ThoughtWorks®

TECHNOLOGY RADAR

Una guía con opiniones sobre las tecnologías de vanguardia

Vol.22

thoughtworks.com/es/radar

#TWTechRadar

Contribuyentes

El Radar Tecnológico está preparado por la Junta Asesora de Tecnología de ThoughtWorks

La Junta Asesora de Tecnología (TAB) es un grupo de alrededor de 20 tecnologistas senior de ThoughtWorks. La TAB se reúne presencialmente dos veces al año y bi-semanalmente por teléfono. Su rol principal es ser un grupo de asesores que apoyen a Rebecca Parsons, CTO de ThoughtWorks.

La TAB actúa como un solo individuo que puede analizar temas que afectan la tecnología y a tecnologistas de ThoughtWorks. Usualmente creamos el Radar en encuentros presenciales, pero dada la pandemia global que hemos estado viviendo, este es el primer Radar Tecnológico que se crea a través de un evento virtual.


Rebecca Parsons (CTO)


Martin Fowler (Chief Scientist)


Bharani Subramaniam


Birgitta Böckeler


Camilla Crispim


Erik Dörnenburg


Evan Bottcher


Fausto de la Torre


Hao Xu


Ian Cartwright


James Lewis


Jonny LeRoy


Lakshminarasimhan Sudarshan


Mike Mason


Neal Ford


Ni Wang


Rachel Laycock


Scott Shaw


Shangqi Liu


Zhamak Dehghani

Equipo de traducción al español: Marcela Ramírez, Jorge Agudo, Cecilia Barudi, Manu Escudero, Luis Azcona, Jonathan Fernández Román, Zully Arellano, Abel Guillén, Eduardo Winpenny, Jesús Cardenal Escribano, Ricardo Rodríguez, Yago Pereiro, Elizabeth Velasco, David Montaño, Mateo Rojas, Alexandra Granda, Diana Suasnavas, Pamela Guamán, David Corrales, Mayfe Yépez, Byron Torres, Tex Albuja, Juan Carlos Flores, Carlos Oquendo, Slin Castro, Javiera Laso, Gonzalo Rodríguez, Piero Divasto Martinez, Araceli Correa, Nico Singh, Marcelo Morales, Katherine Ayala, Natalia Rivera, María José Lalama, Fausto de la Torre, Alexandre Goedert y Chris Ford.

Sobre el Radar

Nuestros Thoughtworkers son apasionados por la tecnología. La construimos, investigamos, probamos, liberamos su código fuente, escribimos sobre ella y constantemente queremos mejorarla para todas las personas. Nuestra misión es liderar la excelencia tecnológica y revolucionar las TI.
En soporte de estamisión, creamos y compartimos el Radar Tecnológico de ThoughtWorks. La Junta Asesora de Tecnología de ThoughtWorks, un grupo de líderes senior en la tecnología dentro de ThoughtWorks, son quienes crean el Radar.
Se reunen regularmente para discutir la estrategia tecnológica global de ThoughtWorks y las tendencias tecnológicasque impactan significantemente nuestra industria.

El Radar captura los resultados de las discusiones de la Junta Asesora de Tecnología en un formato que provee valor a un amplio rango de personas interesadas, desde gente desarrolladora hasta CTOs Elcontenido pretende ser un resumen consiso.

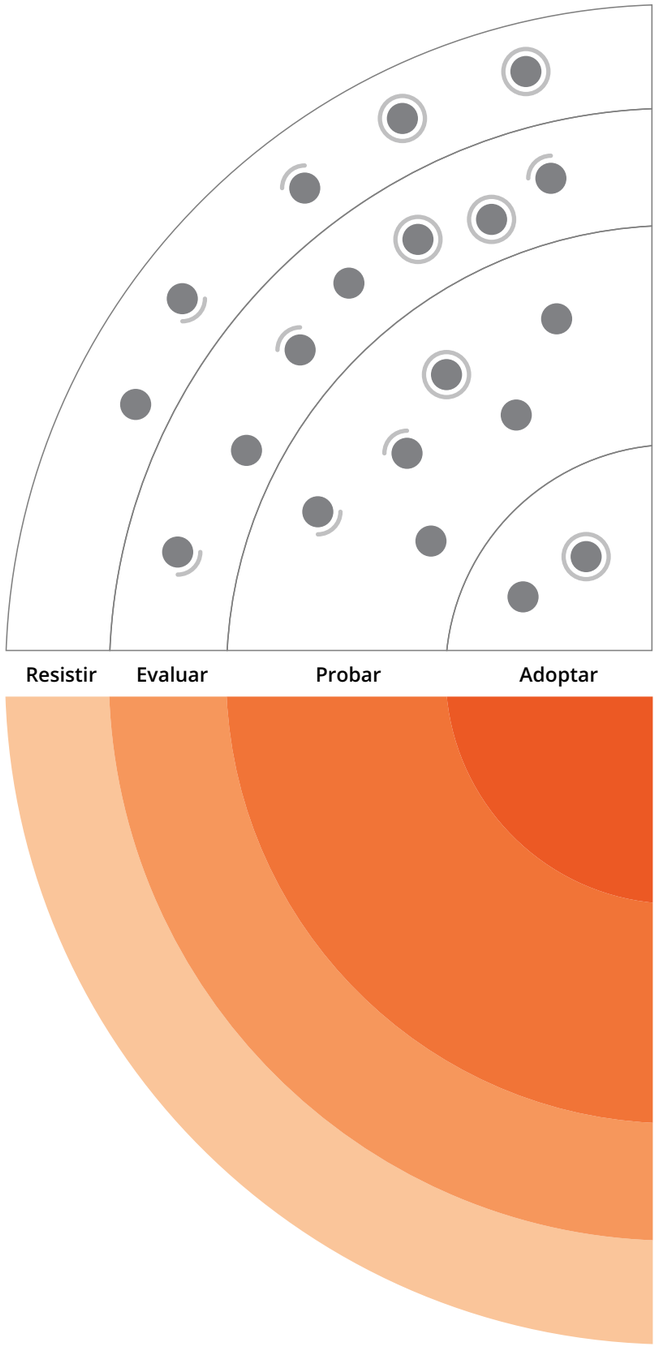
Te alentamos a explorar estas tecnologías. El Radar es gráfico por naturaleza, agrupando items en técnicas, herramientas, plataformas y lenguajes & frameworks. Cuando items del Radar llegan a aparecer en múltiples cuadrantes, elegimos el que parezca más apropiado. Después agrupamos estos items en cuatro anillos para reflejar nuestra opinión actual sobre ellos.

Para más información del Radar, entra en thoughtworks.com/es/radar/faq.

Un vistazo al Radar

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

Un blip es la tecnología o técnica que juega un rol en el desarollo de software. Los blips son cosas que están en constante movimiento — es decir, su posición en el Radar está cambiando — generalmente indicando que estamos encontrando una creciente confianza en ellos a medida que avanzan por los anillos.



Nuevo

Desplazado adentro/afuera

Ningún cambio

Nuestro Radar tiene una visión hacia el futuro. Para hacer espacio a nuevos items, hemos retirado los items que no han sufrido cambios recientes, lo cual no es un reflejo de su valor sino más bien del espacio limitado disponible en nuestro Radar.

Adoptar

Estamos convencidos de que la industria debería adoptar estos ítems. Nosotros los utilizamos cuando es apropiado para 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 la comprensión de cómo podría afectar a su empresa.

Resistir

Proceder con precuación.

Temas de esta edición

El elefante en el Zoom

"La necesidad es la madre de la invención"
— Proverbio

Muchas compañías han experimentado con el concepto del trabajo remoto ya que las tecnologías que lo facilitan han madurado. Pero de repente, una pandemia global ha forzado a negocios de todo el mundo a cambiar su filosofía de trabajo, rápidamente y desde sus cimientos, para seguir siendo productivos. Como muchos han observado "trabajar desde casa" es destacablemente distinto a "ser forzado a trabajar desde casa durante una pandemia" y creemos que hay mucho camino por recorrer hasta que seamos plenamente productivos en este nuevo contexto.

Nunca se nos ocurrió que crear un Radar en remoto fuera posible, y aquí estamos: este es el primer Radar, en su historia, que producimos sin reunirnos en persona. Muchos de los blips propuestos trataban de la urgente necesidad de posibilitar una colaboración remota de primera. No queríamos ignorar al elefante en la habitación y desentendernos de la crisis, pero lograr colaborar de forma efectiva en remoto es una tema extenso y que presenta dificultades y sin duda no todos nuestros consejos encajarían en el formato del Radar. Por ello, acompañando a esta edición encontrarás un podcast donde discutimos nuestras vivencias durante la creación del Radar de forma remota, un artículo escrito con experiencias incluyendo consejos para la productividad enfocada al remoto, un webinar que cubre estrategias tecnológicas para una crisis, y enlaces a otros materiales de ThoughtWorks, incluyendo nuestra libreta para el trabajo remoto. Esperamos que estos, junto con otros recursos disponibles en internet, sean de ayuda para las organizaciones que intentan navegar en estas aguas inciertas.

X también es Software

Con frecuencia alentamos a que otras partes del ecosistema de entrega de software adopten prácticas de ingeniería beneficiosas, en las que los equipos de desarrollo ágil han sido pioneros, y volvemos a este tema a menudo porque siguen apareciendo nichos donde vemos poco progreso hacia estas recomendaciones. Para este Radar, hemos decidido enfocarnos una vez más en infraestructura como código y en pipelines como código, y también hemos tenido conversaciones sobre configuraciones de infraestructura, pipelines de ML y otras áreas relacionadas. Encontramos que los equipos responsables de estas áreas no adoptan comúnmente prácticas de ingeniería duraderas, como aplicar principios de diseño de software, automatización, integración continua o pruebas, entre otras. Entendemos que hay muchos factores que dificultan moverse rápidamente hacia algunas prácticas de ingeniería: complejidad (esencial y accidental), falta de conocimiento, impedimentos políticos, falta de herramientas adecuadas, etc. Sin embargo, los beneficios para las organizaciones que adoptan las prácticas ágiles de entrega de software son claras y vale la pena invertir algún esfuerzo para conseguirlas.

Perspectivas de datos que Maduran y se Expanden

Un tema que apareció en varios de los *blips* y cuadrantes en esta edición, hace mención a la madurez en los datos, particularmente las técnicas y herramientas que rodean los datos analíticos y el aprendizaje automático. Nosotros observamos muchas innovaciones continuas en el espacio de procesamiento del lenguaje natural (PNL). También acogemos con beneplácito la aparición y la madurez continuada de los conjuntos de herramientas de aprendizaje automático de ciclo de vida completo, combinando prácticas de ingeniería duraderas con combinaciones de herramientas que funcionan bien de manera iterativa, lo que demuestra que "el aprendizaje automático también es software". Finalmente, para las arquitecturas distribuidas, como los microservicios, vemos un gran interés en la malla de datos como una forma efectiva de servir y utilizar datos analíticos a escala en sistemas distribuidos. A medida que la industria piensa más diligentemente acerca de cómo deberían funcionar los datos en los sistemas modernos, la dirección general y las perspectivas de apertura en este campo nos alientan y esperamos ver innovaciones emocionantes en el futuro cercano.

Explosión Cámbrica de Kubernetes & Co.

A medida que Kubernetes continúa consolidando su dominio del mercado, prospera el inevitable ecosistema de soporte. Discutimos una serie de puntos que rodean a Kubernetes en los cuadrantes de herramientas, plataformas y técnicas, mostrando cuán generalizado se ha vuelto este tema. Por ejemplo, Lens y k9s simplifican la administración de los *clusters*, kind ayuda con las pruebas locales y Gloo ofrece una *API Gateway* alternativa. Hydra es un servidor OAuth optimizado para ejecutarse en Kubernetes, y Argo CD utiliza la administración del estado deseado nativo de Kubernetes para implementar un servidor de entrega continua. Estos desarrollos indican que Kubernetes está perfectamente preparado para crear un ecosistema de apoyo; es decir, ofrece capacidades críticas pero con abstracciones que a menudo son de muy bajo nivel o avanzadas para la mayoría de los usuarios. Por lo tanto, el vacío de complejidad se llena de herramientas para facilitar la configuración y el uso de Kubernetes o proporcionar algo que falta en la funcionalidad central. A la par que Kubernetes sigue dominando, vemos crecer y expandirse a un rico ecosistema que permite aprovechar sus fortalezas y encarar sus debilidades. Con la madurez de este ecosistema, esperamos su evolución hacia un nuevo conjunto de abstracciones de alto nivel que ofrezcan los beneficios de Kubernetes sin la desconcertante gama de opciones.

El Radar

Técnicas

Adoptar
1. Aplicar gestión de producto a plataformas internas
2. Infraestructura como código
3. Micro frontends
4. Pipelines como código
5. Programación en pares remota con pragmatismo
6. Usar Feature toggles lo más simple posible

Probar
7. Entrega contínua para aprendizaje automático (CD4ML)
8. Pruebas de sesgo ético
9. GraphQL para agregar recursos del lado del servidor
10. Micro frontends para aplicaciones móviles
11. Equipos de producto de ingeniería de plataformas
12. Política de seguridad como código
13. Ciclos de aprendizaje semi supervisados
14. Transferencia de aprendizaje para PNL
15. Uso de procesos y enfoques “nativos remotos”
16. Arquitectura de confianza cero (ZTA)

Evaluar
17. Malla de datos
18. Identidad descentralizada
19. Definición declarativa de pipelines de datos
20. DeepWalk
21. Gestión de sistemas con estado mediante orquestación de contenedores
22. Compilaciones preliminares

Resitir
23. Trasladarse a la Nube
24. Paridad de funcionalidades en migraciones heredadas
25. Agregación de registros para análisis de negocios
26. Ramas de larga duración con Gitflow
27. Usar solo pruebas basadas en capturas de pantalla

Plataformas

Adoptar
28. .NET Core
29. Istio

Probar
30. Anka
31. Argo CD
32. Crowdin
33. eBPF
34. Firebase
35. Hot Chocolate
36. Hydra
37. OpenTelemetry
38. Snowflake

Evaluar
39. Anthos
40. Apache Pulsar
41. Cosmos
42. Google BigQuery ML
43. JupyterLab
44. Marquez
45. Matomo
46. MeiliSearch
47. Stratos
48. Trillian

Resistir
49. Node en exceso

Nuevo · Desplazado adentro/afuera · Ningún cambio

Herramientas

Adoptar
50. Cypress
51. Figma

Probar
52. Dojo
53. DVC
54. Herramientas de seguimiento de experimentos para aprendizaje automático
55. Goss
56. Jaeger
57. k9s
58. kind
59. mkcert
60. MURAL
61. Open Policy Agent (OPA)
62. Optimal Workshop
63. Phrase
64. ScoutSuite
65. Herramientas de pruebas de regresión visual
66. Visual Studio Live Share

Evaluar
67. Apache Superset
68. AsyncAPI
69. ConfigCat
70. Gitpod
71. Gloo
72. Lens
73. Manifold
74. Sizzy
75. Snowpack
76. tfsec

Resistir

Languajes & Frameworks

Adoptar
77. React Hooks
78. Biblioteca de Pruebas React
79. Vue.js

Probar
80. CSS-in-JS
81. Exposed
82. GraphQL Inspector
83. Karate
84. Koin
85. NestJS
86. PyTorch
87. Rust
88. Sarama
89. SwiftUI

Evaluar
90. Clinic.js Bubbleprof
91. Deequ
92. ERNIE
93. MediaPipe
94. Tailwind CSS
95. Tamer
96. Wire
97. XState

Resistir
98. Enzyme

RADAR TECNOLÓGICO Vol. 22

Ténicas

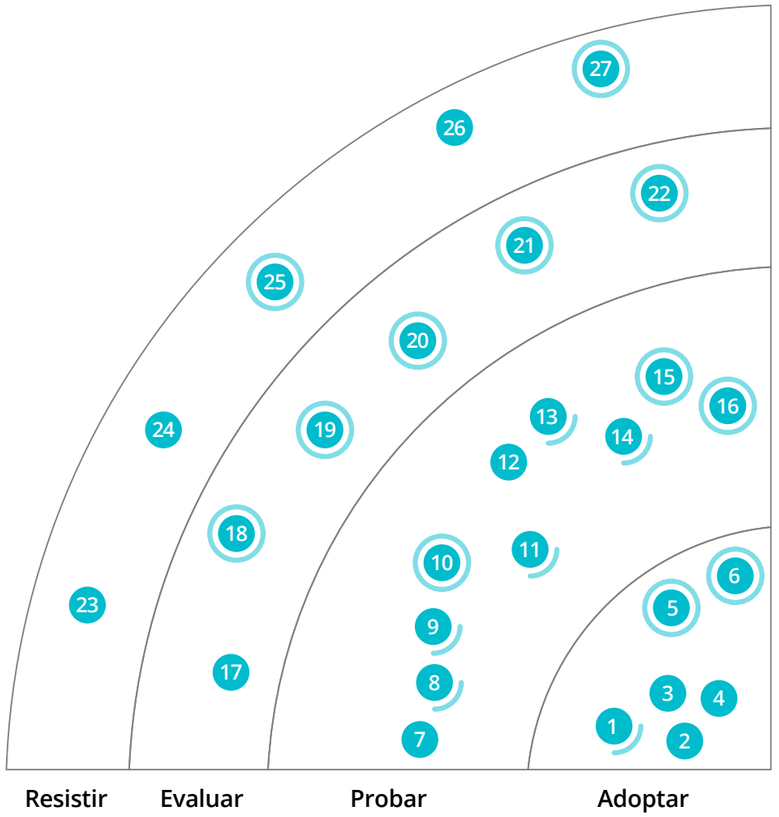
Técnicas

Aplicar gestión de producto a plataformas internas
Adoptar

Cada vez más empresas están construyendo plataformas internas para desplegar nuevas soluciones digitales de forma rápida y eficiente. Las compañías que tienen éxito con esta estrategia están aplicando técnicas de gestión de producto a plataformas internas. Esto significa establecer empatía con los clientes internos (los equipos de desarrollo) y colaborar con ellos en el diseño. Los gerentes de producto de plataforma crean hojas de ruta y se aseguran que la plataforma entrega valor al negocio y mejora la experiencia de los interesados. Desafortunadamente, estamos viendo estrategias menos exitosas, donde los equipos crean plataformas en el vacío, basándose en supuestos no verificados y sin clientes internos. Estas plataformas, a menudo, y a pesar de agresivas tácticas internas, terminan por ser subutilizadas y reducen la capacidad de entrega de la empresa. Como es habitual, una buena gestión de producto trata de conseguir construir productos que encantan a los consumidores.

Infraestructura como código
Adoptar

Aunque la infraestructura como código es una técnica relativamente antigua (la expusimos en el Radar en el 2011) se ha vuelto vitalmente importante en la era moderna de la nube, donde el acto de configurar la infraestructura se ha convertido en el paso de instrucciones de configuración a una plataforma en la nube. Cuando decimos "como código", significa que todas las buenas prácticas que hemos aprendido en el mundo del software deberían ser aplicadas a la infraestructura. El uso del control de versiones, adherirnos al principio DRY, modularización, mantenibilidad, y el uso de pruebas y despliegues automatizados son todas prácticas críticas. Aquellos de nosotros con fuerte trasfondo de software e infraestructura necesitamos empatizar y ayudar a los colegas que no lo tengan. Decir "tratemos la infraestructura como código" no es suficiente, necesitamos asegurarnos de que los aprendizajes que nos costaron tanto obtener del mundo del software sean también aplicados consistentemente a lo largo del dominio de la infraestructura.



Micro frontends
Adoptar

Hemos visto beneficios significativos al utilizar microservicios, que han permitido a los equipos escalar la entrega de servicios desplegados y mantenidos de manera independiente. Desafortunadamente, también hemos visto a equipos crear *frontends* monolíticos (aplicaciones web grandes y complejas sobre servicios del *backend*) que neutralizan en gran parte los beneficios de los microservicios. Los micro *frontends* continúan ganando popularidad desde que se introdujeron por primera vez. Hemos visto a muchos equipos adoptar de una u otra forma esta arquitectura para manejar la complejidad de múltiples personas

Adoptar
1. Aplicar gestión de producto a plataformas internas
2. Infraestructura como código
3. Micro frontends
4. Pipelines como código
5. Programación en pares remota con pragmatismo
6. Usar Feature toggles lo más simple posible

Probar
7. Entrega contínua para aprendizaje automático (CD4ML)
8. Pruebas de sesgo ético
9. GraphQL para agregar recursos del lado del servidor
10. Micro frontends para aplicaciones móviles
11. Equipos de producto de ingeniería de plataformas
12. Política de seguridad como código
13. Ciclos de aprendizaje semi supervisados
14. Transferencia de aprendizaje para PNL
15. Uso de procesos y enfoques "nativos remotos"
16. Arquitectura de confianza cero (ZTA)

Evaluar
17. Malla de datos
18. Identidad descentralizada
19. Definición declarativa de pipelines de datos
20. DeepWalk
21. Gestión de sistemas con estado mediante orquestación de contenedores
22. Compilaciones preliminares

Resitir
23. Levantar y cambiar a la Nube
24. Paridad de funcionalidades en migraciones heredadas
25. Agregación de registros para análisis de negocios
26. Ramas de larga duración con Gitflow
27. Usar solo pruebas basadas en capturas de pantalla

Técnicas

Creemos firmemente en la programación en pares. Pero en un mundo de trabajo remoto post COVID-19, la programación en pares exitosa requiere una dosis saludable de pragmatismo para ser efectivo.

(Programación en pares remota con pragmatismo)

Te alentamos a usar los feature toggles más simples posibles en vez de usar herramientas innecesariamente complejas.

(Usar feature toggles lo más simple posible)

desarrolladoras y equipos contribuyendo en la misma experiencia de usuario. En junio del año pasado, uno de los autores de esta técnica publicó un artículo introductorio que sirve de referencia para los micro *frontends*. Este artículo muestra cómo se puede implementar este estilo utilizando varios mecanismos de programación web e indica un ejemplo usando React.js. Confiamos en que este estilo crecerá en popularidad a medida que las organizaciones dividan el desarrollo de interfaces de usuario entre varios equipos.

Pipelines como código

Adoptar

La técnica de Pipelines como código enfatiza que la configuración de los pipelines de entrega que se encargan de construir, probar y desplegar nuestras aplicaciones o infraestructura deben crearse como código; deben colocarse bajo control de código fuente y modularizarse en componentes reutilizables con pruebas y despliegue automatizados. A medida que las organizaciones se trasladan a la creación de equipos autónomos descentralizados microservicios o micro frontends, La necesidad de prácticas de ingeniería en la gestión de pipelines como código aumenta para mantener la construcción y la implementación de software coherente dentro de la organización. Esta necesidad ha dado lugar a la entrega de plantillas de pipelines y herramientas que permitan una forma estandarizada de construir, desplegar servicios y aplicaciones. Dichas herramientas utilizan *pipelines de entrega declarativos* de aplicaciones adoptando un plan de pipelines para ejecutar las tareas subyacentes y las diversas etapas de un ciclo de vida de entrega, como compilación, prueba y despliegue; y que además abstraen los detalles de implementación. La capacidad de construir, probar y desplegar pipelines como código debería ser uno de los criterios de evaluación para elegir una herramienta de CI/CD.

Programación en pares remota con pragmatismo

Adoptar

Estamos convencidos que la programación en pares mejora la calidad del código, difunde el conocimiento entre el equipo y sobre todo permite una entrega de software más rápida. Sin embargo, en un mundo post COVID-19, muchos equipos de desarrollo de software se encontrarán geográficamente distribuidos o completamente remotos. En esta situación recomendamos trabajar en pares de forma remota y pragmática : ajustar las prácticas de trabajo en pares lo mejor posible con las herramientas disponibles a mano. Considera a herramientas como Visual Studio Live Share para colaborar eficientemente y con baja latencia. Recurre a la compartición de pantalla si ambos participantes residen en una zona geográficamente próxima y disponen de conexiones de internet suficientemente buenas. Emplea la programación en pares entre personas que se encuentren en zonas horarias similares en vez de esperar que esta técnica funcione para todas las personas sin importar su ubicación. Si el trabajo en pares no funciona debido a razones logísticas, recurre a prácticas tales como la programación individual aumentada con revisiones de código, colaboración a través de *pull-requests* (pero evitando tener ramas de larga duración con Gitflow) o ten sesiones muy cortas de trabajo en pares para hacer frente a secciones críticas del programa. Nos hemos dedicado al trabajo en pares remotos por ya bastante tiempo y hemos encontrado que es efectivo si se realiza con una dosis de pragmatismo.

Usar feature toggles lo más simple posible

Adoptar

Desafortunadamente, los *feature toggles* son menos comunes de lo que nos gustaría, y, con frecuencia, vemos personas mezclando sus tipos y casos de uso. Es bastante común encontrarse con equipos que utilizan plataformas muy pesadas como LaunchDarkly para implementarlos, inclusive *release toggles*, para beneficiarse de la integración continua, cuando todo lo que necesitan son condiciones if/else. Por lo tanto, a menos que necesites realizar pruebas A/B o *canary releases* o entregar la responsabilidad de la liberación de funcionalidades a la gente de negocios, te recomendamos usar los *feature toggles* más simples posibles en vez de usar herramientas innecesariamente complejas.

Entrega contínua para aprendizaje automático (CD4ML)

Probar

Aplicar aprendizaje automático para hacer inteligentes a las aplicaciones y servicios de negocio no es solo entrenar modelos y servirlos. Requiere implementar, de principio a fin y continuamente, ciclos de entrenamiento, pruebas, despliegues, monitoreo y operación de los modelos. La entrega continua para aprendizaje automático (CD4ML) es una técnica que habilita la realización de ciclos fiables, de principio a fin, de desarrollo, despliegue y monitoreo de modelos de aprendizaje automático. La tecnología subyacente sobre la que se sustenta CD4ML incluye herramientas para el acceso y el descubrimiento de datos, control de versiones de artefactos (como los datos, el modelo y el código), *pipelines* de entrega continua, aprovisionamiento automatizado de entornos para distintas entregas y experimentos, evaluación y seguimiento del rendimiento del modelo y observabilidad operacional del modelo. Las compañías pueden elegir su propio conjunto de herramientas dependiendo de su *stack* tecnológico actual. CD4ML se focaliza en la automatización y la eliminación de traspasos manuales. CD4ML es nuestro

enfoque de facto para desarrollar modelos de ML

Pruebas de sesgo ético

Probar

En el último año, hemos visto un cambio de interés acerca del machine-learning y en redes neuronales profundas en particular. Hasta ahora, el desarrollo de herramientas y técnicas ha sido impulsado por por el entusiasmo acerca de las extraordinarias capacidades de estos modelos. Actualmente, sin embargo, hay una preocupación creciente de que estos modelos puedan causar daño accidentalmente. Por ejemplo, un modelo podría ser entrenado para, inadvertidamente, tomar decisiones de crédito rentables, simplemente excluyendo aspirantes desfavorecidos. Afortunadamente, estamos viendo un creciente interés en pruebas de sesgo ético que pueden ayudar a descubrir decisiones potencialmente dañinas. Herramientas como lime, AI Fairness 360 o What-If Tool pueden ayudar a descubrir imprecisiones que puedan resultar en grupos de baja representados en datos de entrenamiento, y herramientas de visualización como Google Facets o Facets Dive puede ser usada para descubrir subgrupos dentro de un cuerpo de datos de entrenamiento. Hemos usado lime (local interpretable model-agnostic explanations) adicionalmente a estas técnicas para poder entender las predicciones de cualquier clasificador de aprendizaje automático y que están haciendo estos clasificadores (o modelos).

GraphQL para agregar recursos del lado del servidor

Probar

Vemos cada vez más y más herramientas, como Apollo Federation, que pueden agregar múltiples endpoints de GraphQL en un solo grafo. Sin embargo, alertamos contra el mal uso de GraphQL, especialmente cuando se transforma en un protocolo sevidor-a-servidor. Nuestra experiencia es usar solamente GraphQL para agregar recursos del lado del servidor. Cuando se usa este patrón, los microservicios continúan exponiendo APIs RESTful bien definidas mientras que, por detrás, los servicios agregados o el uso del patron BFF (Backend for Frontends) usan los controladores de GraphQL como la implementación para juntar recursos de otros servicios. La composición del grafo está dirigida por ejercicios de modelado del dominio para asegurar que el lenguaje ubicuo se limita a subgrafos solo donde es necesario (en el caso de un microservicio por "bounded context"). Esta técnica simplifica la implementación interna de agregar servicios o BFFs, mientras que fomenta el uso de un buen modelado de los servicios para evitar REST anémico.

Micro frontends para aplicaciones móviles

Probar

Desde su introducción en el Radar en 2016, hemos visto una adopción generalizada de micro frontends para interfaces de usuario web. Recientemente, hemos visto proyectos que amplían este estilo arquitectónico para incluir también micro frontends para aplicaciones móviles. Cuando la aplicación se vuelve lo suficientemente grande y compleja, se hace necesario distribuir el desarrollo entre múltiples equipos. Esto presenta el desafío de mantener la autonomía del equipo mientras se integra su trabajo en una sola aplicación. Aunque hemos visto equipos que escriben sus propios frameworks para habilitar este estilo de desarrollo, los frameworks de modularización existentes como Atlas y Beehive también pueden simplificar el problema de integrar el desarrollo de aplicaciones con múltiples equipos.

Equipos de producto de ingeniería de plataformas

Probar

La adopción de la nube y de DevOps, al tiempo que aumentan la productividad de los equipos, que ahora pueden moverse más rápido y con menos dependencias hacia los equipos de operaciones centralizados y a la infraestructura, también ha limitado a los equipos que no tienen la habilidad de autogestionar una aplicación completa y el juego de operaciones. Algunas organizaciones han abordado este desafío creando equipos de producto de ingeniería de plataforma. Estos equipos mantienen una plataforma interna que les permite a los equipos de entrega implementar y operar sistemas en menos tiempo y con herramientas más sencillas. El énfasis está en el autoservicio basado en APIs y en las herramientas de soporte, manteniendo a los equipos de entrega responsables de soportar lo que desplieguen en la plataforma. Las organizaciones que consideran establecer un equipo de plataforma de este tipo deben tener cuidado de no crear accidentalmente un equipo de DevOps separado, ni tampoco simplemente renombrar su estructura existente de alojamiento y operaciones a plataforma. Si te preguntas cuál es la mejor forma de configurar los equipos de plataforma, hemos estado usando los conceptos de topologías de equipo para dividir los equipos de plataforma de nuestros proyectos en equipos de habilitación, equipos centrales de "plataforma dentro de una plataforma" y equipos enfocados en iniciativas.

Política de seguridad como código

Probar

Las políticas de seguridad son reglas y procedimientos que protegen a nuestros sistemas de amenazas e interrupciones. Por

Técnicas

Existe una creciente preocupación de que algunos modelos de aprendizaje automático puedan causar daños involuntarios. Afortunadamente, estamos viendo un creciente interés en las pruebas de sesgo ético que ayudarán a descubrir decisiones potencialmente dañinas.

(Pruebas de sesgo ético)

Micro frontends se han adoptado ampliamente para las interfaces de usuario web. Ahora estamos viendo que este estilo arquitectónico también llega a mobile.

(Micro frontends para aplicaciones móviles)

Técnicas

La srquitectura de confianza cero enfatiza la importancia de asegurar todo el acceso y las comunicaciones y hacer cumplir las políticas como código basado privilegios de acceso temporal.

(Arquitectura de confianza cero (ZTA))

ejemplo, las políticas de control de acceso definen y resguardan quiénes pueden acceder a qué tipo de servicios y recursos, y bajo qué circunstancias; o las políticas de seguridad de redes pueden limitar dinámicamente la velocidad del tráfico a un servicio en particular. El complejo panorama tecnológico de hoy exige tratar las es decir, definirlas y mantenerlas bajo control de versiones, validarlas, desplegarlas automáticamente y monitorear su desempeño. Herramientas tales como el Open Policy Agento o plataformas como Istio proveen mecanismos flexibles de definición y ejecución de políticas que apoyan la práctica de tratar las políticas de seguridad como código.

Ciclos de aprendizaje semi supervisados

Probar

Los ciclos de aprendizaje semi supervisados son un tipo de flujo de trabajo iterativo de aprendizaje automático (*machine-learning*) que aprovechan las relaciones existentes entre los datos sin etiquetar. Estas técnicas pueden mejorar los modelos mediante la combinación de conjuntos de datos etiquetados y no etiquetados de varias maneras. En otros casos, comparan modelos entrenados con diferentes subconjuntos de los datos. A diferencia del aprendizaje no supervisado donde una máquina infiere clases a partir de datos no etiquetados o del aprendizaje supervisado donde los datos de entrenamiento están completamente etiquetados, las técnicas de aprendizaje semi supervisado toman ventaja de un pequeño conjunto de datos etiquetados y un conjunto mucho más grande de datos no etiquetados. Además, el aprendizaje semi supervisado está estrechamente relacionado con técnicas de aprendizaje activas, donde un humano es dirigido a etiquetar selectivamente datos ambiguos. Ya que los humanos expertos que puedan etiquetar datos con exactitud son un recurso escaso, y que etiquetar los datos es usualmente la tarea que requiere más tiempo en el flujo de trabajo del aprendizaje automático, las técnicas de aprendizaje semi supervisado reducen el coste del entrenamiento y hacen el aprendizaje automático viable para una nueva clase de usuarios. También observamos la aplicación de técnicas de aprendizaje débilmente supervisadas donde datos etiquetados por una máquina son usados, pero se confía menos en ellos que los datos etiquetados por un humano.

Transferencia de aprendizaje para PNL

Probar

Anteriormente habíamos puesto esta técnica en Evaluar. Las innovaciones en el panorama de la PNL continúan a un ritmo acelerado y podemos aprovecharlas en nuestros proyectos gracias a la ubicua transferencia de aprendizaje para PNL. Se ha visto un progreso significativo en los puntajes de referencia de GLUE (un conjunto de tareas de comprensión del lenguaje) durante los últimos años, con puntajes promedio que pasaban de 70.0 en un inicio y sobrepasando los 90.0 en abril de 2020 para algunos de los líderes. Muchos de nuestros proyectos en los dominios de PNL pueden lograr un progreso significativo si empiezan con modelos pre-entrenados de ELMo, BERT y ERNIE, entre otros, y luego afinarlos en función de las necesidades del proyecto.

Uso de procesos y enfoques “nativos remotos”

Probar

Los equipos distribuidos tienen varias formas y configuraciones mientras que los equipos de entrega ubicados al 100% en un entorno y ubicación conjunta, se han convertido en la excepción. La mayoría de nuestros equipos están en varias ubicaciones o tienen al menos algunos miembros trabajando remotamente. Por lo tanto, utilizar procesos y enfoques “nativos remotos” por defecto puede ayudar significativamente en la eficiencia y en el flujo general del equipo. El primer paso es asegurarse de que todas las personas tengan acceso remoto a los sistemas necesarios. Además, el uso de herramientas como Visual Studio Live Share, MURAL o Jamboard convierten los talleres virtuales y el trabajo remoto en pares en rutinas en lugar de ineficaces excepciones. Pero el ser “nativo remoto” va más allá del traslado de prácticas de entornos presenciales al mundo digital: hay que aceptar una comunicación más asíncrona, tener una mayor disciplina alrededor de la documentación de decisiones, e incluso tener reuniones donde “todas las personas están en remoto”. Estos enfoques, que nuestros equipos practican por defecto, sirven para optimizar y permitir fluidez en la ubicación.

Arquitectura de confianza cero (ZTA)

Probar

La realidad tecnológica actual es bastante más compleja para las organizaciones con activos (datos, funciones, infraestructura y usuarios) repartidos a través de varios límites de seguridad como servidores locales, múltiples proveedores de nube y una variedad de servicios SaaS. Esto requiere un cambio de paradigma en la planificación de seguridad a nivel empresarial y arquitectura de sistemas, pasando de un manejo estático de las políticas

de seguridad, basadas en zonas de confianza y configuraciones de red, a la aplicación dinámica de políticas de seguridad detalladas basadas en privilegios de acceso temporal.

La Arquitectura de Confianza Cero (zero trust architecture, ZTA) es una estrategia y un proceso de una organización para implementar principios de seguridad de confianza cero para todos sus activos, como dispositivos, infraestructura, servicios, datos y usuarios, ya que esto incluye la implementación de prácticas como asegurar todo tipo de acceso y comunicación sin importar el lugar de la red, aplicando políticas como código basadas en el menor privilegio y con la mayor granularidad posible, y el monitoreo continuo y la mitigación automatizada de amenazas. Nuestro Radar refleja muchas de las técnicas habilitadoras, tales como políticas de seguridad como código, sidecars para seguridad de endpoints, y BeyondCorp. Si estas en el proceso de la implementación de ZTA, revisa la publicación del NIST sobre ZTA para aprender más sobre principios, componentes de tecnologías habilitadoras y patrones de migración así como las publicaciones de Google sobre BeyondProd.

Malla de datos
Evaluar

La malla de datos (*data mesh*) es un paradigma de arquitectura y de organización que desafía la vieja presunción de que se debe centralizar los grandes datos analíticos para utilizarlos, tener todos los datos en un mismo lugar o gestionarlos a través de un equipo de datos centralizado para entregar valor. Este paradigma afirma que, para que *big data* promueva la innovación, su propiedad debe ser federada entre los dueños de los datos de dominio quienes son responsables de proveer sus datos como productos (con el soporte de una plataforma de datos de autoservicio para abstraer la complejidad técnica que supone servir productos de datos); también se debe adoptar una nueva forma de gobierno federado a través de la automatización que permita la interoperabilidad de los productos de datos orientados a dominios. La descentralización, junto con la interoperabilidad y el enfoque en la experiencia para los consumidores de datos, son clave para la democratización de la innovación usando datos.
Si en la organización existe un gran número de dominios con varios sistemas y equipos generando datos o un conjunto diverso de casos de uso y patrones de acceso basados en datos, sugerimos evaluar la malla de datos. La implementación de este paradigma requiere invertir en la construcción de una plataforma de datos de autoservicio y aceptar y promover un cambio organizacional para que los dominios tomen la propiedad a largo plazo de sus productos de datos, así como una estructura de incentivos que premien a los dominios que sirvan y utilicen datos como producto.

Identidad descentralizada
Evaluar

Desde el nacimiento de la internet, el panorama tecnológico ha experimentado una evolución acelerada hacia la descentralización. Mientras los protocolos como HTTP y los patrones de arquitectura como microservicios o mallas de datos permiten realizar implementaciones descentralizadas, la gestión de la identidad continúa centralizada. Sin embargo, la aparición de tecnologías de libros de transacciones distribuidas (digital ledger technologies, DLT), brinda fundamentos al concepto de identidad descentralizada. En un sistema de este tipo, las entidades (unidades discretas identificables, como personas, organizaciones y cosas) son libres de usar cualquier raíz compartida de confianza.
En contraste, los sistemas convencionales de gestión de la identidad se basan en autoridades y registros centrales como los servicios de directorio corporativo, autoridades de certificación o registros de nombres de dominio.

El desarrollo de identificadores descentralizados (que son únicos globalmente, persistentes y son identificadores auto-soberanos verificables criptográficamente) es un estándar habilitador mayor. A pesar que todavía son pocas las implementaciones a escala de identificadores descentralizados, nos emociona la premisa de este movimiento y hemos empezado a usar el concepto en nuestra arquitectura. Para conocer los más recientes experimentos y colaboraciones de la industria, accede a la *Decentralized Identity Foundation*.

Definición declarativa de pipelines de datos
Evaluar

Muchos pipelines de datos se definen usando secuencias de comandos largas, en forma de código, más o menos, imperativo, escritas en Python o Scala. Estos códigos contienen tanto la lógica de los pasos individuales como la lógica que enlaza los pasos entre sí. Cuando los desarrolladores se enfrentaron a una situación similar con las pruebas de Selenium, descubrieron el patrón “Page Object”, y posteriormente muchos frameworks dirigidos por comportamiento, BDD por sus siglas en inglés, implementaron una división entre definición de los pasos y su composición. Algunos equipos están experimentando con la aplicación del mismo enfoque en ingeniería de datos. Una definición declarativa de pipeline de datos separada, tal vez escrita en YAML, contiene únicamente la declaración y la secuencia de pasos. Declara los tipos de

Técnicas

The foundation of BitCoin — tecnología de contabilidad distribuida (DLT) — está permitiendo el surgimiento de identidades descentralizadas.

(Identidad descentralizada)

Al trabajar sobre conjuntos de datos representados como grafos, uno de los problemas principales es extraer características del grafo. Aquí es donde DeepWalk puede ayudar.

(DeepWalk)

Técnicas

Logs destinados a la observabilidad técnica a menudo son inadecuados para inferir una comprensión profunda del cliente.

(Agregación de registros para análisis de negocios)

datos de entrada y salida pero puede ejecutar comandos imperativos en otros archivos cuando se necesita una lógica más compleja. Con A La Mode, vemos a la primera herramienta de código abierto que aparece en este espacio.

DeepWalk
Evaluar

DeepWalk es un algoritmo que ayuda a aplicar aprendizaje automático en grafos. Al trabajar sobre conjuntos de datos representados como grafos, uno de los problemas principales es extraer características del grafo. Aquí es donde DeepWalk puede ayudar. Usa SkipGram para construir incrustaciones de nodos al ver el grafo como un lenguaje donde cada nodo es una palabra única en el lenguaje y caminos aleatorios de longitud finita sobre el grafo constituyen una oración. Estas incrustaciones pueden entonces ser usadas en varios modelos de aprendizaje automático. DeepWalk es una de las técnicas que estamos probando en algunos de nuestros proyectos donde hemos necesitado aplicar aprendizaje automático en grafos.

Gestión de sistemas con estado mediante orquestación de contenedores
Evaluar

Recomendamos precaución con la gestión de sistemas con estado mediante la orquestación de contenedores en plataformas como Kubernetes. Algunas bases de datos no han sido construidas con soporte nativo para la orquestación (no esperan que un planificador termine el proceso y las reubique en una máquina distinta). La construcción de un servicio con alta disponibilidad sobre estas bases de datos no es sencilla y aún recomendamos ejecutarlas en máquinas dedicadas (bare metal hosts) o una máquina virtual en lugar de obligarlas a ejecutarlas de manera forzada en una plataforma de orquestación de contenedores.

Compilaciones preliminares
Evaluar

Aunque abogamos fervientemente por la integración continua en lugar de Gitflow, sabemos que subir cambios commit directamente al trunk y ejecutar los procesos de integración continua en una rama master puede ser ineficiente si el equipo es demasiado grande, si la construcción es lenta o inconsistente, o si el equipo carece de la disciplina para ejecutar el conjunto completo de pruebas localmente. En esta situación, una compilación fallida puede bloquear a múltiples personas o pares de desarrollo. En lugar de corregir la causa subyacente del problema (compilaciones lentas, incapacidad de ejecutar pruebas localmente o arquitecturas monolíticas que requieren que muchas personas trabajen en la misma área), los equipos generalmente confían en las ramas por funcionalidades (feature branch) para esquivar estos problemas. Desaconsejamos este tipo de ramas, ya que pueden requerir un esfuerzo significativo para resolver conflictos al integrar cambios e introducen ciclos de retroalimentación más largos y posibles errores durante la resolución de conflictos. En su lugar, proponemos utilizar como alternativa: son compilaciones basadas en peticiones de cambios (pull request) para “micro ramas” que existen sólo durante la ejecución del pipeline de compilación y que se abren con cada commit. Para ayudar a automatizar este flujo de trabajo, nos hemos encontrado con bots como Bors que automatizan la integración de cambios con master y la eliminación de la micro rama en caso de que su compilación tenga éxito. Estamos considerando este flujo, y también deberías hacerlo; pero no para resolver el problema incorrecto, ya que puede llevar al mal uso de las ramas y puede causar más daño que beneficio.

Trasladarse a la Nube
Resistir

Es algo curioso, que luego de una década de experiencia de la industria con migraciones a la nube, que siga siendo necesario manifestarse frente a la idea de simplemente trasladarse (*lift-and-shift*) a la nube; donde la nube es vista simplemente como una solución de alojamiento, y la arquitectura existente, las prácticas de seguridad, los modelos de tecnología son simplemente replicados en la nube. Esto no cumple con las promesas de agilismo e innovación digital. Una verdadera migración a la nube requiere cambios intencionales en múltiples ejes para alcanzar un estado nativo de la nube, y dependiendo de circunstancias específicas de los procesos de migración, cada organización puede terminar en algún lugar en el espectro entre “traslado a la nube” y “nativo en la nube”. La arquitectura de sistemas, por ejemplo, es uno de los pilares de la entrega ágil y a menudo requiere ser cambiada. La tentación de simplemente trasladar sistemas existentes a contenedores puede ser muy fuerte. Aún cuando esta estrategia puede acelerar la migración a la nube, se queda corta cuando se trata de crear agilismo y entregar funcionalidades y valor. La seguridad empresarial en la nube es fundamentalmente diferente a la tradicional (seguridad basada en perímetros, típicamente compuesta de zonas y cortafuegos) y exige una transición hacía arquitecturas de cero confianza.

Los modelos operacionales de tecnología tienen que ser renovados para proveer con seguridad servicios en la nube a través de plataformas de autoservicio automatizadas y empoderar a los equipos a tomar más de las responsabilidades operacionales y ganar autonomía. Y último, pero no menos importante, las organizaciones deben construir las bases para habilitar cambios continuos, tales como *pipelines* con pruebas continuas a las aplicaciones e infraestructuras como parte de la migración. Esto ayudará al proceso y tendrá como resultado un sistema más robusto y mejor estructurado, y dará a las organizaciones una forma para continuar evolucionando y mejorando sus sistemas.

Paridad de funcionalidades en migraciones heredadas

Resistir

Hemos descubierto que cada vez más y más organizaciones tienen la necesidad de reemplazar sistemas heredados, para satisfacer las demandas de sus clientes (tanto internos como externos). Un antipatrón que continuamos viendo es la paridad de funcionalidades en migraciones de sistemas heredados, el deseo de retener una paridad de funcionalidad con lo antiguo. Consideramos que esto es una enorme oportunidad perdida. A menudo, los sistemas antiguos se han sobrecargado con el paso del tiempo, con muchas funcionalidades que los usuarios no utilizan (el 50% de acuerdo al reporte de 2014 de Standish Group) y procesos de negocio que han evolucionado con el tiempo. Reemplazar estas funcionalidades es perder el tiempo. Nuestro consejo: convence a tus clientes a dar un paso atrás y entender lo que sus usuarios necesitan actualmente, para luego priorizar estas necesidades en base a resultados comerciales y métricas — esto usualmente es más fácil de decir que de hacer. Esto significa que se debe conducir una investigación de necesidades del usuario y aplicar prácticas modernas de desarrollo de producto en lugar de simplemente reemplazar las existentes.

Agregación de registros para análisis de negocios

Resistir

Hace varios años, surgió una nueva generación de plataformas de agregación de registros (logs) que eran capaces de almacenar y buscar en grandes cantidades de datos de registros para descubrir tendencias y conocimientos sobre datos operativos. Splunk fue el más destacado pero de ninguna manera el único ejemplo de estas herramientas. Debido a que estas plataformas proporcionan una amplia visibilidad operativa y de seguridad en todo el conjunto de aplicaciones, los administradores y desarrolladores se han vuelto cada vez más dependientes de ellas. Este entusiasmo se extendió cuando las partes interesadas descubrieron que podían usar la agregación de registros (*logs*) para análisis de negocios. Sin embargo, las necesidades comerciales pueden superar rápidamente la flexibilidad y la facilidad de uso de estas herramientas. Los registros destinados a la observación técnica a menudo son inadecuados para inferir una comprensión profunda del cliente. Preferimos usar herramientas y métricas diseñadas específicamente para el análisis de los clientes o adoptar un enfoque de observabilidad más orientado a eventos, donde los eventos comerciales y operativos se recopilan y almacenan de manera que puedan reproducirse y procesarse con herramientas especialmente diseñadas para ese propósito.

Ramas de larga duración con Gitflow

Resistir

Hace cinco años ya resaltamos los problemas de las ramas de larga duración con Gitflow. En esencia, este tipo de ramas son lo contrario a integrar de forma continua todos los cambios del código fuente y, en nuestra experiencia, la integración contínua es el mejor enfoque para la mayoría de los proyectos de desarrollo de software. Un tiempo después, extendimos nuestra advertencia al mismo Gitflow porque observamos que algunos equipos lo usaban exclusivamente con ramas de larga duración. Hoy en día, aún hay equipos en entornos donde se tiene como objetivo hacer entrega continua de sistemas para la web, que se ven forzados a usar ramas de larga duración. Por ello, saludamos que el autor de Gitflow haya añadido una nota a su artículo original, explicando que Gitflow no estaba pensado para dicho uso.

Usar solo pruebas basadas en capturas de pantalla

Resistir

El valor de las pruebas basadas en capturas de pantalla es innegable cuando se trabaja con sistemas heredados ya que asegura que el sistema continúa funcionando y que el código heredado no falla. Sin embargo, hemos observado que es común usar como mecanismo principal de pruebas la perjudicial práctica de probar solamente con capturas de pantalla. Este tipo de pruebas validan la salida exacta generada en el DOM por un componente, no su comportamiento. Es por ello que pueden ser frágiles y poco fiables, fomentando la mala práctica de “eliminar la captura y volverla a generar”. En vez de eso, se debería probar la lógica y el comportamiento de los componentes emulando lo que harían los usuarios. Esta mentalidad es la recomendada por herramientas de la familia Testing Library.

Técnicas

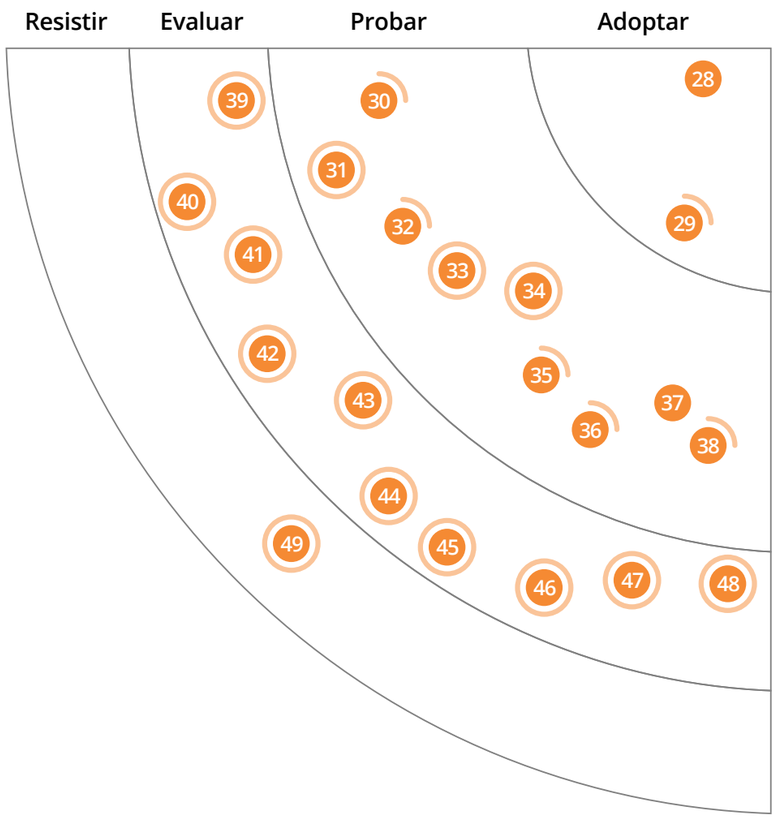
El valor de las pruebas basadas en capturas de pantalla es innegable cuando se trabaja con sistemas heredados. Pero no debería ser el mecanismo de prueba principal para tales sistemas.

(Usar solo pruebas basadas en capturas de pantalla)

RADAR TECNOLÓGICO Vol. 22

Plataformas

Plataformas



.NET Core
Adoptar

Anteriormente teníamos .NET Core en Adoptar, indicando que se ha convertido en la opción predeterminada para proyectos .NET. Sin embargo, creemos que merece la pena resaltarlo nuevamente. Con la versión 3.x de .NET Core liberada el año pasado la mayor parte de las características de .NET Framework han sido migradas a .NET Core. Con el anuncio de que .NET Framework está en su última versión Microsoft ha reforzado la opinión de que .NET Core es el futuro de .NET. Microsoft ha realizado mucho trabajo para hacer .NET Core fácil de usar con contenedores. La mayoría de nuestros proyectos basados en .NET Core apuntan a Linux y a menudo se despliegan en contenedores. La próxima versión de .NET 5 parece prometedora, y estamos deseando que llegue.

Istio
Adoptar

Si estás construyendo y operando una arquitectura de microservicios a escala con Kubernetes, adoptar una malla de servicios para gestionar todos los aspectos transversales del funcionamiento de la arquitectura es una elección segura. Entre las diversas implementaciones de malla de servicios, Istio ha obtenido una adopción mayoritaria. Tiene un amplio conjunto de características, que incluyen descubrimiento de servicios, administración de tráfico, seguridad de servicio a servicio y de origen a servicio, observabilidad (incluyendo telemetría y rastreo distribuido), lanzamiento de nuevas versiones recurrentes y resiliencia. Su experiencia de usuario ha mejorado en las últimas versiones lanzadas, debido a su facilidad de instalación y arquitectura del panel de control. Istio ha bajado la vara en lo que se refiere a la implementación de microservicios a gran escala y con calidad operativa para muchos de nuestros clientes, pero también hay que admitir que la operación de instancias propias de Istio y Kubernetes requiere de una serie de conocimientos y recursos apropiados, lo que, lastimosamente, no es para cualquier persona.

Anka
Probar

Anka es un conjunto de herramientas para crear, gestionar, distribuir, construir y probar ambientes virtuales reproducibles de macOS para iOS y macOS. Provee una

Adoptar
28. .NET Core
29. Istio

Probar
30. Anka
31. Argo CD
32. Crowdin
33. eBPF
34. Firebase
35. Hot Chocolate
36. Hydra
37. OpenTelemetry
38. Snowflake

Evaluar
39. Anthos
40. Apache Pulsar
41. Cosmos
42. Google BigQuery ML
43. JupyterLab
44. Marquez
45. Matomo
46. MeiliSearch
47. Stratos
48. Trillian

Resistir
49. Node en exceso

Plataformas

Aunque esta tecnología no es nueva, es ahora cuando se le puede sacar todo el partido con el incremento en el uso de microservicios desplegados como contenedores orquestados.

(eBPF)

Esta clara separación de identidad del resto del marco de trabajo de OAuth2 hace más fácil integrar a Hydra con un ecosistema de autenticación existente.

(Hydra)

experiencia semejante a la de Docker para los ambientes macOS: arranque instantáneo, una interfaz de línea de comandos (*CLI)* para administrar máquinas virtuales y un registro para versionar y etiquetar máquinas virtuales para distribución. Hemos utilizado Anka para construir una nube privada de ambientes macOS para un cliente. Vale tomar en cuenta a esta herramienta cuando se trata de virtualizar ambientes macOS e iOS.

Argo CD
Probar

Sin juzgar la técnica de GitOps, nos gustaría hablar sobre Argo CD dentro del ámbito de desarrollo y monitoreo de aplicaciones en ambientes Kubernetes. En base a su capacidad para automatizar el despliegue del estado deseado de las aplicaciones en los ambientes de destino en Kubernetes, y por la buena experiencia al solucionar problemas con despliegues fallidos, para verificar registros de eventos y para monitorear el estado de los despliegues, recomendamos que pruebes a Argo CD. Podrás ver en forma gráfica lo que está pasando en el cluster, cómo se propagan los cambios e incluso ver cómo los pods son creados y destruidos en tiempo real.

Crowdin
Probar

La mayoría de proyectos con soporte para múltiples idiomas comienzan con equipos de desarrollo creando funcionalidades en un idioma y manejando el resto a través de traducciones fuera de línea, a través de correos electrónicos y hojas de cálculo. Aunque esta forma simple de trabajo funciona, las cosas pueden salirse de las manos rápidamente. Es posible debas que seguir respondiendo a las mismas preguntas de diferentes traductores, restándole energía a la colaboración entre traductores, correctores y el equipo de desarrollo. Crowdin es una de las pocas plataformas que ayudan a organizar de manera efectiva el flujo de trabajo de internacionalización de tu proyecto. Con Crowdin, el equipo de desarrollo puede continuar creando funcionalidades, mientras la plataforma organiza el texto que necesita ser traducido en un flujo de trabajo en línea. Nos gusta que Crowdin empuja a los equipos para que incorporen traducciones de forma continua e incremental en lugar de administrarlas en grandes lotes al final del desarrollo.

eBPF
Probar

Durante varios años, el núcleo de Linux ha incluido la máquina virtual extendida para el Filtro de Paquetes de Berkeley (eBPF) y ha proporcionado la posibilidad de adjuntar filtros eBPF a sockets concretos. Pero la extensión BPF va más allá del filtrado de paquetes y permite ejecutar scripts personalizados en varios puntos del kernel con muy poca sobrecarga. Aunque esta tecnología no es nueva, es ahora cuando se le puede sacar todo el partido con el incremento en el uso de microservicios desplegados como contenedores orquestados. Las comunicaciones servicio-servicio pueden ser complejas en estos sistemas, haciendo difícil relacionar problemas de latencia o rendimiento con una llamada a una API. Ahora estamos encontrando publicaciones de herramientas con scripts eBPF ya escritos para recolectar y visualizar tráfico de paquetes o informar sobre el uso de la CPU. Con el ascenso de Kubernetes, nos encontramos con una nueva generación de aplicaciones de seguridad e instrumentación basadas en scripts eBPF para dominar la complejidad de grandes despliegues de microservicios.

Firebase
Probar

Google Firebase ha experimentado una evolución significativa desde que lo mencionamos como parte de una arquitectura sin servidor en 2016. Firebase es una plataforma completa para construir aplicaciones móviles y web de una manera compatible con la infraestructura subyacente y escalable de Google. En particular nos gusta la Distribución de Aplicaciones de Firebase, que facilita publicar versiones de prueba de una aplicación a través de un pipeline de entrega continua, y Configuración Remota de Firebase, que permite que los cambios de configuración se envíen dinámicamente a las aplicaciones sin necesidad de volver a publicarlas.

Hot Chocolate
Probar

El ecosistema y la comunidad de GraphQL siguen creciendo. Hot Chocolate es un servidor de GraphQL para .NET (Core y Clásico). Este permite construir y alojar esquemas para luego servir consultas contra ellos utilizando los mismos componentes base de GraphQL: data *loaders, resolvers*, esquemas, operaciones y tipos. El equipo detrás de Hot Chocolate recientemente agregó la funcionalidad de schema stitching, que permite la existencia de un único punto de entrada para consultar a través de múltiples esquemas agregados desde diferentes ubicaciones. A pesar de la posibilidad de hacer un mal uso de este enfoque, nuestros equipos están contentos con Hot Chocolate porque está

bien documentado y nos permite ofrecer valor rápidamente a nuestros clientes.

Hydra
Probar

No todos necesitan una solución alojada de OAuth2, pero si es así, vale la pena mirar a Hydra, un servidor de código abierto completamente compatible con la especificación OAuth2 y además proveedor de conexiones OpenID. Hydra tiene soporte para almacenamiento en memoria para ambientes de desarrollo y una base de datos relacional (PostgreSQL) para los casos de uso en producción. Hydra no tiene estado y fácilmente escala horizontalmente en plataformas como Kubernetes. Dependiendo de los requerimientos de rendimiento, podría ser necesario afinar el número de instancias de la base de datos al escalar las instancias de Hydra. Y dado que no provee ninguna solución de administración de identidades de manera predeterminada, se puede integrar con cualquier gestor de identidad disponible, a través de una API limpia. Esta clara separación de identidad del resto del marco de trabajo de OAuth2 hace más fácil integrar a Hydra con un ecosistema de autenticación existente.

OpenTelemetry
Probar

OpenTelemetry es un proyecto de código abierto que integra OpenTracing y OpenCensus. El proyecto de OpenTelemetry incluye especificaciones, librerías, agentes y otros componentes requeridos para capturar telemetría desde los servicios para poder observar, mantener y depurarlos de mejor manera. Cubre los tres pilares de la observabilidad (rastreo distribuido, métricas y logs, actualmente en beta) y su especificación conecta estas tres piezas a traves de correlaciones; por lo tanto se pueden usar *métricas* para determinar un problema, localizar los *rastros* correspondientes para descubrir dónde se produjo el problema y finalmente estudiar los *registros de eventos* correspondientes para encontrar la causa raíz. Los componentes de OpenTelemetry pueden conectarse con sistemas de observabilidad en *backend* como Prometheus y Jaeger entre otros. La formación de OpenTracing es un paso positivo hacia la convergencia de estándares y la simplificación de las herramientas.

Snowflake
Probar

Snowflake ha probado ser un SaaS robusto para soluciones de almacenamiento de big data, warehouse, o lago de datos para muchos de nuestros clientes. Tiene una arquitectura superior para escalar almacenamiento, cómputo y servicios para cargar, descargar y utilizar datos. También es muy flexible: soporta almacenamiento de datos estructurados, semi estructurados y sin estructura; provee una lista creciente de conectores para distintos patrones de acceso como Spark para ciencia de datos y SQL para analíticas; y corre en múltiples proveedores de nube. Nuestro consejo para muchos de nuestros clientes es hacer uso de servicios gestionados según la utilidad de su tecnología, como almacenamiento de big data; sin embargo, si el riesgo y las regulaciones prohíben el uso de servicios gestionados, entonces Snowflake es un buen candidato para compañías con grandes volúmenes de datos y cargas pesadas de trabajo. Si bien hemos tenido éxito utilizando Snowflake en empresas medianas, nos falta aún experimentar Snowflake en ecosistemas grandes donde los datos necesitan ser reconocidos como propiedad de varios segmentos de la organización.

Anthos
Evaluar

Observamos un cambio de los planes casuales de migración a la nube híbrida o pura, a sofisticados planes pensados intencionalmente para estrategias híbridas, multi-cloud, o portables, donde las organizaciones aplican principios multidimensionales para establecer y ejecutar su estrategia en la nube: donde albergar sus múltiples datos y activos funcionales basados en riesgo, habilidad de control y perfiles de rendimiento; cómo utilizar sus inversiones en infraestructura on-premise al mismo tiempo que reducen costes operacionales y cómo aprovechar los múltiples proveedores cloud y sus servicios únicos y diferenciados sin crear complejidad y fricción para los usuarios y aplicaciones existentes.

Anthos es la respuesta de Google para habilitar estrategias híbridas y multi-cloud brindando un nivel de control y gestión de alto nivel sobre un conjunto de tecnologías de código abierto como GKE, Service Mesh y una gestión de configuración basada en Git. Posibilita la ejecución de cargas de trabajo portables y otros componentes en diferentes entornos de hosting, incluyendo Google Cloud y hardware on-premises. Mientras otros proveedores cloud tienen ofertas comparativas, Anthos apunta más allá de la nube híbrida, a posibilitar una nube portable usando componentes de código libre, pero todavía está por verse. Vemos un aumento de interés en Athos. Mientras la aproximación

Plataformas

The OpenTelemetry es un proyecto que incluye especificaciones, librerías, agentes y otros componentes requeridos para capturar telemetría desde los servicios para poder observar, mantener y debuggearlos de mejor manera.

(OpenTelemetry)

Anthos es la respuesta de Google para habilitar estrategias híbridas y multi-cloud. Posibilita la ejecución de cargas de trabajo portables y otros componentes en diferentes entornos de hosting, incluyendo Google Cloud y hardware on-premises

(Anthos)

Plataformas

BigQuery posee la barra para usar ML para hacer predicciones y recomendaciones, particularmente para exploraciones rápidas.

(Google BigQuery ML)

de Google a una nube híbrida gestionada parece prometedora, no es una bala de plata y requiere cambios tanto en la nube existente como en los componentes on-premise. Nuestro consejo para nuestros clientes que están considerando Anthos, es buscar un balance seleccionando tanto servicios del ecosistema de Google Cloud como otras opciones, para mantener su nivel de neutralidad y control.

Apache Pulsar
Evaluar

Apache Pulsar es una plataforma de código abierto para la publicación y suscripción de mensajería/streaming, compitiendo en un espacio similar con Apache Kafka. Provee la funcionalidad esperada, como la entrega de mensajes síncrona y asíncrona con baja latencia y almacenamiento escalable y persistente de mensajes, así como varias bibliotecas de clientes. Lo que nos ha llevado a evaluar a Pulsar es su facilidad para escalar, en particular en organizaciones grandes con múltiples segmentos de usuarios. Pulsar soporta nativamente la tenencia múltiple, geo-replicación, control de acceso basado en roles y segregación de facturación. También vemos a Pulsar para resolver el problema de un registro sin fin de mensajes de registros de eventos para nuestros sistemas de datos de alta escala donde se espera que los eventos se persistan indefinidamente y que los suscriptores puedan empezar a consumir mensajes en retrospectiva. Esto se soporta con un modelo de almacenamiento escalonado. Aunque Pulsar es una plataforma prometedora para grandes organizaciones, hay posibilidades de mejora. Su instalación actual requiere administrar ZooKeeper y BookKeeper entre otras piezas de tecnología. Esperamos que con su adopción en aumento, los usuarios puedan pronto contar con un apoyo comunitario más amplio.

Cosmos
Evaluar

El desempeño de la tecnología blockchain ha mejorado ostensiblemente desde que la evaluamos inicialmente en el Radar. Sin embargo, aún no existe ni una sola *blockchain* que haya logrado alcanzar un rendimiento de "nivel de internet". A medida que se han desarrollado varias plataformas *blockchain*, vemos nuevos datos y silos de valor. Esto explica el porqué la tecnología "*cross-chain*" siempre ha sido un tema clave en la comunidad blockchain: el futuro de blockchain puede ser una red de *blockchains* paralelas e independientes. Esta es también la visión de Cosmos. Cosmos publica Tendermint y CosmosSDK para permitir a las personas desarrolladoras configurar blockchains independientes. Estas blockchains paralelas pueden intercambiar valores mediante el protocolo de Inter-Blockchain Communication (IBC) y Peg-Zones. Nuestros equipos han tenido buenas experiencias con CosmosSDK, y el protocolo IBC está madurando. Esta arquitectura puede resolver los problemas de interoperabilidad y escalabilidad de *blockchain.*

Google BigQuery ML
Evaluar

A menudo entrenar y predecir los resultados en modelos de aprendizaje automático (*machine-learning*) requiere código para llevar los datos al modelo. Google BigQuery ML invierte esto llevando el modelo a los datos. Google BigQuery es un almacén de datos diseñados para servir consultas a gran escala usando SQL, para casos de usos analitico. Google BigQuery ML extiende esta funcionalidad y su interfaz SQL para crear, entrenar y evaluar modelos de aprendizaje automático usando sus conjuntos de datos; y eventualmente correr modelos predictivos para crear un nuevo conjunto de datos BigQuery. Soporta un limitado conjunto de modelos out of the box, como regresión lineal para predicción o regresión binaria y multiclase para clasificación. También soporta, con limitada funcionalidad, la importación de modelos de TensorFlow previamente entrenados. Aunque BigQuery ML y su enfoque basado en SQL disminuye la barra para usar machine learning para hacer predicciones y recomendaciones, particularmente para exploración rápida, esto puede ser una desventaja al dejar de lado aspectos del entrenamiento del modelo tal como pruebas de sesgo ético, explicabilidad y entrega continua para modelos de machine learning.

JupyterLab
Evaluar

JupyterLab es la nueva generación de la interfaz de usuario basada en web para el Proyecto Jupyter. Si has estado usando Jupyter Notebooks, es recomendable que pruebes JupyterLab; te provee de un ambiente interactivo para datos y código en Jupyter Notebooks. Vemos esto como una evolución de Jupyter Notebook: entrega una mejor experiencia al extender las habilidades originales que permiten código, visualización y documentación coexistiendo en un solo lugar.

Marquez
Evaluar

Marquez es un proyecto de código abierto relativamente nuevo para recolectar y servir información de metadatos sobre un ecosistema de datos. Marquez representa un modelo de datos simple para capturar metadatos como el linaje, las tareas de procesamiento ascendentes y descendentes de los datos y su estado de ejecución, y un conjunto flexible de etiquetas para capturar los atributos de los conjuntos de datos. Provee un API RESTful simple para gestionar los metadatos, que simplifica la integración de Marquez con otras herramientas dentro del ecosistema de datos.

Hemos usado Marquez como punto de partida y lo hemos extendido para que se adapte a nuestras necesidades, como aplicar políticas de seguridad, y hemos hecho cambios a su lenguaje de dominio. Si estás buscando una herramienta pequeña y simple para comenzar con el almacenamiento y la visualización de tus tareas de procesamiento de datos y conjuntos de datos, Marquez está bien para empezar.

Matomo
Evaluar

Matomo (antes conocido como Piwik) es una plataforma de código abierto para analíticas web que proporciona control total sobre los datos. Puedes usar Matomo en modo auto-alojado y proteger los datos de analíticas web de los proveedores externos. Matomo también facilita la integración de estos datos con tu plataforma de datos corporativa y te permite construir modelos de uso ajustados a tus necesidades.

MeiliSearch
Evaluar

MeiliSearch es un motor de búsqueda de texto rápido, de fácil uso y despliegue. A lo largo de los años Elasticsearch se ha vuelto la elección popular para búsquedas de texto escalables. No obstante, si no posees un volumen de datos que justifique una solución distribuida pero aún deseas proveer un motor de búsqueda rápido y tolerante a errores de tipografía, recomendamos evaluar MeiliSearch.

Stratos
Evaluar

Ultraleap (anteriormente Leap Motion) ha sido un líder en el espacio de XR desde hace un tiempo, creando hardware de seguimiento de manos extraordinario, que permite a las manos del usuario dar el salto a la realidad virtual. Stratos es la plataforma háptica, de sensores y de software subyacente a Ultraleap, pudiendo ser usada para dirigir ultrasonidos para crear respuestas hápticas en el aire. Un posible caso de uso sería responder a los gestos manuales del conductor para cambiar el aire acondicionado del coche y proporcionar respuesta háptica como parte del interfaz. Estamos emocionados con ver el desarrollo de esta tecnología y cómo va a ser utilizada por gente técnica creativa para incorporarla a sus casos de uso.

Trillian
Evaluar

Trillian es un almacén de datos centralizado criptográficamente verificable. Para entornos descentralizados y sin confianza podemos usar registros de transacciones distribuidos basados en *blockchain*. Sin embargo, para entornos empresariales, donde la gran cantidad de CPU utilizada por los protocolos de consenso resulta en costos injustificados, recomendamos probar Trillian.

Node en exceso
Resistir

Las tecnologías, especialmente las ampliamente populares, tienen tendencia a ser sobre-utilizadas. Lo que estamos viendo en este momento es un uso excesivo de Node , una tendencia a usar Node.js indiscriminadamente o por razones equivocadas. Entre ellas, en nuestra opinión destacan dos. La primera, escuchamos con frecuencia que se debe usar Node para que todo el código pueda ser escrito en el mismo lenguaje de programación. Nuestra visión sigue siendo que la programación políglota es una mejor aproximación, y esto funciona en ambos sentidos. La segunda, en ocasiones escuchamos a equipos citar el rendimiento como razón para elegir Node.js. Aunque hay infinidad de pruebas comparativas más o menos razonables, esta percepción radica en la historia. Cuando Node.js se hizo popular, fue el principal framework en adoptar el modelo de programación no bloqueante y esto le permitió ser muy eficiente en tareas con alta carga de E/S (ya lo mencionamos cuando escribimos sobre Node.js en 2012), pero puesto que ahora los frameworks con capacidades no bloqueantes — algunos con modernas y elegantes APIs — existen en otras plataformas, el rendimiento ya no es una razón para elegir Node.js

Plataformas

MeiliSearch es un motor de búsqueda de texto rápido, de fácil uso y despliegue. Es idea si no posees un volumen de datos que justifique una solución distribuida pero aún deseas proveer un motor de búsqueda rápido y tolerante a errores de tipografía.

(MeiliSearch)

Stratos es la plataforma háptica, de sensores y de software que enciende XR subyacente a Ultraleap (anteriormente Leap Motion).

(Stratos)

RADAR TECNOLÓGICO Vol. 22

Herramientas

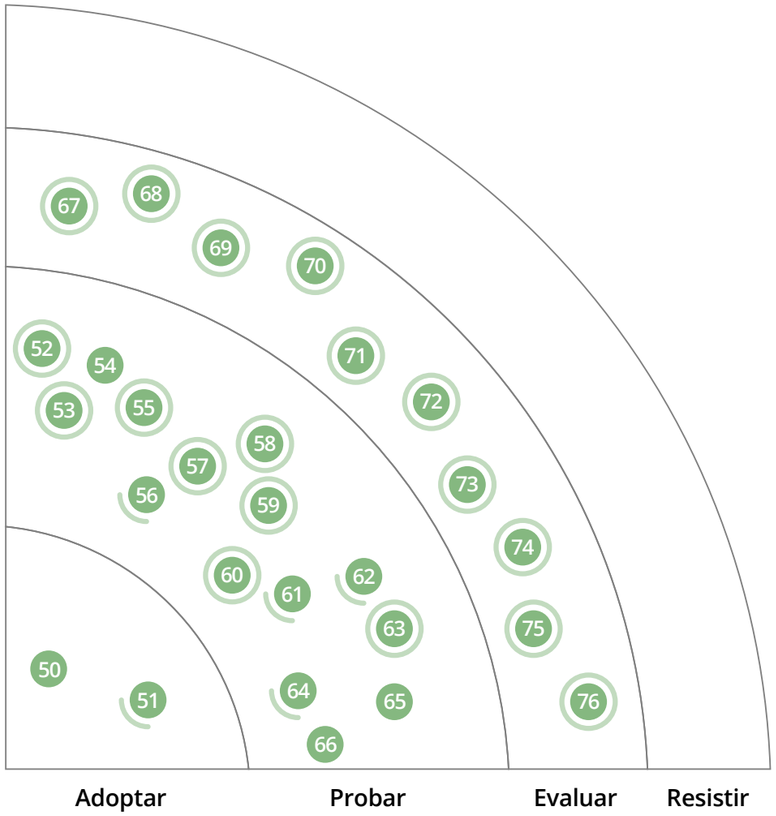
Herramientas

Cypress
Adoptar

Cypress continúa siendo una de las herramientas favoritas de nuestros equipos cuando los desarrolladores implementan y mantienen pruebas end-to-end, como parte de una pirámide de pruebas saludable. Decidimos mencionarlo nuevamente en este Radar porque las últimas versiones de Cypress han añadido soporte para Firefox e insistimos en realizar pruebas en varios navegadores ya que el dominante uso de Chrome y de navegadores basados en Chromium ha llevado a la preocupante tendencia de que algunos equipos solo ejecuten sus pruebas en Chrome, lo que puede llevar a desagradables sorpresas.

Figma
Adoptar

Figma ha demostrado ser una herramienta confiable para el diseño colaborativo, no solo para diseñadores, sino también para equipos multidisciplinarios. Permite a las personas desarrolladora y a otros roles ver y comentar los diseños a través de una interfaz web sin necesidad de instalar ninguna aplicación. En comparación con sus competidores (por ejemplo, Invision o Sketch), que necesitan de herramientas adicionales para versionamiento, colaboración y el intercambio de diseños, Figma provee todas estas características en una sola herramienta, permitiendo a nuestros equipos descubrir nuevas ideas juntos de forma sencilla. Nuestros equipos encuentran que Figma es también muy útil para actividades de diseño distribuido y remoto. Además de sus capacidades para el diseño en tiempo real y para la colaboración, Figma ofrece un API que ayuda a mejorar el proceso de DesignOps.



Dojo
Probar

Hace unos años, Docker, y los contenedores en general, cambiaron radicalmente la manera en que empaquetamos, desplegamos y ejecutamos nuestras aplicaciones. A pesar de esta mejora en producción, las personas desarrolladoras todavía gastan mucho tiempo configurando los ambientes de desarrollo y a menudo suceden problemas del tipo “pero funciona en mi máquina”. El objetivo de Dojo es arreglar esto creando ambientes de desarrollo estándar, versionados y publicados como imágenes de Docker. Varios de nuestros equipos ya lo utilizan para racionalizar y simplificar el desarrollo, la ejecución de pruebas y la compilación del código desde el ambiente local y a través de las líneas de producción.

Adoptar
50. Cypress
51. Figma

Probar
52. Dojo
53. DVC
54. Herramientas de seguimiento de experimentos para aprendizaje automático
55. Goss
56. Jaeger
57. k9s
58. kind
59. mkcert
60. MURAL
61. Open Policy Agent (OPA)
62. Optimal Workshop
63. Phrase
64. ScoutSuite
65. Herramientas de pruebas de regresión visual
66. Visual Studio Live Share

Evaluar
67. Apache Superset
68. AsyncAPI
69. ConfigCat
70. Gitpod
71. Gloo
72. Lens
73. Manifold
74. Sizzy
75. Snowpack
76. tfsec

Resistir

Herramientas

Jaeger es un sistema de rastreo distribuido de código abierto. hemos usado con éxito con Istio y Envoy en Kubernetes y nos gusta su UI.

(Jaeger)

k9s una herramienta que provee una interfaz interactiva para básicamente todo lo que kubectl puede hacer. Y para empezar, no es una aplicación web, sino que se ejecuta dentro de una ventana .

(k9s)

DVC
Probar

En el 2018 mencionamos a DVC junto con el versionamiento de datos para analíticas reproducibles. Desde entonces se ha convertido en la herramienta favorita para gestionar experimentos en proyectos de aprendizaje automático (*machine learning, ML*). Ya que se encuentra basado en Git, DVC es un ambiente familiar para las personas desarrolladoras de software que llevan sus costumbres de ingeniería a la práctica de ML. Puesto que versiona el código que procesa datos junto con los mismos datos y monitorea los estados en un *pipeline*, ayuda a ordenar las actividades de modelado sin interrumpir el flujo de trabajo de los analistas.

Herramientas de seguimiento de experimentos para aprendizaje automático
Probar

El trabajo diario del aprendizaje automático a menudo se reduce a una serie de experimentos para seleccionar un enfoque de modelamiento y de la topología de la red, datos para el entrenamiento y la optimización o ajuste de dicho modelo. Los científicos de datos deben usar la experiencia y la intuición para hipotetizar cambios, y luego medir el impacto de los mismos en el rendimiento general del modelo. A medida que esta práctica ha madurado, nuestros equipos han visto la creciente necesidad de adoptar herramientas para el seguimiento de experimentos de aprendizaje automático. Estas herramientas ayudan a los investigadores a mantener una continuidad de los experimentos y trabajarlos metódicamente. A pesar de que no ha surgido una clara herramienta ganadora, existen alternativas como MLflow y plataformas como Comet o Neptune que han introducido rigor y repetibilidad en todo el flujo de trabajo de aprendizaje automático.

Goss
Probar

Hemos mencionado a Goss, una herramienta para pruebas de aprovisionamiento, en radares pasados, por ejemplo, cuando describimos la técnica TDD'ing contenedores. Aunque Goss no es siempre la alternativa a Serverspec, simplemente porque no ofrece la misma cantidad de funcionalidades, podrías querer considerarla, cuando sus funcionalidades satisfagan tus necesidades, especialmente porque viene como un artefacto binario pequeño y auto contenido (en lugar de requerir un entorno Ruby). Un anti-patrón común al usar herramientas como Goss es la doble modificaciones de código, donde cada cambio en los archivos de la infraestructura como código actual requiere un correspondiente cambio en las validaciones de las pruebas. Tales pruebas son pesadas de mantener y debido a la correspondencia entre el código y las pruebas, la mayoría de las fallas ocurren cuando un ingeniero actualiza un lado y olvida el otro. Y estas pruebas raramente capturan problemas genuinos.

Jaeger
Probar

Jaeger es un sistema de rastreo distribuido de código abierto. Al igual que Zipkin, ha sido inspirado por el documento de Google Dapper y cumple con OpenTelemetry. Nosotros hemos usado con éxito Jaeger con Istio y Envoy en Kubernetes y nos gusta su interfaz de usuario. Jaeger expone métricas de rastreo en el formato Prometheus para que puedan estar disponibles para otras herramientas. Sin embargo, una nueva generación de herramientas, como Honeycomb, integran rastreo y métricas en un mismo flujo de observabilidad para tener un análisis agregado más simple. Jaeger se unió a CNCF en 2017 y recientemente ha sido elevado al nivel más alto de madurez de la CNCF, lo que indica su despliegue extendido en sistemas en producción.

k9s
Probar

Seguimos siendo fervientes partidarios de la infraestructura cómo código y creemos que una solución de monitoreo robusta es un prerrequisito para la operación de aplicaciones distribuidas. En ocasiones, una herramienta interactiva como la consola web de AWS puede ser una adición muy útil. Nos permite explorar todo tipo de recursos en una forma ad-hoc, sin tener que recordar oscuros comandos. Sin embargo utilizar una herramienta interactiva para hacer cambios manuales sobre la marcha, es todavía una práctica cuestionada. Para Kubernetes, tenemos ahora k9s, una herramienta que provee una interfaz interactiva para básicamente todo lo que kubectl puede hacer. Y para empezar, no es una aplicación web, sino que se ejecuta dentro de una ventana de la terminal, evocando, para algunas personas, buenos recuerdos de Midnight Commander.

kind
Probar

kind es una herramienta para la ejecución local de clusters de Kubernetes usando nodos de contenedores Docker. Con la

integración de kubetest, kind hace fácil la realización de pruebas de extremo a extremo (*end-to-end*) sobre Kubernetes. Hemos utilizado kind para crear clusters de Kubernetes efímeros para probar recursos de Kubernetes como Operadores y Definiciones de Recursos Personalizadas (CRDs) en nuestros pipelines de integración continua.

mkcert
Probar

mkcert es una buena herramienta para crear certificados de desarrollo confiables en el ambiente local. El uso de certificados otorgados por autoridades de certificación reales (*Certificate Authorities,* CAs) para el desarrollo local puede ser difícil e incluso imposible (para servidores tales como example.test, localhost o 127.0.0.1). En dichas situaciones, los certificados auto-firmados pueden ser la única opción. mkcert permite generar certificados auto-firmados e instala el CA local en el almacén de certificados raíz del sistema. Para cualquier otro ambiente diferente a desarrollo y pruebas locales, definitivamente se recomienda el uso de certificados de CAs reales para evitar problemas de confianza.

MURAL
Probar

MURAL se describe a sí misma como "un espacio de trabajo digital para la colaboración visual" y permite a los equipos interactuar en un espacio de trabajo compartido basado en la metáfora del tablero (pizarra) y notas adhesivas. Entre sus características se incluyen votaciones, comentarios, notas y la capacidad de "seguir" a quien está presentando. Nos gusta particularmente la capacidad de utilizar plantillas, que permite al facilitador diseñar y reutilizar sesiones guiadas con un equipo. Todos los paquetes de colaboración más importantes tienen una herramienta en este espacio (por ejemplo, Google Jamboard y Microsoft Whiteboard) y vale la pena probarlas, pero hemos comprobado que MURAL es veloz, efectiva y flexible.

Open Policy Agent (OPA)
Probar

Open Policy Agent (OPA) se ha convertido rápidamente en un componente aconsejable de muchas soluciones distribuidas nativas en la nube que construimos para nuestros clientes. OPA brinda un marco de trabajo uniforme y un lenguaje para declarar, imponer, y controlar políticas para varios componentes en una solución nativa en la nube. Es un buen ejemplo de una herramienta que implementa políticas de seguridad como código. Hemos tenido una experiencia fluida usando OPA en múltiples escenarios, incluyendo el despliegue de recursos a clusters de K8s, imponiendo control de acceso a través de servicios en una malla de servicios y controles de seguridad minuciosos como código para acceder a recursos de aplicación. Una oferta comercial reciente, Servicio de Autorización Declarativa de Styra (*Declarative Authorization Service* - DAS), facilita la adopción de OPA para compañías mediante la inclusión de una herramienta de gestión, o plano de control, a OPA para K8s con una librería de políticas pre-construida, análisis de impacto de las políticas y capacidad para monitoreo de logs. Anhelamos la madurez y extensión de OPA más allá de servicios operacionales a soluciones centradas en (grandes) datos.

Optimal Workshop
Probar

La investigación de experiencia de usuario (UX) requiere de la recopilación de datos y el análisis de los mismos para tomar mejores decisiones sobre los productos que se deben construir. Nuestros equipos encuentran útil a la herramienta Optimal Workshop porque facilita la validación de prototipos y la configuración de pruebas para la captura de datos y así tomar buenas decisiones. Características como el primer click, clasificación de contenidos o un mapa de calor de la interacción del usuario ayudan a validar los prototipos, para mejorar la navegación y visualización del sitio web. Es una herramienta ideal para equipos distribuidos, ya que permite conducir la investigación de forma remota.

Phrase
Probar

Como se mencionó en nuestro resumen sobre Crowdin, ahora existe la opción de usar plataformas para gestionar la traducción de un producto a varios idiomas en lugar de enviar por correo electrónico grandes hojas de cálculo. Hemos recibido de nuestros equipos experiencias favorables con Phrase, recalcando en que es fácil de usar para todos los grupos de usuarios. Para quienes traducen, presenta una interfaz de usuario sencilla que corre en el navegador. Las personas desarrolladoras pueden acceder a Phrase localmente y desde el pipeline de compilación. Una característica que merece la pena destacar es la capacidad de versionar las traducciones a través de etiquetas, lo que hace posible comparar el resultado de diferentes traducciones dentro del producto real.

Herramientas

OPA brinda un marco de trabajo uniforme y un lenguaje para declarar, imponer, y controlar políticas para varios componentes en una solución nativa en la nube.

(Open Policy Agent (OPA))

Nuestros equipos encuentran útil a la herramienta Optimal Workshop porque facilita la validación de prototipos y la configuración de pruebas para la captura de datos y así tomar buenas decisiones.

(Optimal Workshop)

Herramientas

AsyncAPI es una iniciativa de código abierto para crear un evento necesario y asíncrono con herramientas API de estandarización y desarrollo.

(AsyncAPI)

ConfigCat soporta feature toggles simples, segmentación de usuarios, y pruebas A/B y tiene un generoso plan gratuito para casos de uso de bajo volumen o para aquellos que están recién empezando

(ConfigCat)

ScoutSuite
Probar

ScoutSuite es una herramienta ampliada y actualizada basada en Scout2 (presentado en el Radar en 2018) que proporciona una evaluación de la postura de seguridad en AWS, Azure, GCP y otros proveedores de la nube. Funciona agregando automáticamente datos de configuración para un ambiente y aplicando reglas para auditar el mismo. Hemos encontrado esto muy útil en diversos proyectos para realizar evaluaciones de seguridad en un punto en el tiempo.

Herramientas de pruebas de regresión visual
Probar

Desde que en 2014 mencionamos por primera vez a las herramientas de pruebas de regresión visual , el uso de la técnica se ha difundido y el conjunto de herramientas ha evolucionado. BackstopJS sigue siendo una excelente opción que está adquiriendo nuevas funcionalidades con regularidad, incluyendo el soporte para ser ejecutada dentro de contenedores Docker. Loki apareció en nuestro último Radar. Applitools, CrossBrowserTesting y Percy son soluciones SaaS. Otra notable mención es Resemble.js, una biblioteca que permite la detección de diferencias en imágenes. Aunque la mayoría de equipos la usan indirectamente como parte de BackstopJS, algunos la utilizan directamente para analizar y comparar imágenes de sitios web. En general, nuestra experiencia muestra que las herramientas de regresión visual son menos útiles en las fases iniciales del desarrollo, cuando la interfaz pasa por cambios significativos, pero definitivamente muestran su valor cuando el producto madura y la interfaz se estabiliza.

Visual Studio Live Share
Probar

Visual Studio Live Share es un paquete de extensiones para Visual Studio Code y Visual Studio. En un momento en el que los equipos buscan buenas opciones para la colaboración remota, queremos destacar esta excelente herramienta. Live Share provee una buena experiencia para trabajo en pares de forma remota, con bajas latencias, y requiere mucho menos ancho de banda que el enfoque de fuerza bruta de compartir el escritorio entero. Es de resaltar que las personas desarrolladoras pueden trabajar con sus configuraciones, extensiones y mapas de teclas preferidas durante una sesión de trabajo en pareja. Además de la colaboración en tiempo real para editar y depurar código, Live Share permite la realización de llamadas de voz y la compartición de terminales y servidores.

Apache Superset
Evaluar

Apache Superset ees una herramienta de Inteligencia de Negocio (BI, por sus siglas en inglés) para explorar y visualizar datos, excelente para trabajar con grandes configuraciones de *data lake* y *data warehouse*. Funciona, por ejemplo, con Presto, Amazon Athena y Amazon Redshift y además puede integrarse con soluciones empresariales de autenticación. Más aún, no se necesita ser alguien enfocado en la ingeniería de datos para utilizar Superset, la herramienta busca beneficiar a todos quienes realizan ingeniería en su trabajo diario de exploración de datos.

Cabe recalcar que Apache Superset está actualmente en período de incubación en la Apache Software Foundation (ASF), lo que significa que aún no es respaldada totalmente por la ASF.

AsyncAPI
Evaluar

Los estándares abiertos son uno de los pilares fundacionales para la construcción de sistemas distribuidos. Por ejemplo, la especificación OpenAPI (anteriormente Swagger), como un estándar de la industria para definir APIs RESTful, ha sido crucial para el éxito de arquitecturas distribuidas como los microservicios. Ha permitido la proliferación de herramientas que soportan la construcción, prueba y monitoreo de las APIs RESTful. Sin embargo, una estandarización similar ha seguido faltando para sistemas distribuidos que implementan APIs dirigidas por eventos.

AsyncAPI es una iniciativa de código abierto para crear la tan necesaria estandarización para APIs asíncronas y dirigidas por eventos, y para desarrollar las herramientas necesarias. La especificación de AsyncAPI, inspirada en la especificación de OpenAPI, describe y documenta las APIs dirigidas por eventos en un formato legible por máquina. Es agnóstica a los protocolos, por lo que puede utilizarse para APIs que trabajan sobre múltiples protocolos, incluyendo MQTT, WebSockets y Kafka. Estamos ansiosos de ver la evolución continua de AsyncAPI y la madurez de su ecosistema de herramientas.

ConfigCat
Evaluar

Si estás buscando un servicio que te permita implementar *feature toggles* dinámicos (teniendo en cuenta que un *feature toggle* simple también funciona bien), revisa ConfigCat. Lo podemos describir como "similar a LaunchDarkly pero más barato y algo menos sofisticado" y encontramos que ofrece gran parte de lo que necesitamos. ConfigCat soporta *feature toggles* simples, segmentación de usuarios, y pruebas A/B. Además, tiene un generoso plan gratuito para casos de uso de bajo volumen o para aquellos que están recién empezando.

Gitpod
Evaluar

La mayoría del software se puede compilar siguiendo un sencillo proceso de dos pasos: obtener una copia (*check out*) de un repositorio y después ejecutar un único script de construcción. Sin embargo, el proceso de configurar un entorno de programación completo puede ser incómodo. Gitpod soluciona este problema proporcionando entornos basados en la nube, listos para programar, para repositorios de Github o GitLab. Ofrece un IDE basado en Visual Studio Code que se ejecuta dentro del navegador web. Por defecto, estos entornos se lanzan en Google Cloud Platform, aunque también se pueden desplegar en servidores locales. Vemos el atractivo inmediato, especialmente para proyectos de software de código abierto en los que esta aproximación puede bajar la barrera de entrada para quienes contribuyen ocasionalmente. Sin embargo, queda por ver cuán viable será esta aproximación en entornos corporativos.

Gloo
Evaluar

Con la creciente adopción de Kubernetes y malla de servicios, los API gateways han experimentado una crisis existencial en los sistemas distribuidos cloud-native. Después de todo, muchas de sus capacidades (tales como control de tráfico, seguridad, enrutamiento y observabilidad) son ofrecidas ahora por el controlador de entrada del cluster y el gateway de la malla (*mesh gateway*). Gloo es una API gateway ligera que incorpora este cambio; utiliza Envoy como su tecnología de gateway, mientras provee valor añadido, como por ejemplo ofreciendo una vista coherente de las APIs a usuarios externos y aplicaciones. También provee una interfaz de administración para controlar los gateways de Envoy y puede correr e integrarse con multiples implementaciones de service mesh como Linkerd, Istio y AWS App Mesh. Aunque su implementacion de codigo abierto provee todas las capacidades basicas que se puede esperar de una API gateway, su edicion corporativa tiene un conjunto más maduro de controles de seguridad, como gestión de claves de API o integración con OPA. Gloo es una prometedora API gateway ligera que puede integrarse bien con el ecosistema de tecnologías y arquitecturas cloud-native,mientras evita la mala práctica de usar los API gateway para alojar lógica de negocio uniendo APIs para el usuario final.

Lens
Evaluar

Una de las fortalezas de Kubernetes es su flexibilidad y rango de posibilidades de configuración junto con los mecanismos de configuración programables, controlados por APIs, y la visibilidad y el control por línea de comandos usando archivos de manifiesto. Sin embargo, esa fortaleza puede ser también una debilidad: cuando los despliegues son complejos o cuando se administran varios *clusters*, puede ser difícil obtener una imagen clara del estado general solamente a través de argumentos de la línea de comando y archivos de manifiesto. Lens intenta resolver este problema con un ambiente integrado para observar el estado actual del *cluster* y sus cargas de trabajo, visualizando métricas y cambiando configuraciones a través de un editor de texto integrado. Más que una simple interfaz, Lens reúne las herramientas que un administrador ejecutaría desde una línea de comandos en una única interfaz web navegable. Esta herramienta es uno de los varios enfoques que intentan disminuir la complejidad de la gestión de Kubernetes. Todavía no hemos visto un claro ganador en este espacio, pero Lens logra un equilibrio interesante entre una interfaz gráfica de usuario y herramientas de línea de comandos.

Manifold
Evaluar

Manifold es un depurador visual independiente del modelo para aprendizaje automático. Las personas desarrolladoras de modelos de aprendizaje automático invierten gran cantidad de tiempo en iterar un modelo ya existente para mejorarlo en vez de crear uno nuevo. Al cambiar el enfoque del espacio del modelo al espacio de datos, Manifold complementa las métricas de rendimiento ya existentes con las características visuales del conjunto de datos que influyen en el rendimiento del modelo. Creemos que Manifold será una herramienta útil a evaluar dentro del ecosistema de aprendizaje automático.

Herramientas

Lens es uno de los varios enfoques que intentan disminuir la complejidad de la gestión de Kubernetes.

(Lens)

Herramientas

tfsec es una herramienta de análisis estático que ayuda a escanear plantillas de Terraform y encontrar problemas de seguridad.

(tfsec)

Sizzy
Evaluar

La creación de aplicaciones web para que se vean exactamente como se espera en una gran cantidad de dispositivos y tamaños de pantalla puede ser engorrosa. Sizzy es una solución SaaS que muestra muchas ventanas gráficas *(viewport*) en una sola ventana del navegador. La aplicación se presenta en todas las ventanas gráficas simultáneamente y las interacciones con la aplicación también se sincronizan a través de todas ellas. En nuestra experiencia, interactuar con una aplicación de esta manera puede facilitar la detección de posibles problemas de manera temprana, antes de que las herramientas de pruebas de regresión visual encuentren el problema en el *pipeline*. Sin embargo, debemos mencionar que algunas personas que probaron Sizzy durante un tiempo prefirieron, en general, trabajar con las herramientas de desarrollo proporcionadas por Chrome.

Snowpack
Evaluar

Snowpack es un nuevo e interesante jugador en el terreno de las herramientas de construcción para JavaScript. La mejora clave por sobre el resto de soluciones es que Snowpack permite construir aplicaciones con tecnologías modernas como React.js, Vue.js y Angular sin la necesidad de un empaquetador (*bundler*). Retirar el paso de empaquetado, reduce de forma sustancial el ciclo de retroalimentación durante el desarrollo, porque los cambios se muestran en el navegador casi inmediatamente. Para que esto funcione, Snowpack transforma las dependencias de `node_modules` en archivos JavaScript individuales dentro de un nuevo directorio `web_modules`, desde el cual se pueden importar como módulos de ECMAScript (ESM). Existen soluciones alternas para IE11 y otros navegadores que no soportan ESM. Desafortunadamente, ya que en la actualidad ningún navegador puede importar CSS desde JavaScript, el uso de módulos CSS no es sencillo.

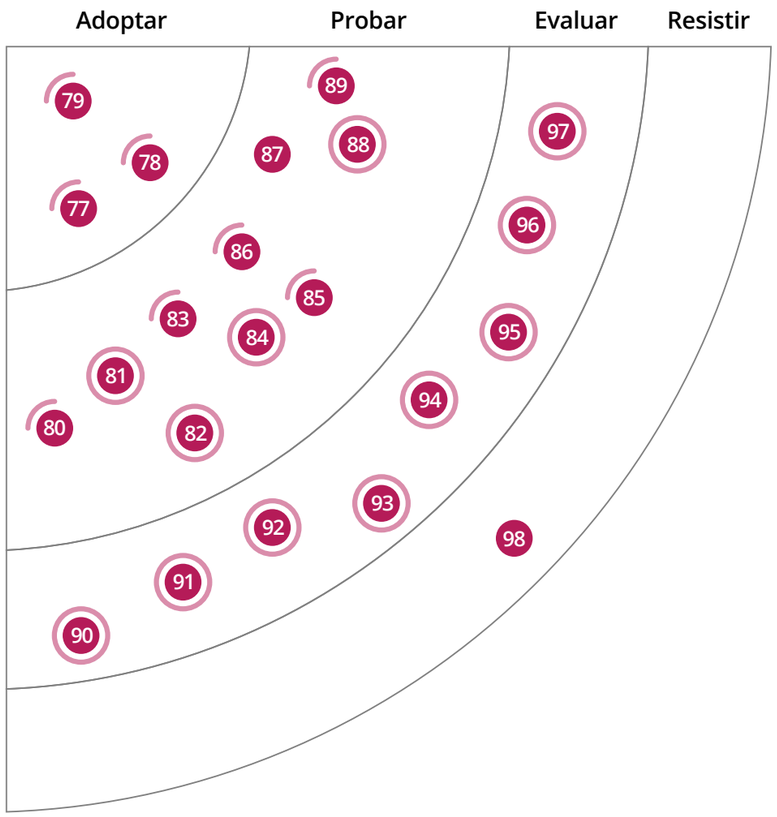
tfsec
Evaluar

La seguridad es una preocupación de todos y capturar riesgos temprano siempre es mejor que enfrentar problemas más adelante. En el contexto de la infraestructura como código, donde Terraform es una elección obvia para administrar ambientes en la nube, ahora tenemos a tfsec, que es una herramienta de análisis estático que ayuda a escanear plantillas de Terraform y encontrar cualquier potencial problema de seguridad. Viene con reglas predefinidas para distintos proveedores de la nube incluyendo a AWS y Azure. Siempre nos gustan las herramientas que ayudan a mitigar riesgos de seguridad, y tfsec no solo se destaca en la identificación de riesgos de seguridad, sino que es también fácil de instalar y de usar.

RADAR TECNOLÓGICO Vol. 22

Languajes & Frameworks

Languajes & Frameworks



React Hooks
Adoptar

React Hooks ha introducido un nuevo enfoque para gestionar la lógica con estado; dado que los componentes en React siempre han sido más cercanos a las funciones que a las clases, Hooks ha tomado esto en cuenta y ha llevado el estado a las funciones, en lugar de llevar funciones como métodos al estado con clases. En base a nuestra experiencia, Hooks mejora la reutilización de funcionalidades entre componentes y la legibilidad del código. Dadas las mejoras en la capacidad de realización de pruebas de Hooks, por el uso de React Test Renderer y React Testing Library, y el crecimiento del apoyo de la comunidad, lo consideramos nuestro enfoque preferido.

Biblioteca de Pruebas React
Adoptar

El mundo de JavaScript evoluciona rápidamente, y a medida que adquirimos más experiencia en el uso de un framework, nuestras recomendaciones van variando. La Biblioteca de Pruebas de React es un buen ejemplo de un framework que con un uso más profundo, ha ido eclipsando otras alternativas para convertirse en la opción por defecto a la hora de probar frontends basados en React. A nuestros equipos les gusta el hecho de que los tests escritos con este framework son menos frágiles que con frameworks alternativos como Enzyme, porque se les anima a probar las relaciones de los componentes individualmente en lugar de probar todos los detalles de implementación. Esta forma de pensar traída por Testing Library en la cual la biblioteca de pruebas de react es una parte, vale resaltar que esta familia de librerías también incluye otras como por ejemplo: Angular y Vue.js.

Vue.js
Adoptar

Vue.js ha llegado a ser uno de los marcos de trabajo JavaScipt para *frontend* preferidos, confiables y que se ha utilizado con éxito entre nuestra comunidad. Aunque hay otras alternativas con buena adopción como React.js, la simplicidad del diseño de la API de Vue.js, su clara segregación de directivas y componentes (un archivo para cada componente de código) y su gestión simplificada de estados lo han convertido en una opción atractiva frente a otras.

Adoptar
77. React Hooks
78. Biblioteca de Pruebas React
79. Vue.js

Probar
80. CSS-in-JS
81. Exposed
82. GraphQL Inspector
83. Karate
84. Koin
85. NestJS
86. PyTorch
87. Rust
88. Sarama
89. SwiftUI

Evaluar
90. Clinic.js Bubbleprof
91. Deequ
92. ERNIE
93. MediaPipe
94. Tailwind CSS
95. Tamer
96. Wire
97. XState

Resistir
98. Enzyme

Languajes & Frameworks

Koin es un marco de trabajo de Kotlin que gestiona uno de los problemas más comunes en el desarrollo de software, para la inyección de dependencias.

(Koin)

Es un marco de trabajo basado en TypeScript que hace que el desarrollo de aplicaciones Node.js sea más seguro y menos susceptible a errores.

(NestJS)

CSS-in-JS
Probar

Desde que mencionamos por primera vez *CSS-en-JS* como una técnica emergente en el 2007, se ha vuelto mucho más popular, y es una tendencia que también vemos en nuestro trabajo. Con una sólida experiencia de producción en nuestro historial, ahora podemos recomendar CSS-en-JS como una técnica a probar. Un buen punto de partida es el marco de trabajo denominado componentes de estilo que mencionamos en nuestro último Radar. A pesar de todos los aspectos positivos que tiene esta técnica, puede haber una desventaja al utilizar CSS-en-JS: el cálculo de los estilos en tiempo de ejecución puede ocasionar retardos notables al usar la aplicación. Con Linaria ahora vemos una nueva clase de marcos de trabajo que fueron creados con este problema en mente. Linaria utiliza una serie de técnicas para mover la mayor parte de la carga sobre el rendimiento al tiempo de construcción. Desafortunadamente, esto trae consigo algunas dificultades, especialmente la falta de soporte para los estilos dinámicos en IE11.

Exposed
Probar

Con el uso extendido de Kotlin, nuestros equipos de desarrollo han ganado experiencia con más marcos de trabajo y librerías diseñadas específicamente para este lenguaje, en lugar de usar marcos de trabajo y librerías de Java con Kotlin. Aún cuando ha estado presente por un tiempo, Exposed ha llamado nuestra atención por ser un mapeador objeto-relacional (ORM) ligero. Exposed tiene dos variantes para el acceso a las bases de datos: un DSL interno con tipado seguro (*typesafe*) basado en SQL y una implementación del patrón de objetos de acceso a datos (DAO). Tiene las características que se esperan de un ORM maduro como el manejo de referencias de muchos a muchos, carga temprana de datos, y soporte para juntarlas (*joins*) entre entidades. Nos gusta también que la implementación trabaja sin clases intermedias (*proxies*) y no depende del uso de reflexión, lo que resulta beneficioso para el rendimiento.

GraphQL Inspector
Probar

GraphQL Inspector te permite comparar cambios entre 2 esquemas de GraphQL. En el pasado hemos advertido sobre el uso de GraphQL, y ahora nos alegra ver algunas mejoras en las herramientas que lo rodean. La mayoría de nuestros equipos continúa usando GraphQL para la agregación de recursos en el servidor, y ahora al introducir el Inspector de GraphQL en sus pipelines de integración continua, somos capaces de identificar potenciales cambios de última hora en los esquemas de GraphQL.

Karate
Probar

Dada nuestra experiencia en que las pruebas son las únicas especificaciones de API que realmente importan, siempre estamos atentos a nuevas herramientas que ayuden al desarrollo guiado por pruebas. Karate es un framework para pruebas de interfaz de programación de aplicaciones (API) con la característica singular de que los tests se escriben directamente en un lenguaje de sintaxis basado en Gherkin, sin depender de un lenguaje de programación de propósito general para implementar su comportamiento. Karate utiliza un lenguaje específico de dominio para describir pruebas de API basadas en HTTP. A nuestros equipos les agrada la especificación legible que se consigue con esta herramienta, recomiendan conservar las pruebas realizadas con Karate en los niveles superiores de la pirámide de pruebas y no sobreutilizarlo haciendo assertions demasiado detalladas.

Koin
Probar

Con el mayor uso de Kotlin para el desarrollo móvil y de aplicaciones de servidor, el ecosistema asociado continúa evolucionando. Koin es un marco de trabajo de Kotlin que gestiona uno de los problemas más comunes en el desarrollo de software: la inyección de dependencias. Aunque se puede elegir entre una variedad de marcos de trabajo para la inyección de dependencias para Kotlin, nuestros equipos han llegado a preferir la simplicidad de Koin. Koin evita el uso de anotaciones e inyecta dependencias a través de constructores o imitando la inicialización diferida (*lazy initialization*) para que los objetos sean inyectados solo cuando se les necesite. Esto contrasta con el marco de trabajo estáticamente compilado Dagger para Android. A nuestras personas desarrolladoras les gusta la naturaleza ligera de este marco de trabajo y las capacidades incorporadas para la escritura de pruebas.

NestJS
Probar

El crecimiento de la popularidad de Node.js y las tendencias como Node en exceso, han llevado al uso de Node.js para el desarrollo de aplicaciones de negocio. Con frecuencia vemos problemas de escalabilidad y mantenibilidad en aplicaciones grandes basadas en JavaScript. NestJS es un marco de trabajo basado en TypeScript que hace que el desarrollo de aplicaciones Node.js sea más seguro y menos susceptible a errores. NestJS es prescriptivo y trae a

la mesa, de manera predeterminada, la aplicación de los principios SOLID y una arquitectura inspirada en Angular. NestJS es uno de los marcos de trabajo que nuestros equipos utilizan regularmente al construir microservicios con Node.js; permite la creación de aplicaciones que se puedan probar, sean escalables, con bajo acoplamiento y fácilmente mantenibles.

PyTorch
Probar

Nuestros equipos han venido usando, y les sigue gustando, el framework de machine learning PyTorch, y muchos equipos prefieren PyTorch sobre TensorFlow. PyTorch expone el funcionamiento interno de ML que TensorFlow oculta, haciendo que sea más fácil depurarlo, y contiene componentes con los que la mayoría de las personas programadoras están familiarizadas, como bucles y acciones. Los lanzamientos recientes han mejorado el desempeño de PyTorch, y nosotros lo hemos estado usando con éxito en proyectos en producción.

Rust
Probar

Rust continúa ganando popularidad. Hemos tenido intensas discusiones sobre qué lenguaje es mejor, entre Rust, C++ o Go, sin un claro vencedor. Sin embargo, desde que lo mencionamos en la anterior edición del Radar, nos complace ver que Rust ha mejorado significativamente con la adición de nuevas APIs y su estabilización, incluyendo el soporte asíncrono avanzado. Además, Rust ha inspirado el diseño de nuevos lenguajes. Por ejemplo, el lenguaje Move de Libra imita la manera en la que Rust gestiona la memoria para administrar recursos, asegurando que dichos recursos digitales no pueden ser copiados o descartados implícitamente.

Sarama
Probar

Sarama es una biblioteca cliente escrita en Go para Apache Kafka. Si estás desarrollando APIs en Go, encontrarás que Sarama es bastante sencilla de configurar y manejar, ya que no depende de ninguna biblioteca nativa. Sarama tiene dos tipos de APIs: un API de alto nivel para la producción y consumo fácil de mensajes, y un API de bajo nivel para controlar la transmisión de los datos.

SwiftUI
Probar

Apple ha dado un gran paso adelante con su nuevo framework SwiftUI para implementar interfaces de usuario en las plataformas macOS e iOS. Nos gusta que SwiftUI va más allá de la relación un tanto rudimentaria entre Interface Builder y Xcode y adopta un enfoque coherente, declarativo y centrado en el código. Ahora se puede ver el código y la interfaz resultante lado a lado en Xcode 11, mejorando mucho la experiencia de desarrollo. El framework SwiftUI se inspira en el mundo de React.js que ha dominado el desarrollo web en los últimos años. Los valores inmutables en los modelos de vista y un mecanismo de actualización asíncrono crean un modelo unificado de programación reactiva. Esto se convierte, para los equipos de desarrollo, en una alternativa completamente nativa, similar a marcos de trabajo reactivos como React Native o Flutter. SwiftUI definitivamente representa el futuro para el desarrollo de interfaces de usuario para Apple y a pesar de ser nuevo, ha mostrado sus beneficios. Hemos tenido una gran experiencia con este marco de trabajo y su suave curva de aprendizaje. Es importante señalar que se debe conocer muy bien el caso de uso antes de empezar a usar SwiftUI, ya que no es compatible con iOS12 o versiones anteriores.

Clinic.js Bubbleprof
Evaluar

Con el objetivo de mejorar el rendimiento de nuestro código, las herramientas de perfilado son útiles para identificar cuellos de botella o retardos en el código que pueden ser difíciles de identificar, especialmente en operaciones asíncronas. Clinic.js Bubbleprof representa visualmente las operaciones asíncronas dentro de los procesos de Node.js, dibujando un mapa de los retardos en el flujo de la aplicación. Nos gusta esta herramienta porque ayuda a los equipos de desarrollo a identificar y establecer prioridades sobre qué mejorar en el código.

Deequ
Evaluar

Aún existen algunas brechas de herramientas cuando se aplican buenas prácticas de ingeniería de software en la ingeniería de datos. Uno de nuestros equipos, al intentar automatizar las revisiones de calidad de datos entre las distintas etapas de un pipeline de datos, se sorprendió cuando apenas encontraron herramientas en este espacio. Finalmente, se decidieron por usar Deequ, una biblioteca para escribir pruebas para conjuntos de datos que se asemejan a las pruebas unitarias. Deequ está construido sobre Apache Spark, y aunque es publicado por AWS Labs puede ser usado en entornos diferentes a AWS.

Languajes & Frameworks

Deequ es una biblioteca para escribir pruebas para conjuntos de datos que ayuda con las revisiones de calidad entre las distintas etapas de un pipeline de datos.

(Deequ)

ERNIE proporciona modelos de lenguaje no supervisados pre-entrenados, que pueden ser ajustados agregando capas de salida para crear modelos en estado del arte para una variedad de tareas NLP.

(ERNIE)

Languajes & Frameworks

Tailwind CSS propone un enfoque interesante al proporcionar clases utilitarias de CSS de bajo nivel para crear bloques de construcción sin estilos predefinidos y apuntando a una fácil personalización.

(Tailwind CSS)

Wire es una herramienta de inyección de dependencias en tiempo de compilación que genera código y conecta los componentes.

(Wire)

ERNIE
Evaluar

En la anterior edición del Radar presentamos BERT - un hito en el panorama de Procesamiento de Lenguaje Natural. El año pasado, Baidu publicó ERNIE que mejoró BERT en siete tareas de comprensión del lenguaje GLUE y en las 9 tareas NLP para el lenguaje chino. ERNIE, como BERT, ofrece modelos de lenguaje no supervisados pre-entrenados, que pueden ser ajustados agregando capas de salida para crear modelos en estado del arte para una variedad de tareas NLP. ERNIE se diferencia de los métodos pre-entrenados tradicionales en que es un framework de pre-entrenamiento continuo. En lugar de entrenar con un pequeño número de objetivos de pre-entrenamiento, puede introducir constantemente una gran variedad de tareas de pre-entrenamiento para ayudar al modelo a aprender representaciones del lenguaje de manera eficiente. Estamos muy emocionados sobre los avances en NLP y estamos deseando experimentar con ERNIE en nuestros proyectos.

MediaPipe
Evaluar

MediaPipe es un marco de trabajo para construir pipelines aplicados de aprendizaje automático multimodales (como de video, audio, datos de series de tiempo, etc.), y para diferentes plataformas (por ejemplo, Android, iOS, Web, y dispositivos perimetrales). MediaPipe provee múltiples características, incluyendo detección de rostros, seguimiento de manos, detección de gestos, y detección de objetos. Aunque MediaPipe se implementa principalmente en dispositivos móviles, se comenzó a mostrar en el navegador gracias a WebAssembly y XNNPack ML Inference Library. Estamos explorando MediaPipe para algunos casos de uso de realidad aumentada y nos gusta lo que hemos visto hasta ahora.

Tailwind CSS
Evaluar

Las herramientas y marcos de trabajo de CSS ofrecen componentes prediseñados para resultados rápidos; sin embargo, luego de un tiempo, pueden complicar la personalización. Tailwind CSS propone un enfoque interesante al proporcionar clases utilitarias de CSS de bajo nivel para crear bloques de construcción sin estilos predefinidos y apuntando a una fácil personalización. La amplitud de las utilidades de bajo nivel permiten evitar escribir cualquier clase o CSS por tu cuenta lo que lleva a una base de código más mantenible a largo plazo. Parece que Tailwind CSS ofrece el correcto balance entre reusabilidad y personalización para construir componentes visuales.

Tamer
Evaluar

Si lo que se requiere es la ingesta de datos hacia un tópico de Kafka, piensa en Tamer, que se autodenomina “un conector JDBC domesticado para el origen de datos para Kafka”. A pesar de ser un marco de trabajo relativamente nuevo, hemos notado que Tamer es más eficiente que el conector JDBC propio de Kafka, especialmente cuando se trata de grandes volúmenes de datos.

Wire
Evaluar

La comunidad de Golang ha tenido su cuota de escépticos en cuanto a la inyección de dependencias, debido en parte a la confusión del patrón con frameworks específicos; y a las personas desarrolladoras con experiencia en programación de sistemas generalmente no les gusta el impacto en tiempo de ejecución causado por la reflexión. Entonces apareció Wire, una herramienta de inyección de dependencias en tiempo de compilación que genera código y conecta los componentes. Wire no produce impactos adicionales en tiempo de ejecución y su gráfico de dependencias estático es fácil de comprender. Ya sea que escribas código manualmente o utilices algún marco de trabajo, recomendamos utilizar inyección de dependencias para propiciar diseños modulares y verificables.

XState
Evaluar

Ya habíamos destacado varias bibliotecas para el manejo de estado en ediciones anteriores del Radar, pero XState adopta un enfoque ligeramente distinto. Es un marco de trabajo simple para JavaScript y TypeScript que permite crear máquinas de estado finito y visualizarlas como mapas de estado. Se integra con los marcos de trabajo reactivos más populares de JavaScript, como (Vue.js, Ember.js, React.js y RxJS y está basado en el estándar para máquinas de estado finito de la W3C. Otra característica notable es la serialización de las definiciones de máquina. Algo que nos ha resultado útil al crear máquinas de estado finito en otros contextos (particularmente al escribir lógica para juegos) es la posibilidad de visualizar los estados y sus posibles transiciones; nos gusta que el visualizador de XState lo vuelve realmente fácil.

Enzyme
Resistir

No acostumbramos a mover herramientas obsoletas a Resistir en el Radar. Sin embargo, nuestros equipos creen firmemente que Enzyme ha sido ya reemplazado, en el campo de las pruebas unitarias de componentes de UI para React por React Testing Library . Los equipos que usan Enzyme han visto que esta herramienta se centra en probar las partes internas de los componentes, lo que genera pruebas frágiles y difíciles de mantener.

ThoughtWorks®

Somos una consultora de software global y una comunidad de individuos apasionados guiados por propósitos, 7000+ personas en 43 oficinas en 14 países. A lo largo de nuestros más de 25 años de historia, hemos ayudado a nuestros clientes a resolver problemas comerciales complejos donde la tecnología es el diferenciador. Cuando la única constante es el cambio, te preparamos para lo impredecible.

¿Quieres estár al tanto con toda las noticias e insights relacionados al Radar?

Síguenos en tu red social favorita
o conviértete en un suscriptor/a.

suscríbete ahora

ThoughtWorks®

thoughtworks.com/es/radar

#TWTechRadar