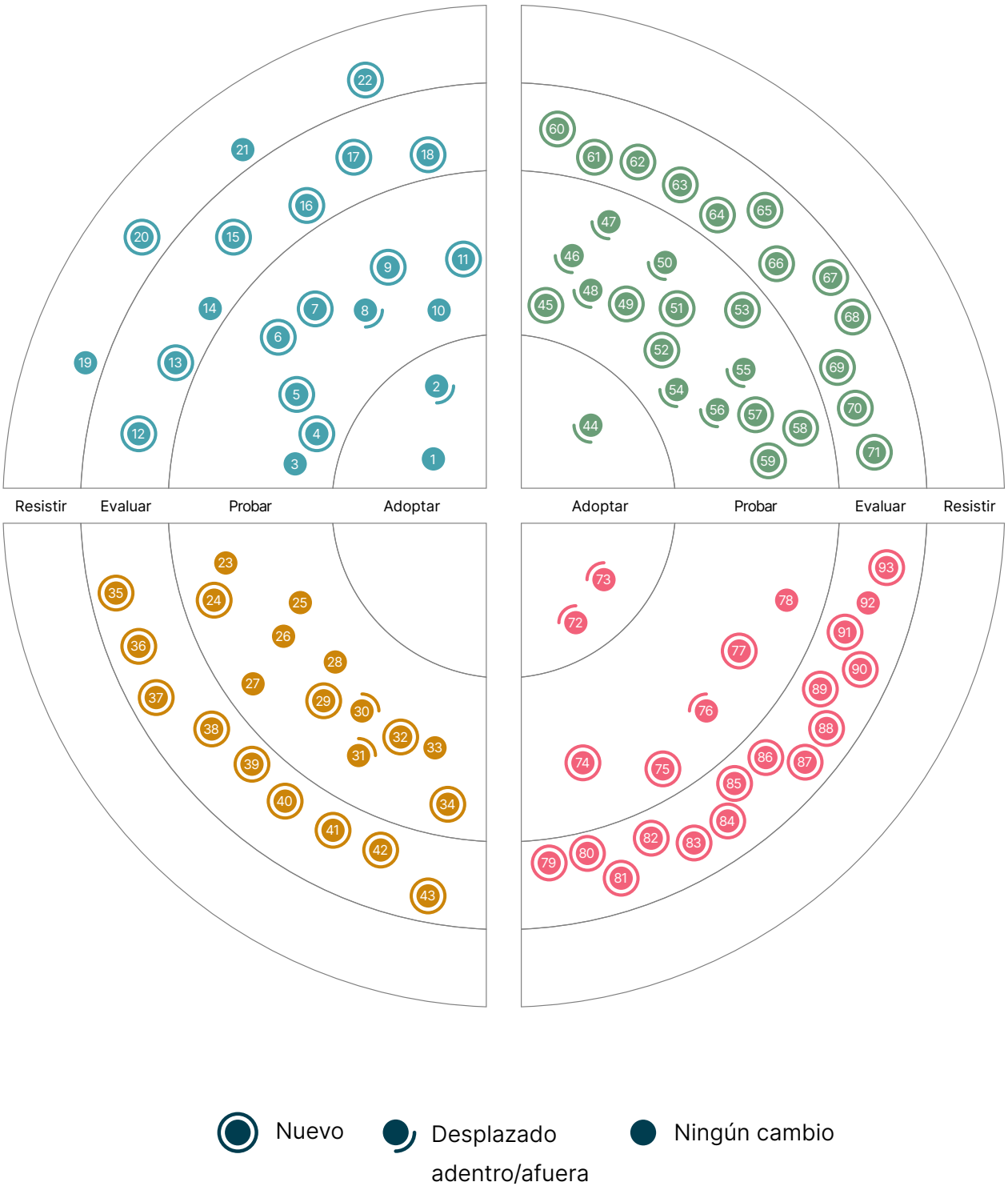
Distintos tipos de frameworks emergentes tienden a aparecer de manera regular en el Radar: un framework se hace popular, seguido por numerosas herramientas que crean un ecosistema de deficiencias y mejoras comunes, dando lugar a la consolidación de varias herramientas populares. Sin embargo, React state management parece ser resistente a esta tendencia. Desde que Redux se lanzó, hemos visto una corriente de herramientas y frameworks que manejan el estado de maneras ligeramente diferentes; cada una de ellas con una serie de desventajas. No sabemos porqué, sólo podemos especular: ¿Es ésta la dinámica de reemplazo que JavaScript pareciera que promueve? ¿Acaso hay una deficiencia debajo de React, un problema divertido y aparentemente manejable que anima a los desarrolladores a experimentar? O es la permanente discordancia entre el formato de lectura de documentos (navegadores web) y la interactividad (y el estado) necesarios para acoger el desarrollo de aplicaciones sobre documentos? No sabemos la razón, pero esperamos a la siguiente ronda de intentos de solucionar este problema a simple vista permanente.

La interminable búsqueda de El Master Data Catalog

El deseo de sacar más valor de los activos de datos corporativos está detrás de la mayoría de las inversiones que estamos viendo en el ámbito de la tecnología digital. Principalmente, este esfuerzo se centra en crear maneras más efectivas de encontrar y tener acceso a datos que sean relevantes. Desde el comienzo en el que las compañías empezaron a recabar datos digitales, han existido esfuerzos de racionalizar y crear un sistema de acceso único, y de arriba hacia abajo, a un directorio de datos corporativo. Sin embargo, una y otra vez, esta idea en principio atractiva se da de bruces con la realidad compleja, redundante y ambigua propias de grandes organizaciones. Recientemente hemos identificado un renovado interés en catálogos de datos corporativos y un surgimiento intermitente de propuestas en el Radar para nuevas e ingeniosas herramientas como **Collibra** y **DataHub**. Estas herramientas pueden proveer acceso regular y detectabilidad al linaje y metadatos a través de silos, pero su cada vez más grande conjunto de características también se extiende a la gobernanza, gestión de calidad, publicación y más.

En contraste con esta moda, también parece haber un creciente movimiento que se aleja del manejo de datos desde arriba hacia abajo y centralizado, hacia una gobernanza federada y el descubrimiento basado en una arquitectura de malla de datos. Este método ataja la complejidad inherente de los datos corporativos fijando expectativas y estándares de una manera central, pero segregando la custodia del dato a través de las líneas de dominio. Los equipos de productos de datos orientados al dominio controlan y comparten sus propios metadatos incluyendo detectabilidad, calidad y otra información. En este escenario, el catálogo es sólo una forma de mostrar información para búsqueda y navegación. Los catálogos de datos resultantes son más simples, más fáciles de mantener y reducen la necesidad de plataformas enriquecidas de catalogación y plataformas de gobernanza.

El Radar



El Radar

Técnicas

Adoptar

1. Cuatro métricas clave
2. Single team remote wall

Probar

3. Malla de datos (Data mesh)
4. Definición de listo para producción
5. Cuadrantes de documentación
6. Repensando las reuniones diarias remotas
7. Interfaz de usuario basada en servidor
8. Software Bill of Materials
9. Bifurcación Táctica
10. Carga cognitiva del equipo
11. Arquitectura de transición

Evaluar

12. CUPID
13. Diseño inclusivo
14. Patrón de operador para recursos no agrupados
15. Malla de servicios sin sidecar
16. SLSA
17. Almacén de datos de streaming
18. TinyML

Resistir

19. Azure Data Factory para orquestación
20. Equipos de plataformas misceláneas
21. Datos de producción en entornos de prueba
22. SPA por defecto

Plataformas

Adoptar

—

Probar

23. Azure DevOps
24. Las plantillas de Azure Pipelines
25. CircleCI
26. Couchbase
27. eBPF
28. GitHub Actions
29. GitLab CI/CD
30. Google BigQuery ML
31. Google Cloud Dataflow
32. Flujos de trabajo reutilizables en Github Actions
33. Sealed Secrets
34. VerneMQ

Evaluar

35. actions-runner-controller
36. Apache Iceberg
37. Blueboat
38. Cloudflare Pages
39. Colima
40. Collibra
41. CycloneDX
42. Embeddinghub
43. Temporal

Resistir

—

El Radar

Herramientas

Adoptar
44. tfsec

Probar
45. AKHQ
46. cert-manager
47. Cloud Carbon Footprint
48. Conftest
49. kube-score
50. Lighthouse
51. Metaflow
52. Micrometer
53. NUKE
54. Pactflow
55. Podman
56. Grafo de Código Fuente
57. Syft
58. Volta
59. Web Test Runner

Evaluar
60. CDKTF
61. Chrome Recorder panel
62. Excalidraw
63. GitHub Codespaces
64. GoReleaser
65. Grype
66. Infracost
67. jc
68. skopeo
69. SQLFluff
70. Validador de Terraform
71. Typesense

Probar
—

Lenguajes y Frameworks

Adoptar
72. SwiftUI
73. Testcontainers

Probar
74. Bob
75. Widget de Flutter-Unity
76. Kotest
77. Swift Package Manager
78. Vowpal Wabbit

Evaluar
79. Android Gradle plugin - Kotlin DSL
80. Azure Bicep
81. Capacitor
82. Java 17
83. Jetpack Glance
84. Jetpack Media3
85. MistQL
86. npm workspaces
87. Remix
88. ShedLock
89. SpiceDB
90. sqlc
91. La Arquitectura Componible
92. Ensamblaje Web
93. Zig

Probar
—

Técnicas

Adoptar

1. Cuatro métricas clave
2. Single team remote wall

Probar

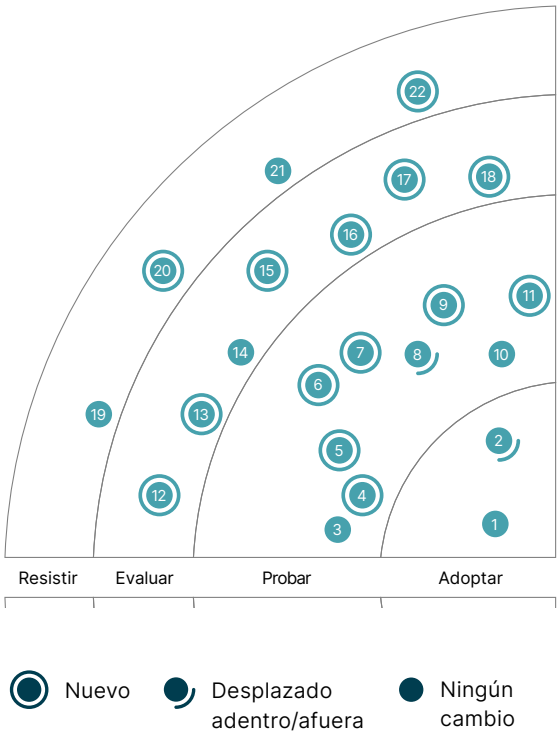
3. Malla de datos (Data mesh)
4. Definición de listo para producción
5. Cuadrantes de documentación
6. Repensando las reuniones diarias remotas
7. Interfaz de usuario basada en servidor
8. Software Bill of Materials
9. Bifurcación Táctica
10. Carga cognitiva del equipo
11. Arquitectura de transición

Evaluar

12. CUPID
13. Diseño inclusivo
14. Patrón de operador para recursos no agrupados
15. Malla de servicios sin sidecar
16. SLSA
17. Almacén de datos de streaming
18. TinyML

Resistir

19. Azure Data Factory para orquestación
20. Equipos de plataformas misceláneas
21. Datos de producción en entornos de prueba
22. SPA by default



1. Cuatro métricas clave

Adoptar

Para medir el rendimiento de la entrega de software, cada vez más organizaciones recurren por defecto a las cuatro métricas clave definidas por el programa **DORA research**: tiempo de espera de los cambios, frecuencia de despliegue, tiempo medio de restauración (MTTR) y porcentaje de fallos en los cambios. Esta investigación y su análisis estadístico han demostrado una clara relación entre el alto rendimiento de la entrega y estas métricas; proporcionan un gran indicador de cómo lo está haciendo una organización de entrega en su conjunto.

Seguimos siendo grandes defensores de estas métricas, pero también hemos aprendido algunas lecciones. Seguimos observando enfoques erróneos con herramientas que ayudan a los equipos a medir estas métricas basándose únicamente en sus conductos de entrega continua (CD). En particular, cuando se trata de las métricas de estabilidad (MTTR y porcentaje de cambios fallidos), los datos del pipeline de CD por sí solos no proporcionan suficiente información para determinar qué es un fallo de despliegue con impacto real en el usuario. Las métricas de estabilidad sólo tienen sentido si incluyen datos sobre incidentes reales que degraden el servicio para los usuarios.

Recomendamos tener siempre presente la intención última de una medición y utilizarla para reflexionar y aprender. Por ejemplo, antes de dedicar semanas a la creación de sofisticadas herramientas de cuadros de mando, considera la posibilidad de limitarse a realizar regularmente la **comprobación rápida del DORA** en las retrospectivas del equipo. Esto da al equipo la oportunidad de reflexionar sobre qué **capacidades** podrían trabajar para mejorar sus métricas, lo que puede ser mucho más eficaz que la creación de herramientas demasiado detalladas. Hay que tener en cuenta que estas cuatro métricas clave se originaron a partir de la investigación a nivel de organización de los equipos de alto rendimiento, y el uso de estas métricas a nivel de equipo debe ser una forma de reflexionar sobre sus propios comportamientos, no sólo otro conjunto de métricas para añadir al cuadro de mando.

2. Single team remote wall

Adoptar

Un single team remote wall es una técnica sencilla para reintroducir virtualmente el muro del equipo. Recomendamos que los equipos distribuidos adopten este enfoque; una de las cosas que escuchamos de los equipos que empezaron a trabajar en remoto es que echan de menos tener el muro físico del equipo. Este era un espacio único donde se podían mostrar todas las tarjetas de historias de usuario, tareas, su estado y progreso, actuando como punto de información y centro de referencia para el equipo. El muro actuó como un punto de integración con los datos reales almacenados en diferentes sistemas. A medida que los equipos se volvieron remotos, tuvieron que acudir de nuevo a los distintos sistemas para buscar la información, por lo que obtener una "visión general" de un proyecto se ha vuelto muy complicado. Si bien puede haber algo de esfuerzo extra para mantenerlo actualizado, creemos que los beneficios para el equipo valen la pena. Para algunos equipos, actualizar el muro físico formaba parte de las "ceremonias" diarias que el equipo hacía en conjunto, y lo mismo se puede hacer con un muro remoto.

3. Malla de datos (Data mesh)

Probar

La malla de datos **Data mesh** es un enfoque organizativo y técnico descentralizado a la hora de compartir, acceder y gestionar los datos para la analítica y el ML. Su objetivo es crear un enfoque

sociotécnico que amplíe la obtención de valor de los datos a medida que crezca la complejidad de la organización y proliferen los casos de uso de los datos y se diversifiquen las fuentes de los mismos. Esencialmente, crea un modelo de intercambio de datos responsable que está en consonancia con el crecimiento de la organización y el cambio continuo. Según nuestra experiencia, el interés por la aplicación de la malla de datos ha crecido enormemente. Este enfoque ha inspirado a muchas organizaciones a adoptarlo y a los proveedores de tecnología a readaptar sus tecnologías existentes para una implantación de malla. A pesar del gran interés y la creciente experiencia en la malla de datos, sus implantaciones se enfrentan a un elevado coste de integración. Además, su adopción sigue limitada a secciones de grandes organizaciones y los proveedores de tecnología están distrayendo a las organizaciones de los aspectos socio más duros de la malla de datos: la propiedad descentralizada de los datos y un modelo operativo de gobierno federado.

Estas ideas se exploran en ***Data Mesh, Delivering Data-Driven Value at Scale***, que guía a los profesionales, arquitectos, líderes técnicos y responsables de la toma de decisiones en su transición desde una arquitectura tradicional de big data a la malla de datos. Proporciona una introducción completa a los principios de la malla de datos y sus componentes; cubre cómo diseñar una arquitectura de malla de datos, guiar y ejecutar una estrategia de malla de datos y navegar por el diseño organizativo hacia un modelo de propiedad de datos descentralizado. El objetivo del libro es crear un nuevo marco para profundizar en las conversaciones y conducir a la siguiente fase de madurez de la malla de datos.

4. Definición de listo para producción

Probar

En una organización que practica el principio “lo construyes, lo ejecutas”, una definición de listo para producción (DPR, por sus siglas en inglés), es una técnica útil para ayudar a los equipos a evaluar y preparar la disponibilidad operacional de nuevos servicios. Implementada como una lista de puntos o una plantilla, un DPR da a los equipos una guía sobre qué tener en cuenta y en qué pensar antes de llevar un nuevo servicio a producción. Aunque los DRP no definen objetivos de servicio específicos (SLOs, por sus siglas en inglés) a cumplir (esto pueden ser difíciles de definir de forma general), recuerdan a los equipos en qué categoría de SLOs pensar, qué estándares organizacionales cumplir, y qué documentación es necesaria. Los DRP, nos dan una fuente de información que los equipos convierten en los requerimientos de producto específicos, por ejemplo: observabilidad y fiabilidad, para incluir en sus backlog de producto.

Los DPR están íntimamente relacionados con el concepto propuesto por Google en **production readiness review (PRR)**. En organizaciones demasiado pequeñas como para tener un equipo de ingeniería de fiabilidad (SRE, por sus siglas en inglés), o preocupadas porque un proceso de panel de revisión pueda impactar el flujo de salida a producción de un equipo, tener un DPR puede al menos ofrecer una guía documentada de criterios acordados a nivel de organización. Para nuevos servicios de alta criticidad, se puede añadir un mayor nivel de escrutinio en cunplimiento de DPR mediante un PRR.

5. Cuadrantes de documentación

Probar

Escribir buena documentación es un aspecto que se pasa por alto en el desarrollo de software y que suele dejarse para el último momento y se realiza de manera desordenada. Algunos de nuestros equipos han encontrado en los **cuadrantes de documentación** una manera práctica de garantizar que se produzcan los artefactos correctos. Esta técnica clasifica los artefactos sobre dos ejes: El primer eje está relacionado con la naturaleza de la información, práctica o teórica; el segundo eje describe el contexto en el que el artefacto es usado, estudiado o trabajado. Esta técnica define cuatro cuadrantes en los cuales artefactos como tutoriales, guías prácticas o páginas de referencia pueden ser clasificadas y entendidas. Este sistema de clasificación no solo asegura que artefactos críticos no sean pasados por alto sino que también guía la presentación del contenido. Hemos encontrado esto particularmente útil a la hora de crear documentación de onboarding que facilita la rápida puesta al día de nuevos desarrolladores cuando se unen a un nuevo equipo.

6. Repensando las reuniones diarias remotas

Probar

El término standup tiene su origen en la idea de ponerse de pie durante esta reunión de sincronización diaria, con el objetivo de hacerla breve. Se trata de un principio común que muchos equipos intentan cumplir en sus reuniones diarias: que sean breves y directas. Pero ahora estamos viendo que los equipos desafían ese principio y están "repensando las reuniones diarias remotas". Cuando se está en un mismo lugar, hay muchas oportunidades durante el resto del día para sincronizarnos entre nosotros de forma espontánea, como complemento a la breve reunión diaria. De forma remota, algunos de nuestros equipos están experimentando un formato de reuniones más largas, similar a lo que la gente de Honeycomb llama "**sincronización de equipo serpenteante**."

No se trata de deshacerse por completo de una sincronización diaria; todavía lo encontramos muy importante y valioso, especialmente en un formato remoto. En cambio, se trata de extender el tiempo bloqueado en los calendarios de todos para la sincronización diaria hasta en una hora, y usarlo de una manera que haga obsoletas algunas de las otras reuniones y acerque al equipo. En las actividades aún se puede incluir el bien probado recorrido por el tablero del equipo, para luego ampliarlo con discusiones de aclaración más detalladas, decisiones rápidas y tomarse el tiempo para socializar. La técnica se considera exitosa si se reduce la carga general de la reunión y mejora la unión del equipo.

7. Interfaz de usuario basada en servidor

Probar

Al armar un nuevo volumen del Radar, a menudo nos invade una sensación de déjà vu, y la técnica de server-driven UI genera un caso particularmente sólido con la llegada de los frameworks que permiten a los desarrolladores de aplicaciones móviles tomar ventaja de los ciclos de cambio más rápidos mientras que no se falle en las políticas de las tiendas de apps en torno a la revalidación de la propia aplicación móvil. Lo hemos comentado antes desde la perspectiva de permitir el desarrollo móvil para **escalar a través de los equipos**. La interfaz de usuario basada en servidor separa la representación en un contenedor genérico en la aplicación móvil mientras que la estructura y la data para cada vista es provista por el servidor. Esto significa que los cambios que alguna vez requirieron un viaje de ida y vuelta a una tienda de apps pueden ahora ser logrados a través de cambios simples en las respuestas que envía el servidor. Nota, no estamos recomendando este acercamiento para todos los desarrollos de interfaz de usuario, de hecho hemos experimentado algunos problemas terribles y demasiado configurados, pero con el respaldo de gigantes como

AirBnB y Lyft, sospechamos que no solo nosotros en Thoughtworks nos cansamos de **hacer todo del lado del cliente**. Mire este espacio.

8. Software Bill of Materials

Probar

Con la continua presión de mantener los sistemas seguros y sin reducción en el panorama general de amenazas, un machine-readable Software Bill of Materials (SBOM) puede ayudar a los equipos a estar al tanto de los problemas de seguridad de las librerías en las que confían. El más reciente, el día cero del exploit remoto **Log4Shell** ue critico y ampliamente conocido. Si los equipos hubieran tenido un SBOM listo, este hubiera escaneado y corregido rápidamente el mismo. Ahora hemos tenido experiencia en ambientes de producción usando SBOMs en proyectos que se encuentran tanto en compañías pequeñas como en grandes multinacionales, hasta en departamentos de gobierno, y estamos convencidos que proveen beneficios. Herramientas como **Syft** hacen fácil el uso de SBOM para detectar vulnerabilidades.

9. Bifurcación Táctica

Probar

Bifucarción táctica es una técnica que puede ayudar con la reestructuración o migración de monolitos a microservicios. Específicamente, esta técnica ofrece una posible alternativa al enfoque más común de primero modularizar completamente el código, que en muchas circunstancias puede tomar mucho tiempo o ser muy difícil de alcanzar. Con la bifurcación táctica un equipo puede crear una nueva bifurcación del código y usarla para tratar y extraer un área o problema en particular mientras se elimina el código innecesario. El uso de ésta técnica probablemente sólo sería parte de un plan a largo plazo para el monolito en general.

10. Carga cognitiva del equipo

Probar

La arquitectura de los sistemas imita la estructura de la organización y su comunicación. No es novedad que las interacciones de equipo deben ser deliberadas. Ver, por ejemplo, la **maniobra inversa de Conway (Inverse Conway Maneuver)**. La interacción del equipo es una de las variables de la rapidez y facilidad con la que los equipos pueden entregar valor a sus clientes. Estuvimos felices de encontrar una manera de medir estas interacciones, usamos la **evaluación** del autor de **Topologías de Equipo**, que te ayuda a comprender qué tan fácil o difícil les resulta a los equipos desarrollar, testear y mantener sus servicios. Midiendo la carga cognitiva del equipo, pudimos aconsejar mejor a nuestros clientes sobre cómo cambiar la estructura de sus equipos y fomentar sus interacciones.

11. Arquitectura de transición

Probar

Una **arquitectura de transición** es una práctica útil utilizada para el reemplazo de los sistemas heredados. Al igual que los andamios pueden construirse, reconfigurarse y finalmente retirarse durante la construcción o renovación de un edificio, pasos arquitectónicos provisionales durante el desplazamiento de lo heredado. Las arquitecturas de transición se eliminarán o sustituirán más adelante, pero no son un trabajo desechable, dado el importante rol que desempeñan en la reducción del riesgo y en permitir que un problema difícil se divida en pasos más pequeños. Por

tanto, ayudan a evitar la trampa de caer en un enfoque de reemplazo heredado “big bang”, porque no se puede hacer que los pasos intermedios más pequeños se alineen con una visión arquitectónica final. Hay que tener cuidado para asegurarse de que el “andamiaje” arquitectónico se elimine finalmente, para que no se convierta en una deuda técnica más adelante.

12. CUPID

Evaluar

¿Cómo te planteas la escritura de un buen código? ¿Cómo puedes juzgar si has escrito código de calidad? Como desarrolladores de aplicaciones, siempre estamos buscando reglas, principios y patrones que podamos utilizar para compartir un lenguaje y unos valores a la hora de escribir código simple y fácil de modificar.

Daniel Terhorst-North ha hecho recientemente un nuevo intento para crear una lista de control para un buen código. Sostiene que, en lugar de ceñirse a un conjunto de reglas como **SOLID**, es más aplicable el uso de un conjunto de propiedades a las que aspirar. Ha ideado lo que llama las propiedades **CUPID** para describir lo que debemos hacer para conseguir un código “alegre”: el código debe ser componible, seguir la filosofía Unix y ser predecible, idiomático y basado en el dominio.

13. Diseño inclusivo

Evaluar

Recomendamos a las organizaciones a evaluar **diseño inclusivo** como manera de asegurar que la accesibilidad es considerada un requisito de primer nivel. Muy habitualmente los requisitos de accesibilidad e inclusividad son ignorados hasta justo antes, o incluso después, del lanzamiento de software. La forma más barata y simple de incluir estos requisitos, a la vez que damos feedback pronto a los equipos, es integrándolos completamente en el proceso de desarrollo. En el pasado, hemos mencionado técnicas que priorizan los requisitos de seguridad y multifuncionales; una perspectiva de esta técnica es que consigue lo mismo para accesibilidad.

14. Patrón de operador para recursos no agrupados

Evaluar

Continuamos viendo un incremento en el uso del patrón de **Operador de Kubernetes** con propósitos más allá de administrar aplicaciones desplegadas en el clúster. Usando el patrón de operador para recursos no agrupados se aprovechan las definiciones de recurso personalizado y el mecanismo de programación guiado por eventos implementados en el panel de control de Kubernetes para gestionar las actividades que están relacionadas fuera del cluster. Esta técnica se basa en la idea de **Kube-managed cloud services** y la extiende a otras actividades, como despliegue continuo o reacción a cambios en repositorios externos. Una de las ventajas de ésta técnica por encima de una herramienta especialmente diseñada, es que abre una amplia gama de herramientas que ya vienen incluidas en Kubernetes o son parte de un ecosistema más amplio. Puedes usar comandos como diff, dry-run o apply para interactuar con los recursos personalizados del operador. El mecanismo de programación de Kube hace que el desarrollo sea más fácil al eliminar la necesidad de orquestar actividades en el orden correcto. Herramientas de código abierto como **Crossplane**, **Flux** y **Argo CD** aprovechan ésta técnica, y esperamos ver más de estas surgir con el tiempo. Aunque éstas herramientas tienen sus casos de uso, también estamos comenzando a ver un mal uso y abuso de ésta técnica y necesitamos repetir un viejo consejo: Sólo porque puedas hacer algo con una herramienta no significa que debas. Asegúrate de descartar enfoques más sencillos y convencionales antes de crear una definición de recurso personalizada y asumir la complejidad que este enfoque implica.

15. Malla de servicios sin sidecar

Evaluar

Malla de servicios suele implementarse como un proxy inverso, también conocido como sidecar, desplegado a la par de cada instancia de servicio. Aunque estos procesos de sidecar suelen ser ligeros, el coste general y la complejidad a la hora de adoptar una malla de servicios se incrementa con cada nueva instancia de servicio que requiere un sidecar adicional. Sin embargo, con los avances en eBPF, estamos observando un nuevo enfoque de malla de servicios sin sidecar donde las funcionalidades de la malla son transferidas de forma segura al núcleo del SO, habilitando a los servicios que se ejecutan en el mismo nodo a comunicarse de forma transparente a través de sockets, sin necesidad de proxies adicionales. Puedes probar esto con Cilium service mesh y simplificar el despliegue pasando de un proxy por servicio a un proxy por nodo. Estamos intrigados por las capacidades de eBPF y por eso nos parece importante evaluar está evolución de la malla de servicios

16. SLSA

Evaluar

A medida que el software continúa aumentando en complejidad, el vector de ataque a través de dependencias de software se convierte en algo cada vez más difícil contra lo que protegerse. La reciente vulnerabilidad de Log4J nos mostró cómo de difícil puede ser incluso conocer esas dependencias — muchas compañías que no usaban Log4J directamente fueron vulnerables sin saberlo solo por el hecho de que otro software en su ecosistema dependía de esta librería. Supply chain Levels for Software Artifacts, o SLSA (pronunciado “salsa”), es un conjunto de guías curadas por un consorcio para que las organizaciones se protejan de ataques contra la cadena de suministro, ha sido evolucionado a partir de las guías internas que Google ha estado usando por años. Apreciamos que SLSA no promete una “bala de plata” con un enfoque centrado en herramientas para proteger la cadena de suministro, sino que proporciona una lista de amenazas concretas y prácticas junto con un modelo de madurez. El modelo de amenazas es fácil de entender ya que cuenta con ejemplos reales de ataques, y los requisitos proveen una guía para ayudar a las organizaciones a priorizar acciones basadas en niveles incrementales de robustez para mejorar su postura de seguridad en la cadena de suministro. Creemos que SLSA provee consejos aplicables y esperamos que más organizaciones aprendan de ello.

17. Almacén de datos de streaming

Evaluar

La necesidad de responder rápidamente a los conocimientos de los clientes ha impulsado la creciente adopción de arquitecturas basadas en eventos y el procesamiento de flujos. Marcos como Spark, Flink o Kafka Streams ofrecen un paradigma en el que consumidores y productores de eventos simples pueden cooperar en redes complejas para ofrecer información en tiempo real. Pero este estilo de programación requiere tiempo y esfuerzo para dominarlo y, cuando se implementa como aplicaciones de un solo punto, carece de interoperabilidad. Hacer que el procesamiento de flujos funcione universalmente a gran escala puede requerir una importante inversión en ingeniería. Ahora, está surgiendo una nueva cosecha de herramientas que ofrece las ventajas del procesamiento de flujos a un grupo más amplio y establecido de desarrolladores que se sienten cómodos utilizando SQL para implementar los análisis. La estandarización de SQL como lenguaje de flujo universal reduce la barrera para la implementación de aplicaciones de flujo de datos. Herramientas como ksqlDB y Materialize ayudan a transformar estas aplicaciones separadas en plataformas unificadas. En conjunto, una colección de aplicaciones de streaming basadas en SQL en toda una empresa podría constituir un almacén de datos de streaming.

18. TinyML

Evaluar

Hasta hace poco, la ejecución de un modelo de machine-learning (ML) se consideraba costosa desde el punto de vista computacional y, en algunos casos, requería un hardware de propósito especial. Si bien la creación de los modelos todavía entra dentro de esta clasificación, es posible crearlos de forma que puedan ejecutarse en dispositivos pequeños, de bajo coste y bajo consumo de energía. Esta técnica, denominada TinyML, ha abierto la posibilidad de ejecutar modelos de ML en situaciones que muchos podrían considerar inviables. Por ejemplo, en dispositivos que funcionan con baterías o en entornos desconectados con una conectividad limitada o irregular, el modelo puede ejecutarse localmente sin un coste prohibitivo. Si te has planteado utilizar el ML pero has creído que no era realista debido a las limitaciones informáticas o de red, merece la pena evaluar esta técnica.

19. Azure Data Factory para orquestación

Resistir

Para las organizaciones que utilizan Azure como su principal proveedor de servicios cloud, Azure Data Factory es actualmente la herramienta predeterminada en cuanto a la orquestación de pipelines de procesamiento de datos. Es compatible con la ingesta de datos, copiando datos desde y hacia diferentes tipos de almacenamientos locales o en Azure y ejecutando lógica de transformación. Aunque sí que hemos tenido una experiencia adecuada con Azure Data Factory en migraciones sencillas de almacenamientos de datos desde local a la nube, no recomendamos el uso de Azure Data Factory para orquestación de pipelines complejas de procesamiento de datos y flujos de trabajo. Hemos tenido algún éxito con Azure Data Factory cuando lo hemos usado principalmente para mover datos entre sistemas. Para los pipelines de datos más complejos, todavía tiene sus retos, incluyendo una depuración mediocre y el informe de errores. Adicionalmente, la observabilidad es limitada, ya que la capacidad de registro de Azure Data Factory no se integra con otros productos como Azure Data Lake Storage o Databricks, lo que hace que sea complicado conseguir observabilidad extremo a extremo. Adicionalmente, la disponibilidad de los mecanismos de data source-triggering (desencadenamiento de fuente de datos) está limitada a ciertas regiones. En estos momentos, animamos a usar otras herramientas de código libre para la orquestación (e.g., Airflow para las pipelines de datos complejas y limitar el uso de Azure Data Factory para la copia de datos o la toma de instantáneas. Nuestros equipos siguen utilizando Data Factory para mover y extraer datos, pero para operaciones más amplias recomendamos otras herramientas con un flujo de trabajo más completo.

20. Equipos de plataformas misceláneas

Resistir

Anteriormente, habíamos presentado los equipos de productos de ingeniería de plataformas en Adoptar, como una buena manera de operar para los equipos de plataformas internos, que permite a los equipos de entrega implementar y operar sistemas en menos tiempo y con herramientas más sencillas. Desafortunadamente, estamos viendo la etiqueta de “equipo de plataformas” aplicada a equipos dedicados a proyectos que no tienen resultados claros o un conjunto de clientes bien definido. Como consecuencia, estos equipos de plataformas misceláneas , como los llamamos, tienen dificultades para entregar debido a las altas cargas cognitivas y a la falta de una alineación clara de prioridades, ya que están lidiando con una colección miscelánea de sistemas que no tienen relación entre sí. De hecho, se convierten en otro equipo de apoyo general para las cosas que no encajan, o que no quieren en otros lados. Seguimos creyendo que los equipos de productos de ingeniería de plataformas centrados en un producto (interno) claro y bien definido ofrecen una mejor serie de resultados.

21. Datos de producción en entornos de prueba

Resistir

Seguimos viendo a los datos de producción en entornos de prueba como un área de preocupación. En primer lugar, muchos casos de esto han resultado en un daño considerable, por ejemplo, cuando se ha enviado una alerta incorrecta desde un sistema de prueba a toda una población de clientes. En segundo lugar, el nivel de seguridad, específicamente en torno a la protección de datos privados, tiende a ser menor para los sistemas de prueba. No tiene mucho sentido tener controles elaborados sobre el acceso a los datos de producción si esos datos son copiados a una base de datos de prueba a la que pueden acceder las personas desarrolladoras y QAs. Aunque es posible ofuscar la información, esto tiende a aplicarse sólo a campos específicos, tal como números de tarjetas de crédito. Por último, copiar datos de producción a sistemas de prueba puede infringir las leyes de privacidad, por ejemplo, cuando los sistemas de prueba se alojan o son accedidos desde un país o región diferente. Este último escenario es especialmente problemático con despliegues complejos en la nube. Los datos falsos son un enfoque más seguro, y existen herramientas para ayudar en su creación. Reconocemos que hay razones para copiar elementos específicos de los datos de producción, por ejemplo, en la reproducción de una incidencia o para el entrenamiento de modelos ML específicos. Aquí nuestro consejo es proceder con precaución.

22. SPA por defecto

Resistir

Generalmente evitamos poner blips en Espera cuando consideramos que la recomendación es demasiado obvia, lo que incluye seguir a ciegas un estilo más architectural sin poner atención a los trade-offs. Sin embargo, debido a la dominancia pura por parte de los equipos al elegir aplicaciones de una página (Single-Page Applications o SPA) cuando realmente lo que necesitan es una página web, nos preocupa hasta tal punto que nos sugiere que no están ni tan siquiera reconociendo SPAs como un estilo de arquitectura y, en vez de ello, se lanzan directamente a la elección de la infraestructura. SPAs generan una complejidad extra que no existe con las tradicionales páginas web (server-web based): herramientas de optimización, gestión y administración del historial de búsqueda, análisis web, tiempo de carga de la página principal, etc. Esta complejidad está generalmente justificada por temas relacionados con la experiencia de usuario, y tanto es así, que se continúan desarrollando herramientas para poder abordar con mayor facilidad estas preocupaciones (aunque la agitación en la comunidad de React en torno a la gestión de estado nos da una pista de lo difícil que puede ser el obtener una solución de aplicabilidad general).

Sin embargo, muy a menudo nos encontramos con equipos que no realizan análisis de trade-off, sino que aceptan a ciegas esa complejidad extra de “SPA por defecto” incluso cuando las necesidades de negocio no la justifican. De hecho, hemos empezado a notar que muchos desarrolladores “novatos” ni siquiera son conscientes de un método alternativo a SPA, pues han pasado la mayoría de su carrera trabajando con infraestructuras similares a React. Creemos que muchas páginas web se beneficirarán de la simplicidad de la lógica de servidor server-side, y nos apoyamos en técnicas como **Hotwire** que ayudan a acortar ese brecha en la experiencia de usuario.

Plataformas

Adoptar

—

Probar

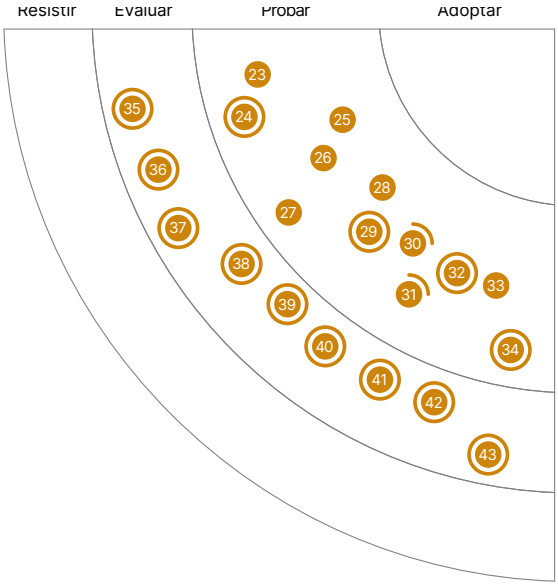
23. Azure DevOps
24. Las plantillas de Azure Pipelines
25. CircleCI
26. Couchbase
27. eBPF
28. GitHub Actions
29. GitLab CI/CD
30. Google BigQuery ML
31. Google Cloud Dataflow
32. Flujos de trabajo reutilizables en Github Actions
33. Kubernetes
34. VerneMQ

Evaluar

35. actions-runner-controller
36. Apache Iceberg
37. Blueboat
38. Cloudflare Pages
39. Colima
40. Collibra
41. CycloneDX
42. Embeddinghub
43. Temporal

Resistir

—



Nuevo

Desplazado adentro/afuera

Ningún cambio

23. Azure DevOps

Probar

Debido a que el ecosistema Azure DevOps sigue creciendo, nuestros equipos lo utilizan con éxito cada vez más. Estos servicios contienen un conjunto de servicios administrados, que incluyen repositorios Git alojados, pipelines de compilación y despliegue, herramientas de pruebas automatizadas, utilidades de administración de tareas pendientes y repositorio de artefactos. Hemos visto a nuestros equipos adquirir experiencia en esta plataforma con muy buenos resultados, lo que significa que Azure DevOps está madurando. Particularmente nos gusta su flexibilidad; permitiéndonos usar los servicios que uno desea incluso si estos son de diferentes proveedores. Por ejemplo, es posible usar un repositorio Git externo al mismo tiempo que los servicios de pipelines de Azure DevOps. Nuestros equipos están especialmente entusiasmados con Azure DevOps Pipelines. A medida que el ecosistema madura, vemos un aumento en la incorporación de equipos que ya se encuentran utilizando Azure, ya que se integra fácilmente con el resto del mundo de Microsoft.

24. Las plantillas de Azure Pipelines

Probar

Las plantillas de Azure Pipelines te permiten eliminar la duplicidad cuando defines tus Azure Pipelines, a través de dos mecanismos. Con “incluir” plantillas, puedes hacer referencia a una plantilla, de tal forma que se expandirá en línea como una macro parametrizado de C++, permitiendo así una forma sencilla de reutilizar una configuración común en todas las etapas, trabajos y pasos. Con “extender” plantilla, puedes definir una capa exterior con la configuración común de la pipeline y con la aprobación requerida de la plantilla, puedes hacer que la compilación falle si la pipeline no extiende de ciertas plantillas, previniendo ataques maliciosos contra la propia configuración de la pipeline. Junto con [CircleCI](/es/radar/platforms/circleci), Orbs y el todavía más reciente GitHub Actions Reusable Workflows, las plantillas de Azure Pipeline son parte de la tendencia sobre la creación de modularidad en el diseño de las pipelines a lo largo de múltiples plataformas y varios de nuestros equipos han estado contentos utilizándolo.

25. CircleCI

Probar

Muchos de nuestros equipos eligen CircleCI para sus necesidades de integración continua y aprecian su capacidad para ejecutar pipelines complejas de manera eficiente. Los desarrolladores de CircleCI continúan agregando nuevas funcionalidades con CircleCI, ahora en la versión 3.0. Orbs y executors fueron llamados por nuestros equipos como particularmente útiles. Los Orbs son fragmentos de código reutilizables que automatizan procesos repetidos, aceleran la configuración de proyectos y facilitan la integración con herramientas de terceros. La amplia variedad de tipos de ejecutores brinda flexibilidad para configurar trabajos en máquinas virtuales en Docker, Linux, macOS o Windows..

26. Couchbase

Probar

Cuando cubrimos Couchbase originalmente en 2013, se consideraba principalmente un caché persistente que evolucionó de una fusión entre Membase y CouchDB. Desde ese entonces, ha mejorado constantemente y un ecosistema de herramientas relacionadas y oferta comercial se ha desarrollado en torno a él. Entre las novedades de la suite del producto está Couchbase Mobile y el Couchbase Sync Gateway. Estas funcionalidades trabajan juntas para mantener actualizados los datos persistentes en dispositivos periféricos, incluso cuando el dispositivo está fuera de red por

periodos extendidos de tiempo debido a conectividad intermitente. A medida que estos dispositivos proliferan, vemos una necesidad más grande de persistencia embebida que continúa funcionando esté el dispositivo conectado o no. Recientemente, uno de nuestros equipos valoró Couchbase por su capacidad de sincronización offline, y descubrió que esta capacidad existente les ahorró considerable esfuerzo que de otra manera lo habrían tenido que invertir en tiempo propio.

27. eBPF

Probar

Desde hace varios años, el kernel de Linux incluye el filtro de paquetes Berkeley extendido (**eBPF**), una máquina virtual que ofrece la posibilidad de adjuntar filtros a determinados sockets. Pero eBPF va mucho más allá del filtrado de paquetes y permite activar scripts personalizados en varios puntos del kernel con muy poca sobrecarga. Aunque esta tecnología no es nueva, está cobrando importancia con el creciente uso de microservicios desplegados como contenedores orquestados. Los Kubernetes y la tecnología de malla de servicios (Services Mesh) como **Istio** se utilizan habitualmente, y emplean sidecars para implementar la funcionalidad de control. Con las nuevas herramientas, — **Bumblebee** en particular, hace que la construcción, ejecución y distribución de programas eBPF sea mucho más fácil - eBPF puede ser visto como una alternativa al sidecar tradicional. Un mantenedor de **Cilium**, una herramienta en este espacio, ha llegado a proclamar **la desaparición del sidecar**. Un enfoque basado en eBPF reduce parte de la sobrecarga de rendimiento y operación que conllevan los sidecars, pero no da soporte a funciones comunes como la terminación SSL.

28. GitHub Actions

Probar

GitHub Actions ha crecido considerablemente el año pasado. Ha demostrado que puede asumir flujos de trabajo más complejos y llamar a otras acciones en acciones compuestas, entre otras cosas. Sin embargo, todavía tiene algunas deficiencias, como su incapacidad para volver a activar un solo trabajo de un flujo de trabajo. Aunque el ecosistema en el **GitHub Marketplace** tiene sus ventajas obvias, al dar acceso a las Acciones de GitHub de terceros a tu pipeline de construcción se corre el riesgo de compartir secretos de forma insegura (recomendamos seguir los consejos de GitHub sobre **security hardening**). Sin embargo, la comodidad de crear tu flujo de trabajo de compilación directamente en GitHub junto a tu código fuente, combinada con la capacidad de ejecutar las Acciones de GitHub localmente utilizando herramientas de código abierto como **act** es una opción convincente que ha facilitado la configuración y la incorporación de nuestros equipos.

29. GitLab CI/CD

Probar

Si estás usando **GitLab** para administrar la entrega de software, también deberías darle un vistazo a **GitLab CI/CD** para tus necesidades de integración y entrega continua. Hemos comprobado que es especialmente útil cuando se utiliza con GitLab con ejecutores en servidores OnPremise y auto-hospedados, ya que esta combinación evita los dolores de cabeza de autorización que suelen producirse al utilizar una solución basada en la nube. Los ejecutores auto-hospedados pueden configurarse completamente para tus propósitos con el sistema operativo y las dependencias correctas instaladas y, como resultado, los pipelines pueden ejecutarse mucho más rápido que cuando se utiliza un ejecutor hospedado en la nube que necesita configurarse cada vez.
Además de las operaciones básicas de compilación, pruebas y despliegue de pipelines, este producto de GitLab soporta el uso de Services, Auto Devops y ChatOps, entre otras características avanzadas. Los Services son útiles al ejecutar servicios Docker como Postgres o **Testcontainer**

vinculados a un job para integración y pruebas de extremo a extremo. Auto Devops crea pipelines que no requieren configuraciones, lo que es muy útil para los equipos que son nuevos en la entrega contínua o para las organizaciones con muchos repositorios que de otro modo tendrían que crear muchos pipelines manualmente.

30. Google BigQuery ML

Probar

Desde la última vez que hablamos sobre **Google BigQuery ML**, se han agregado modelos más sofisticados, como Redes Neuronales Profundas y de Tablas con Autoaprendizaje, al conectar BigQuery ML con TensorFlow y Vertex AI como backend. BigQuery también ha introducido soporte para previsión de series temporales. Una de nuestra preocupaciones anteriormente era **explicabilidad**. A principios de este año fue anunciado **BigQuery Explainable AI**, dando un paso para abordar esto. También podemos exportar modelos de BigQuery ML a Cloud Storage como Tensorflow SavedModel y usarlos para predicción online. Aún quedan cosas pendientes como la dificultad de "entrega continua para aprendizaje automático" pero con su facilidad de uso, BigQuery ML sigue siendo una opción atractiva, sobre todo si la información ya está en BigQuery.

31. Google Cloud Dataflow

Probar

Google Cloud Dataflow es un servicio de procesamiento de datos basado en la nube para aplicaciones de transmisión de datos en tiempo real y por lotes. Nuestros equipos usan Dataflow para crear flujos de procesamiento para integrar, preparar y analizar grandes conjuntos de datos, con el modelo de programación unificado de **Apache Beam** para facilitar su administración. Presentamos Dataflow por primera vez en 2018, y su estabilidad, rendimiento y conjunto completo de funciones nos dan confianza para moverlo a la sección Trial en esta edición del Radar.

32. Flujos de trabajo reutilizables en Github Actions

Probar

Hemos visto un creciente interés en **GitHub Actions** desde que se hizo blip por primera vez hace dos Radars. Con el lanzamiento de **los flujos de trabajo reutilizables**, GitHub ha continuado evolucionando el producto de forma que aborda algunas de sus limitaciones iniciales. Los flujos de trabajo reutilizables en GitHub Actions aportan modularidad al diseño de pipeline, permitiendo la reutilización parametrizada incluso a través de repositorios (siempre y cuando el repositorio de flujo de trabajo sea público). Soportan el paso explícito de valores confidenciales en forma de secretos, y el resultado es devuelto al proceso que lo ha llamado. Con unas cuantas lineas de YAML, Github Actions ahora aporta el tipo de flexibilidad que se ve en **CircleCI** Orbs o **Azure Pipeline Templates**, pero sin tener que dejar la plataforma GitHub.

33. Sealed Secrets

Probar

Kubernetes soporta de manera nativa un objeto valor-clave (key-value en inglés) como secretos (secrets en inglés). Sin embargo, por defecto los secret en Kubernetes no son realmente secretos. Estos son manipulados separadamente de otros datos key-value con precauciones o con controles de acceso que pueda ser aplicados de manera separada. Existe soporte para encriptar secrets antes de que sean guardados en **etcd**, pero estos empiezan como campos de texto plano en archivos de configuración.

Secretos sellados (Sealed Secrets en inglés) es una combinación entre el operador y la utilidad de la línea de comando que usa claves asimétricas para encriptar secrets de manera que solo pueden ser descifrados por un controlador del clúster. Este proceso asegura que los secrets no serán comprometidos mientras se encuentren en los archivos de configuración que definen el deployment en Kubernetes. Una vez encriptados, estos archivos pueden ser compartidos o almacenados de manera segura junto a otros artefactos del deployment.

34. VerneMQ

Probar

VerneMQ es un broker MQTT de código abierto, de alto rendimiento y distribuído. En el pasado hemos analizado otros brokers MQTT como Mosquitto y EMQ. Al igual que EMQ y RabbitMQ, VerneMQ está también basado en Erlang/OTP, lo que lo convierte en altamente escalable. Escala tanto horizontalmente como verticalmente sobre hardware estándar para soportar un gran número de productores y consumidores, manteniendo una baja latencia y tolerancia a fallos. En nuestros bancos de pruebas internos, hemos sido capaces de alcanzar algunos millones de conexiones concurrentes en un único clúster. Aunque no es nuevo, llevamos utilizándolo ya en producción durante algún tiempo y nos ha funcionado bien.

35. actions-runner-controller

Evaluar

actions-runner-controller es un controlador que opera runners auto hospedados para GitHub Actions en tu clúster de Kubernetes. Con esta herramienta creas un recurso runner en Kubernetes, que ejecutará y operará el runner auto hospedado. Los runners auto hospedados son de mucha ayuda en escenarios dónde el trabajo o job que tu GitHub Actions ejecuta necesita acceder a recursos a los que los runners en la nube de GitHub no pueden acceder o tienen requerimientos específicos para un sistema operativo y entorno que difieren de los que provee GitHub. En esos casos dónde tienes un clúster de Kubernetes, puedes ejecutar tus runners auto hospedados como un pod de Kubernetes, con la capacidad de escalar conectándose con los eventos de los webhooks de GitHub.

36. Apache Iceberg

Evaluar

Apache Iceberg Es un formato de tabla abierta para conjuntos de datos analíticos muy grandes. Iceberg admite operaciones de datos analíticos modernos, como la inserción, actualización y eliminación a nivel de registro, time-travel queries, transacciones ACID, partición oculta y evolución completa del esquema. Soporta múltiples formatos de almacenamiento de archivos subyacentes como Apache Parquet, Apache ORC y Apache Avro. Muchos motores de procesamiento de datos soportan Apache Iceberg, incluyendo motores SQL como Dremio y Trino así como motores de streaming (estructurado) como Apache Spark y Apache Flink.

Apache Iceberg está en la misma categoría que Delta Lake y Apache Hudi. Todos ellos soportan más o menos características similares, pero cada uno difiere en las implementaciones subyacentes y en las listas de características detalladas. Iceberg es un formato independiente y no es nativo de ningún motor de procesamiento específico, por lo que es soportado por un número creciente de plataformas, incluyendo AWS Athena y Snowflake. Por la misma razón, Apache Iceberg, a diferencia de los formatos nativos como Delta Lake, puede no beneficiarse de las optimizaciones cuando se utiliza con Spark.

37. Blueboat

Evaluar

Blueboat es una plataforma multiplataforma para aplicaciones web sin servidor. Aprovecha el popular motor JavaScript V8 e implementa bibliotecas de aplicaciones web de uso común de forma nativa en Rust para seguridad y rendimiento. Se puede pensar en Blueboat como una alternativa a CloudFlare Workers o Deno Deploy pero con una distinción importante — debe operar y administrar la infraestructura subyacente. Dicho esto, le recomendamos que evalúe cuidadosamente Blueboat para sus necesidades locales sin servidor.

38. Cloudflare Pages

Evaluar

Cuando los Cloudflare Workers fueron liberados, los destacamos como una temprana función como servicio (FaaS) para cómputo de borde con una implementación interesante. La liberación de las Cloudflare Pages del pasado Abril no se sintió como digna de mención, debido a que Pages es solo una en una clase de muchas soluciones respaldadas por Git y de alojamiento de sitios. Tenía vistas previas continuas, una función útil que no se encuentra en la mayoría de las alternativas. Ahora, sin embargo, Cloudfare tiene unos Workers y Pages integrados, más enfocados, creando una solución totalmente integrada Jamstack corriendo sobre el CDN. La inclusión de un almacenamiento de clave-valor y una coordinación primitiva fuerte y consistente mejoran aún más el atractivo de la nueva versión de Cloudflare Pages.

39. Colima

Evaluar

Colima se está convirtiendo en una popular alternativa abierta para Docker de Escritorio. Provee el tiempo de ejecución del contenedor Docker en una máquina virtual (VM) Lima, configura la interfaz de línea de comando (CLI) en macOS y se encarga del port-forwarding y montaje de volúmenes. Colima utiliza containerd como tiempo de ejecución, que es el más común en la mayoría de servicios Kubernetes (lo cual mejora la relación dev-prod). Con Colima se puede utilizar y probar fácilmente las últimas características de containerd, tal como lazy loading para imágenes en contenedores. Por su buen desempeño, vemos a Colima como una alternativa de software abierto con un gran potencial frente a Docker para Escritorio.

40. Collibra

Evaluar

En el ambiente cada vez más concurrido del mercado corporativo de catálogos de data, nuestros equipos han disfrutado al trabajar con Collibra. Les ha gustado la flexibilidad en el despliegue sea de un SaaS o de una instancia self-hosted, la gran amplitud de funcionalidad incluida out of the box, tales como gobernanza de datos, linaje, calidad y observabilidad. Los usuarios también tienen la opción de utilizar un pequeño subconjunto de capacidades requeridas desde una aproximación más descentralizada como por ejemplo un data mesh. El verdadero punto extra se lo lleva su servicio al cliente muchas veces pasado por alto, quienes se han mostrado colaborativos y de gran soporte. Por supuesto, existe tensión entre catálogos de data simples y plataformas corporativas con mayores funcionalidades, pero hasta ahora los equipos se encuentran muy contentos en cómo Collibra ha soportado sus necesidades.

41. CycloneDX

Evaluar

CycloneDX es un estándar para describir una Lista de materiales de software (SBOM) legible por una máquina. A medida que las estructuras del software y la computación crecen en complejidad, el software se vuelve más difícil de definir. Originario de OWASP, CycloneDX mejora el antiguo estándar SPDX con una definición más amplia que se extiende más allá de las dependencias de la máquina local para incluir dependencias de servicio en tiempo de ejecución. También encontrarás implementaciones en varios lenguajes, un ecosistema de soporte de integraciones y una herramienta de línea de comando CLI que te permite analizar y cambiar SBOMs con la firma y verificación correspondientes.

42. Embeddinghub

Evaluar

Embeddinghub es una base de datos vectorial para embeddings, de machine learning, bastante similar a Milvus. Sin embargo, dado que tiene soporte disponible para operaciones de vecino más cercano (NN) aproximado, particionamiento, control de versiones y control de accesos, recomendamos evaluar Embeddinghub para sus casos de embedding vectorial.

43. Temporal

Evaluar

Temporal es una plataforma para desarrollar flujos de trabajo de larga duración, particularmente para arquitecturas de microservicios. Es una variante del anterior proyecto de OSS Cadence de Uber, tiene un modelo de aprovisionamiento de eventos para flujos de trabajo de larga duración para que puedan sobrepasar caídas de procesos o infraestructura. A pesar de que no recomendamos el uso de transacciones distribuidas en arquitectura de microservicios, si realmente necesitas implementarlas o larga duración Sagas, puedes considerar darle un vistazo a Temporal.

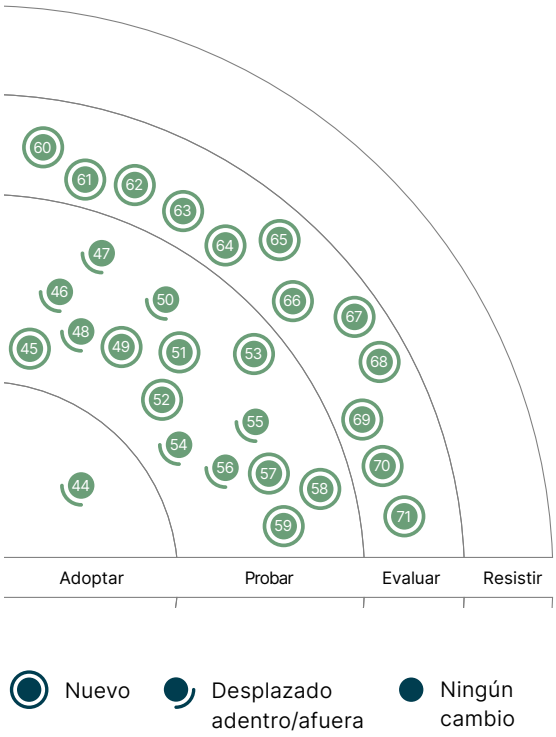
Herramientas

Adoptar
44. tfsec

Probar
45. AKHQ
46. cert-manager
47. Cloud Carbon Footprint
48. Conftest
49. kube-score
50. Lighthouse
51. Metaflow
52. Micrometer
53. NUKE
54. Pactflow
55. Podman
56. Grafo de Código Fuente
57. Syft
58. Volta
59. Web Test Runner

Evaluar
60. CDKTF
61. Chrome Recorder panel
62. Excalidraw
63. GitHub Codespaces
64. GoReleaser
65. Grype
66. Infracost
67. jc
68. skopeo
69. SQLFluff
70. Validador de Terraform
71. Typesense

Resistir
—



44. tfsec

Adoptar

Para nuestros proyectos en los que usamos Terraform, tfsec se ha convertido rápidamente en una herramienta de análisis estático predeterminada para detectar potenciales riesgos de seguridad. Es fácil de integrar en un CI pipeline y tiene una librería en crecimiento de verificaciones contra todos los principales proveedores y plataformas de la nube como Kubernetes. Dado que es sencillo de usar, creemos que tfsec podría ser una buena adición para cualquier proyecto de Terraform.

45. AKHQ

Probar

AKHQ es una interfaz de usuario para Apache Kafka que permite administrar topics, información sobre los topics, grupos de consumidores y mucho más. Algunos de nuestros equipos han encontrado a AKHQ como una herramienta efectiva para mirar en tiempo real el estado de su aplicación Kafka. Puedes, por ejemplo, buscar los distintos topics dentro de tu grupo. Por cada topic, puedes visualizar el nombre, el número de mensajes almacenados, el espacio de disco usado, el tiempo del último registro, el número de particiones, el factor de replicación con la cantidad de in-sync y el grupo de consumidores. Con opciones de deserialización para Avro y Protobuf, AKHQ puede ayudarte a entender el flujo de información en tu aplicación Kafka.

46. cert-manager

Probar

cert-manager es una herramienta para administrar los certificados X.509 dentro de tu cluster de Kubernetes. Modela certificados y emisores como tipos de recursos de primera clase y provee certificados como servicio de manera segura a los desarrolladores y aplicaciones que se ejecutan dentro del cluster de Kubernetes. La opción obvia cuando se utiliza el controlador ingress predeterminado de Kubernetes, también se recomienda para otros y es preferible por sobre el despliegue de tu propia gestión manual de certificados. Varios de nuestros equipos han estado utilizando cert-manager ampliamente y también hemos comprobado que su usabilidad ha mejorado mucho en los últimos meses.

47. Cloud Carbon Footprint

Probar

Cloud Carbon Footprint (CCF) es una herramienta de código abierto que utiliza APIs en la nube para proporcionar visualizaciones de emisiones de carbono estimadas en función del uso de AWS, GCP y Azure. El equipo de Thoughtworks ha usado con éxito la herramienta con varias organizaciones, incluyendo empresas de tecnología energética, minoristas, proveedores de servicios digitales y empresas que utilizan AI. Los proveedores de plataformas en la nube se dan cuenta de que es importante ayudar a sus clientes a comprender el impacto de carbono del uso de sus servicios, por lo que han comenzado a crear una funcionalidad similar por su cuenta. Debido a que CCF es independiente de la nube, permite a los usuarios ver el uso de energía y las emisiones de carbono de múltiples proveedores de nube en un solo lugar, al tiempo que muestra las huellas de carbono en un impacto en el mundo real, como vuelos o árboles plantados.

En versiones recientes, CCF comenzó a incluir recomendaciones de optimización de Google Cloud y AWS junto con ahorros potenciales de energía y CO2, así como para admitir más tipos de instancias en la nube, como instancias de GPU. Dada la tracción que ha recibido la herramienta y la adición continua de nuevas características, nos sentimos confiados en trasladarla a la versión de prueba.

48. Conftest

Probar

Conftest es una herramienta para escribir pruebas usando datos de configuración estructurados. Se apoya en el lenguaje Rego de Open Policy Agent para escribir pruebas de configuraciones de Kubernetes definición de pipelines de Tekton o incluso planes de Terraform. Hemos tenido muy buenas experiencias con Conftest; y con su fácil curva de aprendizaje. Con la rapidez de los tests, nuestros equipos pueden iterar más rápido y seguro cuando hacen cambios de configuración de Kubernetes.

49. kube-score

Probar

kube-score Es una herramienta que permite realizar análisis de código estático de las definiciones de objetos Kubernetes. El resultado es una lista de recomendaciones sobre lo que se puede mejorar para que su aplicación sea más segura y resistente. Tiene una lista de verificaciones predefinidas que incluye las mejores prácticas, como ejecutar contenedores con privilegios non-root y especificar correctamente los límites de recursos. El kuber-score ha existido durante algún tiempo y lo hemos usado en algunos proyectos como parte de un canal CD para los manifiestos de Kubernetes. Un inconveniente importante de kube-score es que no puede agregar políticas personalizadas. Normalmente lo complementamos con otras herramientas como Conftest en estos casos.

50. Lighthouse

Probar

Lighthouse es una herramienta desarrollada por Google para evaluar aplicaciones y páginas web, recolectando métricas de desempeño y aprendizajes sobre buenas prácticas de desarrollo. Desde hace tiempo hemos abogado por las pruebas de desempeño como ciudadanos de primera categoría, y las actualizaciones a Lighthouse que mencionamos hace cinco años ciertamente ayudaron con eso. Nuestro concepto alrededor de funciones de aptitud arquitectural creó una motivación fuerte para tener herramientas como Lighthouse para ser ejecutadas en secuencias automatizadas de compilación. Con la introducción de Lighthouse CI, se ha vuelto más fácil que nunca incluir Lighthouse en secuencias automatizadas de compilación administradas por varias herramientas.

51. Metaflow

Probar

Metaflow es una librería de Python y un servicio de back-end fácil de usar que ayuda a los data scientists e ingenieros de data a construir y administrar el procesamiento de datos que está listo para producción, Machine Learning (ML) y flujos de inferencia. Metaflow provee APIs de Python que estructuran el código como un grafo dirigido de pasos. Cada paso puede ser decorado con configuraciones flexibles como los recursos requeridos de procesamiento y de almacenamiento. Los artefactos de código y de data para ejecutar cada uno de los pasos (conocido como tarea) están almacenados y pueden ser recuperados ya sea para futuras ejecuciones o para los próximos pasos en el flujo, permitiéndole recuperarse de los errores, ejecuciones repetitivas y seguimiento de versiones de modelos y sus dependencias a través de múltiples ejecuciones.

La propuesta de valor de Metaflow es la simplicidad de la librería idiomática de Python: se integra completamente con la infraestructura construida y con el tiempo de ejecución para habilitar la ejecución de la ingeniería de data y las tareas científicas en ambientes de producción locales

y escalados. Al momento de escribir este artículo, Metaflow se integra fuertemente con los servicios AWS como S3 por su servicio de almacenamiento de data y las funciones de paso para la orquestación. Metaflow admite el lenguaje R además de Python. Sus funcionalidades principales son open source.

Si estás construyendo y desplegando tu producción de Machine Learning (ML) y de data-processing pipelines en AWS, Metaflow es una estructura alterna, ligera y full-stack para plataformas más complejas como MLflow.

52. Micrometer

Probar

Micrometer es una librería de tipo plataforma-agnóstica para la instrumentación de métricas en la máquina virtual de Java (JVM) que soporta Graphite, New Relic, CloudWatch y muchas otras integraciones. Hemos encontrado que Micrometer ha beneficiado tanto a autores de librerías como a equipos: autores de librerías pueden incluir código de instrumentación de métricas en sus librerías sin necesidad de dar soporte a cada uno de los sistemas de métricas que los usuarios estén utilizando; y los equipos pueden soportar muchos tipos de distintos de métricas en registros de back-end, lo que permite que las organizaciones colecten métricas de forma consistente.

53. NUKE

Probar

NUKE es una plataforma de compilación para .NET y una alternativa a los tradicionales MSBuild o Cake y Fake que hemos presentado anteriormente en el Radar. NUKE representa las instrucciones de compilación como un DSL de C#, lo que lo hace fácil de aprender y con un buen soporte de IDE. En nuestra experiencia, NUKE hizo realmente simple la automatización de la compilación de proyectos .NET. Nos gusta la precisión de las comprobaciones de código estático y las sugerencias. También nos gusta que podamos utilizar cualquier paquete NuGet sin problemas y que el código de automatización pueda compilarse para evitar problemas en tiempo de ejecución. NUKE no es nuevo, pero su novedoso enfoque -utilizando un DSL de C#- y nuestra positiva experiencia en general nos llevó a incluirlo aquí.

54. Pactflow

Probar

Hemos utilizado Pact para testeo de contratos el tiempo suficiente como para ver parte de la complejidad que se produce a escala. Alguno de nuestros equipos han utilizado Pactflow para reducir fricción. Pactflow se ejecuta tanto como SaaS como una aplicación “On premise” con la misma funcionalidad, y añade usabilidad, seguridad y auditorías mejoradas con respecto a la versión de código abierto Pact Broker. Estamos satisfechos con su uso hasta el momento y contentos de ver un esfuerzo continuado para eliminar parte de la sobrecarga de gestionar el testeo de contratos a escala.

55. Podman

Probar

Como una alternativa a Docker, Podman ha sido validada por muchos de nuestros equipos. Podman presenta un motor “daemonless” (sin dependencias de fondo) para el manejo y ejecución de contenedores, lo que es una acercamiento interesante en comparación a lo que Docker hace. Adicionalmente, Podman puede ser ejecutado fácilmente como un usuario normal sin ser necesario

tener privilegios de administrador, lo que reduce la superficie de ataque. Usando tanto Iniciativas Abiertas de Contenedores (OCI su sigla en inglés) imágenes construidas por Buildah o imágenes de Docker, Podman se puede adaptar a la mayoría de los casos de uso de contenedores. Aparte de algunos problemas de compatibilidad con macOS, nuestro equipo ha tenido en general buenas experiencias con Podman en distribuciones Linux.

56. Grafo de Código Fuente

Probar

En el anterior Radar, presentamos dos herramientas que se encargaban de la búsqueda y el reemplazo de código usando la representación de un árbol de sintaxis abstracta (AST), Comby y Sourcegraph. Si bien estas herramientas comparten ciertas similitudes, estas se diferencian de varias maneras. Sourcegraph es una herramienta comercial (con una capa gratuita de hasta 10-usuarios). Es utilizada particularmente para búsqueda, navegación y referenciación cruzada entre largas bases de código, con énfasis en una experiencia de desarrollo interactiva. Por el contrario, Comby es una herramienta de línea de comandos, ligera, de código abierto utilizada para automatizar tareas repetitivas. Debido a que Sourcegraph es un servicio de hosting, también tiene la capacidad de monitorear continuamente las bases de código y enviar alertas cuando una coincidencia es encontrada. Ahora que hemos ganado más experiencia con Sourcegraph, decidimos moverlo al ring de prueba para reflejar nuestra experiencia positiva, lo que no significa que Sourcegraph sea mejor que Comby. Cada herramienta se enfoca en un nicho diferente.

57. Syft

Probar

Uno de los elementos clave para mejorar la "seguridad de la cadena de suministro" es usar una Lista de materiales de software (SBOM), por eso es cada vez más importante publicar un SBOM junto con el producto de software. Syft Es una herramienta CLI y una biblioteca Go para generar un SBOM a partir de imágenes de contenedores y sistemas de archivos. Puede generar la salida SBOM en múltiples formatos, incluyendo JSON, CycloneDX y SPDX. La salida SBOM de Syft puede ser utilizada por Grype para el escaneo de vulnerabilidades. Una forma de publicar el SBOM generado junto con la imagen es agregarlo como testigo usando Cosign. Esto permite que los consumidores de la imagen verifiquen el SBOM y lo utilicen para un análisis posterior.

58. Volta

Probar

Cuando se trabaja con varias bases de código JavaScript al mismo tiempo, a menudo es necesario utilizar diferentes versiones de Node y otras herramientas JavaScript. En los equipos de los desarrolladores, estas herramientas generalmente suelen estar instaladas en la cuenta de usuario o en la propia máquina, lo que significa que se necesita una solución para cambiar entre múltiples instalaciones. Para Node ya existe nvm , pero queremos destacar Volta como una alternativa que nuestros equipos están usando. Volta tiene varias ventajas sobre nvm: es capaz de gestionar otras herramientas de JavaScript como Yarn ; también tiene la capacidad de fijar una versión específica de la cadena de herramientas en base al proyecto, lo que significa que los desarrolladores pueden simplemente utilizar las herramientas en un directorio de código dado sin tener que preocuparse de cambiar manualmente entre versiones: Volta usa shims en la ruta para seleccionar la versión fijada. Escrito en Rust, Volta es rápido y se distribuye como un binario sin dependencias.

59. Web Test Runner

Probar

Web Test Runner es un paquete dentro del proyecto Modern Web, que proporciona varias herramientas de alta calidad para el desarrollo web moderno con soporte para estándares web como Módulos ES. Web Test Runner es un ejecutor de pruebas para aplicaciones web. Una de sus ventajas en comparación con los ejecutores de prueba existentes es que ejecuta pruebas en el navegador (que podría no tener interfaz). Es compatible con múltiples lanzadores de navegadores, incluidos Titiritero, Playwright y Selenium, y usa Mocha de forma predeterminada para el marco de prueba. Las pruebas se ejecutan bastante rápido y nos gusta que podamos abrir una ventana del navegador con devtools durante la depuración. Web Test Runner utiliza internamente Web Dev Server lo que nos permite aprovechar su excelente API de complementos para agregar complementos personalizados para nuestro conjunto de pruebas. Las herramientas de Modern Web parecen una cadena de herramientas para desarrolladores muy prometedora y ya las estamos usando en algunos proyectos.

60. CDKTF

Evaluar

A estas alturas, muchas organizaciones han creado paisajes extensos de servicios en la nube. Por supuesto, esto solo es posible cuando se usa infraestructura como código y herramientas maduras. Todavía nos gusta Terraform, sobre todo por su rico y creciente ecosistema. Sin embargo, la falta de abstracciones en HCL, el lenguaje de configuración predeterminado de Terraform, crea efectivamente un techo de cristal. El uso de Terragrunt lo lleva un poco más allá, pero cada vez más, nuestros equipos se encuentran anhelando las abstracciones que ofrecen los lenguajes de programación modernos. El Kit de desarrollo en la nube para Terraform (CDKTF), que resultó de una colaboración entre el CDK de AWS y Hashicorp, hace posible que los equipos utilicen varios lenguajes de programación, incluidos TypeScript y Java, para definir y aprovisionar la infraestructura. Con este enfoque sigue el ejemplo de Pulumi mientras permanece en el ecosistema Terraform. Hemos tenido buenas experiencias con CDKTF, pero hemos decidido mantenerlo en el anillo de Evaluación hasta que salga de la versión beta.

61. Chrome Recorder panel

Evaluar

Chrome Recorder panel es una función de vista previa en Google Chrome 97 que permite grabar y reproducir de forma sencilla las interacciones de los usuarios. Aunque definitivamente no es una idea nueva, la forma en que está integrada en Chrome permite crear, editar y ejecutar rápidamente scripts. El panel también se integra muy bien con el panel de rendimiento, lo que facilita la obtención de información repetida y coherente sobre el rendimiento de la página. Aunque las pruebas de estilo de grabación/reproducción siempre deben utilizarse con cuidado para evitar pruebas frágiles, creemos que vale la pena evaluar esta función de vista previa, especialmente si ya estás utilizando el panel de rendimiento de Chrome para medir tus páginas.

62. Excalidraw

Evaluar

Excalidraw es una herramienta de dibujo online, sencilla pero potente, que les encanta utilizar a nuestros equipos. A veces, los equipos sólo necesitan una imagen rápida y no un diagrama formal; para los equipos trabajando en remoto, Excalidraw proporciona una forma rápida de crear y compartir diagramas. A nuestros equipos también les gusta el aspecto de “baja fidelidad” de los

diagramas que puede generar, que recuerda a los diagramas que habrían dibujado en una pizarra si estuvieran trabajando en el mismo lugar. Una advertencia: presta atención a la configuración de seguridad predeterminada, en el momento de escribir, cualquier persona que tenga el enlace puede ver el diagrama. La versión de pago proporciona autenticación más avanzada.

63. GitHub Codespaces

Evaluar

GitHub Codespaces permite a los desarrolladores crear entornos de desarrollo en la nube y acceder a ellos a través de un IDE como si el entorno fuera local. GitHub no es la primera empresa en implementar esta idea; anteriormente hablamos sobre Gitpod. Nos gusta que Codespaces permite que los entornos se estandarizan mediante el uso de archivos de configuración, lo que agiliza la incorporación de nuevos miembros al equipo, y que ofrecen máquinas virtuales con hasta 32 núcleos y 64 GB de memoria. Estas máquinas virtuales se pueden activar en menos de diez segundos, lo que ofrece potencialmente entornos más potentes que una computadora portátil de desarrollo.

64. GoReleaser

Evaluar

GoReleaser es una herramienta que automatiza el proceso de construcción y liberación de un proyecto Go para diferentes arquitecturas a través de múltiples repositorios y canales, una necesidad común para los proyectos Go dirigidos a diferentes plataformas. La herramienta se ejecuta desde la máquina local a través de CI, con la herramienta disponible a través de varios servicios de CI, minimizando así la configuración y el mantenimiento. GoReleaser se encarga de construir, empaquetar, publicar y anunciar cada versión y soporta diferentes combinaciones de formato de paquete, repositorio de paquetes y control de fuentes. Aunque existe desde hace unos años, nos sorprende que no haya más equipos que lo utilicen. Si estás lanzando regularmente una base de código Go, vale la pena evaluar esta herramienta.

65. Grype

Evaluar

Asegurar la cadena de suministro de software se ha convertido en una preocupación habitual entre los equipos de entrega, una preocupación que se refleja en el creciente número de nuevas herramientas en este espacio. Grype es una nueva y ligera herramienta de exploración de vulnerabilidades para imágenes Docker y OCI. Esta herramienta puede ser instalada como un binario, puede escanear imágenes antes de que sean subidas a un registro docker, y no requiere un demonio Docker para ejecutarse en los agentes de construcción. Grype proviene del mismo equipo que está detrás de Syft, el cual genera SBOMs en varios formatos a partir de imágenes de contenedores. Grype puede consumir la salida SBOM de Syft para escanearla en busca de vulnerabilidades.

66. Infracost

Evaluar

Una de las ventajas que se mencionan frecuentemente cuando se habla de moverse hacia la nube es la transparencia alrededor del coste de infraestructura. En nuestra experiencia, no siempre se da este caso. Los equipos no siempre piensan en las decisiones que toman en torno a la infraestructura en términos de coste financiero, y es por ello que ya mencionamos con anterioridad él coste de ejecución como función de aptitud de arquitectura. Tenemos curiosidad por el lanzamiento de una nueva herramienta llamada Infracost que se propone dar visibilidad a los compromisos de coste en

las pull request en Terraform. Es un software de código abierto y está disponible para macOS, Linux, Windows y Docker y muestra los precios para AWS, GCP y Microsoft Azure listos para usar. También proporciona una API pública que puede ser consultada para conocer los datos de coste actualizados. Nuestros equipos están entusiasmados con su potencial, especialmente cuando se trata de ganar mejor visibilidad en los costes en el IDE (Entorno de desarrollo integrado).

67. jc

Evaluar

En el Radar anterior, colocamos **Modern Unix Commands** cómo tecnología a evaluar. Uno de los comandos presentes en esa colección de herramientas era jq, un sed (stream editor) para JSON. **jc** realiza una tarea similar: recoge la salida de comandos de Unix habituales y la convierte a formato JSON. Los dos comandos juntos ofrecen un puente entre el mundo de Unix CLI y el conjunto de librerías y herramientas que funcionan con JSON. Al escribir scripts simples, por ejemplo, para el despliegue de software o la obtención de información de diagnóstico, tener toda una gama de formatos de salida de diferentes comandos Unix mapeados a un formato JSON bien definido puede ahorrarnos mucho tiempo y esfuerzo. Al igual que con jq, es necesario asegurarse de que el comando está disponible. Puede ser instalado desde muchos de los repositorios más conocidos.

68. skopeo

Evaluar

skopeo es una utilidad de línea de comando que realiza varias operaciones sobre imágenes de contenedores y repositorios de imágenes. No requiere que un usuario sea administrador para realizar la mayoría de sus operaciones ni precisa de un demonio para ejecutarse. Es una parte útil del pipeline de integración continua; lo hemos utilizado para copiar imágenes de un registro a otro a medida que se promovían las imágenes. Es mejor que realizar un "pull" y un "push" ya que no es necesario almacenar las imágenes localmente. No es una herramienta nueva, pero es suficientemente útil y poco utilizada por lo que pensamos que vale la pena mencionarla.

69. SQLFluff

Evaluar

Si bien el linting (el análisis estático del código fuente) es una práctica antigua en el mundo del software, su adopción en el mundo de los datos ha sido más lenta. **SQLFluff** es un linter (herramienta para hacer linting) SQL de dialecto cruzado escrito en Python, que se lanza a través de línea de comandos (CLI), lo que hace que sea fácil incorporarlo en las pipelines de CI/CD. Si estás cómodo con las convenciones predeterminadas, entonces SQLFluff trabaja sin ninguna otra configuración adicional después de instalarlo, imponiendo un conjunto de estándares de formato fuertemente definidos; establecer tus propias convenciones requiere añadir un fichero de configuración. La interfaz por línea de comandos (CLI) puede corregir automáticamente ciertos tipos de violaciones que incluyen problemas de formato como los espacios en blanco o las mayúsculas en las palabras clave. SQLFluff es todavía nuevo, pero estamos emocionados por ver a SQL tener más presencia en el mundo del linting.

70. Validador de Terraform

Evaluar

Las organizaciones que han adoptado **infraestructura como código** y plataformas de infraestructura de autoservicio están buscando formas de brindar a los equipos la máxima autonomía mientras se mantienen aplicando buenas prácticas de seguridad y políticas organizacionales. Hemos resaltado **tfsec** before and are moving it into the Adopt category in this Radar. For teams working on GCP, **Terraform Validator** antes y lo estamos moviendo a la categoría Adoptar en este Radar. Para los equipos que trabajan en GCP, Terraform Validator podría ser una opción al crear una biblioteca de políticas, un conjunto de restricciones que se comparan contra las configuraciones de Terraform.

71. Typesense

Evaluar

Typesense Es un motor de búsqueda de texto rápido y tolerante a errores tipográficos. Para casos de uso con grandes volúmenes de datos, Elasticsearch podría seguir siendo una buena opción, ya que proporciona una solución de búsqueda horizontalmente escalable basada en disco. Sin embargo, si estás construyendo una aplicación de búsqueda sensible a la latencia con un tamaño de índice de búsqueda que puede caber en memoria, Typesense es una alternativa poderosa y otra opción a evaluar junto a herramientas como **Meilisearch**.

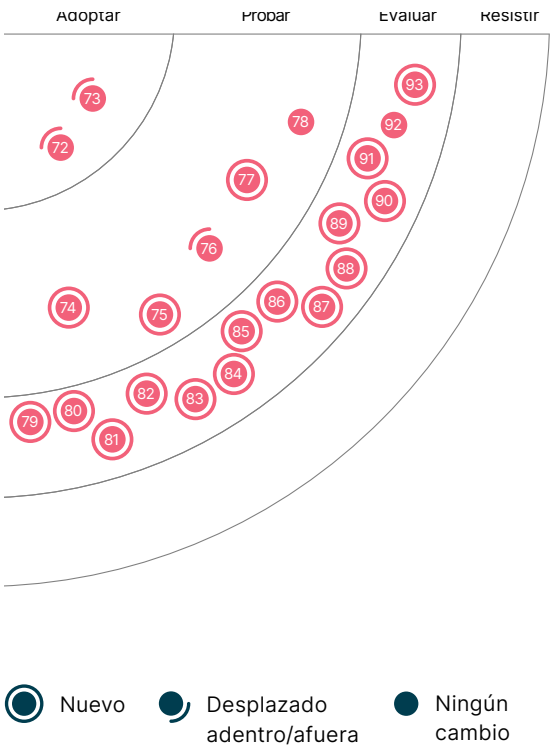
Lenguajes y Frameworks

Adoptar
72. SwiftUI
73. Testcontainers

Probar
74. Bob
75. Widget de Flutter-Unity
76. Kotest
77. Swift Package Manager
78. Vowpal Wabbit

Evaluar
79. Android Gradle plugin - Kotlin DSL
80. Azure Bicep
81. Capacitor
82. Java 17
83. Jetpack Glance
84. Jetpack Media3
85. MistQL
86. npm workspaces
87. Remix
88. ShedLock
89. SpiceDB
90. sqlc
91. La Arquitectura Componible
92. Ensamblaje Web
93. Zig

Resistir
—



72. SwiftUI

Adoptar

Cuando Apple presentó **SwiftUI** hace unos años, fue un gran paso para la implementación de interfaces de usuario en todo tipo de dispositivos fabricados por Apple. Desde el principio, nos gustó el enfoque declarativo, centrado en el código y el modelo de programación reactiva proporcionado por **Combine**. Sin embargo, nos dimos cuenta de que escribir muchas pruebas de vista, que todavía se necesitan con un patrón modelo-vista-modelovista (MVVM), no se alineaba realmente con el marco de automatización XCUITest proporcionado por Apple. Esta brecha ha sido cerrada por **ViewInspector**. Un último obstáculo era la versión mínima del sistema operativo requerida. En el momento del lanzamiento, sólo las últimas versiones de iOS y macOS podían ejecutar aplicaciones escritas con SwiftUI, pero debido a la cadencia regular de actualizaciones de Apple, las aplicaciones de SwiftUI ahora pueden ejecutarse en prácticamente todas las versiones de macOS y iOS que reciben actualizaciones de seguridad.

73. Testcontainers

Adoptar

Tenemos suficiente experiencia en el uso de **Testcontainers** sobre la cual creemos que es por defecto la opción más adecuada para crear entornos confiables para la ejecución de pruebas. Es una librería que se puede usar en **múltiples lenguajes**, y dockeriza las librerías de test más comunes (incluyendo varios tipos de bases de datos, tecnologías de encolamiento, servicios en la nube y dependencias de UI testing como navegadores) con la habilidad de ejecutar Dockerfiles personalizados cuando sea necesario. Funciona bien en frameworks de testeo como JUnit, es suficientemente flexible como para dejar que los usuarios gestionen el ciclo de vida y la conexiones de red de los contenedores para que rápidamente tengan seteado un entorno integrado de pruebas. Nuestros equipos la encuentran como una librería programable, ligera y una manera fácil de manejar contenedores para hacer de sus tests aún más confiables.

74. Bob

Probar

Cuando se construye una aplicación con React Native, a veces nos vemos en la necesidad de crear nuestros propios módulos. Por ejemplo, nos hemos encontrado con esta necesidad de crear una librería de componentes de UI para una aplicación React Native. Desarrollar módulos no es tan sencillo pero nuestros equipos han demostrado solvencia usando **Bob** para automatizar esta tarea. Bob proporciona una CLI para crear el andamiaje (scaffolding) para diferentes tipos de proyectos. El andamiaje no se limita a la funcionalidad base, sino que, opcionalmente, puede incluir código de ejemplo, linters, configuraciones de pipelines de construcción, etc.

75. Widget de Flutter-Unity

Probar

Flutter es cada vez más popular para crear aplicaciones móviles multiplataforma, y Unity es excelente para crear experiencias AR/VR. Una pieza clave en el rompecabezas para integrar Unity y Flutter es el **widget de Flutter-Unity**, que permite incorporar aplicaciones de Unity dentro de aplicaciones Flutter. Una de las capacidades clave que ofrece el widget es la comunicación bidireccional entre Flutter y Unity. Hemos encontrado que su rendimiento también es bastante bueno, y esperamos aprovechar Unity en más aplicaciones de Flutter.

76. Kotest

Probar

Kotest (anteriormente llamado Kotlin Test) es una herramienta de testing independiente para el Kotlin ecosistema que continua ganando fuerza entre nuestros equipos a través de varias implementaciones de Kotlin - nativo, JVM o JavaScript. Las principales ventajas son que ofrece una gran variedad de estilos de testing para organizar los test suites y que viene con un amplio y completo conjunto de matchers, los cuales permiten tests 'expresivos' conjuntamente con una DSL interna elegante. Además de su soporte para property-based testing — una técnica que hemos destacado previamente en el Radar – a nuestros equipos les gusta el sólido plugin de IntelliJ y la creciente comunidad de soporte al rededor.

77. Swift Package Manager

Probar

Algunos lenguajes de programación, especialmente los más nuevos, tienen una solución integrada para administración de paquetes y dependencias. Cuando fue presentado en 2014, Swift no venía con un administrador de paquetes, y entonces la comunidad de desarrolladores para macOS y iOS simplemente continuó usando CocoaPods y Carthage, tlas soluciones de terceros que habían sido creadas para Objective-C. Un par de años más tarde Swift Package Manager ((SwiftPM) fue iniciado como un proyecto de código abierto oficial de Apple, y entonces tomó unos pocos años más antes que Apple pudiera añadir su soporte para Xcode. Incluso en ese punto, muchos equipos de desarrollo continuaban usando CocoaPods y Carthage, en su mayoría porque muchos paquetes simplemente no estaban disponibles a través de SwiftPM. Ahora que la mayoría de los paquetes pueden ser añadidos a través de SwiftPM y los procesos han sido simplificados tanto para creadores como consumidores de los paquetes, nuestros equipos están confiando cada vez más en SwiftPM.

78. Vowpal Wabbit

Probar

Vowpal Wabbit es una una librería de uso general para aprendizaje automático (machine learning). Fue creada originalmente por Yahoo! Research hace más de una década, Vowpal Wabbit sigue implementando nuevos algoritmos de refuerzo del aprendizaje. Queremos resaltar que Vowpal Wabbit 9.0, ha publicado una nueva versión importante después de seis años, y alentarlos a planear la migración ya que tiene varias mejoras de usabilidad, nuevas reducciones y arregla varios defectos.

79. Android Gradle plugin - Kotlin DSL

Evaluar

Android Gradle plugin Kotlin DSL ha agregado soporte para Kotlin Script como alternativa a Groovy para los scripts de compilación de Gradle. El objetivo de reemplazar Groovy con Kotlin es proporcionar un mejor soporte para la refactorización y una edición más simple en los IDE (Entorno de desarrollo integrado) y, en última instancia, producir código que sea más fácil de leer y mantener. Para equipos que ya usan Kotlin, también significa trabajar en la compilación en un lenguaje familiar. Hemos tenido un equipo con un script de compilación de 450 líneas de al menos siete años de antigüedad que migró en unos pocos días. Si tienen scripts de compilación de Gradle grandes o complejos, entonces vale la pena evaluar si Kotlin Script producirá mejores resultados para sus equipos.

80. Azure Bicep

Evaluar

Para aquellos que prefieren un lenguaje más natural que JSON para código de infraestructura, **Azure Bicep** es un lenguaje específico de dominio (DSL) que utiliza una sintaxis declarativa. Soporta plantillas parametrizadas reutilizables para definiciones de recursos modulares. Una **extensión de Visual Studio Code** proporciona seguridad de tipo instantánea, intellisense y verificación de sintaxis, y el compilador permite la transpilación bidireccional hacia y desde plantillas ARM. El DSL orientado a recursos de Bicep y la integración nativa con el ecosistema de Azure lo convierten en una opción convincente para el desarrollo de infraestructura de Azure.

81. Capacitor

Evaluar

Hemos estado discutiendo los méritos de las herramientas de desarrollo móvil multi-plataforma tanto tiempo como el que hemos llevado publicando el Radar Tecnológico. En 2011, nos dimos cuenta por primera vez de una nueva generación de herramientas cuando creamos el blip sobre **plataformas multi-dispositivo**. Aunque permanecimos escépticos al principio, estas herramientas se han perfeccionado y han sido adoptadas ampliamente a lo largo de los años. Y nadie puede discutir la popularidad duradera y la utilidad de **React Native**. **Capacitor** es la última generación de una línea de herramientas que comenzó con PhoneGap, y luego fue renombrada a **Apache Cordova**. Capacitor es una reescritura completa de Ionic que abraza el estilo de las **aplicaciones web progresivas** para aplicaciones independientes. Hasta ahora, a nuestras desarrolladoras les gusta que pueden conseguir crear aplicaciones web, iOS y Android con una única base de código y que pueden gestionar las plataformas nativas de forma separada con acceso a APIs nativas cuando es necesario. Capacitor ofrece una alternativa a React Native, que tiene muchos años de experiencia multi-plataforma a sus espaldas.

82. Java 17

Evaluar

No solemos presentar nuevas versiones de lenguajes de forma rutinaria, pero queríamos destacar la nueva versión de soporte a largo plazo (LTS) de Java, la versión 17. Si bien hay nuevas funciones prometedoras, como la vista previa de **búsqueda de patrones**, es el cambio al nuevo proceso de LTS lo que debería interesar a muchas organizaciones. Recomendamos a las organizaciones que evalúen nuevas versiones de Java a medida que estén disponibles, asegurándose de adoptar nuevas funciones y versiones según convenga. Sorprendentemente, muchas organizaciones no adoptan nuevas versiones de lenguajes de manera rutinaria, a pesar de que las actualizaciones periódicas ayudan a mantener las cosas pequeñas y manejables. Con suerte, el nuevo proceso de LTS, junto con las organizaciones pasando a hacer actualizaciones periódicas, ayudará a evitar la trampa de "demasiado caro actualizar" que acaba con el software de producción ejecutándose en una versión de Java al final de su vida útil.

83. Jetpack Glance

Evaluar

Android 12 ha traído cambios significativos en los widgets de las aplicaciones, las cuales han mejorado la experiencia del usuario final y del desarrollador. Para realizar aplicaciones de Android, hemos expresado nuestra preferencia por **Jetpack Compose** como una forma moderna de construir interfaces de usuario nativas. Ahora, con **Jetpack Glance**, el cual es construido sobre el Compose runtime, los desarrolladores pueden usar similares APIs declarativas de Kotlin para escribir widgets.

Recientemente Glance ha sido **extendido** para soportar Tiles en Wear OS.

84. Jetpack Media3

Evaluar

Hoy en día, Android tiene una gran variedad de APIs: Jetpack Media, también conocido como MediaCompat, JetpackMedia2 y ExoPlayer. Desafortunadamente, estas bibliotecas fueron desarrolladas independientemente, con distintos objetivos que sobreponen funcionalidades. Las personas que desarrollan con Android no solamente tenían que escoger qué biblioteca usar, sino que también tenían que encargarse de hacer adaptadores u otras medidas para poder usar distintas bibliotecas a la vez. **Jetpack Media3** es un esfuerzo, actualmente en acceso anticipado, de crear una nueva API que contiene las funcionalidades comunes de las APIs que ya existen (incluyendo interfaz de usuario, playback y manejo de sesiones de multimedia) conbinándolas para crear una unida y refinada API. La interfaz de ExoPlayer también ha sido actualizada, mejorada y simplificada para ejercer como interfaz común para Media3.

85. MistQL

Evaluar

MistQL es un pequeño lenguaje específico de dominio que sirve para hacer cálculos en estructuras similares a JSON. Originalmente construido para la extracción manual de funcionalidades de modelos de machine learning en front end, MistQL actualmente soporta una implementación en JavaScript para navegadores y una implementación en Python para casos de uso del lado del servidor. Nos gusta su limpia sintaxis componible y funcional. Animamos a evaluarlo basándose en sus necesidades.

86. npm workspaces

Evaluar

Mientras que muchas herramientas se pueden usar en el mundo de desarrollo multipaquete de node.js, npm 7 añade soporte directo con **npm workspaces**. Gestionar paquetes relacionados en conjunto facilita el desarrollo, permitiéndote, por ejemplo, guardar multiples librerías relacionadas en un solo repositorio. Con npm workspaces, una vez hayas añadido una configuración en un package.json de alto nivel que haga referencia a uno o más archivos de package.json anidados, comandos como npm install trabajarán a través de multiples paquetes, enlazando simbólicamente los paquetes fuente dependientes en el directorio raíz node_modules. Otros comandos también estarán al tanto de los workspaces, permitiéndote, por ejemplo, ejecutar los comandos npm run y npm test a través de múltiples paquetes con un solo comando. Tener esta flexibilidad desde el primer momento reduce la necesidad de algunos equipos de llegar a usar otro gestor de paquetes.

87. Remix

Evaluar

Hemos sido testigos de la migración de renderizar sitios web desde el lado del servidor a una aplicación de una sola página dentro del navegador, ahora el péndulo del desarrollo web parece balancearse al medio de todo. **Remix** es uno de estos ejemplos. Es un framework fullstack para JavaScript. El framework provee una carga rápida para las páginas utilizando sistemas distribuidos y navegadores nativos en lugar de compilaciones estáticas ineficientes. Ha logrado algunas optimizaciones en enrutamientos anidados y en la carga de páginas, lo que hace que la representación de la página parezca especialmente rápida. Mucha gente comparará Remix con **Next.js**, que está similarmente posicionada. Nos complace ver a este framework combinar ingeniosamente el tiempo de ejecución del navegador con el tiempo de ejecución del servidor

para dar una mejor experiencia de usuario.

88. ShedLock

Evaluar

Ejecutar una tarea programada una única vez en un cluster de procesadores distribuidos es un requerimiento relativamente común. Por ejemplo, la situación podría surgir cuando se ingesta un lote de datos, se envía una notificación o se ejecuta alguna actividad de limpieza periódica. Pero, este es un problema notablemente complicado. ¿Cómo es que un grupo de procesos cooperan de manera segura sobre redes lentas y poco confiables? Algún mecanismo de bloqueo es necesario para coordinar las acciones a través del cluster. Afortunadamente, una variedad de almacenamientos distribuidos pueden implementar un bloqueo. Sistemas como **ZooKeeper** y **Consul,** así como también bases de datos como DynamoDB o **Couchbase** poseen mecanismos subyacentes para administrar el consenso dentro del cluster. **ShedLock** es una pequeña biblioteca para aprovecharse de estos proveedores en tu código Java, si estás buscando implementar tus propias tareas programadas. Cuenta con un API para adquirir y liberar bloqueos, así como también conectores para una amplia variedad de proveedores de bloqueo. Si estás escribiendo tus propias tareas distribuidas pero quieres evitar la complejidad que implica una plataforma de orquestación como **Kubernetes**, vale la pena darle un vistazo a ShedLock.

89. SpiceDB

Evaluar

SpiceDB es un sistema de base de datos inspirado en **Zanzibar**, (creado por Google), para gestionar permisos en aplicaciones. Con SpiceDB, defines un esquema para modelar los requisitos y usas **el cliente** para aplicar el esquema a una de las **base de datos soportadas**, insertar permisos o pedir respuesta a preguntas del tipo “¿Tiene este usuario acceso a este rescurso?” o incluso al revés “¿Cuales son todos los recursos a los que este usuario tiene acceso?” Normalmente abogamos por la separación de las reglas de autorización y el código, pero SpiceDB va un paso más allá separando los recursos y las reglas almacenando como un grafo para responder eficientemente a consultas sobre autorización. Debido a esta separación, debes asegurarte que los cambios en los datos de tu aplicación principal se vean reflejados en SpiceDB. Entre otras implementaciones inspiradas en Zanzibar, creemos que SpiceDB es una herramienta a valorar para gestionar autorizaciones en tu aplicación.

90. sqlc

Evaluar

sqlc es un compilador que genera código Go idiomático y con seguridad de tipos a partir de SQL. A diferencia de otros enfoques basados en un mapeo objeto-relacional (ORM), puedes seguir escribiendo sencillamente SQL para lo que necesites. Una vez invocado, sqlc comprueba la corrección del SQL y genera un código Go eficiente, que puede ser llamado directamente desde el resto de la aplicación. Con un soporte estable tanto para PostgreSQL como MySQL, merece la pena echarle un vistazo a sqlc y te animamos a que lo pruebes.

91. La Arquitectura Componible

Evaluar

Desarrollar aplicaciones para iOS se ha convertido con el paso del tiempo en un proceso más eficiente, y **SwiftUI** migrando a Adopt es una señal de ello. Yendo más allá de la naturaleza general de SwiftUI y otros marcos de trabajo comunes, **La Arquitectura Componible** (LAC) es a la vez una librería y un estilo arquitectónico para construir aplicaciones. Fue diseñado en el transcurso de

una serie de videos, y los autores han declarado que tenían en mente la composición, las pruebas y la ergonomía, basándose en ideas de La Arquitectura Elm y Redux. Como era de esperar, el ámbito reducido y el adoctrinamiento son a la vez la fortaleza y la debilidad de LAC. Sentimos que los equipos que no tienen mucha pericia escribiendo aplicaciones iOS, que a menudo son equipos que se encargan de múltiples códigos fuente relacionados con diferentes stacks tecnológicos, son los que más pueden beneficiarse de usar un marco de trabajo doctrinario como LAC, y nos gustan las opiniones expresadas en LAC.

92. Ensamblaje Web

Evaluar

WebAssembly (WASM) es el estándar W3C que provee capacidades para ejecutar código en el navegador. Soportado por todos los principales navegadores y compatible con sus versiones anteriores, es un formato de compilación binaria diseñado para ejecutarse en el navegador a velocidades casi nativas. Abre el rango de idiomas que se puede utilizar para escribir funcionalidades de front-end, con enfoque inicial en C, C++ y Rust, y es también un objetivo de **LLVM compilation** tCuando se ejecuta en el sandbox, puede interactuar con JavaScript y compartir los mismos permisos y modelo de seguridad. Portabilidad y seguridad son capacidades clave, que habilitarán la mayoría de las plataformas, incluyendo mobile y IoT.

93. Zig

Evaluar

Zig es un nuevo lenguaje que comparte muchos atributos con C pero con un tipado más fuerte, asignación de memoria más fácil, soporte para espacios de nombres y una serie de otras características. Su sintaxis, sin embargo, recuerda a JavaScript más que a C, lo que algunos pueden reprocharle. El objetivo de Zig es proporcionar un lenguaje muy simple con una compilación sencilla que minimice efectos laterales y produzca una ejecución predecible y fácil de seguir. Zig también proporciona acceso a la capacidad de **compilación cruzada**. de LLVM. Algunas de nuestras desarrolladoras han encontrado esta característica tan útil, que están utilizando Zig como un compilador cruzado incluso aunque no estén escribiendo código en Zig. Zig es un lenguaje novedoso e interesante de investigar para aplicaciones donde C está siendo considerado o ya está en uso, así como para aplicaciones de sistema de bajo nivel que requieren manipulación explícita de memoria.

¿Quieres estar al tanto de todas las noticias e insights relacionadas al Radar?

Síguenos en tu red social favorita o suscríbete.

Suscríbete ahora

Thoughtworks es una consultora global de tecnología que integra estrategia, diseño e ingeniería para impulsar la innovación digital. Somos 10,000+ personas en 49 oficinas en 17 países. En los últimos 25+ años, hemos logrado un impacto extraordinario junto con nuestros clientes, ayudándoles a resolver problemas complejos de negocio a través de la tecnología como elemento diferenciador.

/thoughtworks